

EVALUATION COPY



Compute Express Link™ (CXL™)

Specification

August 2023

Revision 3.1

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

© 2019-2023 COMPUTE EXPRESS LINK CONSORTIUM, INC. ALL RIGHTS RESERVED.

This CXL Specification Revision 3.1 (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 at <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: “**© 2019-2023 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.

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: (i) 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

1.0	Introduction	48
1.1	Audience	48
1.2	Terminology/Acronyms	48
1.3	Reference Documents	59
1.4	Motivation and Overview	60
1.4.1	CXL	60
1.4.2	Flex Bus	63
1.5	Flex Bus Link Features	65
1.6	Flex Bus Layering Overview	65
1.7	Document Scope	66
2.0	CXL System Architecture	69
2.1	CXL Type 1 Device	70
2.2	CXL Type 2 Device	70
2.2.1	Back-Invalidate Snoop Coherence for HDM-DB	71
2.2.2	Bias-based Coherency Model for HDM-D Memory	71
2.2.2.1	Host Bias	72
2.2.2.2	Device Bias	72
2.2.2.3	Mode Management	73
2.3	CXL Type 3 Device	74
2.4	Multi Logical Device (MLD)	75
2.4.1	LD-ID for CXL.io and CXL.mem	75
2.4.1.1	LD-ID for CXL.mem	75
2.4.1.2	LD-ID for CXL.io	75
2.4.2	Pooled Memory Device Configuration Registers	76
2.4.3	Pooled Memory and Shared FAM	77
2.4.4	Coherency Models for Shared FAM	77
2.5	Multi-Headed Device	79
2.5.1	LD Management in MH-MLDs	80
2.6	CXL Device Scaling	80
2.7	CXL Fabric	81
2.8	Global FAM (G-FAM) Type 3 Device	81
2.9	Manageability Overview	81
3.0	CXL Transaction Layer	83
3.1	CXL.io	83
3.1.1	CXL.io Endpoint	84
3.1.2	CXL Power Management VDM Format	84
3.1.2.1	Credit and PM Initialization	88
3.1.3	CXL Error VDM Format	89
3.1.4	Optional PCIe Features Required for CXL	90
3.1.5	Error Propagation	90
3.1.6	Memory Type Indication on ATS	91
3.1.7	Deferrable Writes	91
3.1.8	PBR TLP Header (PTH)	91
3.1.8.1	Transmitter Rules Summary	92
3.1.8.2	Receiver Rules Summary	92
3.1.9	VendPrefixL0	94
3.1.10	CXL DevLoad (CDL) Field in UIO Completions	95
3.1.11	CXL Fabric-related VDMs	95
3.1.11.1	Host Management Transaction Flows of GFD	97
3.1.11.2	Downstream Proxy Command (DPCmd) VDM	99
3.1.11.3	Upstream Command Pull (UCPull) VDM	100

EVALUATION COPY

Contents

3.1.11.4	Downstream Command Request (DCReq, DCReq-Last, DCReq-Fail) VDMs	101
3.1.11.5	Upstream Command Response (UCRsp, UCRsp-Last, UCRsp-Fail) VDMs	102
3.1.11.6	GFD Async Message (GAM) VDM	102
3.1.11.7	Route Table Update (RTUpdate) VDM	103
3.1.11.8	Route Table Update Response (RTUpdateAck, RTUpdateNak) VDMs	103
3.2	CXL.cache	104
3.2.1	Overview	104
3.2.2	CXL.cache Channel Description	105
3.2.2.1	Channel Ordering	105
3.2.2.2	Channel Crediting	105
3.2.3	CXL.cache Wire Description	106
3.2.3.1	D2H Request	106
3.2.3.2	D2H Response	107
3.2.3.3	D2H Data	108
3.2.3.4	H2D Request	108
3.2.3.5	H2D Response	109
3.2.3.6	H2D Data	110
3.2.4	CXL.cache Transaction Description	110
3.2.4.1	Device-attached Memory Flows for HDM-D/HDM-DB	110
3.2.4.2	Device to Host Requests	111
3.2.4.3	Device to Host Response	123
3.2.4.4	Host to Device Requests	124
3.2.4.5	Host to Device Response	125
3.2.5	Cacheability Details and Request Restrictions	127
3.2.5.1	GO-M Responses	127
3.2.5.2	Device/Host Snoop-GO-Data Assumptions	127
3.2.5.3	Device/Host Snoop/WritePull Assumptions	127
3.2.5.4	Snoop Responses and Data Transfer on CXL.cache Evicts	128
3.2.5.5	Multiple Snoops to the Same Address	128
3.2.5.6	Multiple Reads to the Same Cacheline	128
3.2.5.7	Multiple Evicts to the Same Cacheline	128
3.2.5.8	Multiple Write Requests to the Same Cacheline	128
3.2.5.9	Multiple Read and Write Requests to the Same Cacheline	128
3.2.5.10	Normal Global Observation (GO)	129
3.2.5.11	Relaxed Global Observation (FastGO)	129
3.2.5.12	Evict to Device-attached Memory	129
3.2.5.13	Memory Type on CXL.cache	129
3.2.5.14	General Assumptions	129
3.2.5.15	Buried Cache State Rules	130
3.2.5.16	H2D Req Targeting Device-attached Memory	131
3.3	CXL.mem	132
3.3.1	Introduction	132
3.3.2	CXL.mem Channel Description	133
3.3.2.1	Direct P2P CXL.mem for Accelerators	134
3.3.2.2	Snoop Handling with Direct P2P CXL.mem	134
3.3.3	Back-Invalidate Snoop	135
3.3.4	QoS Telemetry for Memory	136
3.3.4.1	QoS Telemetry Overview	136
3.3.4.2	Reference Model for Host/Peer Support of QoS Telemetry	137
3.3.4.3	Memory Device Support for QoS Telemetry	138
3.3.5	M2S Request (Req)	148
3.3.6	M2S Request with Data (RwD)	152
3.3.6.1	Trailer Present for Rwd (256B Flit)	154
3.3.7	M2S Back-Invalidate Response (BIRsp)	154
3.3.8	S2M Back-Invalidate Snoop (BISnsp)	155

EVALUATION COPY

Contents

3.3.8.1	Rules for Block Back-Invalidate Snoops	156
3.3.9	S2M No Data Response (NDR)	157
3.3.10	S2M Data Response (DRS)	158
3.3.10.1	Trailer Present for DRS (256B Flit).....	159
3.3.11	Forward Progress and Ordering Rules	160
3.3.11.1	Buried Cache State Rules for HDM-D/HDM-DB.....	160
3.4	Transaction Ordering Summary	162
3.5	Transaction Flows to Device-attached Memory	165
3.5.1	Flows for Back-Invalidate Snoops on CXL.mem	165
3.5.1.1	Notes and Assumptions.....	165
3.5.1.2	BISnP Blocking Example	166
3.5.1.3	Conflict Handling.....	167
3.5.1.4	Block Back-Invalidate Snoops	169
3.5.2	Flows for Type 1 Devices and Type 2 Devices.....	171
3.5.2.1	Notes and Assumptions.....	171
3.5.2.2	Requests from Host.....	172
3.5.2.3	Requests from Device in Host and Device Bias	179
3.5.3	Type 2 Memory Flows and Type 3 Memory Flows.....	184
3.5.3.1	Speculative Memory Read	184
3.6	Flows to HDM-H in a Type 3 Device.....	185
4.0	CXL Link Layers	187
4.1	CXL.io Link Layer	187
4.2	CXL.cache and CXL.mem 68B Flit Mode Common Link Layer	188
4.2.1	Introduction	188
4.2.2	High-Level CXL.cachemem Flit Overview	190
4.2.3	Slot Format Definition.....	197
4.2.3.1	H2D and M2S Formats	198
4.2.3.2	D2H and S2M Formats	203
4.2.4	Link Layer Registers	208
4.2.5	68B Flit Packing Rules	208
4.2.6	Link Layer Control Flit.....	210
4.2.7	Link Layer Initialization.....	214
4.2.8	CXL.cachemem Link Layer Retry	215
4.2.8.1	LLR Variables.....	215
4.2.8.2	LLCRD Forcing	217
4.2.8.3	LLR Control Flits.....	218
4.2.8.4	RETRY Framing Sequences	219
4.2.8.5	LLR State Machines	220
4.2.8.6	Interaction with vLSM Retrain State.....	224
4.2.8.7	CXL.cachemem Flit CRC	224
4.2.9	Viral.....	226
4.3	CXL.cachemem Link Layer 256B Flit Mode	226
4.3.1	Introduction	226
4.3.2	Flit Overview	226
4.3.3	Slot Format Definition.....	231
4.3.3.1	Implicit Data Slot Decode	245
4.3.3.2	Trailer Decoder	246
4.3.4	256B Flit Packing Rules	247
4.3.5	Credit Return	249
4.3.6	Link Layer Control Messages.....	252
4.3.6.1	Link Layer Initialization	254
4.3.6.2	Viral Injection and Containment	254
4.3.6.3	Late Poison	255
4.3.6.4	Link Integrity and Data Encryption (IDE)	256
4.3.7	Credit Return Forcing	256
4.3.8	Latency Optimizations	256

EVALUATION COPY

Contents

4.3.8.1	Empty Flit	257
5.0	CXL ARB/MUX.....	258
5.1	vLSM States	259
5.1.1	Additional Rules for Local vLSM Transitions.....	262
5.1.2	Rules for vLSM State Transitions across Link.....	262
5.1.2.1	General Rules	262
5.1.2.2	Entry to Active Exchange Protocol	263
5.1.2.3	Status Synchronization Protocol	263
5.1.2.4	State Request ALMP	265
5.1.2.5	L0p Support	270
5.1.2.6	State Status ALMP.....	272
5.1.2.7	Unexpected ALMPs (68B Flit Mode Only).....	273
5.1.3	Applications of the vLSM State Transition Rules for 68B Flit Mode	274
5.1.3.1	Initial Link Training	274
5.1.3.2	Status Exchange Snapshot Example	277
5.1.3.3	L1 Abort Example.....	278
5.2	ARB/MUX Link Management Packets	279
5.2.1	ARB/MUX Bypass Feature.....	282
5.3	Arbitration and Data Multiplexing/Demultiplexing.....	282
6.0	Flex Bus Physical Layer	283
6.1	Overview	283
6.2	Flex Bus.CXL Framing and Packet Layout.....	284
6.2.1	Ordered Set Blocks and Data Blocks	284
6.2.2	68B Flit Mode	285
6.2.2.1	Protocol ID[15:0]	285
6.2.2.2	x16 Packet Layout.....	286
6.2.2.3	x8 Packet Layout	289
6.2.2.4	x4 Packet Layout	291
6.2.2.5	x2 Packet Layout	291
6.2.2.6	x1 Packet Layout	291
6.2.2.7	Special Case: CXL.io — When a TLP Ends on a Flit Boundary ..	292
6.2.2.8	Framing Errors.....	292
6.2.3	256B Flit Mode	293
6.2.3.1	256B Flit Format	294
6.2.3.2	CRC Corruption for Containment with 256B Flits	300
6.3	256B Flit Mode Retry Buffers.....	301
6.4	Link Training.....	301
6.4.1	PCIe Mode vs. Flex Bus.CXL Mode Selection.....	301
6.4.1.1	Hardware Autonomous Mode Negotiation.....	302
6.4.1.2	Virtual Hierarchy vs. Restricted CXL Device Negotiation	307
6.4.1.3	256B Flit Mode.....	308
6.4.1.4	Flit Mode and VH Negotiation	309
6.4.1.5	Flex Bus.CXL Negotiation with Maximum Supported Link Speed of 8 GT/s or 16 GT/s	310
6.4.1.6	Link Width Degradation and Speed Downgrade	310
6.5	68B Flit Mode: Recovery.Idle and Config.Idle Transitions to L0	310
6.6	L1 Abort Scenario.....	310
6.7	68B Flit Mode: Exit from Recovery	311
6.8	Retimers and Low Latency Mode.....	311
6.8.1	68B Flit Mode: SKP Ordered Set Frequency and L1/Recovery Entry	312
7.0	Switching	315
7.1	Overview	315
7.1.1	Single VCS	315
7.1.2	Multiple VCS.....	316
7.1.3	Multiple VCS with MLD Ports	317

EVALUATION COPY

Contents

7.2	7.1.4 vPPB Ordering	317
	Switch Configuration and Composition.....	318
7.2.1	CXL Switch Initialization Options	318
	7.2.1.1 Static Initialization	319
	7.2.1.2 Fabric Manager Boots First	320
	7.2.1.3 Fabric Manager and Host Boot Simultaneously	322
7.2.2	Sideband Signal Operation	323
7.2.3	Binding and Unbinding.....	324
	7.2.3.1 Binding and Unbinding of a Single Logical Device Port	324
	7.2.3.2 Binding and Unbinding of a Pooled Device.....	326
7.2.4	PPB and vPPB Behavior for MLD Ports	329
	7.2.4.1 MLD Type 1 Configuration Space Header	330
	7.2.4.2 MLD PCI*-compatible Configuration Registers	330
	7.2.4.3 MLD PCIe Capability Structure	330
	7.2.4.4 MLD PPB Secondary PCIe Capability Structure.....	333
	7.2.4.5 MLD Physical Layer 16.0 GT/s Extended Capability.....	333
	7.2.4.6 MLD Physical Layer 32.0 GT/s Extended Capability.....	334
	7.2.4.7 MLD Lane Margining at the Receiver Extended Capability	334
7.2.5	MLD ACS Extended Capability	335
7.2.6	MLD PCIe Extended Capabilities	335
7.2.7	MLD Advanced Error Reporting Extended Capability	335
7.2.8	MLD DPC Extended Capability	336
7.2.9	Switch Mailbox CCI	337
7.3	CXL.io, CXL.cachemem Decode and Forwarding.....	338
7.3.1	CXL.io	338
	7.3.1.1 CXL.io Decode	338
	7.3.1.2 RCD Support	338
7.3.2	CXL.cache.....	338
7.3.3	CXL.mem.....	339
	7.3.3.1 CXL.mem Request Decode.....	339
	7.3.3.2 CXL.mem Response Decode.....	339
7.3.4	FM-owned PPB CXL Handling	339
7.4	CXL Switch PM.....	339
7.4.1	CXL Switch ASPM L1.....	339
7.4.2	CXL Switch PCI-PM and L2	339
7.4.3	CXL Switch Message Management.....	339
7.5	CXL Switch RAS	340
7.6	Fabric Manager Application Programming Interface	340
7.6.1	CXL Fabric Management.....	341
7.6.2	Fabric Management Model	341
7.6.3	CCI Message Format and Transport Protocol	342
	7.6.3.1 Transport Details for MLD Components	343
7.6.4	CXL Switch Management.....	343
	7.6.4.1 Initial Configuration.....	343
	7.6.4.2 Dynamic Configuration.....	344
	7.6.4.3 MLD Port Management.....	344
7.6.5	MLD Component Management	344
7.6.6	Management Requirements for System Operations	345
	7.6.6.1 Initial System Discovery.....	345
	7.6.6.2 CXL Switch Discovery	346
	7.6.6.3 MLD and Switch MLD Port Management.....	346
	7.6.6.4 Event Notifications	346
	7.6.6.5 Binding Ports and LDs on a Switch.....	347
	7.6.6.6 Unbinding Ports and LDs on a Switch	347
	7.6.6.7 Hot-Add and Managed Hot-Removal of Devices	347
	7.6.6.8 Surprise Removal of Devices.....	348

EVALUATION COPY

Contents

7.6.7	Fabric Management Application Programming Interface.....	348
7.6.7.1	Physical Switch Command Set	349
7.6.7.2	Virtual Switch Command Set	355
7.6.7.3	MLD Port Command Set	359
7.6.7.4	MLD Component Command Set.....	364
7.6.7.5	Multi-Headed Device Command Set	371
7.6.7.6	DCD Management Command Set.....	374
7.6.8	Fabric Management Event Records	384
7.6.8.1	Physical Switch Event Records	384
7.6.8.2	Virtual CXL Switch Event Records	386
7.6.8.3	MLD Port Event Records	387
7.7	CXL Fabric Architecture	387
7.7.1	CXL Fabric Use Case Examples.....	388
7.7.1.1	Machine-learning Accelerators.....	388
7.7.1.2	HPC/Analytics Use Case	389
7.7.1.3	Composable Systems.....	389
7.7.2	Global-Fabric-Attached Memory (G-FAM).....	390
7.7.2.1	Overview	390
7.7.2.2	Host Physical Address View	391
7.7.2.3	G-FAM Capacity Management	392
7.7.2.4	G-FAM Request Routing, Interleaving, and Address Translations	394
7.7.2.5	G-FAM Access Protection	398
7.7.2.6	Global Memory Access Endpoint	400
7.7.2.7	Event Notifications from GFDs.....	401
7.7.3	Global Integrated Memory (GIM).....	402
7.7.3.1	Host GIM Physical Address View	403
7.7.3.2	Use Cases	404
7.7.3.3	Transaction Flows and Rules for GIM.....	405
7.7.3.4	Restrictions with Host-to-Host UIO Usages	407
7.7.4	Non-GIM Usages with VendPrefixL0	408
7.7.5	HBR and PBR Switch Configurations.....	408
7.7.5.1	PBR Forwarding Dependencies, Loops, and Deadlocks	410
7.7.6	PBR Switching Details.....	412
7.7.6.1	Virtual Hierarchies Spanning a Fabric.....	412
7.7.6.2	PBR Message Routing across the Fabric.....	413
7.7.6.3	PBR Message Routing within a Single PBR Switch.....	415
7.7.6.4	PBR Switch vDSP/vUSP Bindings and Connectivity	416
7.7.6.5	PID Use Models and Assignments	418
7.7.6.6	CXL Switch Message Format Conversion	419
7.7.6.7	HBR Switch Port Processing of CXL Messages	423
7.7.6.8	PBR Switch Port Processing of CXL Messages	425
7.7.6.9	PPB and vPPB Behavior of PBR Link Ports	427
7.7.7	Inter-Switch Links (ISLs)	433
7.7.7.1	.io Deadlock Avoidance on ISLs/PBR Fabric.....	433
7.7.8	PBR TLP Header (PTH) Rules.....	436
7.7.9	PBR Support for UIO Direct P2P to HDM	436
7.7.9.1	FAST Decoder Use for UIO Direct P2P to G-FAM.....	437
7.7.9.2	LDST Decoder Use for UIO Direct P2P to LD-FAM	437
7.7.9.3	ID-Based Re-Router for UIO Completions with LD-FAM	438
7.7.9.4	LDST and ID-Based Re-Router Access Protection	439
7.7.10	PBR Support for Direct P2P CXL.mem for Accelerators.....	439
7.7.10.1	Message Routing for Direct P2P CXL.mem Accesses with GFD.	440
7.7.10.2	Message Routing for Direct P2P CXL.mem Accesses with MLD.	440
7.7.11	PBR Link Events and Messages	441
7.7.11.1	PBR Link Fundamentals	442
7.7.11.2	CXL VDMs	443
7.7.11.3	Single VH Events.....	443

EVALUATION COPY

Contents

	7.7.11.4	Shared Link Events.....	447
	7.7.11.5	Switch Reported Events	448
	7.7.11.6	PBR Link CCI Message Format and Transport Protocol	449
7.7.12		PBR Fabric Management	450
	7.7.12.1	Fabric Boot and Initialization.....	450
	7.7.12.2	PBR Fabric Discovery	451
	7.7.12.3	Assigning and Binding PIDs	452
	7.7.12.4	Reporting Fabric Route Performance via CDAT.....	452
	7.7.12.5	Configuring CacheID in PBR Fabric.....	453
	7.7.12.6	Dynamic Fabric Changes	453
7.7.13		PBR Switch Command Set.....	454
	7.7.13.1	Identify PBR Switch (Opcode 5700h)	454
	7.7.13.2	Fabric Crawl Out (Opcode 5701h).....	455
	7.7.13.3	Get PBR Link Partner Info (Opcode 5702h)	457
	7.7.13.4	Get PID Target List (Opcode 5703h)	458
	7.7.13.5	Configure PID Assignment (Opcode 5704h)	459
	7.7.13.6	Get PID Binding (Opcode 5705h).....	460
	7.7.13.7	Configure PID Binding (Opcode 5706h)	461
	7.7.13.8	Get Table Descriptors (Opcode 5707h).....	462
	7.7.13.9	Get DRT (Opcode 5708h)	463
	7.7.13.10	Set DRT (Opcode 5709h)	464
	7.7.13.11	Get RGT (Opcode 570Ah)	464
	7.7.13.12	Set RGT (Opcode 570Bh)	466
	7.7.13.13	Get LDST/IDT Capabilities (Opcode 570Ch)	466
	7.7.13.14	Set LDST/IDT Configuration (Opcode 570Dh).....	467
	7.7.13.15	Get LDST Segment Entries (Opcode 570Eh).....	468
	7.7.13.16	Set LDST Segment Entries (Opcode 570Fh)	469
	7.7.13.17	Get LDST IDT DPID Entries (Opcode 5710h)	470
	7.7.13.18	Set LDST IDT DPID Entries (Opcode 5711h)	471
	7.7.13.19	Get Completer ID-Based Re-Router Entries (Opcode 5712h) ..	472
	7.7.13.20	Set Completer ID-Based Re-Router Entries (Opcode 5713h)...	473
	7.7.13.21	Get LDST Access Vector (Opcode 5714h).....	474
	7.7.13.22	Get VCS LDST Access Vector (Opcode 5715h).....	475
	7.7.13.23	Configure VCS LDST Access (Opcode 5716h)	475
7.7.14		Global Memory Access Endpoint Command Set	476
	7.7.14.1	Identify GAE (Opcode 5800h)	476
	7.7.14.2	Get PID Interrupt Vector (Opcode 5801h).....	477
	7.7.14.3	Get PID Access Vectors (Opcode 5802h)	478
	7.7.14.4	Get FAST/IDT Capabilities (Opcode 5803h).....	479
	7.7.14.5	Set FAST/IDT Configuration (Opcode 5804h)	480
	7.7.14.6	Get FAST Segment Entries (Opcode 5805h)	481
	7.7.14.7	Set FAST Segment Entries (Opcode 5806h)	483
	7.7.14.8	Get IDT DPID Entries (Opcode 5807h)	483
	7.7.14.9	Set IDT DPID Entries (Opcode 5808h).....	484
	7.7.14.10	Proxy GFD Management Command (Opcode 5809h)	485
	7.7.14.11	Get Proxy Thread Status (Opcode 580Ah).....	486
	7.7.14.12	Cancel Proxy Thread (Opcode 580Bh)	487
7.7.15		Global Memory Access Endpoint Management Command Set.....	487
	7.7.15.1	Identify VCS GAE (Opcode 5900h).....	488
	7.7.15.2	Get VCS PID Access Vectors (Opcode 5901h).....	488
	7.7.15.3	Configure VCS PID Access (Opcode 5902h)	489
	7.7.15.4	Get VendPrefixL0 State (Opcode 5903h)	490
	7.7.15.5	Set VendPrefixL0 State (Opcode 5904h)	490
8.0	Control and Status Registers	492
8.1	Configuration Space Registers.....	493
8.1.1	PCIe Designated Vendor-Specific Extended Capability (DVSEC) ID Assignment.....	493
8.1.2	CXL Data Object Exchange (DOE) Type Assignment.....	494

EVALUATION COPY

Contents

8.1.3	PCIe DVSEC for CXL Devices	494
8.1.3.1	DVSEC CXL Capability (Offset 0Ah).....	496
8.1.3.2	DVSEC CXL Control (Offset 0Ch)	497
8.1.3.3	DVSEC CXL Status (Offset 0Eh).....	498
8.1.3.4	DVSEC CXL Control2 (Offset 10h).....	498
8.1.3.5	DVSEC CXL Status2 (Offset 12h)	499
8.1.3.6	DVSEC CXL Lock (Offset 14h)	500
8.1.3.7	DVSEC CXL Capability2 (Offset 16h)	500
8.1.3.8	DVSEC CXL Range Registers	500
8.1.3.9	DVSEC CXL Capability3 (Offset 38h)	506
8.1.4	Non-CXL Function Map DVSEC	507
8.1.4.1	Non-CXL Function Map Register 0 (Offset 0Ch).....	508
8.1.4.2	Non-CXL Function Map Register 1 (Offset 10h).....	508
8.1.4.3	Non-CXL Function Map Register 2 (Offset 14h).....	508
8.1.4.4	Non-CXL Function Map Register 3 (Offset 18h).....	508
8.1.4.5	Non-CXL Function Map Register 4 (Offset 1Ch).....	509
8.1.4.6	Non-CXL Function Map Register 5 (Offset 20h).....	509
8.1.4.7	Non-CXL Function Map Register 6 (Offset 24h).....	509
8.1.4.8	Non-CXL Function Map Register 7 (Offset 28h).....	509
8.1.5	CXL Extensions DVSEC for Ports.....	510
8.1.5.1	CXL Port Extension Status (Offset 0Ah)	510
8.1.5.2	Port Control Extensions (Offset 0Ch)	511
8.1.5.3	Alternate Bus Base (Offset 0Eh)	512
8.1.5.4	Alternate Bus Limit (Offset 0Fh)	512
8.1.5.5	Alternate Memory Base (Offset 10h)	512
8.1.5.6	Alternate Memory Limit (Offset 12h)	512
8.1.5.7	Alternate Prefetchable Memory Base (Offset 14h)	513
8.1.5.8	Alternate Prefetchable Memory Limit (Offset 16h)	513
8.1.5.9	Alternate Memory Prefetchable Base High (Offset 18h)	513
8.1.5.10	Alternate Prefetchable Memory Limit High (Offset 1Ch)	513
8.1.5.11	CXL RCRB Base (Offset 20h).....	513
8.1.5.12	CXL RCRB Base High (Offset 24h).....	514
8.1.6	GPF DVSEC for CXL Port	514
8.1.6.1	GPF Phase 1 Control (Offset 0Ch)	515
8.1.6.2	GPF Phase 2 Control (Offset 0Eh)	515
8.1.7	GPF DVSEC for CXL Device	516
8.1.7.1	GPF Phase 2 Duration (Offset 0Ah)	516
8.1.7.2	GPF Phase 2 Power (Offset 0Ch).....	517
8.1.8	PCIe DVSEC for Flex Bus Port	517
8.1.9	Register Locator DVSEC	517
8.1.9.1	Register Offset Low (Offset: Varies).....	518
8.1.9.2	Register Offset High (Offset: Varies)	519
8.1.10	MLD DVSEC	519
8.1.10.1	Number of LD Supported (Offset 0Ah)	520
8.1.10.2	LD-ID Hot Reset Vector (Offset 0Ch)	520
8.1.11	Table Access DOE	520
8.1.11.1	Read Entry	521
8.1.12	Memory Device Configuration Space Layout.....	521
8.1.12.1	PCI Header - Class Code Register (Offset 09h)	521
8.1.12.2	Memory Device PCIe Capabilities and Extended Capabilities ...	522
8.1.13	Switch Mailbox CCI Configuration Space Layout	522
8.1.13.1	PCI Header - Class Code Register (Offset 09h)	522
8.2	Memory Mapped Registers.....	522
8.2.1	RCD Upstream Port and RCH Downstream Port Registers.....	524
8.2.1.1	RCH Downstream Port RCRB.....	524
8.2.1.2	RCD Upstream Port RCRB.....	526
8.2.1.3	Flex Bus Port DVSEC	528
8.2.2	Accessing Component Registers	533

EVALUATION COPY

Contents

8.2.3	Component Register Layout and Definition	533
8.2.4	CXL.cache and CXL.mem Registers.....	534
8.2.4.1	CXL Capability Header Register (Offset 00h)	536
8.2.4.2	CXL RAS Capability Header (Offset: Varies)	537
8.2.4.3	CXL Security Capability Header (Offset: Varies).....	537
8.2.4.4	CXL Link Capability Header (Offset: Varies)	537
8.2.4.5	CXL HDM Decoder Capability Header (Offset: Varies)	537
8.2.4.6	CXL Extended Security Capability Header (Offset: Varies)	538
8.2.4.7	CXL IDE Capability Header (Offset: Varies)	538
8.2.4.8	CXL Snoop Filter Capability Header (Offset: Varies)	538
8.2.4.9	CXL Timeout and Isolation Capability Header (Offset: Varies) .	539
8.2.4.10	CXL.cachemem Extended Register Capability (Offset: Varies) .	539
8.2.4.11	CXL BI Route Table Capability Header (Offset: Varies).....	539
8.2.4.12	CXL BI Decoder Capability Header (Offset: Varies).....	539
8.2.4.13	CXL Cache ID Route Table Capability Header (Offset: Varies) .	540
8.2.4.14	CXL Cache ID Decoder Capability Header (Offset: Varies)	540
8.2.4.15	CXL Extended HDM Decoder Capability Header (Offset: Varies).....	541
8.2.4.16	CXL Extended Metadata Capability Header (Offset: Varies) ...	541
8.2.4.17	CXL RAS Capability Structure.....	541
8.2.4.18	CXL Security Capability Structure	549
8.2.4.19	CXL Link Capability Structure.....	550
8.2.4.20	CXL HDM Decoder Capability Structure.....	556
8.2.4.21	CXL Extended Security Capability Structure	569
8.2.4.22	CXL IDE Capability Structure	570
8.2.4.23	CXL Snoop Filter Capability Structure.....	574
8.2.4.24	CXL Timeout and Isolation Capability Structure	574
8.2.4.25	CXL.cachemem Extended Register Capability	580
8.2.4.26	CXL BI Route Table Capability Structure.....	581
8.2.4.27	CXL BI Decoder Capability Structure	583
8.2.4.28	CXL Cache ID Route Table Capability Structure	585
8.2.4.29	CXL Cache ID Decoder Capability Structure	588
8.2.4.30	CXL Extended HDM Decoder Capability Structure.....	589
8.2.4.31	CXL Extended Metadata Capability Register	590
8.2.5	CXL ARB/MUX Registers	591
8.2.5.1	ARB/MUX PM Timeout Control Register (Offset 00h)	591
8.2.5.2	ARB/MUX Uncorrectable Error Status Register (Offset 04h) ...	592
8.2.5.3	ARB/MUX Uncorrectable Error Mask Register (Offset 08h) ...	592
8.2.5.4	ARB/MUX Arbitration Control Register for CXL.io (Offset 180h)	592
8.2.5.5	ARB/MUX Arbitration Control Register for CXL.cache and CXL.mem (Offset 1C0h)	593
8.2.6	BAR Virtualization ACL Register Block	593
8.2.6.1	BAR Virtualization ACL Size Register (Offset 00h)	594
8.2.7	CPMU Register Interface	594
8.2.7.1	Per CPMU Registers	595
8.2.7.2	Per Counter Unit Registers	598
8.2.8	CXL Device Register Interface.....	601
8.2.8.1	CXL Device Capabilities Array Register (Offset 00h)	602
8.2.8.2	CXL Device Capability Header Register (Offset: Varies).....	602
8.2.8.3	Device Status Registers (Offset: Varies).....	603
8.2.8.4	Mailbox Registers (Offset: Varies).....	603
8.2.8.5	Memory Device Capabilities	609
8.2.8.6	Switch Mailbox CCI Capability	611
8.2.9	Component Command Interface.....	611
8.2.9.1	Information and Status Command Set	615
8.2.9.2	Events	618
8.2.9.3	Firmware Update	646
8.2.9.4	Timestamp	651

8.2.9.5	Logs	652
8.2.9.6	Features	669
8.2.9.7	Maintenance	674
8.2.9.8	PBR Component Command Set	689
8.2.9.9	Memory Device Command Sets	692
8.2.9.10	FM API Commands	766
9.0	Reset, Initialization, Configuration, and Manageability	772
9.1	CXL Boot and Reset Overview	772
9.1.1	General	772
9.1.2	Comparing CXL and PCIe Behavior	773
9.1.2.1	Switch Behavior	773
9.2	CXL Device Boot Flow	774
9.3	CXL System Reset Entry Flow	774
9.4	CXL Device Sleep State Entry Flow	775
9.5	Function Level Reset (FLR)	776
9.6	Cache Management	777
9.7	CXL Reset	778
9.7.1	Effect on the Contents of the Volatile HDM	779
9.7.2	Software Actions	779
9.7.3	CXL Reset and Request Retry Status (RRS)	780
9.8	Global Persistent Flush (GPF)	780
9.8.1	Host and Switch Responsibilities	780
9.8.2	Device Responsibilities	781
9.8.3	Energy Budgeting	783
9.9	Hot-Plug	785
9.10	Software Enumeration	788
9.11	RCD Enumeration	788
9.11.1	RCD Mode	788
9.11.2	PCIe Software View of an RCH and RCD	789
9.11.3	System Firmware View of an RCH and RCD	789
9.11.4	OS View of an RCH and RCD	789
9.11.5	System Firmware-based RCD Enumeration Flow	790
9.11.6	RCD Discovery	790
9.11.7	eRCDs with Multiple Flex Bus Links	792
9.11.7.1	Single CPU Topology	792
9.11.7.2	Multiple CPU Topology	793
9.11.8	CXL Devices Attached to an RCH	794
9.12	CXL VH Enumeration	796
9.12.1	CXL Root Ports	797
9.12.2	CXL Virtual Hierarchy	797
9.12.3	Enumerating CXL RPs and DSPs	798
9.12.4	eRCD Connected to a CXL RP or DSP	799
9.12.4.1	Boot time Reconfiguration of CXL RP or DSP to Enable an eRCD	799
9.12.5	CXL eRCD below a CXL RP and DSP - Example	803
9.12.6	Mapping of Link and Protocol Registers in CXL VH	804
9.13	Software View of HDM	806
9.13.1	Memory Interleaving	806
9.13.1.1	Legal Interleaving Configurations: 12-way, 6-way, and 3-way	810
9.13.2	CXL Memory Device Label Storage Area	810
9.13.2.1	Overall LSA Layout	811
9.13.2.2	Label Index Blocks	812
9.13.2.3	Common Label Properties	814
9.13.2.4	Region Labels	815

EVALUATION COPY

Contents

9.13.3	Dynamic Capacity Device (DCD)	817
9.13.3.1	DCD Management By FM	821
9.13.3.2	Setting up Memory Sharing	822
9.13.3.3	Extent List Tracking.....	822
9.13.4	Capacity or Performance Degradation	823
9.14	Back-Invalidate Configuration	823
9.14.1	Discovery	824
9.14.2	Configuration	824
9.14.3	Mixed Configurations	827
9.14.3.1	BI-capable Type 2 Device.....	828
9.14.3.2	Type 2 Device Fallback Modes.....	828
9.14.3.3	BI-capable Type 3 Device.....	829
9.15	Cache ID Configuration and Routing.....	829
9.15.1	Host Capabilities	829
9.15.2	Downstream Port Decode Functionality	829
9.15.3	Upstream Switch Port Routing Functionality.....	830
9.15.4	Host Bridge Routing Functionality.....	831
9.16	UIO Direct P2P to HDM.....	833
9.16.1	Processing of UIO Direct P2P to HDM Messages	834
9.16.1.1	UIO Address Match (DSP and Root Port).....	834
9.16.1.2	UIO Address Match (CXL.mem Device).....	835
9.17	Direct P2P CXL.mem for Accelerators	835
9.17.1	Peer SLD Configuration	836
9.17.2	Peer MLD Configuration	836
9.17.3	Peer GFD Configuration.....	836
9.18	CXL OS Firmware Interface Extensions	836
9.18.1	CXL Early Discovery Table (CEDT)	836
9.18.1.1	CEDT Header	837
9.18.1.2	CXL Host Bridge Structure (CHBS).....	837
9.18.1.3	CXL Fixed Memory Window Structure (CFMWS).....	838
9.18.1.4	CXL XOR Interleave Math Structure (CXIMS).....	841
9.18.1.5	RCEC Downstream Port Association Structure (RDPAS)	842
9.18.1.6	CXL System Description Structure (CSDS)	842
9.18.2	CXL _OSC	843
9.18.2.1	Rules for Evaluating _OSC	845
9.18.3	CXL Root Device Specific Methods (_DSM).....	846
9.18.3.1	_DSM Function for Retrieving QTG ID	847
9.19	Manageability Model for CXL Devices.....	848
9.20	Component Command Interface	849
9.20.1	CCI Properties	849
9.20.2	MCTP-based CCI Properties	850
10.0	Power Management.....	851
10.1	Statement of Requirements	851
10.2	Policy-based Runtime Control - Idle Power - Protocol Flow	851
10.2.1	General	851
10.2.2	Package-level Idle (C-state) Entry and Exit Coordination	851
10.2.2.1	PMReq Message Generation and Processing Rules	852
10.2.3	PkgC Entry Flows	853
10.2.4	PkgC Exit Flows	855
10.2.5	CXL Physical Layer Power Management States	856
10.3	CXL Power Management.....	856
10.3.1	CXL PM Entry Phase 1.....	857
10.3.2	CXL PM Entry Phase 2.....	858

EVALUATION COPY

Contents

10.3.3	CXL PM Entry Phase 3.....	860
10.3.4	CXL Exit from ASPM L1	861
10.3.5	L0p Negotiation for 256B Flit Mode.....	862
10.4	CXL.io Link Power Management.....	862
10.4.1	CXL.io ASPM Entry Phase 1 for 256B Flit Mode	862
10.4.2	CXL.io ASPM L1 Entry Phase 1 for 68B Flit Mode.....	862
10.4.3	CXL.io ASPM L1 Entry Phase 2	863
10.4.4	CXL.io ASPM Entry Phase 3	863
10.5	CXL.cache + CXL.mem Link Power Management.....	863
11.0	CXL Security	864
11.1	CXL IDE Overview	864
11.2	CXL.io IDE.....	866
11.3	CXL.cachemem IDE	866
11.3.1	CXL.cachemem IDE Architecture in 68B Flit Mode.....	867
11.3.2	CXL.cachemem IDE Architecture in 256B Flit Mode	870
11.3.3	Encrypted PCRC.....	876
11.3.4	Cryptographic Keys and IV	878
11.3.5	CXL.cachemem IDE Modes	878
11.3.5.1	Discovery of Integrity Modes and Settings	879
11.3.5.2	Negotiation of Operating Mode and Settings.....	879
11.3.5.3	Rules for MAC Aggregation	879
11.3.6	Early MAC Termination	882
11.3.7	Handshake to Trigger the Use of Keys.....	885
11.3.8	Error Handling	885
11.3.9	Switch Support.....	886
11.3.10	IDE Termination Handshake	887
11.4	CXL.cachemem IDE Key Management (CXL_IDE_KM).....	887
11.4.1	CXL_IDE_KM Protocol Overview	889
11.4.2	Secure Messaging Layer Rules	889
11.4.3	CXL_IDE_KM Common Data Structures.....	890
11.4.4	Discovery Messages	891
11.4.5	Key Programming Messages	892
11.4.6	Activation/Key Refresh Messages	895
11.4.7	Get Key Messages.....	898
11.5	CXL Trusted Execution Environments Security Protocol (TSP)	901
11.5.1	Overview	901
11.5.2	Scope.....	902
11.5.3	Threat Model	902
11.5.3.1	Definitions.....	902
11.5.3.2	Assumptions.....	903
11.5.3.3	Threats and Mitigations	905
11.5.4	Reference Architecture	905
11.5.4.1	Architectural Scope	905
11.5.4.2	Determining TSP Support	906
11.5.4.3	CMA/SPDM.....	906
11.5.4.4	Authentication and Attestation	908
11.5.4.5	TE State Changes and Access Control	908
11.5.4.6	Memory Encryption	915
11.5.4.7	Transport Security.....	919
11.5.4.8	Configuration.....	920
11.5.4.9	Component Command Interfaces	927
11.5.4.10	Dynamic Capacity	927
11.5.5	TSP Requests and Responses.....	928
11.5.5.1	TSP Request Overview	928
11.5.5.2	TSP Response Overview	929

11.5.5.3	Request Response and CMA/SPDM Sessions.....	930
11.5.5.4	Version	930
11.5.5.5	Target Capabilities	931
11.5.5.6	Target Configuration.....	934
11.5.5.7	Optional Explicit TE State Change Requests and Responses....	943
11.5.5.8	Optional Target-based Memory Encryption Requests and Responses.....	945
11.5.5.9	Optional Delayed Completion Requests and Responses.....	953
11.5.5.10	Error Response	955
12.0 Reliability, Availability, and Serviceability	956
12.1	Supported RAS Features	956
12.2	CXL Error Handling	956
12.2.1	Protocol and Link Layer Error Reporting	956
12.2.1.1	RCH Downstream Port-detected Errors.....	957
12.2.1.2	RCD Upstream Port-detected Errors.....	958
12.2.1.3	RCD RCiEP-detected Errors.....	959
12.2.1.4	Header Log and Handling of Multiple Errors.....	960
12.2.2	CXL Root Ports, Downstream Switch Ports, and Upstream Switch Ports	961
12.2.3	CXL Device Error Handling	961
12.2.3.1	CXL.cache and CXL.mem Errors	962
12.2.3.2	Memory Error Logging and Signaling Enhancements	963
12.2.3.3	CXL Device Error Handling Flows	964
12.3	Isolation on CXL.cache and CXL.mem.....	964
12.3.1	CXL.cache Transaction Layer Behavior during Isolation	965
12.3.2	CXL.mem Transaction Layer Behavior during Isolation	966
12.4	CXL Viral Handling.....	966
12.4.1	Switch Considerations.....	966
12.4.2	Device Considerations	967
12.5	Maintenance.....	968
12.6	CXL Error Injection	968
13.0 Performance Considerations	969
13.1	Performance Recommendations.....	969
13.2	Performance Monitoring	971
14.0 CXL Compliance Testing	976
14.1	Applicable Devices under Test (DUTs)	976
14.2	Starting Configuration/Topology (Common for All Tests)	976
14.2.1	Test Topologies	977
14.2.1.1	Single Host, Direct Attached SLD EP (SHDA).....	977
14.2.1.2	Single Host, Switch Attached SLD EP (SHSW)	977
14.2.1.3	Single Host, Fabric Managed, Switch Attached SLD EP (SHSW-FM).....	978
14.2.1.4	Dual Host, Fabric Managed, Switch Attached SLD EP (DHSW-FM).....	979
14.2.1.5	Dual Host, Fabric Managed, Switch Attached MLD EP (DHSW-FM-MLD)	980
14.2.1.6	Cascaded Switch Topologies	981
14.3	CXL.io and CXL.cache Application Layer/Transaction Layer Testing	982
14.3.1	General Testing Overview	982
14.3.2	Algorithms	983
14.3.3	Algorithm 1a: Multiple Write Streaming.....	983
14.3.4	Algorithm 1b: Multiple Write Streaming with Bogus Writes.....	984
14.3.5	Algorithm 2: Producer Consumer Test.....	985
14.3.6	Test Descriptions	986
14.3.6.1	Application Layer/Transaction Layer Tests	986
14.4	Link Layer Testing	991

EVALUATION COPY

Contents

14.4.1	RSVD Field Testing CXL.cachemem.....	991
14.4.1.1	Device Test	991
14.4.1.2	Host Test	991
14.4.2	CRC Error Injection RETRY_PHY_REINIT.....	992
14.4.3	CRC Error Injection RETRY_ABORT	993
14.5	ARB/MUX	994
14.5.1	Reset to Active Transition.....	994
14.5.2	ARB/MUX Multiplexing	994
14.5.3	Active to L1.x Transition (If Applicable).....	995
14.5.4	L1.x State Resolution (If Applicable)	996
14.5.5	Active to L2 Transition	997
14.5.6	L1 to Active Transition (If Applicable)	997
14.5.7	Reset Entry	998
14.5.8	Entry into L0 Synchronization	998
14.5.9	ARB/MUX Tests Requiring Injection Capabilities.....	999
14.5.9.1	ARB/MUX Bypass	999
14.5.9.2	PM State Request Rejection	999
14.5.9.3	Unexpected Status ALMP.....	1000
14.5.9.4	ALMP Error	1000
14.5.9.5	Recovery Re-entry	1001
14.5.10	L0p Feature	1001
14.5.10.1	Positive ACK for L0p	1001
14.5.10.2	Force NAK for L0p Request	1002
14.6	Physical Layer.....	1002
14.6.1	Tests Applicable to 68B Flit Mode	1002
14.6.1.1	Protocol ID Checks	1002
14.6.1.2	NULL Flit.....	1003
14.6.1.3	EDS Token	1003
14.6.1.4	Correctable Protocol ID Error	1004
14.6.1.5	Uncorrectable Protocol ID Error	1004
14.6.1.6	Unexpected Protocol ID.....	1004
14.6.1.7	Recovery.Idle/Config.Idle Transition to L0	1005
14.6.1.8	Uncorrectable Mismatched Protocol ID Error	1005
14.6.2	Drift Buffer (If Applicable)	1006
14.6.3	SKP OS Scheduling/Alternation (If Applicable).....	1006
14.6.4	SKP OS Exiting the Data Stream (If Applicable).....	1007
14.6.5	Link Initialization Resolution	1007
14.6.6	Hot Add Link Initialization Resolution	1008
14.6.7	Link Speed Advertisement.....	1009
14.6.8	Link Speed Degradation - CXL Mode	1010
14.6.9	Link Speed Degradation below 8 GT/s.....	1010
14.6.10	Tests Requiring Injection Capabilities.....	1010
14.6.10.1	TLP Ends on Flit Boundary	1010
14.6.10.2	Failed CXL Mode Link Up	1011
14.6.11	Link Initialization in Standard 256B Flit Mode	1011
14.6.12	Link Initialization in Latency-Optimized 256B Flit Mode.....	1012
14.6.13	Sync Header Bypass (If Applicable)	1012
14.7	Switch Tests.....	1013
14.7.1	Introduction to Switch Types	1013
14.7.2	Compliance Testing	1013
14.7.2.1	HBR Switch Assumptions.....	1013
14.7.2.2	PBR Switch Assumptions	1015
14.7.3	Unmanaged HBR Switch.....	1016
14.7.4	Reset Propagation	1017
14.7.4.1	Host PERST# Propagation	1017
14.7.4.2	LTSSM Hot Reset	1018

EVALUATION COPY

Contents

14.7.5	14.7.4.3 Secondary Bus Reset (SBR) Propagation	1020
	Managed Hot-Plug - Adding a New Endpoint Device	1024
	14.7.5.1 Managed Add of an SLD Component	1024
	14.7.5.2 Managed Add of an MLD Component (HBR Switch Only)	1025
	14.7.5.3 Managed Add of an MLD Component to an SLD Port (HBR Switch Only)	1025
14.7.6	Managed Hot-Plug Removal of an Endpoint Device.....	1026
	14.7.6.1 Managed Removal of an SLD Component from a VCS (HBR Switch).....	1026
	14.7.6.2 Managed Removal of an SLD Component (PBR Switch).....	1026
	14.7.6.3 Managed Removal of an MLD Component from a Switch (HBR Switch Only)	1026
14.7.7	14.7.6.4 Removal of a Device from an Unbound Port	1027
	Bind/Unbind and Port Access Operations	1027
	14.7.7.1 Binding and Granting Port Access of Pooled Resources to Hosts	1027
	14.7.7.2 Unbinding Resources from Hosts without Removing the Endpoint Devices	1029
14.7.8	Error Injection	1030
	14.7.8.1 AER Error Injection.....	1030
14.8	Configuration Register Tests	1032
14.8.1	Device Presence	1032
14.8.2	CXL Device Capabilities.....	1033
14.8.3	DOE Capabilities	1034
14.8.4	DVSEC Control Structure.....	1035
14.8.5	DVSEC CXL Capability.....	1036
14.8.6	DVSEC CXL Control	1036
14.8.7	DVSEC CXL Lock	1037
14.8.8	DVSEC CXL Capability2.....	1038
14.8.9	Non-CXL Function Map DVSEC	1038
14.8.10	CXL Extensions DVSEC for Ports Header.....	1039
14.8.11	Port Control Override.....	1040
14.8.12	GPF DVSEC Port Capability	1041
14.8.13	GPF Port Phase 1 Control	1041
14.8.14	GPF Port Phase 2 Control	1042
14.8.15	GPF DVSEC Device Capability	1042
14.8.16	GPF Device Phase 2 Duration.....	1043
14.8.17	Flex Bus Port DVSEC Capability Header	1044
14.8.18	DVSEC Flex Bus Port Capability.....	1044
14.8.19	Register Locator	1045
14.8.20	MLD DVSEC Capability Header	1045
14.8.21	MLD DVSEC Number of LD Supported	1046
14.8.22	Table Access DOE	1047
14.8.23	PCI Header - Class Code Register	1047
14.9	Reset and Initialization Tests	1048
14.9.1	Warm Reset Test	1048
14.9.2	Cold Reset Test	1048
14.9.3	Sleep State Test	1049
14.9.4	Function Level Reset Test.....	1049
14.9.5	CXL Range Setup Time	1049
14.9.6	FLR Memory	1050
14.9.7	CXL_Reset Test	1050
14.9.8	Global Persistent Flush (GPF).....	1052
	14.9.8.1 Host and Switch Test	1053
	14.9.8.2 Device Test	1053
14.9.9	Hot-Plug Test	1054

EVALUATION COPY

Contents

14.9.10	Device to Host Cache Viral Injection	1054
14.9.11	Device to Host Mem Viral Injection	1055
14.10	Power Management Tests	1055
14.10.1	Pkg-C Entry (Device Test)	1055
14.10.2	Pkg-C Entry Reject (Device Test)	1056
14.10.3	Pkg-C Entry (Host Test)	1057
14.11	Security	1057
14.11.1	Component Measurement and Authentication	1057
14.11.1.1	DOE CMA Instance	1057
14.11.1.2	FLR while Processing DOE CMA Request	1058
14.11.1.3	OOB CMA while in Fundamental Reset	1058
14.11.1.4	OOB CMA while Function Gets FLR	1059
14.11.1.5	OOB CMA during Conventional Reset	1060
14.11.2	Link Integrity and Data Encryption CXL.io IDE	1060
14.11.2.1	CXL.io Link IDE Streams Functional	1061
14.11.2.2	CXL.io Link IDE Streams Aggregation	1061
14.11.2.3	CXL.io Link IDE Streams PCRC	1062
14.11.2.4	CXL.io Selective IDE Stream Functional	1063
14.11.2.5	CXL.io Selective IDE Streams Aggregation	1063
14.11.2.6	CXL.io Selective IDE Streams PCRC	1064
14.11.3	CXL.cachemem IDE	1065
14.11.3.1	CXL.cachemem IDE Capability (SHDA, SHSW)	1065
14.11.3.2	Establish CXL.cachemem IDE (SHDA) in Standard 256B Flit Mode	1065
14.11.3.3	Establish CXL.cachemem IDE (SHSW)	1066
14.11.3.4	Establish CXL.cachemem IDE (SHDA) Latency-Optimized 256B Flit Mode	1067
14.11.3.5	Establish CXL.cachemem IDE (SHDA) 68B Flit Mode	1068
14.11.3.6	Locally Generate IV (SHDA)	1069
14.11.3.7	Data Encryption – Decryption and Integrity Testing with Containment Mode for MAC Generation and Checking	1070
14.11.3.8	Data Encryption – Decryption and Integrity Testing with Skid Mode for MAC Generation and Checking	1070
14.11.3.9	Key Refresh	1071
14.11.3.10	Early MAC Termination	1071
14.11.3.11	Error Handling	1072
14.11.4	Certificate Format/Certificate Chain	1077
14.11.5	Security RAS	1078
14.11.5.1	CXL.io Poison Inject from Device	1078
14.11.5.2	CXL.cache Poison Inject from Device	1079
14.11.5.3	CXL.cache CRC Inject from Device	1080
14.11.5.4	CXL.mem Poison Injection	1082
14.11.5.5	CXL.mem CRC Injection	1082
14.11.5.6	Flow Control Injection	1083
14.11.5.7	Unexpected Completion Injection	1084
14.11.5.8	Completion Timeout Injection	1086
14.11.5.9	Memory Error Injection and Logging	1087
14.11.5.10	CXL.io Viral Inject from Device	1088
14.11.5.11	CXL.cache Viral Inject from Device	1089
14.11.6	Security Protocol and Data Model	1091
14.11.6.1	SPDM GET_VERSION	1091
14.11.6.2	SPDM GET_CAPABILITIES	1092
14.11.6.3	SPDM NEGOTIATE_ALGORITHMS	1093
14.11.6.4	SPDM GET_DIGESTS	1094
14.11.6.5	SPDM GET_CERTIFICATE	1095
14.11.6.6	SPDM CHALLENGE	1096
14.11.6.7	SPDM GET_MEASUREMENTS Count	1097
14.11.6.8	SPDM GET_MEASUREMENTS All	1098

EVALUATION COPY

Contents

14.11.6.9	SPDM GET_MEASUREMENTS Repeat with Signature	1099
14.11.6.10	SPDM CHALLENGE Sequences.....	1100
14.11.6.11	SPDM ErrorCode Unsupported Request.....	1102
14.11.6.12	SPDM Major Version Invalid.....	1103
14.11.6.13	SPDM ErrorCode UnexpectedRequest.....	1103
14.12	Reliability, Availability, and Serviceability.....	1104
14.12.1	RAS Configuration	1105
14.12.1.1	AER Support.....	1105
14.12.1.2	CXL.io Poison Injection from Device to Host	1106
14.12.1.3	CXL.cache Poison Injection	1107
14.12.1.4	CXL.cache CRC Injection	1109
14.12.1.5	CXL.mem Link Poison Injection	1111
14.12.1.6	CXL.mem CRC Injection	1111
14.12.1.7	Flow Control Injection	1112
14.12.1.8	Unexpected Completion Injection	1113
14.12.1.9	Completion Timeout	1113
14.12.1.10	CXL.mem Media Poison Injection	1114
14.12.1.11	CXL.mem LSA Poison Injection.....	1114
14.12.1.12	CXL.mem Device Health Injection.....	1115
14.13	Memory Mapped Registers.....	1115
14.13.1	CXL Capability Header	1115
14.13.2	CXL RAS Capability Header.....	1116
14.13.3	CXL Security Capability Header.....	1116
14.13.4	CXL Link Capability Header.....	1117
14.13.5	CXL HDM Decoder Capability Header	1117
14.13.6	CXL Extended Security Capability Header	1118
14.13.7	CXL IDE Capability Header	1118
14.13.8	CXL HDM Decoder Capability Register	1119
14.13.9	CXL HDM Decoder Commit	1119
14.13.10	CXL HDM Decoder Zero Size Commit	1120
14.13.11	CXL Snoop Filter Capability Header.....	1120
14.13.12	CXL Device Capabilities Array Register	1121
14.13.13	Device Status Registers Capabilities Header Register	1122
14.13.14	Primary Mailbox Registers Capabilities Header Register	1122
14.13.15	Secondary Mailbox Registers Capabilities Header Register	1122
14.13.16	Memory Device Status Registers Capabilities Header Register	1123
14.13.17	CXL Timeout and Isolation Capability Header	1123
14.13.18	CXL.cachemem Extended Register Header.....	1124
14.13.19	CXL BI Route Table Capability Header	1125
14.13.20	CXL BI Decoder Capability Header	1125
14.13.21	CXL Cache ID Route Table Header.....	1126
14.13.22	CXL Cache ID Decoder Capability Header	1126
14.13.23	CXL Extended HDM Decoder Capability Header.....	1127
14.14	Memory Device Tests	1127
14.14.1	DVSEC CXL Range 1 Size Low Registers	1127
14.14.2	DVSEC CXL Range 2 Size Low Registers	1128
14.15	Sticky Register Tests	1129
14.15.1	Sticky Register Test	1129
14.16	Device Capability and Test Configuration Control	1131
14.16.1	CXL Device Test Capability Advertisement	1131
14.16.2	Debug Capabilities in Device.....	1132
14.16.2.1	Error Logging	1132
14.16.2.2	Event Monitors.....	1133
14.16.3	Compliance Mode DOE.....	1134
14.16.3.1	Compliance Mode Capability	1134
14.16.3.2	Compliance Mode Status	1135

14.16.3.3	Compliance Mode Halt All	1136
14.16.3.4	Compliance Mode Multiple Write Streaming.....	1136
14.16.3.5	Compliance Mode Producer-Consumer.....	1137
14.16.3.6	Test Algorithm 1b Multiple Write Streaming with Bogus Writes	1138
14.16.3.7	Inject Link Poison.....	1139
14.16.3.8	Inject CRC	1140
14.16.3.9	Inject Flow Control	1140
14.16.3.10	Toggle Cache Flush	1141
14.16.3.11	Inject MAC Delay	1141
14.16.3.12	Insert Unexpected MAC.....	1142
14.16.3.13	Inject Viral	1142
14.16.3.14	Inject ALMP in Any State.....	1143
14.16.3.15	Ignore Received ALMP	1144
14.16.3.16	Inject Bit Error in Flit.....	1144
14.16.3.17	Inject Memory Device Poison	1145
A	Taxonomy.....	1148
A.1	Accelerator Usage Taxonomy	1148
A.2	Bias Model Flow Example – From CPU	1149
A.3	CPU Support for Bias Modes.....	1150
	A.3.1 Remote Snoop Filter	1150
	A.3.2 Directory in Accelerator-attached Memory	1150
A.4	Giant Cache Model.....	1151
B	Unordered I/O to Support Peer-to-Peer Directly to HDM-DB.....	1152
C	Memory Protocol Tables	1153
C.1	HDM-D and HDM-DB Requests	1155
	C.1.1 Forward Flows for HDM-D.....	1159
	C.1.2 BISnp for HDM-DB	1161
C.2	HDM-H Requests	1163
C.3	HDM-D/HDM-DB RWD	1165
C.4	HDM-H RWD	1166

Figures

1-1	Conceptual Diagram of Device Attached to Processor via CXL.....	60
1-2	Fan-out and Pooling Enabled by Switches.....	61
1-3	Direct Peer-to-Peer Access to an HDM Memory by PCIe/CXL Devices without Going through the Host	62
1-4	Shared Memory across Multiple Virtual Hierarchies	62
1-5	CPU Flex Bus Port Example	63
1-6	Flex Bus Usage Model Examples.....	64
1-7	Remote Far Memory Usage Model Example.....	64
1-8	CXL Downstream Port Connections	65
1-9	Conceptual Diagram of Flex Bus Layering	66
2-1	CXL Device Types	69
2-2	Type 1 Device - Device with Cache	70
2-3	Type 2 Device - Device with Memory	71
2-4	Type 2 Device - Host Bias	72
2-5	Type 2 Device - Device Bias	73
2-6	Type 3 Device - HDM-H Memory Expander	74
2-7	Head-to-LD Mapping in MH-SLDs	79
2-8	Head-to-LD Mapping in MH-MLDs	80

EVALUATION COPY

Figures

3-1	Flex Bus Layers - CXL.io Transaction Layer Highlighted	83
3-2	CXL Power Management Messages Packet Format - Non-Flit Mode	85
3-3	CXL Power Management Messages Packet Format - Flit Mode.....	85
3-4	Power Management Credits and Initialization	88
3-5	CXL EFN Messages Packet Format - Non-Flit Mode.....	90
3-6	CXL EFN Messages Packet Format - Flit Mode.....	90
3-7	ATS 64-bit Request with CXL Indication - Non-Flit Mode	91
3-8	Valid .io TLP Formats on PBR Links	94
3-9	Host Management Transaction Flows of GFD	97
3-10	CXL.cache Channels	105
3-11	CXL.cache Read Behavior	112
3-12	CXL.cache Read0 Behavior	113
3-13	CXL.cache Device to Host Write Behavior	114
3-14	CXL.cache WrInv Transaction	115
3-15	WOWrInv/F with FastGO/ExtCmp	116
3-16	CXL.cache Read0-Write Semantics	117
3-17	CXL.cache Snoop Behavior	124
3-18	CXL.mem Channels for Devices	133
3-19	CXL.mem Channels for Hosts	134
3-20	Flows for Back-Invalidate Snoops on CXL.mem Legend	166
3-21	Example BISnp with Blocking of M2S Req.....	167
3-22	BISnp Early Conflict	168
3-23	BISnp Late Conflict	169
3-24	Block BISnp with Block Response	170
3-25	Block BISnp with Cacheline Response	171
3-26	Flows for Type 1 Devices and Type 2 Devices Legend	172
3-27	Example Cacheable Read from Host	172
3-28	Example Read for Ownership from Host	173
3-29	Example Non Cacheable Read from Host	174
3-30	Example Ownership Request from Host - No Data Required	174
3-31	Example Flush from Host	175
3-32	Example Weakly Ordered Write from Host	176
3-33	Example Write from Host with Invalid Host Caches	177
3-34	Example Write from Host with Valid Host Caches.....	178
3-35	Example Device Read to Device-attached Memory (HDM-D)	179
3-36	Example Device Read to Device-attached Memory (HDM-DB).....	180
3-37	Example Device Write to Device-Attached Memory in Host Bias (HDM-D).....	181
3-38	Example Device Write to Device-attached Memory in Host Bias (HDM-DB).....	182
3-39	Example Device Write to Device-attached Memory	183
3-40	Example Host to Device Bias Flip (HDM-D)	184
3-41	Example MemSpecRd	185
3-42	Read from Host to HDM-H.....	185
3-43	Write from Host to All HDM Regions	186
4-1	Flex Bus Layers - CXL.io Link Layer Highlighted.....	187
4-2	Flex Bus Layers - CXL.cache + CXL.mem Link Layer Highlighted	189
4-3	CXL.cachemem Protocol Flit Overview.....	190
4-4	CXL.cachemem All Data Flit Overview	191
4-5	Example of a Protocol Flit from Device to Host	192
4-6	H0 - H2D Req + H2D Rsp.....	198
4-7	H1 - H2D Data Header + H2D Rsp + H2D Rsp	198

EVALUATION COPY

Figures

4-8	H2 - H2D Req + H2D Data Header	199
4-9	H3 - 4 H2D Data Header	199
4-10	H4 - M2S RwD Header	199
4-11	H5 - M2S Req	200
4-12	H6 - MAC	200
4-13	G0 - H2D/M2S Data	200
4-14	G0 - M2S Byte Enable	201
4-15	G1 - 4 H2D Rsp	201
4-16	G2 - H2D Req + H2D Data Header + H2D Rsp	201
4-17	G3 - 4 H2D Data Header + H2D Rsp	202
4-18	G4 - M2S Req + H2D Data Header	202
4-19	G5 - M2S RwD Header + H2D Rsp	202
4-20	H0 - D2H Data Header + 2 D2H Rsp + S2M NDR	203
4-21	H1 - D2H Req + D2H Data Header	203
4-22	H2 - 4 D2H Data Header + D2H Rsp	204
4-23	H3 - S2M DRS Header + S2M NDR	204
4-24	H4 - 2 S2M NDR	204
4-25	H5 - 2 S2M DRS Header	205
4-26	H6 - MAC	205
4-27	G0 - D2H/S2M Data	205
4-28	G0 - D2H Byte Enable	206
4-29	G1 - D2H Req + 2 D2H Rsp	206
4-30	G2 - D2H Req + D2H Data Header + D2H Rsp	206
4-31	G3 - 4 D2H Data Header	207
4-32	G4 - S2M DRS Header + 2 S2M NDR	207
4-33	G5 - 2 S2M NDR	207
4-34	G6 - 3 S2M DRS Header	208
4-35	LLCRD Flit Format (Only Slot 0 is Valid; Others are Reserved)	212
4-36	RETRY Flit Format (Only Slot 0 is Valid; Others are Reserved)	213
4-37	IDE Flit Format (Only Slot 0 is Valid; Others are Reserved)	213
4-38	INIT Flit Format (Only Slot 0 is Valid; Others are Reserved)	213
4-39	Retry Buffer and Related Pointers	218
4-40	CXL.cachemem Replay Diagram	224
4-41	Standard 256B Flit	227
4-42	Latency-Optimized (LOpt) 256B Flit	227
4-43	256B Packing: Slot and Subset Definition	231
4-44	256B Packing: G0/H0/HS0 HBR Messages	232
4-45	256B Packing: G0/H0 PBR Messages	232
4-46	256B Packing: G1/H1/HS1 HBR Messages	233
4-47	256B Packing: G1/H1 PBR Messages	233
4-48	256B Packing: G2/H2/HS2 HBR Messages	234
4-49	256B Packing: G2/H2 PBR Messages	234
4-50	256B Packing: G3/H3/HS3 HBR Messages	235
4-51	256B Packing: G3/H3 PBR Messages	235
4-52	256B Packing: G4/H4/HS4 HBR Messages	236
4-53	256B Packing: G4/H4 PBR Messages	236
4-54	256B Packing: G5/H5/HS5 HBR Messages	237
4-55	256B Packing: G5/H5 PBR Messages	237
4-56	256B Packing: G6/H6/HS6 HBR Messages	238
4-57	256B Packing: G6/H6 PBR Messages	238

EVALUATION COPY

Figures

4-58	256B Packing: G7/H7/HS7 HBR Messages	239
4-59	256B Packing: G7/H7 PBR Messages	239
4-60	256B Packing: G12/H12/HS12 HBR Messages	240
4-61	256B Packing: G12/H12 PBR Messages	240
4-62	256B Packing: G13/H13/HS13 HBR Messages	241
4-63	256B Packing: G13/H13 PBR Messages	241
4-64	256B Packing: G14/H14/HS14 HBR Messages	242
4-65	256B Packing: G14/H14 PBR Messages	242
4-66	256B Packing: G15/H15/HS15 HBR Messages	243
4-67	256B Packing: G15/H15 PBR Messages	243
4-68	256B Packing: Implicit Data	244
4-69	256B Packing: Implicit Trailer Rwd	244
4-70	256B Packing: Implicit Trailer DRS	244
4-71	256B Packing: Byte-Enable Trailer for D2H Data	245
4-72	Header Slot Decode Example	246
4-73	DRS Trailer Slot Decode Example	247
4-74	256B Packing: H8/HS8 Link Layer Control Message Slot Format	254
4-75	Viral Error Message Injection Standard 256B Flit	255
4-76	Viral Error Message Injection LOpt 256B Flit	255
5-1	Flex Bus Layers - CXL ARB/MUX Highlighted	258
5-2	Entry to Active Protocol Exchange	263
5-3	Example Status Exchange	264
5-4	CXL Entry to Active Example Flow	266
5-5	CXL Entry to PM State Example	267
5-6	Successful PM Entry following PM Retry	268
5-7	PM Abort before Downstream Port PM Acceptance	268
5-8	PM Abort after Downstream Port PM Acceptance	269
5-9	Example of a PMNAK Flow	270
5-10	CXL Recovery Exit Example Flow	272
5-11	CXL Exit from PM State Example	273
5-12	Both Upstream Port and Downstream Port Hide Recovery Transitions from ARB/MUX	274
5-13	Both Upstream Port and Downstream Port Notify ARB/MUX of Recovery Transitions	275
5-14	Downstream Port Hides Initial Recovery, Upstream Port Does Not	276
5-15	Upstream Port Hides Initial Recovery, Downstream Port Does Not	277
5-16	Snapshot Example during Status Synchronization	278
5-17	L1 Abort Example	279
5-18	ARB/MUX Link Management Packet Format	279
5-19	ALMP Byte Positions in Standard 256B Flit	280
5-20	ALMP Byte Positions in Latency-Optimized 256B Flit	280
6-1	Flex Bus Layers - Physical Layer Highlighted	283
6-2	Flex Bus x16 Packet Layout	287
6-3	Flex Bus x16 Protocol Interleaving Example	288
6-4	Flex Bus x8 Packet Layout	289
6-5	Flex Bus x8 Protocol Interleaving Example	290
6-6	Flex Bus x4 Packet Layout	291
6-7	CXL.io TLP Ending on Flit Boundary Example	292
6-8	Standard 256B Flit	294
6-9	CXL.io Standard 256B Flit	294

6-10	Standard 256B Flit Applied to Physical Lanes (x16)	296
6-11	Latency-Optimized 256B Flit	297
6-12	CXLio Latency-Optimized 256B Flit	299
6-13	Different Methods for Generating 6-Byte CRC	300
6-14	Flex Bus Mode Negotiation during Link Training (Sample Flow)	306
6-15	NULL Flit with EDS and Sync Header Bypass Optimization	313
6-16	NULL Flit with EDS and 128b/130b Encoding	314
7-1	Example of a Single VCS	315
7-2	Example of a Multiple VCS with SLD Ports	316
7-3	Example of a Multiple Root Switch Port with Pooled Memory Devices	317
7-4	Static CXL Switch with Two VCSs	319
7-5	Example of CXL Switch Initialization when FM Boots First	320
7-6	Example of CXL Switch after Initialization Completes	321
7-7	Example of Switch with Fabric Manager and Host Boot Simultaneously	322
7-8	Example of Simultaneous Boot after Binding	323
7-9	Example of Binding and Unbinding of an SLD Port	324
7-10	Example of CXL Switch Configuration after an Unbind Command	325
7-11	Example of CXL Switch Configuration after a Bind Command	326
7-12	Example of a CXL Switch before Binding of LDs within Pooled Device	327
7-13	Example of a CXL Switch after Binding of LD-ID 1 within Pooled Device	328
7-14	Example of a CXL Switch after Binding of LD-IDs 0 and 1 within Pooled Device	329
7-15	Multi-function Upstream vPPB.....	337
7-16	Single-function Mailbox CCI.....	337
7-17	CXL Switch with a Downstream Link Auto-negotiated to Operate in RCD Mode	338
7-18	Example of Fabric Management Model	341
7-19	CCI Message Format	342
7-20	Tunneling Commands to an MLD through a CXL Switch	343
7-21	Example of MLD Management Requiring Tunneling	345
7-22	Tunneling Commands to an LD in an MLD.....	360
7-23	Tunneling Commands to an LD in an MLD through a CXL Switch.....	360
7-24	Tunneling Commands to the LD Pool CCI in a Multi-Headed Device	361
7-25	High-level CXL Fabric Diagram	388
7-26	ML Accelerator Use Case	389
7-27	HPC/Analytics Use Case	389
7-28	Sample System Topology for Composable Systems.....	390
7-29	Example Host Physical Address View	392
7-30	Example HPA Mapping to DMPs	393
7-31	G-FAM Request Routing, Interleaving, and Address Translations	395
7-32	Memory Access Protection Levels	399
7-33	GFD Dynamic Capacity Access Protections	400
7-34	PBR Fabric Providing LD-FAM and G-FAM Resources.....	401
7-35	PBR Fabric Providing Only G-FAM Resources	401
7-36	CXL Fabric Example with Multiple Host Domains and Memory Types	403
7-37	Example Host Physical Address View with GFD and GIM	403
7-38	Example Multi-host CXL Cluster with Memory on Host and Device Exposed as GIM.	404
7-39	Example ML Cluster Supporting Cross-domain Access through GIM.....	405
7-40	GIM Access Flows Using FASTs	405
7-41	GIM Access Flows without FASTs.....	406
7-42	Example Supported Switch Configurations	409
7-43	Example PBR Mesh Topology	410

7-44	Example Routing Scheme for a Mesh Topology.....	411
7-45	Physical Topology and Logical View	413
7-46	Example PBR Fabric	417
7-47	ISL Message Class Sub-channels.....	433
7-48	Deadlock Avoidance Mechanism on ISL.....	435
7-49	Update-FC DLLP Format on ISL	436
7-50	Example Topology with Direct P2P CXL.mem with GFD.....	440
7-51	Example Topology with Direct P2P CXL.mem with MLD.....	441
7-52	Single VH	444
7-53	Shared Link Events	447
7-54	Tunneling Commands to Remote Devices	455
7-55	Tunneling Commands to Remote Devices with No Assigned PID	456
8-1	PCIe DVSEC for CXL Devices	495
8-2	Non-CXL Function Map DVSEC	507
8-3	CXL Extensions DVSEC for Ports	510
8-4	GPF DVSEC for CXL Port	514
8-5	GPF DVSEC for CXL Device	516
8-6	Register Locator DVSEC with 3 Register Block Entries	517
8-7	MLD DVSEC	519
8-8	RCD and RCH Memory Mapped Register Regions	524
8-9	RCH Downstream Port RCRB	525
8-10	RCD Upstream Port RCRB	527
8-11	PCIe DVSEC for Flex Bus Port	529
8-12	CXL Device Registers.....	601
8-13	Mailbox Registers	605
9-1	PMREQ/RESETPREP Propagation by CXL Switch.....	774
9-2	CXL Device Reset Entry Flow	775
9-3	CXL Device Sleep State Entry Flow	776
9-4	PCIe Software View of an RCH and RCD	789
9-5	One CPU Connected to a Dual-Headed RCD by Two Flex Bus Links	792
9-6	Two CPUs Connected to One CXL Device by Two Flex Bus Links.....	793
9-7	CXL Device Remaps Upstream Port and Component Registers.....	795
9-8	CXL Device that Does Not Remap Upstream Port and Component Registers	796
9-9	CXL Root Port/DSP State Diagram.....	799
9-10	eRCD MMIO Address Decode - Example	801
9-11	eRCD Configuration Space Decode - Example	802
9-12	Physical Topology - Example	803
9-13	Software View	804
9-14	CXL Link/Protocol Register Mapping in a CXL VH	805
9-15	CXL Link/Protocol Registers in a CXL Switch	805
9-16	One-level Interleaving at Switch - Example	808
9-17	Two-level Interleaving	808
9-18	Three-level Interleaving Example	809
9-19	Overall LSA Layout	811
9-20	Fletcher64 Checksum Algorithm in C	812
9-21	Sequence Numbers in Label Index Blocks	813
9-22	Extent List Example (No Sharing).....	818
9-23	Shared Extent List Example.....	818
9-24	DCD DPA Space Example	819
9-25	UIO Direct P2P to Interleaved HDM	833

EVALUATION COPY

Figures

10-1	PkgC Entry Flow Initiated by Device - Example	853
10-2	PkgC Entry Flows for CXL Type 3 Device - Example	854
10-3	PkgC Exit Flows - Triggered by Device Access to System Memory	855
10-4	PkgC Exit Flows - Execution Required by Processor.....	856
10-5	CXL Link PM Phase 1 for 256B Flit Mode	857
10-6	CXL Link PM Phase 1 for 68B Flit Mode.....	858
10-7	CXL Link PM Phase 2	859
10-8	CXL PM Phase 3.....	861
11-1	68B Flit: CXL.cachemem IDE Showing Aggregation of 5 Flits.....	867
11-2	68B Flit: CXL.cachemem IDE Showing Aggregation across 5 Flits where One Flit Contains MAC Header in Slot 0	868
11-3	68B Flit: More-detailed View of a 5-Flit MAC Epoch Example	869
11-4	68B Flit: Mapping of AAD Bytes for the Example Shown in Figure 11-3	870
11-5	256B Flit: Handling of Slot 0 when it Carries H8	871
11-6	256B Flit: Handling of Slot 0 when it Does Not Carry H8.....	871
11-7	256B Flit: Handling of Slot 15	872
11-8	Mapping of Integrity-only Protected Bits to AAD - Case 1	872
11-9	Mapping of Integrity-only Protected Bits to AAD - Case 2	872
11-10	Mapping of Integrity-only Protected Bits to AAD - Case 3	873
11-11	Standard 256B Flit - Mapping to AAD and P bits when Slot 0 carries H8.....	873
11-12	Standard 256B Flit - Mapping to AAD and P bits when Slot 0 Does Not Carry H8	874
11-13	Latency-Optimized 256B Flit - Mapping to AAD and P Bits when Slot 0 Carries H8..	875
11-14	Latency-Optimized 256B Flit - Mapping to AAD and P Bits when Slot 0 Does Not Carry H8	876
11-15	Inclusion of the PCRC Mechanism in the AES-GCM Advanced Encryption Function ..	877
11-16	Inclusion of the PCRC Mechanism in the AES-GCM Advanced Decryption Function..	877
11-17	MAC Epochs and MAC Transmission in Case of Back-to-Back Traffic (a) Earliest MAC Header Transmit (b) Latest MAC Header Transmit in the Presence of Multi-Data Header	880
11-18	Example of MAC Header Being Received in the First Flit of the Current MAC Epoch.	881
11-19	Early Termination and Transmission of Truncated MAC Flit.....	883
11-20	CXL.cachemem IDE Transmission with Truncated MAC Flit.....	883
11-21	Link Idle Case after Transmission of Aggregation Flit Count Number of Flits	884
11-22	Various Interface Standards that are Referenced by this Specification and their Lineage	888
11-23	Active and Pending Key State Transitions	899
11-24	Reference Architecture	906
11-25	CMA/SPDM, CXL IDE, and CXL TSP Message Relationship	907
11-26	CMA/SPDM Sessions Creation Sequence	908
11-27	Optional Explicit In-band TE State Change Architecture	912
11-28	CKID-based Memory Encryption Utilizing CKID Base	917
11-29	Range-based Memory Encryption	919
11-30	Target TSP Security States	920
12-1	RCH Downstream Port Detects Error	958
12-2	RCD Upstream Port Detects Error	959
12-3	RCD RCiEP Detects Error	960
12-4	CXL Memory Error Reporting Enhancements	963
14-1	Example Test Topology	976
14-2	Example SHDA Topology	977
14-3	Example Single Host, Switch Attached, SLD EP (SHSW) Topology	977

14-4	Example SHSW-FM Topology	978
14-5	Example DHSW-FM Topology	979
14-6	Example DHSW-FM-MLD Topology	980
14-7	Example Topology for Two PBR Switches	981
14-8	Example Topology for a PBR Switch and an HBR Switch	982
14-9	Representation of False Sharing between Cores (on Host) and CXL Devices	983
14-10	Flow Chart of Algorithm 1a	984
14-11	Flow Chart of Algorithm 1b	985
14-12	Execute Phase for Algorithm 2	986
14-13	Compliance Testing Topology for an HBR Switch with a Single Host	1014
14-14	Compliance Testing Topology for an HBR Switch with Two Hosts	1014
14-15	Compliance Testing Topology for Two PBR Switches.....	1015
14-16	Compliance Testing Topology for a PBR Switch and an HBR Switch	1016
14-17	LTSSM Hot Reset Propagation to SLDs (PBR+HBR Switch)	1019
14-18	Secondary Bus Reset (SBR) Hot Reset Propagation to SLDs (PBR+HBR Switch) ...	1022
14-19	PCIe DVSEC for Test Capability.....	1131
A-1	Profile D - Giant Cache Model	1151

Tables

1-1	Terminology/Acronyms	48
1-2	Reference Documents	59
2-1	LD-ID Link Local TLP Prefix	76
2-2	MLD PCIe Registers	76
3-1	CXL Power Management Messages - Data Payload Field Definitions	86
3-2	PMREQ Field Definitions	88
3-3	Optional PCIe Features Required for CXL	90
3-4	PBR TLP Header (PTH) Format	93
3-5	NOP-TLP Header Format	93
3-6	Local Prefix Header Format	93
3-7	VendPrefixL0 on Non-MLD Edge HBR Links	94
3-8	PBR VDM	95
3-9	CXL Fabric Vendor Defined Messages	96
3-10	GAM VDM Payload	102
3-11	RTUpdate VDM Payload	103
3-12	CXL.cache Channel Crediting Summary	106
3-13	CXL.cache - D2H Request Fields.....	106
3-14	Non Temporal Encodings	107
3-15	CXL.cache - D2H Response Fields.....	107
3-16	CXL.cache - D2H Data Header Fields	108
3-17	CXL.cache – H2D Request Fields	108
3-18	CXL.cache - H2D Response Fields	109
3-19	RSP_PRE Encodings	109
3-20	Cache State Encoding for H2D Response	110
3-21	CXL.cache - H2D Data Header Fields	110
3-22	CXL.cache – Device to Host Requests	117
3-23	D2H Request (Targeting Non Device-attached Memory) Supported H2D Responses	121
3-24	D2H Request (Targeting Device-attached Memory) Supported Responses	122

EVALUATION COPY

Tables

3-25	D2H Response Encodings	123
3-26	CXL.cache – Mapping of H2D Requests to D2H Responses	125
3-27	H2D Response Opcode Encodings	125
3-28	Allowed Opcodes for D2H Requests per Buried Cache State	131
3-29	Impact of DevLoad Indication on Host/Peer Request Rate Throttling	137
3-30	Recommended Host/Peer Adjustment to Request Rate Throttling	138
3-31	Factors for Determining IntLoad	139
3-32	Additional Factors for Determining DevLoad in MLDs.....	144
3-33	Additional Factors for Determining DevLoad in MLDs/GFDs	146
3-34	M2S Request Fields	148
3-35	M2S Req Memory Opcodes	149
3-36	Metadata Field Definition	150
3-37	Meta0-State Value Definition (HDM-D/HDM-DB Devices)	150
3-38	Snoop Type Definition	151
3-39	M2S Req Usage	151
3-40	M2S Rwd Fields.....	152
3-41	M2S Rwd Memory Opcodes	153
3-42	M2S Rwd Usage	154
3-43	Rwd Trailers	154
3-44	M2S BIRsp Fields	155
3-45	M2S BIRsp Memory Opcodes	155
3-46	S2M BISnp Fields.....	156
3-47	S2M BISnp Opcodes	156
3-48	Block (Blk) Enable Encoding in Address[7:6]	157
3-49	S2M NDR Fields	157
3-50	S2M NDR Opcodes	157
3-51	DevLoad Definition	158
3-52	S2M DRS Fields	158
3-53	S2M DRS Opcodes	159
3-54	DRS Trailers	159
3-55	Allowed Opcodes for HDM-D/HDM-DB Req and Rwd Messages per Buried Cache State	161
3-56	Upstream Ordering Summary	162
3-57	Downstream Ordering Summary	163
3-58	Device In-Out Ordering Summary	164
3-59	Host In-Out Ordering Summary	165
4-1	CXL.cachemem Link Layer Flit Header Definition	193
4-2	Type Encoding.....	193
4-3	Legal Values of Sz and BE Fields	194
4-4	CXL.cachemem Credit Return Encodings	195
4-5	ReqCrd/DataCrd/RspCrd Channel Mapping	195
4-6	Slot Format Field Encoding	196
4-7	H2D/M2S Slot Formats	196
4-8	D2H/S2M Slot Formats	197
4-9	CXL.cachemem Link Layer Control Types	210
4-10	CXL.cachemem Link Layer Control Details	211
4-11	Control Flits and Their Effect on Sender and Receiver States	219
4-12	Local Retry State Transitions	221
4-13	Remote Retry State Transition	223
4-14	256B G-Slot Formats	228

EVALUATION COPY

Tables

4-15	256B H-Slot Formats	229
4-16	256B HS-Slot Formats	230
4-17	Trailer Size and Modes Supported per Channel	246
4-18	128B Group Maximum Message Rates	248
4-19	Credit Returned Encoding	250
4-20	256B Flit Mode Control Message Details	253
5-1	vLSM States Maintained per Link Layer Interface	259
5-2	ARB/MUX Multiple vLSM Resolution Table	260
5-3	ARB/MUX State Transition Table	261
5-4	vLSM State Resolution after Status Exchange	264
5-5	ALMP Byte 1 Encoding	280
5-6	ALMP Byte 2 and 3 Encodings for vLSM ALMP	281
5-7	ALMP Byte 2 and 3 Encodings for L0p Negotiation ALMP	281
6-1	Flex Bus.CXL Link Speeds and Widths for Normal and Degraded Mode	284
6-2	Flex Bus.CXL Protocol IDs	285
6-3	Protocol ID Framing Errors	293
6-4	256B Flit Mode vs. 68B Flit Mode Operation	293
6-5	256B Flit Header	295
6-6	Flit Type[1:0]	295
6-7	Latency-Optimized Flit Processing for CRC Scenarios	298
6-8	Byte Mapping for Input to PCIe 8B CRC Generation	300
6-9	Modified TS1/TS2 Ordered Set for Flex Bus Mode Negotiation	302
6-10	Additional Information on Symbols 8-9 of Modified TS1/TS2 Ordered Set	303
6-11	Additional Information on Symbols 12-14 of Modified TS1/TS2 Ordered Sets	303
6-12	VH vs. RCD Link Training Resolution	307
6-13	Flit Mode and VH Negotiation	309
6-14	Rules of Enable Low-latency Mode Features	311
6-15	Sync Header Bypass Applicability and Ordered Set Insertion Rate	311
7-1	CXL Switch Sideband Signal Requirements	323
7-2	MLD Type 1 Configuration Space Header	330
7-3	MLD PCI-compatible Configuration Registers	330
7-4	MLD PCIe Capability Structure	330
7-5	MLD Secondary PCIe Capability Structure	333
7-6	MLD Physical Layer 16.0 GT/s Extended Capability	333
7-7	MLD Physical Layer 32.0 GT/s Extended Capability	334
7-8	MLD Lane Margining at the Receiver Extended Capability	334
7-9	MLD ACS Extended Capability	335
7-10	MLD Advanced Error Reporting Extended Capability	336
7-11	MLD PPB DPC Extended Capability	337
7-12	CXL Switch Message Management	340
7-13	CXL Switch RAS	340
7-14	CCI Message Format	342
7-15	FM API Command Sets	349
7-16	Identify Switch Device Response Payload	350
7-17	Get Physical Port State Request Payload	350
7-18	Get Physical Port State Response Payload	351
7-19	Get Physical Port State Port Information Block Format	351
7-20	Physical Port Control Request Payload	353
7-21	Send PPB CXL.io Configuration Request Input Payload	353

EVALUATION COPY

Tables

7-22	Send PPB CXL.io Configuration Request Output Payload	353
7-23	Get Domain Validation SV State Response Payload	354
7-24	Set Domain Validation SV Request Payload.....	354
7-25	Get VCS Domain Validation SV State Request Payload	355
7-26	Get VCS Domain Validation SV State Response Payload	355
7-27	Get Domain Validation SV Request Payload	355
7-28	Get Domain Validation SV Response Payload	355
7-29	Virtual Switch Command Set Requirements	355
7-30	Get Virtual CXL Switch Info Request Payload	356
7-31	Get Virtual CXL Switch Info Response Payload	356
7-32	Get Virtual CXL Switch Info VCS Information Block Format	357
7-33	Bind vPPB Request Payload	358
7-34	Unbind vPPB Request Payload.....	359
7-35	Generate AER Event Request Payload	359
7-36	MLD Port Command Set Requirements.....	360
7-37	Tunnel Management Command Request Payload	362
7-38	Tunnel Management Command Response Payload	362
7-39	Send LD CXL.io Configuration Request Payload	363
7-40	Send LD CXL.io Configuration Response Payload	363
7-41	Send LD CXL.io Memory Request Payload.....	364
7-42	Send LD CXL.io Memory Request Response Payload.....	364
7-43	MLD Component Command Set Requirements	364
7-44	Get LD Info Response Payload	365
7-45	Get LD Allocations Request Payload	365
7-46	Get LD Allocations Response Payload	366
7-47	LD Allocations List Format	366
7-48	Set LD Allocations Request Payload	367
7-49	Set LD Allocations Response Payload	367
7-50	Payload for Get QoS Control Response, Set QoS Control Request, and Set QoS Control Response	367
7-51	Get QoS Status Response Payload.....	369
7-52	Payload for Get QoS Allocated BW Request.....	369
7-53	Payload for Get QoS Allocated BW Response	369
7-54	Payload for Set QoS Allocated BW Request, and Set QoS Allocated BW Response ..	370
7-55	Payload for Get QoS BW Limit Request	370
7-56	Payload for Get QoS BW Limit Response	370
7-57	Payload for Set QoS BW Limit Request, and Set QoS BW Limit Response	371
7-58	Get Multi-Headed Info Request Payload	372
7-59	Get Multi-Headed Info Response Payload	372
7-60	Get Head Info Request Payload.....	372
7-61	Get Head Info Response Payload.....	373
7-62	Get Head Info Head Information Block Format	373
7-63	Get DCD Info Response Payload	374
7-64	Get Host DC Region Configuration Request Payload	375
7-65	Get Host DC Region Configuration Response Payload	375
7-66	DC Region Configuration	376
7-67	Set DC Region Configuration Request and Response Payload	377
7-68	Get DC Region Extent Lists Request Payload	378
7-69	Get DC Region Extent Lists Response Payload	378
7-70	Initiate Dynamic Capacity Add Request Payload	380

EVALUATION COPY

Tables

7-71	Initiate Dynamic Capacity Release Request Payload	381
7-72	Dynamic Capacity Add Reference Request Payload	382
7-73	Dynamic Capacity Remove Reference Request Payload	383
7-74	Dynamic Capacity List Tags Request Payload	383
7-75	Dynamic Capacity List Tags Response Payload	383
7-76	Dynamic Capacity Tag Information	384
7-77	Physical Switch Events Record Format	385
7-78	Virtual CXL Switch Event Record Format	386
7-79	MLD Port Event Records Payload	387
7-80	Differences between LD-FAM and G-FAM	393
7-81	Fabric Segment Size Table	396
7-82	Segment Table Intlv[3:0] Field Encoding	396
7-83	Segment Table Gran[3:0] Field Encoding	397
7-84	PBR Fabric Decoding and Routing, by Message Class	414
7-85	Optional Architected Dynamic Routing Modes	416
7-86	Summary of CacheID Field	420
7-87	Summary of HBR Switch Routing for CXL.cache Message Classes	420
7-88	Summary of PBR Switch Routing for CXL.cache Message Classes	421
7-89	Summary of LD-ID Field	421
7-90	Summary of BI-ID Field	422
7-91	Summary of HBR Switch Routing for CXL.mem Message Classes	422
7-92	Summary of PBR Switch Routing for CXL.mem Message Classes	423
7-93	HBR Switch Port Processing Table for CXL.io	424
7-94	HBR Switch Port Processing Table for CXL.cache	424
7-95	HBR Switch Port Processing Table for CXL.mem	425
7-96	PBR Switch Port Processing Table for CXL.io	426
7-97	PBR Switch Port Processing Table for CXL.cache	426
7-98	PBR Switch Port Processing Table for CXL.mem	427
7-99	ISL Type 1 Configuration Space Header	428
7-100	ISL PCI-compatible Configuration Space Header.....	428
7-101	ISL PCIe Capability Structure.....	428
7-102	ISL Secondary PCIe Extended Capability	430
7-103	ISL Physical Layer 16.0 GT/s Extended Capability.....	431
7-104	ISL Physical Layer 32.0 GT/s Extended Capability.....	431
7-105	ISL Physical Layer 64.0 GT/s Extended Capability.....	432
7-106	ISL Lane Margining at the Receiver Extended Capability	432
7-107	PBR Fabric .io Ordering Table, Non-UIO	434
7-108	PBR Fabric .io Ordering Table, UIO	434
7-109	Link Partner Info Payload	449
7-110	Far End Device Type Detection	451
7-111	Identify PBR Switch Response Payload	455
7-112	Fabric Crawl Out Request Payload	457
7-113	Fabric Crawl Out Response Payload	457
7-114	Get PBR Link Partner Info Request Payload.....	458
7-115	Get PBR Link Partner Info Response Payload.....	458
7-116	Link Partner Info Format	458
7-117	Get PID Target List Request Payload.....	459
7-118	Get PID Target List Response Payload.....	459
7-119	Target List Format	459
7-120	Configure PID Assignment Request Payload	460

EVALUATION COPY

Tables

7-121	PID Assignment	460
7-122	Get PID Binding Request Payload	460
7-123	Get PID Binding Response Payload	461
7-124	Configure PID Binding Request Payload.....	461
7-125	Get Table Descriptors Request Payload	462
7-126	Get Table Descriptors Response Payload	462
7-127	Get Table Descriptor Format	463
7-128	Get DRT Request Payload.....	463
7-129	Get DRT Response Payload	463
7-130	DRT Entry Format	464
7-131	Set DRT Request Payload.....	464
7-132	Get RGT Request Payload.....	465
7-133	Get RGT Response Payload	465
7-134	RGT Entry Format	465
7-135	Set RGT Request Payload.....	466
7-136	Get LDST/IDT Capabilities Request Payload	466
7-137	Get LDST/IDT Capabilities Response Payload	467
7-138	Set LDST/IDT Configuration Request Payload	468
7-139	Get LDST Segment Entries Request Payload	468
7-140	Get LDST Segment Entries Response Payload	469
7-141	LDST Segment Entry Format	469
7-142	Set LDST Segment Entries Request Payload	470
7-143	Get LDST IDT DPID Entries Request Payload.....	471
7-144	Get LDST IDT DPID Entries Response Payload.....	471
7-145	Set LDST IDT DPID Entries Request Payload.....	472
7-146	Get Completer ID-Based Re-Router Entries Request Payload.....	472
7-147	Get Completer ID-Based Re-Router Entries Response Payload.....	473
7-148	Completer ID-Based Re-Router Entry	473
7-149	Set Completer ID-Based Re-Router Entries Request Payload	474
7-150	Get LDST Access Vector Request Payload	474
7-151	Get LDST Access Vector Response Payload	474
7-152	LDST Access Vector	475
7-153	Get VCS LDST Access Vector Request Payload	475
7-154	Configure VCS LDST Access Request Payload.....	476
7-155	Identify GAE Request Payload.....	476
7-156	Identify GAE Response Payload	477
7-157	vPPB Global Memory Support Info	477
7-158	Get PID Interrupt Vector Request Payload	478
7-159	Get PID Interrupt Vector Response Payload	478
7-160	PID Interrupt Vector	478
7-161	Get PID Access Vectors Request Payload	479
7-162	Get PID Access Vectors Response Payload	479
7-163	PID Access Vector	479
7-164	Get FAST/IDT Capabilities Request Payload	480
7-165	Get FAST/IDT Capabilities Response Payload	480
7-166	vPPB PID List Entry Format	480
7-167	Set FAST/IDT Configuration Request Payload.....	481
7-168	Get FAST Segment Entries Request Payload	482
7-169	Get FAST Segment Entries Response Payload	482
7-170	FAST Segment Entry Format	482

EVALUATION COPY

Tables

7-171	Set FAST Segment Entries Request Payload.....	483
7-172	Get IDT DPID Entries Request Payload.....	484
7-173	Get IDT DPID Entries Response Payload.....	484
7-174	Set IDT DPID Entries Request Payload	485
7-175	Proxy GFD Management Command Request Payload.....	486
7-176	Proxy GFD Management Command Response Payload.....	486
7-177	Get Proxy Thread Status Request Payload	486
7-178	Get Proxy Thread Status Response Payload	487
7-179	Cancel Proxy Thread Request Payload.....	487
7-180	Cancel Proxy Thread Response Payload.....	487
7-181	Identify VCS GAE Request Payload	488
7-182	Get VCS PID Access Vectors Request Payload	489
7-183	Configure VCS PID Access Request Payload	489
7-184	Get VendPrefixL0 State Request Payload	490
7-185	Get VendPrefixL0 State Response Payload	490
7-186	Set VendPrefixL0 State Request Payload	491
8-1	Register Attributes	492
8-2	CXL DVSEC ID Assignment	493
8-3	CXL DOE Type Assignment	494
8-4	PCIe DVSEC CXL Devices - Header	495
8-5	Non-CXL Function Map DVSEC - Header	507
8-6	CXL Extensions DVSEC for Ports - Header	510
8-7	GPF DVSEC for CXL Port - Header	515
8-8	GPF DVSEC for CXL Device - Header	516
8-9	Register Locator DVSEC - Header	518
8-10	Designated Vendor Specific Register Block Header	519
8-11	MLD DVSEC - Header	520
8-12	Coherent Device Attributes - Data Object Header	520
8-13	Read Entry Request	521
8-14	Read Entry Response.....	521
8-15	Memory Device PCIe Capabilities and Extended Capabilities	522
8-16	PCI Header - Class Code Register (Offset 09h) for Switch Mailbox CCI	522
8-17	CXL Memory Mapped Register Regions	523
8-18	RCH Downstream Port PCIe Capabilities and Extended Capabilities	525
8-19	RCD Upstream Port PCIe Capabilities and Extended Capabilities	528
8-20	PCIe DVSEC Header Register Settings for Flex Bus Port	529
8-21	CXL Subsystem Component Register Ranges	534
8-22	CXL_Capability_ID Assignment	535
8-23	CXL.cache and CXL.mem Architectural Register Discovery	536
8-24	CXL.cache and CXL.mem Architectural Register Header Example (Primary Range)	536
8-25	CXL.cache and CXL.mem Architectural Register Header Example (Any Extended Range)	536
8-26	Device Trust Level	550
8-27	CXL.mem Read Response - Error Cases	559
8-28	CXL Extended Security Structure Entry Count (Offset 00h)	569
8-29	Root Port n Security Policy Register (Offset 8*n-4)	569
8-30	Root Port n ID Register (Offset 8*n)	570
8-31	BAR Virtualization ACL Register Block Layout.....	593
8-32	CPMU Register Layout (Version=1)	594

EVALUATION COPY

Tables

8-33	Filter ID and Values	600
8-34	Command Return Codes	607
8-35	CXL Memory Device Capabilities Identifiers	610
8-36	Switch Mailbox CCI Capabilities Identifiers	611
8-37	Generic Component Command Opcodes	613
8-38	Identify Output Payload	615
8-39	Background Operation Status Output Payload	616
8-40	Get Response Message Limit Output Payload	616
8-41	Set Response Message Limit Input Payload.....	617
8-42	Set Response Message Limit Output Payload	617
8-43	Common Event Record Format	618
8-44	Component Identifier Format	620
8-45	General Media Event Record	621
8-46	DRAM Event Record	624
8-47	Memory Module Event Record	628
8-48	Memory Sparing Event Record	629
8-49	Vendor Specific Event Record	630
8-50	Dynamic Capacity Event Record	631
8-51	Dynamic Capacity Extent	632
8-52	Get Event Records Input Payload	633
8-53	Get Event Records Output Payload	634
8-54	Clear Event Records Input Payload	635
8-55	Get Event Interrupt Policy Output Payload.....	636
8-56	Set Event Interrupt Policy Input Payload	638
8-57	Payload for Get MCTP Event Interrupt Policy Output, Set MCTP Event Interrupt Policy Input, and Set MCTP Event Interrupt Policy Output	639
8-58	Event Notification Input Payload	640
8-59	Enhanced Event Notification Input Payload	641
8-60	GFD Notification Type Values	642
8-61	GFD Notification Type Triggers	643
8-62	Enhanced Event Notification Input Payload	645
8-63	Get GAM Buffer Response Payload.....	646
8-64	Set GAM Buffer Address Request Payload	646
8-65	Get FW Info Output Payload	647
8-66	Transfer FW Input Payload	650
8-67	Activate FW Input Payload	651
8-68	Get Timestamp Output Payload	652
8-69	Set Timestamp Input Payload	652
8-70	Get Supported Logs Output Payload	653
8-71	Get Supported Logs Supported Log Entry	653
8-72	Get Log Input Payload	654
8-73	Get Log Output Payload	654
8-74	CEL Output Payload.....	654
8-75	CEL Entry Structure	655
8-76	Component State Dump Log Population Methods and Triggers	656
8-77	Component State Dump Log Format	657
8-78	DDR5 Error Check Scrub (ECS) Log	658
8-79	Media Test Capability Log Output Payload	659
8-80	Media Test Capability Log Common Header	659
8-81	Media Test Capability Log Entry Structure	660

EVALUATION COPY

Tables

8-82	Media Test Results Short Log	662
8-83	Media Test Results Short Log Entry Common Header	662
8-84	Media Test Results Short Log Entry Structure	663
8-85	Media Test Results Long Log	664
8-86	Media Test Results Long Log Entry Common Header	664
8-87	Media Test Results Long Log Entry Structure	664
8-88	Error Signature	665
8-89	Get Log Capabilities Input Payload	667
8-90	Get Log Capabilities Output Payload	667
8-91	Clear Log Input Payload.....	668
8-92	Populate Log Input Payload	668
8-93	Get Supported Logs Sub-List Input Payload	669
8-94	Get Supported Logs Sub-List Output Payload	669
8-95	Get Supported Features Input Payload	670
8-96	Get Supported Features Output Payload.....	670
8-97	Get Supported Features Supported Feature Entry	670
8-98	Feature Attribute(s) Value after Reset	671
8-99	Get Feature Input Payload	672
8-100	Get Feature Output Payload	672
8-101	Set Feature Input Payload.....	674
8-102	Perform Maintenance Input Payload	675
8-103	sPPR Maintenance Input Payload	677
8-104	hPPR Maintenance Input Payload	677
8-105	Memory Sparing Input Payload	679
8-106	Device Built-in Test Input Payload	680
8-107	Test Parameters	680
8-108	Common Configuration Parameters for Media Test Subclass	681
8-109	Test Parameters Entry Media Test Subclass	681
8-110	Maintenance Operation: Classes, Subclasses, and Feature UUIDs	682
8-111	Common Maintenance Operation Feature Format	683
8-112	Supported Feature Entry for the sPPR Feature	684
8-113	sPPR Feature Readable Attributes	684
8-114	sPPR Feature Writable Attributes	685
8-115	Supported Feature Entry for the hPPR Feature	686
8-116	hPPR Feature Readable Attributes	686
8-117	hPPR Feature Writable Attributes	687
8-118	Supported Feature Entry for the Memory Sparing Feature	688
8-119	Memory Sparing Feature Readable Attributes	688
8-120	Memory Sparing Feature Writable Attributes	689
8-121	Identify PBR Component Response Payload	690
8-122	Claim Ownership Request Payload.....	691
8-123	Claim Ownership Response Payload.....	691
8-124	Read CDAT Request Payload.....	692
8-125	Read CDAT Response Payload.....	692
8-126	CXL Memory Device Command Opcodes	693
8-127	Identify Memory Device Output Payload	696
8-128	Get Partition Info Output Payload.....	698
8-129	Set Partition Info Input Payload	699
8-130	Get LSA Input Payload.....	699
8-131	Get LSA Output Payload.....	700

EVALUATION COPY

Tables

8-132	Set LSA Input Payload	700
8-133	Get Health Info Output Payload	701
8-134	Get Alert Configuration Output Payload	703
8-135	Set Alert Configuration Input Payload	705
8-136	Get Shutdown State Output Payload	706
8-137	Set Shutdown State Input Payload	706
8-138	Get Poison List Input Payload	708
8-139	Get Poison List Output Payload	708
8-140	Media Error Record	709
8-141	Inject Poison Input Payload	710
8-142	Clear Poison Input Payload	711
8-143	Get Scan Media Capabilities Input Payload	711
8-144	Get Scan Media Capabilities Output Payload	712
8-145	Scan Media Input Payload	713
8-146	Get Scan Media Results Output Payload	714
8-147	Media Operation Input Payload	716
8-148	DPA Range Format	717
8-149	Media Operations Classes and Subclasses	717
8-150	Discovery Operation-specific Arguments	717
8-151	Media Operations Output Payload – Discovery Operation	717
8-152	Supported Operations List Entries	718
8-153	Get Security State Output Payload	718
8-154	Set Passphrase Input Payload	719
8-155	Disable Passphrase Input Payload	720
8-156	Unlock Input Payload	720
8-157	Passphrase Secure Erase Input Payload	721
8-158	Security Send Input Payload	722
8-159	Security Receive Input Payload	723
8-160	Security Receive Output Payload	723
8-161	Get SLD QoS Control Output Payload and Set SLD QoS Control Input Payload	724
8-162	Get SLD QoS Status Output Payload	725
8-163	Get Dynamic Capacity Configuration Input Payload	725
8-164	Get Dynamic Capacity Configuration Output Payload	725
8-165	DC Region Configuration	726
8-166	Get Dynamic Capacity Extent List Input Payload	727
8-167	Get Dynamic Capacity Extent List Output Payload	727
8-168	Add Dynamic Capacity Response Input Payload	728
8-169	Updated Extent	728
8-170	Release Dynamic Capacity Input Payload	729
8-171	Identify GFD Response Payload	730
8-172	Get GFD Status Response Payload	732
8-173	Get GFD DC Region Configuration Request Payload	733
8-174	Get GFD DC Region Configuration Response Payload	734
8-175	GFD DC Region Configuration	734
8-176	Set GFD DC Region Configuration Request Payload	735
8-177	Get GFD DC Region Extent Lists Request Payload	736
8-178	Get GFD DC Region Extent Lists Response Payload	736
8-179	Get GFD DMP Configuration Request Payload	737
8-180	Get GFD DMP Configuration Response Payload	737
8-181	GFD DMP Configuration	737

EVALUATION COPY

Tables

8-182	Set GFD DMP Configuration Request Payload	738
8-183	GFD Dynamic Capacity Add Request Payload	741
8-184	GFD Dynamic Capacity Add Response Payload	741
8-185	Initiate Dynamic Capacity Release Request Payload	743
8-186	GFD Dynamic Capacity Release Response Payload	744
8-187	GFD Dynamic Capacity Add Reference Request Payload	745
8-188	GFD Dynamic Capacity Remove Reference Request Payload	745
8-189	GFD Dynamic Capacity List Tags Request Payload	746
8-190	GFD Dynamic Capacity List Tags Response Payload	746
8-191	GFD Dynamic Capacity Tag Information	746
8-192	Get GFD SAT Entry Request Payload	747
8-193	Get GFD SAT Entry Response Payload	747
8-194	GFD SAT Entry Format	747
8-195	Set GFD SAT Entry Request Payload	748
8-196	GFD SAT Update Format	749
8-197	QOS Payload for Get GFD QoS Control Response, Set GFD QoS Control Request, and Set GFD QoS Control Response	749
8-198	Get GFD QoS Status Response Payload	750
8-199	Payload for Get GFD QoS BW Limit Request	751
8-200	Payload for Get GFD QoS BW Limit Response	751
8-201	Payload for Set GFD BW Limit Request and Set GFD QoS BW Limit Response	752
8-202	Get GDT Configuration Request Payload	752
8-203	Get GDT Configuration Response Payload	753
8-204	GDT Entry Format	753
8-205	Set GDT Configuration Request Payload	754
8-206	Supported Feature Entry for the Device Patrol Scrub Control Feature	755
8-207	Device Patrol Scrub Control Feature Readable Attributes	756
8-208	Device Patrol Scrub Control Feature Writable Attributes	756
8-209	Supported Feature Entry for the DDR5 ECS Control Feature	757
8-210	DDR5 ECS Control Feature Readable Attributes	758
8-211	DDR5 ECS Control Feature Writable Attributes	759
8-212	Supported Feature Entry for the Advanced Programmable Corrected Volatile Memory Error Threshold Feature	759
8-213	Advanced Programmable Corrected Volatile Memory Error Threshold Feature Readable Attributes	760
8-214	Advanced Programmable Corrected Volatile Memory Error Threshold Feature Writable Attributes	764
8-215	CXL FM API Command Opcodes	767
9-1	Event Sequencing for Reset and Sx Flows	773
9-2	CXL Switch Behavior Message Aggregation Rules	773
9-3	GPF Energy Calculation Example	784
9-4	Memory Decode Rules in Presence of One CPU/Two Flex Bus Links	793
9-5	Memory Decode Rules in Presence of Two CPU/Two Flex Bus Links	794
9-6	12-Way Device-level Interleave at IGB	810
9-7	6-Way Device-level Interleave at IGB	810
9-8	3-Way Device-level Interleave at IGB	810
9-9	Label Index Block Layout	813
9-10	Region Label Layout	815
9-11	Namespace Label Layout	816
9-12	Vendor Specific Label Layout	817

EVALUATION COPY

Tables

9-13	Downstream Port Handling of BISnp	824
9-14	Downstream Port Handling of BIRsp	824
9-15	CXL Type 2 Device Behavior in Fallback Operation Mode	828
9-16	Downstream Port Handling of D2H Request Messages	830
9-17	Downstream Port Handling of H2D Response Message and H2D Request Message .	830
9-18	Handling of UIO Accesses	834
9-19	CEDT Header	837
9-20	CEDT Structure Types	837
9-21	CHBS Structure	838
9-22	CFMWS Structure	838
9-23	CXIMS Structure.....	841
9-24	RDPAS Structure.....	842
9-25	CSDS Structure	843
9-26	_OSC Capabilities Buffer DWORDS	843
9-27	Interpretation of CXL _OSC Support Field.....	844
9-28	Interpretation of CXL _OSC Control Field, Passed in via Arg3	844
9-29	Interpretation of CXL _OSC Control Field, Returned Value	845
9-30	_DSM Definitions for CXL Root Device	846
9-31	_DSM for Retrieving QTG, Inputs, and Outputs	848
10-1	Runtime-Control - CXL vs. PCIe Control Methodologies	851
10-2	PMReq(), PMRsp(), and PMGo() Encoding	853
11-1	Mapping of PCIe IDE to CXL.io	866
11-2	CXL_IDE_KM Request Header	890
11-3	CXL_IDE_KM Successful Response Header	890
11-4	CXL_IDE_KM Generic Error Conditions	891
11-5	CXL_QUERY Request	891
11-6	CXL_QUERY Processing Errors	891
11-7	Successful CXL_QUERY_RESP Response.....	891
11-8	CXL_KEY_PROG Request.....	893
11-9	CXL_KEY_PROG Processing Errors	894
11-10	CXL_KP_ACK Response.....	894
11-11	CXL_K_SET_GO Request.....	896
11-12	CXL_K_SET_GO Error Conditions	896
11-13	CXL_K_SET_STOP Request.....	897
11-14	CXL_K_SET_STOP Error Conditions	897
11-15	CXL_K_GOSTOP_ACK Response	897
11-16	CXL_GETKEY Request.....	898
11-17	CXL_GETKEY Processing Error	898
11-18	CXL_GETKEY_ACK Response	899
11-19	Security Threats and Mitigations	905
11-20	Target Behavior for Implicit TE State Changes	910
11-21	Target Behavior for Explicit In-band TE State Changes	913
11-22	Target Behavior for Write Access Control	913
11-23	Target Behavior for Read Access Control	914
11-24	Target Behavior for Invalid CKID Ranges	918
11-25	Target Behavior for Verifying CKID Type	918
11-26	TSP Request Overview	928
11-27	TSP Response Overview	929
11-28	TSP Request Response and CMA/SPDM Sessions	930
11-29	Get Target TSP Version.....	931

EVALUATION COPY

Tables

11-30	Get Target TSP Version Response.....	931
11-31	Get Target Capabilities	932
11-32	Get Target Capabilities Response	932
11-33	Explicit In-band TE State Granularity Entry.....	935
11-34	Set Target Configuration	935
11-35	Set Target Configuration Response.....	938
11-36	Get Target Configuration.....	938
11-37	Get Target Configuration Response	939
11-38	Get Target Configuration Report	941
11-39	Get Target Configuration Report Response	941
11-40	TSP Report	942
11-41	Lock Target Configuration	943
11-42	Lock Target Configuration Response	943
11-43	Memory Range	944
11-44	Set Target TE State.....	945
11-45	Set Target TE State Response.....	945
11-46	Set Target CKID Specific Key.....	946
11-47	Set Target CKID Specific Key Response.....	946
11-48	Set Target CKID Random Key	947
11-49	Set Target CKID Random Key Response.....	948
11-50	Clear Target CKID Key.....	949
11-51	Clear Target CKID Key Response.....	949
11-52	Set Target Range Specific Key	950
11-53	Set Target Range Specific Key Response	950
11-54	Set Target Range Random Key	951
11-55	Set Target Range Random Key Response	952
11-56	Clear Target Range Key	953
11-57	Clear Target Range Key Response	953
11-58	Delayed Response.....	953
11-59	Check Target Delayed Completion	954
11-60	Get Target TE State Change Completion Response	954
11-61	Error Response.....	955
11-62	Error Response — Error Code, Error Data, Extended Error Data	955
12-1	CXL RAS Features	956
12-2	Device-specific Error Reporting and Nomenclature Guidelines	962
13-1	CXL Performance Attributes	969
13-2	Recommended Latency Targets for Selected CXL Transactions	970
13-3	Recommended Maximum Link Layer Latency Targets	970
13-4	CPMU Counter Units	971
13-5	Events under CXL Vendor ID	972
14-1	CRC Error Injection RETRY_PHY_REINIT: Cache CRC Injection Request	992
14-2	CRC Error Injection RETRY_ABORT: Cache CRC Injection Request	993
14-3	Link Initialization Resolution Table	1007
14-4	Hot Add Link Initialization Resolution Table	1008
14-5	Inject MAC Delay Setup	1073
14-6	Inject Unexpected MAC Setup.....	1074
14-7	CXL.io Poison Inject from Device: I/O Poison Injection Request.....	1078
14-8	CXL.io Poison Inject from Device: Multi-Write Streaming Request	1078
14-9	CXL.cache Poison Inject from Device: Cache Poison Injection Request.....	1079

EVALUATION COPY

Tables

14-10	CXL.cache Poison Inject from Device: Multi-Write Streaming Request	1080
14-11	CXL.cache CRC Inject from Device: Cache CRC Injection Request	1081
14-12	CXL.cache CRC Inject from Device: Multi-Write Streaming Request	1081
14-13	CXL.mem Poison Injection: Mem-Poison Injection Request	1082
14-14	CXL.mem CRC Injection: MEM CRC Injection Request	1083
14-15	Flow Control Injection: Flow Control Injection Request.....	1083
14-16	Flow Control Injection: Multi-Write Streaming Request	1084
14-17	Unexpected Completion Injection: Unexpected Completion Injection Request.....	1085
14-18	Unexpected Completion Injection: Multi-Write Streaming Request	1085
14-19	Completion Timeout Injection: Completion Timeout Injection Request.....	1086
14-20	Completion Timeout Injection: Multi-Write Streaming Request	1086
14-21	Memory Error Injection and Logging: Poison Injection Request.....	1087
14-22	Memory Error Injection and Logging: Multi-Write Streaming Request	1087
14-23	CXL.io Viral Inject from Device: I/O Viral Injection Request	1088
14-24	CXL.io Viral Inject from Device: Multi-Write Streaming Request	1089
14-25	CXL.cache Viral Inject from Device: Cache Viral Injection Request	1090
14-26	CXL.cache Viral Inject from Device: Multi-Write Streaming Request	1090
14-27	Register 1: CXL.cachemem LinkLayerErrorInjection	1104
14-28	Register 2: CXL.io LinkLayer Error Injection.....	1105
14-29	Register 3: Flex Bus LogPHY Error Injections	1105
14-30	CXL.io Poison Injection from Device to Host: I/O Poison Injection Request.....	1106
14-31	CXL.io Poison Injection from Device to Host: Multi-Write Streaming Request.....	1106
14-32	Device to Host Poison Injection: Cache Poison Injection Request	1107
14-33	Device to Host Poison Injection: Multi-Write Streaming Request	1108
14-34	Host to Device Poison Injection: Cache Poison Injection Request	1109
14-35	Device to Host CRC Injection: Cache Poison Injection Request	1109
14-36	Host to Device CRC Injection: Cache Poison Injection Request	1110
14-37	DVSEC Registers.....	1131
14-38	DVSEC CXL Test Lock (Offset 0Ah)	1131
14-39	DVSEC CXL Test Configuration Base Low (Offset 14h)	1132
14-40	DVSEC CXL Test Configuration Base High (Offset 18h)	1132
14-41	Register 9: ErrorLog1 (Offset 40h)	1132
14-42	Register 10: ErrorLog2 (Offset 48h)	1132
14-43	Register 11: ErrorLog3 (Offset 50h)	1132
14-44	Register 12: EventCtrl (Offset 60h)	1133
14-45	Register 13: EventCount (Offset 68h)	1133
14-46	Compliance Mode – Data Object Header.....	1134
14-47	Compliance Mode Return Values	1134
14-48	Compliance Mode Availability Request.....	1134
14-49	Compliance Mode Availability Response.....	1134
14-50	Compliance Options Value Descriptions	1135
14-51	Compliance Mode Status.....	1135
14-52	Compliance Mode Status Response.....	1136
14-53	Compliance Mode Halt All	1136
14-54	Compliance Mode Halt All Response.....	1136
14-55	Enable Multiple Write Streaming Algorithm on the Device	1136
14-56	Compliance Mode Multiple Write Streaming Response	1137
14-57	Enable Producer-Consumer Algorithm on the Device	1137
14-58	Compliance Mode Producer-Consumer Response	1138
14-59	Enable Algorithm 1b, Write Streaming with Bogus Writes	1138

EVALUATION COPY

Tables

14-60	Algorithm1b Response	1139
14-61	Enable Poison Injection into	1139
14-62	Poison Injection Response.....	1139
14-63	Enable CRC Error into Traffic	1140
14-64	CRC Injection Response.....	1140
14-65	Enable Flow Control Injection	1140
14-66	Flow Control Injection Response	1140
14-67	Enable Cache Flush Injection	1141
14-68	Cache Flush Injection Response	1141
14-69	MAC Delay Injection	1141
14-70	MAC Delay Response.....	1141
14-71	Unexpected MAC Injection	1142
14-72	Unexpected MAC Injection Response	1142
14-73	Enable Viral Injection	1142
14-74	Flow Control Injection Response	1143
14-75	Inject ALMP Request	1143
14-76	Inject ALMP Response	1143
14-77	Ignore Received ALMP Request	1144
14-78	Ignore Received ALMP Response	1144
14-79	Inject Bit Error in Flit Request.....	1144
14-80	Inject Bit Error in Flit Response.....	1144
14-81	Memory Device Media Poison Injection Request	1145
14-82	Memory Device Media Poison Injection Response	1145
14-83	Memory Device LSA Poison Injection Request	1145
14-85	Inject Memory Device Health Enable Memory Device Health Injection	1146
14-84	Memory Device LSA Poison Injection Response	1146
14-86	Device Health Injection Response.....	1147
A-1	Accelerator Usage Taxonomy.....	1148
C-1	Field Encoding Abbreviations	1153
C-2	Optional Codes	1154
C-3	HDM-D/HDM-DB Memory Requests	1155
C-4	HDM-D Request Forward Sub-table	1159
C-5	HDM-DB BISnp Flow	1161
C-6	HDM-H Memory Request	1163
C-7	HDM-D/HDM-DB Memory RwD	1165
C-8	HDM-H Memory RwD	1166

§ §

Revision History

Revision	Description	Date
3.1	<ul style="list-style-type: none"> • General spec updates <ul style="list-style-type: none"> — Merged CXL 3.0 errata (G3-G23) — Direct P2P CXL.mem for accelerators — Extended Metadata — General typo, grammar, punctuation, and formatting fixes — Changed “R/requestor” to “R/requester” to align with PCIe Base Specification • Added terms and abbreviations to Table 1-1 • Chapter 3.0, “CXL Transaction Layer” and Chapter 4.0, “CXL Link Layers” <ul style="list-style-type: none"> — General clarifications/cleanup • Chapter 6.0, “Flex Bus Physical Layer” <ul style="list-style-type: none"> — Added clarification on CRC corruption for CXL.cachemem viral/late poison and CXL.io nullify/poison • Chapter 7.0, “Switching” <ul style="list-style-type: none"> — CXL PBR fabric decoding and routing defined — CXL PBR fabric FM APIs defined — CXL fabric switch reset flows defined — CXL fabric deadlock avoidance rules defined — UIO direct peer to peer support in CXL fabric defined — GFD details refined — Global Integrated Memory concept defined • Chapter 8.0, “Control and Status Registers” and Chapter 9.0, “Reset, Initialization, Configuration, and Manageability” <ul style="list-style-type: none"> — Addressed ambiguities in CXL Header log definition — Clarified device handling of unrecognized CCI input payload size and feature input payload size — Defined the ability to abort a background operation — Expanded visibility and control over CXL memory device RAS – Memory sparing, media testing, memory scrub and DDR5 ECS — Improved visibility into CXL memory device errors – Advanced programmable corrected volatile error thresholds, added enums to describe detailed memory error causes, leveraged DMTF PLDM standard-based FRU/sub-FRU identification (Component ID field) — Ability to sanitize and zero media via media operations command — Support for setting up one writer-multiple reader type memory sharing • Section 11.5, “CXL Trusted Execution Environments Security Protocol (TSP)” <ul style="list-style-type: none"> — Trusted Execution Environment Security Protocol – Addition of basic security threat model, behaviors and interfaces for utilizing direct attached memory devices with confidential computing scenarios. This addition defines secure CMA/SPDM request and response payloads to retrieve the device’s security capabilities, configure security features on the device, and lock the device to prevent runtime configuration changes that could alter the device’s behavior or leak trusted data. These optional TSP security features can be utilized with CXL IDE and/or PCIe TDISP but are not dependent on either. • Chapter 14.0, “CXL Compliance Testing” <ul style="list-style-type: none"> — Updated test prerequisites and dependencies for uniformity — Moved Test 14.6.1.9 to its correct location at 14.6.13 — ECN fix for Test 14.7.5.3 	August 7, 2023

EVALUATION COPY

Revision	Description	Date
3.0	<ul style="list-style-type: none"> • General spec updates <ul style="list-style-type: none"> — 256B Flit mode updates — Back-Invalidate updates — Multiple CXL.cache devices per VCS — Fabric — Intra-VH P2P using UIO — Memory sharing — Merged CXL 2.0 ECNs and errata — Scrubbed terminology removing references to CXL 1.1 device/host, CXL 2.0 device/switch/host etc. and replaced with feature-specific link mode terminology (VH, RCD, eRCD, etc.) — Removed Symmetric Memory • Major additions on the SW side <ul style="list-style-type: none"> — Multi-headed MLD configuration — Dynamic Capacity Device (DCD) — Performance Monitoring • Chapter 7.0, "Switching" <ul style="list-style-type: none"> — G-FAM architecture definition completed — PBR switch SW view defined at a high level — PBR switch routing of CXL.mem/CXL.cache defined — PBR TLP header defined to route CXL.io traffic through the PBR fabric — DCD management architecture defined • Chapter 4.0, "CXL Link Layers" <ul style="list-style-type: none"> — Updates to 256B Packing Rules — Update BI-ID from 8 to 12 bits — Added clarification and examples for Late-Poison/Viral in 256B flit • Chapter 3.0, "CXL Transaction Layer" <ul style="list-style-type: none"> — Added PBR TLP header definition for CXL.io — Cleanup on HDM vs Device type terms (HDM-H, HDM-D, HDM-DB) — Update BI-ID from 8 to 12 bits — Added examples for BISnp • Chapter 14.0, "CXL Compliance Testing" <ul style="list-style-type: none"> — Re-ordered and organized link layer initialization tests to 68B and 256B modes — Updated switch tests to include routing types (HBR, PBR) — Added security tests to differentiate between 68B and 256B modes — Added Timeout and Isolation capability tests — Corrected DOE response for CXL.mem poison injection requests • Chapter 5.0, "CXL ARB/MUX" <ul style="list-style-type: none"> — Corrections and implementation notes on the different timers for PM/L0p handshakes — Implementation note clarifying there is a common ALMP for both PCI-PM and ASPM handshakes — Added clarity on Physical Layer LTSSM Recovery transitions vs Active to Retrain arc for vLSM • Chapter 6.0, "Flex Bus Physical Layer" <ul style="list-style-type: none"> — Corrections and updates to 6B CRC scheme as well as updated description on how to generate 6B CRC code from 8B CRC — Attached the generator matrix and system Verilog reference code for 6B CRC generation using the two methods described in the text — Flit Type encoding updates — Support for PBR flit negotiation 	August 1, 2022

EVALUATION COPY

Revision	Description	Date
3.0	<ul style="list-style-type: none"> • Updates to Chapter 7.0, "Switching", Chapter 8.0, "Control and Status Registers", Chapter 12.0, "Reliability, Availability, and Serviceability", and Chapter 13.0, "Performance Considerations" <ul style="list-style-type: none"> — Performance Monitor interface — BI registers, discovery and configuration — Cache ID registers, discovery configuration — Create more room for CXL.cache/CXL.mem registers, raised maximum HDM decoders count for switches and RP — FM API over mailbox interface — Merged Features ECN and maintenance ECN • CXL 3.0 IDE updates for 256B Flit mode • 256B Flit Slot Packing Rules and Credit flow control extensively updated • First inclusion for Port Based Routing (PBR) Switching architecture including message definition and new slot packing • Physical Layer: NOP hint optimization added, link training resolution table for CXL 3.0, clean up of nomenclature • ARB/MUX: Corner case scenarios around L0p and Recovery and EIOSQ before ACK • Compliance Mode DOE is now required (was optional) • Compliance DVSEC interface is removed • Test 14.6.5 test updated, now requires Analyzer, and comprehends a retimer • Test 14.8.3 invalid verify step removed • Test 14.11.3.11 bug fix, test title and pass criteria are updated • Test 14.11.3.11.2 test corrected • Test 14.11.3.11.3 test corrected • General typo, grammar, punctuation, and formatting fixes • Added terms and abbreviations to Table 1-1 • Clarified mention of AES-GCM and its supporting document, NIST Special Publication 800-38D • Added mention of DSP0236, DSP0237, and DSP0238 in Section 11.4.1 • Adjusted Chapter 11.0, "CXL Security" section heading levels • In Chapter 14.0, "CXL Compliance Testing", clarified prerequisites, test equipment, and SPDM-related command styles, and updated bit ranges and values to match CXL 3.0 v0.7 updates 	August 1, 2022

EVALUATION COPY

Revision	Description	Date
3.0	<ul style="list-style-type: none">• Chapter 6.0, "Flex Bus Physical Layer" updates for some error cases.• Flex Bus Port DVSEC updates for CXL 3.0 feature negotiation.• ALMP updates for L0p• Chapter 5.0, "CXL ARB/MUX" and Chapter 10.0, "Power Management" updates for CXL 3.0 PM negotiation flow.• Incremental changes to Back-Invalidation Snoop (BISnp) in CXL.mem including flows, ordering rules, and field names.• Added channel description to CXL.mem protocol covering BISnp and Symmetric Memory.• Added Shared FAM overview• Integrated following CXL 2.0 ECNs:<ul style="list-style-type: none">— Compliance DOE 1B— Compliance DOE return value— Error Isolation on CXL.cache and CXL.mem— CXL.cachemem IDE Establishment Flow— Mailbox Ready Time— NULL CXL Capability ID— QoS Telemetry Compliance Testcases— Vendor Specific Extension to Register Locator DVSEC— Devices Operating in CXL 1.1 mode with no RCRB— Component State Dump Log— 3, 6, 12, and 16-way Memory Interleaving• Incorporated Compute Express Link (CXL) 2.0 Errata F1-F34.• Incorporated the following CXL 2.0 ECNs: "Compliance DOE 1B", "Memory Device Error Injection", and "CEDT CFMWS & QTG _DSM"• Updated Chapter 3.0, "CXL Transaction Layer", Chapter 4.0, "CXL Link Layers", Chapter 5.0, "CXL ARB/MUX", and Chapter 6.0, "Flex Bus Physical Layer" with CXL 3.0 feature support (CXL 3.0 flit format; Back-Invalidation Snoops; Symmetric Memory flows; Cache Scaling; Additional fields in CXL 2.0 flit format to enable CXL 3.0 features on devices without CXL 3.0 flit support; updated ARB/MUX flows and ALMP definitions for CXL 3.0)• Enhanced I/O feature description is in separate appendix for now – this will be merged into Chapter 3.0, "CXL Transaction Layer" in a future release.	August 1, 2022

EVALUATION COPY

Revision	Description	Date
2.0	<ul style="list-style-type: none"> • Incorporated Errata for the Compute Express Link Specification Revision 1.1. Renamed L1.1 - L1.4 to L1.0 - L1.3 in the ARB/MUX chapter to make consistent with PCI Express* (PCIe*). • Added new chapter Chapter 7.0, "Switching". Added CXL Integrity and Data Encryption definition to the Security chapter. Added support for Hot-Plug, persistent memory, memory error reporting, and telemetry. • Removed the Platform Architecture chapter • Change to CXL.mem QoS Telemetry definition to use message-based load passing from Device to host using newly defined 2-bit DevLoad field passed in all S2M messages. • Chapter 3.0, "CXL Transaction Layer" - Update to ordering tables to clarify reasoning for "Y" (bypassing). • Chapter 4.0, "CXL Link Layers" - Updates for QoS and IDE support. ARB/MUX chapter clarifications around vLSM transitions and ALMPS status synchronization handshake and resolution. • Chapter 6.0, "Flex Bus Physical Layer" - Updates around retimer detection, additional check during alternate protocol negotiation, and clarifications around CXL operation without 32 GT/s support. Major compliance chapter update for CXL 2.0 features. • Add Register, mailbox command, and label definitions for the enumeration and management of both volatile and persistent memory CXL devices. • Chapter 7.0, "Switching" - Incorporated 0.7 draft review feedback • Chapter 8.0, "Control and Status Registers" - Updated DVSEC ID 8 definition to be more scalable, deprecated Error DOE in favor of CXL Memory Configuration Interface ECR, updated HDM decoder definition to introduce DPASkip, Added CXL Snoop filter capability structure, Merged CXL Memory Configuration Interface ECR, incorporated 0.7 draft review feedback • Chapter 9.0, "Reset, Initialization, Configuration, and Manageability" - Aligned device reset terminology with PCIe, Moved MEFN into different class of CXL VDMs, removed cold reset section, replaced eFLR section with CXL Reset, additional clarifications regarding GPF behavior, added firmware flow for detecting retimer mismatch in CXL 1.1 system, added Memory access/config access/error reporting flows that describe CXL 1.1 device below a switch, added section that describes memory device label storage, added definition of CEDT ACPI table, incorporated 0.7 draft review feedback • Chapter 12.0, "Reliability, Availability, and Serviceability" - Added detailed flows that describe how a CXL 1.1 device, Downstream Port and Upstream Port detected errors are logged and signaled, clarified that CXL 2.0 device must keep track of poison received, updated memory error reporting section per CXL Memory Configuration Interface ECR, incorporated 0.7 draft review feedback • Added Appendix B, "Memory Protocol Tables" to define legal CXL.mem request/response messages and device state for Type 2 and Type 3 devices. • Updated to address member feedback. • Incorporated PCRC updates. • Incorporated QoS (Synchronous Load Reporting) changes. • Updated viral definition to cover the switch behavior. 	October 26, 2020

EVALUATION COPY

Revision	Description	Date
1.1	<ul style="list-style-type: none"> Added Reserved and ALMP terminology definition to Terminology/Acronyms table and also alphabetized the entries. Completed update to CXL* terminology (mostly figures); removed disclaimer re: old terminology. General typo fixes. Added missing figure caption in Transaction Layer chapter. Modified description of Deferrable Writes in Section 3.1.7 to be less restrictive. Added clarification in Section 3.2.5.14 that ordering between CXL. CXL.io traffic and CXL.cache traffic must be enforced by the device (e.g., between MSIs and D2H memory writes). Removed ExtCmp reference in ItoMWr & MemWr. Flit organization clarification: updated Figure 4-2 and added example with Figure 4-4. Fixed typo in Packing Rules MDH section with respect to H4. Clarified that Advanced Error Reporting (AER) is required for CXL. Clarification on data interleave rules for CXL.mem in Section 3.3.11. Updated Table 5-3, "ARB/MUX State Transition Table" to add missing transitions and to correct transition conditions. Updated Section 5.1.2 to clarify rules for ALMP state change handshakes and to add rule around unexpected ALMPs. Updated Section 5.2.1 to clarify that ALMPs must be disabled when multiple protocols are not enabled. Updates to ARB/MUX flow diagrams. Fixed typos in the Physical Layer interleave example figures (LCRC at the end of the TLPs instead of IDLEs). Updated Table 6-3 to clarify Protocol ID error detection and handling. Added Section 6.7 to clarify behavior out of recovery. Increased the HDM size granularity from 1MB to 256MB (defined in the Flex Bus* Device DVSEC in Control and Status Registers chapter). Updated Viral Status in the Flex Bus Device DVSEC to RWS (from RW). Corrected the RCRB BAR definition so fields are RW instead of RWO. Corrected typo in Flex Bus Port DVSEC size value. Added entry to Table 8-21 to clarify that upper 7K of the 64K MEMBAR0 region is reserved. Corrected Table 9-1 so the PME-Turn_Off/Ack handshake is used consistently as a warning for both PCIe and CXL mode. Update Section 10.2.3 and Section 10.2.4 to remove references to EA and L2. Updated Section 12.2.3 to clarify device handling of non-function errors. Added additional latency recommendations to cover CXL.mem flows to Chapter 13.0; also changed wording to clarify that the latency guidelines are recommendations and not requirements. Added compliance test chapter. 	June, 2019
1.0	Initial release.	March, 2019

§ §

1.0 Introduction

1.1 Audience

The information in this document is intended for anyone designing or architecting any hardware or software associated with Compute Express Link (CXL) or Flex Bus.

1.2 Terminology/Acronyms

Refer to PCI Express* (PCIe*) Base Specification for additional terminology and acronym definitions beyond those listed in Table 1-1.

Table 1-1. Terminology/Acronyms (Sheet 1 of 11)

Term/Acronym	Definition
AAD	Additional Authentication Data - data that is integrity protected but not encrypted
Accelerator Acc	Devices that may be used by software running on Host processors to offload or perform any type of compute or I/O task. Examples of accelerators include programmable agents (e.g., GPU/GPGPU), fixed-function agents, or reconfigurable agents such as FPGAs.
ACL	Access Control List
ACPI	Advanced Configuration and Power Interface
ACS	Access Control Services as defined in PCIe Base Specification
ADF	All-Data Flit
AER	Advanced Error Reporting as defined in PCIe Base Specification
AES-GCM	Advanced Encryption Standard-Galois Counter Mode as defined in NIST* publication [AES-GCM]
AIC	Add In Card
Ak	Acknowledgment (bit/field name)
ALMP	ARB/MUX Link Management Packet
ARB/MUX	Arbiter/Multiplexer
ARI	Alternate Routing ID as defined in PCIe Base Specification.
ASI	Advanced Switching Interconnect
ASL	ACPI Source Language as defined in ACPI Specification
ASPM	Active State Power Management
ATS	Address Translation Services as defined in PCIe Base Specification
Attestation	Process of providing a digital signature for a set of measurements from a device and verifying those signatures and measurements on a host.
Authentication	Process of determining whether an entity is who or what it claims to be.
BAR	Base Address Register as defined in PCIe Base Specification
_BBN	Base Bus Number as defined in ACPI Specification
BCC	Base Class Code as defined in PCIe Base Specification
BDF	Bus Device Function

Table 1-1. Terminology/Acronyms (Sheet 2 of 11)

Term/Acronym	Definition
BE	Byte Enable as defined in PCIe Base Specification
BEI	BAR Equivalent Indicator
BEP	Byte-Enables Present
BI	Back-Invalidate
Bias Flip	Bias refers to coherence tracking of HDM-D* memory regions by the owning device to indicate that the host may have a cache copy. The Bias Flip is a process used to change the bias state that indicates the host has a cached copy of the line (Bias=Host) by invalidating the cache state for the corresponding address(es) in the host such that the device has exclusive access (Bias=Device).
BIR	BAR Indicator Register as defined in PCIe Base Specification
BIRsp	Back-Invalidate Response
BiSnp	Back-Invalidate Snoop
BMC	Baseboard Management Controller
BME	Bus Master Enable
BW	Bandwidth
CA	Certificate Authority
	Congestion Avoidance
CAM	Content Addressable Memory
CAS	Column Address Strobe
_CBR	CXL Host Bridge Register Info as defined in ACPI Specification
CCI	Component Command Interface
CCID	CXL Cache ID
CDAT	Coherent Device Attribute Table, a table that describes performance characteristics of a CXL device or a CXL switch.
CDL	CXL DevLoad
CEDT	CXL Early Discovery Table
CEL	Command Effects Log
CFMWS	CXL Fixed Memory Window Structure
CHBCR	CXL Host Bridge Component Registers
CHBS	CXL Host Bridge Structure
_CID	Compatible ID as defined in ACPI Specification
CIE	Correctable Internal Error
CIKMA	CXL.cachemem IDE Key Management Agent
CKID	Context Key IDentifier passed in the protocol flit for identifying security keys utilized for memory encryption using TSP.
CMA	Component Measurement and Authentication as defined in PCIe Base Specification
Coh	Coherency
Cold reset	As defined in PCIe Base Specification
Comprehensive Trust	Security model in which every device available to the TEE is presumed to be trusted by all TEEs in the system.
CPMU	CXL Performance Monitoring Unit
CRD	Credit Return(ed)
CQID	Command Queue ID
CSDS	CXL System Description Structure

Table 1-1. Terminology/Acronyms (Sheet 3 of 11)

Term/Acronym	Definition
CSR	Configuration Space register
CT	Crypto Timeout
CVME	Corrected Volatile Memory Error, a corrected error associated with volatile memory.
CXIMS	CXL XOR Interleave Math Structure
CXL	Compute Express Link, a low-latency, high-bandwidth link that supports dynamic protocol muxing of coherency, memory access, and I/O protocols, thus enabling attachment of coherent accelerators or memory devices.
CXL.cache	Agent coherency protocol that supports device caching of Host memory.
CXL.cachemem	CXL.cache/CXL.mem
CXL.io	PCIe-based non-coherent I/O protocol with enhancements for accelerator support.
CXL.mem	Memory access protocol that supports device-attached memory.
CXL Memory Device	CXL device with a specific Class Code as defined in Section 8.1.12.1 .
D2H	Device to Host
DAPM	Deepest Allowable Power Management state
DC	Dynamic Capacity
DCD	Dynamic Capacity Device
DCOH	Device Coherency agent on the device that is responsible for resolving coherency with respect to device caches and managing Bias states.
DDR	Double Data Rate
DH	Data Header
DHSW-FM	Dual Host, Fabric Managed, Switch Attached SLD EP
DHSW-FM-MLD	Dual Host, Fabric Managed, Switch Attached MLD EP
DLLP	Data Link Layer Packet as defined in PCIe Base Specification
DM	Data Mask
DMP	Device Media Partition
DMTF	Distributed Management Task Force
DOE	Data Object Exchange as defined in PCIe Base Specification
Domain	Set of host ports and devices within a single coherent HPA space
Downstream ES	Within the context of a single VH, an Edge Switch other than the Host ES.
Downstream Port	Physical port that can be a root port or a downstream switch port or an RCH Downstream Port
DPA	Device Physical Address. DPA forms a device-scoped flat address space. An LD-FAM device presents a distinct DPA space per LD. A G-FAM device presents the same DPA space to all hosts. The CXL HDM decoders or GFD decoders map HPA into DPA space.
DPC	Downstream Port Containment as defined in PCIe Base Specification
DPID	Destination PID
DRS	Data Response
DRT	DPID Routing Table
DSAR	Downstream Acceptance Rules
_DSM	Device Specific Method as defined in ACPI Specification
DSM	Device Security Manager. A logical entity in a device that can be admitted into the TCB for a TVM and enforces security policies on the device.
DSMAD	Device Scoped Memory Affinity Domain as defined in Coherent Device Attribute Table (CDAT) Specification

Table 1-1. Terminology/Acronyms (Sheet 4 of 11)

Term/Acronym	Definition
DSMAS	Device Scoped Memory Affinity Structure as defined in Coherent Device Attribute Table (CDAT) Specification
DSP	Downstream Switch Port
DTLB	Data Translation Lookaside Buffer
DUT	Device Under Test
DVSEC	Designated Vendor-Specific Extended Capability as defined in PCIe Base Specification
ECC	Error-correcting Code
ECN	Engineering Change Notice
ECS	Error Check Scrub
ECRC	End-to-End CRC
EDB	End Bad as defined in PCIe Base Specification
Edge DSP	PBR downstream switch port that connects to a device (including a GFD) or an HBR USP
Edge Port	PBR switch port suitable for connecting to an RP device, or HBR USP.
Edge Switch (ES)	Within the context of a single VH, a PBR switch that contains one or more Edge Ports.
Edge USP	PBR upstream switch port that connects to a root port
eDPC	Enhanced Downstream Port Control as defined in PCIe Base Specification
EDS	End of Data Stream
EFN	Event Firmware Notification
EIEOS	Electrical Idle Exit Ordered Set as defined in PCIe Base Specification
EIOS	Electrical Idle Ordered Set as defined in PCIe Base Specification
EIOSQ	EIOS Sequence as defined in PCIe Base Specification
EMD	Extended Metadata
EMV	Extended MetaValue
ENIW	Encoded Number of Interleave Ways
EP	Endpoint
eRCD	Exclusive Restricted CXL Device (formerly known as CXL 1.1 only Device). eRCD is a CXL device component that can operate only in RCD mode.
eRCH	Exclusive Restricted CXL Host (formerly known as CXL 1.1 only host). eRCH is a CXL host component that can operate only in RCD mode.
evPPB	Egress vPPB
Explicit Access Control	TE State ownership tracking where the state change occurs explicitly utilizing a TE State change message that is independent of each memory transaction.
Fabric Port (FPort)	PBR switch port that connects to another PBR switch port.
FAM	Fabric-Attached Memory. HDM within a Type 2 or Type 3 device that can be made accessible to multiple hosts concurrently. Each HDM region can either be pooled (dedicated to a single host) or shared (accessible concurrently by multiple hosts).
FAST	Fabric Address Segment Table
FC	Flow Control
FCBP	Flow Control Backpressured
FEC	Forwarded Error Correction
Flex Bus	A flexible high-speed port that is configured to support either PCIe or CXL.
Flex Bus.CXL	CXL protocol over a Flex Bus interconnect.
Flit	Link Layer Unit of Transfer

Table 1-1. Terminology/Acronyms (Sheet 5 of 11)

Term/Acronym	Definition
FLR	Function Level Reset
FM	The Fabric Manager is an entity separate from the switch or host firmware that controls aspects of the system related to binding and management of pooled ports and devices.
FMLD	Fabric Manager-owned Logical Device
FM-owned PPB	Link that contains traffic from multiple VCSs or an unbound physical port.
Fundamental Reset	As defined in PCIe Base Specification
FW	Firmware
GAM	GFD Async Message
GAE	Global Memory Access Endpoint
GDT	GFD Decoder Table
G-FAM	Global FAM. Highly scalable form of FAM that is presented to hosts without using Logical Devices (LDs). Like LD-FAM, G-FAM presents HDM that can be pooled or shared.
GFD	G-FAM device
GFD decoder	HPA-to-DPA translation mechanism inside a GFD
GI	Generic Affinity
GIM	Global Integrated Memory
GMV	Global Memory Mapping Vector
GO	Global Observation. This is used in the context of coherence protocol as a message to know when data is guaranteed to be observed by all agents in the coherence domain for either a read or a write.
GPF	Global Persistent Flush
H2D	Host to Device
HA	Home Agent. This is the agent on the host that is responsible for resolving system wide coherency for a given address.
HBIG	Host Bridge Interleave Granularity
HBM	High-Bandwidth Memory
HBR	Hierarchy Based Routing
HBR link	Link operating in a mode, where all flits are in HBR format
HBR switch	Switch that supports only HBR
HDM	Host-managed Device Memory. Device-attached memory that is mapped to system coherent address space and accessible to the Host using standard write-back semantics. Memory located on a CXL device can be mapped as either HDM or PDM.
HDM-H	Host-only Coherent HDM region type. Used only for Type 3 Devices.
HDM-D	Device Coherent HDM region type. Used only for Type 2 devices that rely on CXL.cache to manage coherence with the host.
HDM-DB	Device Coherent using Back-Invalidate HDM region type. Can be used by Type 2 devices or Type 3 devices.
_HID	Hardware ID as defined in ACPI Specification
HMAT	Heterogeneous Memory Attribute Table as defined in ACPI Specification
Host ES	Within the context of a single VH, the single Edge Switch connected to the RP.
Hot Reset	As defined in PCIe Base Specification
HPA	Host Physical Address
HPC	High-Performance Computing
hPPR	Hard Post Package Repair
HW	Hardware

EVALUATION COPY

Introduction

Table 1-1. Terminology/Acronyms (Sheet 6 of 11)

Term/Acronym	Definition
IDE	Integrity and Data Encryption
IDT	Interleave DPID Table
IG	Interleave Granularity
IGB	Interleave Granularity in number of bytes
Intermediate Fabric Switch	Within the context of a single VH, a PBR switch that forwards VH traffic but is not a Host ES or downstream ES.
IOMMU	I/O Memory Management Unit as defined in PCIe Base Specification
Implicit Access Control	TE State ownership tracking where the state change occurs implicitly as part of executing each memory transaction.
Initiator	Defined by TSP as a host or accelerator device that issues TSP requests.
IP2PM	Independent Power Manager to (Master) Power Manager, PM messages from the device to the host.
ISL	Inter-Switch Link. PBR link that connects Fabric Ports.
ISP	Interleave Set Position
IV	Initialization Vector
IW	Interleave Ways
IWB	Implicit Writeback
LAV	LDST Access Vector
LD	Logical Device. Entity that represents a CXL Endpoint that is bound to a VCS. An SLD contains one LD. An MLD contains multiple LDs.
LDST	LD-FAM Segment Table
LD-FAM	FAM that is presented to hosts via Logical Devices (LDs).
Link Layer Clock	CXL.cachemem link layer clock rate usually defined by the flit rate during normal operation.
LLC	Last Level Cache
LLCRD	Link Layer Credit
LLCTRL	Link Layer Control
LLR	Link Layer Retry
LLRB	Link Layer Retry Buffer
LOpt	Latency-Optimized
LRSM	Local Retry State Machine
LRU	Least recently used
LSA	Label Storage Area
LSM	Link State Machine
LTR	Latency Tolerance Reporting
LTSSM	Link Training and Status State Machine as defined in PCIe Base Specification
LUN	Logical Unit Number
M2S	Master to Subordinate
MAC	Message Authentication Code; also referred to as Authentication Tag or Integrity value.
MAC epoch	Set of flits which are aggregated together for MAC computation.
MB	Megabyte - 2^{20} bytes (1,048,576 bytes)
	Mailbox
MC	Memory Controller
MCA	Machine Check Architecture

Table 1-1. Terminology/Acronyms (Sheet 7 of 11)

Term/Acronym	Definition
MCTP	Management Component Transport Protocol
MDH	Multi-Data Header
Measurement	Representation of firmware/software or configuration data on a device.
MEC	Multiple Event Counting
MEFN	Memory Error Firmware Notification. An EFN that is used to report memory errors.
Memory Group	Set of DC blocks that can be accessed by the same set of requesters.
Memory Sparing	Repair function that replaces a portion of memory (the "spared memory") with a portion of functional memory at that same DPA.
MESI	Modified, Exclusive, Shared, and Invalid cache coherence protocol
MGT	Memory Group Table
MH-MLD	Multi-headed MLD. CXL component that contains multiple CXL ports, each presenting an MLD or SLD. The ports must correctly operate when connected to any combination of common or different hosts. The FM API is used to configure each LD as well as the overall MH-MLD. Currently, MH-MLDs are architected only for Type 3 LDs. MH-MLDs are considered a specialized type of MLD and, as such, are subject to all functional and behavioral requirements of MLDs.
MH-SLD	Multi-headed SLD. CXL component that contains multiple CXL ports, each presenting an SLD. The ports must correctly operate when connected to any combination of common or different hosts. The FM API is used to configure each SLD as well as the overall MH-SLD. Currently, MH-SLDs are architected only for Type 3 LDs.
ML	Machine Learning
MLD	Multi-Logical Device. CXL component that contains multiple LDs, out of which one LD is reserved for configuration via the FM API, and each remaining LD is suitable for assignment to a different host. Currently MLDs are architected only for Type 3 LDs.
MLD Port	An MLD Port is one that has linked up with an MLD Component. The port is natively bound to an FM-owned PPB inside the switch.
MMIO	Memory Mapped I/O
MR	Multi-Root
MRL	Manually-operated Retention Latch as defined in PCIe Base Specification
MS0	Meta0-State
MSB	Most Significant Bit
MSE	Memory Space Enable
MSI/MSI-X	Message Signaled Interrupt as defined in PCIe Base Specification
N/A	Not Applicable
Native Width	This is the maximum possible expected negotiated link width.
NDR	No Data Response
NG	Number of Groups
NIB	Number of Bitmap Entries
NIW	Number of Interleave Ways
NOP	No Operation
NT	Non-Temporal
NVM	Non-volatile Memory
NXM	Non-existent Memory
OBFF	Optimized Buffer Flush/Fill as defined in PCIe Base Specification
OHC	Orthogonal Header Content as defined in PCIe Base Specification
OOB	Out of Band

Table 1-1. Terminology/Acronyms (Sheet 8 of 11)

Term/Acronym	Definition
_OSC	Operating System Capabilities as defined in ACPI Specification
OSCKID	CKID memory encryption key configured for use with non-TEE data.
OSPM	Operating System-directed configuration and Power Management as defined in ACPI Specification
PA	Physical Address
P2P	Peer to peer
PBR	Port Based Routing
PBR link	Link where all flits are in PBR format
PBR switch	Switch that supports PBR and has PBR enabled
P/C	Producer-Consumer
PCIe	PCI Express
PCRC	CRC-32 calculated on the flit plaintext content. Encrypted PCRC is used to provide robustness against hard and soft faults internal to the encryption and decryption engines.
PDM	Private Device memory. Device-attached memory that is not mapped to system address space or directly accessible to Host as cacheable memory. Memory located on PCIe devices is of this type. Memory located on a CXL device can be mapped as either PDM or HDM.
Peer	Peer device in the context of peer-to-peer (P2P) accesses between devices.
Persistent Memory Device	Device that retains content across power cycling. A CXL Memory device can advertise "Persistent Memory" capability as long as it supports the minimum set of requirements described in Section 8.2.9.9 . The platform owns the final responsibility of determining whether a memory device can be used as Persistent Memory. This determination is beyond the scope of CXL specification.
PI	Programming Interface as defined in PCIe Base Specification
PID	PBR ID
PIF	PID Forward
PLDM	Platform-Level Data Model
PM	Power Management
PME	Power Management Event
PM2IP	(Master) Power Manager to Independent Power Manager, PM messages from the host to the device.
PPB	PCI*-to-PCI Bridge inside a CXL switch that is FM-owned. The port connected to a PPB can be disconnected, or connected to a PCIe component or connected to a CXL component.
PPR	Post Package Repair
PrimarySession	CMA/SPDM session established between the TSM or TSM RoT and the DSM. Utilized for configuring and locking the device and setting and clearing memory encryption keys using TSP.
PSK	CMA/SPDM-defined pre-shared key.
PTH	PBR TLP Header
QoS	Quality of Service
QTG	QoS Throttling Group, the group of CXL.mem target resources that are throttled together in response to QoS telemetry (see Section 3.3.3). Each QTG is identified using a number that is known as QTG ID. QTG ID is a positive integer.
Rank	Set of memory devices on a channel that together execute a transaction.
RAS	Reliability Availability and Serviceability
RC	Root Complex
RCD	Shorthand for a Device that is operating in RCD mode (formerly known as CXL 1.1 Device)
RCD mode	Restricted CXL Device mode (formerly known as CXL 1.1 mode). The CXL operating mode with a number of restrictions. These restrictions include lack of Hot-Plug support and 68B Flit mode being the only supported flit mode. See Section 9.11.1 for the complete list of these restrictions.
RCEC	Root Complex Event Collector, collects errors from PCIe RCIEPs, as defined in PCIe Base Specification.

Table 1-1. Terminology/Acronyms (Sheet 9 of 11)

Term/Acronym	Definition
RCH	Restricted CXL Host. CXL Host that is operating in RCD mode (formerly known as CXL 1.1 Host).
RCIEP	Root Complex Integrated Endpoint
RCRB	Root Complex Register Block as defined in PCIe Base Specification
RDPAS	RCEC Downstream Port Association Structure
Reserved	The contents, states, or information are not defined at this time. Reserved register fields must be read only and must return 0 (all 0s for multi-bit fields) when read. Reserved encodings for register and packet fields must not be used. Any implementation dependent on a Reserved field value or encoding will result in an implementation that is not CXL-spec compliant. The functionality of such an implementation cannot be guaranteed in this or any future revision of this specification. Flit, Slot, and message reserved bits should be cleared to 0 by the sender and the receiver should ignore them.
RFM	Refresh Management
RGT	Routing Group Table
RP	Root Port
RPID	Requester PID. In the context of a GFD, the SPID from an incoming request.
RRS	Request Retry Status
RRSM	Remote Retry State Machine
RSDT	Root System Description Table as defined in ACPI Specification
RSVD or RV	Reserved
RT	Route Table
RTT	Round-Trip Time
RwD	Request with Data
S2M	Subordinate to Master
SAT	SPID Access Table
SBR	Secondary Bus Reset as defined in PCIe Base Specification
SCC	Sub-Class Code as defined in PCIe Base Specification
SDC	Silent Data Corruption
SDS	Start of Data Stream
SecondarySession	One or more optional CMA/SPDM sessions established between a host entity and the DSM. Utilized for setting and clearing memory encryption keys using TSP.
Selective Trust	Security model in which each TEE selects which devices it shall include in its TCB. The device trusted by one TEE may not be trusted by other TEEs within the system.
SF	Snoop Filter
SFSC	Security Features for SCSI Commands
Sharer	Entity that is sharing data with another entity.
SHDA	Single Host, Direct Attached SLD EP
SHSW	Single Host, Switch Attached SLD EP
SHSW-FM	Single Host, Fabric Managed, Switch Attached SLD EP
Sideband	Signal used for device detection, configuration, and Hot-Plug in PCIe connectors, as defined in PCIe Base Specification
SLD	Single Logical Device
Smart I/O	Enhanced I/O with additional protocol support.
SP	Security Protocol
SPDM	Security Protocol and Data Model
SPDM over DOE	Security Protocol and Data Model over Data Object Exchange as defined in PCIe Base Specification

Table 1-1. Terminology/Acronyms (Sheet 10 of 11)

Term/Acronym	Definition
SPID	Source PID
sPPR	Soft Post Package Repair
SRAT	System Resource Affinity Table as defined in ACPI Specification
SRIS	Separate Reference Clocks with Independent Spread Spectrum Clocking as defined in PCIe Base Specification
SV	Secret Value
SVM	Shared Virtual Memory
SW	Software
Target	Defined by TSP as a memory expander device that receives TSP requests.
TC	Traffic Class
TCB	Trusted Computing Base - refers to the set of hardware, software and/or firmware entities that security assurances depend upon
TDISP	PCI-SIG-defined TEE Device Interface Security Protocol
TEE	Trusted Execution Environment
TEE Intent	TEE or non-TEE intent of a memory request from an initiator defined by the opcodes utilized in the transaction.
TE State	TEE Exclusive State maintained by the device for each page or cacheline to enforce access control.
TLP	Transaction Layer Packet as defined in PCIe Base Specification
TMAC	Truncated Message Authentication Code
TRP	Trailer Present
TSM	TEE Security Manager. Logical entity within a host processor that is the TCB for a TVM and enforces security policies on the host.
TSM RoT	TEE Security Manager Root of Trust. The entity on the host that establishes the CMA/SPDM PrimarySession.
TSP	CXL-defined TEE Security Protocol. The collection of requirements and interfaces that allow memory devices to be utilized for confidential computing.
TVM	TEE Virtual Machine. A TEE that has the property of a virtual machine. A TVM does not need to trust the hypervisor that hosts the TVM.
TVMCKID	CKID memory encryption key configured for use with TEE data.
UEFI	Unified Extensible Firmware Interface
UIE	Uncorrectable Internal Error
UIG	Upstream Interleave Granularity
UIO	Unordered Input/Output
UIW	Upstream Interleave Ways
Upstream Port	Physical port that can be an upstream switch port, or an Endpoint port, or an RCD Upstream Port.
UQID	Unique Queue ID
UR	Unsupported Request
USAR	Upstream Acceptance Rules
USP	Upstream Switch Port
UTC	Coordinated Universal Time
UUID	Universally Unique Identifier as defined in the IETF RFC 4122 Specification
VA	Virtual Address
VC	Virtual Channel

EVALUATION COPY

Introduction

Table 1-1. Terminology/Acronyms (Sheet 11 of 11)

Term/Acronym	Definition
VCS	Virtual CXL Switch. Includes entities within the physical switch belonging to a single VH. It is identified using the VCS ID.
VDM	Vendor Defined Message
vDSP	Downstream vPPB in a Host ES that is bound to one vUSP within a specific Downstream ES.
VH	Virtual Hierarchy. Everything from the CXL RP down, including the CXL RP, CXL PPBs, and CXL Endpoints. Hierarchy ID means the same as PCIe.
VH Mode	A mode of operation where CXL RP is the root of the hierarchy
VID	Vendor ID
vLSM	Virtual Link State Machine
VM	Virtual Machine
VMM	Virtual Machine Manager
vPPB	Virtual PCI-to-PCI Bridge inside a CXL switch that is host-owned. Can be bound to a port that is either disconnected, connected to a PCIe component or connected to a CXL component.
VTv	VendPrefixL0 Target Vector
vUSP	Upstream vPPB in a Downstream ES that is bound to one vDSP within a specific Host ES.
Warm Reset	As defined in PCIe Base Specification
XSDT	Extended System Description Table as defined in ACPI Specification

1.3 Reference Documents

Table 1-2. Reference Documents

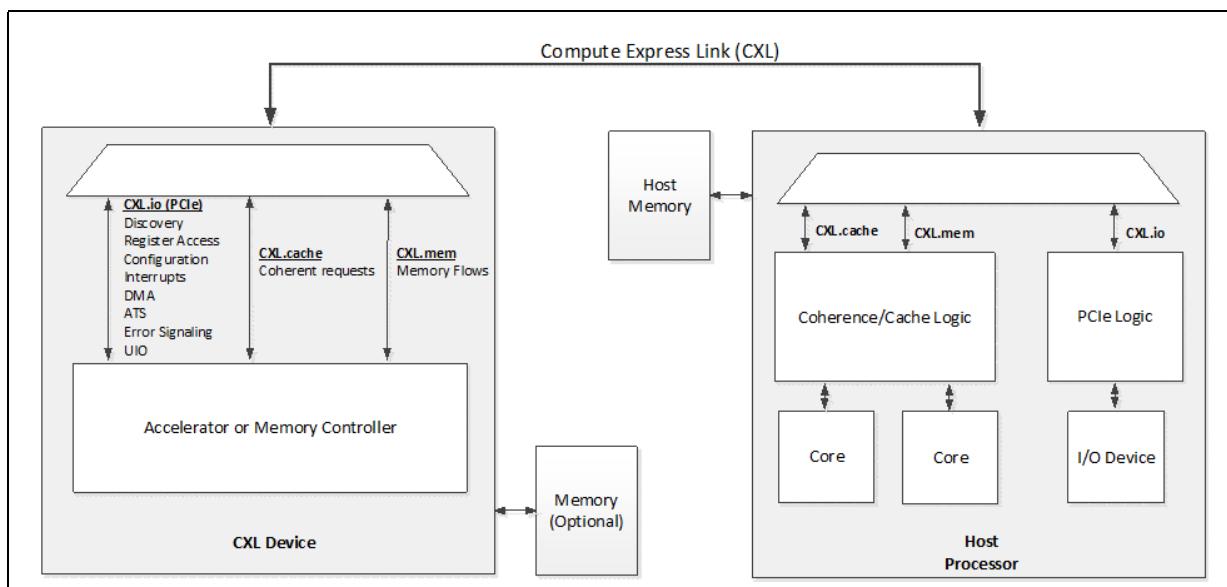
Document	Chapter Reference	Document No./Location
<i>PCI Express Base Specification Revision 6.0</i>	N/A	www.pcisig.com
<i>PCI Firmware Specification (revision 3.3 or later)</i>	Various	www.pcisig.com
<i>ACPI Specification (version 6.5 or later)</i>	Various	www.uefi.org
<i>Coherent Device Attribute Table (CDAT) Specification (version 1.04 or later)</i>	Various	www.uefi.org/acpi
<i>RFC 4122 A Universally Unique IDentifier UUID URN Namespace</i>	Various	www.ietf.org/rfc/rfc4122
<i>UEFI Specification (version 2.10 or later)</i>	Various	www.uefi.org
<i>CXL Fabric Manager API over MCTP Binding Specification (DSP0234) (version 1.0.0 or later)</i>	Chapter 7	www.dmtf.org/dsp/DSP0234
<i>Management Component Transport Protocol (MCTP) Base Specification (DSP0236) (version 1.3.1 or later)</i>	Chapters 7, 8, 11	www.dmtf.org/dsp/DSP0236
<i>Management Component Transport Protocol (MCTP) SMBus/I2C Transport Binding Specification (DSP0237) (version 1.2.0 or later)</i>	Chapter 11	www.dmtf.org/dsp/DSP0237
<i>Management Component Transport Protocol (MCTP) PCIe VDM Transport Binding Specification (DSP0238) (version 1.2.0 or later)</i>	Chapters 7, 11	www.dmtf.org/dsp/DSP0238
<i>Security Protocol and Data Model (SPDM) Specification (DSP0274) (version 1.2.0 or later)</i>	Chapters 11, 14	www.dmtf.org/dsp/DSP0274
<i>Security Protocol and Data Model (SPDM) over MCTP Binding Specification (DSP0275) (version 1.0.0 or later)</i>	Chapter 11	www.dmtf.org/dsp/DSP0275
<i>Secured Messages using SPDM over MCTP Binding Specification (DSP0276) (version 1.0.0 or later)</i>	Chapter 11	www.dmtf.org/dsp/DSP0276
<i>Secured Messages using SPDM Specification (DSP0277) (version 1.0.0 or later)</i>	Chapter 11	www.dmtf.org/dsp/DSP0277
<i>CXL Type 3 Component Command Interface over MCTP Binding Specification (DSP0281) (version 1.0.0 or later)</i>	Chapters 7, 9	www.dmtf.org/dsp/DSP0281
<i>NIST Special Publication 800-38D Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC</i>	Chapters 8, 11	nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf
<i>JEDEC DDR5 Specification, JESD79-5 (5B, version 1.2 or later)</i>	Chapters 8, 13	www.jedec.org
<i>Security Features for SCSI Commands (SFSC)</i>	Chapter 8	webstore.ansi.org
<i>Unordered IO (UIO) ECN to PCI Express Base Specification Revision 6.0</i>	N/A	members.pcisig.com/wg/PCI-SIG/document/19388

1.4 Motivation and Overview

1.4.1 CXL

CXL is a dynamic multi-protocol technology designed to support accelerators and memory devices. CXL provides a rich set of protocols that include I/O semantics similar to PCIe (i.e., CXL.io), caching protocol semantics (i.e., CXL.cache), and memory access semantics (i.e., CXL.mem) over a discrete or on-package link. CXL.io is required for discovery and enumeration, error reporting, P2P accesses to CXL memory¹ and host physical address (HPA) lookup. CXL.cache and CXL.mem protocols may be optionally implemented by the particular accelerator or memory device usage model. An important benefit of CXL is that it provides a low-latency, high-bandwidth path for an accelerator to access the system and for the system to access the memory attached to the CXL device. [Figure 1-1](#) is a conceptual diagram that shows a device attached to a Host processor via CXL.

[Figure 1-1. Conceptual Diagram of Device Attached to Processor via CXL](#)



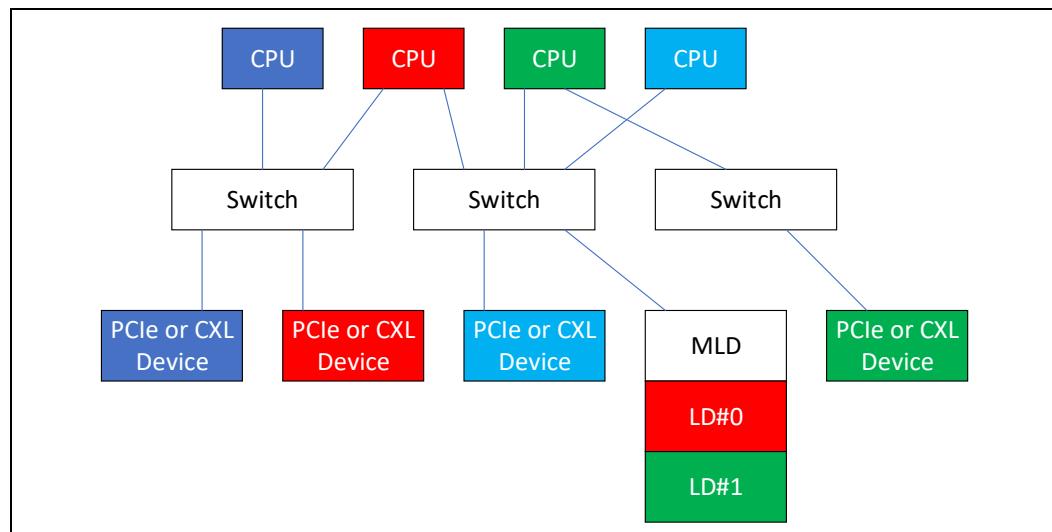
The CXL 2.0 specification enables additional usage models beyond the CXL 1.1 specification, while being fully backward compatible with the CXL 1.1 (and CXL 1.0) specification. It enables managed Hot-Plug, security enhancements, persistent memory support, memory error reporting, and telemetry. The CXL 2.0 specification also enables single-level switching support for fan-out as well as the ability to pool devices across multiple virtual hierarchies, including multi-domain support of memory devices.

[Figure 1-2](#) demonstrates memory and accelerator disaggregation through single level switching, in addition to fan-out, across multiple virtual hierarchies, each represented by a unique color. The CXL 2.0 specification also enables these resources (memory or accelerators) to be off-lined from one domain and on-lined into another domain, thereby allowing the resources to be time-multiplexed across different virtual hierarchies, depending on their resource demand.

1. Peer-to-peer flows to CXL.mem regions are supported using Unordered I/O (UIO) on the CXL.io protocol or with Direct CXL.mem access as described in this specification.

EVALUATION COPY

Figure 1-2. Fan-out and Pooling Enabled by Switches



The CXL 3.0 specification doubles the bandwidth while enabling additional usage models beyond the CXL 2.0 specification. The CXL 3.0 specification is fully backward compatible with the CXL 2.0 specification (and hence with the CXL 1.1 and CXL 1.0 specifications). The maximum Data Rate doubles to 64.0 GT/s with PAM-4 signaling, leveraging the PCIe Base Specification PHY along with its CRC and FEC, to double the bandwidth, with provision for an optional Flit arrangement for low latency. Multi-level switching is enabled with the CXL 3.0 specification, supporting up to 4K Ports, to enable CXL to evolve as a fabric extending, including non-tree topologies, to the Rack and Pod level. The CXL 3.0 specification enables devices to perform direct peer-to-peer accesses to HDM memory using UIO (in addition to MMIO memory that existed before) to deliver performance at scale, as shown in [Figure 1-3](#). Snoop Filter support can be implemented in Type 2 and Type 3 devices to enable direct peer-to-peer accesses using the back-invalidate channels introduced in CXL.mem. Shared memory support across multiple virtual hierarchies is provided for collaborative processing across multiple virtual hierarchies, as shown in [Figure 1-4](#).

CXL protocol is compatible with PCIe CEM Form Factor (4.0 and later), all form factors relating to EDSFF SSF-TA-1009 (revision 2.0 and later) and other form factors that support PCIe.

EVALUATION COPY

Figure 1-3.

Direct Peer-to-Peer Access to an HDM Memory by PCIe/CXL Devices without Going through the Host

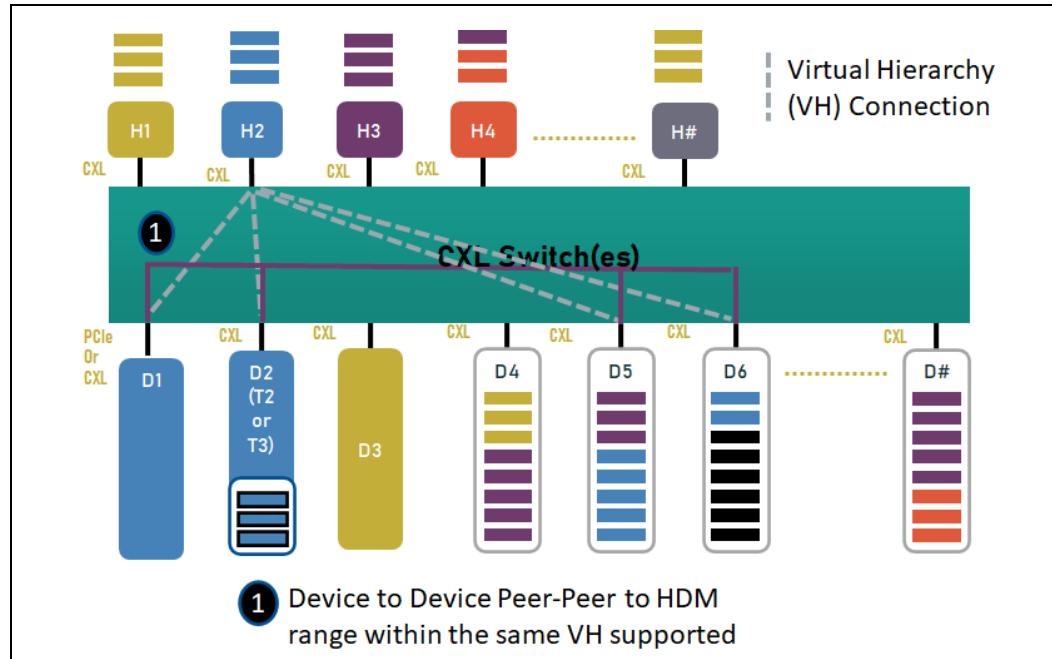
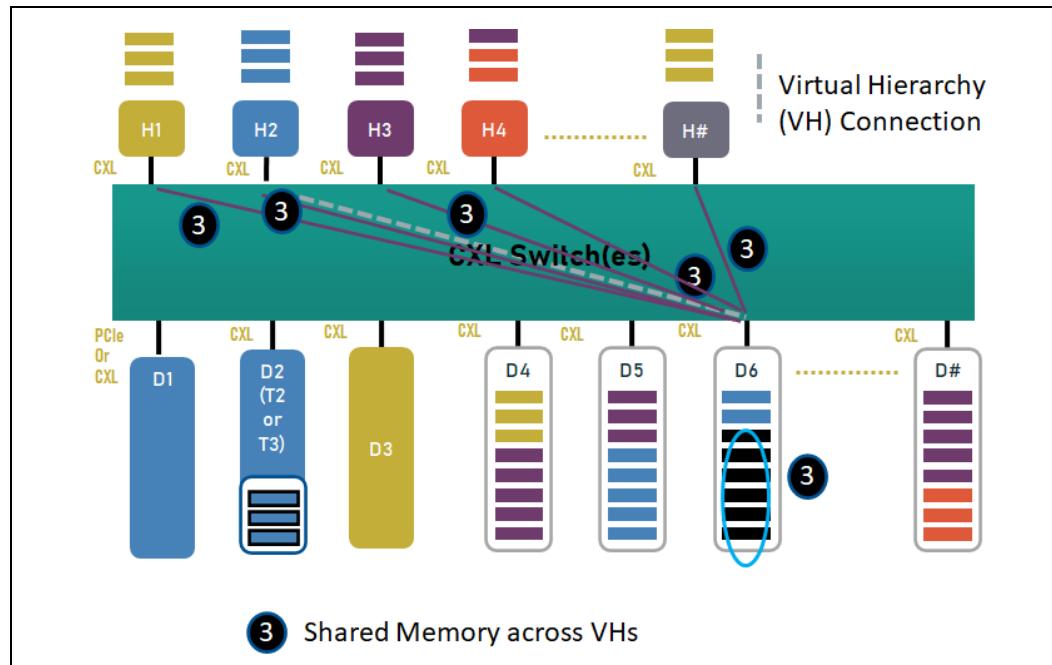


Figure 1-4.

Shared Memory across Multiple Virtual Hierarchies



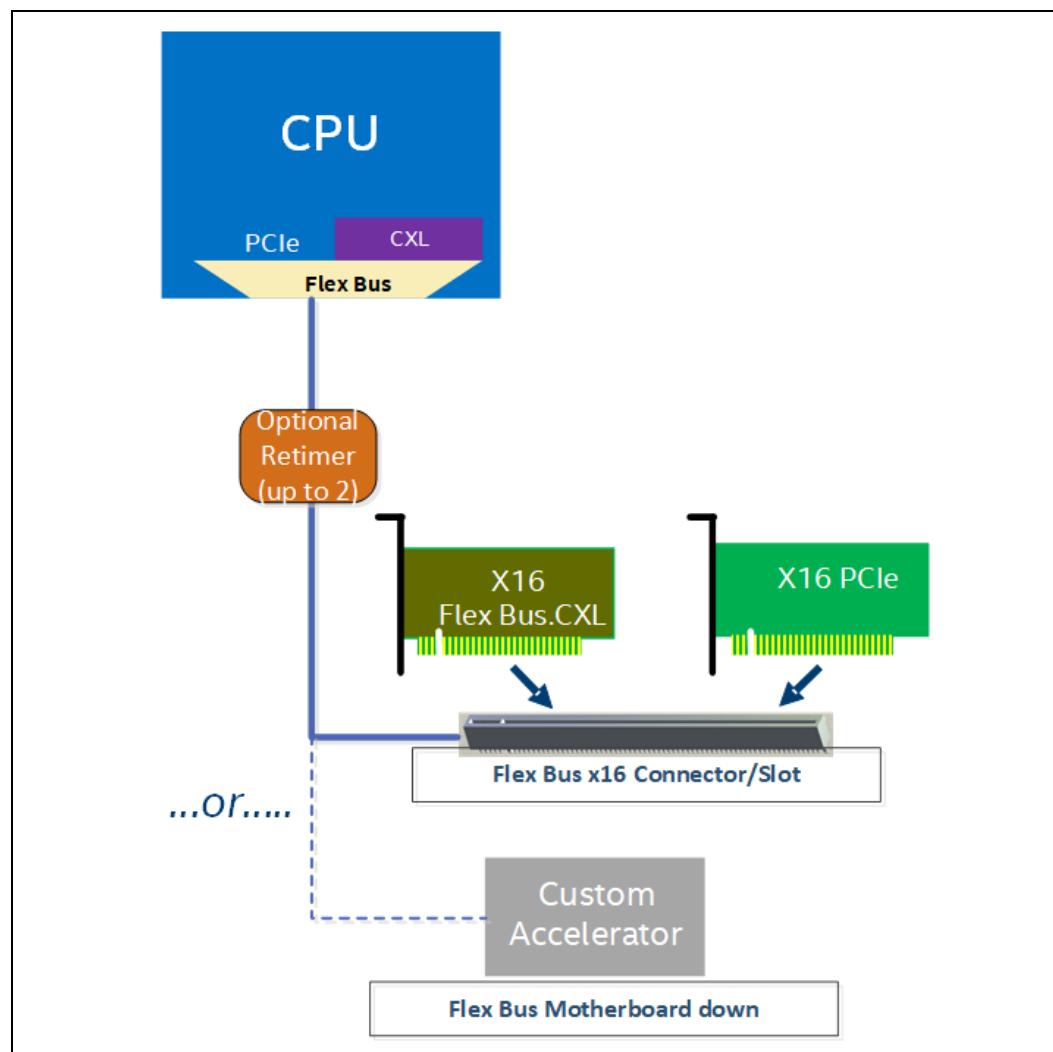
1.4.2 Flex Bus

A Flex Bus port allows designs to choose between providing native PCIe protocol or CXL over a high-bandwidth, off-package link; the selection happens during link training via alternate protocol negotiation and depends on the device that is plugged into the slot. Flex Bus uses PCIe electricals, making it compatible with PCIe retimers and with form factors that support PCIe.

[Figure 1-5](#) provides a high-level diagram of a Flex Bus port implementation, illustrating both a slot implementation and a custom implementation where the device is soldered down on the motherboard. The slot implementation can accommodate either a Flex Bus.CXL card or a PCIe card. One or two optional retimers can be inserted between the CPU and the device to extend the channel length. As illustrated in [Figure 1-6](#), this flexible port can be used to attach coherent accelerators or smart I/O to a Host processor.

[Figure 1-7](#) illustrates how a Flex Bus.CXL port can be used as a memory expansion port.

Figure 1-5. CPU Flex Bus Port Example



EVALUATION COPY

Figure 1-6. Flex Bus Usage Model Examples

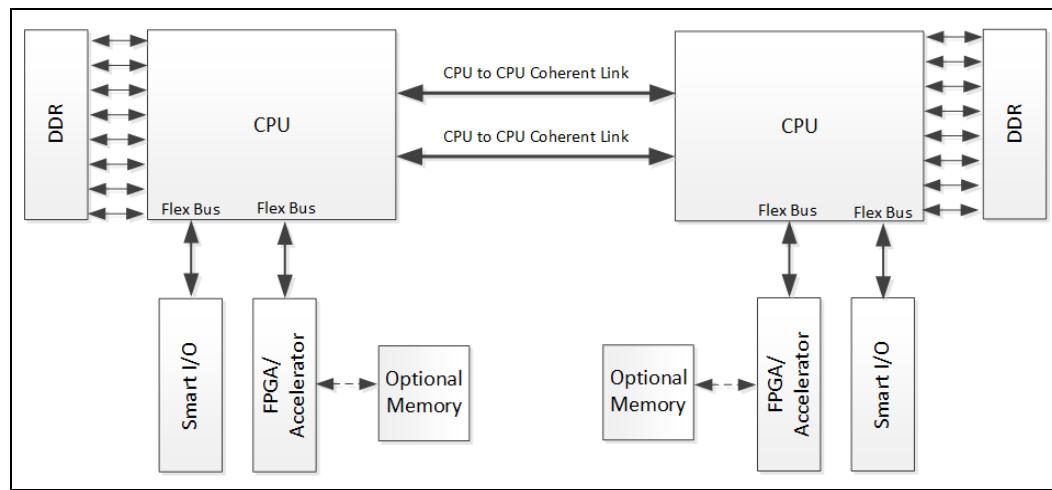


Figure 1-7. Remote Far Memory Usage Model Example

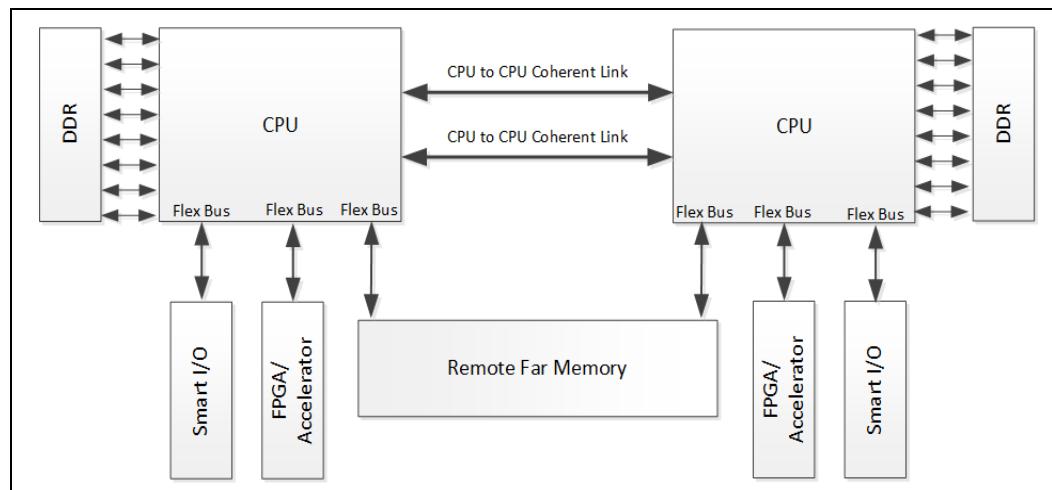
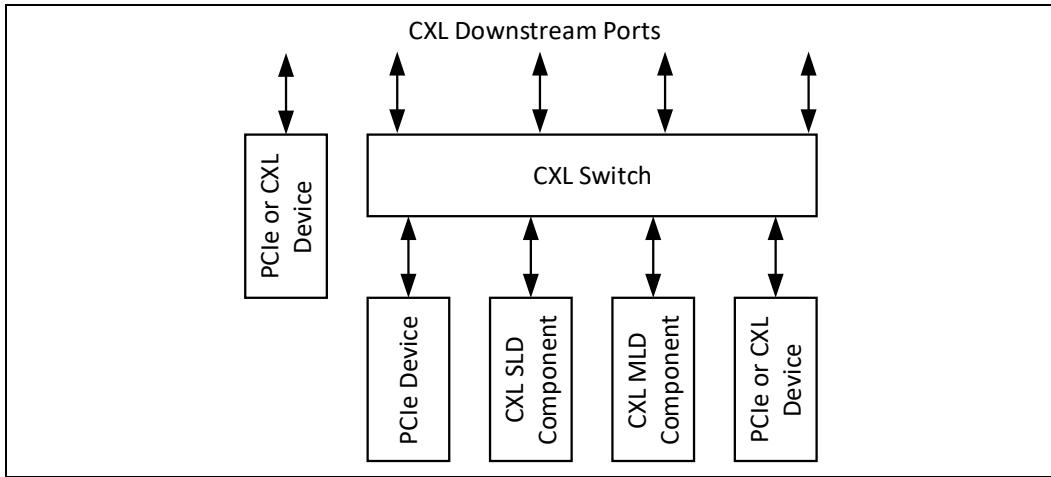


Figure 1-8.

[Figure 1-8](#) illustrates the connections that are supported below a CXL Downstream Port.

CXL Downstream Port Connections

1.5

Flex Bus Link Features

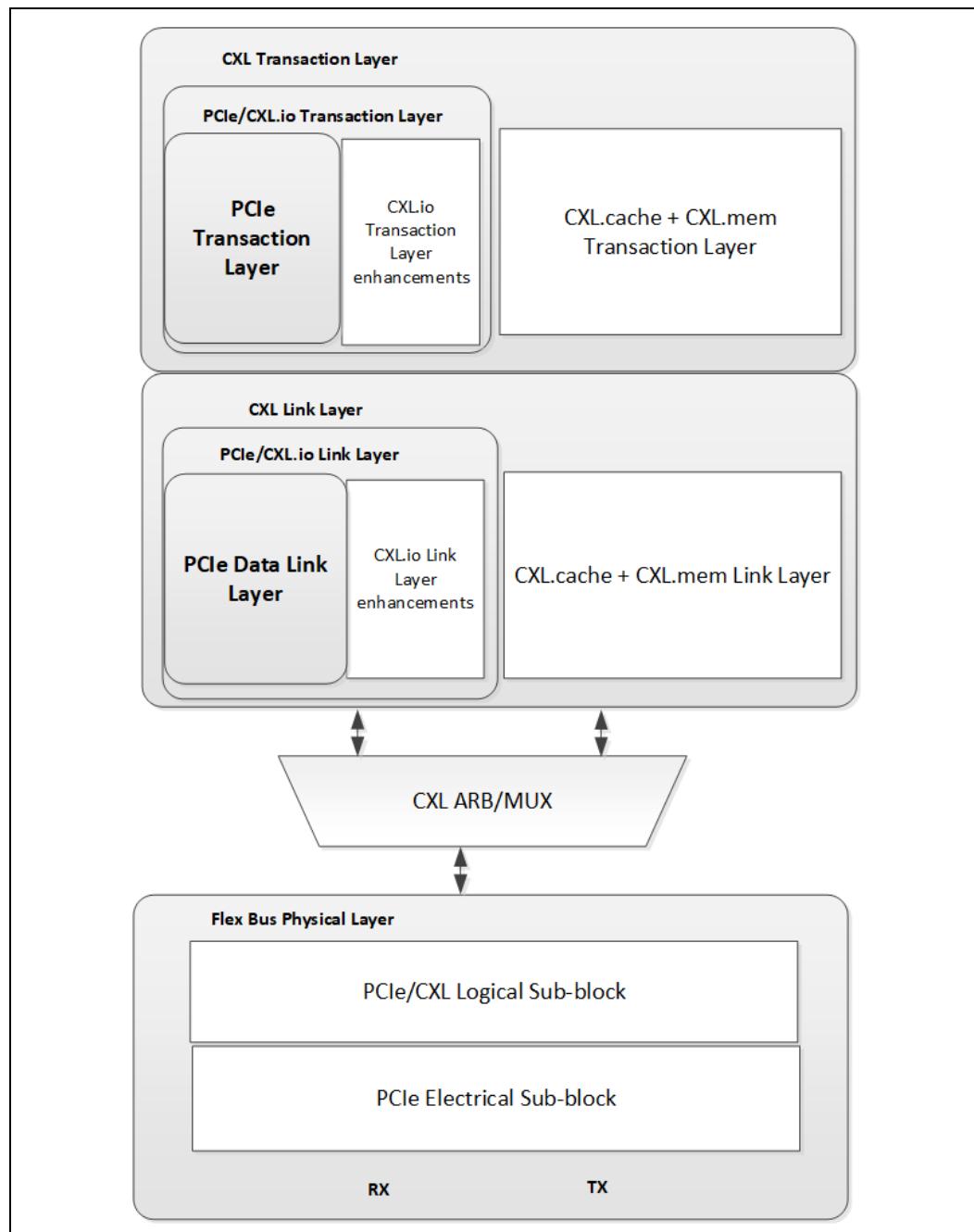
Flex Bus provides a point-to-point interconnect that can transmit native PCIe protocol or dynamic multi-protocol CXL to provide I/O, caching, and memory protocols over PCIe electricals. The primary link attributes include support of the following features:

- Native PCIe mode, full feature support as defined in PCIe Base Specification
- CXL mode, as defined in this specification
- Configuration of PCIe vs. CXL protocol mode
- With PAM4, signaling rate of 64 GT/s, and degraded rates of 32 GT/s, 16 GT/s, or 8 GT/s in CXL mode. Otherwise, signaling rate of 32 GT/s, degraded rate of 16 GT/s or 8 GT/s in CXL mode
- Link width support for x16, x8, x4, x2 (degraded mode), and x1 (degraded mode) in CXL mode
- Bifurcation (aka Link Subdivision) support to x4 in CXL mode

1.6

Flex Bus Layering Overview

Flex Bus architecture is organized as multiple layers, as illustrated in [Figure 1-9](#). The CXL transaction (protocol) layer is subdivided into logic that handles CXL.io and logic that handles CXL.cache and CXL.mem; the CXL link layer is subdivided in the same manner. Note that the CXL.cache and CXL.mem logic are combined within the transaction layer and within the link layer. The CXL link layer interfaces with the CXL ARB/MUX, which interleaves the traffic from the two logic streams. Additionally, the PCIe transaction and data link layers are optionally implemented and, if implemented, are permitted to be converged with the CXL.io transaction and link layers, respectively. As a result of the link training process, the transaction and link layers are configured to operate in either PCIe mode or CXL mode. While a host CPU would most likely implement both modes, an accelerator AIC is permitted to implement only the CXL mode. The logical sub-block of the Flex Bus physical layer is a converged logical physical layer that can operate in either PCIe mode or CXL mode, depending on the results of alternate mode negotiation during link training.

Figure 1-9. Conceptual Diagram of Flex Bus Layering

1.7

Document Scope

This document specifies the functional and operational details of the Flex Bus interconnect and the CXL protocol. It describes the CXL usage model and defines how the transaction, link, and physical layers operate. Reset, power management, and initialization/configuration flows are described. Additionally, RAS behavior is described. Refer to PCIe Base Specification for PCIe protocol details.

The contents of this document are summarized in the following chapter highlights:

- [Chapter 2.0, "CXL System Architecture"](#) – This chapter describes Type 1, Type 2, and Type 3 devices that might attach to a CPU Root Complex or a CXL switch over a CXL-capable link. For each device profile, a description of the typical workload and system resource usage is provided along with an explanation of which CXL capabilities are relevant for that workload. Bias-based and Back-Invalidate-based coherency models are introduced. This chapter also covers multi-headed devices and G-FAM devices and how they enable memory pooling and memory sharing usages. It also provides a summary of the CXL fabric extensions and its scalability.
- [Chapter 3.0, "CXL Transaction Layer"](#) – This chapter is divided into subsections that describe details for CXL.io, CXL.cache, and CXL.mem. The CXL.io protocol is required for all implementations, while the other two protocols are optional depending on expected device usage and workload. The transaction layer specifies the transaction types, transaction layer packet formatting, transaction ordering rules, and crediting. The CXL.io protocol is based on the "Transaction Layer Specification" chapter of PCIe Base Specification; any deltas from PCIe Base Specification are described in this chapter. These deltas include PCIe Vendor Defined Messages for reset and power management, modifications to the PCIe ATS request and completion formats to support accelerators. For CXL.cache, this chapter describes the channels in each direction (i.e., request, response, and data), the transaction opcodes that flow through each channel, and the channel crediting and ordering rules. The transaction fields associated with each channel are also described. For CXL.mem, this chapter defines the message classes in each direction, the fields associated with each message class, and the message class ordering rules. Finally, this chapter provides flow diagrams that illustrate the sequence of transactions involved in completing host-initiated and device-initiated accesses to device-attached memory.
- [Chapter 4.0, "CXL Link Layers"](#) – The link layer is responsible for reliable transmission of the transaction layer packets across the Flex Bus link. This chapter is divided into subsections that describe details for CXL.io and for CXL.cache and CXL.mem. The CXL.io protocol is based on the "Data Link Layer Specification" chapter of PCIe Base Specification; any deltas from PCIe Base Specification are described in this chapter. For CXL.cache and CXL.mem, the 68B flit format and 256B flit format are specified. The flit packing rules for selecting transactions from internal queues to fill the slots in the flit are described. Other features described for 68B flit mode include the retry mechanism, link layer control flits, CRC calculation, and viral and poison.
- [Chapter 5.0, "CXL ARB/MUX"](#) – The ARB/MUX arbitrates between requests from the CXL link layers and multiplexes the data to forward to the physical layer. On the receiver side, the ARB/MUX decodes the flit to determine the target to forward transactions to the appropriate CXL link layer. Additionally, the ARB/MUX maintains virtual link state machines for every link layer it interfaces with, processing power state transition requests from the local link layers and generating ARB/MUX link management packets to communicate with the remote ARB/MUX.
- [Chapter 6.0, "Flex Bus Physical Layer"](#) – The Flex Bus physical layer is responsible for training the link to bring it to operational state for transmission of PCIe packets or CXL flits. During operational state, it prepares the data from the CXL link layers or the PCIe link layer for transmission across the Flex Bus link; likewise, it converts data received from the link to the appropriate format to pass on to the appropriate link layer. This chapter describes the deltas from PCIe Base Specification to support the CXL mode of operation. The framing of the CXL flits and the physical layer packet layout for 68B Flit mode as well as 256B Flit mode are described. The mode selection process to decide between CXL mode or PCIe mode, including hardware autonomous negotiation and software-controlled selection is also described. Finally, CXL low-latency modes are described.
- [Chapter 7.0, "Switching"](#) – This chapter provides an overview of different CXL switching configurations and describes rules for how to configure switches.

EVALUATION COPY

Additionally, the Fabric Manager application interface is specified and CXL Fabric is introduced.

- [Chapter 8.0, “Control and Status Registers”](#) – This chapter provides details of the Flex Bus and CXL control and status registers. It describes the configuration space and memory mapped registers that are located in various CXL components. It also describes the CXL Component Command Interface.
- [Chapter 9.0, “Reset, Initialization, Configuration, and Manageability”](#) – This chapter describes the flows for boot, reset entry, and sleep-state entry; this includes the transactions sent across the link to initiate and acknowledge entry as well as steps taken by a CXL device to prepare for entry into each of these states. Additionally, this chapter describes the software enumeration model of both RCD and CXL virtual hierarchy and how the System Firmware view of the hierarchy may differ from the OS view. This chapter discusses the software view of CXL.mem, CXL extensions to Firmware-OS interfaces, and the CXL device manageability model.
- [Chapter 10.0, “Power Management”](#) – This chapter provides details on protocol specific link power management and physical layer power management. It describes the overall power management flow in three phases: protocol specific PM entry negotiation, PM entry negotiation for ARB/MUX interfaces (managed independently per protocol), and PM entry process for the physical layer. The PM entry process for CXL.cache and CXL.mem is slightly different than the process for CXL.io; these processes are described in separate subsections in this chapter.
- [Chapter 11.0, “CXL Security”](#) – This chapter provides details on the CXL Integrity and Data Encryption (CXL IDE) scheme that is used for securing CXL protocol flits that are transmitted across the link, IDE modes, and configuration flows.
- [Chapter 12.0, “Reliability, Availability, and Serviceability”](#) – This chapter describes the RAS capabilities supported by a CXL host, a CXL switch, and a CXL device. It describes how various types of errors are logged and signaled to the appropriate hardware or software error handling agent. It describes the Link Down flow and the viral handling expectation. Finally, it describes the error injection requirements.
- [Chapter 13.0, “Performance Considerations”](#) – This chapter describes hardware and software considerations for optimizing performance across the Flex Bus link in CXL mode. It also describes the performance monitoring infrastructure for CXL components.
- [Chapter 14.0, “CXL Compliance Testing”](#) – This chapter describes methodologies for ensuring that a device is compliant with the CXL specification.

§ §

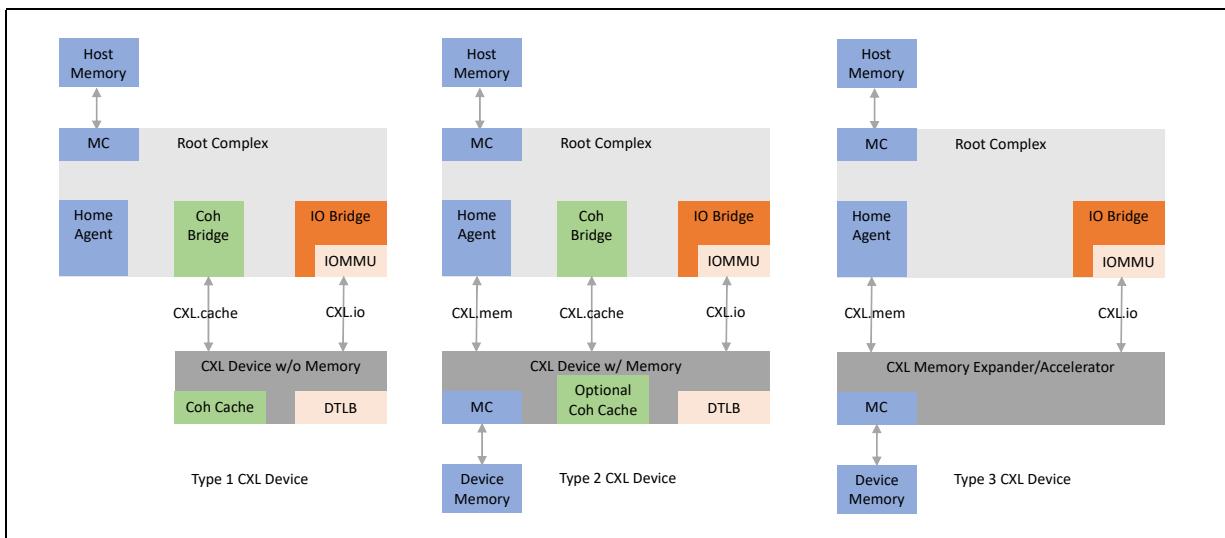
2.0

CXL System Architecture

This chapter describes the performance advantages and main features of CXL. CXL is a high-performance I/O bus architecture that is used to interconnect peripheral devices that can be either traditional non-coherent I/O devices, memory devices, or accelerators with additional capabilities. The types of devices that can attach via CXL and the overall system architecture is described in [Figure 2-1](#).

When Type 2 and Type 3 device memory is exposed to the host, it is referred to as Host-managed Device Memory (HDM). The coherence management of this memory has 3 options: Host-only Coherent (HDM-H), Device Coherent (HDM-D), and Device Coherent using Back-Invalidate Snoop (HDM-DB). The host and device must have a common understanding of the type of HDM for each address region. For additional details, refer to [Section 3.3](#).

Figure 2-1. CXL Device Types



Before diving into the details of each type of CXL device, here's a foreword about where CXL is not applicable.

Traditional non-coherent I/O devices mainly rely on standard Producer-Consumer ordering models and execute against Host-attached memory. For such devices, there is little interaction with the Host except for work submission and signaling on work completion boundaries. Such accelerators also tend to work on data streams or large contiguous data objects. These devices typically do not need the advanced capabilities provided by CXL, and traditional PCIe* is sufficient as an accelerator-attached medium.

The following sections describe various profiles of CXL devices.

2.1

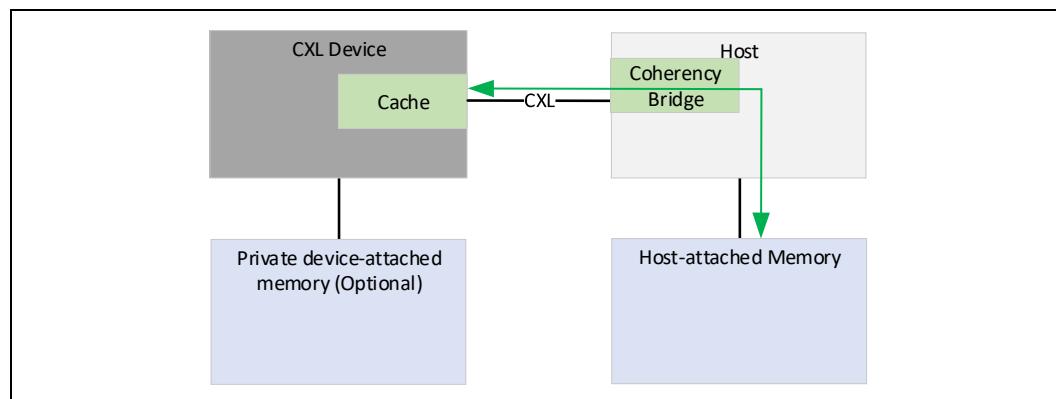
CXL Type 1 Device

CXL Type 1 Devices have special needs for which having a fully coherent cache in the device becomes valuable. For such devices, standard Producer-Consumer ordering models do not work well. One example of a device with special requirements is to perform complex atomics that are not part of the standard suite of atomic operations present on PCIe.

Basic cache coherency allows an accelerator to implement any ordering model it chooses and allows it to implement an unlimited number of atomic operations. These tend to require only a small capacity cache which can easily be tracked by standard processor snoop filter mechanisms. The size of cache that can be supported for such devices depends on the host's snoop filtering capacity. CXL supports such devices using its optional CXL.cache link over which an accelerator can use CXL.cache protocol for cache coherency transactions.

Figure 2-2.

Type 1 Device - Device with Cache

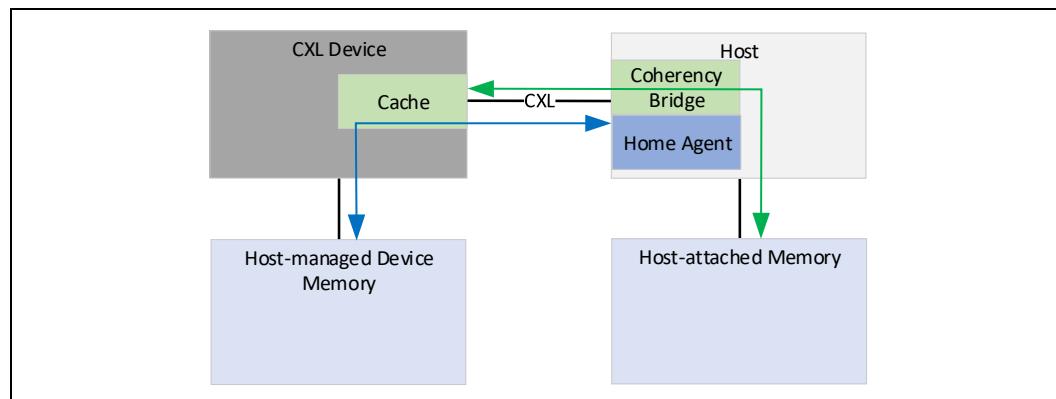


2.2

CXL Type 2 Device

CXL Type 2 are devices that negotiate all three protocols (CXL.cache, CXL.mem, and CXL.io). In addition to fully coherent cache, CXL Type 2 devices also have memory (e.g., DDR, High-Bandwidth Memory (HBM), etc.) attached to the device. These devices execute against memory, but their performance comes from having massive bandwidth between the accelerator and device-attached memory. The main goal for CXL is to provide a means for the Host to push operands into device-attached memory and for the Host to pull results out of device-attached memory such that it does not add software and hardware cost that offsets the benefit of the accelerator. This spec refers to coherent system address mapped device-attached memory as Host-managed Device Memory with Device Managed Coherence (HDM-D/HDM-DB).

There is an important distinction between HDM and traditional I/O and PCIe Private Device Memory (PDM). An example of such a device is a GPGPU with attached GDDR. Such devices have treated device-attached memory as private. This means that the memory is not accessible to the Host and is not coherent with the remainder of the system. It is managed entirely by the device hardware and driver and is used primarily as intermediate storage for the device with large data sets. The obvious disadvantage to a model such as this is that it involves high-bandwidth copies back and forth from the Host memory to device-attached memory as operands are brought in and results are written back. Please note that CXL does not preclude devices with PDM.

Figure 2-3. Type 2 Device - Device with Memory

At a high level, there are two methods of resolving device coherence of HDM. The first uses CXL.cache to manage coherence of the HDM and is referred to as "Device coherent." The memory region supporting this flow is indicated with the suffix of "D" (HDM-D). The second method uses the dedicated channel in CXL.mem called Back-Invalidate Snoop and is indicated with the suffix "DB" (HDM-DB). The following sections will describe these in more detail.

2.2.1

Back-Invalidate Snoop Coherence for HDM-DB

With HDM-DB for Type 2 and Type 3 devices, the protocol enables new channels in the CXL.mem protocol that allow direct snooping by the device to the host using a dedicated Back-Invalidate Snoop (BISnp) channel. The response channel for these snoops is the Back-Invalidate Response (BIRsp) channel. The channels allow devices the flexibility to manage coherence by using an inclusive snoop filter tracking coherence for individual cachelines that may block new M2S Requests until BISnp messages are processed by the host. All device coherence tracking options described in [Section 2.2.2](#) are also possible when using HDM-DB; however, the coherence flows to the host for the HDM-DB must only use the CXL.mem S2M BISnp channel and not the D2H CXL.cache Request channel. HDM-DB support is required for all devices that implement 256B Flit mode, but the HDM-D flows will be supported for compatibility with 68B Flit mode.

For additional details on the flows used in HDM-DB, see [Section 3.5.1, "Flows for Back-Invalidate Snoops on CXL.mem."](#)

2.2.2

Bias-based Coherency Model for HDM-D Memory

The Host-managed Device Memory (HDM) attached to a given device is referred to as device-attached memory to denote that it is local to only that device. The Bias-based coherency model defines two states of bias for device-attached memory: Host Bias and Device Bias. When the device-attached memory is in Host Bias state, it appears to the device just as regular Host-attached memory does. That is, if the device needs to access memory, it sends a request to the Host which will resolve coherency for the requested line. On the other hand, when the device-attached memory is in Device Bias state, the device is guaranteed that the Host does not have the line in any cache. As such, the device can access it without sending any transaction (e.g., request, snoops, etc.) to the Host whatsoever. It is important to note that the Host itself sees a uniform view of device-attached memory regardless of the bias state. In both modes, coherency is preserved for device-attached memory.

The main benefits of Bias-based coherency model are:

- Helps maintain coherency for device-attached memory that is mapped to system coherent address space.
- Helps the device access its local attached memory at high bandwidth without incurring significant coherency overheads (e.g., snoops to the Host).
- Helps the Host access device-attached memory in a coherent, uniform manner, just as it would for Host-attached memory.

To maintain Bias modes, a CXL Type 2 Device will:

- Implement the Bias Table which tracks page-granularity Bias (e.g., 1 per 4-KB page) which can be cached in the device using a Bias Cache.
- Build support for Bias transitions using a Transition Agent (TA). This essentially looks like a DMA engine for “cleaning up” pages, which essentially means to flush the host’s caches for lines belonging to that page.
- Build support for basic load and store access to accelerator local memory for the benefit of the Host.

The bias modes are described in detail below.

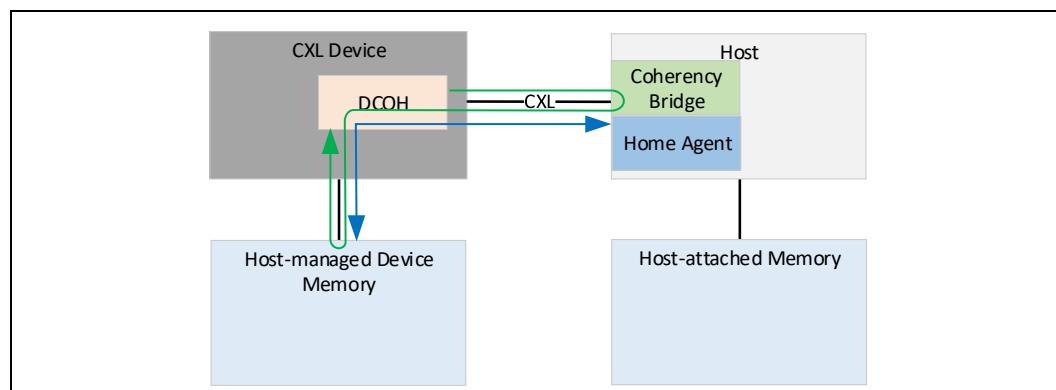
2.2.2.1

Host Bias

Host Bias mode typically refers to the part of the cycle when the operands are being written to memory by the Host during work submission or when results are being read out from the memory after work completion. During Host Bias mode, coherency flows allow for high-throughput access from the Host to device-attached memory (as shown by the bidirectional blue arrow in [Figure 2-4](#) to/from the host-managed device memory, the DCOH in the CXL device, and the Home Agent in the host) whereas device access to device-attached memory is not optimal since they need to go through the host (as shown by the green arrow in [Figure 2-4](#) that loops between the DCOH in the CXL device and the Coherency Bridge in the host, and between the DCOH in the CXL device and the host-managed device memory).

Figure 2-4.

Type 2 Device - Host Bias



2.2.2.2

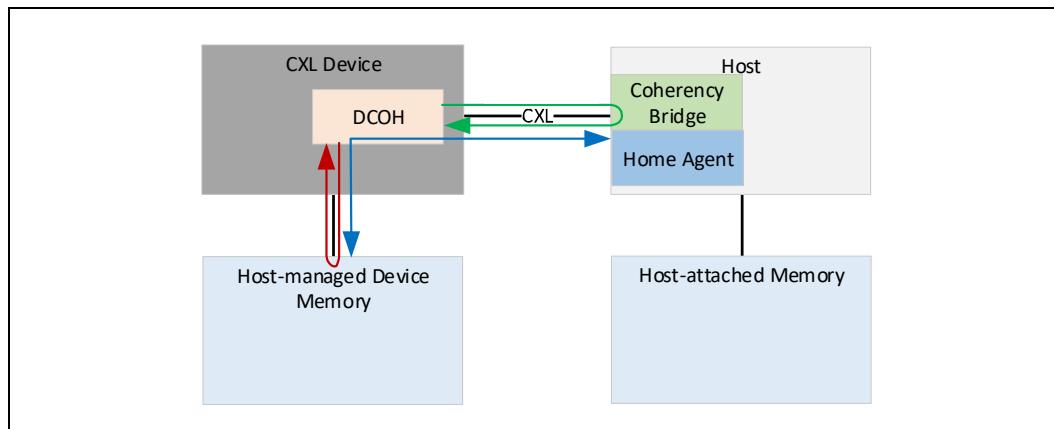
Device Bias

Device Bias mode is used when the device is executing the work, between work submission and completion, and in this mode, the device needs high-bandwidth and low-latency access to device-attached memory.

In this mode, device can access device-attached memory without consulting the Host’s coherency engines (as shown by the red arrow in [Figure 2-5](#) that loops between the DCOH in the CXL device and the host-managed device memory). The Host can still access device-attached memory but may be forced to give up ownership by the

accelerator (as shown by the green arrow in [Figure 2-5](#) that loops between the DCOH in the CXL device and the Coherency Bridge in the host). This results in the device seeing ideal latency and bandwidth from device-attached memory, whereas the Host sees compromised performance.

Figure 2-5. Type 2 Device - Device Bias



2.2.2.3 Mode Management

There are two envisioned Bias Mode Management schemes – Software Assisted and Hardware Autonomous. CXL supports both modes. Examples of Bias Flows are present in [Appendix A](#).

While two modes are described below, it is worth noting that devices do not need to implement any bias. In this case, all the device-attached memory degenerates to Host Bias. This means that all accesses to device-attached memory must be routed through the Host. An accelerator is free to choose a custom mix of Software assisted and Hardware autonomous bias management scheme. The Host implementation is agnostic to any of the above choices.

2.2.2.3.1 Software-assisted Bias Mode Management

With Software Assistance, we rely on software to know for a given page, in which state of the work execution flow the page resides. This is useful for accelerators with phased computation with regular access patterns. Based on this, software can best optimize the coherency performance on a page granularity by choosing Host or Device Bias modes appropriately.

Here are some characteristics of Software-assisted Bias Mode Management:

- Software Assistance can be used to have data ready at an accelerator before computation.
- If data is not moved to accelerator memory in advance, it is generally moved on demand based on some attempted reference to the data by the accelerator.
- In an “on-demand” data-fetch scenario, the accelerator must be able to find work to execute, for which data is already correctly placed, or the accelerator must stall.
- Every cycle that an accelerator is stalled eats into its ability to add value over software running on a core.
- Simple accelerators typically cannot hide data-fetch latencies.

Efficient software assisted data/coherency management is critical to the aforementioned class of simple accelerators.

2.2.2.3.2 Hardware Autonomous Bias Mode Management

Software assisted coherency/data management is ideal for simple accelerators, but of lesser value to complex, programmable accelerators. At the same time, the complex problems frequently mapped to complex, programmable accelerators like GPUs present an enormously complex problem to programmers if software assisted coherency/data movement is a requirement. This is especially true for problems that split computation between Host and accelerator or problems with pointer-based, tree-based, or sparse data sets.

The Hardware Autonomous Bias Mode Management, does not rely on software to appropriately manage page level coherency bias. Rather, it is the hardware that makes predictions on the bias mode based on the requester for a given page and adapts accordingly. The main benefits for this model are:

- Provide the same page granular coherency bias capability as in the software assisted model.
- Eliminate the need for software to identify and schedule page bias transitions prior to offload execution.
- Provide hardware support for dynamic bias transition during offload execution.
- Hardware support for this model can be a simple extension to the software-assisted model.
- Link flows and Host support are unaffected.
- Impact limited primarily to actions taken at the accelerator when a Host touches a Device Biased page and vice-versa.
- Note that even though this is an ostensible hardware driven solution, hardware need not perform all transitions autonomously – though it may do so if desired.

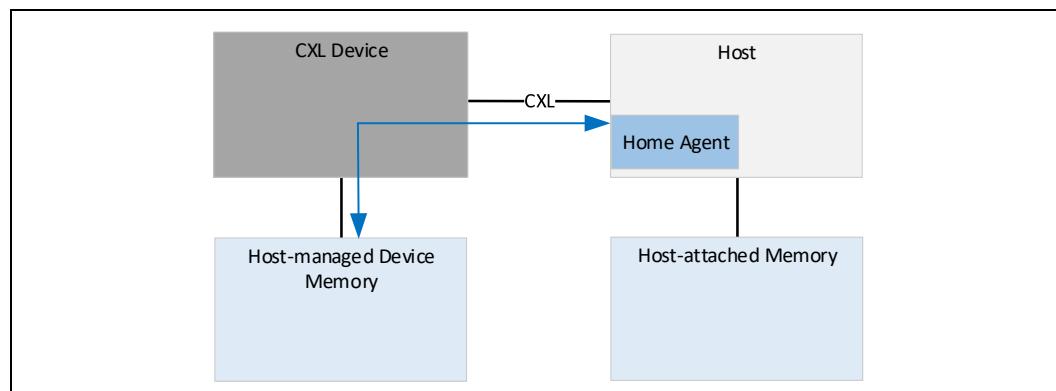
It is sufficient if hardware provides hints (e.g., “transition page X to bias Y now”) but leaves the actual transition operations under software control.

2.3

CXL Type 3 Device

A CXL Type 3 Device supports CXL.io and CXL.mem protocols. An example of a CXL Type 3 Device is an HDM-H memory expander for the Host as shown in [Figure 2-6](#).

Figure 2-6. Type 3 Device - HDM-H Memory Expander



Since this is not a traditional accelerator that operates on host memory, the device does not make any requests over CXL.cache. A passive memory expansion device would use the HDM-H memory region and normally do not directly manipulate the memory content while the memory is exposed to the host (exceptions exist for RAS and Security requirements). The device operates primarily over CXL.mem to service

2.4

requests sent from the Host. The CXL.io protocol is used for device discovery, enumeration, error reporting and management. The CXL.io protocol is permitted to be used by the device for other I/O-specific application usages. The CXL architecture is independent of memory technology and allows for a range of memory organization possibilities depending on support implemented in the Host. Type 3 device Memory that is exposed as an HDM-DB allows the same use of coherence as described in [Section 2.2.1](#) for Type 2 devices and requires the Type 3 device to include an internal Device Coherence engine (DCOH) in addition to what is shown in [Figure 2-6](#) for HDM-H. HDM-DB memory enables the device to behave as an accelerator (one variation of this is in-memory computing) and also enables direct access from peers using UIO on CXL.io or CXL.mem (see [Section 3.3.2.1](#)).

Multi Logical Device (MLD)

A Type 3 Multi-Logical Device (MLD) can partition its resources into up to 16 isolated Logical Devices. Each Logical Device is identified by a Logical Device Identifier (LD-ID) in CXL.io and CXL.mem protocols. Each Logical Device visible to a Virtual Hierarchy (VH) operates as a Type 3 device. The LD-ID is transparent to software accessing a VH. MLD components have common Transaction and Link Layers for each protocol across all LDs. Because LD-ID capability exists only in the CXL.io and CXL.mem protocols, MLDs are constrained to only Type 3 devices.

An MLD component has one LD reserved for the Fabric Manager (FM) and up to 16 LDs available for host binding. The FM-owned LD (FMLD) allows the FM to configure resource allocation across LDs and manage the physical link shared with multiple Virtual CXL Switches (VCSs). The FMLD's bus mastering capabilities are limited to generating error messages. Error messages generated by this function must only be routed to the FM.

The MLD component contains one MLD DVSEC (see [Section 8.1.10](#)) that is only accessible by the FM and addressable by requests that carry an LD-ID of FFFFh in CXL LD-ID TLP Prefix. Switch implementations must guarantee that FM is the only entity that is permitted to use the LD-ID of FFFFh.

An MLD component is permitted to use FM API to configure LDs or have statically configured LDs. In both of these configurations the configured LD resource allocation is advertised through MLD DVSEC. In addition, MLD DVSEC LD-ID Hot Reset Vector register of the FMLD is also used by CXL switch to trigger Hot Reset of one or more LDs. See [Section 8.1.10.2](#) for details.

2.4.1

LD-ID for CXL.io and CXL.mem

LD-ID is a 16-bit Logical Device identifier applicable for CXL.io and CXL.mem requests and responses. All requests targeting, and responses returned by, an MLD must include LD-ID.

See [Section 3.3.5](#) and [Section 3.3.6](#) for CXL.mem header formatting to carry the LD-ID field.

2.4.1.1

LD-ID for CXL.mem

CXL.mem supports only the lower 4 bits of LD-ID and therefore can support up to 16 unique LD-ID values over the link. Requests and responses forwarded over an MLD Port are tagged with LD-ID.

2.4.1.2

LD-ID for CXL.io

CXL.io supports carrying 16 bits of LD-ID for all requests and responses forwarded over an MLD Port. LD-ID FFFFh is reserved and is always used by the FM.

CXL.io utilizes the Vendor Defined Local TLP Prefix to carry 16 bits of LD-ID value. The format for Vendor Defined Local TLP Prefix is as follows. CXL LD-ID Vendor Defined Local TLP Prefix uses the VendPrefixL0 Local TLP Prefix type.

Table 2-1. LD-ID Link Local TLP Prefix

+0								+1								+2								+3							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
PCIe Base Specification Defined								LD-ID[15:0]																RSVD							

2.4.2 Pooled Memory Device Configuration Registers

Each LD is visible to software as one or more PCIe Endpoint (EP) Functions. While LD Functions support all the configuration registers, several control registers that impact common link behavior are virtualized and have no direct impact on the link. Each function of an LD must implement the configuration registers as described in PCIe Base Specification. Unless specified otherwise, the scope of the configuration registers is as described in PCIe Base Specification. For example, Memory Space Enable (MSE) bit in the command register controls a function's response to memory space.

Table 2-2 lists the set of register fields that have modified behavior when compared to PCIe Base Specification.

Table 2-2. MLD PCIe Registers (Sheet 1 of 2)

Register/Capability Structure	Capability Register Fields	LD-ID = FFFFh	All Other LDs
BIST Register	All Fields	Supported	Hardwire to all 0s
Device Capabilities Register	Max_Payload_Size_Supported, Phantom Functions Supported, Extended Tag Field Supported, Endpoint L1 Acceptable Latency	Supported	Mirrors LD-ID = FFFFh
	Endpoint L0s Acceptable Latency	Not supported	Not supported
	Captured Slot Power Limit Value, Captured Slot Power Scale	Supported	Mirrors LD-ID = FFFFh
Link Control Register	All Fields applicable to PCIe Endpoint	Supported (FMLD controls the fields) L0s not supported.	Read/Write with no effect
Link Status Register	All Fields applicable to PCIe Endpoint	Supported	Mirrors LD-ID = FFFFh
Link Capabilities Register	All Fields applicable to PCIe Endpoint	Supported	Mirrors LD-ID = FFFFh
Link Control 2 Register	All Fields applicable to PCIe Endpoint	Supported	Mirrors LD-ID = FFFFh RW fields are Read/Write with no effect
Link Status 2 Register	All Fields applicable to PCIe Endpoint	Supported	Mirrors LD-ID = FFFFh
MSI/MSI-X Capability Structures	All registers	Not supported	Each Functions that requires MSI/MSI-X must support it

Table 2-2. MLD PCIe Registers (Sheet 2 of 2)

Register/Capability Structure	Capability Register Fields	LD-ID = FFFFh	All Other LDs
Secondary PCIe Capability Registers	All register sets related to supported speeds (8 GT/s, 16 GT/s, 32 GT/s, 64 GT/s)	Supported	Mirrors LD-ID = FFFFh RO/Hwinit fields are Read/Write with no effect
	Lane Error Status, Local Data Parity Mismatch Status	Supported	Hardwire to all 0s
	Received Modified TS Data1 register, Received Modified TS Data 2 register, Transmitted Modified TS Data1 register, Transmitted Modified TS Data 2 register	Supported	Mirrors LD-ID = FFFFh
Lane Margining		Supported	Not supported
L1 Substates Extended Capability		Not supported	Not supported
Advanced Error Reporting (AER)	Registers that apply to Endpoint functions	Supported	Supported per LD ¹

1. AER – If an event is uncorrectable to the entire MLD, then it must be reported across all LDs. If the event is specific to a single LD, then it must be isolated to that LD.

2.4.3 Pooled Memory and Shared FAM

Host-managed Device Memory (HDM) that is exposed from a device that supports multiple hosts is referred to as Fabric-Attached Memory (FAM). FAM exposed via Logical Devices (LDs) is known as LD-FAM; FAM exposed in a more-scalable manner using Port Based Routing (PBR) links is known as Global-FAM (G-FAM).

FAM where each HDM region is dedicated to a single host interface is known as “pooled memory” or “pooled FAM”. FAM where multiple host interfaces are configured to access a single HDM region concurrently is known as “Shared FAM”, and different Shared FAM regions may be configured to support different sets of host interfaces.

LD-FAM includes several device variants. A Multi-Logical Device (MLD) exposes multiple LDs over a single shared link. A multi-headed Single Logical Device (MH-SLD) exposes multiple LDs, each with a dedicated link. A multi-headed MLD (MH-MLD) contains multiple links, where each link supports either MLD or SLD operation (optionally configurable), and at least one link supports MLD operation. See [Section 2.5, “Multi-Headed Device”](#) for additional details.

G-FAM devices (GFDs) are currently architected with one or more links supporting multiple host/peer interfaces, where the host interface of the incoming CXL.mem or UIO request is determined by its Source PBR ID (SPID) field included in the PBR message (see [Section 7.7.2](#) for additional details).

MH-SLDs and MH-MLDs should be distinguished from arbitrary multi-ported Type 3 components, such as the ones described in [Section 9.11.7.2](#), which supports a multiple CPU topology in a single OS domain.

2.4.4 Coherency Models for Shared FAM

The coherency model for each shared HDM-DB region is designated by the FM as being either multi-host hardware coherency or software-managed coherency.

Multi-host hardware coherency requires MLD hardware to track host coherence state as defined in [Table 3-37](#) for each cacheline to some varying extents, depending upon the MLD’s implementation-specific tracking mechanism, which generally can be classified as a snoop filter or full directory. Each host can perform arbitrary atomic operations supported by its Instruction-Set Architecture (ISA) by gaining Exclusive access to a cacheline, performing the atomic operation on it within its cache. The data becomes globally observed using cache coherence and follows normal hardware cache eviction flows. MemWr commands to this region of memory must set the SnpType field to No-Op to prevent deadlock, which requires that the host must acquire ownership using the M2S Request channel before issuing the MemWr resulting in 2 phases to complete a write. This is a requirement for hardware coherency model in Shared FAM and Direct P2P CXL.mem (as compared HDM-DB region that is not shared and assigned to a single host root port and can use single phase snoopable Writes).

Shared FAM may also expose memory as simple HDM-H to the host, but this will only support the software coherence model between hosts.

Software-managed coherency does not require MLD hardware to track host coherence state. Instead, software on each host uses software-specific mechanisms to coordinate software ownership of each cacheline. Software may choose to rely on multi-host hardware coherency in other HDM regions to coordinate software ownership of cachelines in software-managed coherency HDM regions. Other mechanisms for software coordinating cacheline ownership are beyond the scope of this specification.

IMPLEMENTATION NOTE

Software-managed coherency relies on software having mechanisms to invalidate and/or flush cache hierarchies as well as relying on caching agents only to issue writebacks resulting from explicit cacheline modifications performed by local software. For performance optimization, many processors prefetch data without software having any direct control over the prefetch algorithm. For a variety of implementation-specific reasons, some caching agents may spontaneously write back clean cachelines that were prefetched by hardware but never modified by local software (e.g., promoting an E to M state without a store instruction execution). Any clean writeback of a cacheline by caching agents in hosts or devices that only intended to read that cacheline can overwrite updates performed by a host or device that executed writes to the cacheline. This breaks software-managed coherency. Note that a writeback resulting from a zero-length write transaction is not considered a clean writeback. Also note that hosts and/or devices may have an internal cacheline size that is larger than 64 bytes and a writeback could require multiple CXL writes to complete. If any of these CXL writes contain software-modified data, the writeback is not considered clean.

Software-managed coherency schemes are complicated by any host or device whose caching agents generate clean writebacks. A “No Clean Writebacks” capability bit is available for a host in the CXL System Description Structure (CSDS; see [Section 9.18.1.6](#)) or for a device in the DVSEC CXL Capability2 register (see [Section 8.1.3.7](#)), with caching agents to set if it guarantees that they will never generate clean writebacks. For backward compatibility, this bit being cleared does not necessarily indicate that any associated caching agents generate clean writebacks. When this bit is set for all caching agents that may access a Shared FAM range, a software-managed coherency protocol targeting that range can provide reliable results. This bit should be ignored by software for hardware-coherent memory ranges.

2.5

Multi-Headed Device

A Type 3 device with multiple CXL ports is considered a Multi-Headed Device. Each port is referred to as a “head”. There are two types of Multi-Headed Devices that are distinguished by how they present themselves on each head:

- MH-SLD, which present SLDs on all heads
- MH-MLD, which may present MLDs on any of their heads

Management of heads in Multi-Headed Devices follows the model defined for the device presented by that head:

- Heads that present SLDs may support the port management and control features that are available for SLDs
- Heads that present MLDs may support the port management and control features that are available for MLDs

Management of memory resources in Multi-Headed Devices follows the model defined for MLD components because both MH-SLDs and MH-MLDs must support the isolation of memory resources, state, context, and management on a per-LD basis. LDs within the device are mapped to a single head.

- In MH-SLDs, there is a 1:1 mapping between heads and LDs.
- In MH-MLDs, multiple LDs are mapped to at most one head. A head in a Multi-Headed Device shall have at least one and no more than 16 LDs mapped. A head with one LD mapped shall present itself as an SLD and a head with more than one LD mapped shall present itself as an MLD. Each head may have a different number of LDs mapped to it.

Figure 2-7 and Figure 2-8 illustrate the mappings of LDs to heads for MH-SLDs and MH-MLDs, respectively.

Figure 2-7. Head-to-LD Mapping in MH-SLDs

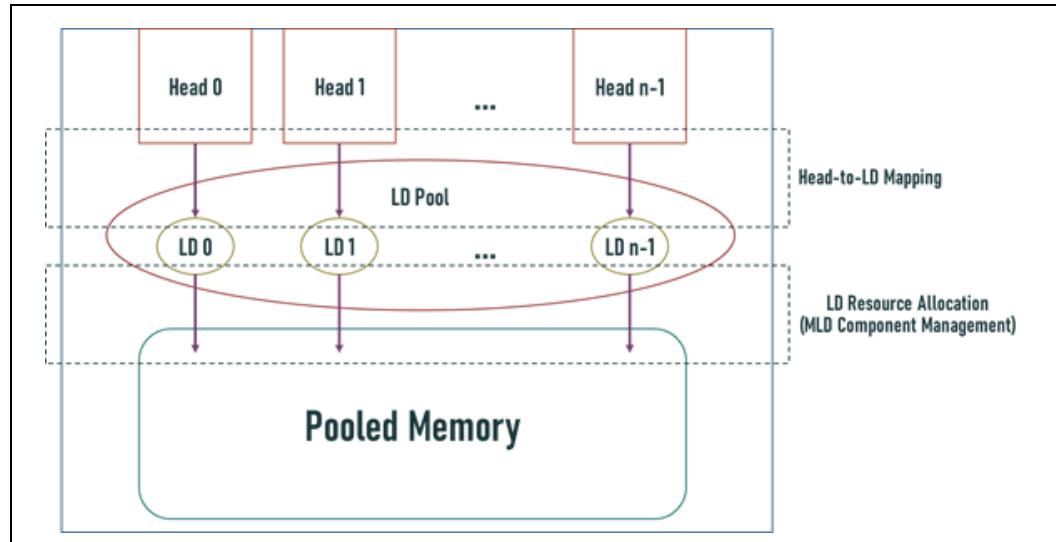
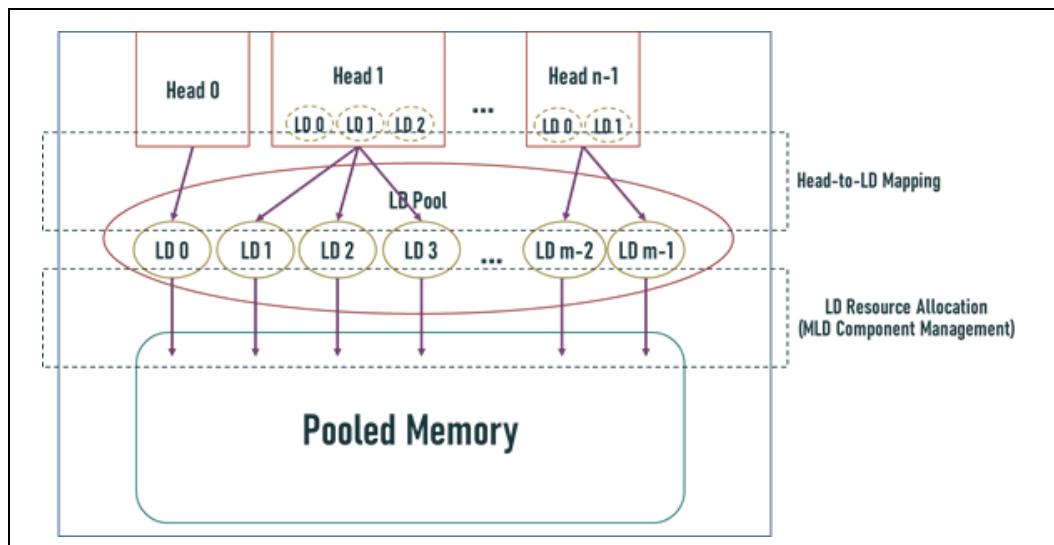


Figure 2-8. Head-to-LD Mapping in MH-MLDs

Multi-Headed Devices shall expose a dedicated Component Command Interface (CCI), the LD Pool CCI, for management of all LDs within the device. The LD Pool CCI may be exposed as an MCTP-based CCI or can be accessed via the Tunnel Management Command command through a head's Mailbox CCI, as detailed in [Section 7.6.7.3.1](#). The LD Pool CCI shall support the Tunnel Management Command for the purpose of tunneling management commands to all LDs within the device.

The number of supported heads reported by a Multi-Headed Device shall remain constant. Devices that support proprietary mechanisms to dynamically reconfigure the number of accessible heads (e.g., dynamic bifurcation of 2 x8 ports into a single x16 head, etc.) shall report the maximum number of supported heads.

2.5.1

LD Management in MH-MLDs

The LD Pool in an MH-MLD may support more than 16 LDs. MLDs exposed via the heads of an MH-MLD use LD-IDs from 0 to n-1 relative to that head, where n is the number of LDs mapped to the head. The MH-MLD maps the LD-IDs received at a head to the device-wide LD index in the MH-MLD's LD pool. The FMLD within each head of an MH-MLD shall expose and manage only the LDs that are mapped to that head.

An LD or FMLD on a head may permit visibility and management of all LDs within the device by using the Tunnel Management command to access the LD Pool CCI, as detailed in [Section 7.6.7.3.1](#).

2.6

CXL Device Scaling

CXL supports the ability to connect up to 16 Type 1 and/or Type 2 devices below a VH. To support this scaling, the Type 2 devices are required to use BISnP channel in the CXL.mem protocol to manage coherence of the HDM region. The BISnP channel introduced in the CXL 3.0 specification definition replaces the use of CXL.cache protocol to manage coherence of the device's HDM region. Type 2 devices that use CXL.cache for HDM-D coherence management are limited to a single device per Host bridge.

2.7

CXL Fabric

CXL Fabric describes features that rely on the Port Based Routing (PBR) messages and flows to enable scalable switching and advanced switching topologies. PBR enables a flexible low-latency architecture supporting up to 4096 PIDs in each fabric. G-FAM device attach (see [Section 2.8](#)) is supported natively into the fabric. Hosts and devices use standard messaging flows translated to and from PBR format through Edge Switches in the fabric. [Section 7.7](#) defines the requirements and use cases.

A CXL Fabric is a collection of one or more switches that are each PBR capable and interconnected with PBR links. A Domain is of a set of Host Ports and Devices within a single coherent Host Physical Address (HPA) space. A CXL Fabric connects one or more Host Ports to the devices within each Domain.

2.8

Global FAM (G-FAM) Type 3 Device

A G-FAM device (GFD) is a Type 3 device that connects to a CXL Fabric using a PBR link and relies on PBR message formats to provide FAM with much-higher scalability compared to LD-FAM devices. The associated FM API documented in [Section 8.2.9.9.10](#) and host mailbox interface details are provided in [Section 7.7.14](#).

Like LD-FAM devices, GFDs can support pooled FAM, Shared FAM, or both. GFDs rely exclusively on the Dynamic Capacity mechanism for capacity management. See [Section 7.7.2.3](#) for details and for other comparisons with LD-FAM devices.

2.9

Manageability Overview

To allow for different types of managed systems, CXL supports multiple types of management interfaces and management interconnects. Some are defined by external standards, while some are defined in the CXL specification.

CXL component discovery, enumeration, and basic configuration are defined by PCI-SIG* and CXL specifications. These functions are accomplished via access to Configuration Space structures and associated MMIO structures.

Security authentication and data integrity/encryption management are defined in PCI-SIG, DMTF, and CXL specifications. The associated management traffic is transported either via Data Object Exchange (DOE) using Configuration Space, or via MCTP-based transports. The latter can operate in-band using PCIe VDMs, or out-of-band using management interconnects such as SMBus, I3C, or dedicated PCIe links.

The Manageability Model for CXL Devices is covered in [Section 9.19](#). Advanced CXL-specific component management is handled using one or more CCIs, which are covered in [Section 9.20](#). CCI commands fall into 4 broad sets:

- Generic Component commands
- Memory Device commands
- FM API commands
- Vendor Specific commands

All 4 sets are covered in [Section 8.2.9](#), specifically:

- Command and capability determination
- Command foreground and background operation
- Event logging, notification, and log retrieval
- Interactions when a component has multiple CCIs

EVALUATION COPY

Each command is mandatory, optional, or prohibited, based on the component type and other attributes. Commands can be sent to devices, switches, or both.

CCIs use several transports and interconnects to accomplish their operations. The mailbox mechanism is covered in [Section 8.2.8.4](#), and mailboxes are accessed via an architected MMIO register interface. MCTP-based transports use PCIe VDMs in-band or any of the previously mentioned out-of-band management interconnects. FM API commands can be tunneled to MLDs and GFDs via CXL switches. Configuration and MMIO accesses can be tunneled to LDs within MLDs via CXL switches.

DMTF's Platform-Level Data Model (PLDM) is used for platform monitoring and control, and can be used for component firmware updates. PLDM may use MCTP to communicate with target CXL components.

Given CXL's use of multiple manageability standards and interconnects, it is important to consider interoperability when designing a system that incorporates CXL components.

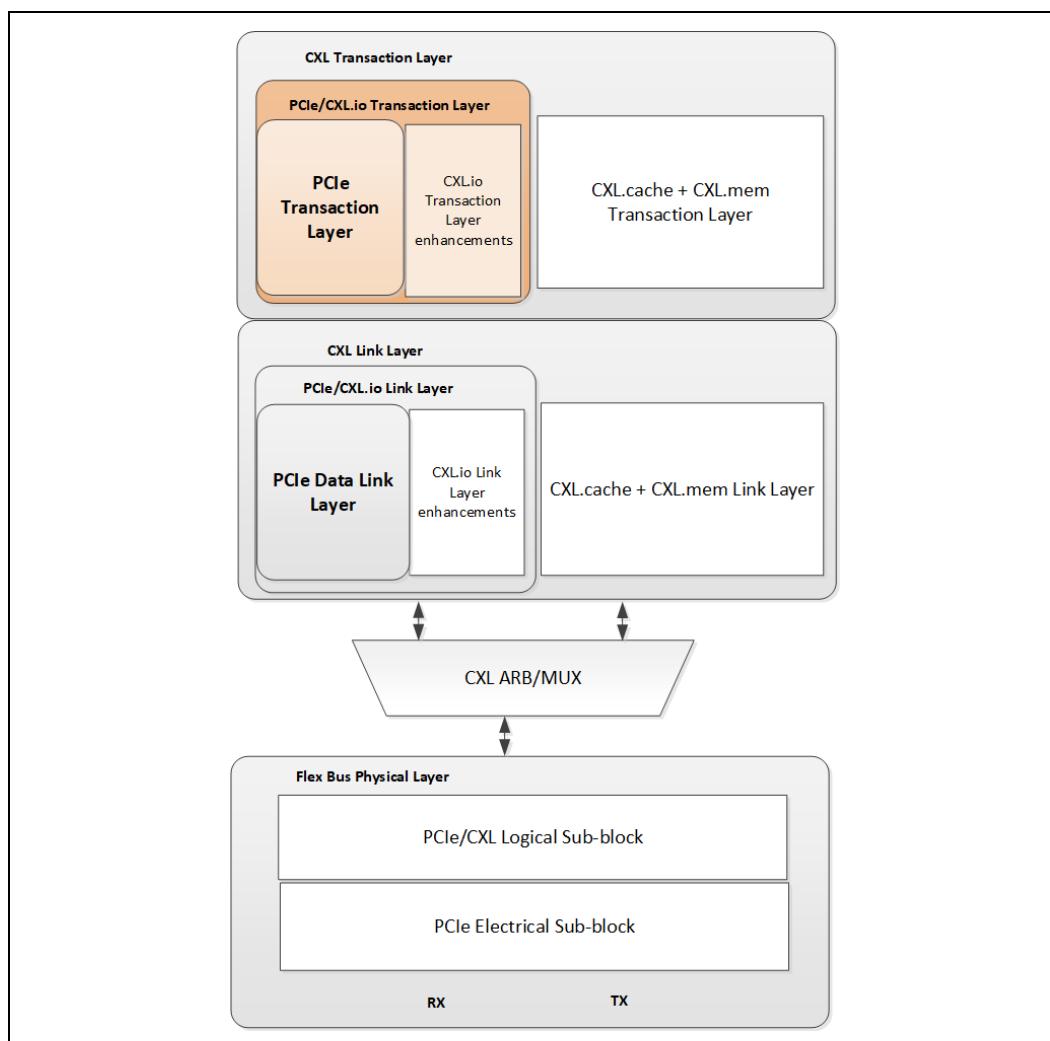
§ §

3.0 CXL Transaction Layer

3.1 CXL.io

CXL.io provides a non-coherent load/store interface for I/O devices. Figure 3-1 shows where the CXL.io transaction layer exists in the Flex Bus layered hierarchy. Transaction types, transaction packet formatting, credit-based flow control, virtual channel management, and transaction ordering rules follow the PCIe* definition; please refer to the “Transaction Layer Specification” chapter of PCIe Base Specification for details. This chapter highlights notable PCIe modes or features that are used for CXL.io.

Figure 3-1. Flex Bus Layers - CXL.io Transaction Layer Highlighted



3.1.1 CXL.io Endpoint

The CXL Alternate Protocol negotiation determines the mode of operation. See [Section 9.11](#) and [Section 9.12](#) for descriptions of how CXL devices are enumerated with the help of CXL.io.

A Function on a CXL device must not generate INTx messages if that Function participates in CXL.cache protocol or CXL.mem protocols. A Non-CXL Function Map DVSEC (see [Section 8.1.4](#)) enumerates functions that do not participate in CXL.cache or CXL.mem. Even though not recommended, these non-CXL functions are permitted to generate INTx messages.

Functions associated with an LD within an MLD component, including non-CXL functions, are not permitted to generate INTx messages.

3.1.2 CXL Power Management VDM Format

The CXL power management messages are sent as PCIe Vendor Defined Type 0 messages with a 4-DWORD data payload. These include the PMREQ, PMRSP, and PMGO messages. [Figure 3-2](#) and [Figure 3-3](#) provide the format for the CXL PM VDMs. The following are the characteristics of these messages:

- Fmt and Type fields are set to indicate message with data. All messages use routing of “Local-Terminate at Receiver.” Message Code is set to Vendor Defined Type 0.
- Vendor ID field is set to 1E98h¹.
- Byte 15 of the message header contains the VDM Code and is set to the value of “CXL PM Message” (68h).
- The 4-DWORD Data Payload contains the CXL PM Logical Opcode (e.g., PMREQ, GPF) and any other information related to the CXL PM message. Details of fields within the Data Payload are described in [Table 3-1](#).

If a CXL component receives PM VDM with poison (EP=1), the receiver shall drop such a message. Because the receiver is able to continue regular operation after receiving such a VDM, it shall treat this event as an advisory non-fatal error.

If the receiver Power Management Unit (PMU) does not understand the contents of PM VDM Payload, it shall silently drop that message and shall not signal an uncorrectable error.

1. **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).

EVALUATION COPY

Figure 3-2. CXL Power Management Messages Packet Format - Non-Flit Mode

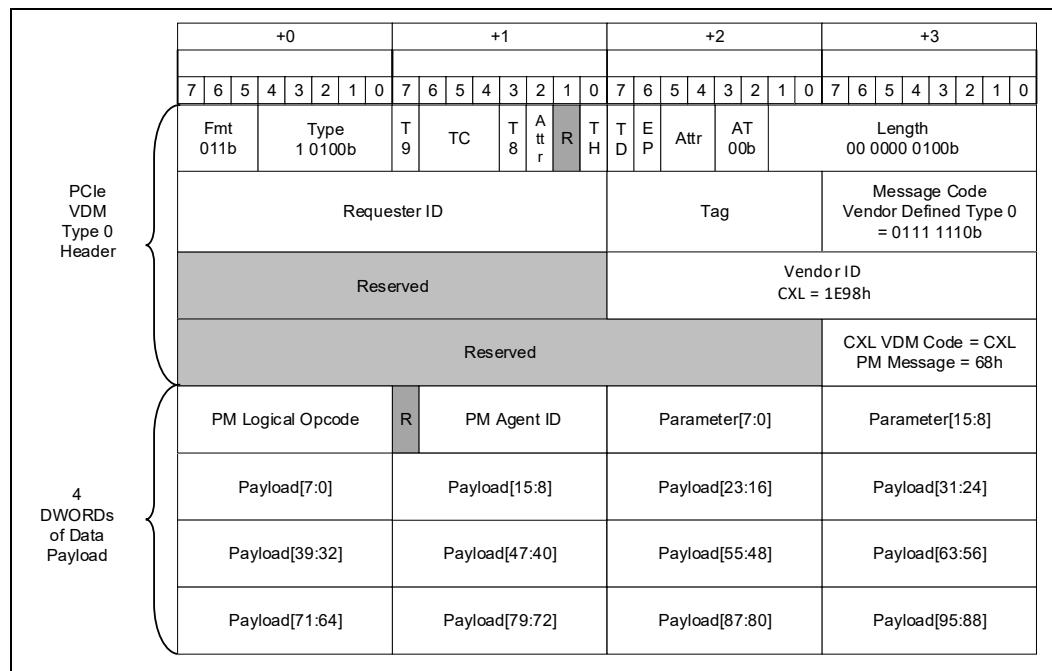


Figure 3-3. CXL Power Management Messages Packet Format - Flit Mode

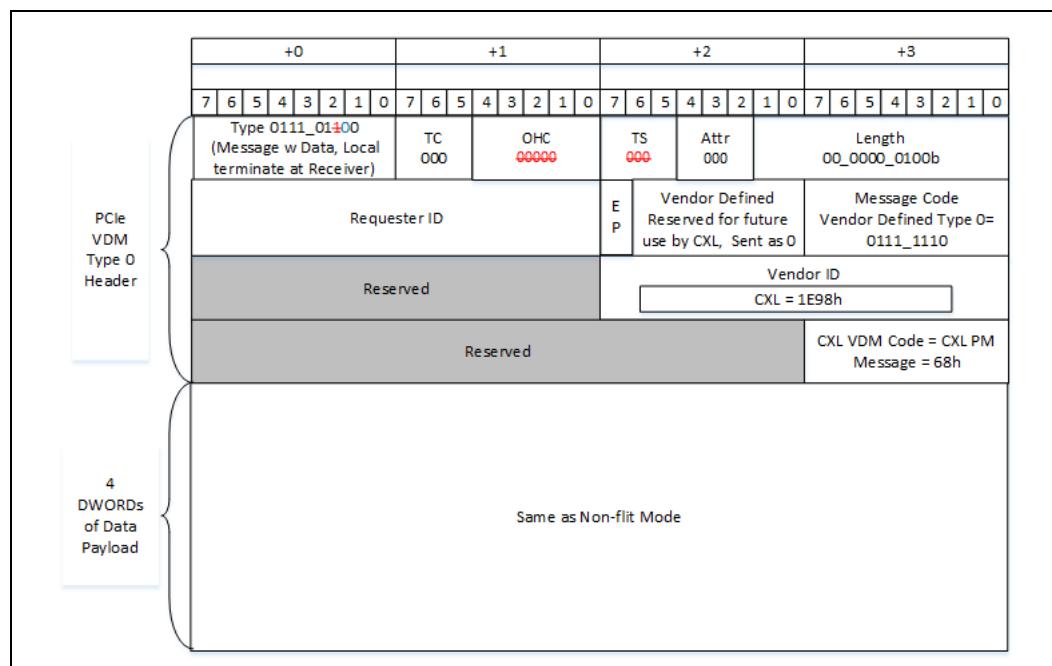


Table 3-1. CXL Power Management Messages - Data Payload Field Definitions (Sheet 1 of 2)

Field	Description	Notes
PM Logical Opcode[7:0]	Power Management Command <ul style="list-style-type: none"> • 00h = AGENT_INFO • 02h = RESETPREP • 04h = PMREQ (PMRSP and PMGO) • 06h = Global Persistent Flush (GPF) • FEh = CREDIT_RTN 	
PM Agent ID[6:0]	PM2IP: Reserved. IP2PM: PM agent ID assigned to the device. Host communicates the PM Agent ID to device via the TARGET_AGENT_ID field of the first CREDIT_RTN message.	A device does not consume this value when it receives a message from the Host.
Parameter[15:0]	CREDIT_RTN (PM2IP and IP2PM): Reserved. AGENT_INFO (PM2IP and IP2PM) <ul style="list-style-type: none"> • Bit[0]: REQUEST (set) /RESPONSE_N (cleared) • Bits[7:1]: INDEX • Bits[15:8]: Reserved PMREQ (PM2IP and IP2PM) <ul style="list-style-type: none"> • Bit[0]: REQUEST (set) /RESPONSE_N (cleared) • Bit[2]: GO • Bits[15:3]: Reserved RESETPREP (PM2IP and IP2PM) <ul style="list-style-type: none"> • Bit[0]: REQUEST (set) /RESPONSE_N (cleared) • Bits[15:1]: Reserved GPF (PM2IP and IP2PM) <ul style="list-style-type: none"> • Bit[0]: REQUEST (set) /RESPONSE_N (cleared) • Bits[15:1]: Reserved 	

Table 3-1. CXL Power Management Messages - Data Payload Field Definitions (Sheet 2 of 2)

Field	Description	Notes
Payload[95:0]	<p>CREDIT_RTN</p> <ul style="list-style-type: none"> Bits[7:0]: NUM_CREDITS (PM2IP and IP2PM) Bits[14:8]: TARGET_AGENT_ID (Valid during the first PM2IP message, reserved in all other cases) Bit[15]: Reserved <p>AGENT_INFO (Request and Response)</p> <p>If Param.Index == 0:</p> <ul style="list-style-type: none"> Bits[7:0]: CAPABILITY_VECTOR <ul style="list-style-type: none"> Bit[0]: Always set to indicate support for PM messages defined in CXL 1.1 spec Bit[1]: Support for GPF messages Bits[7:2]: Reserved All other bits are reserved <p>else: All reserved</p> <ul style="list-style-type: none"> All bits are reserved <p>RESETPREP (Request and Response)</p> <ul style="list-style-type: none"> Bits[7:0]: ResetType <ul style="list-style-type: none"> 01h = System transition from S0 to S1 03h = System transition from S0 to S3 04h = System transition from S0 to S4 05h = System transition from S0 to S5 10h = System reset Bits[15:8]: PrepType <ul style="list-style-type: none"> 00h = General Prep All other encodings are reserved Bits[17:16]: Reserved All other bits are reserved <p>PMREQ</p> <ul style="list-style-type: none"> Bits[31:0]: PCIe LTR format (as defined in Bytes 12-15 of PCIe LTR message, see Table 3-2) All other bits are reserved <p>GPF</p> <ul style="list-style-type: none"> Bits[7:0]: GPFType <ul style="list-style-type: none"> Bit[0]: Set to indicate that a power failure is imminent. Only valid for Phase 1 request messages. Bit[1]: Set to indicate device must flush its caches. Only valid for Phase 1 request messages. Bits[7:2]: Reserved Bits[15:8]: GPF Status <ul style="list-style-type: none"> Bit[8]: Set to indicate that the Cache Flush phase encountered an error. Only valid for Phase 1 responses and Phase 2 requests. Bits[15:9]: Reserved Bits[17:16]: Phase <ul style="list-style-type: none"> 01h = Phase 1 02h = Phase 2 All other encodings are reserved All other bits are reserved 	<p>CXL Agent must treat the TARGET_AGENT_ID field as reserved when returning credits to Host.</p> <p>Only Index 0 is defined for AGENT_INFO. All other Index values are reserved.</p>

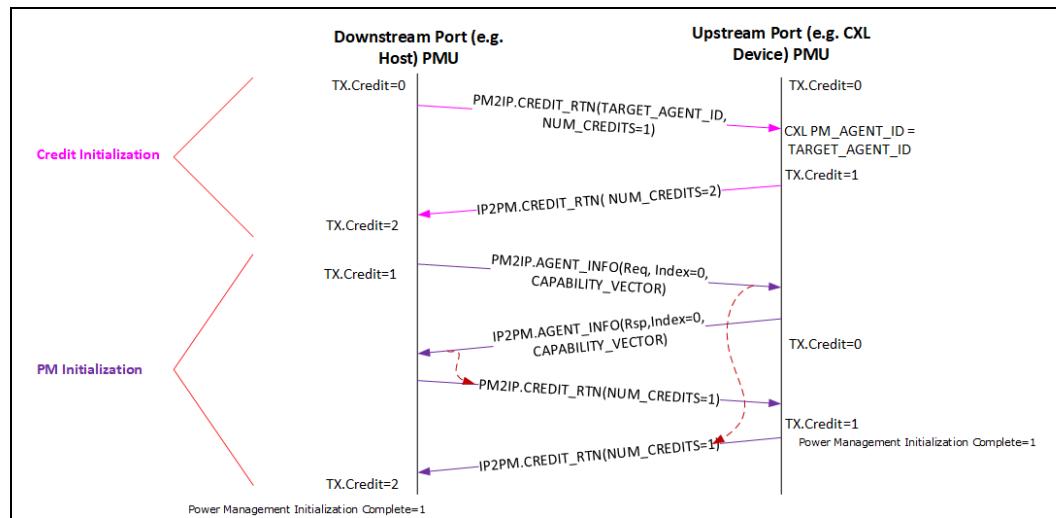
Table 3-2. PMREQ Field Definitions

Payload Bit Position	LTR Field
[31:24]	Snoop Latency[7:0]
[23:16]	Snoop Latency[15:8]
[15:8]	No-Snoop Latency[7:0]
[7:0]	No-Snoop Latency[15:8]

3.1.2.1 Credit and PM Initialization

PM Credits and initialization process is link local. [Figure 3-4](#) illustrates the use of PM2IP.CREDIT_RTN and PM2IP.AGENT_INFO messages to initialize Power Management messaging protocol intended to facilitate communication between the Downstream Port PMU and the Upstream Port PMU. A CXL switch provides an aggregation function for PM messages as described in [Section 9.1.2.1](#).

GPF messages do not require credits and the receiver shall not generate CREDIT_RTN in response to GPF messages.

Figure 3-4. Power Management Credits and Initialization

The CXL Upstream Port PMU must be able to receive and process CREDIT_RTN messages without dependency on any other PM2IP messages. Also, CREDIT_RTN messages do not use a credit. The CREDIT_RTN messages are used to initialize and update the Tx credits on each side, so that flow control can be appropriately managed. During the first CREDIT_RTN message during PM Initialization, the credits being sent via NUM_CREDITS field represent the number of credit-dependent PM messages that the initiator of CREDIT_RTN can receive from the other end. During the subsequent CREDIT_RTN messages, the NUM_CREDITS field represents the number of PM credits that were freed up since the last CREDIT_RTN message in the same direction. The first CREDIT_RTN message is also used by the Downstream Port PMU to assign a PM_AGENT_ID to the Upstream Port PMU. This ID is communicated via the TARGET_AGENT_ID field in the CREDIT_RTN message. The Upstream Port PMU must wait for the CREDIT_RTN message from the Downstream Port PMU before initiating any IP2PM messages.

An Upstream Port PMU must support at least one credit, where a credit implies having sufficient buffering to sink a PM2IP message with 128 bits of payload.

After credit initialization, the Upstream Port PMU must wait for an AGENT_INFO message from the Downstream Port PMU. This message contains the CAPABILITY_VECTOR of the PM protocol of the Downstream Port PMU. Upstream Port PMU must send its CAPABILITY_VECTOR to the Downstream Port PMU in response to the AGENT_INFO Req from the Downstream Port PMU. When there is a mismatch, Downstream Port PMU may implement a compatibility mode to work with a less capable Upstream Port PMU. Alternatively, Downstream Port PMU may log the mismatch and report an error, if it does not know how to reliably function with a less capable Upstream Port PMU.

There is an expectation from the Upstream Port PMU that it restores credits to the Downstream Port PMU as soon as a message is received. Downstream Port PMU can have multiple messages in flight, if it was provided with multiple credits. Releasing credits in a timely manner provides better performance for latency sensitive flows.

The following list summarizes the rules that must be followed by an Upstream Port PMU:

- Upstream Port PMU must wait to receive a PM2IP.CREDIT_RTN message before initiating any IP2PM messages.
- Upstream Port PMU must extract TARGET_AGENT_ID field from the first PM2IP message received from the Downstream Port PMU and use that as its PM_AGENT_ID in future messages.
- Upstream Port PMU must implement enough resources to sink and process any CREDIT_RTN messages without dependency on any other PM2IP or IP2PM messages or other message classes.
- Upstream Port PMU must implement at least one credit to sink a PM2IP message.
- Upstream Port PMU must return any credits to the Downstream Port PMU as soon as possible to prevent blocking of PM message communication over CXL Link.
- Upstream Port PMU are recommended to not withhold a credit for longer than 10 us.

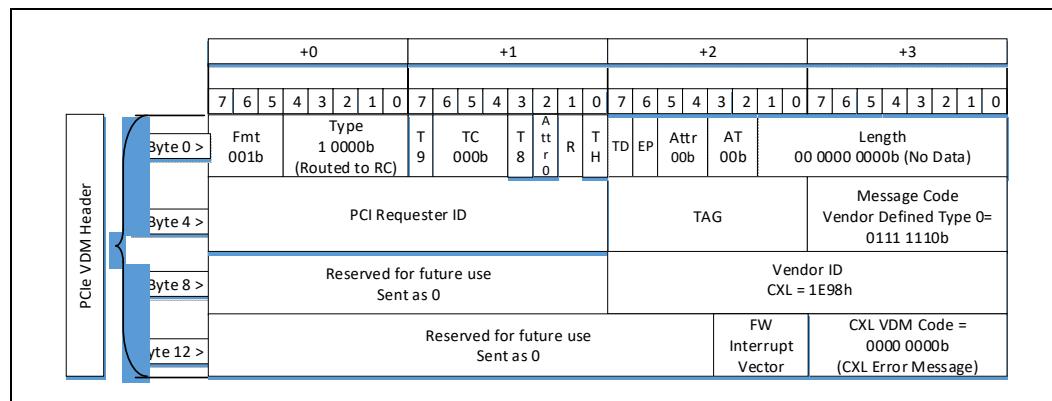
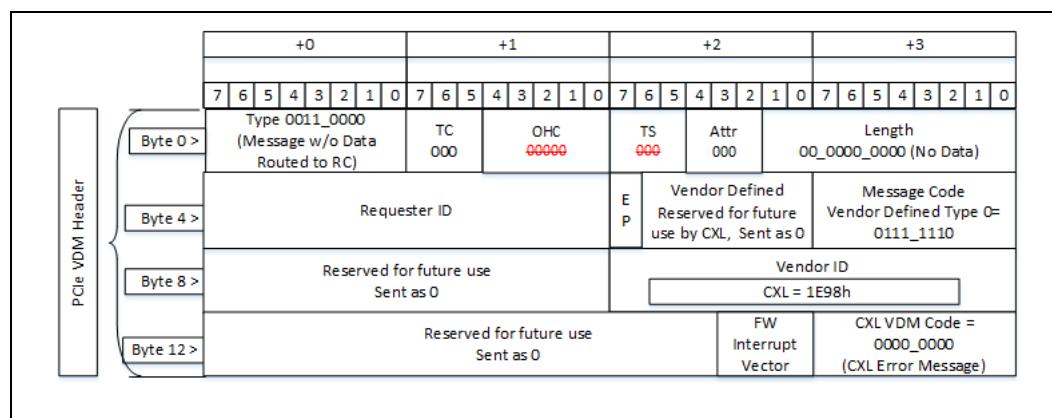
3.1.3

CXL Error VDM Format

The CXL Error Messages are sent as PCIe Vendor Defined Type 0 messages with no data payload. Presently, this class includes a single type of message, namely Event Firmware Notification (EFN). When EFN is utilized to report memory errors, it is referred to as Memory Error Firmware Notification (MEFN). [Figure 3-5](#) and [Figure 3-6](#) provide the format for EFN messages.

The following are the characteristics of the EFN message:

- Fmt and Type fields are set to indicate message with no data.
- The message is sent using routing of “Routed to Root Complex.” It is always initiated by a device.
- Message Code is set to Vendor Defined Type 0.
- Vendor ID field is set to 1E98h.
- Byte 15 of the message header contains the VDM Code and is set to the value of “CXL Error Message” (00h).
- Bytes 8, 9, 12, and 13 are cleared to all 0s.
- Bits[7:4] of Byte 14 are cleared to 0h. Bits[3:0] of Byte 14 are used to communicate the Firmware Interrupt Vector (abbreviated as FW Interrupt Vector in [Figure 3-5](#) and [Figure 3-6](#)).

Figure 3-5. CXL EFN Messages Packet Format - Non-Flit Mode**Figure 3-6. CXL EFN Messages Packet Format - Flit Mode**

Encoding of the FW Interrupt Vector field is Host specific and thus not defined by the CXL specification. A Host may support more than one type of Firmware environment and this field may be used to indicate to the Host which one of these environments is to process this message.

3.1.4 Optional PCIe Features Required for CXL

Table 3-3 lists optional features per PCIe Base Specification that are required for CXL.

Table 3-3. Optional PCIe Features Required for CXL

Optional PCIe Feature	Notes
Data Poisoning by transmitter	
ATS	Only required if CXL.cache is present (e.g., only for Type 1 and Type 2 devices, but not for Type 3 devices)
Advanced Error Reporting (AER)	

3.1.5 Error Propagation

CXL.cache and CXL.mem errors detected by the device are propagated Upstream over the CXL.io traffic stream. These errors are logged as correctable and uncorrectable internal errors in the PCIe AER registers of the detecting component.

3.1.6 Memory Type Indication on ATS

Requests to certain memory regions can only be issued on CXL.io and cannot be issued on CXL.cache. It is up to the host to decide what these memory regions are. For example, on x86 systems, the host may choose to restrict access only to Uncacheable (UC) type memory over CXL.io. The host indicates such regions by means of an indication on ATS completion to the device.

All CXL functions that issue ATS requests must set the Page Aligned Request bit in the ATS Capability register to 1. In addition, ATS requests sourced from a CXL device must set the CXL Src bit.

Figure 3-7. ATS 64-bit Request with CXL Indication - Non-Flit Mode

ATS Request	+0								+1								+2								+3																			
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0												
Byte 0	Fmt 001b	Type 0_0000b	T9	TC	T8	ATT 0	R	R	TD	EP	ATTR	AT 01b	00_00xx_xxx0b								Last DW BE 1111b	1st DW BE 1111b																						
Byte 4	Requester ID								Tag								Untranslated Address [63:32]								Reserved																			
Byte 8	Untranslated Address [31:12]																									CXL Src	R	NW																
Byte 12	Untranslated Address [31:12]																									Reserved																		

DWORD3, Byte 3, Bit 3 in ATS 64-bit request and ATS 32-bit request for both Flit Mode and Non-Flit Mode carries the CXL Src bit. [Figure 3-7](#) shows the position of this bit in ATS 64-bit request (Non-Flit mode). See PCIe Base Specification for the format of the other request messages. The CXL Src bit is defined as follows:

- 0 = Indicates request initiated by a Function that does not support CXL.io Indication on ATS.
- 1 = Indicates request initiated by a Function that supports CXL.io Indication on ATS. All CXL Functions must set this bit.

Note:

This bit is Reserved in the ATS request as defined by PCIe Base Specification.

ATS translation completion from the Host carries the CXL.io bit in the Translation Completion Data Entry. See PCIe Base Specification for the message formats.

The CXL.io bit in the ATS Translation completion is valid when the CXL Src bit in the request is set. The CXL.io bit is as defined as follows:

- 0 = Requests to the page can be issued on all CXL protocols.
- 1 = Requests to the page can be issued by the Function on CXL.io only. It is a violation to issue requests to the page using CXL.cache protocol.

3.1.7 Deferrable Writes

Earlier revisions of this specification captured the “Deferrable Writes” extension to the CXL.io protocol, but this protocol has been adopted by PCIe Base Specification.

3.1.8 PBR TLP Header (PTH)

On PBR links in a PBR fabric, all .io TLPs, with exception of NOP-TLP, carry a fixed 1-DWORD header field called the PBR TLP header (PTH). PBR links are either Inter-Switch Links (ISL) or edge links from PBR switch to G-FAM. See [Section 7.7.8](#) for details of where this header is inserted and deleted when the .io TLP traverses the PBR fabric from source to target.

NOP-TLPs are always transmitted without a preceding PTH. For Non-NOP-TLPs, PTH is always transmitted and it is transmitted on the immediate DWORD preceding the TLP Header base. Local-prefixes, if any, associated with a TLP are always transmitted before the PTH is transmitted. This is pictorially shown in [Figure 3-8](#).

To assist the receiver on a PBR link from disambiguating PTH from an NOP-TLP/Local-Prefix, the PCIe flit mode TLP grammar is modified as follows. Bits[7:6] of the first byte of all DWORDs, from the 1st DWORD of a TLP until a PTH is detected, are encoded as follows:

- 00b = NOP-TLP
- 01b = Rsvd
- 10b = Local Prefix
- 11b = PTH

After the receiver detects a PTH, PCIe TLP grammar rules are applied per PCIe Base Specification until the TLP ends, with the restriction that NOP-TLP and Local prefix cannot be transmitted in this region of the TLP.

3.1.8.1 Transmitter Rules Summary

- For NOP-TLP and Local-Prefix *Type*¹ field encodings, no PTH is pre-pended
- For all other *Type*¹ field encodings, a PTH is pre-pended immediately ahead of the Header base

3.1.8.2 Receiver Rules Summary

- For NOP-TLP, if bits[5:0] are not all 0s, the receiver treats it as a malformed packet and reports the error following the associated error reporting rules
- For a Local Prefix, if bits[5:0] are not one of 00 1101b through 00 1111b, the receiver treats it as a malformed packet and reports the error following the associated error reporting rules
- From beginning of a TLP to when a PTH is detected, receiver silently drops a DWORD if a reserved value of 01b is received for bits[7:6] in the DWORD
- If an NOP-TLP or Local Prefix is received immediately after a PTH, the receiver treats it as a malformed packet and reports the error following the associated error reporting rules

Note: Header queues in PBR switches/devices should be able to handle the additional DWORD of PTH that is needed to be carried between the source and target PBR links.

Note: PTH is included as part of normal link level CRC/FEC calculations/checks on PBR links to ensure reliable PTH delivery over the PBR link. For details regarding the PIF, DSAR, and Hie bits, see [Section 7.7.3.3](#), [Section 7.7.7](#), and [Section 7.7.6.2](#).

On MLD links, in the egress direction, the SPID information in this header is used to generate the LD-ID information on VendPrefixL0 message as defined in [Section 2.4](#). On MLD links, in the ingress direction, LD-ID in the VendPrefixL0 message is used to determine the DPID in the PBR packet.

1. Type[7:0] field as defined in PCIe Base Specification for Flit mode.

Table 3-4. PBR TLP Header (PTH) Format

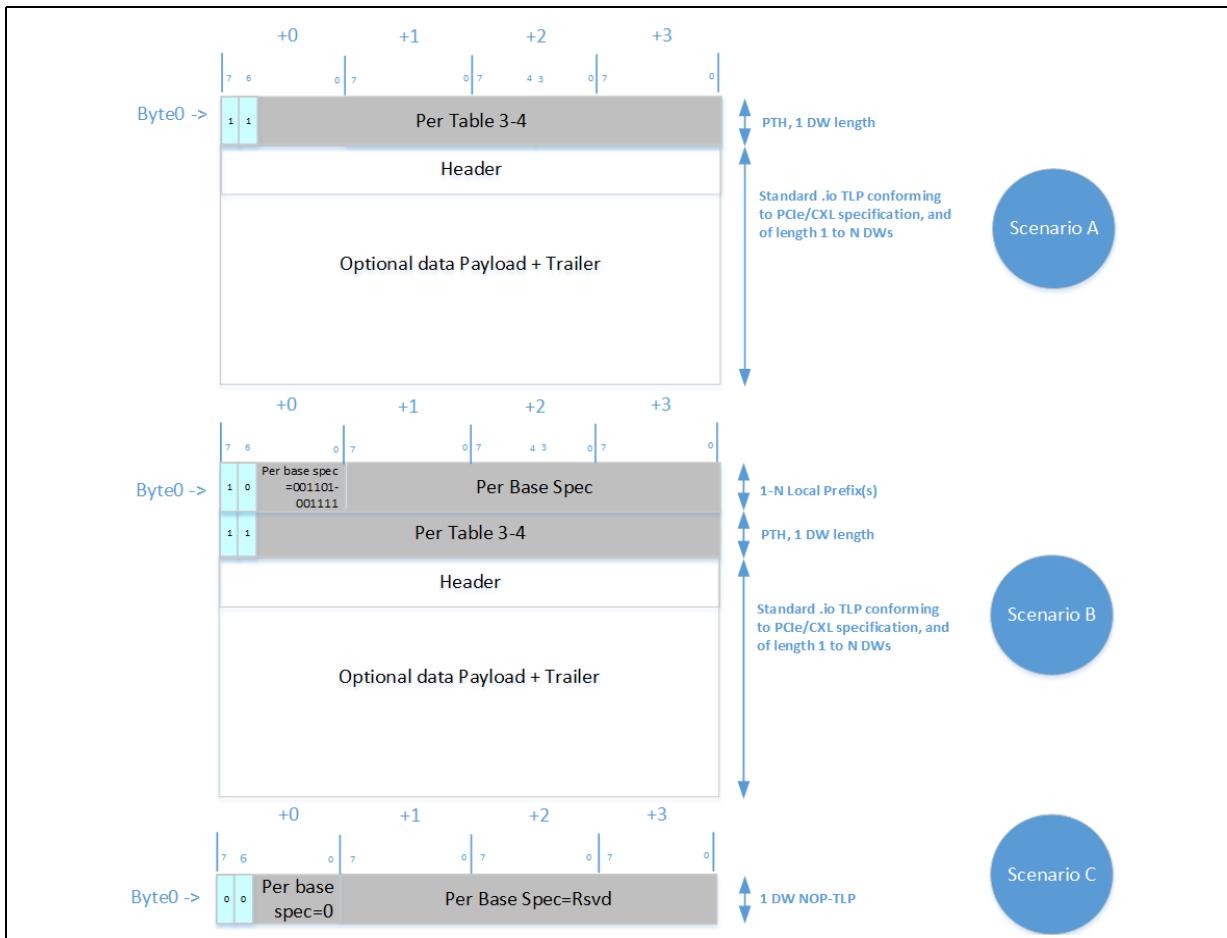
Byte	+0								+1								+2								+3							
Bits	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0 ->	1	1	Rsvd		H i e	D S A R	P I F	SPID[11:0]								DPID[11:0]																

Table 3-5. NOP-TLP Header Format

Byte	+0								+1								+2								+3							
Bits	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0 ->	0	0	00 0000b						Per PCIe Base Specification																							

Table 3-6. Local Prefix Header Format

Byte	+0								+1								+2								+3							
Bits	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0 ->	1	0	0	0	1	1	01b/ 10b/ 11b		Per PCIe Base Specification																							

Figure 3-8. Valid .io TLP Formats on PBR Links

3.1.9 VendPrefixL0

Section 2.4.1.2 describes VendPrefixL0 usage on MLD links. For non-MLD HBR links, VendPrefixL0 carries the PBR-ID field to facilitate inter-domain communication between hosts and devices (e.g., GIM; see Section 7.7.3) and other vendor-proprietary usages (see Section 7.7.4). HBR links that use this form of the prefix must be directly attached to a PBR switch. On the switch ingress side, this prefix carries the DPID of the target edge link. On the egress side, this message carries the SPID of the source link that originated the TLP. The prefix format is shown in Table 3-51.

Table 3-7. VendPrefixL0 on Non-MLD Edge HBR Links

Byte	+0								+1								+2								+3							
Bits	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0 ->	As defined in PCIe Base Specification								PBR-ID[11:0]								Rsvd															

On the switch side, handling of this prefix is disabled by default. The FM can enable this functionality on each edge USP and DSP, via CCI mailbox. The method that the FM uses to determine the set of USPs/DSPs that are capable and trustworthy of enabling this functionality is beyond the scope of this specification.

Note: Edge PCIe links are not precluded from using this prefix for the same purpose described above. However, such usages are beyond the scope of this specification.

See [Section 7.7.3](#) and [Section 7.7.4](#) for transaction flows that involve TLPs with this prefix.

3.1.10 CXL DevLoad (CDL) Field in UIO Completions

To support QoS Telemetry (see [Section 3.3.4](#)) with UIO Direct P2P to HDM (see [Section 7.7.9](#)), UIO Completions contain the 2-bit CDL field, which carries the CXL DevLoad indication from HDM devices that support UIO Direct P2P. If an HDM device supports UIO Direct P2P to HDM, the HDM device shall populate the CDL field with values as defined in [Table 3-51](#). The CDL field exists in UIOWrCpl, UIORdCpl, and UIORdCpID TLPs.

3.1.11 CXL Fabric-related VDMs

In CXL Fabric (described in [Section 7.7](#)), there are many different uses for a CXL VDM. The uses fall into two categories: within a PBR Fabric, and outside a PBR Fabric.

When a VDM has a CXL Vendor ID, bytes 14 and 15 in the VDM header distinguish the use case via a CXL VDM Code and whether the use is within a PBR fabric. If within a PBR fabric, there is also a PBR Opcode. Additionally for PBR Fabric CXL VDMs, many of the traditional PCIe-defined fields such as Requester ID have no meaning and thus are reserved or in some cases, repurposed. See [Table 3-8](#) for a breakdown of the VDM header bytes for PBR Fabric VDMs.

Table 3-8. PBR VDM

Byte	+0						+1						+2						+3											
	7	6	5	4	3	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1
0	Type 0011 0100b Type 0111 0100b						TC 000b	OHC 0 0000b						TS 000b		Attr 000b		Length 00 0000 0000b Length 00 00xx xxxx ¹ b												
4	PCIe Requester ID / Rsvd												E P	Vendor Defined Rsvd				Message Code Vendor Defined Type 0 0111 1110b												
8	Rsvd												Vendor ID CXL = 1E98h																	
12	Reserved	CmdSeq	SeqLen: 00h = 256 DWORDS						R s v d	SeqNum			PBR Opcode			CXL VDM Code														

1. x indicates don't care.

[Table 3-8](#) shows two Type encodings, a VDM without data and a VDM with data, both routed as "terminate at receiver". If a payload is not needed, the VDM without data is used. If any payload is required, the VDM with data is used. Because the PBR VDMs use PTH to route, the 'receiver' is the end of the tunnel (i.e., the matching DPID). PBR VDMs with data can have at most 128B (=32 DWORDs) of payload. If the SeqLen is more than 32 DWORDs, multiple VDMs will be needed to convey the entire sequence of VDMs (for UCPull VDM).

Depending on the CXL VDM Code, other fields in the VDM header may have meaning. Use of these additional fields will be defined in the section covering that particular encoding. These fields include:

- **PBR Opcode:** Subclass of PBR Fabric VDMs
- **CmdSeq:** Sequence number of the Host management transaction flow
- **SeqLen:** Length of the VDM sequence, applies to UCPull VDM
- **SeqNum:** Sequential VDM count with wrap if a message requires multiple sequential VDMs

Table 3-9 summarizes the various CXL vendor defined messages, each with a CXL VDM code and PBR Opcode, a message destination, and a brief summary of the message's use. The CXL VDM Code provides the category of VDM, while the PBR Opcode makes distinctions within that category. The remainder of this section deals with GFD management-related VDMs and Route Table Update related VDMs. For details of other VDMs, see [Section 7.7.11](#).

Table 3-9. CXL Fabric Vendor Defined Messages

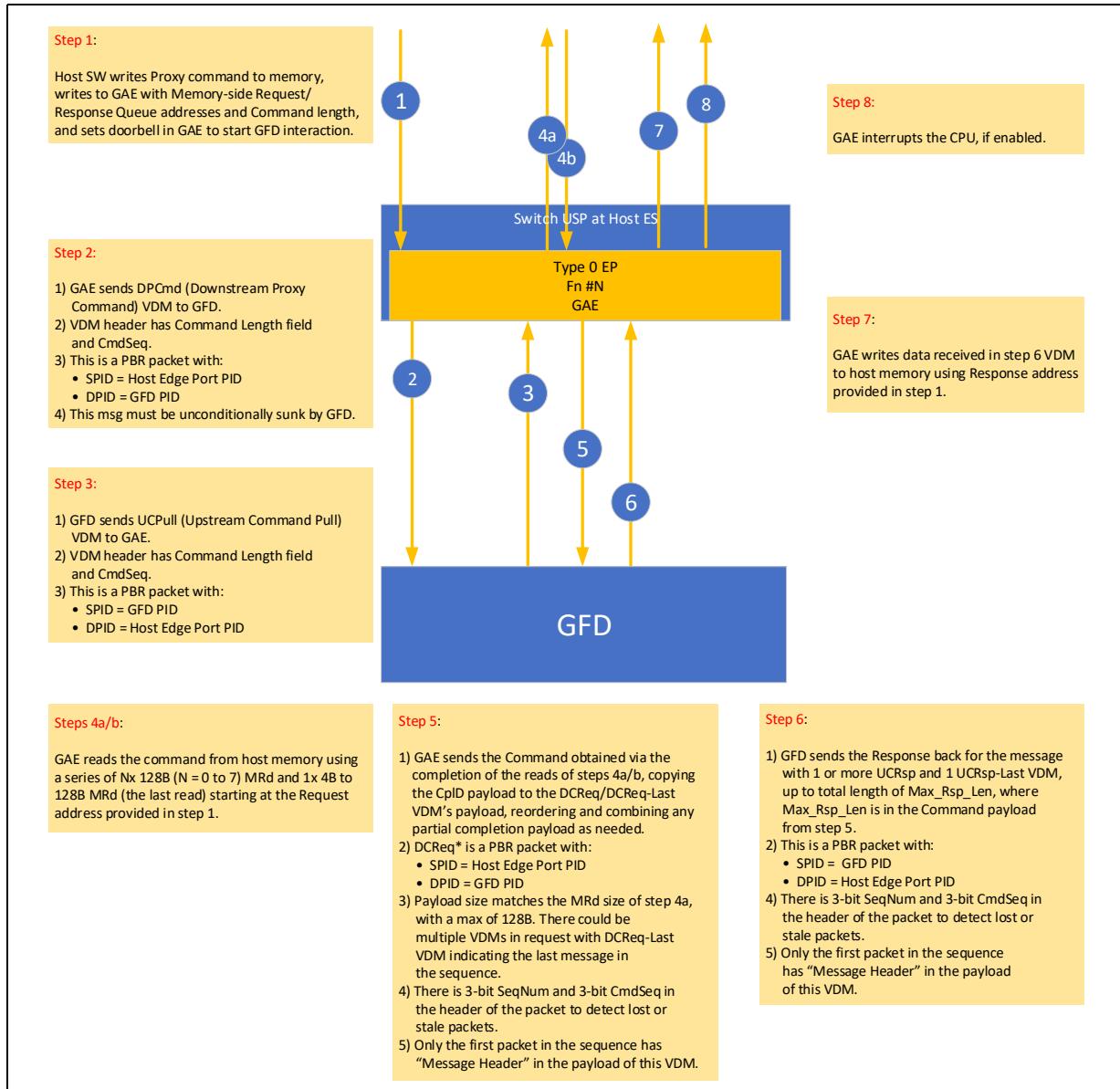
Message	USAR/ DSAR	CXL VDM Code	PBR Opcode	Destination	Payload (DWORDs)	Comment
Assert PERST#	DSAR	80h	0h	vUSP	0	Propagate fundamental reset downstream from vDSP.
Assert Reset	DSAR	80h	1h	vUSP	0	Propagate hot reset downstream from vDSP.
Deassert Reset	DSAR	80h	3h	vUSP	0	Propagate reset deassertion downstream from vDSP.
Link Up	DSAR	80h	4h	vDSP	0	Send upstream to vDSP, changing link state to L0 from detect.
PBR Link Partner Info	DSAR	90h	0h	Link Partner	16	Message with Data. Data saved in recipient.
DPCmd	DSAR	A0h	0	GFD	0	Downstream Proxy Command from GAE.
UCPull	DSAR	A1h	1	Host ES	0	Upstream command pull from GFD.
DCReq	DSAR	A0h	2	GFD	32	Downstream Command Request from GAE.
DCReq-Last	DSAR	A0h	3	GFD	1 – 32	Last DCReq.
UCRsp	DSAR	A1h	4	Host ES	32	Upstream Completion Response from GFD.
UCRsp-Last	DSAR	A1h	5	Host ES	1 – 32	Last UCRsp.
UCRsp-Fail	DSAR	A1h	6	Host ES	0	Failed DCReq receipt.
GAM	DSAR	A1h	7	Host ES	8	GFD log to host.
DCReq-Fail	DSAR	A0h	8	GFD	0	Failed UCPull response.
RTUpdate	DSAR	A1h	10h	Host ES	1 – 8	CacheID bus update from Downstream ES.
RTUpdateAck	DSAR	A1h	12h	Downstream ES	0	Acknowledgment of RTUpdate from Host ES.
RTUpdateNak	DSAR	A1h	13h	Downstream ES	0	Nak for RTUpdate from Host ES.
CXL PM	DSAR	68h	0	Varies	4	CXL Power Management.
CXL Error	USAR	00h	0	Host	0	CXL Error.

Although they exist outside the PBR Fabric, CXL VDM Codes 00h and 68h are listed to show the complete CXL VDM mapping. Their VDM Header is defined by PCI-SIG and thus does not match the fields provided for a PBR VDM header. These two VDMs will pass through the PBR Fabric using a hierarchical route and using the VDM Header originally defined in [Section 3.1.2](#) for CXL PM and in [Section 3.1.3](#) for CXL Error.

3.1.11.1 Host Management Transaction Flows of GFD

Figure 3-9 summarizes the Host Management Transaction Flows of GFD.

Figure 3-9. Host Management Transaction Flows of GFD



The Host ES has one GAE per host port. The GAE and GFD communicate via PID-routed VDMs.

Each GAE has an array of active messages, such that a host can communicate with multiple GFDs in parallel. Host software shall ensure that there is only one host-GFD management flow active per host-GFD pair.

The Host-to-GFD message flow consists of the following steps, after first storing the GFD command in host memory:

1. Host writes to GAE.
 - Writes pointer to GFD command in host memory (for UCPull read)
 - Writes pointer to write responses from GFD in host memory (for UCRsp data)
 - Writes command length
 - Writes CmdSeq
 - Write a mailbox command doorbell register (see [Section 8.2.8.4.4](#)), which causes the GAE to start the host management flow with step 2
2. Host ES creates CXL PBR VDM “DPCmd” with command length that targets the GFD PID and CmdSeq to identify the current command sequence.
 - This is an unsolicited message from the GFD point of view, and the GFD must be able to sink one such message for any supported RPID and drop any message from an unsupported RPID
3. GFD responds with a CXL PBR VDM “UCPull”, pulling the command for the indicated CmdSeq. The GFD response time may be delayed by responding to other doorbells from other RPIDs.
4. GAE converts CXL PBR VDM “UCPull” to one or more PCIe MRd TLPS.
 - a. GAE sends a series of MRds to read the command, starting at the address pointer supplied to the GAE in the Proxy GFD Management Command input payload. Each MRd size is a maximum of 128B. A command larger than 128B shall require multiple MRd to gather the full command. A total of (Nx) 128B MRd (with N from 0 to 7) and 1x (1B to 128B) MRd is needed to read any command of size up to 1024B.
 - b. The host completes each MRd with one or two CplD TLPS.
5. GAE gathers the read completion data in step 4b, re-ordering and combining partial completions as needed, to create a VDM payload. The GAE sends a series of DCReq/DCReq-Last VDMs with the completion data as VDM payload in the order that matches the series of MRd in step 4. The maximum payload for PBR VDMs is 128B (= 32 DWORDS).
 - Each VDM header contains the following:
 - An incrementing SeqNum, to allow detection of missing messages, starting with 0
 - A CmdSeq, to identify the current command for this Host – GFD thread
 - The last VDM in the sequence will be DCReq-Last.
 - VDMs before the last, if needed, will be DCReq.
 - The first VDM in the sequence shall start with a payload that matches the CCI Message header and payload as defined in [Section 7.6.3](#). Subsequent VDM’s payload shall contain only the remaining payload portion of the CCI Message and not repeat the header.
 - A failed command pull shall result in a DCReq-Fail VDM response instead of any DCReq and DCReq-Last.
6. GFD processes the command after it receives the last VDM (the “DCReq-Last”). The GFD shall send UCRsp/UCRsp-Last VDMs in response to the Host ES.
 - Each VDM header contains the following:
 - An incrementing SeqNum, to allow detection of missing messages, starting with 0
 - A CmdSeq, to identify the current command for this Host – GFD thread
 - The last VDM in the sequence will be UCRsp-Last
 - VDMs before the last, if needed, will be UCRsp

- If instead the GFD received DCReq-Fail, a UCRsp-Last shall be sent without processing the (incomplete) command
7. GAE converts “UCRsp” and “UCRsp-Last” series to a series of MWr with the payload the same as the VDM payload. The MWr address is supplied to the GAE in the Proxy Command input payload.
- After the UCRsp-Last payload is written, the GAE mailbox control doorbell described in [Section 8.2.8.4.4](#) is cleared. If the MB Doorbell Interrupt is set, an interrupt will be sent by the GAE to the host.

If at any point the GAE disables the GFD access vector, any incoming UCRsp/UCRsp-Last VDMs from the disabled GFD shall be dropped, and any UCPull shall be replied to with a DML-Fail VDM.

The CmdSeq is used to synchronize the GAE and GFD to be working on the same command sequence. A host may issue a subsequent command with a different CmdSeq to abort a prior command that may not have completed the sequence. Both the GAE (step 3 UCPull and step 6 UCRsp*) and the GFD (step 2 DPCmd and step 5 DCReq*) shall check that the command sequence number is the current one for communication with the partner PID (GAE uses GFD’s PID, and GFD uses GAE’s PID). Any stale command sequence VDM will be dropped and logged. The GFD will always update its current CmdSeq[GAE’s PID] based on the value received in step 2 DPCmd.

The host management flow of a GFD also includes an asynchronous notification from the GFD to inform the host of events in the GFD, using a GAM (GFD Async Message) VDM. The GAM has a payload of up to 32B (8 DWORDs). This payload passes through the GAE to write to an address supplied to the GAE in the Proxy Command input payload. Each GAM write starts at a 32B-aligned offset.

All CXL.io TLPS sent over a PBR link shall have a PTH. The host management flow of GFD VDMs have PTH fields restricted to the following values:

- SPID =
 - From Host ES: Host Edge Port PID
 - From GFD: GFD PID
- DPID =
 - To GFD: GFD PID
 - To Host ES: Host Edge Port PID
- DSAR flag = 1

VDM header fields for GFD Message VDMs:

- CXL VDM code of A0h (to GFD) or A1h (to Host ES)
- PBR Opcode 0 – 8 to indicate the particular VDM
- **CmdSeq**: Holds the command sequence number issued initially in step 2, DPCmd.
- **SeqLen**: Holds the length in DWORDs of the subsequent stage DCReq sequence or UCRsp sequence
- **SeqNum**: Holds the sequence number for multi-VDM command or multi-VDM response, starting at 0h and wrapping after 7h back to 0h
- A list of all the CXL VDMs is provided in [Table 3-8](#)

3.1.11.2 Downstream Proxy Command (DPCmd) VDM

Initiating a Proxy GFD Management Command on the GAE shall cause the Host ES to create a ‘DPCmd’ VDM that targets the GFD.

The 'DPCmd' VDM fields are as follows. PTH holds:

- SPID = Host Edge Port PID
- DPID = GFD PID
- DSAR flag = 1

VDM header fields for 'DPCmd' VDMs:

- CXL VDM Code of A0h
- PBR Opcode 0 (DPCmd) DPC
- **CmdSeq**: Current host management command sequence
- **SeqLen**: Command Length (DWORDs, 1 to 256 DWORD max — value of 00h is 256 DWORDs)

A 'DPCmd' VDM is an unsolicited message from the GFD point of view. A GFD must be able to successfully record every 'DPCmd' VDM that it receives, up to one from each of its registered RPIDs. The 'DPCmd' VDM is a message without data. The SeqLen part of the VDM header holds the command length that will be pulled by the 'UCPull'.

Only one active DPCmd at a time is allowed per Host Edge Port PID/GFD PID pair. A DPCmd is considered active until the GAE receives a UCRsp-Last VDM in response to a DPCmd.

A GFD should receive only a single active DPCmd per Host PID. If a second DPCmd is received from the same Host PID, the first shall be silently aborted. If a second DPCmd is received before the current DPCmd completes, the GFD updates its current command sequence to the new DPCmd CmdSeq and aborts the prior command sequence.

3.1.11.3 Upstream Command Pull (UCPull) VDM

A GFD shall issue a 'UCPull' VDM when it services a received 'DPCmd' VDM. A single UCPull shall be issued for each DPCmd received, with its command length matching the command length of the DPCmd.

The 'UCPull' VDM fields are as follows. PTH holds:

- SPID = GFD PID
- DPID = Host Edge Port PID
- DSAR flag = 1

VDM header fields for 'UCPull' VDMs:

- CXL VDM Code of A1h
- PBR Opcode 1 (UCPull)
- **CmdSeq**: Matching current command sequence from DPCmd
- **SeqLen**: Length of command to pull (1 to 256 DWORDs)

A GAE must be able to successfully service every 'UCPull' VDM that it receives. The GAE advertises a maximum number of outstanding proxy threads, which defines the maximum number of UCPull VDMs that it would need to track.

A 'UCPull' is a message without data and consists of a single VDM (there is no sequence of UCPulls). The SeqLen field in the VDM header contains the targeted command length to pull from host memory via the GAE. The CmdSeq contains the current command sequence.

The CmdSeq should be checked to match the current command sequence for the GFD thread; if the CmdSeq does not match, the UCPull is dropped and logged. The UCPull SeqLen shall exactly match the DPCmd SeqLen. The GAE shall issue one or more MRds to pull the command. The last MRd may be 1 to 32 DWORDs. Any prior MRd shall be for exactly 32 DWORDs. The sum of all the MRd lengths shall be the SeqLen.

3.1.11.4 Downstream Command Request (DCReq, DCReq-Last, DCReq-Fail) VDMs

When the Host ES reads the command from host memory in response to a UCPull VDM, the completions for those reads are then conveyed to the GFD over a sequence of zero or more DCReq VDMs plus exactly one DCReq-Last VDM. Each completion payload is copied directly to the VDM payload. The Host ES is responsible for combining any partial completions together to make a single payload for the VDM. Each MRd issued to the host will result, when the CplDs for that MRd all return, in a single DCReq VDM or DCReq-Last VDM. The order of the DCReq/DCReq-Last VDMs shall match the order of the MRd. The DCReq-Last VDM represents the end of the Downstream Command Request series. Any missing DCReq/DCReq-Last VDMs in the sequence should result in the GFD failing the command.

The 'DCReq' / 'DCReq-Last' / 'DCReq-Fail' VDM fields are as follows. PTH holds:

- SPID = Host Edge Port PID
- DPID = GFD PID
- DSAR flag = 1

VDM header fields for 'DCReq' / 'DCReq-Last' / 'DCReq-Fail' VDMs:

- CXL VDM Code of A0h
- PBR Opcode 2 (DCReq) / PBR Opcode 3 (DCReq-Last) / PBR Opcode 8 (DCReq-Fail)
- **CmdSeq**: Command sequence to be checked by Receiver
- **SeqLen**: Defined only for DCReq-Last, holds the expected length of the Response in the next step (UCRsp); 0 for DCReq and DCReq-Fail
- **SeqNum**:
 - Defined for all DCReq and DCReq-Last VDMs, initialized to 0 at the start of the sequence and incremented for each subsequent VDM; 0 of DCReq-Fail
 - Holds the DCReq* VDM sequence number, starting at 0h and incrementing for each subsequent VDM

Any 'DCReq' VDM shall have a payload of exactly 32 DWORDs. A short command may not have any DCReq VDMs. Every Downstream Command Request sequence shall have exactly one DCReq-Last VDM. The DCReq-Last VDM can have any payload length from 1 to 32 DWORDs.

The DCReq-Last VDM header has SeqLen defined to indicate the next step UCRsp length in DWORDs.

The GFD that is receiving the DCReq* VDMs checks that the CmdSeq matches its current command sequence for that Host Edge Port PID; if the CmdSeq does not match, the DCReq* VDM is dropped and logged.

The DCReq-Fail VDM shall be sent if CmdSeq is correct but the PID of the GFD is not enabled in the host's GAE's GMV and a UCPull from that GFD is received.

3.1.11.5 Upstream Command Response (UCRsp, UCRsp-Last, UCRsp-Fail) VDMs

When a GFD receives a ‘DCReq-Last’ VDM, the GFD checks that the CmdSeq is the current command sequence for that Host Edge Port PID and that all DCReq VDMs and DCReq-Last VDM were received.

If either check fails, the command sequence stops. If all DCReq are not received, as determined by a missing SeqNum, a UCRsp-Fail VDM shall be sent.

If the checks pass, a GFD will issue a ‘UCRsp’ VDM after the GFD processes the earlier-received ‘DCReq-Last’ VDM. The total length of the Response is dictated by the SeqLen provided in the ‘DCReq-Last’ SeqLen in the VDM header.

There will be zero or more ‘UCRsp’ VDMs and always exactly one ‘UCRsp-Last’ VDM, where the ‘UCRsp-Last’ VDM ends the sequence and is sent last. The sum of the DWORDs of response will match the length requested in the ‘DCReq-Last’ VDM SeqLen field. Each ‘UCRsp’ VDM will be 32 DWORDs. The ‘UCRsp-Last’ VDM can be from 1 to 32 DWORDs. Each ‘UCRsp’ / ‘UCRsp-Last’ VDM in the sequence increments the sequence number, starting at 0 and wrapping from 7 back to 0. Any missing UCRsp VDM in the sequence should result in a response error being flagged in the GAE.

The ‘UCRsp’ / ‘UCRsp-Last’ / ‘UCRsp-Fail’ VDM fields are as follows. PTH holds:

- SPID = GFD PID
- DPID = Host Edge Port PID
- DSAR flag = 1

VDM header fields for ‘UCRsp’ / ‘UCRsp-Last’ / ‘UCRsp-Fail’ VDMs:

- CXL VDM Code of A1h
- PBR Opcode 4 (UCRsp) / PBR Opcode 5 (UCRsp-Last) / PBR Opcode 6 (UCRsp-Fail)
- **CmdSeq**: Current command sequence
- **SeqNum**: Holds the UCRsp* VDM sequence number, starting at 0h and incrementing for each subsequent VDM; 0 for UCRsp-Fail

3.1.11.6 GFD Async Message (GAM) VDM

The GAM VDM is used to notify a host of some issue with its use of the GFD. The payload of the GAM should pass through to the host GAM buffer at a 32B-aligned offset. The GAM payload is fixed at 8 DWORDs, as shown in [Table 3-10](#).

Table 3-10. GAM VDM Payload

Byte	+3								+2								+1								+0																																							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0																																
0	Memory Req Opcode								GFD Notification								Reserved				PID[11:0]																																											
4	Reserved				RPID[11:0]																Flags[15:0]																																											
8																																																																
12																																																																
16																																																																
20																																																																
24																																																																
28																																																																

The GAM payload shall be written by the GAE endpoint to the GAE's circular GAM Buffer as described in [Section 7.7.2.7](#).

The 'GAM' VDM fields are as follows. PTH holds:

- SPID = GFD PID
- DPID = Host Edge Port PID
- DSAR flag = 1

VDM header fields for 'GAM' VDMs:

- CXL VDM Code of A1h
- PBR Opcode 7 (GAM)

3.1.11.7 Route Table Update (RTUpdate) VDM

On a PBR link, the CacheID of a CXL.cache message is replaced with a PID. A table is needed at both the Host ES and Downstream ES to swap between PID and CacheID.

A VDM from the Downstream ES is needed to convey the information, a list of pairs of (PID and CacheID), to the Host ES with a maximum of 16 pairs, corresponding to 8 DWORDs. The flow to the RTUpdate VDM is described in more detail in [Section 7.7.12.5](#).

An RTUpdate VDM is sent from Downstream ES firmware to Host ES firmware. The DPID is the Host PID, allowing for a route to the Host ES. However, the Host ES ingress shall trap on the CXL VDM Code of A1h and handle the VDM in the Host ES.

The 'RTUpdate' VDM fields are as follows. PTH holds:

- SPID = vUSP's fabric port's PID
- DPID = Host Edge Port PID
- DSAR flag = 1

VDM header fields for 'RTUpdate' VDMs:

- CXL VDM Code of A1h
- PBR Opcode 10h (RTUpdate)

[Table 3-11](#) shows the RTUpdate VDM payload format. Note that a value of FFFh for DSP_PID in the payload indicates that the PID is invalid and hence the PID to CacheID information pair needs to be discarded.

Table 3-11. RTUpdate VDM Payload

Byte	+3								+2								+1								+0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	CacheID[3:0]				DSP PID[11:0]								CacheID[3:0]				DSP PID[11:0]								CacheID[3:0]				DSP PID[11:0]			
4	CacheID[3:0]				DSP PID[11:0]								CacheID[3:0]				DSP PID[11:0]								CacheID[3:0]				DSP PID[11:0]			
...	...																															
28	CacheID[3:0]				DSP PID[11:0]								CacheID[3:0]				DSP PID[11:0]								CacheID[3:0]				DSP PID[11:0]			

3.1.11.8 Route Table Update Response (RTUpdateAck, RTUpdateNak) VDMs

The response to the RTUpdate VDM shall be one of the following:

- RTUpdateAck VDM if the update is successful

- RTUpdateNak VDM if the update is unsuccessful
- RTUpdateNak VDM if a VDM in the sequence was lost

The DPID is set to the vUSP's fabric port's PID, which routes the RTUpdateAck VDM back to the Downstream ES. However, the Downstream ES ingress shall trap on the CXL VDM Code of A1h to direct the VDM to switch firmware.

The Downstream ES, upon receipt of the RTUpdateAck VDM, shall set the commit complete bit in the CacheID table.

The 'RTUpdateAck' / 'RTUpdateNak' VDM fields are as follows. PTH holds:

- SPID = Host Edge Port PID
- DPID = vUSP's fabric port's PID
- DSAR flag = 1

VDM header fields for 'RTUpdateAck' / 'RTUpdateNak' Response VDMs are as follows:

- CXL VDM Code of A1h
- PBR Opcode 12h (RTUpdateAck) / PBR Opcode 13h (RTUpdateNak)

3.2

CXL.cache

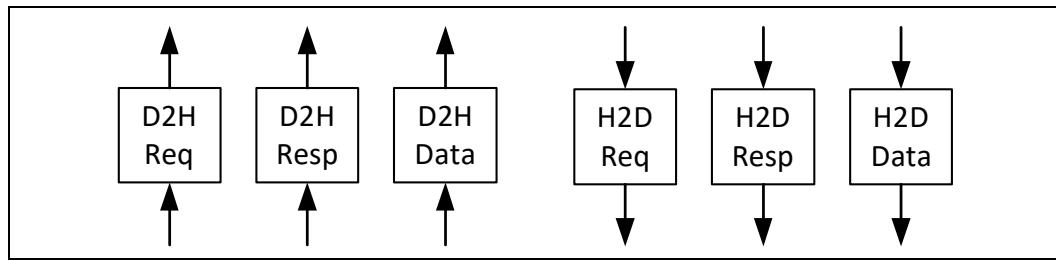
3.2.1

Overview

The CXL.cache protocol defines the interactions between the device and host as a number of requests that each have at least one associated response message and sometimes a data transfer. The interface consists of three channels in each direction: Request, Response, and Data. The channels are named for their direction, D2H for device to host and H2D for host to device, and the transactions they carry, Request, Response, and Data as shown in [Figure 3-10](#). The independent channels allow different kinds of messages to use dedicated wires and achieve both decoupling and a higher effective throughput per wire.

A D2H Request carries new requests from the Device to the Host. The requests typically target memory. Each request will receive zero, one, or two responses and at most one 64-byte cacheline of data. The channel may be back pressured without issue. D2H Response carries all responses from the Device to the Host. Device responses to snoops indicate the state the line was left in the device caches, and may indicate that data is being returned to the Host to the provided data buffer. They may still be blocked temporarily for link layer credits. D2H Data carries all data and byte enables from the Device to the Host. The data transfers can result either from implicit (as a result of snoop) or explicit write-backs (as a result of cache capacity eviction). A full 64-byte cacheline of data is always transferred. D2H Data must make progress or deadlocks may occur. D2H Data may be temporarily blocked for link layer credits, but must not require any other D2H transaction to complete to free the credits.

An H2D Request carries requests from the Host to the Device. These are snoops to maintain coherency. Data may be returned for snoops. The request carries the location of the data buffer to which any returned data should be written. H2D Requests may be back pressured for lack of device resources; however, the resources must free up without needing D2H Requests to make progress. H2D Response carries ordering messages and pulls for write data. Each response carries the request identifier from the original device request to indicate where the response should be routed. For write data pull responses, the message carries the location where the data should be written. H2D Responses can only be blocked temporarily for link layer credits. H2D Data delivers the data for device read requests. In all cases a full 64-byte cacheline of data is transferred. H2D Data transfers can only be blocked temporarily for link layer credits.

Figure 3-10. CXL.cache Channels

3.2.2 CXL.cache Channel Description

3.2.2.1 Channel Ordering

In general, all the CXL.cache channels must work independently of one another to ensure that forward progress is maintained. For example, because requests from the device to the Host to a given address X will be blocked by the Host until it collects all snoop responses for this address X, linking the channels would lead to deadlock.

However, there is a specific instance where ordering between channels must be maintained for the sake of correctness. The Host needs to wait until Global Observation (GO) messages, sent on H2D Response, are observed by the device before sending subsequent snoops for the same address. To limit the amount of buffering needed to track GO messages, the Host assumes that GO messages that have been sent over CXL.cache in a given cycle cannot be passed by snoops sent in a later cycle.

For transactions that have multiple messages on a single channel with an expected order (e.g., WritePull and GO for WrInv) the Device/Host must ensure they are observed correctly using serializing messages (e.g., the Data message between WritePull and GO for WrInv as shown in [Figure 3-14](#)).

3.2.2.2 Channel Crediting

To maintain the modularity of the interface no assumptions can be made on the ability to send a message on a channel because link layer credits may not be available at all times. Therefore, each channel must use a credit for sending any message and collect credit returns from the receiver. During operation, the receiver returns a credit whenever it has processed the message (i.e., freed up a buffer). It is not required that all credits are accounted for on either side, it is sufficient that credit counter saturates when full. If no credits are available, the sender must wait for the receiver to return one.

[Table 3-12](#) describes which channels must drain to maintain forward progress and which can be blocked indefinitely. Additionally, [Table 3-12](#) defines a summary of the forward progress and crediting mechanisms in CXL.cache, but this is not the complete definition. See [Section 3.4](#) for the complete set of the ordering rules that are required for protocol correctness and forward progress.

Table 3-12. CXL.cache Channel Crediting Summary

Channel	Forward Progress Condition	Blocking Condition	Description
D2H Request (Req)	Credited to Host	Can be blocked by all other message classes in CXL.cachemem.	Needs Host buffer, could be held by earlier requests
D2H Response (Rsp)	Pre-allocated	Temporary link layer back pressure is allowed. Host may block waiting for H2D Response to drain.	Headed to specified Host buffer
D2H Data	Pre-allocated	Temporary link layer back pressure is allowed. Host may block for H2D Data to drain.	Headed to specified Host buffer
H2D Request (Req)	Credited to Device	Must make progress. Temporary back pressure is allowed.	May be back pressured temporarily due to lack of availability of D2H Response or D2H Data credits
H2D Response (Rsp)	Pre-allocated	Link layer only, must make progress. Temporary back pressure is allowed.	Headed to specified device buffer
H2D Data	Pre-allocated	Link layer only, must make progress. Temporary back pressure is allowed.	Headed to specified device buffer

3.2.3 CXL.cache Wire Description

The definition of each of the fields for each CXL.cache Channel is provided below. Each message will support 3 variants: 68B Flit, 256B Flit, and PBR Flit. The use of each of these will be negotiated in the physical layer for each link as defined in [Chapter 6.0](#).

3.2.3.1 D2H Request

Table 3-13. CXL.cache - D2H Request Fields (Sheet 1 of 2)

D2H Request	Width (Bits)			Description
	68B Flit	256B Flit	PBR Flit	
Valid	1			The request is valid.
Opcode	5			The opcode specifies the operation of the request. Details in Table 3-22 .
CQID	12			<p>Command Queue ID: The CQID field contains the ID of the tracker entry that is associated with the request. When the response and data are returned for this request, the CQID is sent in the response or data message indicating to the device which tracker entry originated this request.</p> <p>IMPLEMENTATION NOTE</p> <p>CQID usage depends on the round-trip transaction latency and desired bandwidth. A 12-bit ID space allows for 4096 outstanding requests which can saturate link bandwidth for a x16 link at 64 GT/s with average latency of up to 1 us¹.</p>
NT	1			For cacheable reads, the NonTemporal bit is used as a hint to indicate to the host how it should be cached. Details in Table 3-14 .
CacheID	0	4	0	Logical CacheID of the source of the message. Not supported in 68B flit messages. Not applicable in PBR messages where DPID infers this field.

Table 3-13. CXL.cache - D2H Request Fields (Sheet 2 of 2)

D2H Request	Width (Bits)			Description
	68B Flit	256B Flit	PBR Flit	
Address[51:6]	46			Carries the physical address of coherent requests.
SPID	0	12		Source PID
DPID	0	12		Destination PID
RSVD	14	7		
Total	79	76	96	

1. Formula assumed in this calculation is: "Latency Tolerance in ns" = "number of Requests" * (64B per Request) / "Peak Bandwidth in GB/s". Assuming a peak bandwidth of 256 GB/s (raw bidirectional bandwidth of a x16 CXL port at 64 GT/s) results in a latency tolerance of 1024 ns.

Table 3-14. Non Temporal Encodings

NonTemporal	Definition
0	Default behavior. This is Host implementation specific.
1	Requested line should be moved to Least Recently Used (LRU) position

3.2.3.2 D2H Response

Table 3-15. CXL.cache - D2H Response Fields

D2H Response	Width (Bits)			Description
	68B Flit	256B Flit	PBR Flit	
Valid	1			The response is valid.
Opcode	5			The opcode specifies the what kind of response is being signaled. Details in Table 3-25 .
UQID	12			Unique Queue ID: This is a reflection of the UQID sent with the H2D Request and indicates which Host entry is the target of the response.
DPID	0	12		Destination PID
RSVD	2	6		
Total	20	24	36	

3.2.3.3 D2H Data

Table 3-16. CXL.cache - D2H Data Header Fields

D2H Data Header	Width (Bits)			Description
	68B Flit	256B Flit	PBR Flit	
Valid	1			The Valid signal indicates that this is a valid data message.
UQID	12			Unique Queue ID: This is a reflection of the UQID sent with the H2D Response and indicates which Host entry is the target of the data transfer.
ChunkValid	1	0		In case of a 32B transfer on CXL.cache, this indicates what 32-byte chunk of the cacheline is represented by this transfer. If not set, it indicates the lower 32B and if set, it indicates the upper 32B. This field is ignored for a 64B transfer.
Bogus	1			The Bogus bit indicates that the data associated with this evict message was returned to a snoop after the D2H request was sent from the device, but before a WritePull was received for the evict. This data is no longer the most current, so it should be dropped by the Host.
Poison	1			The Poison bit is an indication that this data chunk is corrupted and should not be used by the Host.
BEP	0	1		Byte-Enables Present: Indication that 5 data slots are included in the message where the 5 th data slot carries the 64-bit Byte Enables. This field is carried as part of the Flit header bits in 68B Flit mode.
DPID	0		12	Destination PID
RSVD	1	8		
Total	17	24	36	

3.2.3.3.1 Byte Enables (68B Flit)

In 68B Flit mode, the presence of data byte enables is indicated in the flit header, but only when one or more of the byte enable bits has a value of 0. In that case, the byte enables are sent as a data chunk as described in [Section 4.2.2](#).

3.2.3.3.2 Byte-Enables Present (256B Flit)

In 256B Flit mode, a BEP (Byte-Enables Present) bit is included with the message header that indicates BE slot is included at the end of the message. The Byte Enable field is 64 bits wide and indicates which of the bytes are valid for the contained data.

3.2.3.4 H2D Request

Table 3-17. CXL.cache – H2D Request Fields (Sheet 1 of 2)

H2D Request	Width (Bits)			Description
	68B Flit	256B Flit	PBR Flit	
Valid	1			The Valid signal indicates that this is a valid request.
Opcode	3			The Opcode field indicates the kind of H2D request. Details in Table 3-26 .
Address[51:6]	46			The Address field indicates which cacheline the request targets.

Table 3-17. CXL.cache – H2D Request Fields (Sheet 2 of 2)

H2D Request	Width (Bits)			Description
	68B Flit	256B Flit	PBR Flit	
UQID	12			Unique Queue ID: This indicates which Host entry is the source of the request.
CacheID	0	4	0	Logical CacheID of the destination of the message. Value is assigned by Switch edge ports and not observed by the device. Host implementation may constrain the number of encodings that the Host can support. Not applicable with PBR messages where DPID infers this field.
SPID	0			Source PID
DPID	0			Destination PID
RSVD	2	6		
Total	64	72	92	

3.2.3.5 H2D Response

Table 3-18. CXL.cache - H2D Response Fields

H2D Response	Width (Bits)			Description
	68B Flit	256B Flit	PBR Flit	
Valid	1			The Valid bit 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 3-27 .
RspData	12			The response Opcode determines how the RspData field is interpreted as shown in Table 3-27 . Thus, depending on Opcode, it can either contain the UQID or the MESI information in bits [3:0] as shown in Table 3-20 .
RSP_PRE	2			RSP_PRE carries performance monitoring information. Details in Table 3-19 .
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.
CacheID	0	4	0	Logical CacheID of the destination of the message. This value is returned by the host based on the CacheID sent in the D2H request. Not applicable with PBR messages where DPID infers this field.
DPID	0			Destination PID
RSVD	1	5		
Total	32	40	48	

Table 3-19. RSP_PRE Encodings

RSP_PRE[1:0]	Response
00b	Host Cache Miss to Local CPU socket memory
01b	Host Cache Hit
10b	Host Cache Miss to Remote CPU socket memory
11b	Reserved

Table 3-20. Cache State Encoding for H2D Response

Cache State	Encoding
Invalid (I)	0011b
Shared (S)	0001b
Exclusive (E)	0010b
Modified (M)	0110b
Error (Err) ¹	0100b

1. Covers error conditions not covered by poison such as errors in coherence resolution.

3.2.3.6 H2D Data

Table 3-21. CXL.cache - H2D Data Header Fields

H2D Data Header	Width (Bits)			Description
	68B Flit	256B Flit	PBR Flit	
Valid	1			The Valid bit indicates that this is a valid data to the device.
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 data transfer.
ChunkValid	1	0		In case of a 32B transfer on CXL.cache, this indicates what 32-byte chunk of the cacheline is represented by this transfer. If not set, it indicates the lower 32B and if set, it indicates the upper 32B. This field is ignored for a 64B transfer.
Poison	1			The Poison bit indicates to the device that this data is corrupted and as such should not be used.
GO-Err	1			The GO-ERR bit indicates to the agent that this data is the result of an error condition and should not be cached or provided as response to snoops. Covers error conditions not covered by poison such as errors in coherence resolution.
CacheID	0	4	0	Logical CacheID of the destination of the message. Host and switch must support this field to set a nonzero value. Not applicable in PBR messages where DPID infers this field.
DPID	0		12	Destination PID
RSVD	8	9		
Total	24	28	36	

3.2.4 CXL.cache Transaction Description

3.2.4.1 Device-attached Memory Flows for HDM-D/HDM-DB

When a CXL Type 2 device exposes memory to the host using Host-managed Device Memory Device-Coherent (HDM-D/HDM-DB), the device is responsible to resolve coherence of HDM between the host and device. CXL defines two protocol options for this:

- CXL.cache Requests which is used for HDM-D
- CXL.mem Back-Invalidate Snoop (BISnp) which is used with HDM-DB

Endpoint devices supporting 256B Flit mode must support BISnp mechanism and can optionally use CXL.cache mechanism when connected to a host that has only 68B flit mode. When using CXL.cache, the host detects the address as coming from the device that owns the region which triggers the special flow that returns Mem*Fwd, in most cases, as captured in [Table 3-24](#).

3.2.4.2 Device to Host Requests

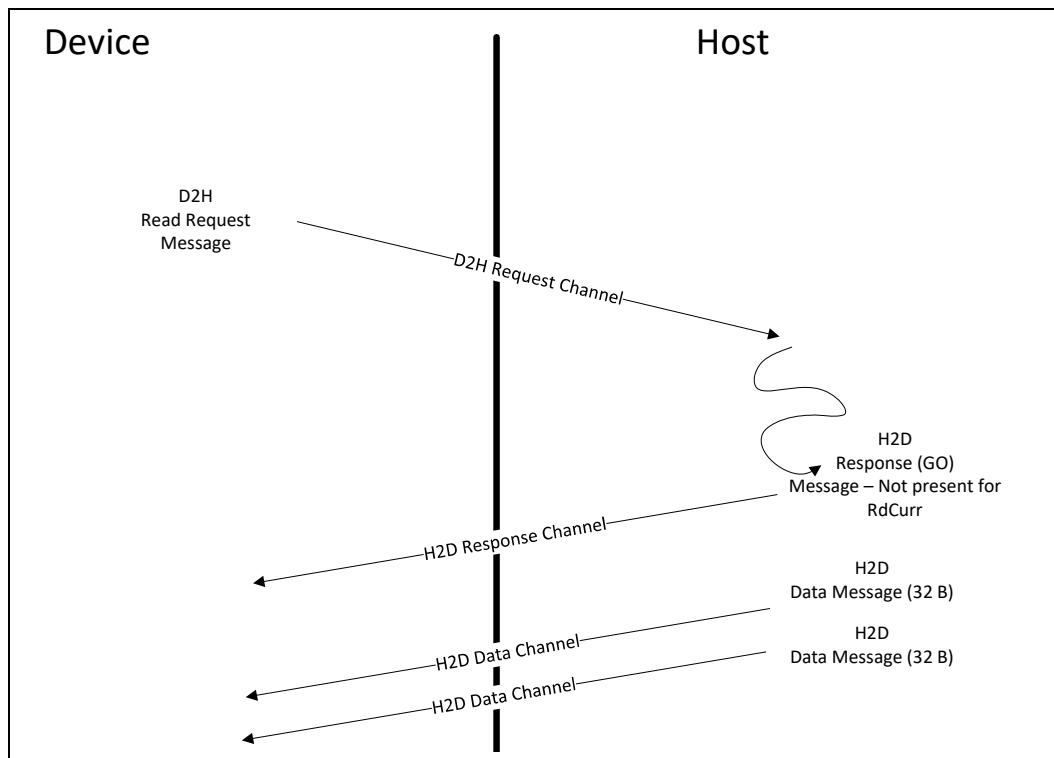
3.2.4.2.1 Device to Host (D2H) CXL.cache Request Semantics

For device to Host requests, there are four different semantics: CXL.cache Read, CXL.cache Read0, CXL.cache Read0/Write, and CXL.cache Write. All device to Host CXL.cache transactions fall into one of these four semantics, though the allowable responses and restrictions for each request type within a given semantic are different.

3.2.4.2.2 CXL.cache Read

CXL.cache Reads must have a D2H request credit and send a request message on the D2H CXL.cache request channel. CXL.cache Read requests require zero or one response (GO) message and data messages totaling a single 64-byte cacheline of data. Both the response, if present, and data messages are directed at the device tracker entry provided in the initial D2H request packet's CQID field. The device entry must remain active until all the messages from the Host have been received. To ensure forward progress, the device must have a reserved data buffer able to accept 64 bytes of data immediately after the request is sent. However, the device may temporarily be unable to accept data from the Host due to prior data returns not draining. Once both the response message and the data messages have been received from the Host, the transaction can be considered complete and the entry deallocated from the device.

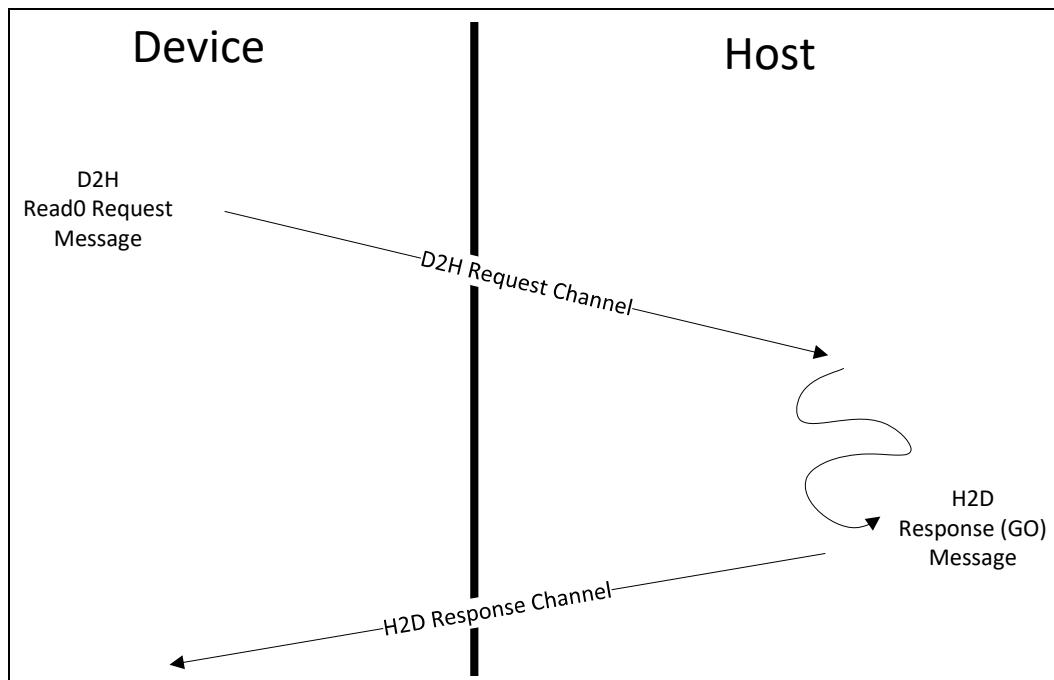
[Figure 3-11](#) shows the elements required to complete a CXL.cache Read. Note that the response (GO) message can be received before, after, or between the data messages.

Figure 3-11. CXL.cache Read Behavior

3.2.4.2.3 CXL.cache Read0

CXL.cache Read0 must have a D2H request credit and send a message on the D2H CXL.cache request channel. CXL.cache Read0 requests receive a response message but no data messages. The response message is directed at the device entry indicated in the initial D2H request message's CQID value. Once the GO message is received for these requests, they can be considered complete and the entry deallocated from the device. A data message must not be sent by the Host for these transactions. Most special cycles (e.g., CLFlush) and other miscellaneous requests fall into this category. See [Table 3-22](#) for details.

[Figure 3-12](#) shows the elements required to complete a CXL.cache Read0 transaction.

Figure 3-12. CXL.cache Read0 Behavior**3.2.4.2.4 CXL.cache Write**

CXL.cache Write must have a D2H request credit before sending a request message on the D2H CXL.cache request channel. Once the Host has received the request message, it is required to send a GO message and a WritePull message. The WritePull message is not required for CleanEvictNoData. The GO and the WritePull can be a combined message for some requests. The GO message must never arrive at the device before the WritePull, but it can arrive at the same time in the combined message. If the transaction requires posted semantics, then a combined GO-I/WritePull message can be used. If the transaction requires non-posted semantics, then WritePull is issued first followed by the GO-I when the non-posted write is globally observed.

Upon receiving the GO-I message, the device will consider the store done from a memory ordering and cache coherency perspective, giving up snoop ownership of the cacheline (if the CXL.cache message is an Evict).

The WritePull message triggers the device to send data messages to the Host totaling exactly 64 bytes of data, though any number of byte enables can be set.

A CXL.cache write transaction is considered complete by the device once the device has received the GO-I message, and has sent the required data messages. At this point the entry can be deallocated from the device.

The Host considers a write to be done once it has received all 64 bytes of data, and has sent the GO-I response message. All device writes and Evicts fall into the CXL.cache Write semantic.

See [Section 3.2.5.8](#) for more information on restrictions around multiple active write transactions.

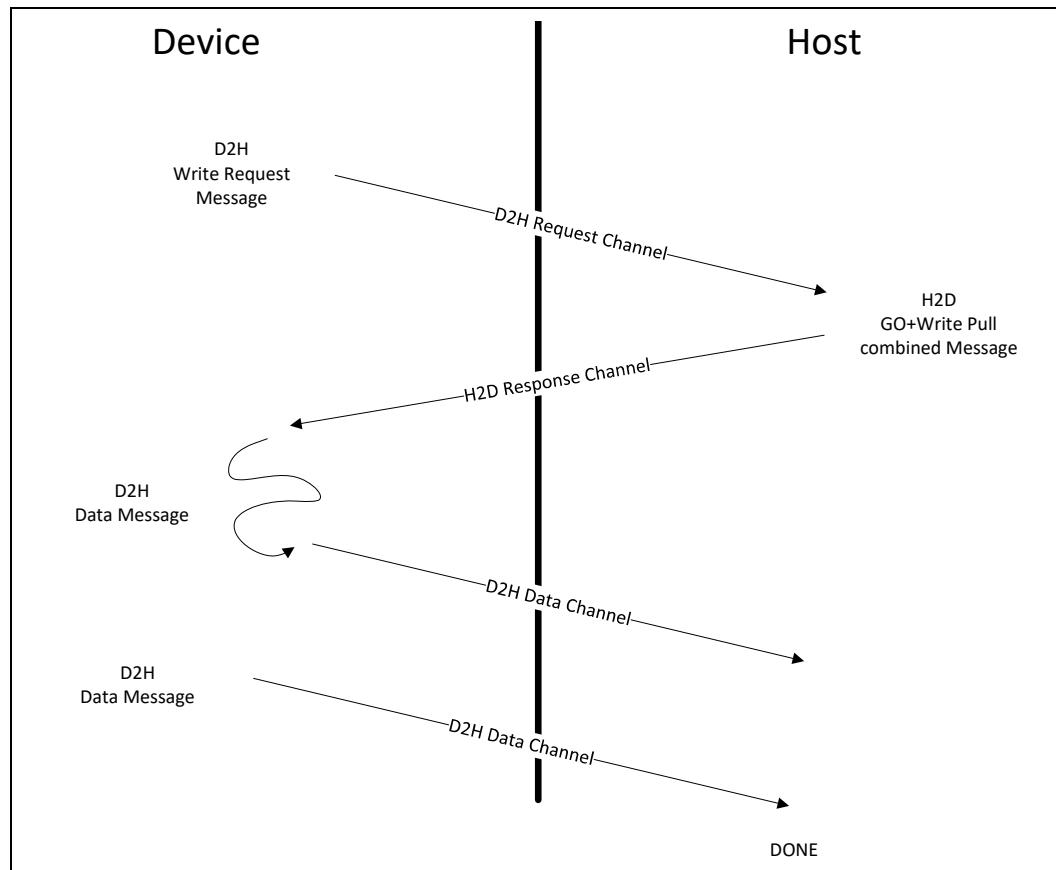
EVALUATION COPY

[Figure 3-13](#) shows the elements required to complete a CXL.cache Write transaction (that matches posted behavior). The WritePull (or the combined GO_WritePull) message triggers the data messages. There are restrictions on Snoops and WritePulls. See [Section 3.2.5.3](#) for more details.

[Figure 3-14](#) shows a case where the WritePull is a separate message from the GO (for example: strongly ordered uncacheable write).

[Figure 3-15](#) shows the Host FastGO plus ExtCmp responses for weakly ordered write requests.

Figure 3-13. CXL.cache Device to Host Write Behavior



EVALUATION COPY

Figure 3-14. CXL.cache WrInv Transaction

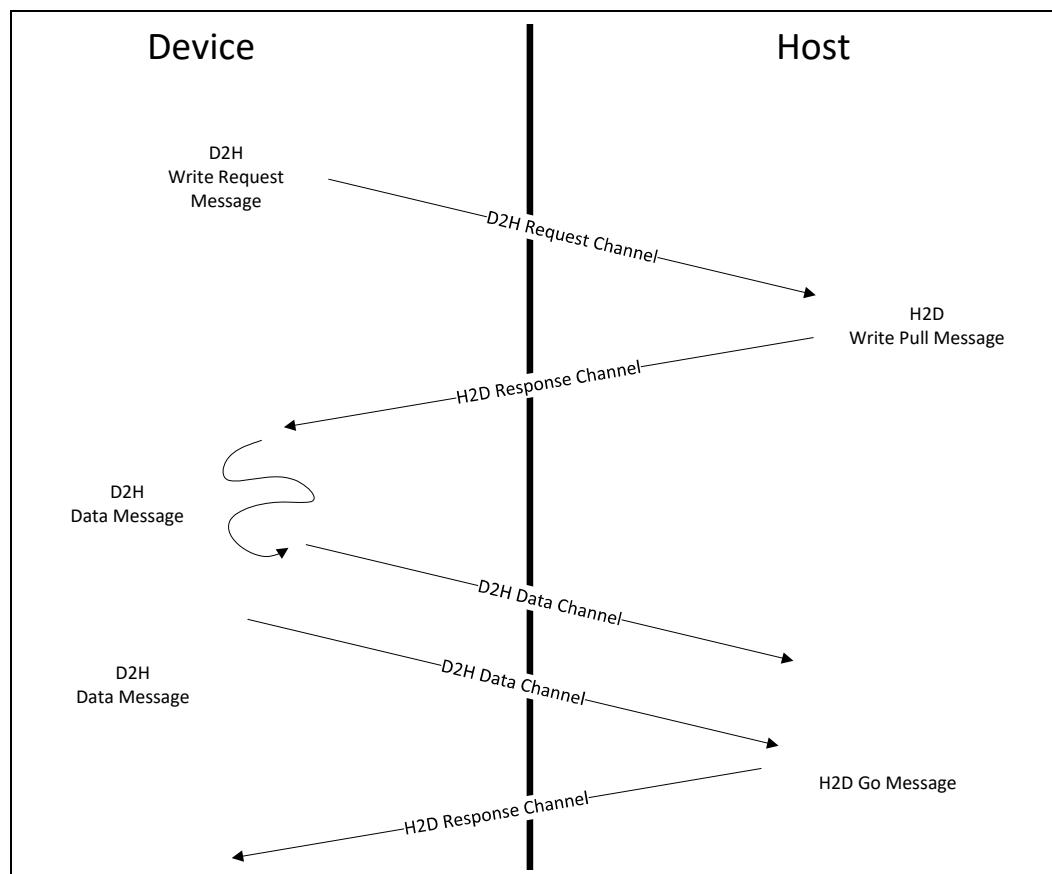
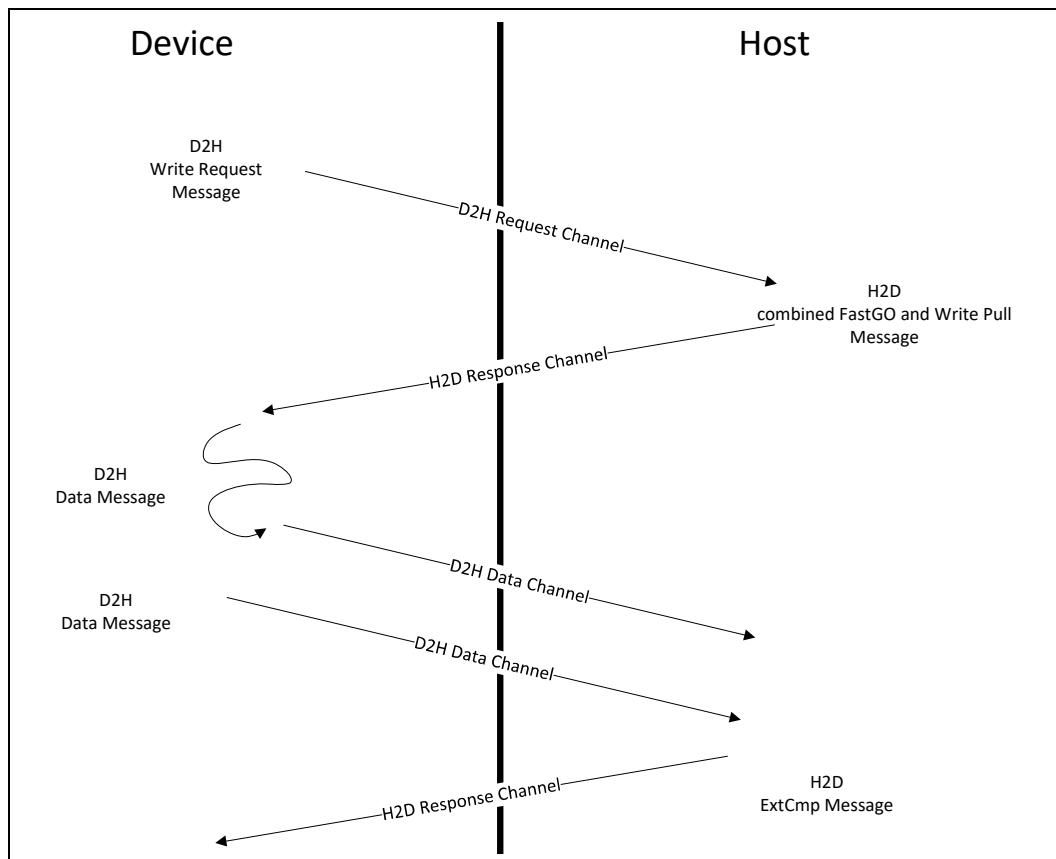


Figure 3-15. WOWrInv/F with FastGO/ExtCmp**3.2.4.2.5****CXL.cache Read0-Write Semantics**

CXL.cache Read0-Write requests must have a D2H request credit before sending a request message on the D2H CXL.cache request channel. Once the Host has received the request message, it is required to send one merged GO-I and WritePull message.

The WritePull message triggers the device to send the data messages to the Host, which together transfer exactly 64 bytes of data though any number of byte enables can be set.

A CXL.cache Read0-Write transaction is considered complete by the device once the device has received the GO-I message, and has sent the all required data messages. At this point the entry can be deallocated from the device.

The Host considers a Read0-Write to be done once it has received all 64 bytes of data, and has sent the GO-I response message. ItoMWr falls into the Read0-Write category.

EVALUATION COPY

Figure 3-16. CXL.cache Read0-Write Semantics

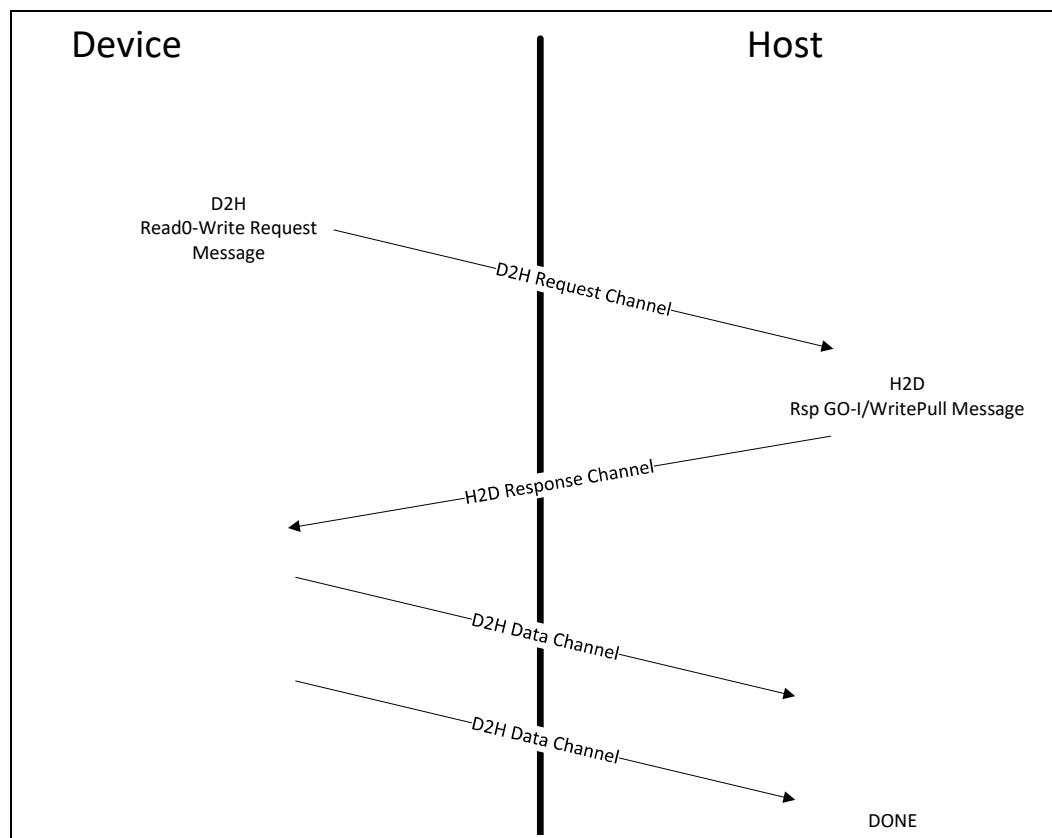


Table 3-22 summarizes all the opcodes that are available from the Device to the Host.

Table 3-22. CXL.cache – Device to Host Requests

CXL.cache Opcode	Semantic	Opcode
RdCurr	Read	0 0001b
RdOwn	Read	0 0010b
RdShared	Read	0 0011b
RdAny	Read	0 0100b
RdOwnNoData	Read0	0 0101b
ItoMWr	Read0-Write	0 0110b
WrCur	Read0-Write	0 0111b
CLFlush	Read0	0 1000b
CleanEvict	Write	0 1001b
DirtyEvict	Write	0 1010b
CleanEvictNoData	Write	0 1011b
WOWrInv	Write	0 1100b
WOWrInvF	Write	0 1101b
WrInv	Write	0 1110b
CacheFlushed	Read0	1 0000b

3.2.4.2.6 RdCurr

These are full cacheline read requests from the device for lines to get the most current data, but not change the existing state in any cache, including in the Host. The Host does not need to track the cacheline in the device that issued the RdCurr. RdCurr gets a data but no GO. The device receives the line in the Invalid state which means that the device gets one use of the line and cannot cache it.

3.2.4.2.7 RdOwn

These are full cacheline read requests from the device for lines to be cached in any writeable state. Typically, RdOwn request receives the line in Exclusive (GO-E) or Modified (GO-M) state. Lines in Modified state must not be dropped, and have to be written back to the Host.

Under error conditions, a RdOwn request may receive the line in Invalid (GO-I) or Error (GO-Err) state. Both return synthesized data of all 1s. The device is responsible for handling the error appropriately.

3.2.4.2.8 RdShared

These are full cacheline read requests from the device for lines to be cached in Shared state. Typically, RdShared request receives the line in Shared (GO-S) state.

Under error conditions, a RdShared request may receive the line in Invalid (GO-I) or Error (GO-Err) state. Both will return synthesized data of all 1s. The device is responsible for handling the error appropriately.

3.2.4.2.9 RdAny

These are full cacheline read requests from the device for lines to be cached in any state. Typically, RdAny request receives the line in Shared (GO-S), Exclusive (GO-E) or Modified (GO-M) state. Lines in Modified state must not be dropped, and have to be written back to the Host.

Under error conditions, a RdAny request may receive the line in Invalid (GO-I) or Error (GO-Err) state. Both return synthesized data of all 1s. The device is responsible for handling the error appropriately.

3.2.4.2.10 RdOwnNoData

These are requests to get exclusive ownership of the cacheline address indicated in the address field. The typical response is Exclusive (GO-E).

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

Note:

A device that uses this command to write data must be able to update the entire cacheline or may drop the E-state if it is unable to perform the update. There is no support partial M-state data in a device cache. To perform a partial write in the device cache, the device must read the cacheline using RdOwn before merging with the partial write data in the cache.

3.2.4.2.11 ItoMWr

This command requests exclusive ownership of the cacheline address indicated in the address field and atomically writes the cacheline back to the Host. The device guarantees the entire line will be modified, so no data needs to be transferred to the

device. The typical response is GO_WritePull, which is sent once the request is granted ownership. The device must not retain a copy of the line. If a cache exists in the host cache hierarchy before memory, the data should be written there.

If an error occurs, then GO-Err-WritePull is sent instead. The device sends the data to the Host, which drops it. The device is responsible for handling the error as appropriate.

3.2.4.2.12 WrCur

The command behaves like the ItoMWr in that it atomically requests ownership of a cacheline and then writes a full cacheline back to the Fabric. However, it differs from ItoMWr in where the data is written. Only if the command hits in a cache will the data be written there; on a Miss, the data will be written directly to memory. The typical response is GO_WritePull once the request is granted ownership. The device must not retain a copy of the line.

If an error occurs, then GO-Err-WritePull is sent instead. The device sends the data to the Host, which drops it. The device is responsible for handling the error as appropriate.

Note:

In earlier revisions of the specification (CXL 2.0 and CXL 1.x), this command was called "MemWr", but this was a problem because that same message name is used in the CXL.mem protocol, so a new name was selected. The opcode and behavior are unchanged.

3.2.4.2.13 CLFlush

This is a request to the Host to invalidate the cacheline specified in the address field. The typical response is GO-I which is sent from the Host upon completion in memory.

However, the Host may keep tracking the cacheline in Shared state if the Core has issued a Monitor to an address belonging in the cacheline. Thus, the Device that exposes an HDM-D region must not rely on CLFlush/GO-I as a sufficient condition for which to flip a cacheline in the HDM-D region 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 to Device Bias mode to allow subsequent cacheline access 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.

3.2.4.2.14 CleanEvict

This is a request to the Host to evict a full 64-byte Exclusive cacheline from the device. Typically, CleanEvict receives GO-WritePull or GO-WritePullDrop. The response will cause the device to relinquish snoop ownership of the line. For GO-WritePull, the device will send the data as normal. For GO-WritePullDrop, the device simply drops the data.

Once the device has issued this command and the address is subsequently snooped, but before the device has received the GO-WritePull, the device must set the Bogus field in all D2H Data messages to indicate that the data is now stale.

CleanEvict requests also guarantee to the Host that the device no longer contains any cached copies of this line. Only one CleanEvict from the device may be pending on CXL.cache for any given cacheline address.

CleanEvict is only expected for a host-attached memory range of addresses. For a device-attached memory range, the equivalent operation can be completed internally within the device without sending a transaction to the Host.

3.2.4.2.15 DirtyEvict

This is a request to the Host to evict a full 64-byte Modified cacheline from the device. Typically, DirtyEvict receives GO-WritePull from the Host at which point the device must relinquish snoop ownership of the line and send the data as normal.

Once the device has issued this command and the address is subsequently snooped, but before the device has received the GO-WritePull, the device must set the Bogus field in all D2H Data messages to indicate that the data is now stale.

DirtyEvict requests also guarantee to the Host that the device no longer contains any cached copies of this line. Only one DirtyEvict from the device may be pending on CXL.cache for any given cacheline address.

In error conditions, a GO-Err-WritePull is received. The device sends the data as normal, and the Host drops it. The device is responsible for handling the error as appropriate.

DirtyEvict is only expected for host-attached memory address ranges. For device-attached memory range, the equivalent operation can be completed internally within the device without sending a transaction to the Host.

3.2.4.2.16 CleanEvictNoData

This is a request for the device to update the Host that a clean line is dropped in the device. The sole purpose of this request is to update any snoop filters in the Host and no data is exchanged.

CleanEvictNoData is only expected for host-attached memory address ranges. For device-attached memory range, the equivalent operation can be completed internally within the device without sending a transaction to the Host.

3.2.4.2.17 WOWrInv

This is a weakly ordered write invalidate line request of 0-63 bytes for write combining type stores. Any combination of byte enables may be set.

Typically, WOWrInv receives a FastGO-WritePull followed by an ExtCmp. Upon receiving the FastGO-WritePull the device sends the data to the Host. For host-attached memory, the Host sends the ExtCmp once the write is complete in memory.

FastGO does not provide “Global Observation”.

In error conditions, a GO-Err-WritePull is received. The device sends the data as normal, and the Host drops it. The device is responsible for handling the error as appropriate. An ExtCmp is still sent by the Host after the GO-Err in all cases.

3.2.4.2.18 WOWrInvF

Same as WOWrInv (rules and flows), except it is a write of 64 bytes.

3.2.4.2.19 WrInv

This is a write invalidate line request of 0-64 bytes. Typically, WrInv receives a WritePull followed by a GO. Upon getting the WritePull, the device sends the data to the Host. The Host sends GO once the write completes in memory (both, host-attached or device-attached).

In error conditions, a GO-Err is received. The device is responsible for handling the error as appropriate.

3.2.4.2.20 CacheFlushed

This is an indication sent by the device to inform the Host that its caches are flushed, and it no longer contains any cachelines in the Shared, Exclusive or Modified state (a device may exclude addresses that are part of its “Device-attached Memory” mapped as HDM-D/HDM-DB). The Host can use this information to clear its snoop filters, block snoops to the device, and return a GO. Once the device receives the GO, it is guaranteed to not receive any snoops from the Host until the device sends the next cacheable D2H Request.

When a CXL.cache device is flushing its cache, the device must wait for all responses for cacheable access before sending the CacheFlushed message. This is necessary because the Host must observe CacheFlushed only after all inflight messages that impact device coherence tracking in the Host are complete.

IMPLEMENTATION NOTE

Snoops may be pending to the device when the Host receives the CacheFlushed command and the Host may complete the CacheFlushed command (sending a GO) while those snoops are outstanding. From the device point of view, this can be observed as receiving snoops after the CacheFlushed message is complete. The device should allow for this behavior without creating long stall conditions on the snoops by waiting for snoop queues to drain before initiating any power state transition (e.g., L1 link state) that could stall snoops.

**Table 3-23. D2H Request (Targeting Non Device-attached Memory)
Supported H2D Responses**

D2H Request	H2D Response									
	WritePull	GO_WritePull	ExtCmp	GO_WritePull_Drop	Fast_GO_WritePull	GO_ERR_WritePull	GO-I	GO-S	GO-E	GO-M
RdCurr										
RdOwn						X	X		X	X
RdShared						X	X	X		
RdAny						X	X	X	X	X
RdOwnNoData						X			X	
ItoMWr	X				X					
WrCur	X				X					
CLFlush						X	X			
CleanEvict	X		X							
DirtyEvict	X				X					
CleanEvictNoData							X			
WOWrInv		X		X	X					
WOWrInvF		X		X	X					
WrInv	X					X	X			
CacheFlushed							X			

For requests that target device-attached memory mapped as HDM-D, if the region is in Device Bias, no transaction is expected on CXL.cache because the Device can internally complete those requests. If the region is in Host Bias, [Table 3-24](#) shows how the device should expect the response. For devices with BISnp channel support in which the memory is mapped as HDM-DB, the resolution of coherence happens separately on the CXL.mem protocol and the “Not Supported” cases in the table are never sent from a device to the device-attached memory address range. The only commands supported on CXL.cache to this address region when BISnp is enabled are ItoMWr, WrCur, and WrInv.

Table 3-24. D2H Request (Targeting Device-attached Memory) Supported Responses

D2H Request	Response on CXL.cache		Response on CXL.mem	
	Without BISnp (HDM-D)	With BISnp (HDM-DB)	Without BISnp (HDM-D)	With BISnp (HDM-DB)
RdCurr	GO-Err Bit set in H2D DH, Synthesized Data with all 1s (For Error Conditions)	Not Supported	MemRdFwd (For Success Conditions)	Not Supported
RdOwn	GO-Err on H2D Response, Synthesized Data with all 1s (For Error Conditions)	Not Supported	MemRdFwd (For Success Conditions)	Not Supported
RdShared	GO-Err on H2D Response, Synthesized Data with all 1s (For Error Conditions)	Not Supported	MemRdFwd (For Success Conditions)	Not Supported
RdAny	GO-Err on H2D Response, Synthesized Data with all 1s (For Error Conditions)	Not Supported	MemRdFwd (For Success Conditions)	Not Supported
RdOwnNoData	GO-Err on H2D Response (For Error Conditions)	Not Supported	MemRdFwd (For Success Conditions)	Not Supported
ItoMWr	Same as host-attached memory ¹		None	
WrCur	Same as host-attached memory ¹		None	
CLFlush	GO-Err on H2D Response (For Error Conditions)	Not Supported	MemRdFwd (For Success Conditions)	Not Supported
CleanEvict	Not Supported			
DirtyEvict	Not Supported			
CleanEvictNoData	Not Supported			
WOWrInv	GO_ERR_WritePull on H2D Response (For Error Conditions)	Not Supported	MemWrFwd (For Success Conditions)	Not Supported
WOWrInvF	GO_ERR_WritePull on H2D Response (For Error Conditions)	Not Supported	MemWrFwd (For Success Conditions)	Not Supported
WrInv	Same as host-attached memory ¹		None	
CacheFlushed	N/A ²		None	

- Flow for these commands follow the same flow as host memory regions and are not expected to check against CXL.mem coherence tracking (Bias Table or Snoop Filter) before issuing. The host will resolve coherence with the device using the CXL.mem protocol.
- There is no address in this command and the host must assume that this applies only to host memory regions (excluding device-attached memory).

CleanEvict, DirtyEvict, and CleanEvictNoData targeting device-attached memory should always be completed internally by the device, regardless of bias state. For D2H Requests that receive a response on CXL.mem, the CQID associated with the CXL.cache request is reflected in the Tag of the CXL.mem MemRdFwd or MemWrFwd command. For MemRdFwd, the caching state of the line is reflected in the MetaValue field as described in [Table 3-37](#).

3.2.4.3 Device to Host Response

Responses are directed at the Host entry indicated in the UQID field in the original H2D request message.

Table 3-25. D2H Response Encodings

Device CXL.cache Rsp	Opcode
RspIHitI	0 0100b
RspVHitV	0 0110b
RspIHitSE	0 0101b
RspSHitSE	0 0001b
RspSFwdM	0 0111b
RspIFwdM	0 1111b
RspVFwdV	1 0110b

3.2.4.3.1 RspIHitI

In general, this is the response that a device provides to a snoop when the line was not found in any caches. If the device returns RspIHitI for a snoop, the Host can assume the line has been cleared from that device.

3.2.4.3.2 RspVHitV

In general, this is the response that a device provides to a snoop when the line was hit in the cache and no state change occurred. If the device returns an RspVHitV for a snoop, the Host can assume a copy of the line is present in one or more places in that device.

3.2.4.3.3 RspIHitSE

In general, this is the response that a device provides to a snoop when the line was hit in a clean state in at least one cache and is now invalid. If the device returns an RspIHitSE for a snoop, the Host can assume the line has been cleared from that device.

3.2.4.3.4 RspSHitSE

In general, this is the response that a device provides to a snoop when the line was hit in a clean state in at least one cache and is now downgraded to shared state. If the device returns an RspSHitSE for a snoop, the Host should assume the line is still in the device.

3.2.4.3.5 RspSFwdM

This response indicates to the Host that the line being snooped is now in S state in the device, after having hit the line in Modified state. The device may choose to downgrade the line to Invalid. This response also indicates to the Host snoop tracking logic that 64 bytes of data is transferred on the D2H CXL.cache Data Channel to the Host data buffer indicated in the original snoop's destination (UQID).

3.2.4.3.6 RspIFwdM

This response indicates to the Host that the line being snooped is now in I state in the device, after having hit the line in Modified state. The Host may now assume the device contains no more cached copies of this line. This response also indicates to the Host

snoop tracking logic that 64 bytes of data will be transferred on the D2H CXL.cache Data Channel to the Host data buffer indicated in the original snoop's destination (UQID).

3.2.4.3.7 RspVFwdV

This response indicates that the device with E or M state (but not S state) is returning the current data to the Host and leaving the state unchanged. The Host must only forward the data to the requester because there is no state information.

3.2.4.4 Host to Device Requests

Snoops from the Host need not gain any credits besides local H2D request credits. The device will always send a Snoop Response message on the D2H CXL.cache Response channel. If the response is of the Rsp*Fwd* format, then the device must respond with 64 bytes of data via the D2H Data channel, directed at the UQID from the original snoop request message. If the response is not Rsp*Fwd*, the Host can consider the request complete upon receiving the snoop response message. The device can stop tracking the snoop once the response has been sent for non-data forwarding cases, or after both the last chunk of data has been sent and the response has been sent.

[Figure 3-17](#) shows the elements required to complete a CXL.cache snoop. Note that the response message can be received by the Host in any relative order with respect to the data messages. The byte enable field is always all 1s for Snoop data transfers.

Figure 3-17. CXL.cache Snoop Behavior

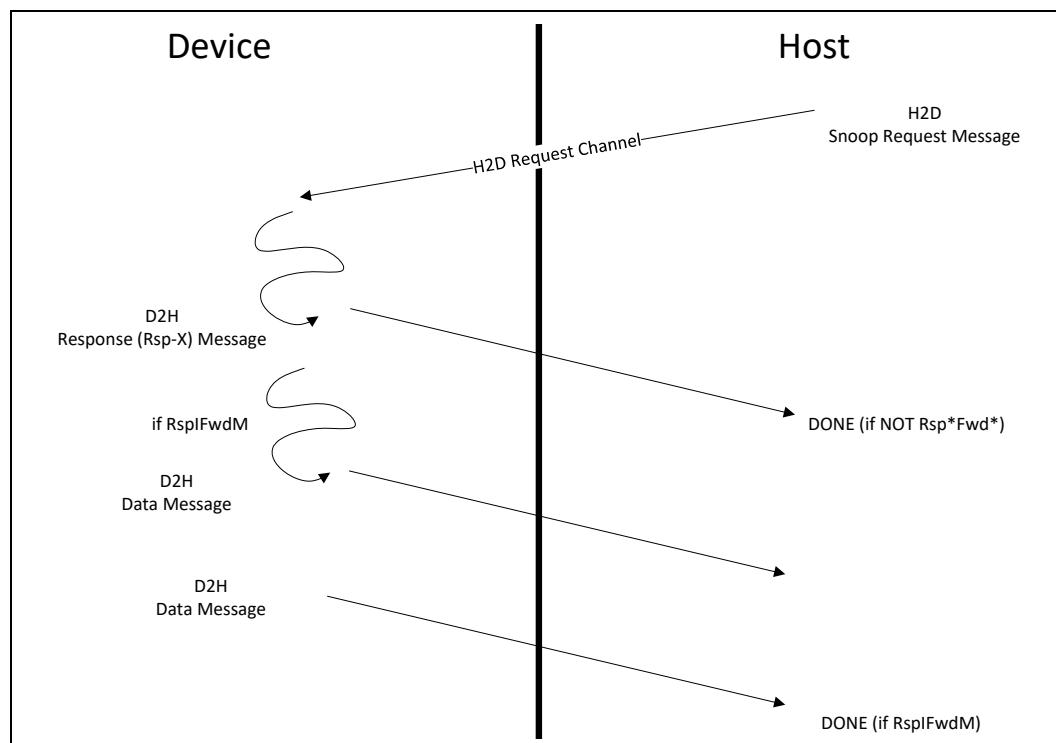


Table 3-26. CXL.cache – Mapping of H2D Requests to D2H Responses

	Opcode	RspIHitI	RspVhitV	RspSHitSE	RspIHitSE	RspSFwdM	RspIFwdM	RspVFwdV
SnpData	001b	X		X		X	X	
SnpInv	010b	X			X		X	
SnpCur	011b	X	X	X		X	X	X

3.2.4.4.1**SnpData**

These are snoop requests from the Host for lines that are intended to be cached in either Shared or Exclusive state at the requester (the Exclusive state can be cached at the requester only if all devices respond with RspI). This type of snoop is typically triggered by data read requests. A device that receives this snoop must either invalidate or downgrade all cachelines to Shared state. If the device holds dirty data it must return it to the Host.

3.2.4.4.2**SnpInv**

These are snoop requests from the Host for lines that are intended to be granted ownership and Exclusive state at the requester. This type of snoop is typically triggered by write requests. A device that receives this snoop must invalidate all cachelines. If the device holds dirty data it must return it to the Host.

3.2.4.4.3**SnpCur**

This snoop gets the current version of the line, but doesn't require change of any cache state in the hierarchy. It is only sent on behalf of the RdCurr request. If the device holds data in Modified state it must return it to the Host. The cache state can remain unchanged in both the device and Host, and the Host should not update its caches. To allow for varied cache implementations, devices are allowed to change cache state as captured in Table 3-26, but it is recommended to not change cache state.

3.2.4.5**Host to Device Response****Table 3-27.****H2D Response Opcode Encodings**

H2D Response Class	Encoding	RspData
WritePull	0001b	UQID
GO	0100b	MESI ¹
GO_WritePull	0101b	UQID
ExtCmp	0110b	Don't Care
GO_WritePull_Drop	1000b	UQID
Reserved	1100b	Don't Care
Fast_GO_WritePull	1101b	UQID
GO_ERR_WritePull	1111b	UQID

1. 4-bit MESI encoding is in LSB and the upper bits are Reserved.

3.2.4.5.1 WritePull

This response instructs the device to send the write data to the Host, but not to change the state of the line. This is used for WrInv where the data is needed before the GO-I can be sent. This is because GO-I is the notification that the write was completed.

3.2.4.5.2 GO

The Global Observation (GO) message conveys that read requests are coherent and that write requests are coherent and consistent. It is an indication that the transaction has been observed by the system device and the MESI state that is encoded in the RspType field indicates into which state the data associated with the transaction should be placed for the requester's caches. Details in [Table 3-20](#).

If the Host returns Modified state to the device, then the device is responsible for the dirty data and cannot drop the line without writing it back to the Host.

If the Host returns Invalid or Error state to the device, then the device must use the data at most once and not cache the data. Error responses to reads and cacheable write requests (for example, RdOwn or ItoMWr) will always be the result of an abort condition, so modified data can be safely dropped in the device.

3.2.4.5.3 GO_WritePull

This is a combined GO + WritePull message. No cache state is transferred to the device. The GO+WritePull message is used for write types that do not require a later message to know whether write data is visible.

3.2.4.5.4 ExtCmp

This response indicates that the data that was previously locally ordered (FastGO) has been observed throughout the system. Most importantly, accesses to memory will return the most up-to-date data.

3.2.4.5.5 GO_WritePull_Drop

This message has the same semantics as GO_WritePull, except that the device should not send data to the Host. This response can be sent in place of GO_WritePull when the Host determines that the data is not required. This response will never be sent for partial writes because the byte enables will always need to be transferred.

3.2.4.5.6 Fast_GO_WritePull

Similar to GO_WritePull, but only indicates that the request is locally observed. There will be a later ExtCmp message when the transaction is fully observable in memory. Devices that do not implement the Fast_Go feature may ignore the GO message and wait for the ExtCmp. Data must always be sent for the WritePull. No cache state is transferred to the device.

Locally Observed, in this context, is a host-specific coherence domain that may be a subset of the global coherence domain. An example is a Last Level Cache that the requesting device shares with other CXL.cache devices that are connected below a host-bridge. In that example, local observation is only within the Last Level Cache and not between other Last Level Caches.

3.2.4.5.7 GO_ERR_WritePull

Similar to GO_WritePull, but indicates that there was an error with the transaction that should be handled correctly in the device. Data must be sent to the Host for the WritePull, and the Host will drop the data. No cache state is transferred to the device (assumed Error). An ExtCmp is still sent if it is expected by the originating request.

3.2.5 Cacheability Details and Request Restrictions

These details and restrictions apply to all devices.

3.2.5.1 GO-M Responses

GO-M responses from the host indicate that the device is being granted the sole copy of modified data. The device must cache this data and write it back when it is done.

3.2.5.2 Device/Host Snoop-GO-Data Assumptions

When the host returns a GO response to a device, the expectation is that a snoop arriving to the same address of the request receiving the GO would see the results of that GO. For example, if the host sends GO-E for an RdOwn request followed by a snoop to the same address immediately afterwards, then one would expect the device to transition the line to M state and reply with an RspIFwdM response back to the Host. To implement this principle, the CXL.cache link layer ensures that the device will receive the two messages in separate slots to make the order completely unambiguous.

When the host is sending a snoop to the device, the requirement is that no GO response will be sent to any requests with that address in the device until after the Host has received a response for the snoop and all implicit writeback (IWB) data (dirty data forwarded in response to a snoop) has been received.

When the host returns data to the device for a read type request, and GO for that request has not yet been sent to the device, the host may not send a snoop to that address until after the GO message has been sent. Because the new cache state is encoded in the response message for reads, sending a snoop to an address without having received GO, but after having received data, is ambiguous to the device as to what the snoop response should be in that situation.

Fundamentally, the GO that is associated with a read request also applies to the data returned with that request. Sending data for a read request implies that data is valid, meaning the device can consume it even if the GO has not yet arrived. The GO will arrive later and inform the device what state to cache the line in (if at all) and whether the data was the result of an error condition (e.g., hitting an address region that the device was not allowed to access).

3.2.5.3 Device/Host Snoop/WritePull Assumptions

The device requires that the host cannot have both a WritePull and H2D Snoop active on CXL.cache to a given 64-byte address. The host may not launch a snoop to a 64-byte address until all WritePull data from that address has been received by the host. Conversely, the host may not launch a WritePull for a write until the host has received the snoop response (including data in case of Rsp*Fwd*) for any snoops to the pending writes address. Any violation of these requirements will mean that the Bogus field on the D2H Data channel will be unreliable.

3.2.5.4

Snoop Responses and Data Transfer on CXL.cache Evicts

To snoop cache evictions (for example, DirtyEvict) and maintain an orderly transfer of snoop ownership from the device to the host, cache evictions on CXL.cache must adhere to the following protocol.

If a device Evict transaction has been issued on the CXL.cache D2H request channel, but has not yet processed its WritePull from the host, and a snoop hits the writeback, the device must track this snoop hit if cache state is changed, which excludes the case when SnpCur results in a RspVFwdV response. When the device begins to process the WritePull, if snoop hit is tracked the device must set the Bogus field in all the D2H data messages sent to the host. The intent is to communicate to the host that the request data was already sent as IWB data, so the data from the Evict is potentially stale.

3.2.5.5

Multiple Snoops to the Same Address

The host is only allowed to have one snoop pending at a time per cacheline address per device. The host must wait until it has received both the snoop response and all IWB data (if any) before sending the next snoop to that address.

3.2.5.6

Multiple Reads to the Same Cacheline

Multiple read requests (cacheable or uncacheable) to the same cacheline 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 that use HDM-D memory, in addition to RdCurr and/or CLFlush, multiple RdOwnNoData (bias flip requests) are 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 an ambiguous state between the device and the host.

3.2.5.7

Multiple Evicts to the Same Cacheline

Multiple Evicts to the same cacheline are not allowed. All Evict messages from the device provide a guarantee to the host that the evicted cacheline will no longer be present in the device's caches.

Thus, it is a coherence violation to send another Evict for the same cacheline without an intervening cacheable Read/Read0 request to that address.

3.2.5.8

Multiple Write Requests to the Same Cacheline

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

3.2.5.9

Multiple Read and Write Requests to the Same Cacheline

Multiple RdCur/CLFlush/WrInv/WOWrInv/ItoMWr/WrCur may be issued in parallel from devices to the same cacheline address. Other reads need to issue one at a time (also known as "serialize"). To serialize, the read must not be issued until all other

outstanding accesses to same cacheline address have received GO. Additionally, after the serializing read is issued, no other accesses to the same cacheline address may be issued until it has received GO.

3.2.5.10 Normal Global Observation (GO)

Normal Global Observation (GO) responses are sent only after the host has guaranteed that request will have next ownership of the requested cacheline. GO messages for requests carry the cacheline state permitted through the MESI state or indicate that the data should only be used once and whether an error occurred.

3.2.5.11 Relaxed Global Observation (FastGO)

FastGO is only allowed for requests that do not require strict ordering. The Host may return the FastGO once the request is guaranteed next ownership of the requested cacheline within an implementation dependent sub-domain (e.g., CPU socket), but not necessarily within the system. Requests that receive a FastGO response and require completion messages are usually of the write combining memory type and the ordering requirement is that there will be a final completion (ExtCmp) message indicating that the request is at the stage where it is fully observed throughout the system. To make use of FastGO, devices have specific knowledge of the FastGO boundary of the CXL hierarchy and know the consumer of the data is within that hierarchy; otherwise, they must wait for the ExtCmp to know the data will be visible.

3.2.5.12 Evict to Device-attached Memory

Device Evicts to device-attached memory are not allowed on CXL.cache. Evictions are expected to go directly to the device's own memory; however, a device may use non-Evict writes (e.g., ItoMWr, WrCur) to write data to the host to device-attached memory.

3.2.5.13 Memory Type on CXL.cache

To source requests on CXL.cache, devices need to get the Host Physical Address (HPA) from the Host by means of an ATS request on CXL.io. Due to memory type restrictions, on the ATS completion, the Host indicates to the device if an HPA can only be issued on CXL.io as described in [Section 3.1.6](#). The device is not allowed to issue requests to such HPAs on CXL.cache. For requests that target ranges within the Device's local HDM range, the HPA is permitted to be obtained by means of an ATS request on CXL.io, or by using device-specific means.

3.2.5.14 General Assumptions

1. The Host will NOT preserve ordering of the CXL.cache requests as delivered by the device. The device must maintain the ordering of requests for the case(s) where ordering matters. For example, if D2H memory writes need to be ordered with respect to an MSI (on CXL.io), it is up to the device to implement the ordering. This is made possible by the non-posted nature of all requests on CXL.cache.
2. The order chosen by the Host will be conveyed differently for reads and writes. For reads, a Global Observation (GO) message conveys next ownership of the addressed cacheline; the data message conveys ordering with respect to other transactions. For writes, the GO message conveys both next ownership of the line and ordering with respect to other transactions.
3. The device may cache ownership and internally order writes to an address if a prior read to that address received either GO-E or GO-M.
4. For reads from the device, the Host transfers ownership of the cacheline with the GO message, even if the data response has not yet been received by the device. The device must respond to a snoop to a cacheline which has received GO, but if

data from the current transaction is required (e.g., a RdOwn to write the line) the data portion of the snoop is delayed until the data response is received.

5. The Host must not send a snoop for an address where it has sent a data response for a previous read transaction but has not yet sent the GO. Ordering will ensure that the device observes the GO in this case before any later snoop. Refer to [Section 3.2.5.2](#) for additional details.
6. Write requests (other than Evicts) such as WrInv, WOWrInv*, ItoMWr, and WrCur will never respond to WritePulls with data marked as Bogus.
7. The Host must not send two cacheline data responses to the same device request. The device may assume one-time use ownership (based on the request) and begin processing for any part of a cacheline received by the device before the GO message. Final state information will arrive with the GO message, at which time the device can either cache the line or drop it depending on the response.
8. For a given transaction, H2D Data transfers must come in consecutive packets in natural order with no interleaved transfers from other lines.
9. D2H Data transfer of a cacheline must come in consecutive packets with no interleaved transfers from other lines. The data must come in natural chunk order, that is, 64B transfers must complete the lower 32B half first because snoops are always cacheline aligned.
10. Device snoop responses in D2H Response must not be dependent on any other channel or on any other requests in the device besides the availability of credits in the D2H Response channel. The Host must guarantee that the responses will eventually be serviced and return credits to the device.
11. The Host must not send a second snoop request to an address until all responses, plus data if required, for the prior snoop are collected.
12. H2D Response and H2D Data messages to the device must drain without the need for any other transaction to make progress.
13. The Host must not return GO-M for data that is not actually modified with respect to memory.
14. The Host must not write unmodified data back to memory.
15. Except for WOWrInv and WOWrInF, all other writes are strongly ordered

3.2.5.15 Buried Cache State Rules

Buried Cache state refers to the state of the cacheline registered in the Device's Coherency engine (DCOH) when a CXL.cache request is being sent for that cacheline from the device.

The Buried Cache state rules for a device when issuing CXL.cache requests are as follows:

- Must not issue a Read if the cacheline is buried in Modified, Exclusive, or Shared state.
- Must not issue RdOwnNoData if the cacheline is buried in Modified or Exclusive state. The Device may request for ownership in Exclusive state as an upgrade request from Shared state.
- Must not issue a Read0-Write if the cacheline is buried in Modified, Exclusive, or Shared state.
- All *Evict opcodes must adhere to apropos use case. For example, the Device is allowed to issue DirtyEvict for a cacheline only when it is buried in Modified state. For performance benefits, it is recommended that the Device should not silently drop a cacheline in Exclusive or Shared state and instead use CleanEvict* opcodes toward the Host.

- The CacheFlushed Opcode is not specific to a cacheline, it is an indication to the Host that all the Device's caches are flushed. Thus, the Device must not issue CacheFlushed if there is any cacheline buried in Modified, Exclusive, or Shared state.

[Table 3-28](#) describes which Opcodes in D2H requests are allowed for a given Buried Cache State.

IMPLEMENTATION NOTE

Buried state rules are requirements at the requester's Transaction Layer. It is possible snoops in flight to change the state observed at the host before the host processes the request. An example case is SnpInv sent from the host at the same time as the device issues a CleanEvictNoData from E-state, the snoop will cause the cache state in the device to change to I-state before the CleanEvictNoData is processed in the host, so the host must allow for this degraded cache state in its coherence tracking.

Table 3-28. Allowed Opcodes for D2H Requests per Buried Cache State

D2H Requests		Buried Cache State			
Opcodes	Semantic	Modified	Exclusive	Shared	Invalid
RdCurr	Read				X
RdOwn	Read				X
RdShared	Read				X
RdAny	Read				X
RdOwnNoData	Read0			X	X
ItoMWr	Read0-Write				X
WrCur	Read0-Write				X
CLFlush	Read0				X
CleanEvict	Write		X		
DirtyEvict	Write	X			
CleanEvictNoData	Write		X	X	
WOWrInv	Write				X
WOWrInvF	Write				X
WrInv	Write				X
CacheFlushed	Read0				X

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 received from 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 by using the M2S Req channel. These statements combined could lead a Type 2 Device to assume that H2D Req messages can never target addresses that belong 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 occur for multiple reasons, including coarse tracking and proprietary RAS features. In that context, a host may generate an

H2D Req to a Type 2 device on addresses that belong to the Device-attached Memory. An H2D Req from the host that targets 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 Type 2 devices and HDM-DB Type 2 devices are required to:

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

3.3

CXL.mem

3.3.1

Introduction

The CXL Memory Protocol is called CXL.mem, and it is a transactional interface between the CPU and Memory. It uses the phy and link layer of CXL when communicating across dies. The protocol can be used for multiple different Memory attach options including when the Memory Controller is located in the Host CPU, when the Memory Controller is within an Accelerator device, or when the Memory Controller is moved to a memory buffer chip. It applies to different Memory types (e.g., volatile, persistent, etc.) and configurations (e.g., flat, hierarchical, etc.) as well.

The CXL.mem provides 3 basic coherence models for CXL.mem Host-managed Device Memory (HDM) address regions exposed by the CXL.mem protocol:

- HDM-H (Host-only Coherent): Used only for Type 3 Devices
- HDM-D (Device Coherent): Used only for legacy Type 2 Devices that rely on CXL.cache to manage coherence with the Host
- HDM-DB (Device Coherent using Back-Invalidate): Can be used by Type 2 Devices or Type 3 Devices

Note:

The view of the address region must be consistent on the CXL.mem path between the Host and the Device.

The coherency engine in the CPU interfaces with the Memory (Mem) using CXL.mem requests and responses. In this configuration, the CPU coherency engine is regarded as the CXL.mem Master and the Mem device is regarded as the CXL.mem Subordinate. The CXL.mem Master is the agent which is responsible for sourcing CXL.mem requests (e.g., reads, writes, etc.) and a CXL.mem Subordinate is the agent which is responsible for responding to CXL.mem requests (e.g., data, completions, etc.).

When the Subordinate maps HDM-D/HDM-DB, CXL.mem protocol assumes the presence of a device coherency engine (DCOH). This agent is assumed to be responsible for implementing coherency related functions such as snooping of device caches based on CXL.mem commands and update of Metadata fields.

Support for memory with Metadata is optional but this needs to be negotiated with the Host in advance. The negotiation mechanisms are beyond the scope of this specification. If Metadata is not supported by device-attached memory, the DCOH will still need to use the Host supplied Metadata updates to interpret the commands. If Metadata is supported by device-attached memory, it can be used by Host to implement a coarse snoop filter for CPU sockets. In the HDM-H address region, the usage is defined by the Host. The protocol allows for 2 bits of Metadata to be stored and returned.

CXL.mem transactions from Master to Subordinate are called "M2S" and transactions from Subordinate to Master are called "S2M".

Within M2S transactions, there are three message classes:

- Request without data - generically called Requests (Req)
- Request with Data - (RwD)
- Back-Invalidate Response - (BIRsp)

Similarly, within S2M transactions, there are three message classes:

- Response without data - generically called No Data Response (NDR)
- Response with data - generically called Data Response (DRS)
- Back-Invalidate Snoop - (BISnp)

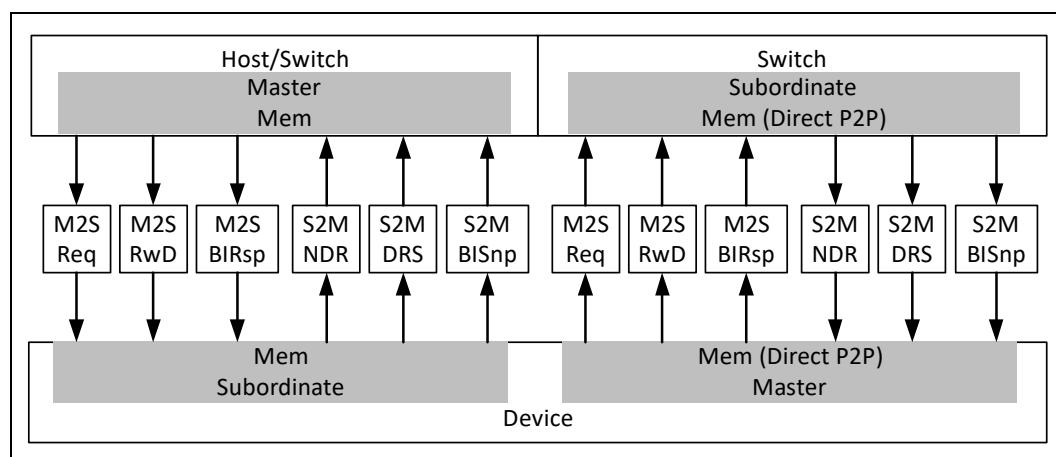
The next sections describe the above message classes and opcodes in detail. Each message will support 3 variants: 68B Flit, 256B Flit, and PBR Flit. The use of each of these will be negotiated in the physical layer for each link as defined in [Chapter 6.0](#).

3.3.2 CXL.mem Channel Description

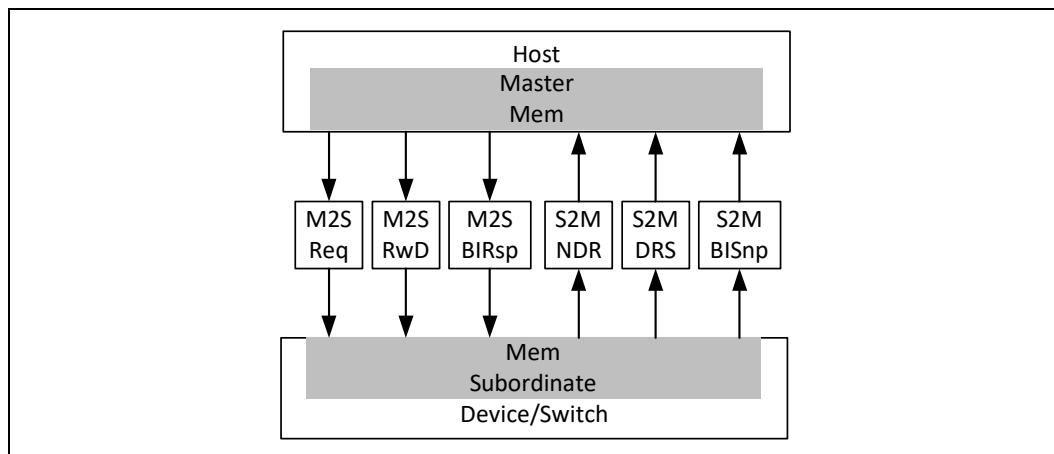
In general, the CXL.mem channels work independently of one another to ensure that forward progress is maintained. Details of the specific ordering allowances and requirements between channels are captured in [Section 3.4](#). Within a channel there are no ordering rules, but exceptions to this are described in [Section 3.3.11](#).

The device interface for CXL.mem defines 6 channels on primary memory protocol and an additional 6 to support direct P2P as shown in [Figure 3-18](#). Devices that support HDM-DB must support the BI* channels (S2M BISnp and M2S BIRsp). Type 2 devices that use the HDM-D memory region may not have the BI* channels. Type 3 devices (Memory Expansion) may support HDM-DB to support direct peer-to-peer on CXL.io. MLD and G-FAM devices may use HDM-DB to enable multi-host coherence and direct peer-to-peer on CXL.mem. The HDM-DB regions will be known by software and programmed as such in the decode registers and these regions will follow the protocol flows, using the BISnp channels as defined in [Appendix C, "Memory Protocol Tables."](#)

Figure 3-18. CXL.mem Channels for Devices



For Hosts, the number of channels are defined in [Figure 3-19](#). The channel definition is the same as for devices.

Figure 3-19. CXL.mem Channels for Hosts

3.3.2.1

Direct P2P CXL.mem for Accelerators

In certain topologies, an accelerator (Type 1, Type 2, or Type 3) device may optionally be enabled to communicate with peer Type 3 memories with CXL.mem protocol. Support for such communication is provided by an additional set of CXL.mem channels, with their directions reversed from conventional CXL.mem as shown in [Figure 3-18](#). These channels exist only on a link between the device and the switch downstream port to which the link is attached. Ordering requirements, message formats, and channel semantics are the same as for conventional CXL.mem. Topologies supporting Direct P2P.mem require an accelerator (requester device) and a target Type 3 peer memory device which are both directly connected to a PBR Edge DSP. PBR routing is required because not all CXL.mem messages contain sufficient information for an HBR switch to determine whether to route between a device and the host or a device and a peer device. Edge DSPs contain tables (FAST and LDST) which enable routing to the proper destination.

Details related to the device in-out dependence covering standard CXL.mem target and the source of Direct P2P CXL.mem and are covered in [Table 3-58](#).

3.3.2.2

Snoop Handling with Direct P2P CXL.mem

It is possible for a device that is using the Direct P2P CXL.mem interface to receive a snoop on H2D Req for an address that the device had previously requested over its P2P CXL.mem interface. This could occur, for example, if the host has snoop filtering disabled. Conversely, the device could receive an S2M BISnp from a peer for a line that it had acquired over CXL.cache through the host.

As a result, devices that use the Direct P2P CXL.mem interface are required to track which interface was used when a cacheline was requested and respond normally to snoops using this channel. If the device receives a snoop on a different interface, the device shall respond as though it does not have the address cached returning RspIHitI or BIRspI and shall not change the cacheline state.

IMPLEMENTATION NOTE

How the device tracks which interface was used to request each cacheline is implementation dependent. One method of tracking could be for the device to maintain a table of address ranges, programmed by software with an indication for each range whether the CXL.cache or Direct P2P CXL.mem interface should be used. This table could then be looked up when snoops are received. Other methods may also be used.

3.3.3

Back-Invalidate Snoop

To enable a device to implement an inclusive Snoop Filter for tracking host caching of device memory, a Back-Invalidate Snoop (BISnp) is initiated from the device to change the cache state of the host. The flows related to this channel are captured in [Section 3.5.1](#). The definition of “inclusive Snoop Filter” for the purpose of CXL is a device structure that tracks cacheline granular host caching and is a limited size that is a small subset of the total Host Physical Address space supported by the device.

In 68B flits, only the CXL.cache D2H Request flows can be used for device-attached memory to manage coherence with the host as shown in [Section 3.5.2.3](#). This flow is used for addresses with the HDM-D memory attribute. A major constraint with this flow is that the D2H Req channel can be blocked waiting on forward progress of the M2S Request channel which disallows an inclusive Snoop Filter architecture. For the HDM-DB memory region, the BISnp channel (instead of CXL.cache) is used to resolve coherence. CXL host implementations may have a mix of devices with HDM-DB and HDM-D below a Root Port.

The rules related to Back-Invalidate are spread around in different areas of the specification. The following list captures a summary and pointers to requirements:

- Ordering rules in [Section 3.4](#)
- Conflict detection flows and blocking in [Section 3.5.1](#)
- Protocol Tables in [Section C.1.2](#)
- BI-ID configuration in [Section 9.14](#)
- If an outstanding S2M BISnp is pending to an address the device must block M2S Req to the same address until the S2M BISnp is completed with the corresponding M2S BIRsp
- M2S Rwd channel must complete/drain without dependence on M2S Req or S2M BISnp

IMPLEMENTATION NOTE

Detailed performance implications of the implementation of an Inclusive Snoop Filter are beyond the scope of this specification, but high-level considerations are captured here:

- The number of cachelines that are tracked in an Inclusive Snoop Filter is determined based on host-processor caching of the address space. This is a function of the use model and the cache size in the host processor with upsizing of 4x or more. The 4x is based on an imprecise estimation of the unknowns in future host implementations and mismatch in Host cache ways/sectors as compared to Snoop-Filter ways/sectors.
- Device should have the capability to track BISnp messages triggered by Snoop Filter capacity evictions without immediately blocking requests on the M2S Req channel when the Inclusive Snoop Filter becomes full. In the case that the BISnp tracking structure becomes full the M2S Req channel will need to be blocked for functional correctness, but the design should size this BISnp tracker to ensure that blocking of the M2S Req channel is a rare event.
- The state per cacheline could be implemented as 2 states or 3 states. For 2 states, it would track the host in I vs. A, where A-state would represent "Any" possible MESI state in the host. For 3 states, it would add the precision of S-state tracking in which the Host may have at most a shared copy of the cacheline.

3.3.4

QoS Telemetry for Memory

QoS Telemetry for Memory is a mechanism for memory devices to indicate their current load level (DevLoad) in each response message for CXL.mem requests and each completion for (CXL.io) UIO requests. This enables the host or peer requester to meter the issue rate of CXL.mem requests and UIO requests to portions of devices, individual devices, or groups of devices as a function of their load level, optimizing the performance of those memory devices while limiting fabric congestion. This is especially important for CXL hierarchies containing multiple memory types (e.g., DRAM and persistent memory), Multi-Logical-Device (MLD) components, and/or G-FAM Devices (GFDs).

In addition to use cases with hosts that access memory devices, QoS Telemetry for memory supports the UIO Direct P2P to HDM (see [Section 7.7.9](#)) and Direct P2P CXL.mem for Accelerators (see [Section 7.7.10](#)) use cases. For these, the peer requester for each UIO or .mem request receives a DevLoad indication in each UIO completion or .mem response. For the UIO Direct P2P use case, the peer requester may be native PCIe or CXL. Within this section, "hosts/peers" is a shorthand for referring to host and/or peer requesters that access HDM devices.

Certain aspects of QoS Telemetry are mandatory for current CXL memory devices while other aspects are optional. CXL switches have no unique requirements for supporting QoS Telemetry. It is strongly recommended for Hosts to support QoS Telemetry as guided by the reference model contained in this section. For peer requesters, the importance of supporting QoS Telemetry depends on the device type, its capabilities, and its specific use case(s).

3.3.4.1

QoS Telemetry Overview

The overall goal of QoS Telemetry is for memory devices to provide immediate and ongoing DevLoad feedback to their associated hosts/peers, for use in dynamically adjusting their request-rate throttling. If a device or set of Devices become overloaded, the associated hosts/peers increase their amount of request rate throttling. If such

Devices become underutilized, the associated hosts/peers reduce their amount of request rate throttling. QoS Telemetry is architected to help hosts/peers avoid overcompensating and/or undercompensating.

Host/peer memory request rate throttling is optional and primarily implementation specific.

Table 3-29. Impact of DevLoad Indication on Host/Peer Request Rate Throttling

DevLoad Indication Returned in Responses	Host/Peer Request Rate Throttling
Light Load	Reduce throttling (if any) soon
Optimal Load	Make no change to throttling
Moderate Overload	Increase throttling immediately
Severe Overload	Invoke heavy throttling immediately

To accommodate memory devices supporting multiple types of memory more optimally, a device is permitted to implement multiple QoS Classes, which are identified sets of traffic, between which the device supports differentiated QoS and significant performance isolation. For example, a device supporting both DRAM and persistent memory might implement two QoS Classes, one for each type of supported memory. Providing significant performance isolation may require independent internal resources (e.g., individual request queues for each QoS Class).

This version of the specification does not provide architected controls for providing bandwidth management between device QoS Classes.

MLDs provide differentiated QoS on a per-LD basis. MLDs have architected controls specifying the allocated bandwidth fraction for each LD when the MLD becomes overloaded. When the MLD is not overloaded, LDs can use more than their allocated bandwidth fraction, up to specified fraction limits based on maximum sustained device bandwidth.

GFDs provide differentiated QoS on a per-host/peer basis. GFDs have architected controls that specify a QoS Limit Fraction value for each host/peer, based on maximum sustained device bandwidth.

HDM-DB devices send BISnp requests and receive BIRsp responses as a part of processing requests that they receive from host/peer requesters. BISnp and BIRsp messages shall not be tracked by QoS Telemetry mechanisms. If a BISnp triggers a host/peer requester writing back cached data, those transactions will be tracked by QoS Telemetry.

The DevLoad indication from CXL 1.1 memory devices will always indicate Light Load, allowing those devices to operate as best they can with hosts/peers that support QoS Telemetry, though they cannot have their memory request rate actively metered by the host/peer. Light Load is used instead of Optimal Load in case any CXL 1.1 devices share the same host/peer throttling range with current memory devices. If CXL 1.1 devices were to indicate Optimal Load, they would overshadow the DevLoad of any current devices indicating Light Load.

3.3.4.2 Reference Model for Host/Peer Support of QoS Telemetry

Host/peer support for QoS Telemetry is strongly recommended but not mandatory.

QoS Telemetry provides no architected controls for mechanisms in hosts/peers. However, if a host/peer implements independent throttling for multiple distinct sets of memory devices through a given port, the throttling must be based on HDM ranges, which are referred to as host/peer throttling ranges.

The reference model in this section covers recommended aspects for how a host/peer should support QoS Telemetry. Such aspects are not mandatory, but they should help maximize the effectiveness of QoS Telemetry in optimizing memory device performance while providing differentiated QoS and reducing CXL fabric congestion.

Each host/peer is assumed to support distinct throttling levels on a throttling-range basis, represented by Throttle[Range]. Throttle[Range] is periodically adjusted by conceptual parameters NormalDelta and SevereDelta. During each sampling period for a given Throttle[Range], the host/peer records the highest DevLoad indication reported for that throttling range, referred to as LoadMax.

Table 3-30. Recommended Host/Peer Adjustment to Request Rate Throttling

LoadMax Recorded by Host/Peer	Recommended Adjustment to Request Rate Throttling
Light Load	Throttle[Range] decremented by NormalDelta
Optimal Load	Throttle[Range] unchanged
Moderate Overload	Throttle[Range] incremented by NormalDelta
Severe Overload	Throttle[Range] incremented by SevereDelta

Any increments or decrements to Throttle[Range] should not overflow or underflow legal values, respectively.

Throttle[Range] is expected to be adjusted periodically, every tH nanoseconds unless a more immediate adjustment is warranted. The tH parameter should be configurable by platform-specific software, and ideally configurable on a per-throttling-range basis. When tH expires, the host/peer should update Throttle[Range] based on LoadMax, as shown in [Table 3-30](#), and then reset LoadMax to its minimal value.

Round-trip fabric time is the sum of the time for a request message to travel from host/peer to Device, plus the time for a response message to travel from Device to host/peer. The optimal value for tH is anticipated to be a bit larger than the average round-trip fabric time for the associated set of devices (e.g., a few hundred nanoseconds). To avoid overcompensation by the host/peer, time is needed for the received stream of DevLoad indications in responses to reflect the last Throttle[Range] adjustment before the host/peer makes a new adjustment.

If the host/peer receives a Moderate Overload or Severe Overload indication, it is strongly recommended for the host/peer to make an immediate adjustment in throttling, without waiting for the end of the current tH sampling period. Following that, the host/peer should reset LoadMax and then wait tH nanoseconds before making an additional throttling adjustment, to avoid overcompensating.

3.3.4.3 Memory Device Support for QoS Telemetry

3.3.4.3.1 QoS Telemetry Register Interfaces

An MLD must support a specified set of MLD commands from the MLD Component Command Set as documented in [Section 7.6.7.4](#). These MLD commands provide access to a variety of architected capability, control, and status registers for a Fabric Manager to use via the FM API.

A GFD must support a specified set of GFD commands from the GFD Component Management Command Set as documented in [Section 8.2.9.9.10](#). These GFD commands provide access to a variety of architected capability, control, and status registers for a Fabric Manager to use via the FM API.

If an SLD supports the Memory Device Command set, it must support a specified set of SLD QoS Telemetry commands. See [Section 8.2.9.9](#). These SLD commands provide access to a variety of architected capability, control, and status fields for management by system software via the CXL Device Register interface.

Each “architected QoS Telemetry” register is one that is accessible via the above mentioned MLD commands, GFD commands, and/or SLD commands.

3.3.4.3.2

Memory Device QoS Class Support

Each CXL memory device may support one or more QoS Classes. The anticipated typical number is one to four, but higher numbers are not precluded. If a device supports only one type of media, it may be common for it to support one QoS Class. If a device supports two types of media, it may be common for it to support two QoS Classes. A device supporting multiple QoS Classes is referred to as a multi-QoS device.

This version of the specification does not provide architected controls for providing bandwidth management between device QoS Classes. Still, it is strongly recommended that multi-QoS devices track and report DevLoad indications for different QoS Classes independently, and that implementations provide as much performance isolation between different QoS Classes as possible.

3.3.4.3.3

Memory Device Internal Loading (IntLoad)

A CXL memory device must continuously track its internal loading, referred to as IntLoad. A multi-QoS device should do so on a per-QoS-Class basis.

A device must determine IntLoad based at least on its internal request queuing. For example, a simple device may monitor the instantaneous request queue depth to determine which of the four IntLoad indications to report. It may also incorporate other internal resource utilizations, as summarized in [Table 3-31](#).

Table 3-31.

Factors for Determining IntLoad

IntLoad	Queuing Delay inside Device	Device Internal Resource Utilization
Light Load	Minimal	Readily handles more requests
Optimal Load	Modest to Moderate	Optimally utilized
Moderate Overload	Significant	Limiting throughput and/or degrading efficiency
Severe Overload	High	Heavily overloaded and/or degrading efficiency

The actual method of IntLoad determination is device-specific, but it is strongly recommended that multi-QoS devices implement separate request queues for each QoS Class. For complex devices, it is recommended for them to determine IntLoad based on internal resource utilization beyond just request queue depth monitoring.

Although the IntLoad described in this section is a primary factor in determining which DevLoad indication is returned in device responses, there are other factors that may need to be considered, depending upon the situation (see [Section 3.3.4.3.4](#) and [Section 3.3.4.3.5](#)).

3.3.4.3.4

Egress Port Backpressure

Even under a consistent Light Load, a memory device may experience flow control backpressure at its egress port. This is readily caused if an RP is oversubscribed by multiple memory devices below a switch. Prolonged egress port backpressure usually indicates that one or more upstream traffic queues between the device and the RP are full, and the delivery of responses from the device to the host/peer is significantly

delayed. This makes the QoS Telemetry feedback loop less responsive and the overall mechanism less effective. Egress Port Backpressure is an optional normative mechanism to help mitigate the negative effects of this condition.

IMPLEMENTATION NOTE

Egress Port Backpressure Leading to Larger Request Queue Swings

When the QoS Telemetry feedback loop is less responsive, the device's request queue depth is prone to larger swings than normal.

When the queue depth is increasing, the delay in the host/peer receiving Moderate Overload or Severe Overload indications results in the queue getting more full than normal, in extreme cases filling completely and forcing the ingress port to exert backpressure to incoming downstream traffic.

When the queue depth is decreasing, the delay in the host/peer receiving Light Load indications results in the queue getting more empty than normal, in extreme cases emptying completely, and causing device throughput to drop unnecessarily.

Use of the Egress Port Backpressure mechanism helps avoid upstream traffic queues between the device and its RP from filling for extended periods, reducing the delay of responses from the device to the host/peer. This makes the QoS Telemetry feedback loop more responsive, helping avoid excessive request queue swings.

IMPLEMENTATION NOTE

Minimizing Head-of-Line Blocking with Upstream Responses from MLDs/GFDs

When one or more upstream traffic queues become full between the MLD and one or more of its congested RPs, head-of-line (HOL) blocking associated with this congestion can delay or block traffic targeting other RPs that are not congested.

Egress port backpressure for extended periods usually indicates that the ingress port queue in the Downstream Switch Port above the device is often full. Responses in that queue targeting congested RPs can block responses targeting uncongested RPs, reducing overall device throughput unnecessarily.

Use of the Egress Port Backpressure mechanism helps reduce the average depth of queues carrying upstream traffic. This reduces the delay of traffic targeting uncongested RPs, increasing overall device throughput.

The Egress Port Congestion Supported capability bit and the Egress Port Congestion Enable control bit are architected QoS Telemetry bits, which indicate support for this optional mechanism plus a means to enable or disable it. The architected Backpressure Average Percentage status field returns a current snapshot of the measured egress port average congestion.

QoS Telemetry architects two thresholds for the percentage of time that the egress port experiences flow control backpressure. This condition is defined as the egress port having flits or messages waiting for transmission but is unable to transmit them due to a lack of suitable flow control credits. If the percentage of congested time is greater than or equal to Egress Moderate Percentage, the device may return a DevLoad indication of Moderate Overload. If the percentage of congested time is greater than or

equal to Egress Severe Percentage, the device may return a DevLoad indication of Severe Overload. The actual DevLoad indication returned for a given response may be the result of other factors as well.

A hardware mechanism for measuring Egress Port Congestion is described in [Section 3.3.4.3.9](#).

3.3.4.3.5

Temporary Throughput Reduction

There are certain conditions under which a device may temporarily reduce its throughput. Envisioned examples include a non-volatile memory (NVM) device undergoing media maintenance, a device cutting back its throughput for power/thermal reasons, and a DRAM device performing refresh. If a device is significantly reducing its throughput capacity for a temporary period, it may help mitigate this condition by indicating Moderate Overload or Severe Overload in its responses shortly before the condition occurs and only as long as really necessary. This is a device-specific optional mechanism.

The Temporary Throughput Reduction mechanism can give proactive advanced warning to associated hosts/peers, which can then increase their throttling in time to avoid the device's internal request queue(s) from filling up and potentially causing ingress port congestion. The optimum amount of time for providing advanced warning is highly device-specific, and a function of several factors, including the current request rate, the amount of device internal buffering, the level/duration of throughput reduction, and the fabric round-trip time.

A device should not use the mechanism unless conditions truly warrant its use. For example, if the device is currently under Light Load, it's probably not necessary or appropriate to indicate an Overload condition in preparation for a coming event. Similarly, a device that indicates an Overload condition should not continue to indicate the Overload condition past the point where it's needed.

The Temporary Throughput Reduction Supported capability bit and the Temporary Throughput Reduction Enable control bit are architected QoS Telemetry bits, which indicate support for this optional mechanism plus a means to enable or disable it.

IMPLEMENTATION NOTE

Avoid Unnecessary Use of Temporary Throughput Reduction

Ideally, a device should be designed to limit the severity and/or duration of its temporary throughput reduction events enough to where the use of this mechanism is not needed.

3.3.4.3.6

DevLoad Indication by Multi-QoS and Single-QoS SLDs

For SLDs, the DevLoad indication returned in each response is determined by the maximum of the device's IntLoad, Egress Port Congestion state, and Temporary Throughput Reduction state, as detailed in [Section 3.3.4.3.3](#), [Section 3.3.4.3.4](#), and [Section 3.3.4.3.5](#). For example, if IntLoad indicates Light Load, Egress Port Congestion indicates Moderate Overload, and Temporary Throughput Reduction does not indicate an overload, the resulting DevLoad indication for the response is Moderate Overload.

3.3.4.3.7

DevLoad Indication by Multi-QoS and Single-QoS MLDs

For MLDs, the DevLoad indication returned in each response is determined by the same factors as for SLDs, with additional factors used for providing differentiated QoS on a per-LD basis. Architected controls specify the allocated bandwidth for each LD as a fraction of total LD traffic when the MLD becomes overloaded. When the MLD is not

overloaded, LDs can use more than their allocated bandwidth fraction, up to specified fraction limits based on maximum sustained device bandwidth, independent of overall LD activity.

Bandwidth utilization for each LD is measured continuously based on current requests being serviced, plus the recent history of requests that have been completed.

Current requests being serviced are tracked by ReqCnt[LD] counters, with one counter per LD. The ReqCnt counter for an LD is incremented each time a request for that LD is received. The ReqCnt counter for an LD is decremented each time a response by that LD is transmitted. ReqCnt reflects instantaneous “committed” utilization, allowing the rapid reflection of incoming requests, especially when requests come in bursts.

The recent history of requests completed is tracked by CmpCnt[LD, Hist] registers, with one set of 16 Hist registers per LD. An architected configurable Completion Collection Interval control for the MLD determines the time interval over which transmitted responses are counted in the active (newest) Hist register/counter. At the end of each interval, the Hist register values for the LD are shifted from newer to older Hist registers, with the oldest value being discarded, and the active (newest) Hist register/counter being cleared. Further details on the hardware mechanism for CmpCnt[LD, Hist] are described in [Section 3.3.4.3.10](#).

Controls for LD bandwidth management consist of per-LD sets of registers called QoS Allocation Fraction[LD] and QoS Limit Fraction[LD]. For each LD, QoS Allocation Fraction specifies the fraction of current device utilization allocated for the LD across all its QoS classes. QoS Limit Fraction for each LD specifies the fraction of maximum sustained device utilization as a fixed limit for the LD across all its QoS classes, independent of overall MLD activity.

Bandwidth utilization for each LD is based on the sum of its associated ReqCnt and CmpCnt[Hist] counters/registers. CmpCnt[Hist] reflects recently completed requests, and Completion Collection Interval controls how long this period of history covers (i.e., how quickly completed requests are “forgotten”). CmpCnt reflects recent utilization to help avoid overcompensating for bursts of requests.

Together, ReqCnt and CmpCnt[Hist] provide a simple, fair, and tunable way to compute average utilization. A shorter response history emphasizes instantaneous committed utilization, improving responsiveness. A longer response history smooths the average utilization, reducing overcompensation.

ReqCmpBasis is an architected control register that provides the basis for limiting each LD’s utilization of the device, independent of overall MLD activity. Because ReqCmpBasis is compared against the sum of ReqCnt[] and CmpCnt[], its maximum value must be based on the maximum values of ReqCnt[] and CmpCnt[] summed across all configured LDs. The maximum value of Sum(ReqCnt[*]) is a function of the device’s internal queuing and how many requests it can concurrently service. The maximum value of Sum(CmpCnt[*,*]) is a function of the device’s maximum request service rate over the period of completion history recorded by CmpCnt[], which is directly influenced by the setting of Completion Collection Interval.

The FM programs ReqCmpBasis, the QoS Allocation Fraction array, and the QoS Limit Fraction array to control differentiated QoS between LDs. The FM is permitted to derate ReqCmpBasis below its maximum sustained estimate as a means of limiting power and heat dissipation.

To determine the DevLoad indication to return in each response, the device performs the following calculation:

```
Calculate TotalLoad = max(IntLoad[QoS], Egress Port Congestion state, Temporary Throughput Reduction state);
```

```
Calculate ReqCmpTotal and populate ReqCmpCnt[LD] array element
```

```
ReqCmpTotal = 0;  
  
For each LD  
  
    ReqCmpCnt[LD] = ReqCnt[LD] + Sum(CmpCnt[LD, *]);  
  
    ReqCmpTotal += ReqCmpCnt[LD];
```

IMPLEMENTATION NOTE

Avoiding Recalculation of ReqCmpTotal and ReqCmpCnt[] Array

ReqCmpCnt[] is an array that avoids having to recalculate its values later in the algorithm.

To avoid recalculating ReqCmpTotal and ReqCmpCnt[] array from scratch to determine the DevLoad indication to return in each response, it is strongly recommended that an implementation maintains these values on a running basis, only incrementally updating them as new requests arrive and responses are transmitted. The details are implementation specific.

IMPLEMENTATION NOTE

Calculating the Adjusted Allocation Bandwidth

When the MLD is overloaded, some LDs may be over their allocation while others are within their allocation.

- Those LDs under their allocation (especially inactive LDs) contribute to a “surplus” of bandwidth that can be distributed across active LDs that are above their allocation.
- Those LDs over their allocation claim “their fair share” of that surplus based on their allocation, and the load value for these LDs is based on an “adjusted allocated bandwidth” that includes a prorated share of the surplus.

This adjusted allocation bandwidth algorithm avoids anomalies that otherwise occur when some LDs are using well below their allocation, especially if they are idle.

In subsequent algorithms, certain registers have integer and fraction portions, optimized for implementing the algorithms in dedicated hardware. The integer portion is described as being 16 bits unsigned, although it is permitted to be smaller or larger as needed by the specific implementation. It must be sized such that it will never overflow during normal operation. The fractional portion must be 8 bits. These registers are indicated by their name being in italics.

IMPLEMENTATION NOTE

Registers with Integer and Fraction Portions

These registers can hold the product of a 16-bit unsigned integer and an 8-bit fraction, resulting in 24 bits with the radix point being between the upper 16 bits and the lower 8 bits. Rounding to an integer is readily accomplished by adding 0000.80h (0.5 decimal) and truncating the lower 8 bits.

If TotalLoad is Moderate Overload or Severe Overload, calculate the adjusted

```

allocated bandwidth:

    ClaimAllocTotal = 0;

    SurplusTotal = 0;

    For each LD

        AllocCnt = QoS Allocation Fraction[LD] * ReqCmpTotal;

        If this LD is the (single) LD associated with the response

            AllocCntSaved = AllocCnt;

        If ReqCmpCnt[LD] > AllocCnt then

            ClaimAllocTotal += AllocCnt;

        Else

            SurplusTotal += AllocCnt - ReqCmpCnt[LD];

    For the single LD associated with the response

        If ReqCmpCnt[LD] > (AllocCntSaved + AllocCntSaved * SurplusTotal / ClaimAllocTotal) then LD is over its adjusted allocated bandwidth; // Use this result in the subsequent table

```

IMPLEMENTATION NOTE

Determination of an LD Being Above its Adjusted Allocated Bandwidth

The preceding equation requires a division, which is relatively expensive to implement in hardware dedicated for this determination. To enable hardware making this determination more efficiently, the following derived equivalent equation is strongly recommended:

$$\begin{aligned}
 &\text{ReqCmpCnt[LD]} > (\text{AllocCntSaved} + \text{AllocCntSaved} * \text{SurplusTotal} / \text{ClaimAllocTotal}) \\
 &(\text{ReqCmpCnt[LD]} * \text{ClaimAllocTotal}) > (\text{AllocCntSaved} * \text{ClaimAllocTotal} + \text{AllocCntSaved} * \text{SurplusTotal}) \\
 &(\text{ReqCmpCnt[LD]} * \text{ClaimAllocTotal}) > (\text{AllocCntSaved} * (\text{ClaimAllocTotal} + \text{SurplusTotal}))
 \end{aligned}$$

```

// Perform the bandwidth limit calculation for this LD

If ReqCmpCnt[LD] > QoS Limit Fraction [LD] * ReqCmpBasis then LD is over its limit BW;

```

Table 3-32. Additional Factors for Determining DevLoad in MLDs (Sheet 1 of 2)

TotalLoad	LD over Limit BW?	LD over Adjusted Allocated BW?	Returned DevLoad Indication
Light Load or Optimal Load	No	-	TotalLoad
	Yes	-	Moderate Overload

Table 3-32. Additional Factors for Determining DevLoad in MLDs (Sheet 2 of 2)

TotalLoad	LD over Limit BW?	LD over Adjusted Allocated BW?	Returned DevLoad Indication
Moderate Overload	No	No	Optimal Load
	No	Yes	Moderate Overload
	Yes	-	Moderate Overload
Severe Overload	-	No	Moderate Overload
	-	Yes	Severe Overload

The preceding table is based on the following policies for LD bandwidth management:

- The LD is always subject to its QoS Limit Fraction
- For TotalLoad indications of Light Load or Optimal Load, the LD can exceed its QoS Allocation Fraction, up to its QoS Limit Fraction
- For TotalLoad indications of Moderate Overload or Severe Overload, LDs with loads up to QoS Allocation Fraction get throttled less than LDs with loads that exceed QoS Allocation Fraction

3.3.4.3.8 DevLoad Indication by Multi-QoS and Single-QoS GFDs

DevLoad indication for GFDs is similar to that for MLDs, with the exception that 12-bit GFD host/peer requester PIDs (RPIDs) scale much higher than the 4-bit LDs for MLDs, and the QoS Allocation Fraction mechanism (based on current device utilization) is not supported for GFDs due to architectural scaling challenges. However, the QoS Limit Fraction mechanism (based on a fixed maximum sustained device utilization) is supported for GFDs, and architected controls specify the fraction limits.

Bandwidth utilization for each RPID is measured continuously based on current requests being serviced, plus the recent history of requests that have been completed.

Current requests that are being serviced are tracked by ReqCnt[RPID] counters, with one counter per RPID. The ReqCnt counter for an RPID is incremented each time a request from that RPID is received. The ReqCnt counter for an RPID is decremented each time a response to that RPID is transmitted. ReqCnt reflects instantaneous “committed” utilization, allowing the rapid reflection of incoming requests, especially when requests come in bursts.

The recent history of requests completed is tracked by CmpCnt[RPID, Hist] registers, with one set of 16 Hist registers per RPID. An architected configurable Completion Collection Interval control for the GFD determines the time interval over which transmitted responses are counted in the active (newest) Hist register/counter. At the end of each interval, the Hist register values for the RPID are shifted from newer to older Hist registers, with the oldest value being discarded, and the active (newest) Hist register/counter being cleared. Further details on the hardware mechanism for CmpCnt[RPID, Hist] are described in [Section 3.3.4.3.10](#).

Controls for RPID bandwidth management consist of per-RPID sets of registers called QoS Limit Fraction[RPID]. QoS Limit Fraction for each RPID specifies the fraction of maximum sustained device utilization as a fixed limit for the RPID across all its QoS classes, independent of overall GFD activity.

Bandwidth utilization for each RPID is based on the sum of its associated ReqCnt and CmpCnt[Hist] counters/registers. CmpCnt[Hist] reflects recently completed requests, and Completion Collection Interval controls how long this period of history covers (i.e., how quickly completed requests are “forgotten”). CmpCnt reflects recent utilization to help avoid overcompensating for bursts of requests.

Together, ReqCnt and CmpCnt[Hist] provide a simple, fair, and tunable way to compute average utilization. A shorter response history emphasizes instantaneous committed utilization, thus improving responsiveness. A longer response history smooths the average utilization, thus reducing overcompensation.

ReqCmpBasis is an architected control register that provides the basis for limiting each RPID's utilization of the device, independent of overall GFD activity. Because ReqCmpBasis is compared against the sum of ReqCnt[] and CmpCnt[], its maximum value must be based on the maximum values of ReqCnt[] and CmpCnt[] summed across all configured RPIDs. The maximum value of Sum(ReqCnt[*]) is a function of the device's internal queuing and how many requests it can concurrently service. The maximum value of Sum(CmpCnt[*,*]) is a function of the device's maximum request service rate over the period of completion history recorded by CmpCnt[], which is directly influenced by the setting of Completion Collection Interval.

The FM programs ReqCmpBasis and the QoS Limit Fraction array to control differentiated QoS between RPIDs. The FM is permitted to derate ReqCmpBasis below its maximum sustained estimate as a means of limiting power and heat dissipation.

To determine the DevLoad indication to return in each response, the device performs the following calculation:

```
Calculate TotalLoad = max(IntLoad[QoS], Egress Port Congestion state, Temporary Throughput Reduction state);

// Perform the bandwidth limit calculation for this RPID

If ReqCmpCnt[RPID] > QoS Limit Fraction[RPID] * ReqCmpBasis then the RPID is over its limit BW;
```

Table 3-33. Additional Factors for Determining DevLoad in MLDs/GFDs

TotalLoad	RPID over Limit BW?	Returned DevLoad Indication
Light Load or Optimal Load	No	TotalLoad
	Yes	Moderate Overload
Moderate Overload	No	Moderate Overload
	Yes	Severe Overload
Severe Overload	-	Severe Overload

Table 3-33 is based on the following policies for RPID bandwidth management:

- The RPID is always subject to its QoS Limit Fraction
- For TotalLoad indications of Moderate Overload, RPIDs with loads up to QoS Limit Fraction get throttled less than RPIDs with loads that exceed QoS Limit Fraction

3.3.4.3.9 Egress Port Congestion Measurement Mechanism

This hardware mechanism measures the average egress port congestion on a rolling percentage basis.

FCBP (Flow Control Backpressured): this binary condition indicates the instantaneous state of the egress port. It is true if the port has messages or flits available to transmit but is unable to transmit any of them due to a lack of suitable flow control credits.

Backpressure Sample Interval register: this architected control register specifies the fixed interval in nanoseconds at which FCBP is sampled. It has a range of 0-31. One hundred samples are recorded, so a setting of 1 yields 100 ns of history. A setting of 31 yields 3.1 us of history. A setting of 0 disables the measurement mechanism, and it must indicate an average congestion percentage of 0.

BPhist[100] bit array: this stores the 100 most-recent FCBP samples. It is not accessible by software.

Backpressure Average Percentage: when this architected status register is read, it indicates the current number of Set bits in BPhist[100]. It ranges in value from 0 to 100.

The actual implementation of BPhist[100] and Backpressure Average Percentage is device specific. Here is a possible implementation approach:

- BPhist[100] is a shift register
- Backpressure Average Percentage is an up/down counter
- With each new FCBP sample:
 - If the new sample (not yet in BPhist) and the oldest sample in BPhist are both 0 or both 1, no change is made to Backpressure Average Percentage.
 - If the new sample is 1 and the oldest sample is 0, increment Backpressure Average Percentage.
 - If the new sample is 0 and the oldest sample is 1, decrement Backpressure Average Percentage.
- Shift BPhist[100], discarding the oldest sample and entering the new sample

3.3.4.3.10 Recent Transmitted Responses Measurement Mechanism

This hardware mechanism measures the number of recently transmitted responses on a per-host/peer basis in the most recent 16 intervals of a configured time period. Hosts are identified by a Requester ID (ReqID), which is the LD-ID for MLDs and the RPID for GFDs.

Completion Collection Interval register: this architected control register specifies the interval over which transmitted responses are counted in an active Hist register. It has a range is 0-127. A setting of 1 yields 16 nanoseconds of history. A setting of 127 yields about 2 us of history. A setting of 0 disables the measurement mechanism, and it must indicate a response count of 0.

CmpCnt[ReqID, 16] registers: these registers track the total of recent transmitted responses on a per-host/peer basis. CmpCnt[ReqID, 0] is a counter and is the newest value, while CmpCnt[ReqID, 1:15] are registers. These registers are not directly visible to software.

For each ReqID, at the end of each Completion Collection Interval:

- The 16 CmpCnt[ReqID, *] register values are shifted from newer to older
- The CmpCnt[ReqID, 15] Hist register value is discarded
- The CmpCnt[ReqID, 0] register is cleared and it is armed to count transmitted responses in the next interval

3.3.5 M2S Request (Req)

The Req message class generically contains reads, invalidates, and signals going from the Master to the Subordinate.

Table 3-34. M2S Request Fields (Sheet 1 of 2)

Field	Width (Bits)			Description
	68B Flit	256B Flit	PBR Flit	
Valid	1			The valid signal indicates that this is a valid request
MemOpcode	4			Memory Operation: This specifies which, if any, operation needs to be performed on the data and associated information. Details in Table 3-35 .
SnpType	3			Snoop Type: This specifies what snoop type, if any, needs to be issued by the DCOH and the minimum coherency state required by the Host. Details in Table 3-38 . This field is used to indicate the Length Index for the TEUpdate opcode.
MetaField	2			Metadata Field: Up to 3 Metadata Fields can be addressed. This specifies which, if any, Metadata Field needs to be updated. Details of Metadata Field in Table 3-36 . If the Subordinate does not support memory with Metadata, this field will still be used by the DCOH for interpreting Host commands as described in Table 3-37 .
MetaValue	2			Metadata Value: When MetaField is not No-Op, this specifies the value to which the field needs to be updated. Details in Table 3-37 . If the Subordinate does not support memory with Metadata, this field will still be used by the device coherence engine for interpreting Host commands as described in Table 3-37 . For the TEUpdate message, this field carries the TE state change value where 00b is TE cleared and 01b is TE set.
Tag	16			The Tag field is used to specify the source entry in the Master which is pre-allocated for the duration of the CXL.mem transaction. This value needs to be reflected with the response from the Subordinate so the response can be routed appropriately. The exceptions are the MemRdFwd and MemWrFwd opcodes as described in Table 3-35 . Note: The Tag field has no explicit requirement to be unique.
Address[5]	1	0		Address[5] is provisioned for future usages such as critical chunk first for 68B flit, but this is not included in a 256B flit.
Address[51:6]	46			This field specifies the Host Physical Address associated with the MemOpcode.
LD-ID[3:0]	4		0	Logical Device Identifier: This identifies a Logical Device within a Multiple-Logical Device. Not applicable in PBR mode where SPID infers this field.
SPID	0		12	Source PID
DPID	0		12	Destination PID
CKID	0	13		Context Key ID: Optional key ID that references preconfigured key material utilized for device-based data-at-rest encryption. If the device has been configured to utilize CKID-based device encryption and locked utilizing the CXL Trusted Execution Environment (TEE) Security Protocol (TSP), then this field shall be valid for Data Read access types (MemRd/MemRdTEE/MemRdData*/MemSpecRd*) and treated as reserved for other messages.

Table 3-34. M2S Request Fields (Sheet 2 of 2)

Field	Width (Bits)			Description
	68B Flit	256B Flit	PBR Flit	
RSVD	6	7		Reserved
TC	2			Traffic Class: This can be used by the Master to specify the Quality of Service associated with the request. This is reserved for future usage.
Total	87	100	120	

Table 3-35. M2S Req Memory Opcodes (Sheet 1 of 2)

Opcode	Description	Encoding
MemInv	Invalidation request from the Master. Primarily for Metadata updates. No data read or write required. If SnpType field contains valid commands, perform required snoops.	0000b
MemRd	Normal memory data read operation. If MetaField contains valid commands, perform Metadata updates. If SnpType field contains valid commands, perform required snoops.	0001b
MemRdData	Normal Memory data read operation. MetaField has no impact on the coherence state. MetaValue is to be ignored. Instead, update Meta0-State as follows: If initial Meta0-State value = 'I', update Meta0-State value to 'A' Else, no update required If SnpType field contains valid commands, perform required snoops. MetaField encoding of Extended Meta-State (EMS) follows the rules for it in Table 3-36 .	0010b
MemRdFwd	This is an indication from the Host that data can be directly forwarded from device-attached memory to the device without any completion to the Host. This is only sent as a result of a CXL.cache D2H read request to device-attached memory that is mapped as HDM-D. The Tag field contains the reflected CQID sent along with the D2H read request. The SnpType is always No-Op for this Opcode. The caching state of the line is reflected in the Meta0-State value. Note: This message is not sent to devices that have device-attached memory that is mapped only as HDM-H or HDM-DB.	0011b
MemWrFwd	This is an indication from the Host to the device that it owns the line and can update it without any completion to the Host. This is only sent as a result of a CXL.cache D2H write request to device-attached memory that is mapped as HDM-D. The Tag field contains the reflected CQID sent along with the D2H write request. The SnpType is always No-Op for this Opcode. The caching state of the line is reflected in the Meta0-State value. Note: This message is not sent to devices that have device-attached memory that is mapped only as HDM-H or HDM-DB.	0100b
MemRdTEE ¹	Same as MemRd but with the Trusted Execution Environment (TEE) attribute. See Section 11.5.4.5 for description of TEE attribute handling.	0101b
MemRdDataTEE ¹	Same as MemRdData but with the Trusted Execution Environment (TEE) attribute. See Section 11.5.4.5 for description of TEE attribute handling.	0110b
MemSpecRd	Memory Speculative Read is issued to start a memory access before the home agent has resolved coherence to reduce access latency. This command does not receive a completion message. The Tag, MetaField, MetaValue, and SnpType are reserved. See Section 3.5.3.1 for a description of the use case.	1000b
MemInvNT	This is similar to the MemInv command except that the NT is a hint that indicates the invalidation is non-temporal and the writeback is expected soon. However, this is a hint and not a guarantee.	1001b
MemClnEvct	Memory Clean Evict is a message that is similar to MemInv, but intent to indicate host going to I-state and does not require Meta0-State return. This message is supported only to the HDM-DB address region.	1010b

Table 3-35. M2S Req Memory Opcodes (Sheet 2 of 2)

Opcode	Description	Encoding
MemSpecRdTEE ¹	Same as MemSpecRd but with Trusted Execution Environment (TEE) attribute. See Section 11.5.4.5 for description of TEE attribute handling.	1100b
TEUpdate ¹	Update of the TE state for the memory region. The memory region update is defined by the length-index field (passed in SnpType bits). The lower address bits in the message may be set to allow routing of the message to reach the correct interleave set target; however, the lower bits are masked to the natural alignment of the length when updating TE state. The MetaValue field defines the new TE state that supports 00b to clear and 01b to set. See details of the use of this message in Section 11.5.4.5.3 .	1101b
Reserved	Reserved	<Others>

1. Supported only in 256B and PBR Flit messages and considered Reserved in 68B Flit messages.

Table 3-36. Metadata Field Definition

MetaField	Description	Encoding
Meta0-State	Update the Metadata bits with the value in the Metadata Value field. Details of MetaValue associated with Meta0-State in Table 3-37 .	00b
Extended Meta-State (EMS)	This encoding has different interpretation in different channels: <ul style="list-style-type: none"> M2S Req usage indicates that the request requires the Extended MetaValue to be returned from the device in the response unless an error condition occurs. M2S Rwd and S2M DRS use this to indicate that the Extended MetaValue is attached to the message as a Trailer. This size of the MetaValue is configurable up to 32 bits. Other channels do not use this encoding and it should be considered Reserved. For HDM-DB, the MetaValue is defined in Table 3-37 for coherence resolution, Reserved for HDM-H. This encoding is not used for HDM-D.	01b
Reserved	Reserved	10b
No-Op	No Metadata operation. The MetaValue field is Reserved. For NDR/DRS messages that would return Metadata, this encoding can be used in case of an error in Metadata storage (standard 2-bits or EMD) or if the device does not store Metadata.	11b

Table 3-37. Meta0-State Value Definition (HDM-D/HDM-DB Devices)¹

Description	Encoding
Invalid (I): Indicates the host does not have a cacheable copy of the line. The DCOH can use this information to grant exclusive ownership of the line to the device.	
Note: When paired with a MemOpcodes = MemInv and SnpType = SnpInv, this is used to communicate that the device should flush this line from its caches, if cached, to device-attached memory resulting in all caches ending in I.	00b
Explicit No-Op: Used only when MetaField is Extended Meta-State in HDM-DB requests to indicate that a coherence state update is not requested. For all other cases this is considered a Reserved.	01b
Any (A): Indicates the host may have a shared, exclusive, or modified copy of the line. The DCOH can use this information to interpret that the Host likely wants to update the line and the device should not be given a copy of the line without resolving coherence with the host using the flow appropriate for the memory type.	10b
Shared (S): Indicates the host may have at most a shared copy of the line. The DCOH can use this information to interpret that the Host does not have an exclusive or modified copy of the line. If the device wants a shared or current copy of the line, the DCOH can provide this without informing the Host. If the device wants an exclusive copy of the line, the DCOH must resolve coherence with the Host using the flow appropriate for the memory type.	11b

1. HDM-H use case in Type 3 devices have Meta0-State definition that is host specific, so the definition in this table does not apply for the HDM-H address region in devices.

Table 3-38. Snoop Type Definition

SnpType Description	Description	Encoding
No-Op	No snoop needs to be performed	000b
SnpData	Snoop may be required - the requester needs at least a Shared copy of the line. Device may choose to give an exclusive copy of the line as well.	001b
SnpCur	Snoop may be required - the requester needs the current value of the line. Requester guarantees the line will not be cached. Device need not change the state of the line in its caches, if present.	010b
SnpInv	Snoop may be required - the requester needs an exclusive copy of the line.	011b
Reserved	Reserved	1xxb

Valid uses of M2S request semantics are described in [Table 3-39](#) but are not the complete set of legal flows. For a complete set of legal combinations, see [Appendix C](#).

Table 3-39. M2S Req Usage

M2S Req	MetaField	Meta Value	SnpType	S2M NDR	S2M DRS	Description
MemRd	Meta0-State	A	SnpInv	Cmp-E	MemData	The Host wants an exclusive copy of the line
MemRd	Meta0-State	S	SnpData	Cmp-S or Cmp-E	MemData	The Host wants a shared copy of the line
MemRd	No-Op	N/A ¹	SnpCur	Cmp	MemData	The Host wants a non-cacheable but current value of the line
MemRd	No-Op	N/A ¹	SnpInv	Cmp	MemData	The Host wants a non-cacheable value of the line and the device should invalidate the line from its caches
MemInv	Meta0-State	A	SnpInv	Cmp-E	N/A	The Host wants ownership of the line without data
MemInvNT	Meta0-State	A	SnpInv	Cmp-E	N/A	The Host wants ownership of the line without data. However, the Host expects this to be non-temporal and may do a writeback soon.
MemInv	Meta0-State	I	SnpInv	Cmp	N/A	The Host wants the device to invalidate the line from its caches
MemRdData	No-Op	N/A ¹	SnpData	Cmp-S or Cmp-E	MemData	The Host wants a cacheable copy in either exclusive or shared state
MemClnEvct	Meta0-State	I	No-Op	Cmp	N/A	Host is dropping E or S state from its cache and leaving the line in I-state. This message allows the Device to clean the Snoop Filter (or BIAS table).

1. N/A in the MetaValue indicates that the entire field is considered Reserved (cleared to 0 by sender and ignored by receiver).

3.3.6 M2S Request with Data (RwD)

The Request with Data (RwD) message class generally contains writes from the Master to the Subordinate.

Table 3-40. M2S RwD Fields (Sheet 1 of 2)

Field	Width (Bits)			Description
	68B Flit	256B Flit	PBR Flit	
Valid	1		The valid signal indicates that this is a valid request.	Memory Operation: This specifies which, if any, operation needs to be performed on the data and associated information. Details in Table 3-41 .
MemOpcode	4			
SnpType	3		Snoop Type: This specifies what snoop type, if any, needs to be issued by the DCOH and the minimum coherency state required by the Host. Details in Table 3-38 .	Metadata Field: Up to 3 Metadata Fields can be addressed. This specifies which, if any, Metadata Field needs to be updated. Details of Metadata Field in Table 3-36 . If the Subordinate does not support memory with Metadata, this field will still be used by the DCOH for interpreting Host commands as described in Table 3-37 .
MetaField	2			
MetaValue	2		Metadata Value: When MetaField is not No-Op, this specifies the value the field needs to be updated to. Details in Table 3-37 . If the Subordinate does not support memory with Metadata, this field will still be used by the device coherence engine for interpreting Host commands as described in Table 3-37 .	The Tag field is used to specify the source entry in the Master which is pre-allocated for the duration of the CXL.mem transaction. This value needs to be reflected with the response from the Subordinate so the response can be routed appropriately. For BIConflict, the tag encoding must use the same value as the pending M2S Req message (if one exists) which the BISnp found to be in conflict. This requirement is necessary to use Tag for fabric ordering of S2M NDR (Cmp* and BIConflictAck ordering for same tag). Note: The Tag field has no explicit requirement to be unique.
Tag	16			
Address[51:6]	46		This field specifies the Host Physical Address associated with the MemOpcode.	The Poison bit indicates that the data contains an error. The handling of poisoned data is device specific. See Chapter 12.0 for more details.
Poison	1			
TRP (formerly BEP)	0	1	Trailer Present: Indicates that a trailer is included on the message. The trailer size for RwD is defined in Table 3-43 . The trailer is observed in the Link Layer as a G-Slot following a 64B data payload. The baseline requirement for this bit is to enable only Byte Enables for partial writes (MemWrPtl). This bit is also optionally extended for Extend-Metadata indication. Note: This bit was formerly referred to as Byte-Enables Present (BEP), but has been redefined as part of an optional extension to support message trailers.	Logical Device Identifier: This identifies a logical device within a multiple-logical device. Not applicable in PBR messages where SPID infers this field.
LD-ID[3:0]	4	0		
SPID	0	12	Source PID	Destination PID
DPID	0	12		

Table 3-40. M2S RWD Fields (Sheet 2 of 2)

Field	Width (Bits)			Description
	68B Flit	256B Flit	PBR Flit	
CKID	0	13		Context Key ID: Optional key ID that references preconfigured key material utilized for device-based data-at-rest encryption. If the device has been configured to utilize CKID-based device encryption and locked utilizing the CXL Trusted Execution Environment (TEE) Security Protocol (TSP), then this field shall be valid for accesses that carry a non-reserved payload or cause a memory read to occur (MemWr*, MemRdFill*) and reserved for other cases (BICConflict).
RSVD	6	9		Reserved
TC		2		Traffic Class: This can be used by the Master to specify the Quality of Service associated with the request. This is reserved for future usage.
Total	87	104	124	

Table 3-41. M2S RWD Memory Opcodes

Opcode	Description	Encoding
MemWr	Memory write command. Used for full cacheline writes. If MetaField contains valid commands, perform Metadata updates. If SnpType field contains valid commands, perform required snoops. If the snoop hits a Modified cacheline in the device, the DCOH will invalidate the cache and write the data from the Host to device-attached memory.	0001b
MemWrPtl	Memory Write Partial. Contains 64 byte enables, one for each byte of data. If MetaField contains valid commands, perform Metadata updates. If SnpType field contains valid commands, perform required snoops. If the snoop hits a Modified cacheline in the device, the DCOH will need to perform a merge, invalidate the cache, and write the contents back to device-attached memory. Note: This command cannot be used with host-side memory encryption unless byte-enable encodings are aligned with encryption boundaries (32B aligned is an example which may be allowed).	0010b
BICConflict	Part of conflict flow for BISnp indicating that the host observed a conflicting coherent request to the same cacheline address. See Section 3.5.1 for details. This message carries a 64B payload as required by the RWD channel, but the payload bytes are reserved (cleared to all 0s). This message is sent on the RWD channel because the dependence rules on this channel allow for a low-complexity flow from a deadlock-avoidance point of view.	0100b
MemRdFill ¹	This is a simple read command equivalent to MemRd but never changes coherence state (MetaField=No-Op, SnpType=No-Op). The use of this command is intended for partial write data that is merging in the host with host-side encryption. With host-side encryption, it is not possible to merge partial data in the device as an attribute of the way encryption works. This message carries a 64B payload as required by the RWD channel; however, the payload bytes are reserved (i.e., cleared to all 0s). This message is sent on the RWD channel because the dependence rules on this channel allow for a low-complexity flow from a deadlock-avoidance point of view.	0101b
MemWrTEE ¹	Same as MemWr but with the Trusted Execution Environment (TEE) attribute. See Section 11.5.4.5 for description of TEE attribute handling.	1001b
MemWrPtlTEE ¹	Same as MemWrPtl but with the Trusted Execution Environment (TEE) attribute. See Section 11.5.4.5 for description of TEE attribute handling.	1010b
MemRdFillTEE ¹	Same as MemRdFill but with the Trusted Execution Environment (TEE) attribute. See Section 11.5.4.5 for description of TEE attribute handling.	1101b
Reserved	Reserved	<Others>

1. Supported only in 256B and PBR Flit messages and considered reserved in 68B Flit messages.

The definition of other fields are consistent with M2S Req (see [Section 3.3.11](#)). Valid uses of M2S RWD semantics are described in [Table 3-42](#) but are not complete set of legal flows. For a complete set of legal combinations, see [Appendix C](#).

Table 3-42. M2S RWD Usage

M2S Req	MetaField	Meta Value	SnpType	S2M NDR	Description
MemWr	Meta0-State	I	No-Op	Cmp	The Host wants to write the cacheline back to memory and does not retain a cacheable copy.
MemWr	Meta0-State	A	No-Op	Cmp	The Host wants to write the cacheline back to memory and retains a cacheable copy in shared, exclusive or modified state.
MemWr	Meta0-State	I	SnpInv	Cmp	The Host wants to write the cacheline to memory and does not retain a cacheable copy. In addition, the Host did not get ownership of the cacheline before doing this write and needs the device to snoop-invalidate its caches before performing the writeback to memory.
MemWrPtl	Meta0-State	I	SnpInv	Cmp	Same as the above row except the data being written is partial and the device needs to merge the data if it finds a copy of the cacheline in its caches.

3.3.6.1 Trailer Present for RWD (256B Flit)

In 256B Flit mode, a Trailer Present bit (TRP; formerly BEP, Byte-Enables Present) bit is included with the message header that indicates whether a Trailer slot is included at the end of the message. The trailer can be up to 96 bits.

Byte Enables field is 64 bits wide and indicates which of the bytes are valid for the contained data.

The Extended Metadata (EMD) trailer can be up to 32 bits. [Section 8.2.4.31](#) describes the registers that aid in discovery of device's EMD capability and EMD related configuration of the device. The mechanism for discovering the host's EMD capabilities and EMD related configuration of the host is host-specific. The host and the device must be configured in a consistent manner.

Table 3-43. RWD Trailers

Opcode/Message	MetaField	TRP	Trailer Size Required	Description
MemWr/ MemWrTEE	EMS	1	32 bits	Trailer bits[31:0] defined as EMD.
	No-OP/MS0	0	No Trailer	
MemWrPtl/ MemWrPtlTEE	EMS	1	96 bits	Trailer bits[63:0] defined as Byte Enables. Trailer bits[95:64] defined as EMD.
	No-Op/MS0		64 bits	Trailer bits[63:0] defined as Byte Enables.
<Others>	N/A	0	No Trailer	Other combinations do not encode trailers.

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 3-44. 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 3-45 .
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.
SPID		0	12	Source PID
DPID		0	12	Destination PID
RSVD		9		
Total		40	52	

Table 3-45. M2S BIRsp Memory OpCodes

Opcode	Description	Encoding
BIRspI	Host completed the Back-Invalidate Snoop for one cacheline and the host cache state is I.	0000b
BIRspS	Host completed the Back-Invalidate Snoop for one cacheline and the host cache state is S.	0001b
BIRspE	Host completed the Back-Invalidate Snoop for one cacheline and the host cache state is E.	0010b
BIRspIBlk	Same as BIRspI except that the message applies to the entire block of cachelines. The size of the block is explicit in the BISnp*Blk message for which this is a response.	0100b
BIRspSBlk	Same as BIRspS except that the message applies to the entire block of cachelines. The size of the block is explicit in the BISnp*Blk message for which this is a response.	0101b
BIRspEBlk	Same as BIRspE except that the message applies to the entire block of cachelines. The size of the block is explicit in the BISnp*Blk message for which this is a response.	0110b
Reserved	Reserved	<Others>

3.3.8 S2M Back-Invalidate Snoop (BISnp)

The Back-Invalidate Snoop (BISnp) message class contains Snoop messages from the Subordinate to the Master. This message class is not supported in 68B Flit mode.

Table 3-46. S2M BISnp Fields

Field	Width (Bits)			Description
	68B Flit	256B Flit	PBR Flit	
Valid	N/A	1		The valid signal indicates that this is a valid request.
Opcode		4		Snoop type with encodings in Table 3-47 .
BI-ID		12	0	BI-ID of the device that issued the message. See Section 9.14 for details on how this field is assigned. Not applicable in PBR messages where SPID infers this field.
BITag		12		Tracking ID from the device.
Address[51:6]		46		Host Physical Address. For *Blk opcodes, the lower 2 bits (Address[7:6]) are encoded as defined in Table 3-48 . Used for all other opcodes that represent the standard definition of Host Physical Address.
SPID		0	12	Source PID
DPID		0	12	Destination PID
RSVD		9		
Total		84	96	

Table 3-47. S2M BISnp Opcodes

Opcode	Description	Encoding
BISnpCur	Device requesting Current copy of the line but not requiring caching state.	0000b
BISnpData	Device requesting Shared or Exclusive copy.	0001b
BISnpInv	Device requesting Exclusive Copy.	0010b
BISnpCurBlk	Same as BISnpCur except covering 2 or 4 cachelines that are naturally aligned and contiguous. The Block Enable encoding is in Address[7:6] and defined in Table 3-48 . The host may give per cacheline response or a single block response applying to all cachelines in the block. More details are in Section 3.3.8.1 .	0100b
BISnpDataBlk	Same as BISnpData except covering 2 or 4 cachelines that are naturally aligned and contiguous. The Block Enable encoding is in Address[7:6] and defined in Table 3-48 . The host may give per cacheline response or a single block response applying to all cachelines in the block. More details are in Section 3.3.8.1 .	0101b
BISnpInvBlk	Same as BISnpInv except covering 2 or 4 cachelines that are naturally aligned and contiguous. The Block Enable encoding is in Address[7:6] and defined in Table 3-48 . The host may give per cacheline response or a single block response applying to all cachelines in the block. More details are in Section 3.3.8.1 .	0110b
Reserved	Reserved	<Others>

3.3.8.1 Rules for Block Back-Invalidate Snoops

A Block Back-Invalidate Snoop applies to multiple naturally aligned contiguous cachelines (2 or 4 cachelines). The host must ensure that coherence is resolved for each line and may send combined or individual responses for each in arbitrary order. In the presence of address conflicts, it is necessary that the host resolve conflicts for each cacheline separately. This special address encoding applies only to BISnp*Blk messages.

Table 3-48. Block (Blk) Enable Encoding in Address[7:6]

Addr[7:6]	Description
00b	Reserved
01b	Lower 128B block is valid, Lower is defined as Address[7]=0
10b	Upper 128B block, Upper is defined as Address[7]=1
11b	256B block is valid

3.3.9 S2M No Data Response (NDR)

The NDR message class contains completions and indications from the Subordinate to the Master.

Table 3-49. S2M NDR Fields

Field	Width (Bits)			Description	
	68B Flit	256B Flit	PBR Flit		
Valid	1		The valid signal indicates that this is a valid request.		
Opcode	3		Memory Operation: This specifies which, if any, operation needs to be performed on the data and associated information. Details in Table 3-50 .		
MetaField	2		Metadata Field: For devices that support memory with Metadata, this field may be encoded with Meta0-State in response to an M2S Req. For devices that do not support memory with Metadata or in response to an M2S RwD, this field must be set to the No-Op encoding. No-Op may also be used by devices if the Metadata is unreliable or corrupted in the device.		
MetaValue	2		Metadata Value: If MetaField is No-Op, this field is don't care; otherwise, it is Metadata Field as read from memory.		
Tag	16		Tag: This is a reflection of the Tag field sent with the associated M2S Req or M2S RwD.		
LD-ID[3:0]	4	0	Logical Device Identifier: This identifies a logical device within a multiple-logical device. Not applicable in PBR messages where DPID infers this field.		
DevLoad	2		Device Load: Indicates device load as defined in Table 3-51 . Values are used to enforce QoS as described in Section 3.3.4 .		
DPID	0	12	Destination PID		
RSVD	0	10			
Total	30	40	48		

Opcodes for the NDR message class are defined in [Table 3-50](#).

Table 3-50. S2M NDR Opcodes (Sheet 1 of 2)

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

Table 3-50. S2M NDR Opcodes (Sheet 2 of 2)

Opcode	Description	Encoding
BI-ConflictAck	Completion of the Back-Invalidate conflict handshake.	100b
CmpTEE	Completion for Writes (MemWr*) with TEE intent. Does not apply to any M2S Req.	101b
Reserved	Reserved	<Others>

Table 3-51 defines the DevLoad value used in NDR and DRS messages. The encodings were assigned to allow CXL 1.1 backward compatibility such that the 00b value would cause the least impact in the host.

Table 3-51. DevLoad Definition

DevLoad Value	Queuing Delay inside Device	Device Internal Resource Utilization	Encoding
Light Load	Minimal	Readily handles more requests	00b
Optimal Load	Modest to Moderate	Optimally utilized	01b
Moderate Overload	Significant	Limiting request throughput and/or degrading efficiency	10b
Severe Overload	High	Heavily overloaded and/or degrading efficiency	11b

Definition of other fields are the same as for M2S message classes.

3.3.10 S2M Data Response (DRS)

The DRS message class contains memory read data from the Subordinate to the Master.

The fields of the DRS message class are defined in [Table 3-52](#).

Table 3-52. S2M DRS Fields (Sheet 1 of 2)

Field	Width (Bits)			Description	
	68B Flit	256B Flit	PBR Flit		
Valid	1		The valid signal indicates that this is a valid request.		
Opcode	3		Memory Operation: This specifies which, if any, operation needs to be performed on the data and associated information. Details in Table 3-53 .		
MetaField	2		Metadata Field: For devices that support memory with Metadata, this field can be encoded as Meta0-State. For devices that do not, this field must be encoded as No-Op. No-Op encoding may also be used by devices if the Metadata is unreliable or corrupted in the device.		
MetaValue	2		Metadata Value: If MetaField is No-Op, this field is don't care; otherwise, it must encode the Metadata field as read from Memory.		
Tag	16		Tag: This is a reflection of the Tag field sent with the associated M2S Req or M2S Rwd.		
Poison	1		The Poison bit indicates that the data contains an error. The handling of poisoned data is Host specific. See Chapter 12.0 for more details.		
LD-ID[3:0]	4	0	Logical Device Identifier: This identifies a logical device within a multiple-logical device. Not applicable in PBR mode where DPID infers this field.		
DevLoad	2		Device Load: Indicates device load as defined in Table 3-51 . Values are used to enforce QoS as described in Section 3.3.4 .		

Table 3-52. S2M DRS Fields (Sheet 2 of 2)

Field	Width (Bits)			Description
	68B Flit	256B Flit	PBR Flit	
DPID	0	12		Destination PID
TRP	0	1		Trailer Present: Indicates that a trailer is included after the 64B payload. The Trailer size and legal encodings for DRS are defined in Table 3-54 .
RSVD	9	8		
Total	40	40	48	

Table 3-53. S2M DRS Opcodes

Opcode	Description	Encoding
MemData	Memory read data. Sent in response to Reads.	000b
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
MemDataTEE	Same as MemData but in response to MemRd* with TEE attribute.	010b
Reserved	Reserved	<Others>

3.3.10.1 Trailer Present for DRS (256B Flit)

In 256B Flit mode, a Trailer Present (TRP) bit is included with the message header that indicates whether a trailer slot is included with the message. The trailer can be up to 32 bits for DRS.

The TRP bit can be inferred by other field decode as defined in [Table 3-54](#) for DRS. It is included to enable simple decode in the Link Layer.

The Extended Metadata (EMD) trailer is the only trailer supported. The Extended Metadata (EMD) trailer can be up to 32 bits. [Section 8.2.4.31](#) describes the registers that aid in discovery of device's EMD capability and EMD related configuration of the device. The mechanism for discovering the host's EMD capabilities and EMD related configuration of the host is host-specific. The host and the device must be configured in a consistent manner.

Table 3-54. DRS Trailers

Opcode/ Message	MetaField	TRP	Trailer Size Required	Description
MemData/ MemDataTEE	EMS	1	32 bits	Trailer bits[31:0] defined as EMD.
	No-OP/MS0	0	No Trailer	
<Others>	N/A	0	No Trailer	

3.3.11

Forward Progress and Ordering Rules

- Req may be blocked by BISnp to the Host, but Rwd cannot be blocked by BISnp to the Host.
 - This rule impacts Rwd MemWr* to Shared FAM HDM-DB uniquely requiring SnpType=No-Op to avoid causing BISnp to other requesters that are sharing the memory which could deadlock. The resulting is a requirement that the requester must first get ownership of the cacheline using M2S Req message referred to as a 2-phase write as described in [Section 2.4.4](#).
- A CXL.mem Request in the M2S Req channel must not pass a MemRdFwd or a MemWrFwd, if the Request and MemRdFwd or MemWrFwd are to the same cacheline address.
 - Reason: As described in [Table 3-35](#), MemRdFwd and MemWrFwd opcodes, sent on the M2S Req channel are, in fact, responses to CXL.cache D2H requests. The reason the response for certain CXL.cache D2H requests are on CXL.mem M2S Req channel is to ensure subsequent requests from the Host to the same address remain ordered behind it. This allows the host and device to avoid race conditions. Examples of transaction flows using MemRdFwd are shown in [Figure 3-35](#) and [Figure 3-40](#). Apart from the above, there is no ordering requirement for the Req, Rwd, NDR, and DRS message classes or for different addresses within the Req message class.
- NDR and DRS message classes, each, need to be pre-allocated at the request source. This guarantees that the responses can sink and ensures forward progress.
- On CXL.mem, write data is only guaranteed to be visible to a later access after the write is complete.
- CXL.mem requests need to make forward progress at the device without any dependency on any device initiated request except for BISnp messages. This includes any request from the device on CXL.io or CXL.cache.
- S2M and M2S Data transfer of a cacheline must occur with no interleaved transfers.

IMPLEMENTATION NOTE

There are two cases of bypassing with device-attached memory where messages in the M2S Rwd channel may pass messages for the same cacheline address in M2S Req channel.

1. Host generated weakly ordered writes (as showing in [Figure 3-32](#)) may bypass MemRdFwd and MemWrFwd. The result is the weakly ordered write may bypass older reads or writes from the Device.
2. For Device initiated RdCurr to the Host, the Host will send a MemRdFwd to the device after resolving coherency (as shown in [Figure 3-35](#)). After sending the MemRdFwd the Host may have an exclusive copy of the line (because RdCurr does not downgrade the coherency state at the target) allowing the Host to subsequently modify this line and send a MemWr to this address. This MemWr will not be ordered with respect to the previously sent MemRdFwd.

Both examples are legal because weakly ordered stores (in Case #1) and RdCurr (in Case #2) do not guarantee strong consistency.

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 Master'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 Master-issued CXL.mem Req/RwD messages:

- Must not issue a MemRd/MemInv/MemInvNT (MetaValue=I) if the cacheline is buried in Modified, Exclusive, or Shared state.
- Shall not issue a MemRd/MemInv/MemInvNT (MetaValue=S) or MemRdData if the cacheline is buried in Modified or Exclusive state, but is allowed to issue when the host has Shared or Invalid state.
- May issue a MemRd/MemInv/MemInvNT (MetaValue = A) from any state.
- May issue a MemRd/MemInv/MemInvNT (MetaField = No-Op) 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 results in the buried state being downgraded.
- May issue MemClnEvct from Shared or Exclusive state.
- May issue MemWr with SnpType=SnpInv only from I-state. Use of this encoding is not allowed for HDM-DB memory regions in which coherence extends to multiple hosts (e.g., Coherent Shared FAM as described in [Section 2.4.4](#)).
- MemWr with SnpType=No-Op may be issued only from Modified state.

Note:

The Master may silently degrade clean cache state (E to S, E to I, S to I) and as such the Subordinate may have more conservative view of the Master's cache state. This section is discussing cache state from the Master's view.

[Table 3-55](#) 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 3-55. 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
MemRdData						X	X
MemClnEvct					X	X	
MemRd/ MemInv/ MemInvNT	MS0/EMD	A	All Legal Combinations	X ¹	X	X	X
		S				X	X
		I					X
	No-Op	N/A					
	EMD	Explicit No-Op		X ¹	X	X	X
MemWr	All Legal Combinations		No-Op	X			
			SnpInv				X
MemRdFill/ MemRdFillTEE/ MemRdDataTEE/ MemRdTee/ MemWrTEE	N/A (Commands not supported for HDM-D/HDM-DB)						

- 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.

3.4

Transaction Ordering Summary

This section presents CXL ordering rules in a series of tables and descriptions. [Table 3-56](#) captures the upstream ordering cases. [Table 3-57](#) captures the downstream ordering cases.

For CXL.mem and CXL.cache, the term upstream describes traffic on all S2M and D2H message classes, and the term downstream describes traffic on all M2S and H2D message classes, regardless of the physical direction of travel.

Where upstream and downstream traffic coexist in the same physical direction within PBR switches and on Inter Switch Links (ISLs) or on links from a device that issues direct P2P CXL.mem, the upstream and downstream Ordering Tables each apply to their corresponding subset of the traffic and each subset shall be independent and not block one another.

[Table 3-58](#) lists the Device in-out dependence. [Table 3-59](#) lists the Host in-out dependence. Additional detail is provided in [Section 3.2.2.1](#) for CXL.cache and in [Section 3.3.11](#) for CXL.mem.

In [Table 3-56](#) and [Table 3-57](#), the columns represent a first-issued message and the rows represent a subsequently issued message. The table entry indicates the ordering relationship between the two messages. The table entries are defined as follows:

- Yes: The second message (row) must be allowed to pass the first message (column) to avoid deadlock. (When blocking occurs, the second message is required to pass the first message.)
- Y/N: There are no ordering requirements. The second message may optionally pass the first message or may be blocked by it.
- No: The second message must not be allowed to pass the first message. This is required to support the protocol ordering model.

Note:

Passing, where permitted, must not be allowed to cause the starvation of any message class.

Table 3-56. Upstream Ordering Summary

Row Pass Column?	CXL.io TLPs (Col 2-5)	S2M NDR/DRS D2H Rsp/Data (Col 6)	D2H Req (Col 7)	S2M BISnp (Col 13)
CXL.io TLPs (Row A-D)	PCIe Base	Yes(1)	Yes(1)	Yes(1)
S2M NDR/DRS D2H Rsp/Data (Row E)	Yes(1)	a. No(3)	Yes(2)	Yes(2)(4)
		b. Y/N		
D2H Req (Row F)	Yes(1)	Y/N	Y/N	Y/N
S2M BISnp (Row M)	Yes(1)(4)	Y/N	Yes(4)	Y/N

Explanation of row and column headers:

M7 requires BISnp to pass D2H Req in accordance with dependence relationship: D2H Req depends on M2S Req depends on S2M BISnp.

E6a requires that within the NDR channel, BIConflictAck must not pass prior Cmp* messages with the same Cacheline Address (implied by the tag field).

E6b other cases not covered by rule E6a are Y/N.

Color-coded rationale for cells in Table 3-56	
Yes(1)	CXL architecture requirement for ARB/MUX.
Yes(2)	CXL.cachemem: Required for deadlock avoidance.
No(3)	Type 2/3 devices where BIConflictAck must not pass prior Cmp* to the same address.
Yes(4)	Required for deadlock avoidance with the introduction of the BISnp channel. For CXL.io Unordered I/O, this is necessary because Unordered I/O can trigger BISnp.

Table 3-57. Downstream Ordering Summary

Row Pass Column?	CXL.io TLPs (Col 2-5)	M2S Req (Col 8)	M2S Rwd (Col 9)	H2D Req (Col 10)	H2D Rsp (Col 11)	H2D Data (Col 12)	M2S BIRsp (Col 14)
CXL.io TLPs (Row A-D)	PCIe Base	Yes(1)	Yes(1)	Yes(1)	Yes(1)	Yes(1)	Yes(1)
M2S Req (Row G)	Yes(1)	a. No(5)	Y/N	Y/N(3)	Y/N	Y/N	Y/N
		b. Y/N					
M2S Rwd (Row H)	Yes(1)(6)	a. Yes(6)	Y/N	Yes(3)	Y/N	Y/N	Y/N
		b. Y/N					
H2D Req (Row I)	Yes(1)	Yes(2)(6)	a. Yes(2)	Y/N	a. No(4)	Y/N(3)	Y/N
			b. Y/N		b. Y/N		
H2D Rsp (Row J)	Yes(1)	Yes(2)	Yes(2)	Yes(2)	Y/N	Y/N	Y/N
H2D Data (Row K)	Yes(1)	Yes(2)	Yes(2)	Yes(2)	Y/N	Y/N	Y/N
M2S BIRsp (Row N)	Yes(1)(6)	Yes(2)	Yes(2)	Yes(2)	Y/N	Y/N	Y/N

Explanation of row and column headers:

In Downstream direction pre-allocated channels are kept separate because of unique ordering requirements in each.

Color-coded rationale for cells in Table 3-57	
Yes(1)	CXL architecture requirement for ARB/MUX.
Yes(2)	CXL.cachemem: Required for deadlock avoidance.
Yes(3)	CXL.cachemem: Performance optimization.
Y/N(3)	CXL.cachemem: Non-blocking recommended for performance optimization.
No(4)	Type 1/2 device: Snoop push GO requirement.
No(5)	Type 2 device: MemRd*/MemInv* push Mem*Fwd requirement.
Yes(6)	Required for deadlock avoidance with the introduction of the BISnp channel.

Explanation of table entries:

G8a MemRd*/MemInv* must not pass prior Mem*Fwd messages to the same cacheline address. This rule is applicable only for HDM-D memory regions in devices which result in receiving Mem*Fwd messages (Type 3 devices with no HDM-D don't need to implement this rule). This rule does not apply to Type 2 devices that implement the HDM-DB memory region which use the BI* channels because they do not support Mem*Fwd.

G8b All other cases not covered by rule G8a do not have ordering requirements (Y/N).

H8a applies to components that support the BISnp/BIRsp message classes to ensure that the RWD channel can drain to the device even if the Req channel is blocked.

H8b applies to components that do not support the BISnp/BIRsp message classes.

I9a applies for PBR-capable switches, for ISLs, and for devices that can initiate P2P CXL.mem. (Possible future use case for Host-to-Host CXL.mem will require host to apply this ordering rule.)

I9b applies to all other cases.

I11a Snoops must not pass prior GO* messages to the same cacheline address. GO messages do not carry the address, so implementations where address cannot be inferred from UQID in the GO message will need to strictly apply this rule across all messages.

I11b Other case not covered by I11a are Y/N.

Table 3-58. Device In-Out Ordering Summary

Row (in) Independent of Column (out)?	CXL.io TLPs (Col A-D)	S2M NDR/DRS D2H Rsp/Data (Col E)	D2H Req (Col F)	S2M BISnp (Col M)	M2S Req (Col N) ¹	M2S RWD (Col O) ¹	M2S BIRsp (Col P) ¹
CXL.io TLPs (Row 2-5)	PCIe Base	Y/N(1)	Y/N(1)	Y/N(1)	Y/N(1)	Y/N(1)	Y/N(1)
		Yes(3)	Yes(3)	Yes(3)	Yes(3)	Yes(3)	Yes(3)
M2S Req (Row 8)	Yes(1)	Y/N	Yes(2)	Y/N	Yes(2)	Y/N	Y/N
M2S RWD (Row 9)	Yes(1)(2)	Y/N	Yes(2)	Yes(2)	Yes(2)	Yes(2)	Y/N
H2D Req (Row 10)	Yes(1)	Y/N	Yes(2)	Yes(2)	Yes(2)	Yes(2)	Y/N
H2D Rsp (Row 11)	Yes(1)	Yes(2)	Yes(2)	Yes(2)	Yes(2)	Yes(2)	Yes(2)
H2D Data (Row 12)	Yes(1)	Yes(2)	Yes(2)	Yes(2)	Yes(2)	Yes(2)	Yes(2)
M2S BIRsp (Row 14)	Yes(1)(2)	Yes(2)	Yes(2)	Yes(2)	Yes(2)	Yes(2)	Yes(2)
S2M NDR/DRS (Row 15)¹	Yes(1)	Yes(2)	Yes(2)	Yes(2)	Yes(2)	Yes(2)	Y/N
S2M BISnp (Row 16)¹	Yes(1)	Y/N	Yes(2)	Yes(2)	Yes(2)	Y/N	Y/N

1. These rows and columns are supported only by devices that have Direct P2P CXL.mem enabled.

In the device ordering, the row represents incoming message class and the column represents the outgoing message class. The cases in this table show when incoming must be independent of outgoing (Yes) and when it is allowed to block incoming based on outgoing (Y/N).

Color-coded rationale for cells in Table 3-58							
Yes(1)	CXL.cachemem is independent of outgoing CXL.io.						
Y/N(1)	CXL.io traffic, except UIO Completions, may be blocked by CXL.cachemem.						
Yes(2)	CXL.cachemem: Required for deadlock avoidance.						
Yes(3)	CXL UIO completions are independent of CXL.cachemem.						

Table 3-59. Host In-Out Ordering Summary

Row (in) Independent of Column (out)?	CXL.io TLPs (Col A-D)	M2S Req (Col G)	M2S Rwd (Col H)	H2D Req (Col I)	H2D Rsp (Col J)	H2D Data (Col K)	M2S BIRsp (Col N)
CXL.io TLPs (Row 2-5)	PCIe Base	Y/N(1)	Y/N(1)	Y/N(1)	Y/N(1)	Y/N(1)	Y/N(1)
		Yes(3)	Yes(3)	Yes(3)	Yes(3)	Yes(3)	Yes(3)
S2M NDR/DRS D2H Rsp/Data (Row 6)	Yes(1)(2)	Yes(2)	Yes(2)	Yes(2)	Y/N	Y/N	Y/N
D2H Req (Row 7)	Yes(1)	Y/N	Y/N	Y/N	Y/N	Y/N	Y/N
S2M BISnp (Row 13)	Yes(1)(2)	Yes(2)	Y/N	Y/N	Y/N	Y/N	Y/N

In the host ordering, the row represents incoming message class and the column represents the outgoing message class. The cases in this table show when incoming must be independent of outgoing (Yes) and when it is allowed to block incoming based on outgoing (Y/N).

Color-coded rationale for cells in Table 3-59	
Yes(1)	Incoming CXL.cachemem must not be blocked by outgoing CXL.io.
Y/N(1)	Incoming CXL.io may be blocked by outgoing CXL.cachemem.
Yes(2)	CXL.cachemem: Required for deadlock avoidance.
Yes(3)	CXL UIO completions are independent of CXL.cachemem.

3.5 Transaction Flows to Device-attached Memory

3.5.1 Flows for Back-Invalidate Snoops on CXL.mem

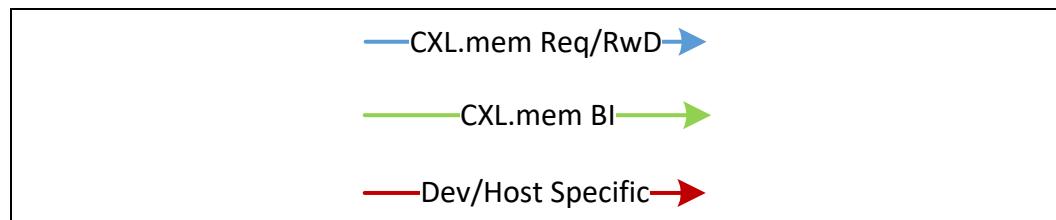
3.5.1.1 Notes and Assumptions

The Back-Invalidate Snoop (BISnp) channel provides a dedicated channel S2M to allow the owner of an HDM region to snoop a host that may have a cached copy of the line. The forward progress rules as defined in [Section 3.4](#) ensure that the device can complete the BISnp while blocking new requests (M2S Req).

The term Snoop Filter (SF) in the following diagrams is a structure in the device that is inclusively tracking any host caching of device memory and is assumed to have a size that may be less than the total possible caching in the host. The Snoop Filter is kept

inclusive of host caching by sending “Back-Invalidate Snoops” to the host when it becomes full. This full trigger that forces the BISnp is referred to as “SF Victim”. In the diagrams, an “SF Miss” that is caused by an M2S request implies that the device must also allocate a new SF entry if the host is requesting a cached copy of the line. When allocating an SF entry, it may also trigger an SF Victim for a different cacheline address if the SF is full. [Figure 3-20](#) provides the legend for the Back-Invalidate Snoop flow diagrams that appear in the subsections that follow. The “CXL.mem BI” type will cover the BI channel messages and any conflict message/flow (e.g., BIConflict) that flow on the Rwd channels. Note that the “Dev/Host Specific” messages are just short-hand flows for the type of flow expected in the host or device.

Figure 3-20. Flows for Back-Invalidate Snoops on CXL.mem Legend

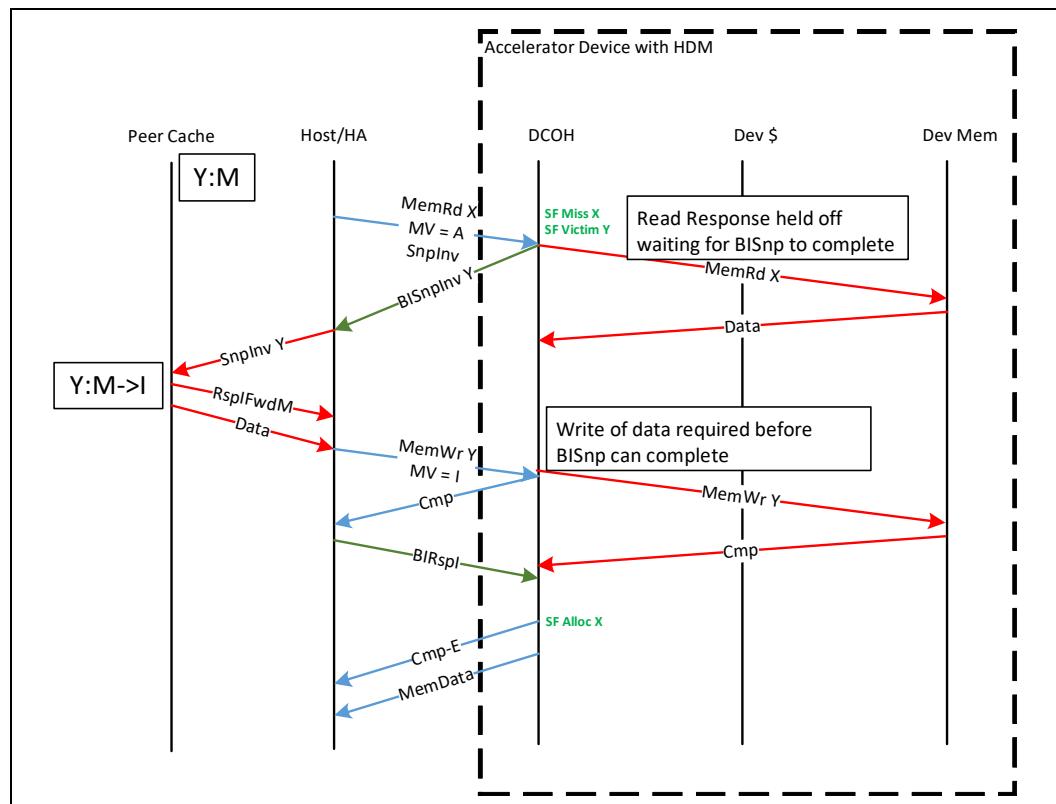


3.5.1.2 BISnp Blocking Example

[Figure 3-21](#) starts out with MemRd that is an SF Miss in the device. The SF is full, which prevents SF allocation; thus, the device must create room in the SF by triggering an SF Victim for Address Y before it can complete the read. In this example, the read to device memory Address X is started in parallel with the BISnpInv to Address Y, but the device will be unable to complete the MemRd until it can allocate an SF which requires the BISnp to Y to complete. As part of the BISnpInv, the host finds modified data for Y which must be flushed to the device before the BISnpInv can complete. The device completes the MemWr to Y, which allows the host to complete the BISnpInv to Y with the BIRspI. That completion allows the SF allocation to occur for Address X, which enables the Cmp-E and MemData to be sent.

EVALUATION COPY

Figure 3-21. Example BISnp with Blocking of M2S Req



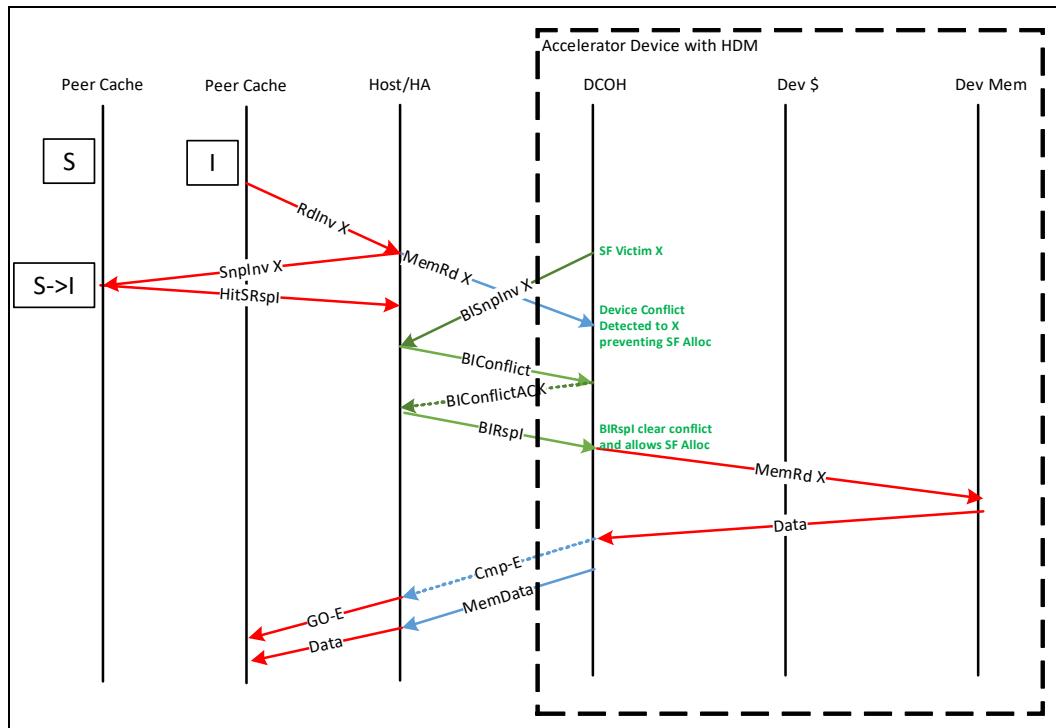
3.5.1.3 Conflict Handling

A conflict is defined as a case where S2M BISnp and M2S Req are active at the same time to the same address. There are two cases to consider: Early Conflict and Late Conflict. The two cases are ambiguous to the host side of the link until observation of a Cmp message relative to BIConflictAck. The conflict handshake starts when by the host detecting a BISnp to the same address as a pending Req. The host sends a BIConflict with the Tag of the M2S Req and device responds to a BIConflict with a BIConflictAck which must push prior Cmp* messages within the NDR channel. This ordering relationship is fundamental to allow the host to correctly resolve the two cases.

The Early Conflict case in Figure 3-22 is defined as a case where M2S Req is blocked (or in flight) at the device while S2M BISnp is active. The host observing BIConflictAck before Cmp-E determines the M2S MemRd is still pending so that it can reply with RspI.

EVALUATION COPY

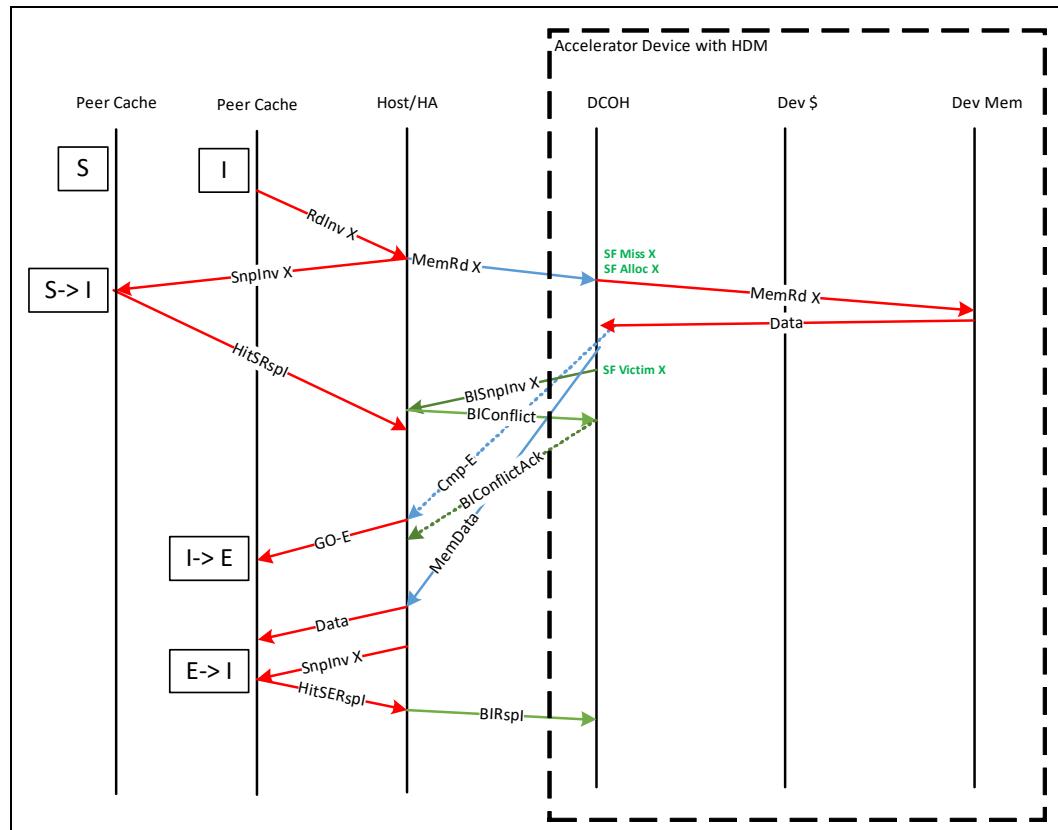
Figure 3-22. BISnp Early Conflict



Late conflict is captured in [Figure 3-23](#) and is defined as the case where M2S Req was processed and completions are in flight when BISnp is started. In the example below, the Cmp-E message is observed at the host before BIConflictAck, so the host must process the BISnpInv with E-state ownership, which requires it to degrade E to I before completing the BISnpInv with BIRspI. Note that MemData has no ordering requirement and can be observed either before or after the BIConflictAck, although this example shows it after which delays the host's ability to immediately process the internal SnpInv X.

EVALUATION COPY

Figure 3-23. BISnp Late Conflict



3.5.1.4

Block Back-Invalidate Snoops

To support increased efficient snooping the BISnp channel defines messages that can Snoop multiple cachelines in the host in a single message. These messages support either 2 or 4 cachelines where the base address must be naturally aligned with the length (128B or 256B). The host is allowed to respond with either a single block response or individual snoop responses per cacheline.

Figure 3-24 is an example of a Block response case. In this example the host receives the BISnpInvBlk for Y, which is a 256B block. Internally the host logic is showing resolving coherence by snooping Y0 and Y2 and the host HA tracker knows the other portions of the block Y1 and Y3 are already in the invalid state, so it does not need to snoop for that portion of the 256B block. Once snoop responses for Y0 and Y2 are completed, the Host HA can send the BIRspIBlk indicating that the entire block is in I-state within the host, thereby allowing the device to have Exclusive access to the block. This results in the SF in I-state for the block and the device cache in E-state.

EVALUATION COPY

Figure 3-24. Block BISnp with Block Response

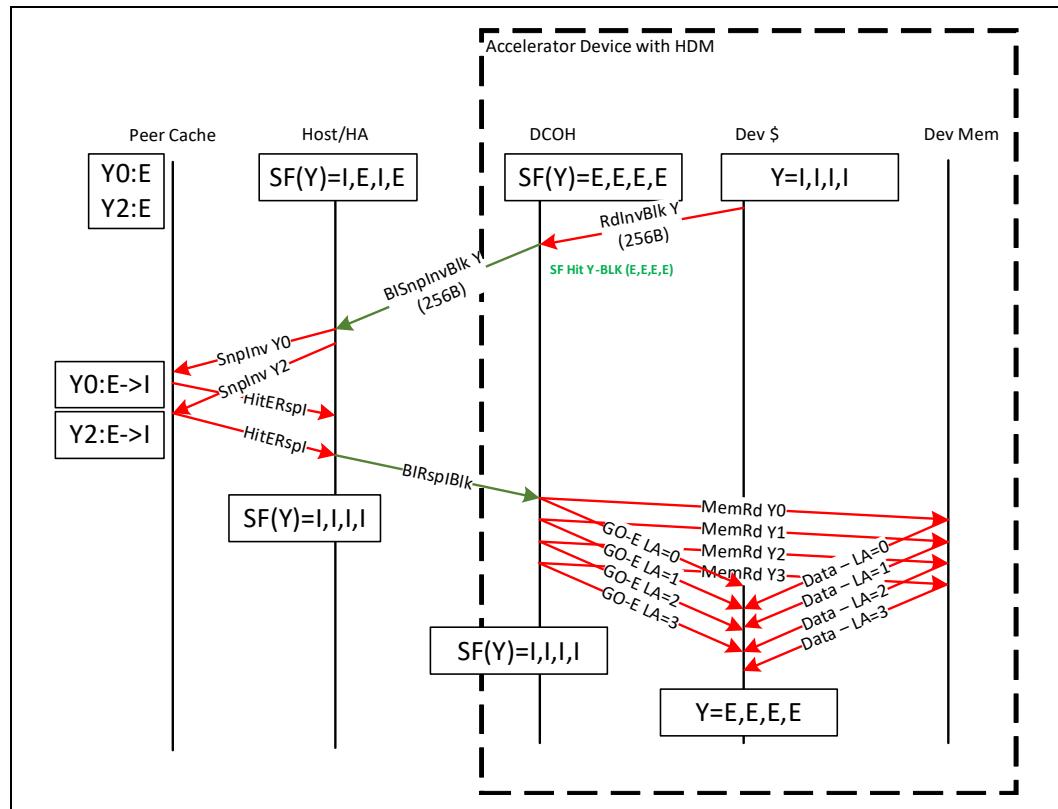
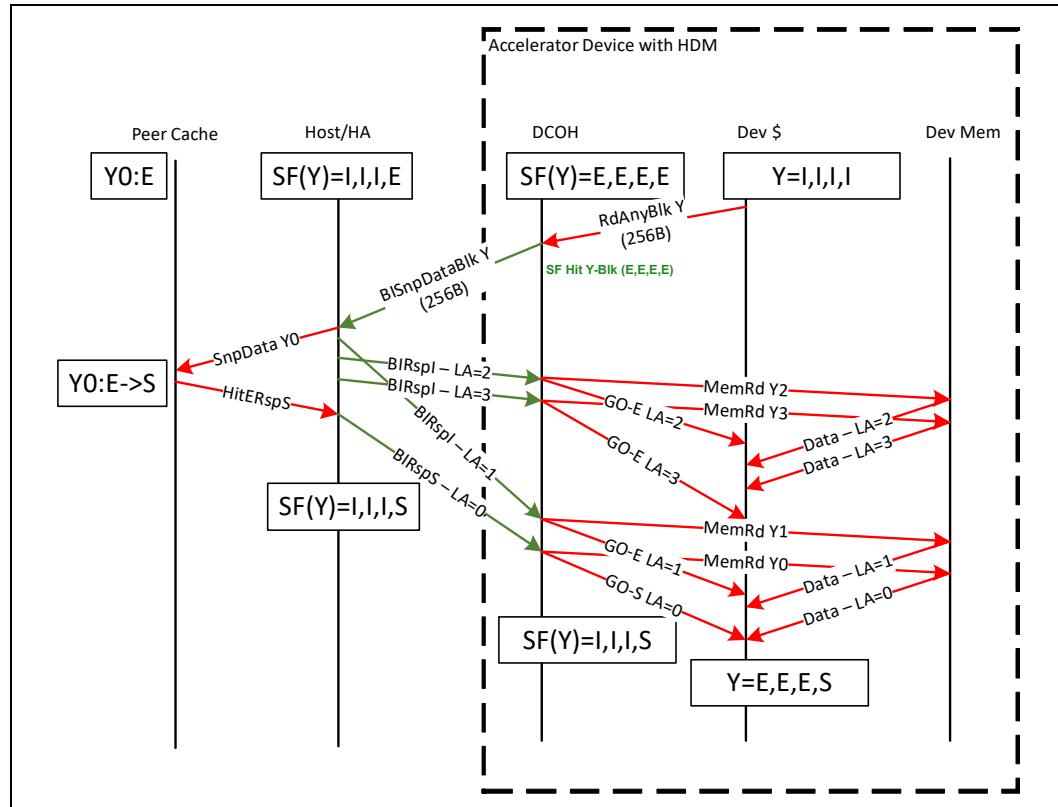


Figure 3-25 is an example where the host sends individual cacheline responses on CXL.mem for each cacheline of the block. The host encodes the 2-bit Lower Address (LowAddr) of the cacheline (Address[7:6]) with each cacheline response to allow the device to determine for which portion of the block the response is intended. The device may see the response messages in any order, which is why LA must be explicitly sent. In a Block, BISnp Address[7:6] is used to indicate the offset and length of the block as defined in [Table 3-48](#) and is naturally aligned to the length.

Figure 3-25. Block BISnp with Cacheline Response



3.5.2 Flows for Type 1 Devices and Type 2 Devices

3.5.2.1 Notes and Assumptions

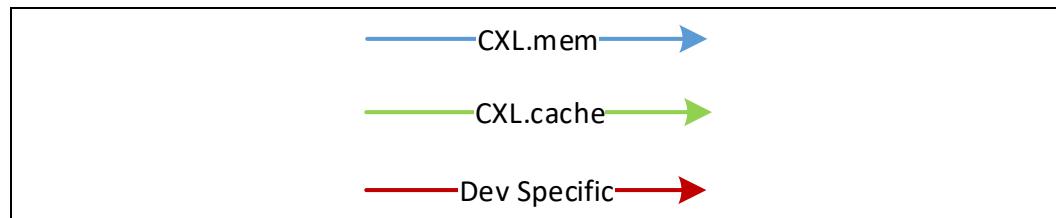
The transaction flow diagrams below are intended to be illustrative of the flows between the Host and device for access to device-attached Memory using the Bias-Based Coherency mechanism described in [Section 2.2.2](#). However, these flows are not comprehensive of every Host and device interaction. The diagrams below make the following assumptions:

- The device contains a coherency engine which is called DCOH in the diagrams below.
- The DCOH contains a Snoop Filter which tracks any caches (called Dev cache) implemented on the device. This is not strictly required, and the device is free to choose an implementation specific mechanism as long as the coherency rules are obeyed.
- The DCOH contains host coherence tracking logic for the device-attached memory. This tracking logic is referred to as a Bias Table in the context of the HDM-D memory region. For HDM-DB, it is referred to as a Directory or a Host Snoop Filter. The implementation of this is device specific.
- The device-specific aspects of the flow, illustrated using red flow arrows, need not conform exactly to the diagrams below. These can be implemented in a device-specific manner.
- Device-attached Memory exposed in a Type 2 device can be either HDM-D or HDM-DB. HDM-D will resolve coherence using a request that is issued on CXL.cache and

the Host will send a Mem*Fwd as a response on the CXL.mem Req channel. The HDM-DB region uses the separate CXL.mem BISnp channel to manage coherence with detailed flows covered in [Section 3.5.1](#). This section will indicate where the flows differ.

[Figure 3-26](#) provides the legend for the diagrams that follow.

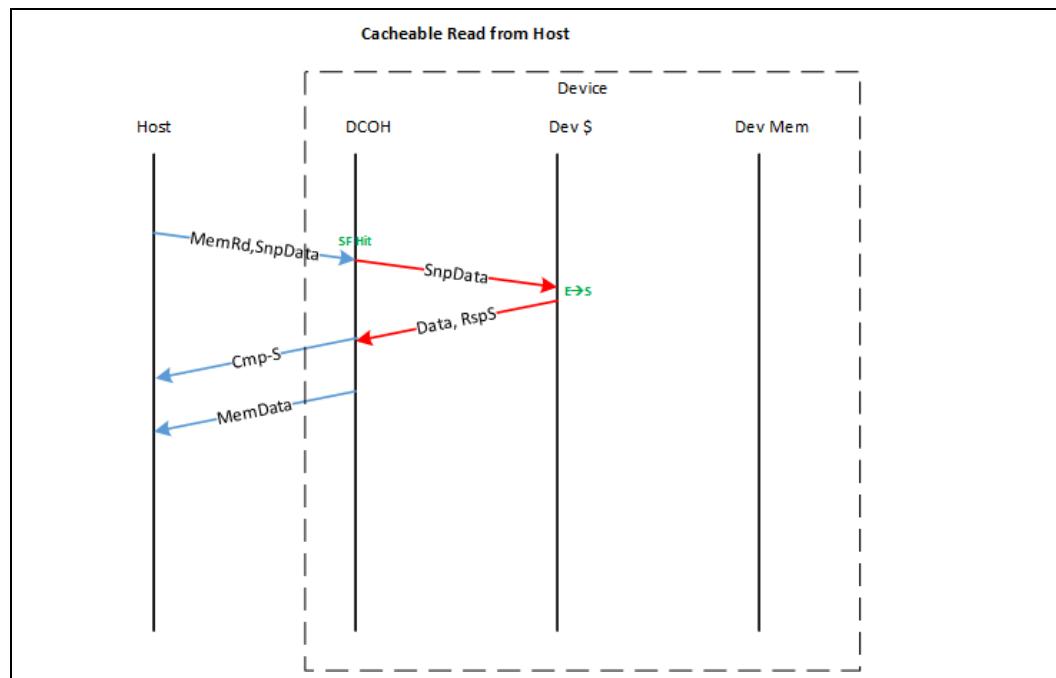
Figure 3-26. Flows for Type 1 Devices and Type 2 Devices Legend



3.5.2.2 Requests from Host

Please note that the flows shown in this section ([Requests from Host](#)) do not change on the CXL interface regardless of the bias state of the target region. This effectively means that the device needs to give the Host a consistent response, as expected by the Host and shown in [Figure 3-27](#).

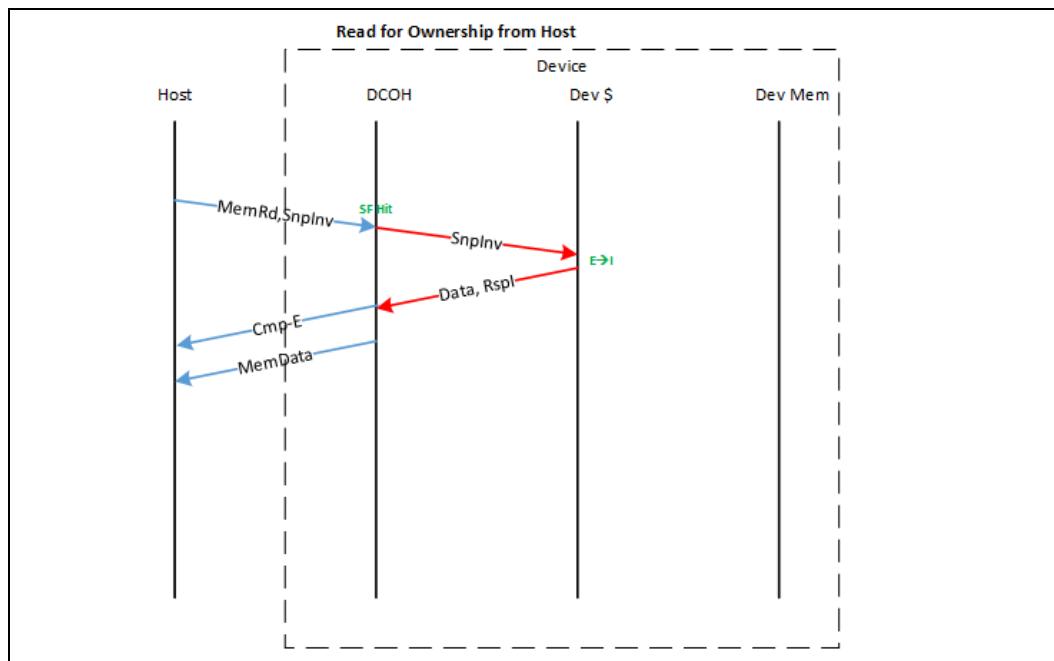
Figure 3-27. Example Cacheable Read from Host



In the above example, the Host requested a cacheable non-exclusive copy of the line. The non-exclusive aspect of the request is communicated using the "SnpData" semantic. In this example, the request got a snoop filter hit in the DCOH, which caused the device cache to be snooped. The device cache downgraded the state from Exclusive to Shared and returned the Shared data copy to the Host. The Host is told of the state of the line using the Cmp-S semantic.

EVALUATION COPY

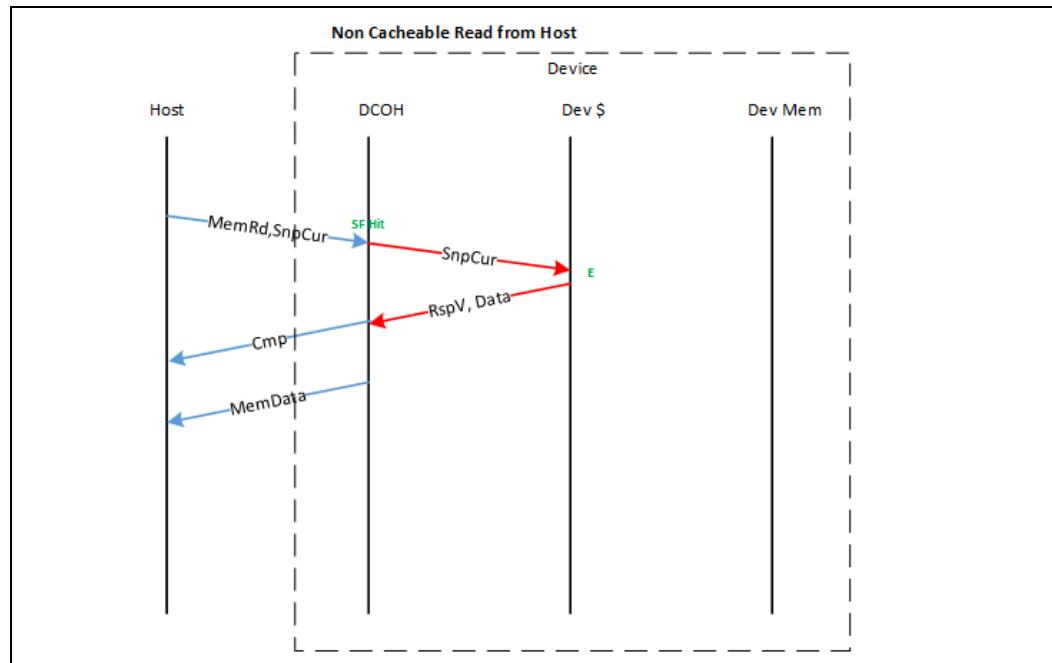
Figure 3-28. Example Read for Ownership from Host



In the above example, the Host requested a cacheable exclusive copy of the line. The exclusive aspect of the request is communicated using the "SnpInv" semantic, which asks the device to invalidate its caches. In this example, the request got a snoop filter hit in the DCOH, which caused the device cache to be snooped. The device cache downgraded the state from Exclusive to Invalid and returned the Exclusive data copy to the Host. The Cmp-E semantic is used to communicate the line state to the Host.

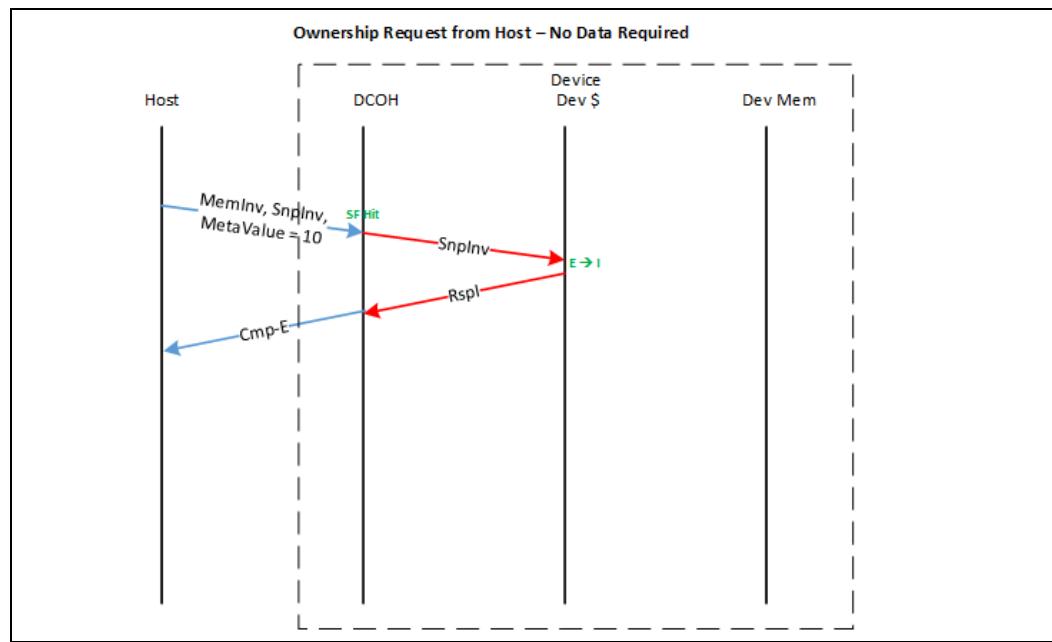
EVALUATION COPY

Figure 3-29. Example Non Cacheable Read from Host



In the above example, the Host requested a non-cacheable copy of the line. The non-cacheable aspect of the request is communicated using the "SnpCur" semantic. In this example, the request got a snoop filter hit in the DCOH, which caused the device cache to be snooped. The device cache did not need to change its caching state; however, it gave the current snapshot of the data. The Host is told that it is not allowed to cache the line using the Cmp semantic.

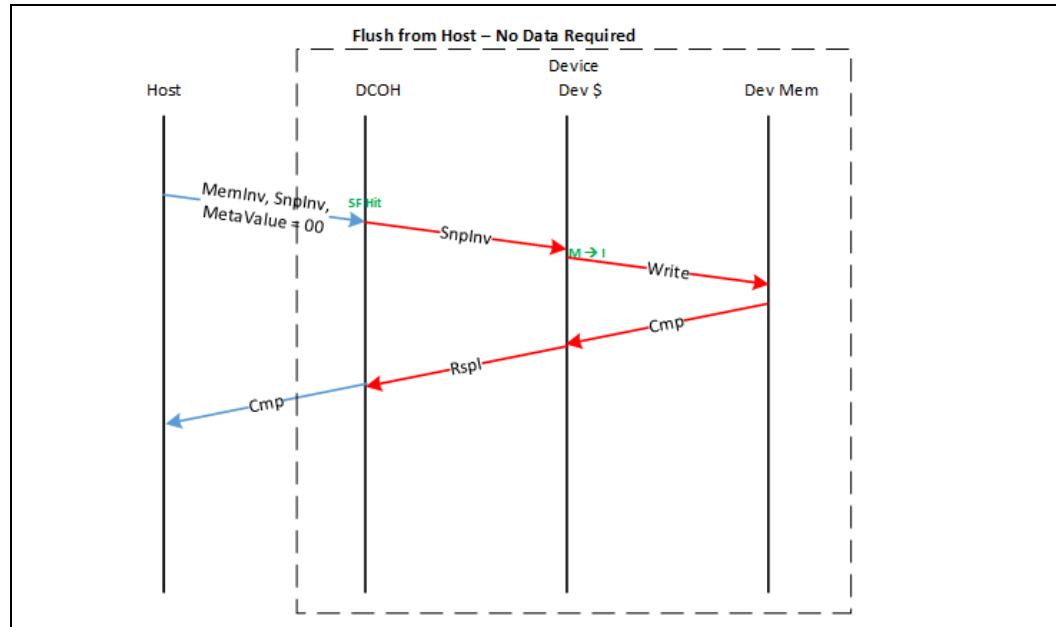
Figure 3-30. Example Ownership Request from Host - No Data Required



EVALUATION COPY

In the above example, the Host requested exclusive access to a line without requiring the device to send data. It communicates that to the device using an opcode of MemInv with a MetaValue of 10b (Any), which is significant in this case. It also asks the device to invalidate its caches with the SnpInv command. The device invalidates its caches and gives exclusive ownership to the Host as communicated using the Cmp-E semantic.

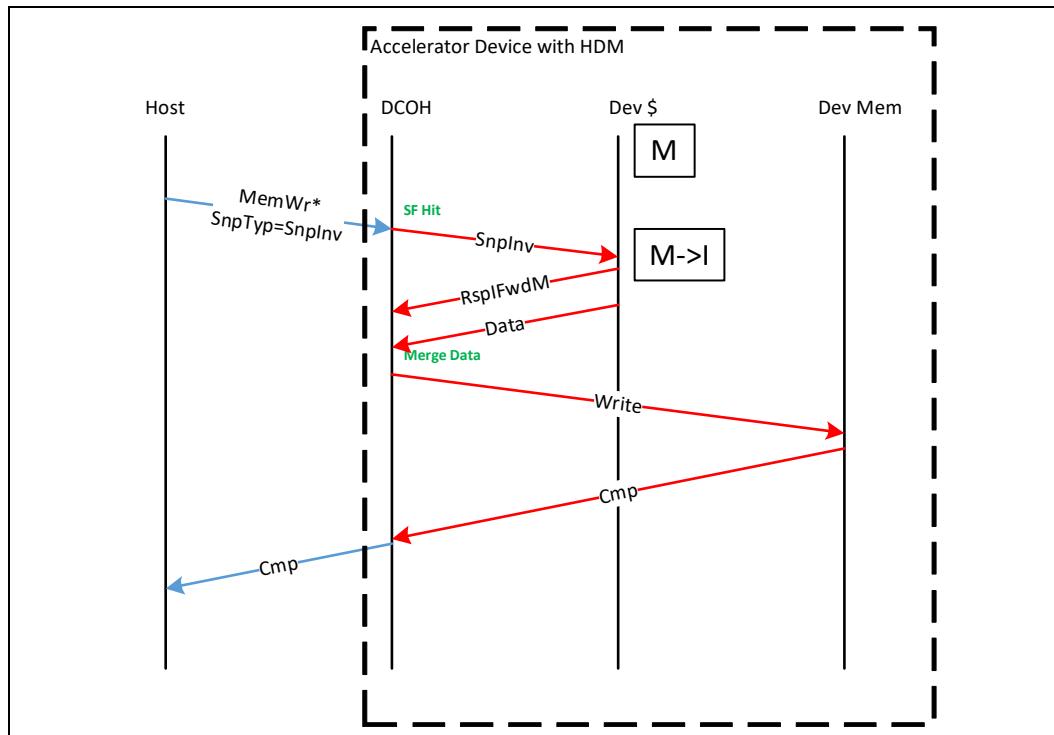
Figure 3-31. Example Flush from Host



In the above example, the Host wants to flush a line from all caches, including the device's caches, to device memory. To do so, it uses an opcode of MemInv with a MetaValue of 00b (Invalid) and a SnpInv. The device flushes its caches and returns a Cmp indication to the Host.

EVALUATION COPY

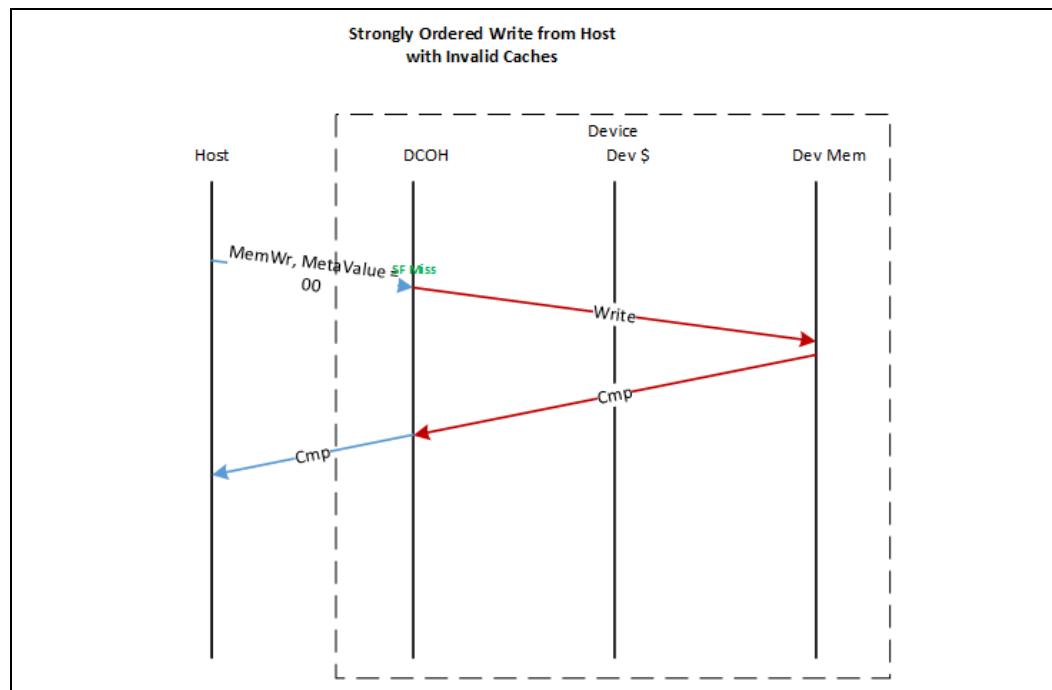
Figure 3-32. Example Weakly Ordered Write from Host



In the above example, the Host issues a weakly ordered write (partial or full line). The weakly ordered semantic is communicated by the embedded SnpInv. In this example, the device had a copy of the line cached. This resulted in a merge within the device before writing it back to memory and sending a Cmp indication to the Host. The term "weakly ordered" in this context refers to an expected-use model in the host CPU in which ordering of the data is not guaranteed until after the Cmp message is received. This is in contrast to a "data visibility is guaranteed with the host" CPU cache in M-state.

EVALUATION COPY

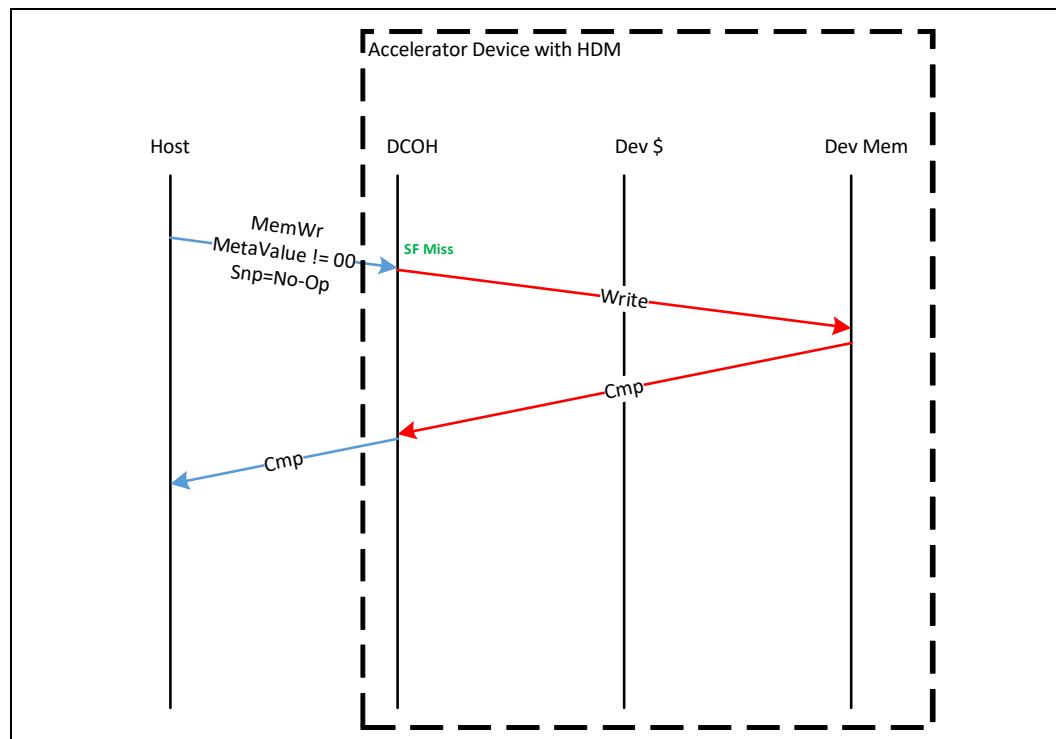
Figure 3-33. Example Write from Host with Invalid Host Caches



In the above example, the Host performed a write while guaranteeing to the device that it no longer has a valid cached copy of the line. The fact that the Host didn't need to snoop the device's caches means that the Host previously acquired an exclusive copy of the line. The guarantee on no valid cached copy is indicated by a MetaValue of 00b (Invalid).

EVALUATION COPY

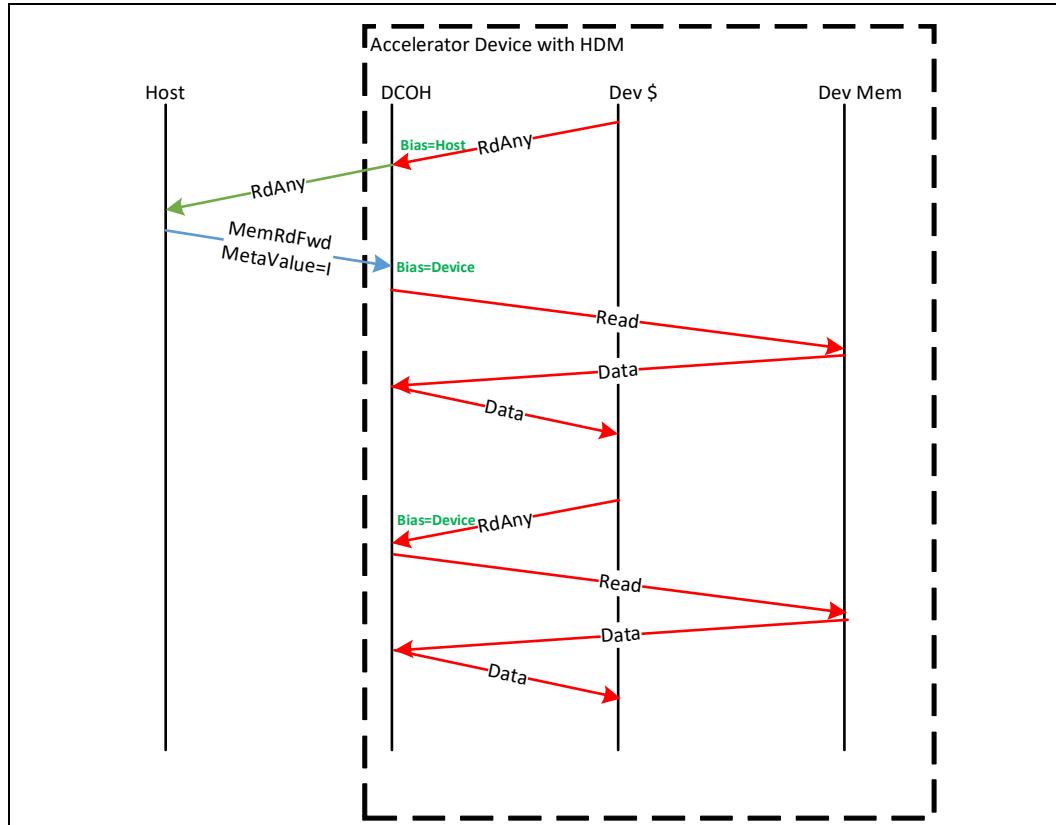
Figure 3-34. Example Write from Host with Valid Host Caches



The above example is the same as the previous one except that the Host chose to retain a valid cacheable copy of the line after the write. This is communicated to the device using a MetaValue of not 00b (Invalid).

3.5.2.3 Requests from Device in Host and Device Bias

Figure 3-35. Example Device Read to Device-attached Memory (HDM-D)



The two flows in [Figure 3-35](#) both start with an internal CXL.cache request (RdAny) that targets the device's HDM-D address region.

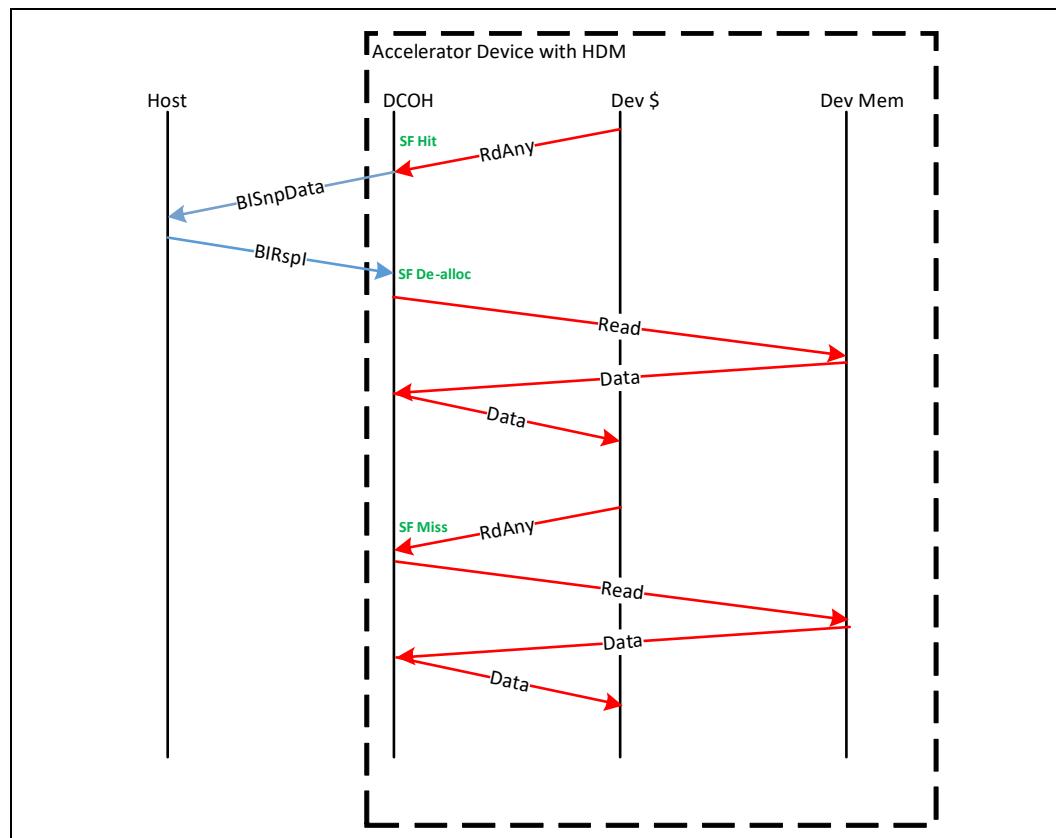
In the first flow in [Figure 3-35](#), a device read to device attached memory happened to find the line in Host bias. Because it is in Host bias, the device needs to send the request to the Host to resolve coherency. The Host, after resolving coherency, sends a MemRdFwd on CXL.mem to complete the transaction, at which point the device can internally complete the read.

In the second flow in [Figure 3-35](#), the device read happened to find the line in Device Bias. Because it is in Device Bias, the read can be completed entirely within the device itself and a request doesn't need to be sent to the Host.

The same device request is shown in [Figure 3-36](#), but in this case the target is the HDM-DB address region, meaning that the BISnp channel is used to resolve coherence with the host. In this flow, the difference is that the SF Hit (similar to BIAS=host) indicates that the host could have a cached copy, so BISnPData is sent to the host to resolve coherence. After the host resolves coherence, the host responds with BIRspI indicating that the host is in I-state and that the device can proceed to access its data.

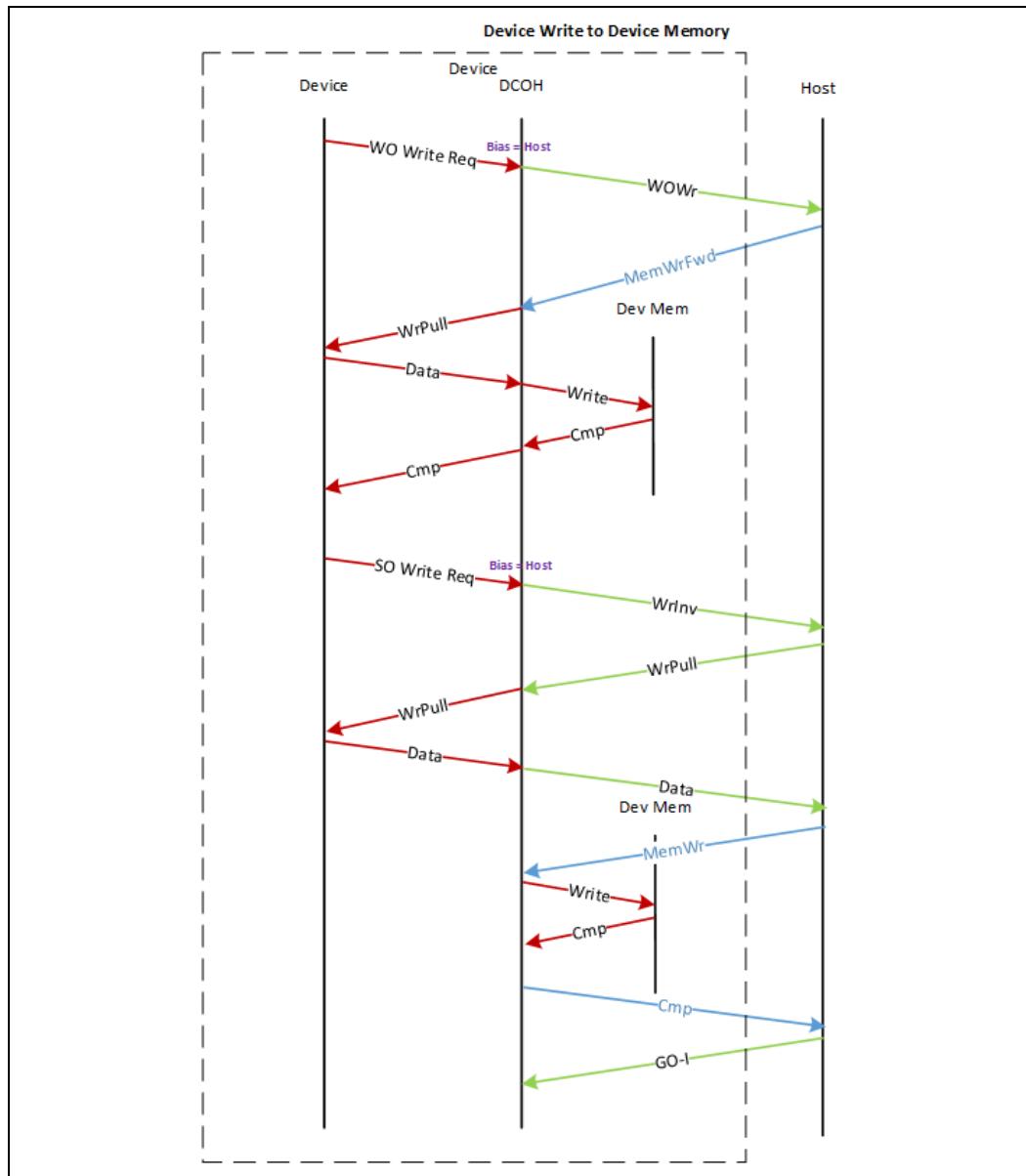
EVALUATION COPY

Figure 3-36. Example Device Read to Device-attached Memory (HDM-DB)



EVALUATION COPY

Figure 3-37. Example Device Write to Device-Attached Memory in Host Bias (HDM-D)



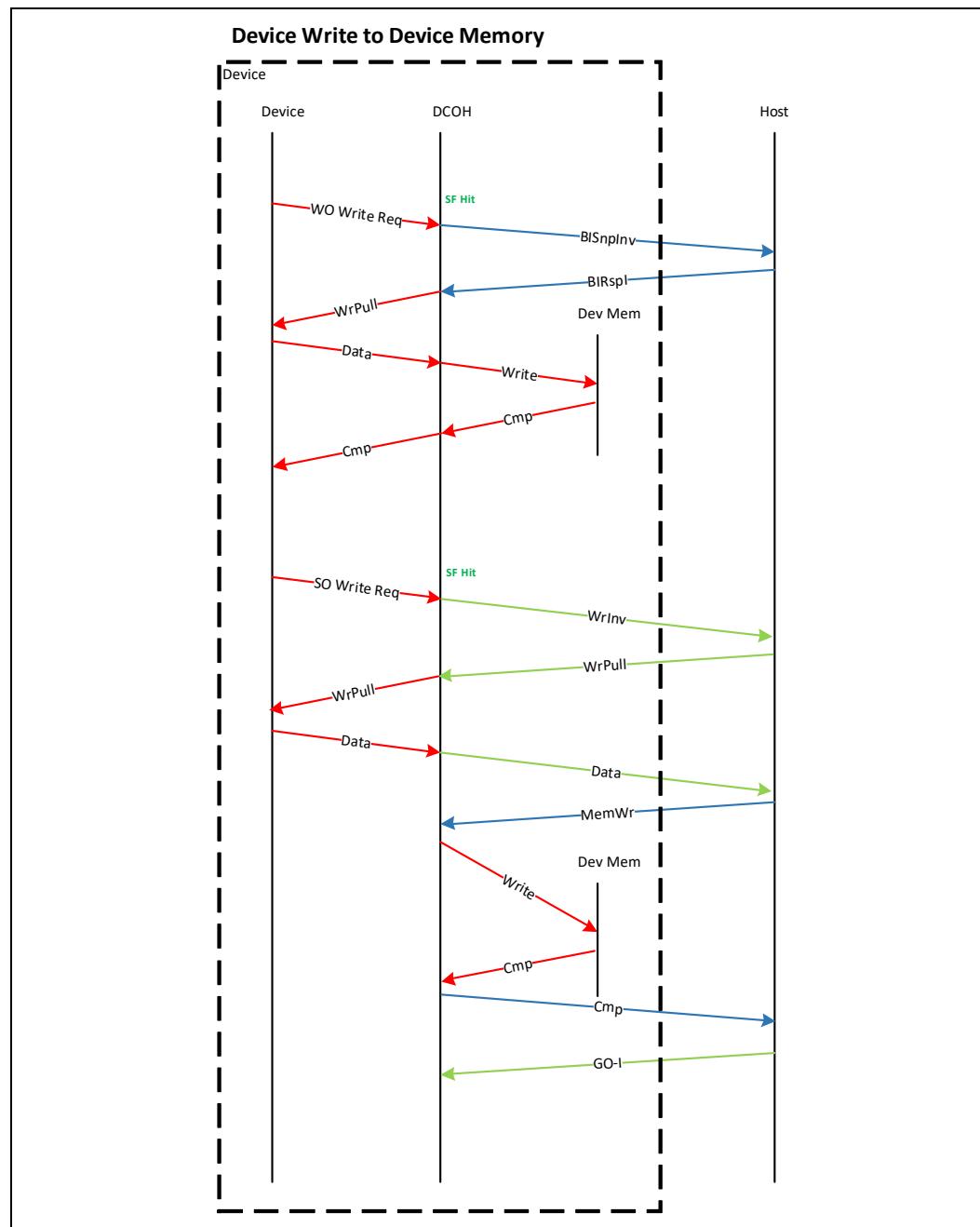
There are two flows shown above in Figure 3-37 for the HDM-D region. Both start with the line in Host Bias: a weakly ordered write request and a strongly ordered write request.

In the case of the weakly ordered write request, the request is issued by the device to the Host to resolve coherency. The Host resolves coherency and sends a CXL.mem MemWrFwd opcode, which carries the completion for the WOWrInv* command on CXL.cache. The CQID associated with the CXL.cache WOWrInv* command is reflected in the Tag of the CXL.mem MemWrFwd command. At this point, the device is allowed to complete the write internally. After sending the MemWrFwd, because the Host no longer prevents future accesses to the same line, this is considered a weakly ordered write.

EVALUATION COPY

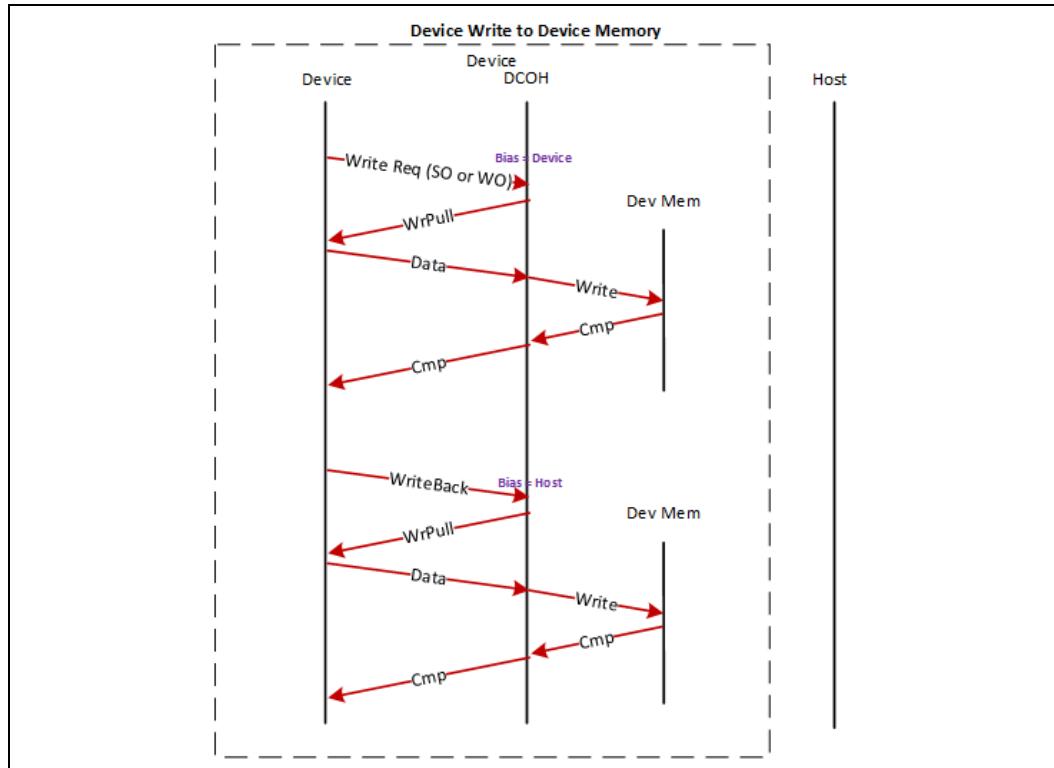
In the second flow, the write is strongly ordered. To preserve the strongly ordered semantic, the Host can prevent future accesses to the same line while this write completes. However, as can be seen, this involves two transfers of the data across the link, which is inefficient. Unless strongly ordered writes are absolutely required, better performance can be achieved with weakly ordered writes.

Figure 3-38. Example Device Write to Device-attached Memory in Host Bias (HDM-DB)



[Figure 3-38](#) for HDM-DB is in contrast to [Figure 3-37](#) for the HDM-D region. In the HDM-DB flow, the BISnp channel in the CXL.mem protocol is used to resolve coherence with the host for the internal weakly ordered write. The strongly ordered write follows the same flow for both HDM-DB and HDM-D.

Figure 3-39. Example Device Write to Device-attached Memory



Again, two flows are shown above in [Figure 3-39](#). In the first case, if a weakly or strongly ordered write finds the line in Device Bias, the write can be completed entirely within the device without having to send any indication to the Host.

The second flow shows a device writeback to device-attached memory. Note that if the device is doing a writeback to device-attached memory, regardless of bias state, the request can be completed within the device without having to send a request to the Host.

The HDM-DB vs. HDM-D regions have the same basic assumption in these flows such that no interaction is required with the host.

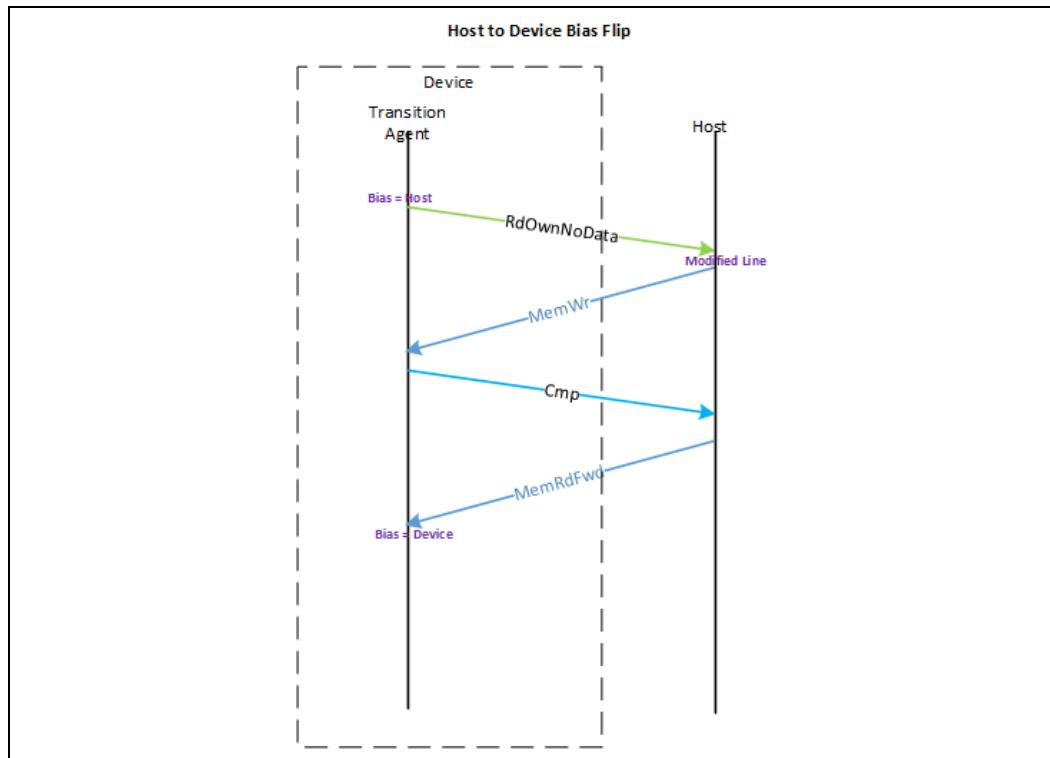
Figure 3-40. Example Host to Device Bias Flip (HDM-D)

Figure 3-40 captures the “Bias Flip” flows for HDM-D memory. For the HDM-DB memory region, see [Section 3.3.3](#) for details regarding how this case is handled. Please note that the MemRdFwd will carry the CQID of the RdOwnNoData transaction in the Tag. The reason for putting the RdOwnNoData completion (MemRdFwd) on CXL.mem is to ensure that subsequent M2S Req Channel requests from the Host to the same address are ordered behind the MemRdFwd. This allows the device to assume ownership of a line as soon as the device receives a MemRdFwd without having to monitor requests from the Host.

3.5.3 Type 2 Memory Flows and Type 3 Memory Flows

3.5.3.1 Speculative Memory Read

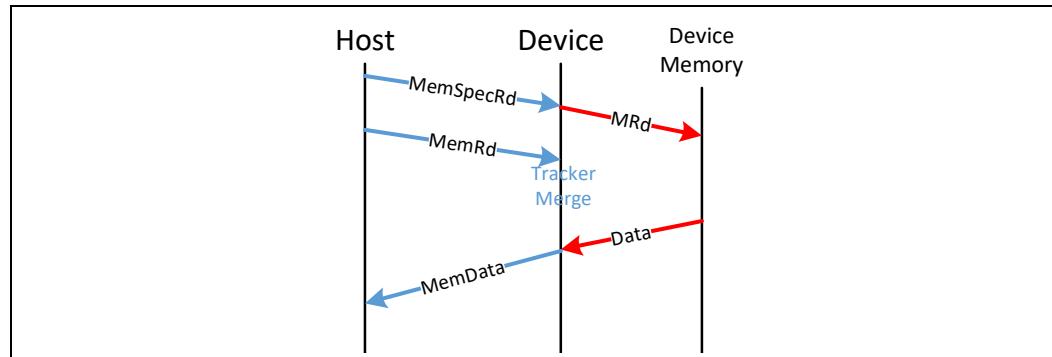
To support latency saving, CXL.mem includes a speculative memory read command (MemSpecRd) which is used to start memory access before the home agent has resolved coherence. This command does not receive a completion message and can be arbitrarily dropped. The host, after resolving coherence, may issue a demand read (i.e., MemRd or MemRdData) that the device should merge with the earlier MemSpecRd to achieve latency savings. See [Figure 3-41](#) for an example of this type of flow.

The MemSpecRd command can be observed while another memory access is in progress in the device to the same cacheline address. In this condition, it is recommended that the device drops the MemSpecRd.

To avoid performance impact, it is recommended that MemSpecRd commands are treated as low priority to avoid adding latency to demand accesses. Under loaded conditions the MemSpecRd can hurt performance because of the extra bandwidth it

consumes and should be dropped when loading of memory or loading of the CXL link is detected. QoS Telemetry data as indicated by the DevLoad field is one way that loading of memory can be detected in the host or switch.

Figure 3-41. Example MemSpecRd



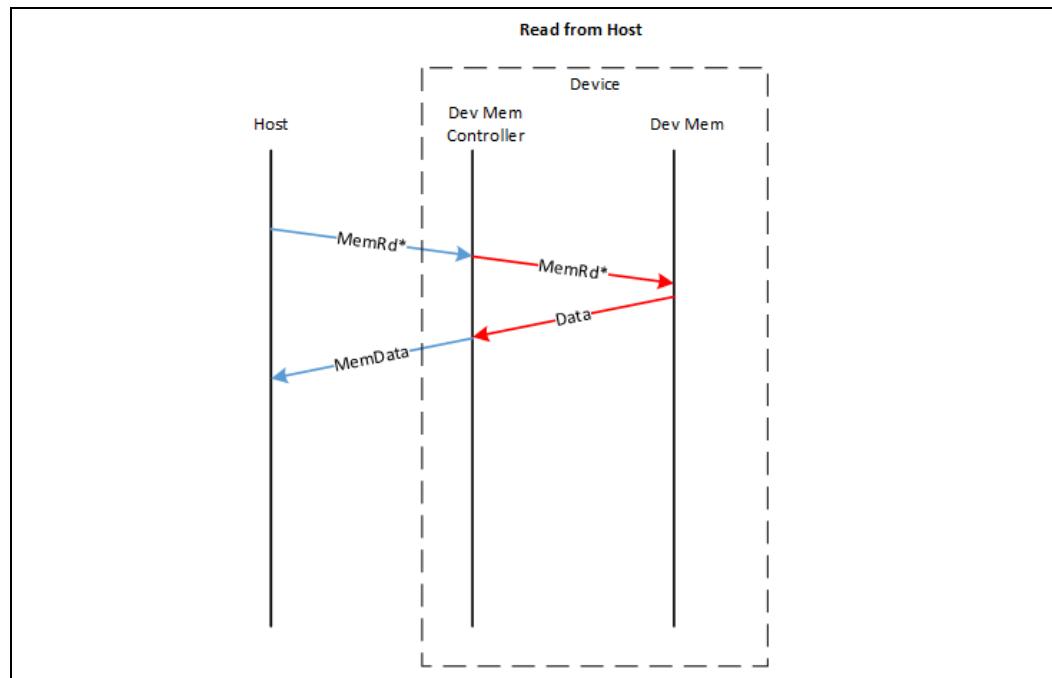
3.6

Flows to HDM-H in a Type 3 Device

The HDM-H address region in a Type 3 device is used as a memory expander or for Shared FAM device with software coherence where the device does not require active management of coherence with the Host. Thus, access to HDM-H does not use a DCOH agent. This allows the transaction flows to HDM-H to be simplified to just two classes, reads and writes, as shown below.

In [Figure 3-42](#), the optimized read flow is shown for the HDM-H address region. In this flow, only a Data message is returned. In contrast, in the HDM-D/HDM-DB address region, both NDR and Data are returned. The legend shown in [Figure 3-26](#) also applies to the transaction flows.

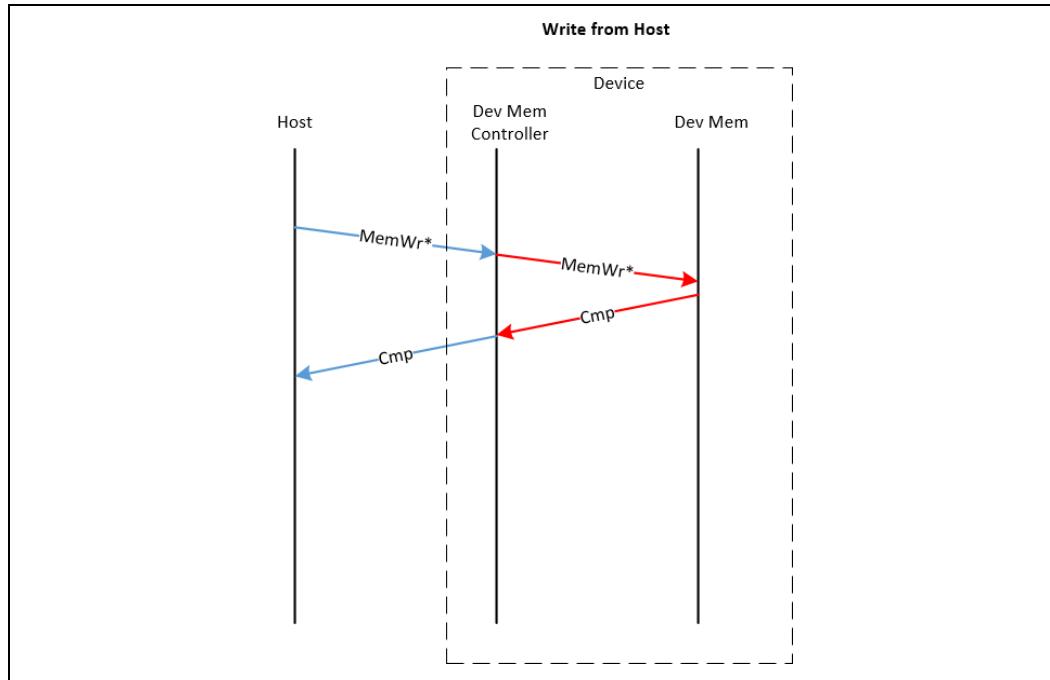
Figure 3-42. Read from Host to HDM-H



EVALUATION COPY

Unlike reads, writes to the HDM-H region use the same flow as the HDM-D/HDM-DB region and always complete with an S2M NDR Cmp message. This common write flow is shown in [Figure 3-43](#).

Figure 3-43. Write from Host to All HDM Regions



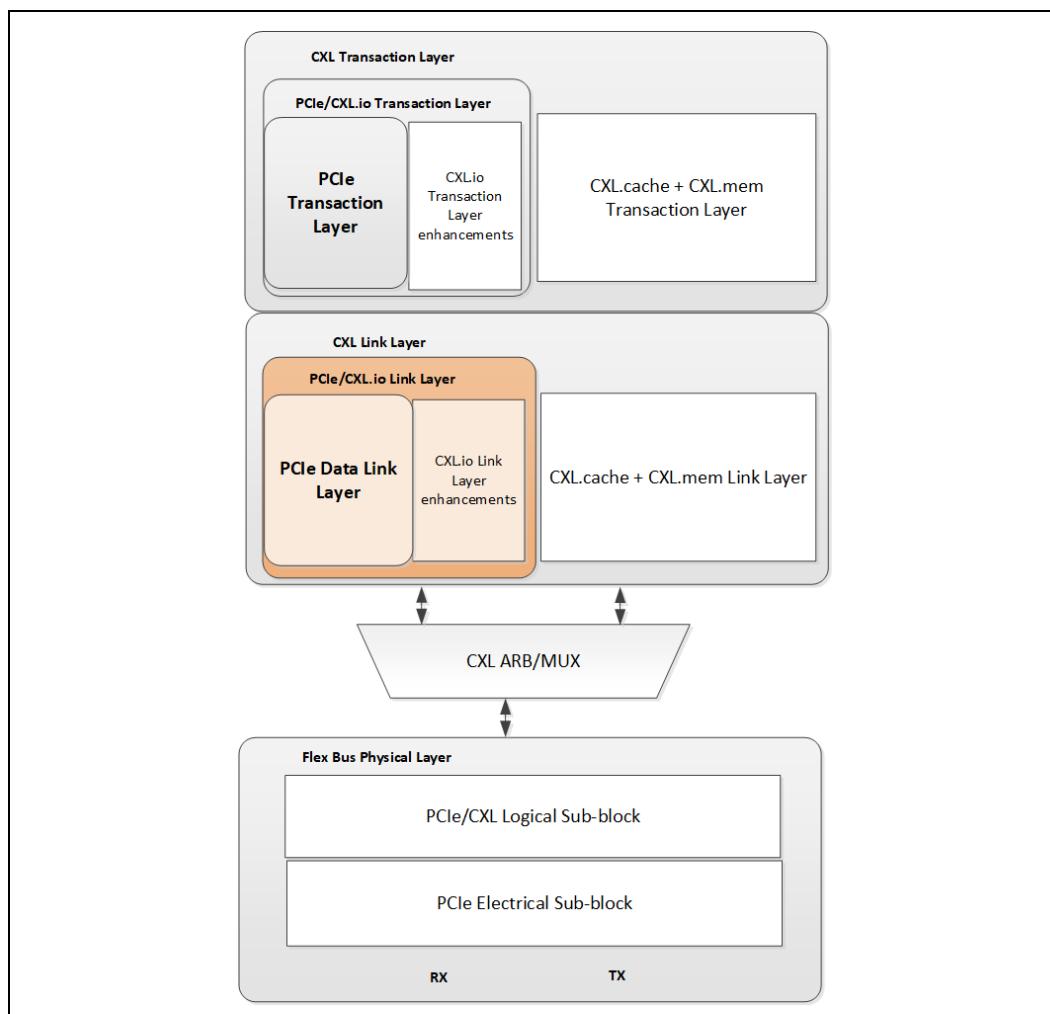
§ §

4.0 CXL Link Layers

4.1 CXL.io Link Layer

The CXL.io link layer acts as an intermediate stage between the CXL.io transaction layer and the Flex Bus Physical layer. Its primary responsibility is to provide a reliable mechanism for exchanging transaction layer packets (TLPs) between two components on the link. The PCIe* Data Link Layer is utilized as the link layer for CXL.io Link layer. Please refer to chapter titled “Data Link Layer Specification” in PCIe Base Specification for details. In 256B Flit mode, the PCIe-defined PM and Link Management DLLPs are not applicable for CXL.io and must not be used.

Figure 4-1. Flex Bus Layers - CXL.io Link Layer Highlighted



In addition, for 68B Flit mode, the CXL.io link layer implements the framing/deframing of CXL.io packets. CXL.io uses the Encoding for 8 GT/s, 16 GT/s, and 32 GT/s data rates only (see “128b/130b Encoding for 8.0 GT/s, 16.0 GT/s, and 32.0 GT/s Data Rates” in PCIe Base Specification for details).

This chapter highlights the notable framing and application of symbols to lanes that are specific for CXL.io. Note that when viewed on the link, the framing symbol-to-lane mapping will be shifted as a result of additional CXL framing (i.e., two bytes of Protocol ID and two reserved bytes) and of interleaving with other CXL protocols.

For CXL.io, only the x16 Link transmitter and receiver framing requirements described in PCIe Base Specification apply regardless of the negotiated link width. The framing related rules for $N = 1, 2, 4$, and 8 do not apply. For downgraded Link widths, where number of active lanes is less than x16, a single x16 data stream is formed using x16 framing rules and transferred over x16/(degraded link width) degraded link width streams.

The CXL.io link layer forwards a framed I/O packet to the Flex Bus Physical layer. The Flex Bus Physical layer framing rules are defined in [Chapter 6.0](#).

For 256B Flit mode, NOP-TLP alignment rules from PCIe Base Specification for PCIe Flit mode are shifted as a result of two bytes of Flit Type at the beginning of the flit.

The CXL.io link layer must guarantee that if a transmitted TLP ends exactly at the flit boundary, there must be a subsequent transmitted CXL.io flit. Please refer to [Section 6.2.2.7](#) for more details.

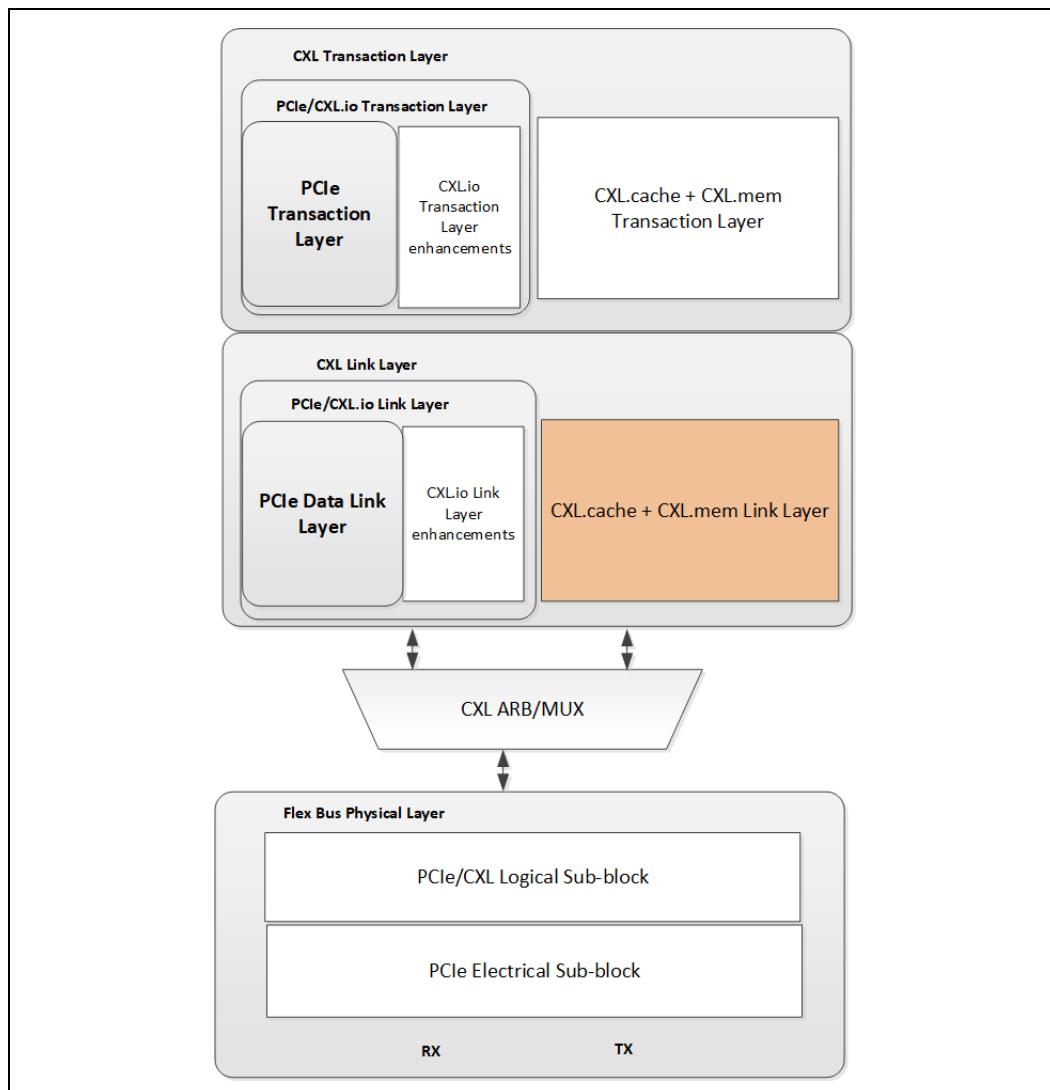
4.2 CXL.cache and CXL.mem 68B Flit Mode Common Link Layer

4.2.1 Introduction

[Figure 4-2](#) shows where the CXL.cache and CXL.mem link layer exists in the Flex Bus layered hierarchy. The link layer has two modes of operation: 68B flit and 256B flit. 68B flit, which defines 66B in the link layer and 2B in the ARB/MUX, supports the physical layer up to 32 GT/s. To support higher speeds a flit definition of 256B is defined; the reliability flows for that flit definition are handled in the Physical layer, so retry flows from 68B Flit mode are not applicable. 256B flits can support any legal transfer rate, but are required for >32 GT/s. The 256B flit definition and requirements are captured in [Section 4.3](#). There are Transaction Layer features that require 256B flits and those features include CacheID, Back-Invalidate Snoop (BISnp), and Port Based Routing (PBR).

EVALUATION COPY

Figure 4-2. Flex Bus Layers - CXL.cache + CXL.mem Link Layer Highlighted

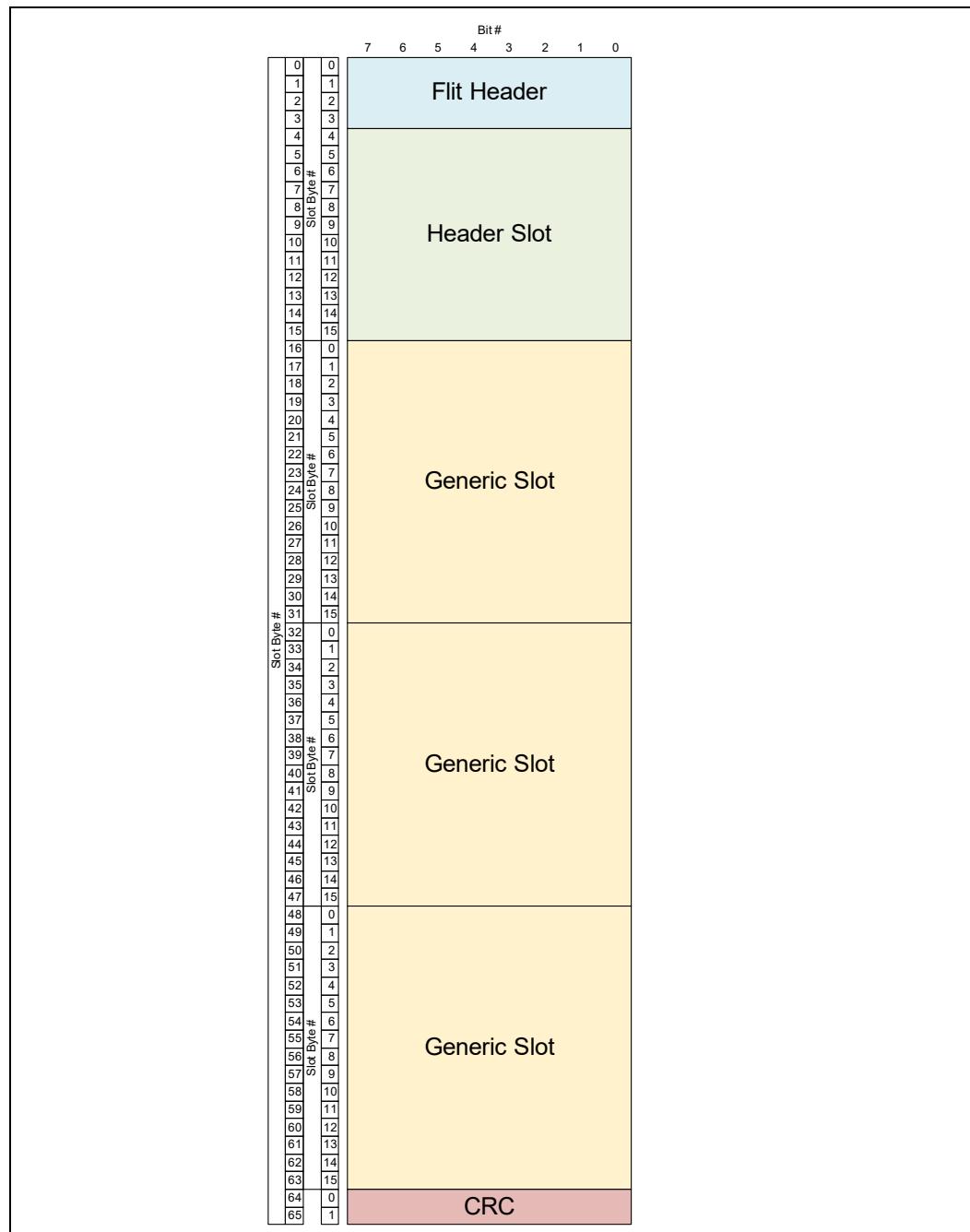


As previously mentioned, CXL.cache and CXL.mem protocols use a common Link Layer. This chapter defines the properties of this common Link Layer. Protocol information, including definition of fields, opcodes, transaction flows, etc., can be found in [Section 3.2](#) and [Section 3.3](#), respectively.

4.2.2 High-Level CXL.cachemem Flit Overview

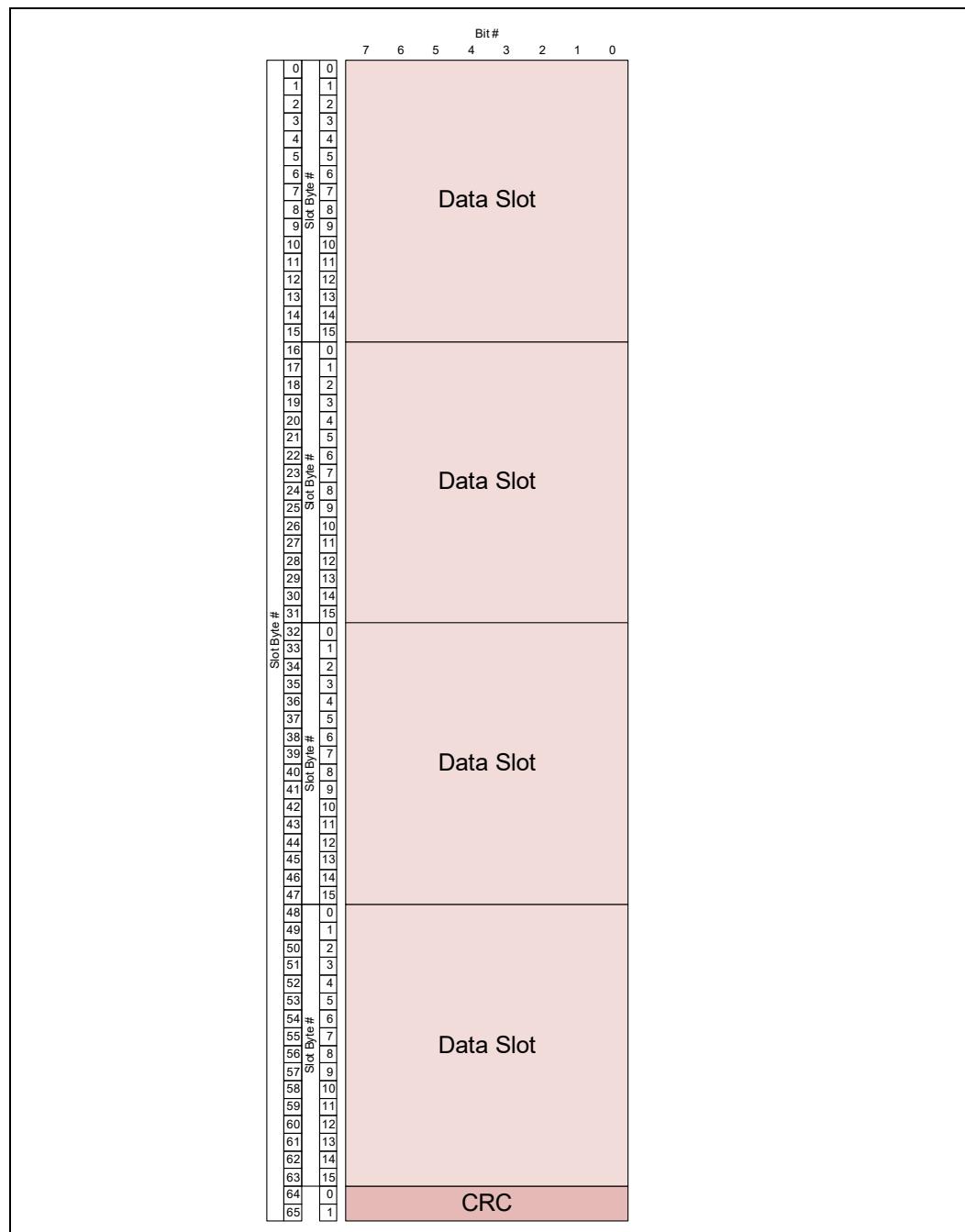
The CXL.cachemem flit size is a fixed 528b. There are 2B of CRC code and 4 slots of 16B each as shown below.

Figure 4-3. CXL.cachemem Protocol Flit Overview



EVALUATION COPY

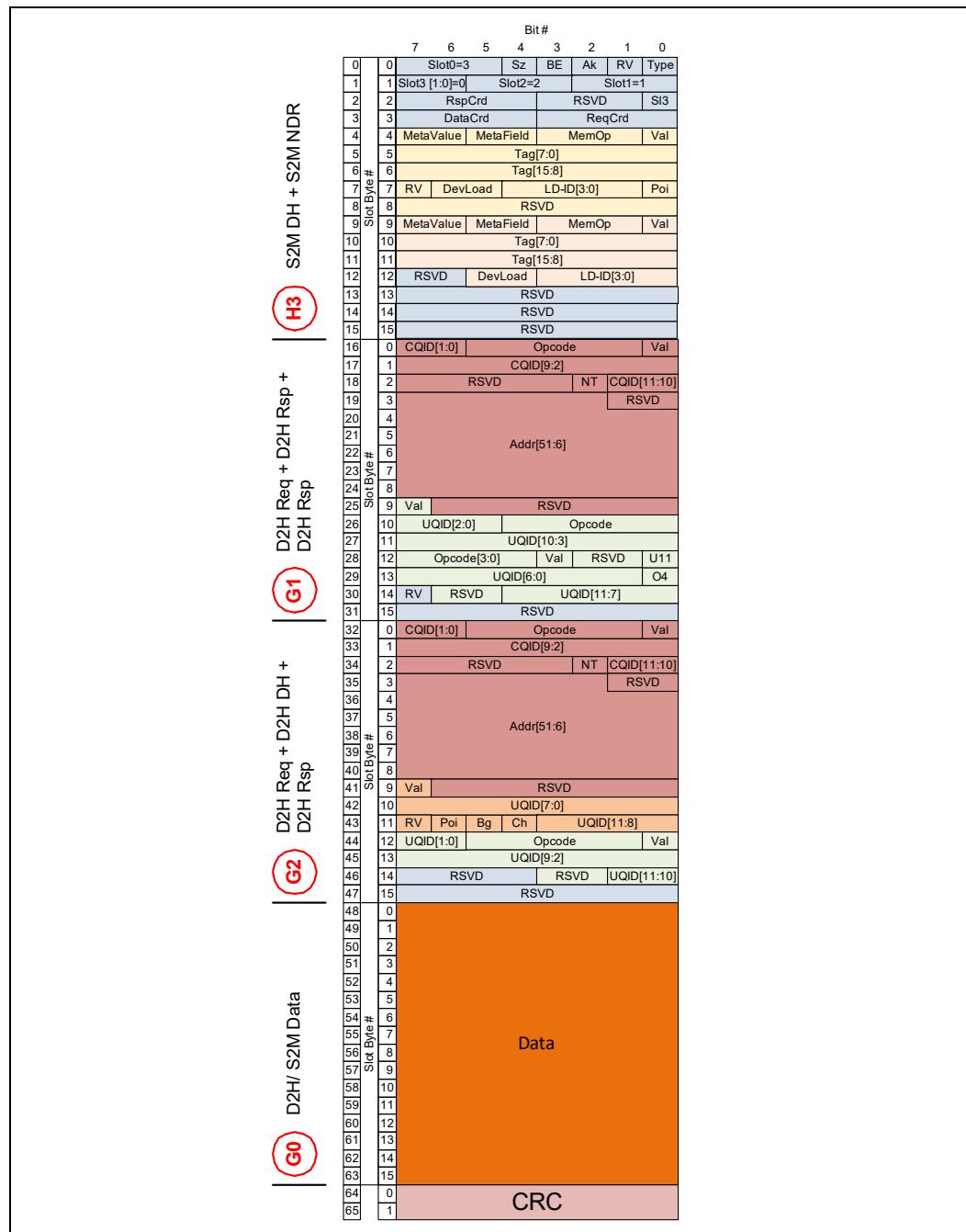
Figure 4-4. CXL.cachemem All Data Flit Overview



EVALUATION COPY

An example of a Protocol flit in the device to Host direction is shown below. For detailed descriptions of slot formats, see [Section 4.2.3](#).

Figure 4-5. Example of a Protocol Flit from Device to Host



A “Header” Slot is defined as one that carries a “Header” of link-layer specific information, including the definition of the protocol-level messages contained in the remainder of the header as well as in the other slots in the flit.

A “Generic” Slot can carry one or more request/response messages or a single 16B data chunk.

The flit can be composed of a Header Slot and 3 Generic Slots or four 16B Data Chunks.

The Link Layer flit header uses the same definition for both the Upstream Ports, as well as the Downstream Ports, as summarized in [Table 4-1](#).

Table 4-1.**CXL.cachemem Link Layer Flit Header Definition**

Field Name	Brief Description	Length in Bits
Type	This field distinguishes between a Protocol or a Control flit.	1
Ak	This is an acknowledgment of 8 successful flit transfers. Reserved for RETRY, and for INIT control flits.	1
BE	Byte Enable (Reserved for control flits).	1
Sz	Size (Reserved for control flits).	1
ReqCrd	Request Credit Return. Reserved for RETRY, and for INIT control flits.	4
DataCrd	Data Credit Return. Reserved for RETRY, and for INIT control flits.	4
RspCrd	Response Credit Return. Reserved for RETRY, and for INIT control flits.	4
Slot 0	Slot 0 Format Type (Reserved for control flits).	3
Slot 1	Slot 1 Format Type (Reserved for control flits).	3
Slot 2	Slot 2 Format Type (Reserved for control flits).	3
Slot 3	Slot 3 Format Type (Reserved for control flits).	3
RSVD	Reserved	4
Total		32

In general, bits or encodings that are not defined will be marked “Reserved” or “RSVD” in this specification. These bits should be cleared to 0 by the sender of the packet and the receiver should ignore them. Please also note that certain fields with static 0/1 values will be checked by the receiving Link Layer when decoding a packet. For example, Control flits have several static bits defined. A Control flit that passes the CRC check but fails the static bit check should be treated as a standard CRC error or as a fatal error when in “RETRY_LOCAL_NORMAL state of the LRSM. Logging and reporting of such errors is device specific. Checking of these bits reduces the probability of silent error under conditions where the CRC check fails to detect a long burst error. However, link layer must not cause fatal error whenever it is under shadow of CRC errors (i.e., its LRSM is not in RETRY_LOCAL_NORMAL state). This is prescribed because all-data-flit can alias to control messages after a CRC error and those alias cases may result in static bit check failure.

The following describes how the flit header information is encoded.

Table 4-2.**Type Encoding**

Value	Flit Type	Description
0	Protocol	This is a flit that carries CXL.cache or CXL.mem protocol-related information.
1	Control	This is a flit inserted by the link layer only for link layer-specific functionality. These flits are not exposed to the upper layers.

The Ak field is used as part of the link layer retry protocol to signal CRC-passing receipt of flits from the remote transmitter. The transmitter sets the Ak bit to acknowledge successful receipt of 8 flits; a cleared Ak bit is ignored by the receiver.

The BE (Byte Enable) and Sz (Size) fields have to do with the variable size of data messages. To reach its efficiency targets, the CXL.cachemem link layer assumes that generally all bytes are enabled for most data, and that data is transmitted at the full cacheline granularity. When all bytes are enabled, the link layer does not transmit the byte enable bits, but instead clears the Byte Enable field of the corresponding flit header. When the receiver decodes that the Byte Enable field is cleared, it must regenerate the byte enable bits as all 1s before passing the data message on to the transaction layer. If the Byte Enable bit is set, the link layer Rx expects an additional data chunk slot that contains byte enable information. Note that this will always be the last slot of data for the associated request.

Similarly, the Sz field reflects the fact that the CXL.cachemem protocol allows transmission of data at the half cacheline granularity. When the Size bit is set, the link layer Rx expects four slots of data chunks, corresponding to a full cacheline. When the Size bit is cleared, it expects only two slots of data chunks. In the latter case, each half cacheline transmission will be accompanied by its own data header. A critical assumption of packing the Size and Byte Enable information in the flit header is that the Tx flit packet may begin at most one data message per flit.

Note: Multi-Data-Headers are not allowed to be sent when Sz=0 or BE=1 as described in the flit packing rules in [Section 4.2.5](#).

Table 4-3 describes legal values of Sz and BE for various data transfers. For cases where a 32B split transfer is sent that includes Byte Enables, the trailing Byte Enables apply only to the 32B sent. The Byte Enable bits that are applicable to that transfer are aligned based on which half of the cacheline is applicable to the transfer (BE[63:32] for Upper half of the cacheline or BE[31:0] for the lower half of the cacheline). This means that each of the split 32B transfers that are used to form a cacheline of data will include Byte Enables if Byte Enables are needed. Illegal use will cause an uncorrectable error. The reserved bits included in the BE slot may not be preserved when passing through a switch.

Table 4-3. [Legal Values of Sz and BE Fields](#)

Type of Data Transfer	32B Transfer Permitted in 68B Flit? ¹	BE Permitted?
CXL.cache H2D Data	Yes	No
CXL.mem M2S Data	No	Yes
CXL.cache D2H Data	Yes	Yes
CXL.mem S2M Data	Yes	No

1. The 32B transfer allowance is only defined for 68B flit definition and does not apply for 256B flit.

The transmitter sets the CRD fields to indicate freed resources that are available in the co-located receiver for use by the remote transmitter. Credits are given for transmission per message class, which is why the flit header contains independent Request, Response, and Data CRD fields. Note that there are no Requests sourced in the S2M direction, and that there are no Responses sourced in the M2S direction. Details of the channel mapping are captured in [Table 4-5](#). Credits returned for channels not supported by the device or the host should be silently discarded. The granularity of credits is per message. These fields are encoded exponentially, as delineated in [Table 4-4](#).

Note: Messages sent on Data channels require a single data credit for the entire message. This means that 1 credit allows for one data transfer, including the header of the message, regardless of whether the transfer is 64B, or 32B, or contains Byte Enables.

The transaction layer requires all messages that carry payload to send 64B and the link layer allows for those to be sent as independent 32B messages to optimize latency for implementation-specific cases in which only 32B of data is ready to send.

Table 4-4. CXL.cachemem Credit Return Encodings

Credit Return Encoding[3]	Protocol	
0	CXL.cache	
1	CXL.mem	
Credit Return Encoding[2:0]		Number of Credits
000b	0	
001b	1	
010b	2	
011b	4	
100b	8	
101b	16	
110b	32	
111b	64	

Table 4-5. ReqCrd/DataCrd/RspCrd Channel Mapping

Credit Field	Credit Bit 3 Encoding	Link Direction	Channel
ReqCrd	0 - CXL.cache	Upstream	D2H Request
		Downstream	H2D Request
	1 - CXL.mem	Upstream	Reserved
		Downstream	M2S Request
DataCrd	0 - CXL.cache	Upstream	D2H Data
		Downstream	H2D Data
	1 - CXL.mem	Upstream	S2M DRS
		Downstream	M2S Rwd
RspCrd	0 - CXL.cache	Upstream	D2H Rsp
		Downstream	H2D Rsp
	1 - CXL.mem	Upstream	S2M NDR
		Downstream	Reserved

Finally, the Slot Format Type fields encode the Slot Format of both the header slot and of the other generic slots in the flit (if the Flit Type bit specifies that the flit is a Protocol flit). The subsequent sections detail the protocol message contents of each slot format, but [Table 4-6](#) provides a quick reference for the Slot Format field encoding.

Note:

Format H6 is defined for use with Integrity and Data Encryption. See details of requirements for its use in [Chapter 11.0](#).

Table 4-6. Slot Format Field Encoding

Slot Format Encoding	H2D/M2S		D2H/S2M	
	Slot 0	Slots 1, 2, and 3	Slot 0	Slots 1, 2, and 3
000b	H0	G0	H0	G0
001b	H1	G1	H1	G1
010b	H2	G2	H2	G2
011b	H3	G3	H3	G3
100b	H4	G4	H4	G4
101b	H5	G5	H5	G5
110b	H6	RSVD	H6	G6
111b	RSVD	RSVD	RSVD	RSVD

[Table 4-7](#) and [Table 4-8](#) describe the slot format and the type of message contained by each format for both directions.

Table 4-7. H2D/M2S Slot Formats

Format to Req Type Mapping	H2D/M2S	
	Type	Length in Bits
H0	CXL.cache Req + CXL.cache Rsp	96
H1	CXL.cache Data Header + 2 CXL.cache Rsp	88
H2	CXL.cache Req + CXL.cache Data Header	88
H3	4 CXL.cache Data Header	96
H4	CXL.mem RwD Header	87
H5	CXL.mem Req Only	87
H6	MAC slot used for link integrity	96
G0	CXL.cache/ CXL.mem Data Chunk	128
G1	4 CXL.cache Rsp	128
G2	CXL.cache Req + CXL.cache Data Header + CXL.cache Rsp	120
G3	4 CXL.cache Data Header + CXL.cache Rsp	128
G4	CXL.mem Req + CXL.cache Data Header	111
G5	CXL.mem RwD Header + CXL.cache Rsp	119

Table 4-8. D2H/S2M Slot Formats

Format to Req Type Mapping	D2H/S2M	
	Type	Length in Bits
H0	CXL.cache Data Header + 2 CXL.cache Rsp + CXL.mem NDR	87
H1	CXL.cache Req + CXL.cache Data Header	96
H2	4 CXL.cache Data Header + CXL.cache Rsp	88
H3	CXL.mem DRS Header + CXL.mem NDR	70
H4	2 CXL.mem NDR	60
H5	2 CXL.mem DRS Header	80
H6	MAC slot used for link integrity	96
G0	CXL.cache/ CXL.mem Data Chunk	128
G1	CXL.cache Req + 2 CXL.cache Rsp	119
G2	CXL.cache Req + CXL.cache Data Header + CXL.cache Rsp	116
G3	4 CXL.cache Data Header	68
G4	CXL.mem DRS Header + 2 CXL.mem NDR	100
G5	2 CXL.mem NDR	60
G6	3 CXL.mem DRS Header	120

4.2.3 Slot Format Definition

Slot diagrams in this section include abbreviations for bit field names to allow them to fit into the diagram. In the diagrams, most abbreviations are obvious, but the following abbreviation list ensures clarity:

- Bg = Bogus
- Ch = ChunkValid
- LA0 = LowerAddr[0]
- LA1 = LowerAddr[1]
- LI3 = LD-ID[3]
- MV0 = MetaValue[0]
- MV1 = MetaValue[1]
- O4 = Opcode[4]
- Op0 = Opcode[0]
- Poi = Poison
- R11 = RspData[11]
- RSVD = Reserved
- RV = Reserved
- SL3 = Slot3[2]
- Tag15 = Tag[15]
- U11 = UQID[11]
- Val = Valid

4.2.3.1 H2D and M2S Formats

Figure 4-6. H0 - H2D Req + H2D Rsp

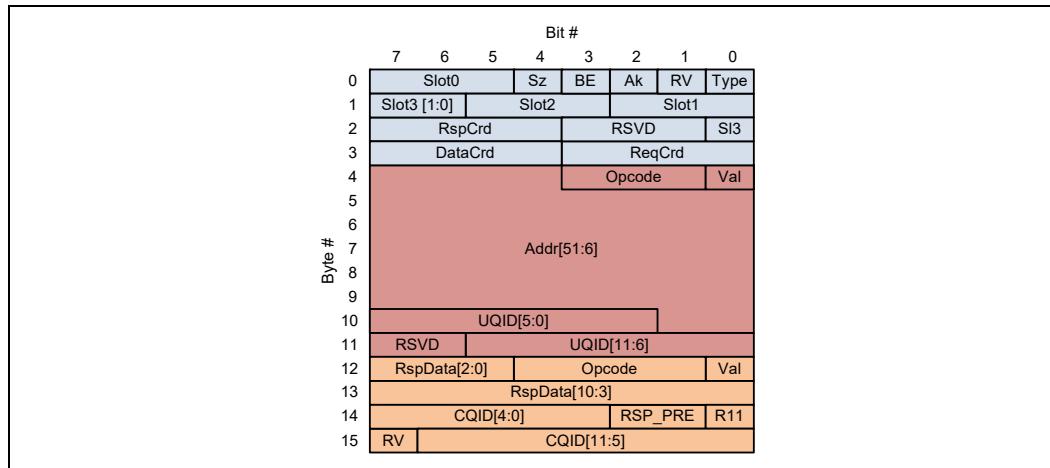
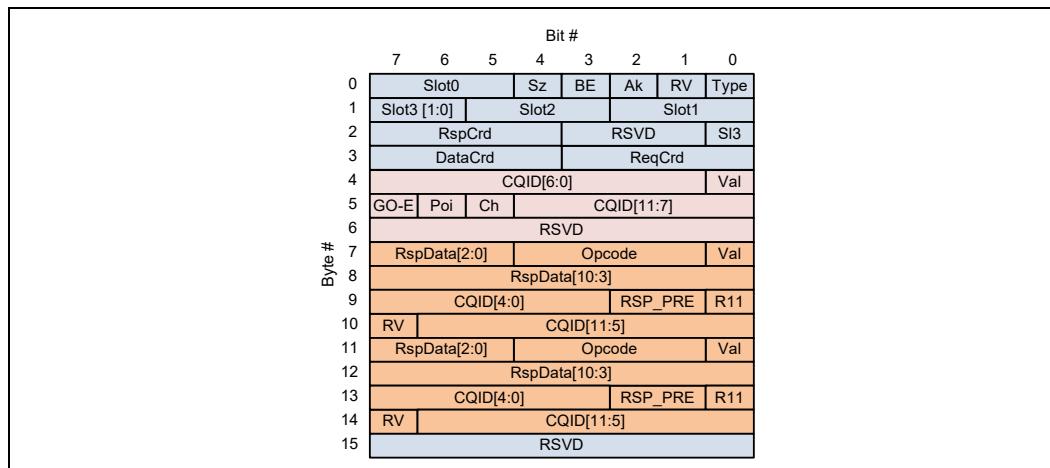


Figure 4-7. H1 - H2D Data Header + H2D Rsp + H2D Rsp



EVALUATION COPY

Figure 4-8. H2 - H2D Req + H2D Data Header

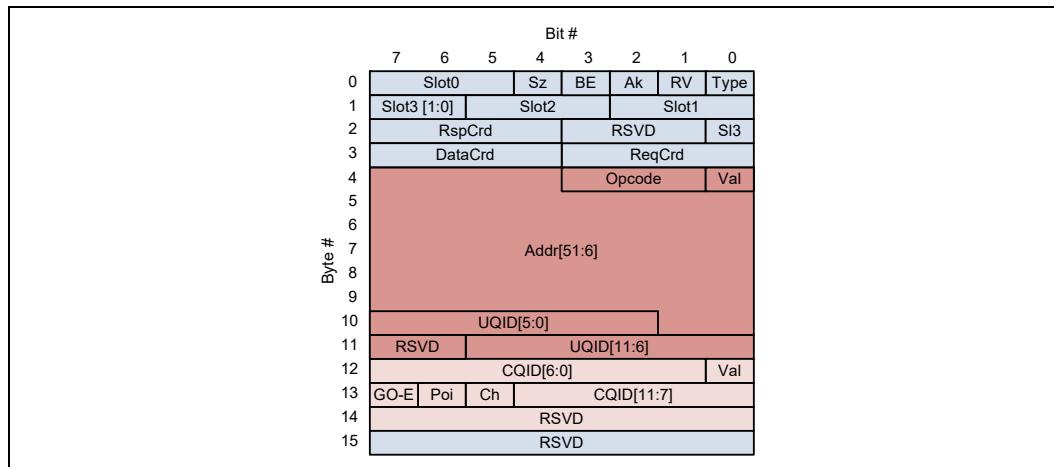


Figure 4-9. H3 - 4 H2D Data Header

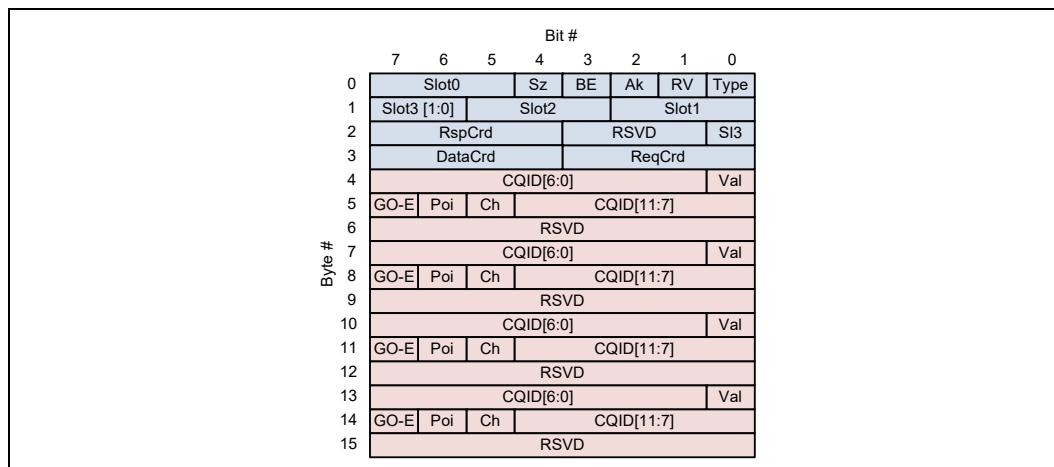
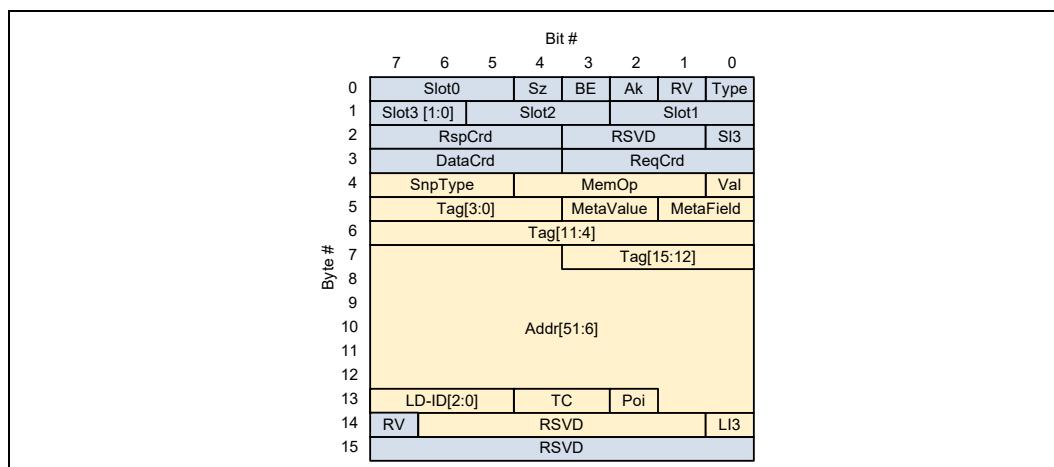


Figure 4-10. H4 - M2S Rwd Header



EVALUATION COPY

Figure 4-11. H5 - M2S Req

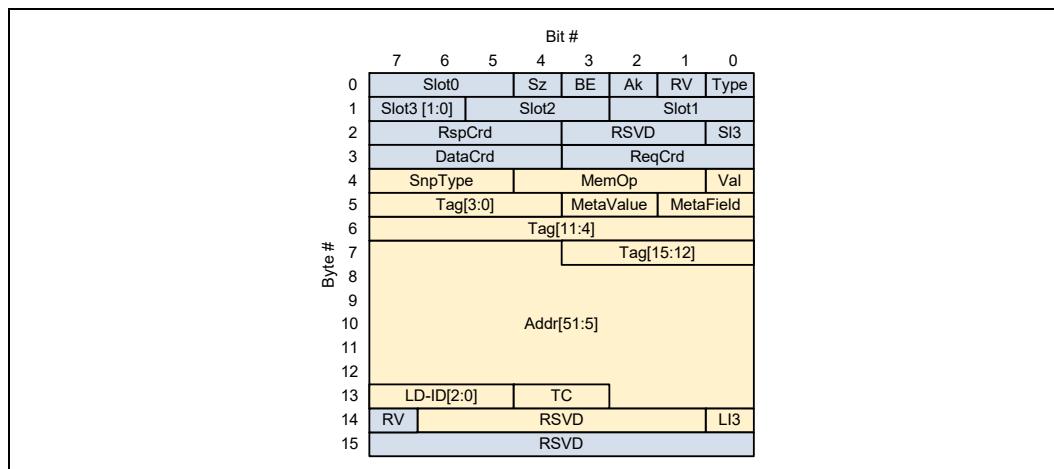


Figure 4-12. H6 - MAC

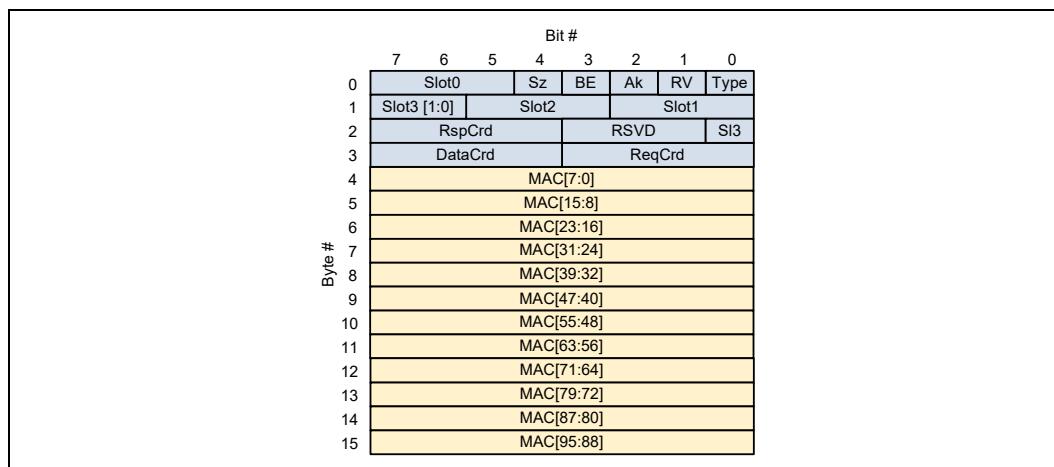
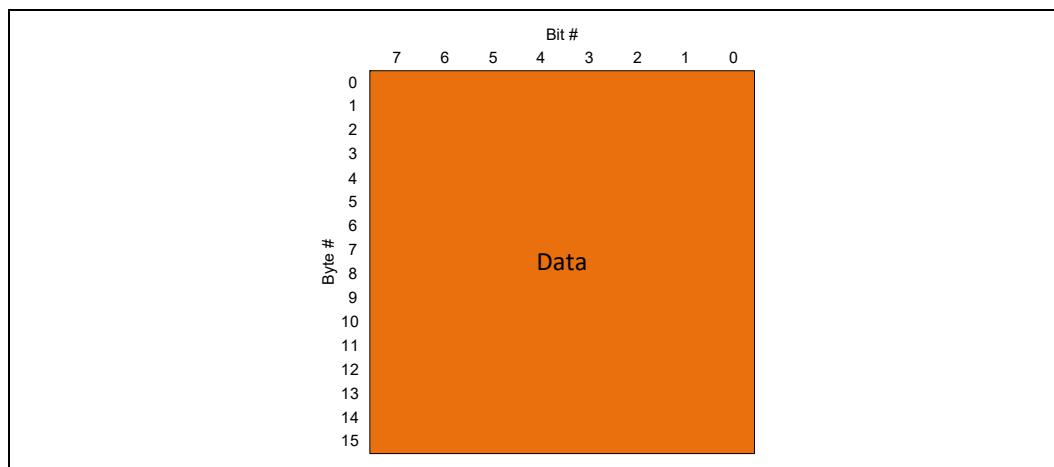


Figure 4-13. G0 - H2D/M2S Data



EVALUATION COPY

Figure 4-14. G0 - M2S Byte Enable

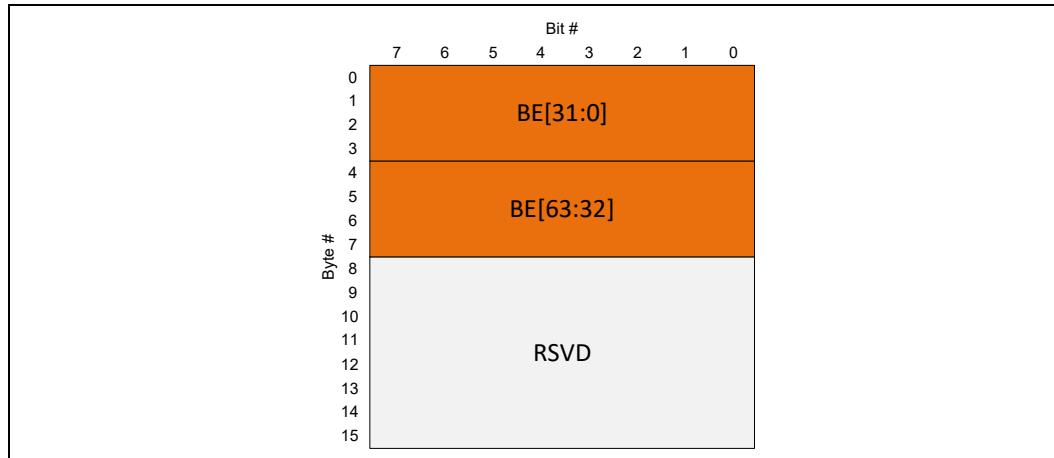


Figure 4-15. G1 - 4 H2D Rsp

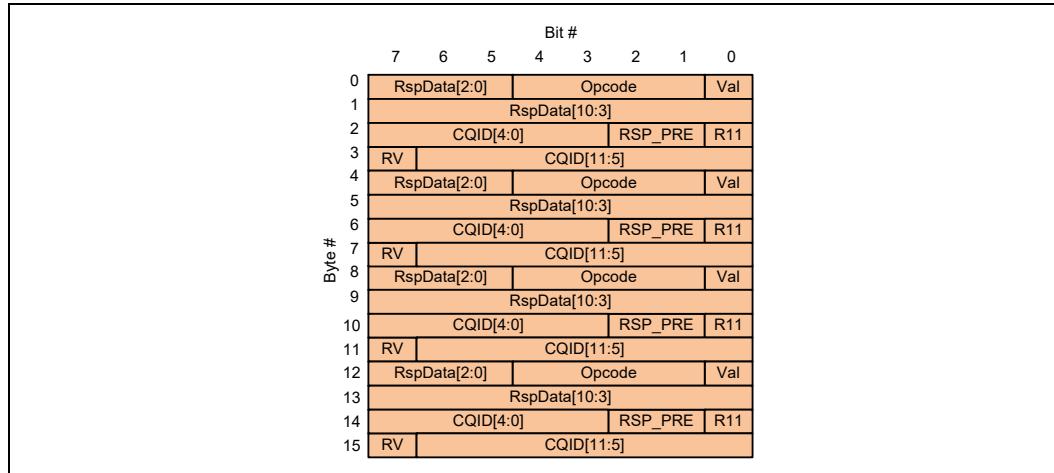
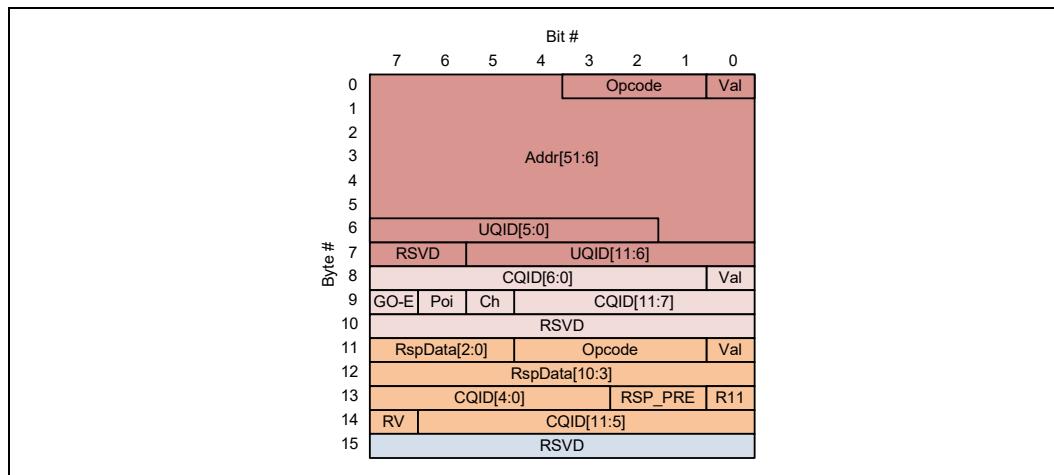


Figure 4-16. G2 - H2D Req + H2D Data Header + H2D Rsp



EVALUATION COPY

Figure 4-17. G3 - 4 H2D Data Header + H2D Rsp

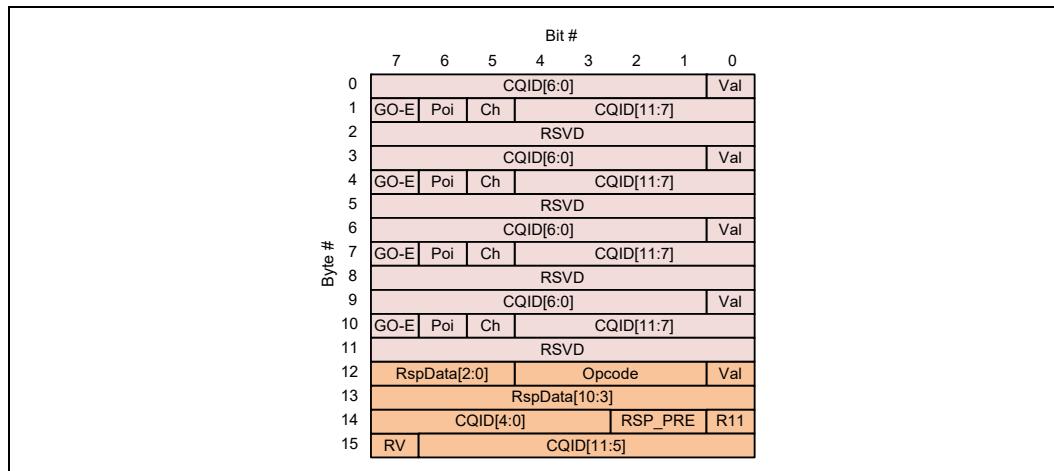


Figure 4-18. G4 - M2S Req + H2D Data Header

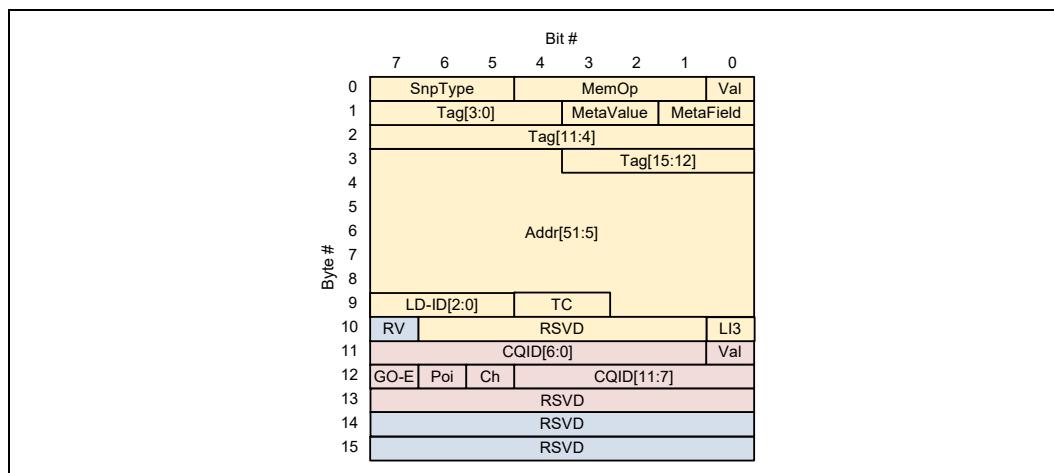
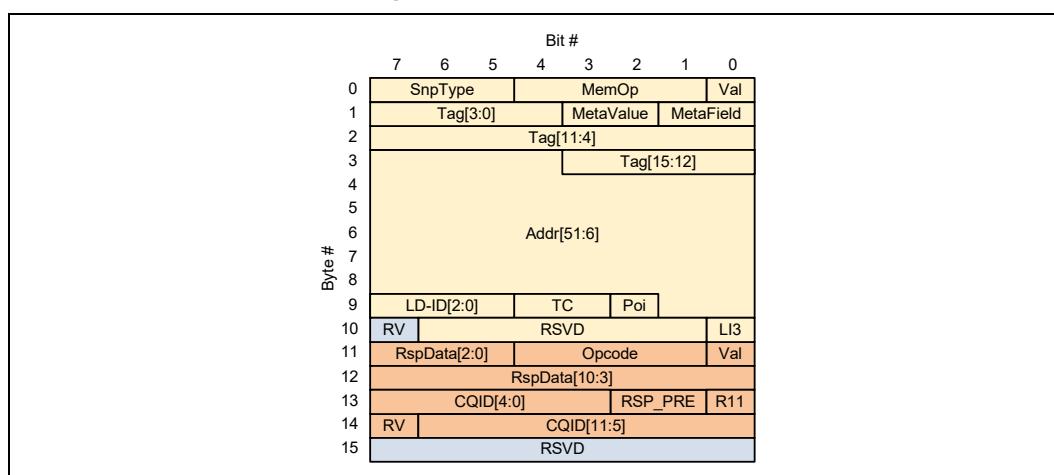


Figure 4-19. G5 - M2S Rwd Header + H2D Rsp



4.2.3.2 D2H and S2M Formats

The original slot definitions ensured that all header bits for a message are in contiguous bits. The S2M NDR message expanded by two bits to fit the 2-bit DevLoad field. Some slot formats that carry NDR messages include non-contiguous bits within the slot to account for the DevLoad. The formats impacted are H4, G4, and G5 and the non-contiguous bits are denoted as "DevLoad*" ("*" is the special indicator with separate color/pattern for the NDR message with non-contiguous bits). By expanding the slots in this way, backward compatibility with the original contiguous bit definition is maintained by ensuring that only RSVD slot bits are used to expand the headers. Other slot formats that carry a single NDR message can be expanded and keep the contiguous header bits because the NDR message is the last message in the slot formats (see Formats H0 and H3).

Figure 4-20. H0 - D2H Data Header + 2 D2H Rsp + S2M NDR

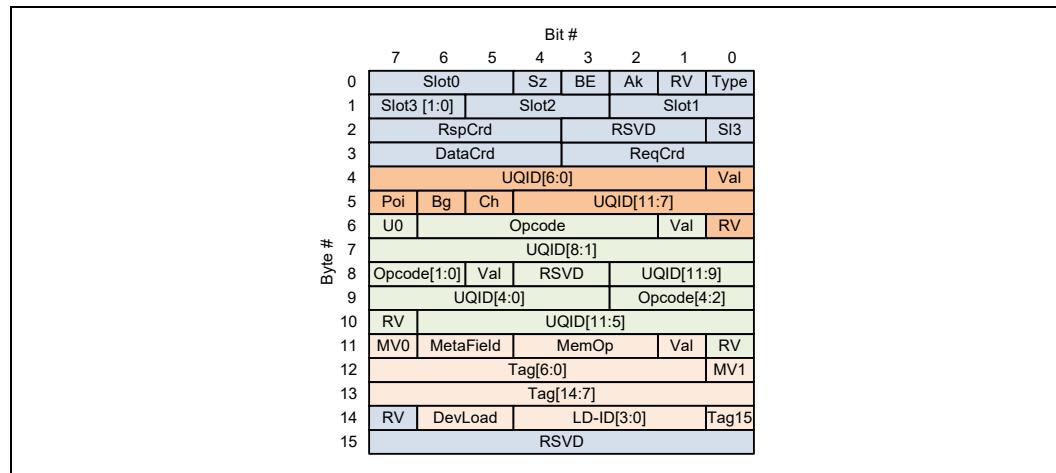
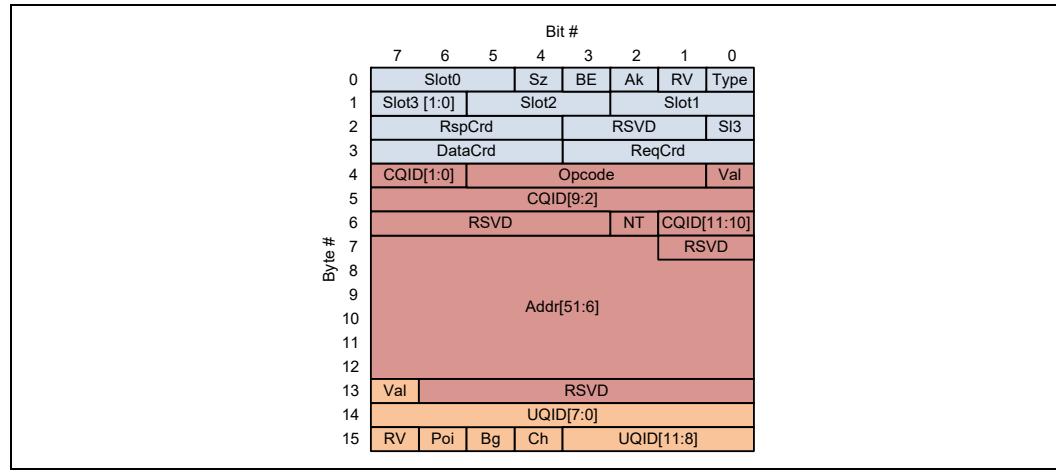


Figure 4-21. H1 - D2H Req + D2H Data Header



EVALUATION COPY

Figure 4-22. H2 - 4 D2H Data Header + D2H Rsp

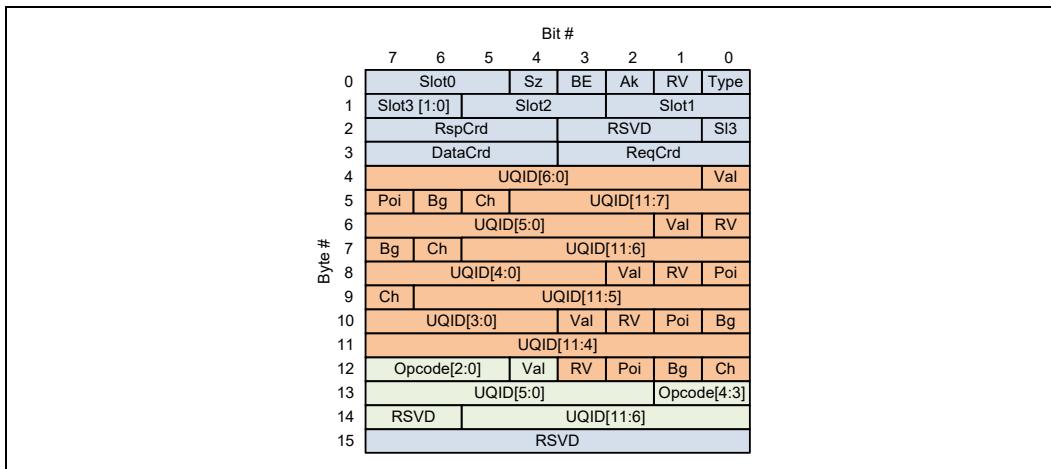


Figure 4-23. H3 - S2M DRS Header + S2M NDR

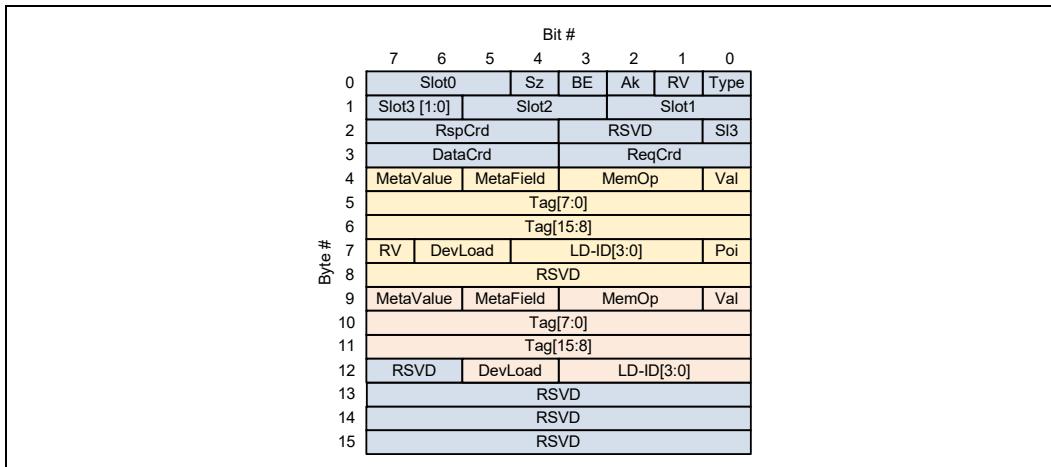
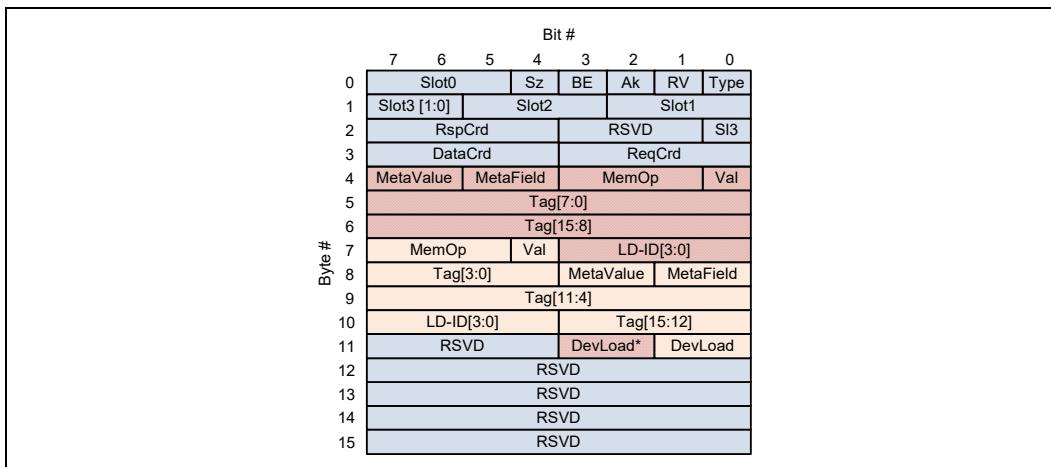


Figure 4-24. H4 - 2 S2M NDR



EVALUATION COPY

Figure 4-25. H5 - 2 S2M DRS Header

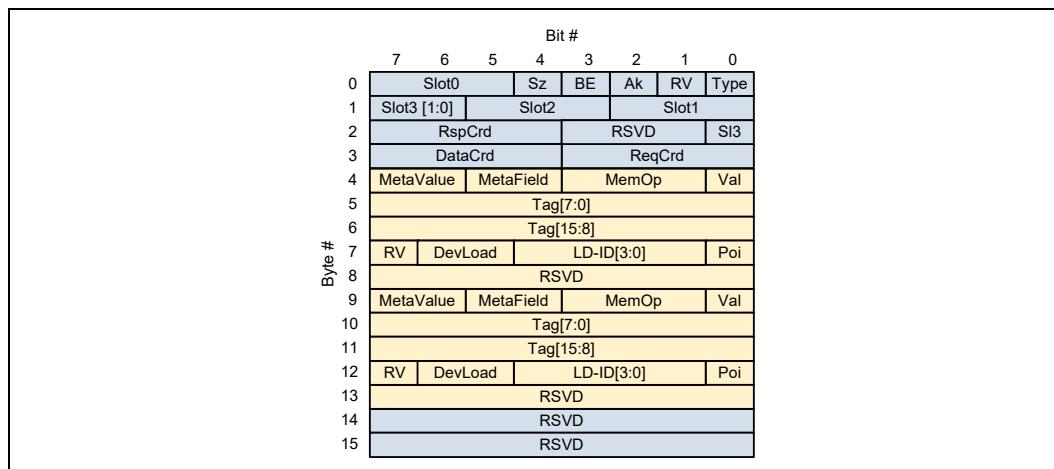


Figure 4-26. H6 - MAC

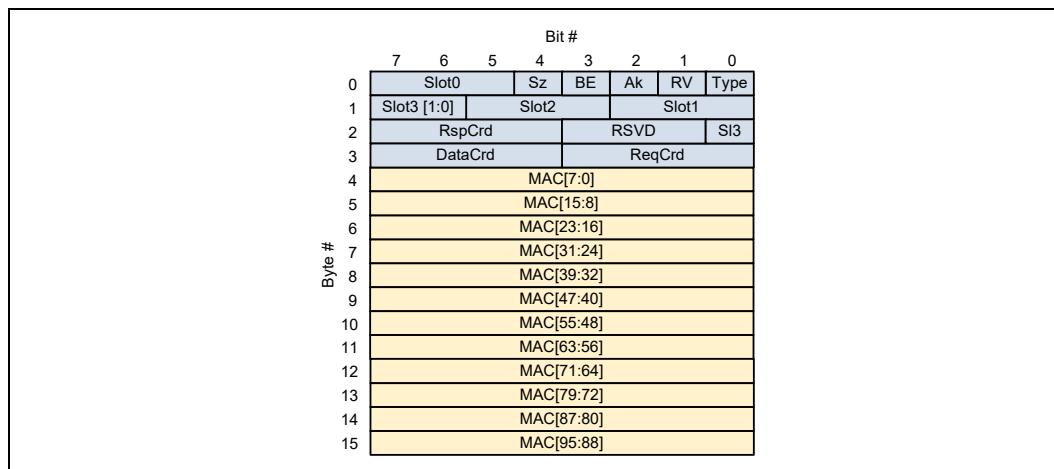
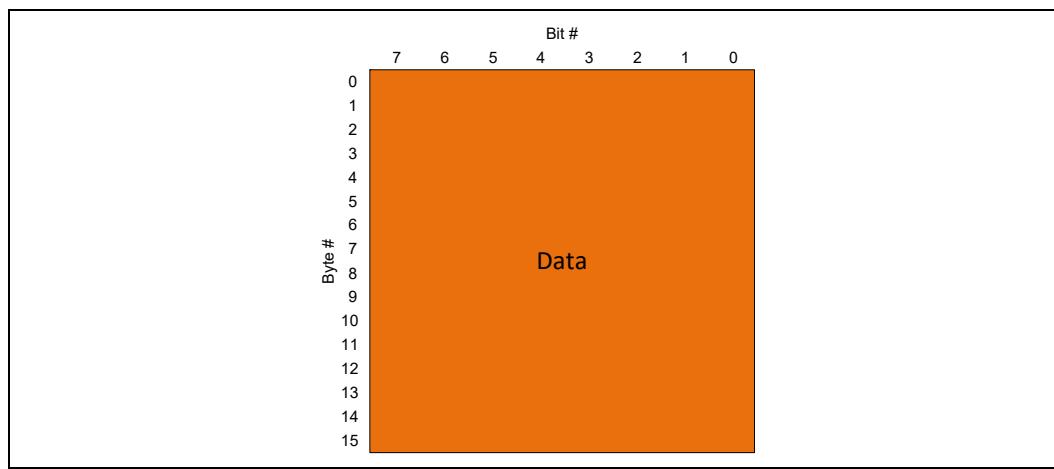


Figure 4-27. G0 - D2H/S2M Data



EVALUATION COPY

Figure 4-28. G0 - D2H Byte Enable

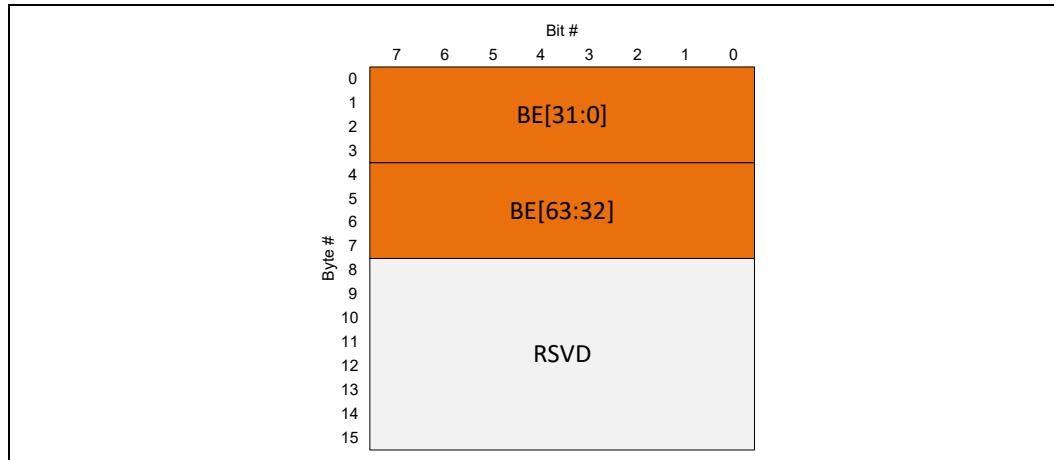


Figure 4-29. G1 - D2H Req + 2 D2H Rsp

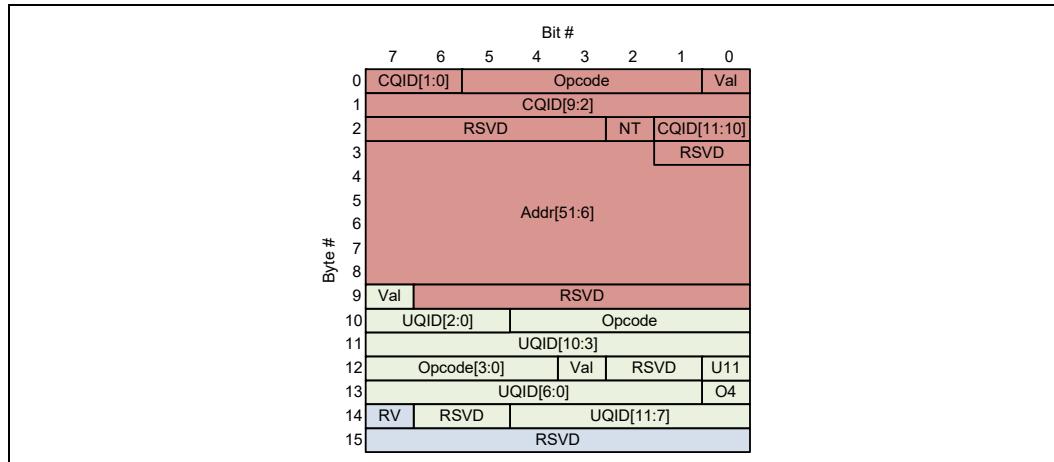
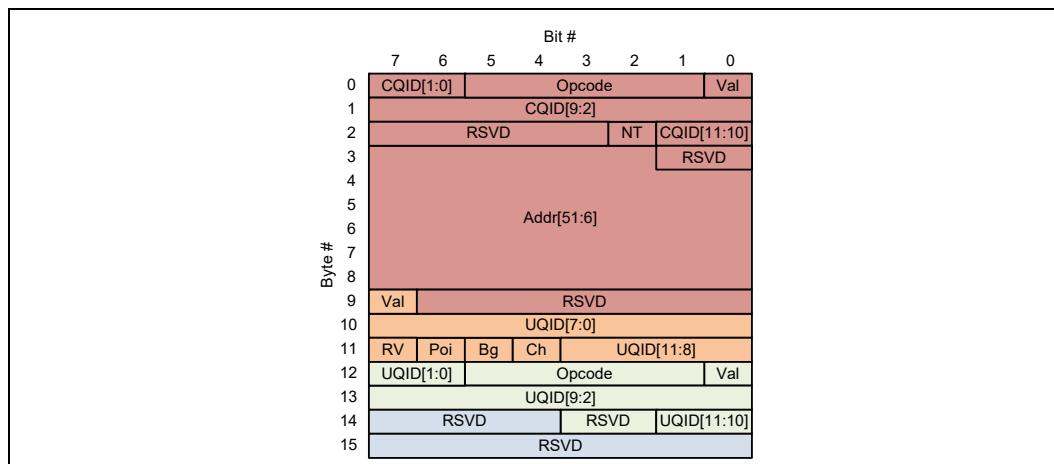


Figure 4-30. G2 - D2H Req + D2H Data Header + D2H Rsp



EVALUATION COPY

Figure 4-31. G3 - 4 D2H Data Header

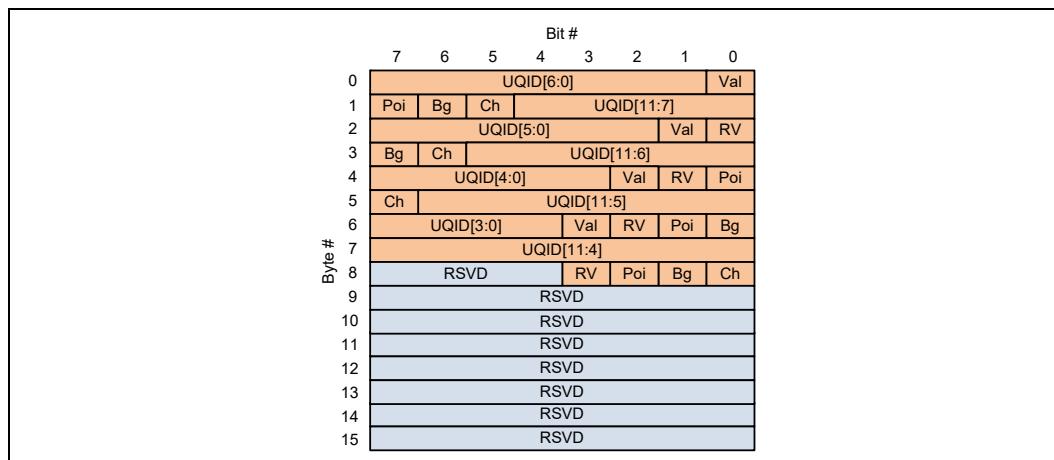


Figure 4-32. G4 - S2M DRS Header + 2 S2M NDR

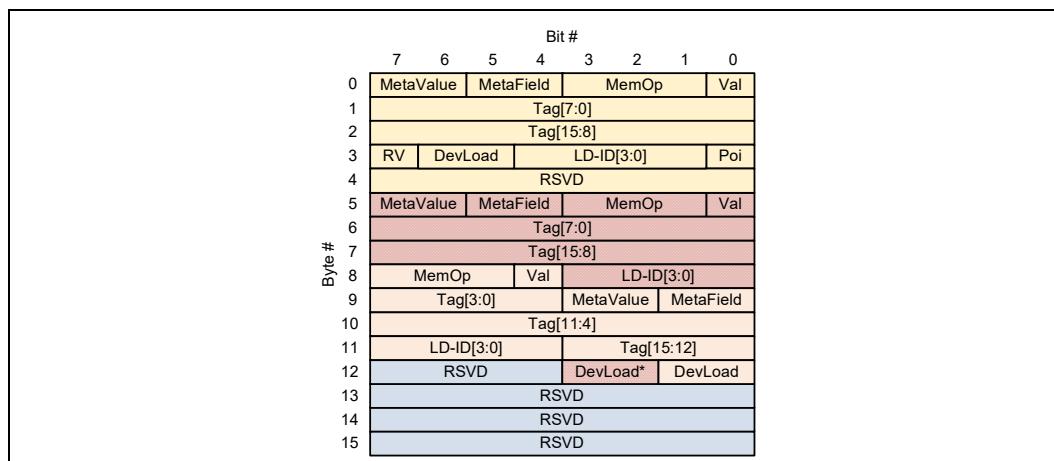


Figure 4-33. G5 - 2 S2M NDR

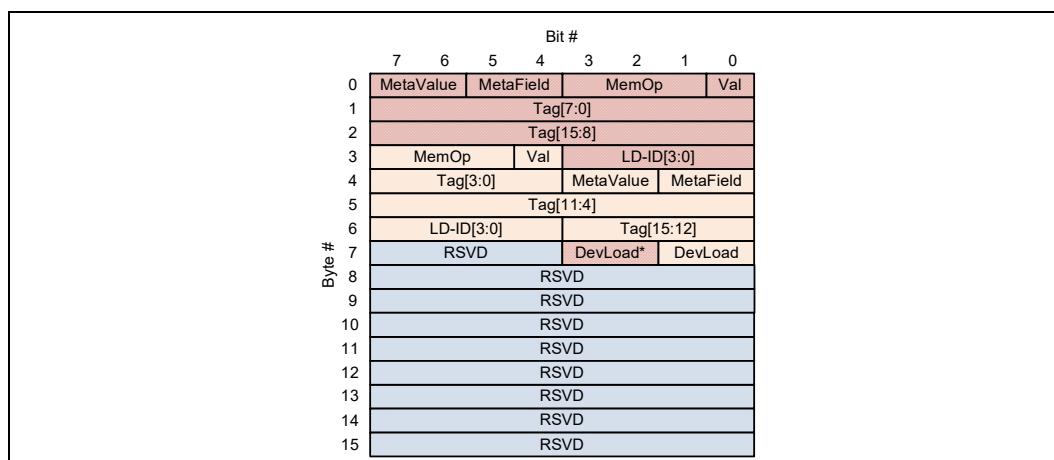
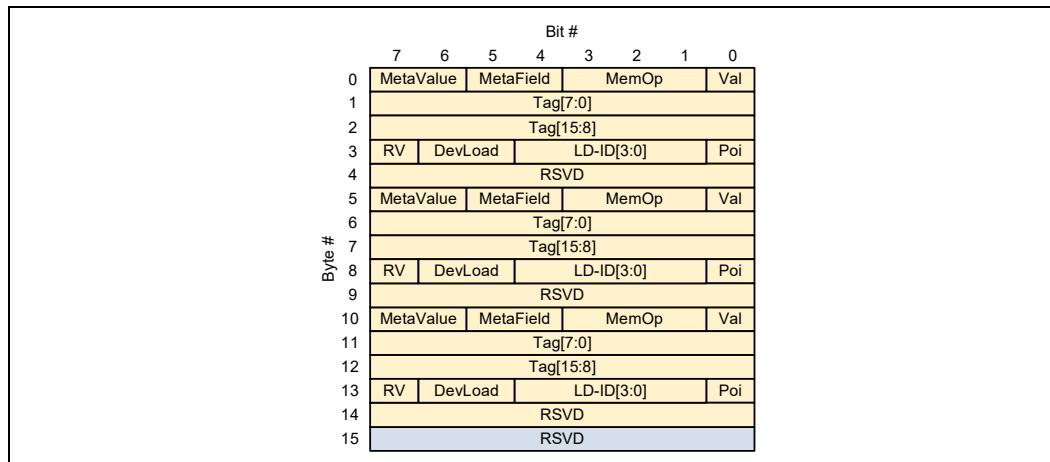


Figure 4-34. G6 - 3 S2M DRS Header

4.2.4 Link Layer Registers

Architectural registers associated with CXL.cache and CXL.mem are defined in [Section 8.2.4.19](#).

4.2.5 68B Flit Packing Rules

The packing rules are defined below. It is assumed that a given queue has credits toward the Rx and any protocol dependencies (SNP-GO ordering, for example) have already been considered:

- Rollover is defined as any time a data transfer needs more than one flit. Note that a data chunk that contains 128b (Format G0), can only be scheduled in Slot 1, Slot 2, and Slot 3 of a protocol flit since Slot 0 has only 96b available, as 32b are taken up by the flit header. The following rules apply to Rollover data chunks:
 - If there's a rollover of more than 3 16B data chunks, the next flit must necessarily be an all-data flit.
 - If there's a rollover of 3 16B data chunks, Slot 1, Slot 2, and Slot 3 must necessarily contain the 3 rollover data chunks. Slot 0 will be packed independently (it is allowed for Slot 0 to have the Data Header for the next data transfer).
 - If there's a rollover of 2 16B data chunks, Slot 1 and Slot 2 must necessarily contain the 2 rollover data chunks. Slot 0 and Slot 3 will be packed independently.
 - If there's a rollover of 1 16B data chunk, Slot 1 must necessarily contain the rollover data chunk. Slot 0, Slot 2, and Slot 3 will be packed independently.
 - If there's no rollover, each of the 4 slots will be packed independently.
- Care must be taken to ensure fairness between packing of CXL.cache and CXL.mem transactions. Similarly, care must be taken to ensure fairness between channels within a given protocol. The exact mechanism to ensure fairness is implementation specific.
- Valid messages within a given slot must be tightly packed. Which means, if a slot contains multiple possible locations for a given message, the Tx must pack the message in the first available location before advancing to the next available location.

EVALUATION COPY

- Valid messages within a given flit must be tightly packed. Which means, if a flit contains multiple possible slots for a given message, the Tx must pack the message in the first available slot before advancing to the next available slot.
- Empty slots are defined as slots without any valid bits set and they may be mixed with other slots in any order as long as all other packing rules are followed. For an example refer to [Figure 4-5](#) where slot H3 could have no valid bits set indicating an empty slot, but the 1st and 2nd generic slots, G1 and G2 in the example, may have mixed valid bits set.
- If a valid Data Header is packed in a given slot, the next available slot for data transfer (Slot 1, Slot 2, Slot 3 or an all-data flit) will be guaranteed to have data associated with the header. The Rx will use this property to maintain a shadow copy of the Tx Rollover counts. This enables the Rx to expect all-data flits where a flit header is not present.
- For data transfers, the Tx must send 16B data chunks in cacheline order. That is, chunk order 01 for 32B transfers and chunk order 0123 for 64B transfers.
- A slot with more than one data header (e.g., H5 in the S2M direction, or G3 in the H2D direction) is called a multi-data header slot or an MDH slot. MDH slots can only be sent for full cacheline transfers when both 32B chunks are immediately available to pack (i.e., BE = 0, Sz = 1). An MDH slot can only be used if both agents support MDH (defeature is defined in [Section 8.2.4.19.7](#)). If MDH is received when it is disabled it is considered a fatal error.
- An MDH slot format may be selected by the Tx only if there is more than 1 valid Data Header to pack in that slot.
- Control flits cannot be interleaved with all-data flits. This also implies that when an all-data flit is expected following a protocol flit (due to Rollover), the Tx cannot send a Control flit before the all-data flit.
- For non-MDH containing flits, there can be at most 1 valid Data Header in that flit. Also, an MDH containing flit cannot be packed with another valid Data Header in the same flit.
- The maximum number of messages that can be sent in a given flit is restricted to reduce complexity in the receiver, which writes these messages into credited queues. By restricting the number of messages across the entire flit, the number of write ports into the receiver's queues are constrained. The maximum number of messages per type within a flit (sum, across all slots) is:

D2H Request --> 4
D2H Response --> 2
D2H Data Header --> 4
D2H Data --> 4*16B
S2M NDR --> 2
S2M DRS Header --> 3
S2M DRS Data --> 4*16B

H2D Request --> 2
H2D Response --> 4
H2D Data Header --> 4
H2D Data --> 4*16B
M2S Req --> 2
M2S Rwd Header --> 1
M2S Rwd Data --> 4*16B

- 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], the least 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.

4.2.6

Link Layer Control Flit

Link Layer Control flits do not follow flow control rules applicable to protocol flits. That is, they can be sent from an entity without any credits. These flits must be processed and consumed by the receiver within the period to transmit a flit on the channel since there are no storage or flow control mechanisms for these flits. [Table 4-9](#) lists all the Controls flits supported by the CXL.cachemem link layer.

Table 4-9. [CXL.cachemem Link Layer Control Types](#)

LLCTRL Encoding	LLCTRL Type Name	Description	Retryable? (Enters the LLRB)
0001b	RETRY	Link layer RETRY flit	No
0000b	LLCRD	Flit containing link layer credit return and/or Ack information, but no protocol information.	Yes
0010b	IDE	Integrity and Data Encryption control messages. Use in flows described in Chapter 11.0 that were introduced in CXL 2.0.	Yes
1100b	INIT	Link layer initialization flit	Yes
Others	Reserved	N/A	N/A

The 3-bit CTL_FMT field was added to control messages and uses bits that were reserved in CXL 1.1 control messages. All control messages used in CXL 1.1 have this field encoded as 000b to maintain backward compatibility. This field is used to distinguish formats added in CXL 2.0 control messages that require a larger payload field. The new format increases the payload field from 64 bits to 96 bits and uses CTL_FMT encoding of 001b.

A detailed description of the control flits is presented below.

Table 4-10. CXL.cachemem Link Layer Control Details (Sheet 1 of 2)

Flit Type	CTL_FMT/ LLCTRL	SubType	SubType Description	Payload	Payload Description
LLCRD	000b/0000b	0000b	RSVD	63:0	RSVD
		0001b	Acknowledge	2:0	Acknowledge[2:0]
				3	RSVD
				7:4	Acknowledge[7:4]
				63:8	RSVD
		Others	RSVD	63:0	RSVD
RETRY	000b/0001b	0000b	RETRY.Idle	63:0	RSVD
		0001b	RETRY.Req	7:0	Requester's Retry Sequence Number (Eseq)
				15:8	RSVD
				20:16	Contains NUM_RETRY
				25:21	Contains NUM_PHY_REINIT (for debug)
				63:26	RSVD
		0010b	RETRY.Ack	0	Empty: The Empty bit indicates that the LLR contains no valid data and therefore the NUM_RETRY value should be reset
				1	Viral: The Viral bit indicates that the transmitting agent is in a Viral state
				2	RSVD
				7:3	Contains an echo of the NUM_RETRY value from the LLR.Req
				15:8	Contains the WrPtr value of the retry queue for debug purposes
				23:16	Contains an echo of the Eseq from the LLR.Req
				31:24	Contains the NumFreeBuf value of the retry queue for debug purposes
				47:32	Viral LD-ID Vector[15:0]: Included for MLD links to indicate which LD-ID is impacted by viral. Applicable only when the Viral bit (bit 1 of this payload) is set. Bit 0 of the vector encodes LD-ID=0, bit 1 is LD-ID=1, etc. Field is treated as Reserved for ports that do not support LD-ID.
		0011b	RETRY.Frame	63:48	RSVD
				63:0	Payload is RSVD. Flit required to be sent before a RETRY.Req or RETRY.Ack flit to allow said flit to be decoded without risk of aliasing.
		Others	RSVD	63:0	RSVD

Table 4-10. CXL.cachemem Link Layer Control Details (Sheet 2 of 2)

Flit Type	CTL_FMT/ LLCTRL	SubType	SubType Description	Payload	Payload Description
IDE	001b/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.
		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. Only used when no protocol messages exist to send.
		Others	RSVD	95:0	RSVD
INIT	000b/1100b	1000b	INIT.Param	3:0	Interconnect Version: Version of CXL the port is compliant with. CXL 1.0/1.1 = 0001b CXL 2.0 and above = 0010b Others Reserved
				7:4	RSVD
				12:8	RSVD
				23:13	RSVD
				31:24	LLR Wrap Value: Value after which LLR sequence counter should wrap to 0.
				63:32	RSVD
				63:0	RSVD

In the LLCRD flit, the total number of flit acknowledgments being returned is determined by creating the Full_Ack return value, where:

Full_Ack = {Acknowledge[7:4],Ak,Acknowledge[2:0]}, where the Ak bit is from the flit header.

The flit formats for the control flit are illustrated below.

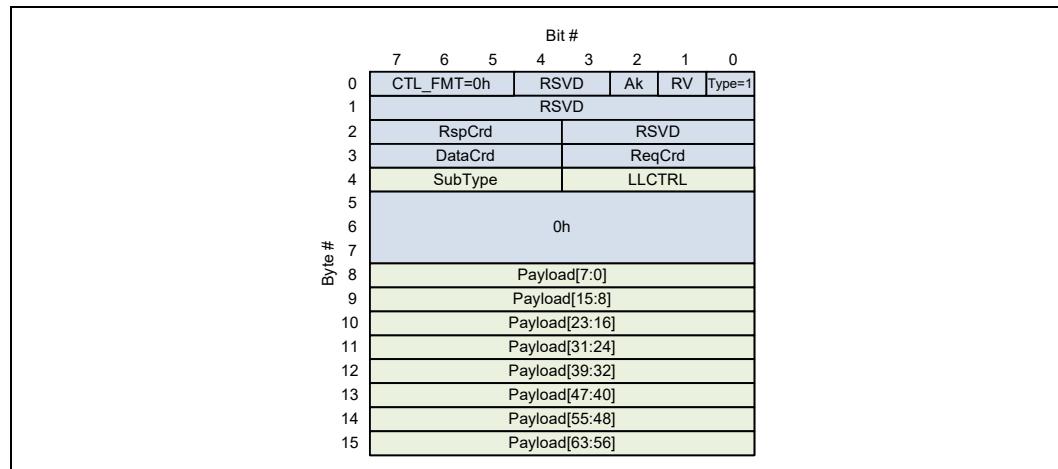
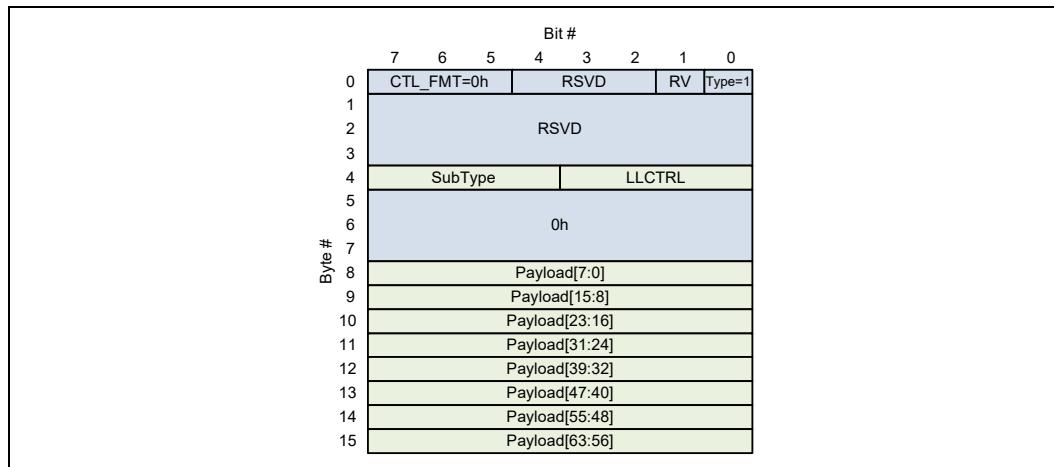
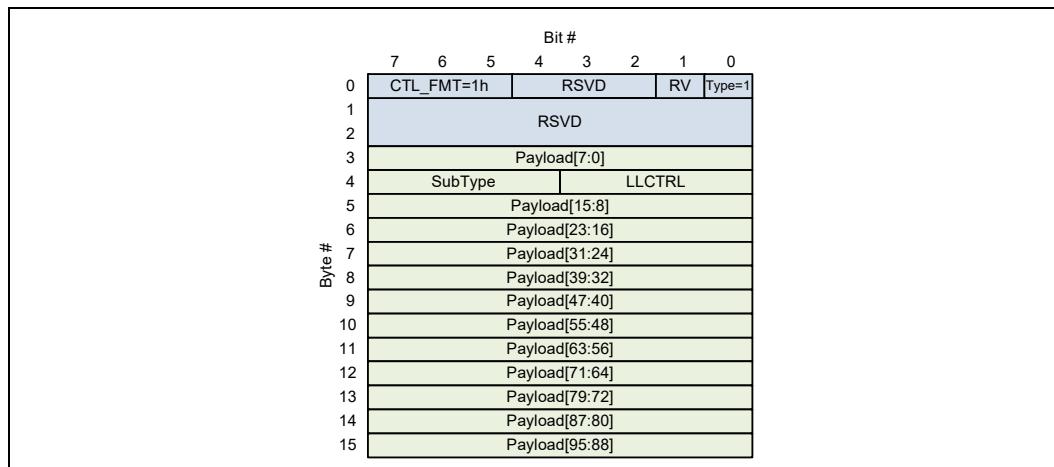
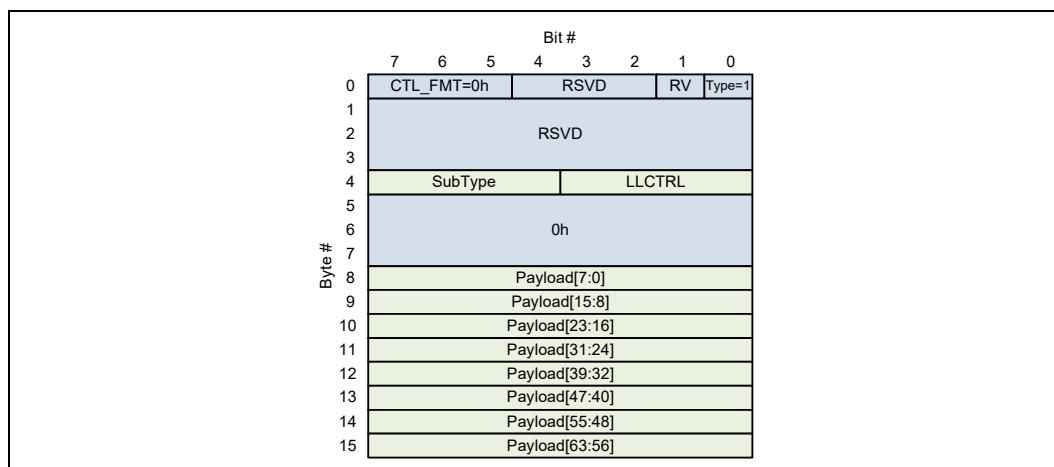
Figure 4-35. LLCRD Flit Format (Only Slot 0 is Valid; Others are Reserved)

Figure 4-36. RETRY Flit Format (Only Slot 0 is Valid; Others are Reserved)**Figure 4-37. IDE Flit Format (Only Slot 0 is Valid; Others are Reserved)****Figure 4-38. INIT Flit Format (Only Slot 0 is Valid; Others are Reserved)**

Note:

The RETRY.Req and RETRY.Ack flits belong to the type of flit to which receiving devices must respond, even in the shadow of a previous CRC error. In addition to checking the CRC of a RETRY flit, the receiving device should also check as many defined bits (those listed as having hardcoded 1/0 values) as possible to increase confidence in qualifying an incoming flit as a RETRY message.

4.2.7**Link Layer Initialization**

Link Layer Initialization must be started after a Physical Layer Link Down to Link Up transition and the link has trained successfully to L0. During Initialization and after the INIT flit has been sent, the CXL.cachemem Link Layer can only send Control-RETRY flits until Link Initialization is complete. The following describes how the link layer is initialized and credits are exchanged.

- 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 Control-INIT.Param flit. Before this condition is met, the Link Layer must transmit only Control-RETRY flits (i.e., RETRY.Frame/Req/Ack/Idle flits).
 - If for any reason the Rx portion of the Link Layer is not ready to begin processing flits beyond Control-INIT and Control-RETRY, the Tx will stall transmission of LLCTR-INIT.Param flit
 - RETRY.Frame/Req/Ack are sent during this time as part of the regular Retry flow.
 - RETRY.Idle flits are sent prior to sending a INIT.Param flit even without a retry condition to ensure the remote agent can observe a valid flit.
- The Control-INIT.Param flit must be the first non-Control-RETRY flit transmitted by the Link Layer
- The Rx portion of the Link Layer must be able to receive a Control-INIT.Param flit immediately upon completion of Physical Layer initialization because the first valid flit may be a Control-INIT.Param
- Received Control-INIT.Param values (i.e., LLR Wrap Value) must be made “active”, that is, applied to their respective hardware states within 8 flit clocks of error-free reception of Control-INIT.Param flit.
 - Until an error-free INIT.Param flit is received and these values are applied, LLR Wrap Value shall assume a default value of 9 for the purposes of ESEQ tracking.
- Any non-RETRY flits received before Control-INIT.Param flit will trigger an Uncorrectable Error.
- Only a single Control-INIT.Param flit is sent. Any CRC error conditions with a Control-INIT.Param flit will be dealt with by the Retry state machine and replayed from the Link Layer Retry Buffer.
- Receipt of a Control-INIT.Param flit after a Control-INIT.Param flit has already been received should be considered an Uncorrectable Error.
- It is the responsibility of the Rx to transmit credits to the sender using standard credit return mechanisms after link initialization. Each entity should know how many buffers it has and set its credit return counters to these values. Then, during normal operation, the standard credit return logic will return these credits to the sender.
- Immediately after link initialization, the credit exchange mechanism will use the LLCRD flit format.
- It is possible that the receiver will make more credits available than the sender can track for a given message class. For correct operation, it is therefore required that the credit counters at the sender be saturating. Receiver will drop all credits it receives for unsupported channels (e.g., Type 3 device receiving any CXL.cache credits).

- Credits should be sized to achieve desired levels of bandwidth considering round-trip time of credit return latency. This is implementation and usage dependent.

4.2.8

CXL.cachemem Link Layer Retry

The link layer provides recovery from transmission errors using retransmission, or Link Layer Retry (LLR). The sender buffers every retryable flit sent in a local Link Layer Retry Buffer (LLRB). To uniquely identify flits in this buffer, the retry scheme relies on sequence numbers which are maintained within each device. Unlike in PCIe, CXL.cachemem sequence numbers are not communicated between devices with each flit to optimize link efficiency. The exchange of sequence numbers occurs only through link layer control flits during an LLR sequence. The sequence numbers are set to a predetermined value (0) during Link Layer Initialization and they are implemented using a wraparound counter. The counter wraps back to 0 after reaching the depth of the retry buffer. This scheme makes the following assumptions:

- The round-trip delay between devices is more than the maximum of the link layer clock or flit period.
- All protocol flits are stored in the retry buffer. See [Section 4.2.8.5.1](#) for further details on the handling of non-retryable control flits.

Note that for efficient operation, the size of the retry buffer must be larger than the round-trip delay. This includes:

- Time to send a flit from the sender
- Flight time of the flit from sender to receiver
- Processing time at the receiver to detect an error in the flit
- Time to accumulate and, if needed, force Ack return and send embedded Ack return back to the sender
- Flight time of the Ack return from the receiver to the sender
- Processing time of Ack return at the original sender

Otherwise, the LLR scheme will introduce latency, as the transmitter will have to wait for the receiver to confirm correct receipt of a previous flit before the transmitter can free space in its LLRB and send a new flit. Note that the error case is not significant because transmission of new flits is effectively stalled until successful retransmission of the erroneous flit anyway.

4.2.8.1

LLR Variables

The retry scheme maintains two state machines and several state variables. Although the following text describes them in terms of one transmitter and one receiver, both the transmitter and receiver side of the retry state machines and the corresponding state variables are present at each device because of the bidirectional nature of the link. Since both sides of the link implement both transmitter and receiver state machines, for clarity this discussion will use the term “local” to refer to the entity that detects a CRC error, and “remote” to refer to the entity that sent the flit that was erroneously received.

The receiving device uses the following state variables to keep track of the sequence number of the next flit to arrive.

- **ESeq:** This indicates the expected sequence number of the next valid flit at the receiving link layer entity. ESeq is incremented by one (modulo the size of the LLRB) on error-free reception of a retryable flit. ESeq stops incrementing after an error is detected on a received flit until retransmission begins (RETRY.Ack message is received). Link Layer Initialization sets ESeq to 0. Note that there is no way for the receiver to know that an error was for a non-retryable vs. retryable flit. For any

CRC error, it will initiate the link layer retry flow as usual, and effectively the transmitter will resend from the first retryable flit sent.

The sending entity maintains two indexes into its LLRB, as indicated below.

- **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 wraparound counter that wraps around to 0 after reaching the depth of the LLRB. Non-Retryable Control flits do not affect the WrPtr. WrPtr stops incrementing after receiving an error indication at the remote entity (RETRY.Req message) except as described in the implementation note below, 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 placed 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 will ensure that All Data Flits not interleaved with other flits are correctly logged into the Link Layer Retry Buffer.

- **RdPtr:** This is used to read the contents out of the LLRB during a retry scenario. The value of this pointer is set by the sequence number sent with the retransmission request (RETRY.Req message). The RdPtr is incremented by one (modulo the size of the LLRB) whenever a flit is sent, either from the LLRB in response to a retry request or when a new flit arrives from the transaction layer and regardless of the states of the local or remote retry state machines. If a flit is being sent when the RdPtr and WrPtr are the same, then it indicates that a new flit is being sent; otherwise, it must be a flit from the retry buffer.

The LLR scheme uses an explicit acknowledgment that is sent from the receiver to the sender to remove flits from the LLRB at the sender. The acknowledgment is indicated via an ACK bit in the headers of flits flowing in the reverse direction. In CXL.cachemem, a single ACK bit represents 8 acknowledgments. Each entity keeps track of the number of available LLRB entries and the number of received flits pending acknowledgment through the following variables.

- **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 queue entries 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 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 with Ak bit set), else an 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.
- **NumAck:** This indicates the number of acknowledgments accumulated at the receiver. NumAck increments by 1 when a retryable flit is received. NumAck is decremented by 8 when the ACK bit is set in the header of an outgoing flit. If the

outgoing flit is coming from the LLRB and its ACK bit is set, NumAck does not decrement. At initialization, NumAck is set to 0. The minimum size of the NumAck field is the size of the LLRB. NumAck at each entity must be able to keep track of at least 255 acknowledgments.

The LLR protocol requires that the number of retry queue entries at each entity must be at least 22 entries (Size of Forced Ack (16) + Max All-Data-Flit (4) + 2) to prevent deadlock.

4.2.8.2 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 asymmetric, this requirement could result in traffic throttling and possibly even starvation.

Suppose that the A→B direction has heavy traffic, but there is no traffic 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 Acknowledges. 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 permit LLCRD Control message forcing (injection of a non-traffic flit to carry an Acknowledge and a Credit return (ACK/CRD)), but this function must be constrained to avoid wasting bandwidth. In CXL, when B has accumulated a programmable minimum number of Acknowledges to return, B's CXL.cachemem link layer will inject an LLCRD flit to return an Acknowledge. 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 (see [Section 8.2.4.19.6](#)).

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 in which 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" instead of "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 CXL.cachemem link layer must accumulate a minimum of 8 Acknowledges to set the ACK bit in a CXL.cachemem flit header. If LLCRD forcing occurred after the accumulation of 8 Acknowledges, 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 Acknowledges to re-accumulate to set the ACK bit. In the worst case, this could double the bandwidth consumption of the CXL.cachemem side. By waiting for at least 16 Acknowledges to accumulate, the CXL.cachemem link layer ensures that it can still opportunistically return Acknowledges in a protocol flit avoiding the need to force an LLCRD for Ack return. It is recommended that the Ack Force Threshold value be set to 16 or greater in the CXL Link Layer Ack Timer Control register to reduce overhead of LLCRD injection.

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

Pseudo-code 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_or_CRD_Flush_Retimer OR ACK_TO_RETURN >= Ack Force_Threshold)

    Force_LLCRD = TRUE

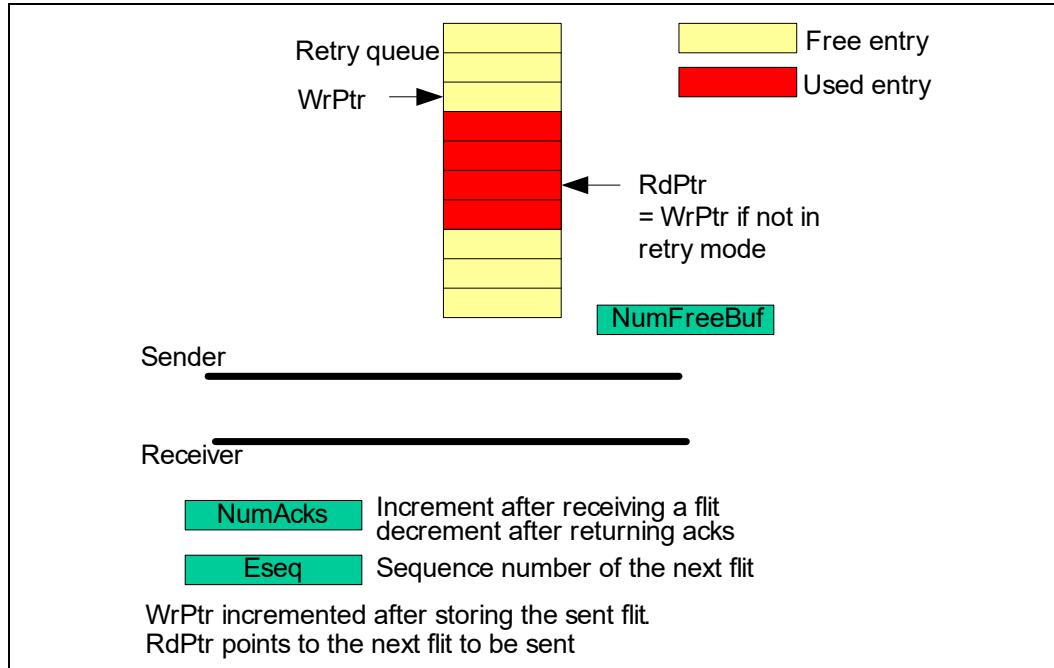
ELSE

    Force_LLCRD=FALSE

```

Note: Ack or CRD Flush Retimer and Ack Force Threshold are values that come from the CXL Link Layer Ack Timer Control register (see [Section 8.2.4.19.6](#)).

Figure 4-39. Retry Buffer and Related Pointers



4.2.8.3 LLR Control Flits

The LLR Scheme uses several link layer control flits of the RETRY format to communicate the state information and the implicit sequence numbers between the entities.

- RETRY.Req: This flit is sent from the entity that received a flit in error to the sending entity. The flit contains the expected sequence number (ESeq) at the receiving entity, indicating the index of the flit in the retry queue at the remote entity that must be retransmitted. It also contains the NUM_RETRY value of the sending entity which is defined in [Section 4.2.8.5.1](#). This message is also triggered as part of the Initialization sequence even when no error is observed as described in [Section 4.2.7](#).

- RETRY.Ack: This flit is sent from the entity that is responding to an error detected at the remote entity. It contains a reflection of the NUM_RETRY value from the corresponding RETRY.Req message. The flit contains the WrPtr value at the sending entity for debug purposes only. The WrPtr value should not be used by the retry state machines in any way. This flit will be followed by the flit identified for retry by the ESeq number.
- RETRY.Idle: This flit is sent during the retry sequence when there are no protocol flits to be sent (see [Section 4.2.8.5.2](#) for details) or a retry queue is not ready to be sent. For example, it can be used for debug purposes for designs that need additional time between sending the RETRY.Ack and the actual contents of the LLR queue.
- RETRY.Frame: This flit is sent along with a RETRY.Req or RETRY.Ack flit to prevent aliased decoding of these flits (see [Section 4.2.8.5](#) for further details).

Table 4-11 describes the impact of RETRY messages on the local and remote retry state machines. In this context, the “sender” refers to the Device sending the message and the “receiver” refers to the Device receiving the message. Note that how this maps to which device detected the CRC error and which sent the erroneous message depends on the message type. For example, for a RETRY.Req sequence, the sender detected the CRC error, but for a RETRY.Ack sequence, it’s the receiver that detected the CRC error.

4.2.8.4 RETRY Framing Sequences

Recall that the CXL.cachemem flit formatting specifies an all-data flit for link efficiency. This flit is encoded as part of the header of the preceding flit and contains no header information of its own. This introduces the possibility that the data contained in this flit could happen to match the encoding of a RETRY flit.

This introduces a problem at the receiver. It must be certain to decode the actual RETRY flit, but it must not falsely decode an aliasing data flit as a RETRY flit. In theory it might use the header information of the stream it receives in the shadow of a CRC error to determine whether it should attempt to decode the subsequent flit. Therefore, the receiver cannot know with certainty which flits to treat as header-containing (decode) and which to ignore (all-data).

CXL introduces the RETRY.Frame flit for this purpose to disambiguate a control sequence from an All-Data Flit (ADF). Due to MDH, 4 ADF can be sent back-to-back. Hence, a RETRY.Req sequence comprises 5 RETRY.Frame flits immediately followed by a RETRY.Req flit, and a RETRY.Ack sequence comprises 5 RETRY.Frame flits immediately followed by a RETRY.Ack flit. This is shown in [Figure 4-40](#).

Table 4-11. Control Flits and Their Effect on Sender and Receiver States

RETRY Message	Sender State	Receiver State
RETRY.Idle	Unchanged.	Unchanged.
RETRY.Frame + RETRY.Req Sequence	Local Retry State Machine (LRSM) is updated. NUM_RETRY is incremented. See Section 4.2.8.5.1 .	Remote Retry State Machine (RRSM) is updated. RdPtr is set to ESeq sent with the flit. See Section 4.2.8.5.3 .
RETRY.Frame + RETRY.Ack Sequence	RRSM is updated.	LRSM is updated.
RETRY.Frame, RETRY.Req, or RETRY.Ack message that is not as part of a valid framed sequence	Unchanged.	Unchanged (drop the flit).

Note:

A RETRY.Ack sequence that arrives when a RETRY.Ack is not expected will be treated as an error by the receiver. Error resolution in this case is device specific though it is recommended that this results in the machine halting operation. It is recommended that this error condition not change the state of the LRSM.

4.2.8.5 LLR State Machines

The LLR scheme is implemented with two state machines: Remote Retry State Machine (RRSM) and Local Retry State Machine (LRSM). These state machines are implemented by each entity and together determine the overall state of the transmitter and receiver at the entity. The states of the retry state machines are used by the send and receive controllers to determine what flit to send and the actions needed to process a received flit.

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_LLREQ: 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 the virtual Link State Machine (vLSM) being in 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_LLREQ. The TIMEOUT counter is also reset when the vLSM transitions from Retrain to Active (the LRSM transition through RETRY_PHY_REINIT to RETRY_LLREQ). 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_LLREQ 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 (the LRSM transition through RETRY_PHY_REINIT to RETRY_LLREQ).

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 that are generated during an LLR sequence due to the number of retries that exceed 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_LLREQ to RETRY_PHY_REINIT due to the number of retries that exceed MAX_NUM_RETRY. If the counter reaches a threshold (called MAX_NUM_PHY_REINIT) instead of transitioning from RETRY_LLREQ 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 at which the retry requesting device times out and sends a new RETRY.Req sequence (by appropriately setting the value of TIMEOUT – see [Section 4.2.8.5.2](#)). 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 (Sheet 1 of 2)

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_LLREQ	Received flit is discarded.
RETRY_LOCAL_NORMAL	PHY_RESET ¹ / PHY_REINIT ² is detected.	RETRY_PHY_REINIT	None.
RETRY_LLREQ	NUM_RETRY == MAX_NUM_RETRY and NUM_PHY_REINIT == MAX_NUM_PHY_REINIT	RETRY_ABORT	Indicate link failure.
RETRY_LLREQ	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.

Table 4-12. Local Retry State Transitions (Sheet 2 of 2)

Current Local Retry State	Condition	Next Local Retry State	Actions
RETRY_LLREQ	NUM_RETRY < MAX_NUM_RETRY and a RETRY.Req sequence has not been sent.	RETRY_LLREQ	If an error-free RETRY.Req or RETRY.Ack sequence is received, process the flit. Any other flit is discarded.
RETRY_LLREQ	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_LLREQ	PHY_RESET ¹ / PHY_REINIT ² is detected.	RETRY_PHY_REINIT	None.
RETRY_LLREQ	Error is detected on a received flit	RETRY_LLREQ	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_LLREQ	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_LLREQ	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 the vLSM informing the Link Layer that it needs to initiate a Link Layer Retry due to exit from Retrain state.
2. PHY_REINIT is the condition of the Link Layer instructing the Phy to retrain.

4.2.8.5.2 TIMEOUT Definition

After the local receiver has detected a CRC error, triggering the LRSM, the local Tx sends a RETRY.Req sequence to initiate LLR. At this time, the local Tx also starts its TIMEOUT counter.

The purpose of this counter is to decide that either the RETRY.Req sequence or corresponding RETRY.Ack sequence has been lost, and that another RETRY.Req attempt should be made. Recall that it is a fatal error to receive multiple RETRY.Ack sequences (i.e., a subsequent Ack without a corresponding Req is unexpected). To reduce the risk of this fatal error condition we check NUM_RETRY value returned to filter out RETRY.Ack messages from the prior retry sequence. This is done to remove fatal condition where a

single retry sequence incurs a timeout while the Ack message is in flight. The TIMEOUT counter should be capable of handling worst-case latency for a RETRY.Req sequence to reach the remote side and for the corresponding RETRY.Ack sequence to return.

Certain unpredictable events (e.g., low power transitions, etc.) that interrupt link availability could add a large amount of latency to the RETRY round-trip. To make the TIMEOUT robust to such events, instead of incrementing per link layer clock, TIMEOUT increments whenever the local Tx transmits a flit, protocol, or control. Due to the TIMEOUT protocol, TIMEOUT must force injection of RETRY.Idle flits if it has no real traffic to send, so that the TIMEOUT counter continues to increment.

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_LLACK: 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 [Table 4-13](#).

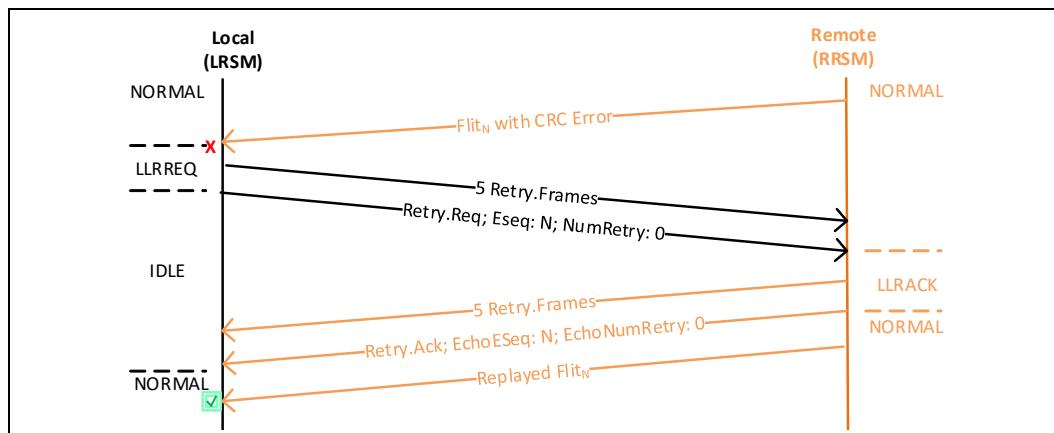
Table 4-13. Remote Retry State Transition

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_LLACK
RETRY_LLACK	RETRY.Ack sequence is not sent.	RETRY_LLACK
RETRY_LLACK	RETRY.Ack sequence is sent.	RETRY_REMOTE_NORMAL
RETRY_LLACK	vLSM in Retrain state.	RETRY_REMOTE_NORMAL

Note: To select the priority of sending flits, the following rules apply:

1. Whenever the RRSM state becomes RETRY_LLACK, the entity must give priority to sending the Control flit with RETRY.Ack.
2. Except RRSM state of RETRY_LLACK, the priority goes to LRSR state of RETRY_LLREQ and in that case the entity must send a Control flit with RETRY.Req over all other flits except an all-data flit sequence.

The overall sequence of replay is shown in [Figure 4-40](#).

Figure 4-40. CXL.cachemem Replay Diagram

4.2.8.6

Interaction with vLSM Retrain State

On 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, 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.

4.2.8.7

CXL.cachemem Flit CRC

The CXL.cachemem Link Layer uses a 16b CRC for transmission error detection. The 16b CRC is over the 528-bit flit. The assumptions about the type errors is as follows:

- Bit ordering runs down each lane.
- Bit Errors occur randomly or in bursts down a lane, with the majority of the errors being single-bit random errors.
- Random errors can statistically cause multiple bit errors in a single flit, so it is more likely to get 2 errors in a flit than 3 errors, and more likely to get 3 errors in a flit than 4 errors, and so on.
- There is no requirement for primitive polynomial (a polynomial that generates all elements of an extension field from a base field) because there is no fixed payload. Primitive may be the result, but it's not required.

4.2.8.7.1

CRC-16 Polynomial and Detection Properties

The CRC polynomial to be used is 1F053h. The 16b CRC Polynomial has the following properties:

- All single, double, and triple bit errors detected
- Polynomial selection based on best 4-bit error detection characteristics and perfect 1-bit, 2-bit, and 3-bit error detection

4.2.8.7.2 CRC-16 Calculation

Below are the 512 bit data masks for use with an XOR tree to produce the 16 CRC bits. Data Mask bits [511:0] for each CRC bit are applied to the flit bits [511:0] and XOR is performed. The resulting CRC bits are included as flit bits [527:512] are defined to be CRC[15:0]. Pseudo code example for CRC bit 15 of this is $\text{CRC}[15] = \text{XOR}(\text{DM}[15][511:0] \text{ AND } \text{Flit}[511:0])$.

The flit Data Masks for the 16 CRC bits are located 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
```

4.2.9

Viral

Viral is a containment feature as described in [Section 12.4, "CXL Viral Handling."](#) As such, when the local socket is in a viral state, it is the responsibility of all off-die interfaces to convey this state to the remote side for appropriate handling. The CXL.cachemem link layer conveys viral status information. As soon as the viral status is detected locally, the link layer forces a CRC error on the next outgoing flit. If there is no traffic to send, the transmitter will send an LLRD flit with a CRC error. It then embeds viral status information in the RETRY.Ack message it generates as part of the defined CRC error recovery flow.

There are two primary benefits to this methodology. First, by using the RETRY.Ack to convey viral status, we do not have to allocate a bit for this in protocol flits. Second, it allows immediate indication of viral and reduces the risk of race conditions between the viral distribution path and the data path. These risks could be particularly exacerbated by the large CXL.cache flit size and the potential limitations in which components (header, slots) allocate dedicated fields for viral indication.

To support MLD components, first introduced in CXL 2.0, a Viral LD-ID Vector is defined in the RETRY.Ack to encode which LD-ID is impacted by the viral state. This allows viral to be indicated to any set of Logical Devices. This vector is applicable only when the primary viral bit is set, and only to links that support multiple LD-ID (referred to as MLD - Multi-Logical Device). Links without LD-ID support (referred to as SLD - Single Logical Device) will treat the vector as Reserved. For MLD, the encoding of all 0s indicates that all LD-ID are in viral and is equivalent to an encoding of all 1s.

4.3

CXL.cachemem Link Layer 256B Flit Mode

4.3.1

Introduction

This mode of operation builds on PCIe Flit mode, in which the reliability flows are handled in the Physical Layer. The flit definition in the link layer defines the slot boundary, slot packing rules, and the message flow control. The flit overall has fields that are defined in the physical layer and are shown in this chapter; however, details are not defined in this chapter. The concept of "all Data" as defined in 68B Flit mode does not exist in 256B Flit mode.

4.3.2

Flit Overview

There are 2 variations of the 256B flit: Standard, and Latency-Optimized (LOpt). The mode of operation must be in sync with the physical layer. The Standard 256B flit supports either standard messages or Port Based Routing (PBR) messages where PBR messages carry additional ID space (DPID and sometimes SPID) to enable more-advanced scaling/routing solutions as described in [Chapter 3.0](#).

Note:

256B flit messages are also referred to as Hierarchy Based Routing (HBR) messages, when comparing to PBR flits/messages. A message default is HBR unless explicitly stated as being PBR.

Figure 4-41 is the Standard 256B flit. The Physical Layer controls 16B of the flit in this mode where the fields are: HDR, CRC, and FEC. All other fields are defined in the link layer.

Figure 4-41. Standard 256B Flit

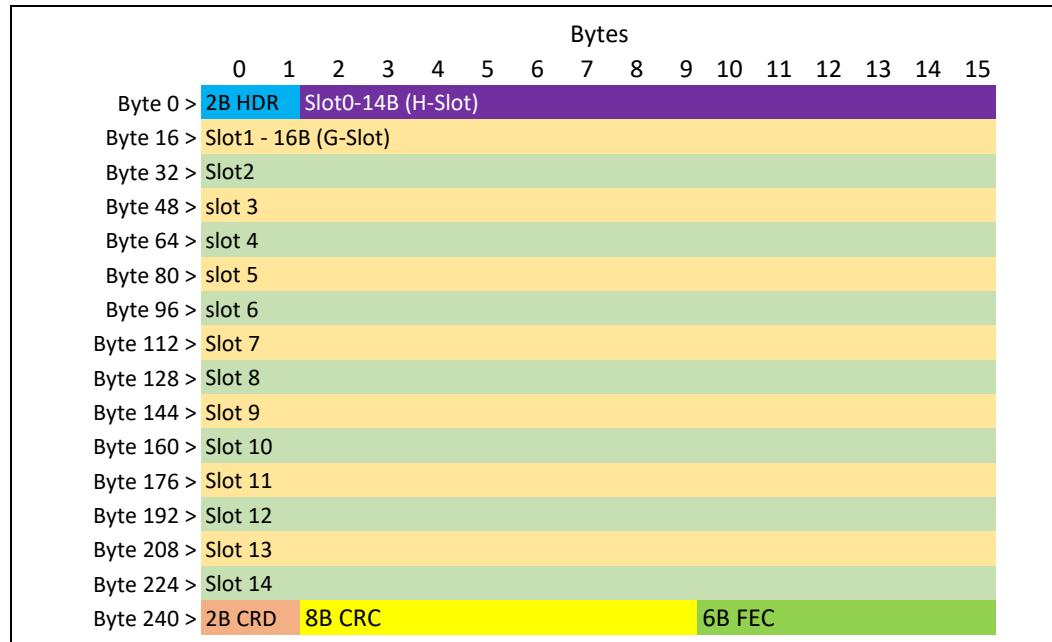
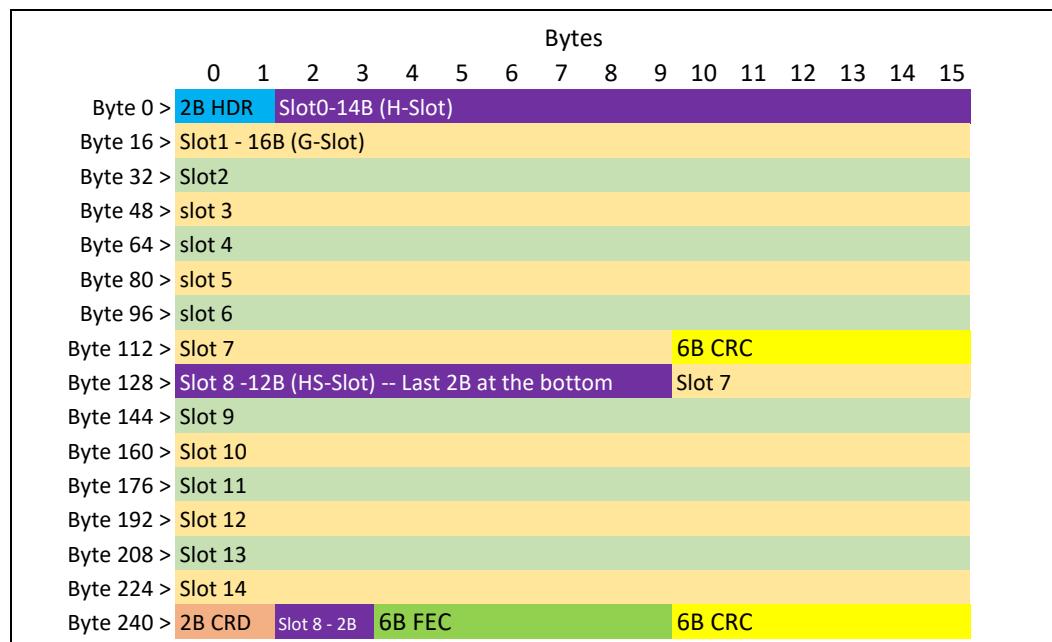


Figure 4-42 is the latency-optimized flit definition. In this definition, more bytes are allocated to the physical layer to enable less store-and-forward when the transmission is error free. In this flit, 20B are allocated to the Physical Layer, where the fields are: 12B CRC (split across 2 6B CRC codes), 6B FEC, and 2B HDR.

Figure 4-42. Latency-Optimized (LOpt) 256B Flit



In both flit modes, the flit message packing rules are common, with the exception of Slot 8, which in LOpt 256B flits is a 12B slot with special packing rules. These are a subset of Slot 0 packing rules. This slot format is referred to as the H Subset (HS) format.

PBR packing is a subset of HBR message packing rules. PBR messages are not supported in LOpt 256B Flits, so HS-Slot does not apply.

Note: Some bits of Slot 7 are split across the 128B halves of the flit, and the result is that some messages in Slot 7 cannot be consumed until the CRC for the second half of the flit is checked.

Slot formats are defined by a 4-bit field at the beginning of each slot that carries header information, which is a departure from the 68B formats, where the 3-bit format field is within the flit header. The packing rules are constrained to a subset of messages for upstream and downstream links to match the Transaction Layer requirements. The encodings are non-overlapping between upstream and downstream except when the message(s) in the format are enabled to be sent in both directions. This is a change from the 68B flit definition where the slot format was uniquely defined for upstream and downstream.

The packing rules for the H-slot are a strict subset of the G-slot rules. The subset relationship is defined by the 14B H-slot size where any G-slot messages that extend beyond the 14th byte are not supported in the H-slot format. HS-slot follows the same subset relationship where the cutoff size is 12B.

For the larger PBR message packing, the messages in each slot are a subset of 256B flit message packing rules because of the larger message size required for PBR. PBR flits and messages can be fully symmetric when flowing between switches where the link is not upstream or downstream (also known as "Cross-Link" or "Inter-Switch Link" (ISL)).

For Data and Byte-Enable Slots, a slot-format field is not explicitly included, but is instead known based on prior header messages that must be decoded. This is similar to the "all-data-flit" definition in 68B flit where expected data slots encompass the flit's entire payload.

Table 4-14 defines the 256B G-Slots for HBR and PBR messages.

Table 4-14. 256B G-Slot Formats (Sheet 1 of 2)

Format	SlotFmt Encoding	HBR				PBR	
		Messages	Downstream¹	Upstream¹	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	D	100	M2S Req	120
G5	0101b	3 M2S BIRsp	X	D	120	2 M2S BIRsp	104
G6	0110b	S2M BISnp + S2M NDR	D	X	124	S2M BISnp	96
G7	0111b	3 S2M NDR	D	X	120	2 S2M NDR	96

Table 4-14. 256B G-Slot Formats (Sheet 2 of 2)

Format	SlotFmt Encoding	HBR				PBR	
		Messages	Downstream ¹	Upstream ¹	Length in Bits (Max 124)	Messages	Length in Bits (Max 124)
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	D	104	M2S Rwd	124
G15	1111b	3 S2M DRS	D	X	120	2 S2M DRS	96

1. D = Supported only for Direct P2P CXL.mem-capable ports.

Table 4-15 captures the H-Slot formats. Notice that “zero extended” is used in PBR messages sent using slot formats H4 and H14 because they do not fit in the slot. This method allows the messages to use this format provided the unused bits are 0s. The zero-extended method can be avoided by using the G-slot format, but use is allowed for these cases to optimize link efficiency. An example PBR H14, in [Figure 4-65, “256B Packing: G14/H14 PBR Messages” on page 242](#), requires that the bits in Bytes 14 and 15 are all 0s to be able to use the format. This includes CKID[12:8], TC field, and reserved bits within those bytes. Any other field, including CKID[7:0], will be sent normally and can have supported encodings.

Table 4-15. 256B H-Slot Formats (Sheet 1 of 2)

Format	SlotFmt Encoding	HBR				PBR	
		Messages	Downstream ¹	Upstream ¹	Length in Bits (Max 108)	Messages	Length in Bits (Max 108)
H0	0000b	H2D Req ²	X		72	H2D Req	92
H1	0001b	2 H2D Rsp ²	X		80	2 H2D Rsp	96
H2	0010b	D2H Req + 1 D2H Rsp ²		X	100	D2H Req	96
H3	0011b	4 D2H Rsp		X	96	3 D2H Rsp	108
H4	0100b	M2S Req	X	D	100	M2S Req (Zero Extended)	108 (120)
H5	0101b	2 M2S BIRsp ²	X	D	80	2 M2S BIRsp	104
H6	0110b	S2M BISnp ²	D	X	84	S2M BISnp	96
H7	0111b	2 S2M NDR ²	D	X	80	2 S2M NDR	96
H8	1000b	LLCTRL				LLCTRL	
H9	1001b	RSVD				RSVD	
H10	1010b						
H11	1011b						
H12	1100b	3 H2D DH ²	X		84	3 H2D DH	108

Table 4-15. 256B H-Slot Formats (Sheet 2 of 2)

Format	SlotFmt Encoding	HBR				PBR	
		Messages	Downstream ¹	Upstream ¹	Length in Bits (Max 108)	Messages	Length in Bits (Max 108)
H13	1101b	4 D2H DH		X	96	3 D2H DH	108
H14	1110b	M2S Rwd	X	D	104	M2S Rwd (Zero Extended)	108 (124)
H15	1111b	2 S2M DRS ²	D	X	80	2 S2M DRS	96

1. D = Supported only for Direct P2P CXL.mem-capable ports.

2. Cases in which the H-Slot is a subset of the corresponding G-slot because not all messages fit into the format.

Table 4-16 captures the HS-Slot formats. The HS-slot format is used only in LOpt 256B flits. Notice that “zero extended” for slot formats are used in HS4 and HS14.

Note: PBR messages never use LOpt 256B flits, and therefore do not use the HS-Slot format.

Table 4-16. 256B HS-Slot Formats

Format	SlotFmt Encoding	HBR			
		Messages	Downstream ¹	Upstream ¹	Length in Bits (Max 92)
HS0	0000b	H2D Req	X		72
HS1	0001b	2 H2D Rsp	X		80
HS2	0010b	D2H Req ²		X	76
HS3	0011b	3 D2H Rsp ²		X	72
HS4	0100b	M2S Req (Zero Extended)	X	D	92 (100)
HS5	0101b	2 M2S BIRsp	X	D	80
HS6	0110b	S2M BISnp	D	X	84
HS7	0111b	2 S2M NDR	D	X	80
HS8	1000b	LLCTRL			
HS9	1001b				
HS10	1010b	RSVD			
HS11	1011b				
HS12	1100b	3 H2D DH	X		84
HS13	1101b	3 D2H DH ²		X	72
HS14	1110b	M2S Rwd (Zero Extended)	X	D	92 (104)
HS15	1111b	2 S2M DRS	D	X	80

1. D = Supported only for Direct P2P CXL.mem-capable ports.

2. Cases in which the HS-Slot is a subset of the corresponding H-slot because not all messages fit into the format.

4.3.3

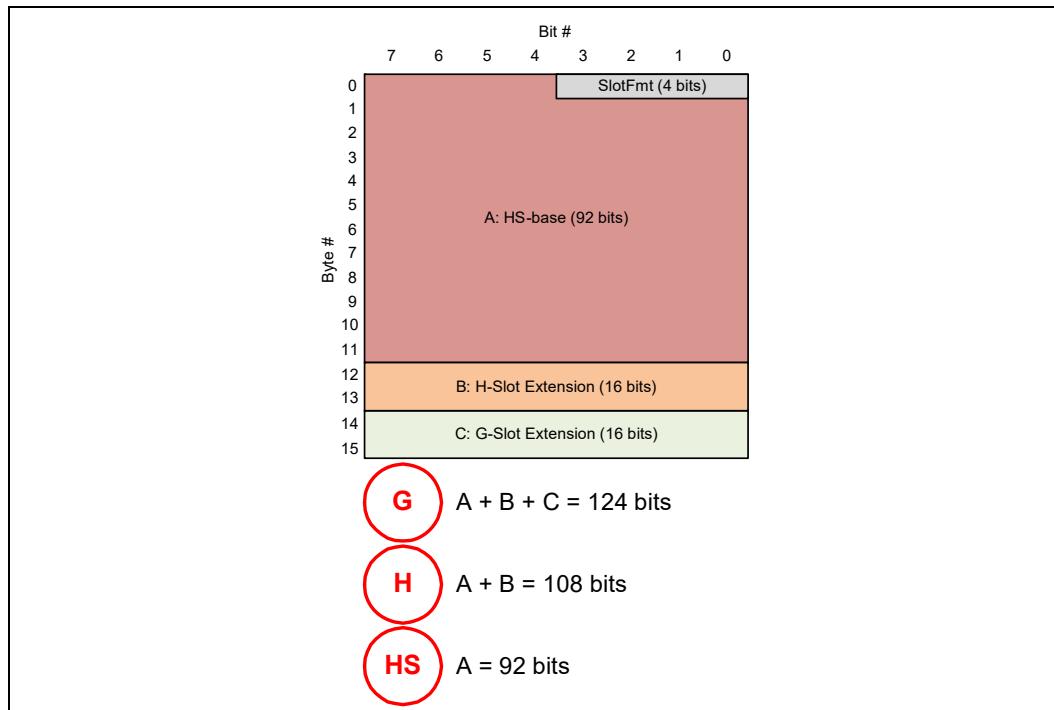
Slot Format Definition

The slot diagrams in this section capture the detailed bit field placement within the slot. Each Diagram is inclusive of G-slot, H-slot, and HS-slot where a subset is created such that H-slot is a subset of G-slot where messages that extend beyond the 14-byte boundary are excluded. Similarly, the HS-slot format is a subset of H-slot and G-slot where messages that extend beyond the 12-byte boundary are excluded.

This G to H to HS subset relationship is captured in [Figure 4-43](#), where the size of each subset is shown.

All messages within the slots are aligned to nibble (4 bit) boundary. This results in some variation in number of reserved bits to align to that boundary.

Figure 4-43. 256B Packing: Slot and Subset Definition



Slot diagrams in the section include abbreviations for bit field names to allow them to fit into the diagram. In the diagrams, most abbreviations are obvious, but the following abbreviation list ensures clarity:

- Bg = Bogus
- BT11 = BITag[11]
- Ch = ChunkValid
- CID3 = CacheID[3]
- CK12 = CKID[12]
- CQ0 = CQID[0]
- CQ11 = CQID[11]
- DP0 = DPID[0]
- LD0 = LD-ID[0]

EVALUATION COPY

- MO3 = MemOpcode[3]
- Op3 = Opcode[3]
- Poi = Poison
- RSVD = Reserved
- RV = Reserved
- SP11 = SPID[11]
- UQ11 = UQID[11]
- Val = Valid

Figure 4-44. 256B Packing: G0/H0/HS0 HBR Messages

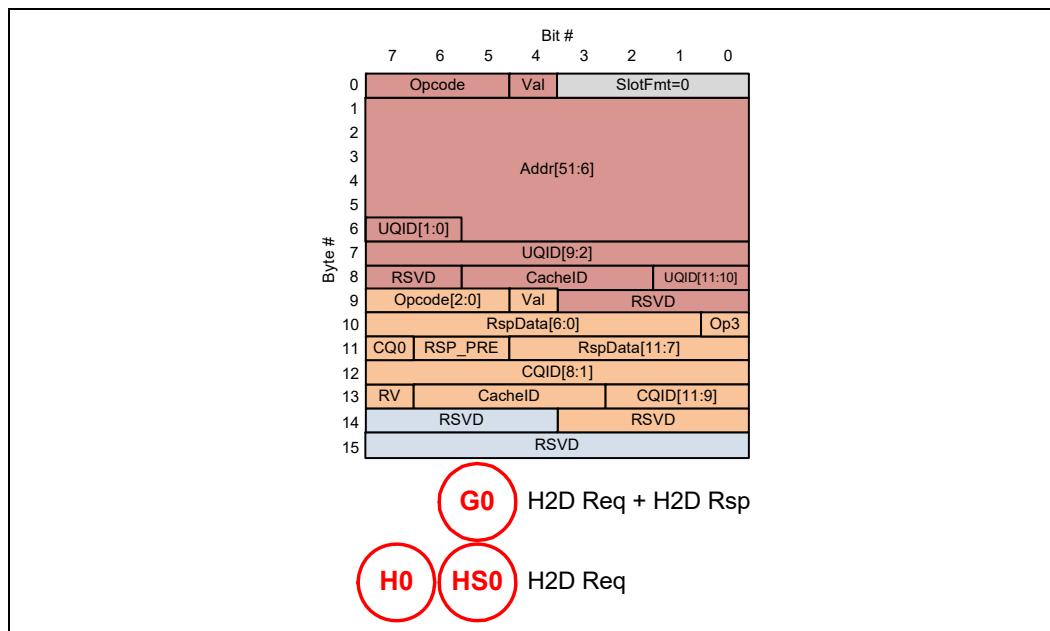
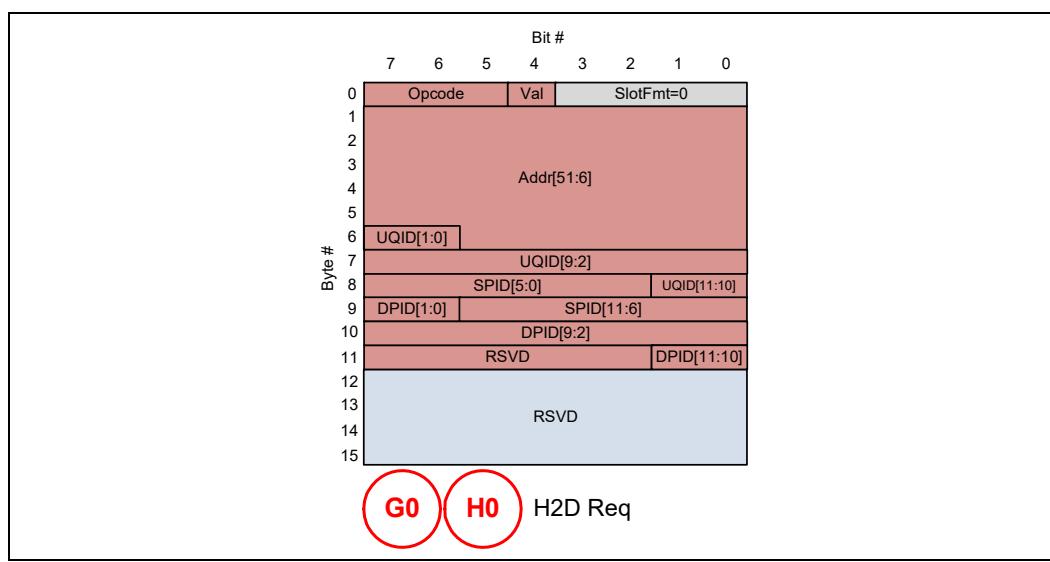


Figure 4-45. 256B Packing: G0/H0 PBR Messages



EVALUATION COPY

Figure 4-46. 256B Packing: G1/H1/HS1 HBR Messages

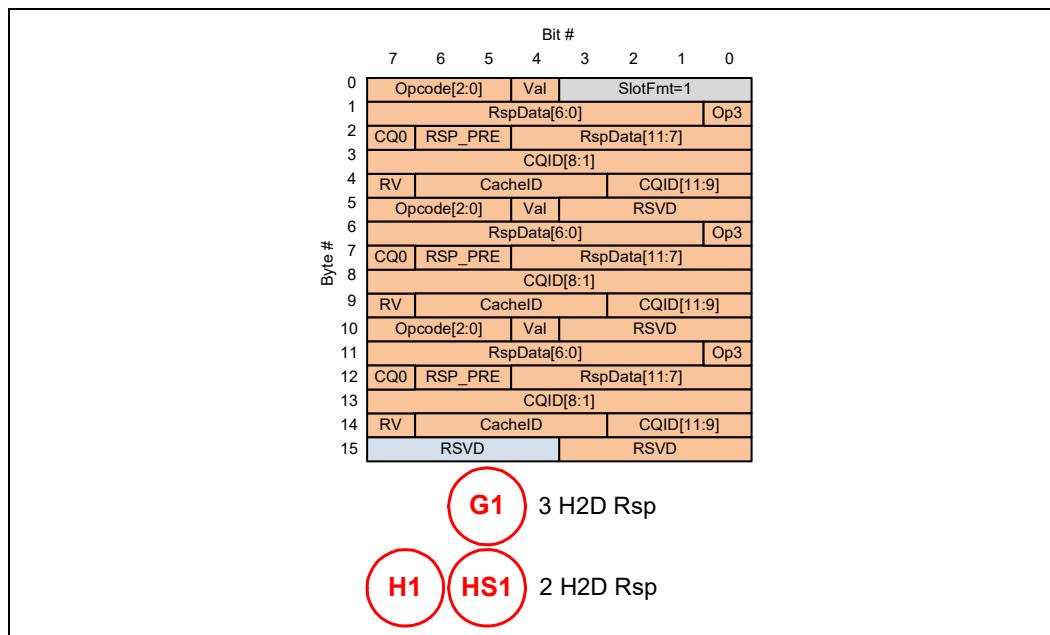
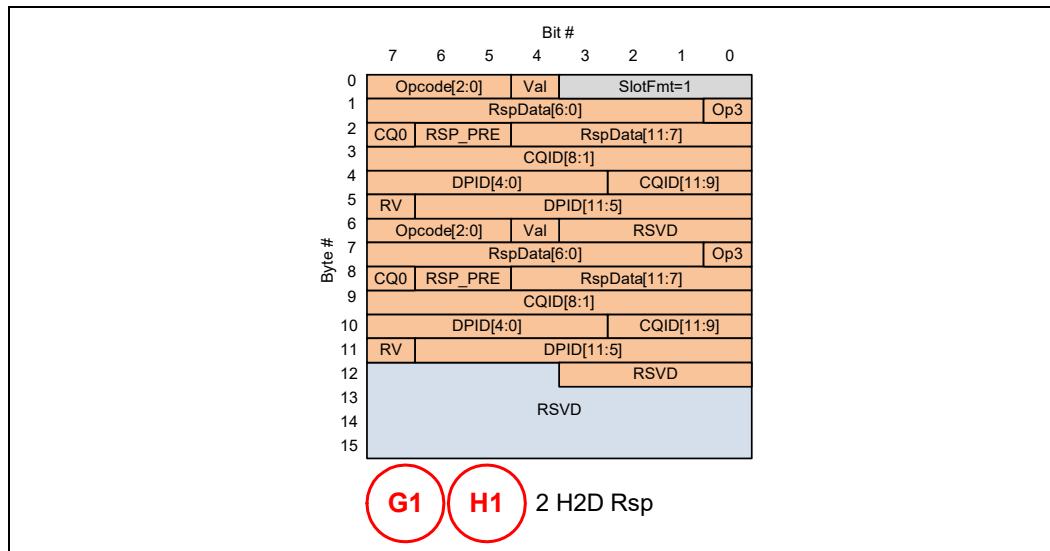


Figure 4-47. 256B Packing: G1/H1 PBR Messages



EVALUATION COPY

Figure 4-48. 256B Packing: G2/H2/HS2 HBR Messages

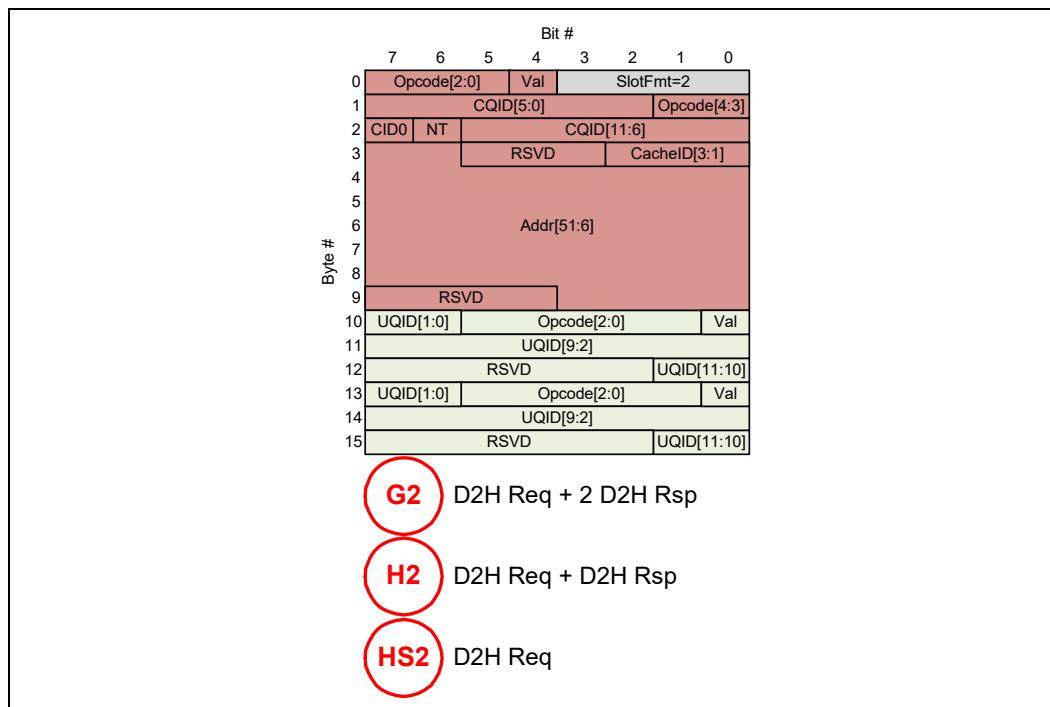
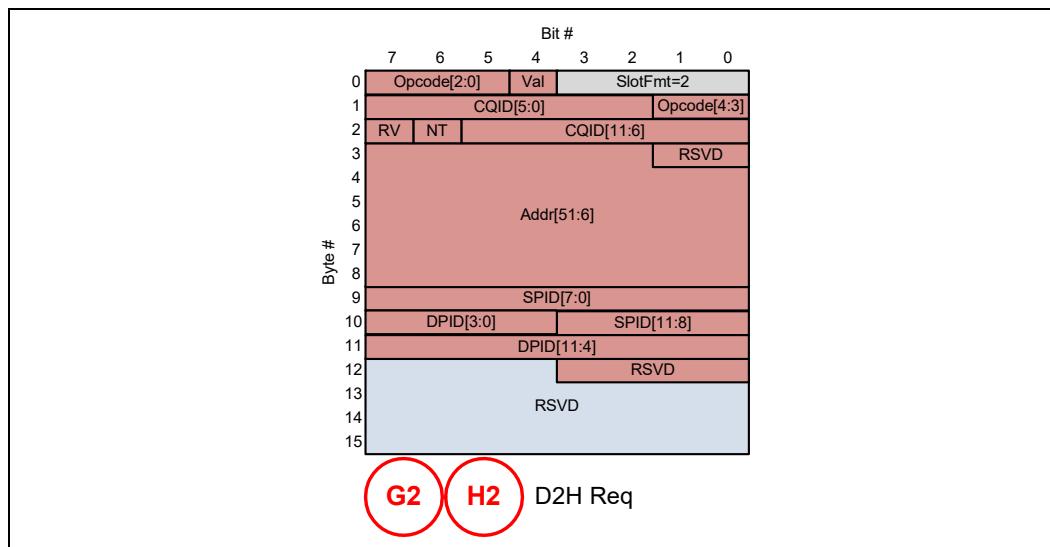


Figure 4-49. 256B Packing: G2/H2 PBR Messages



EVALUATION COPY

Figure 4-50. 256B Packing: G3/H3/HS3 HBR Messages

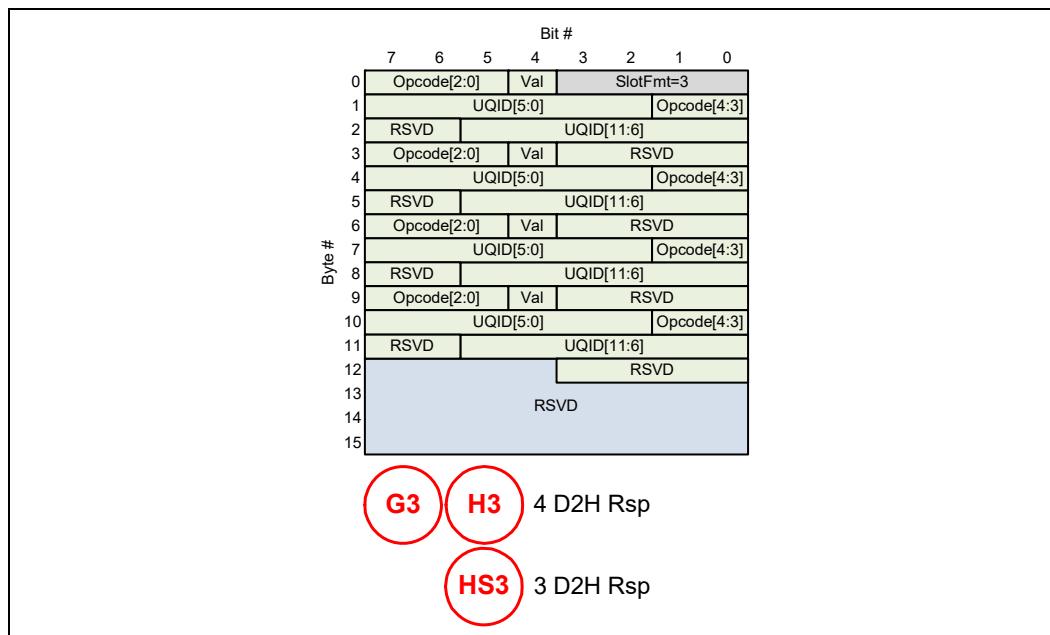
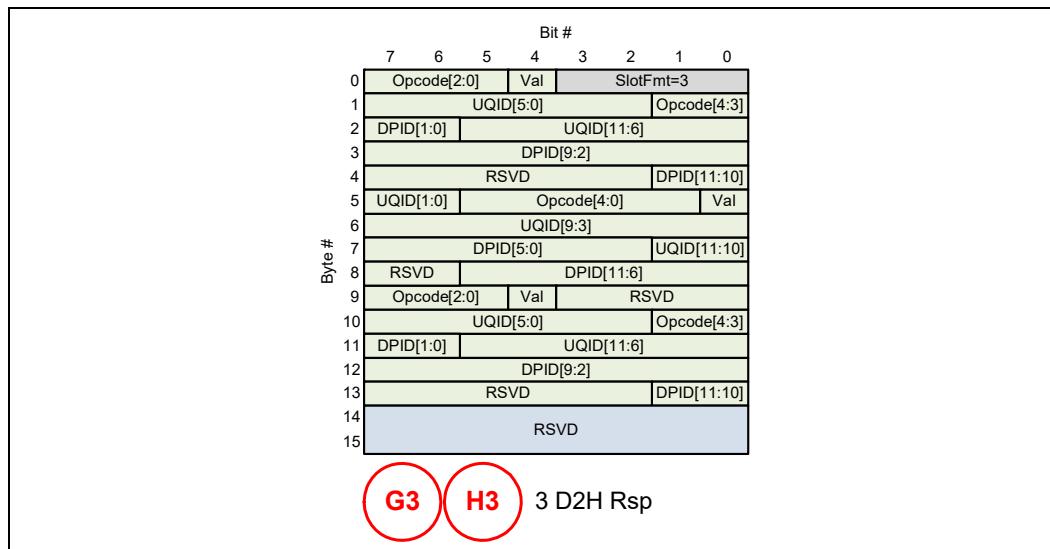


Figure 4-51. 256B Packing: G3/H3 PBR Messages



EVALUATION COPY

Figure 4-52. 256B Packing: G4/H4/HS4 HBR Messages

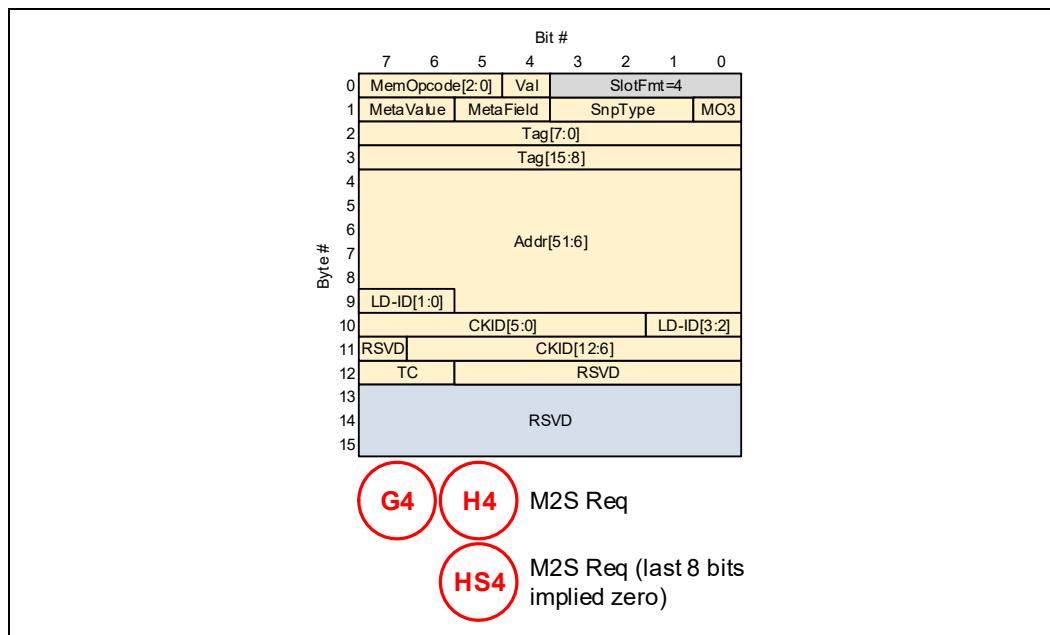
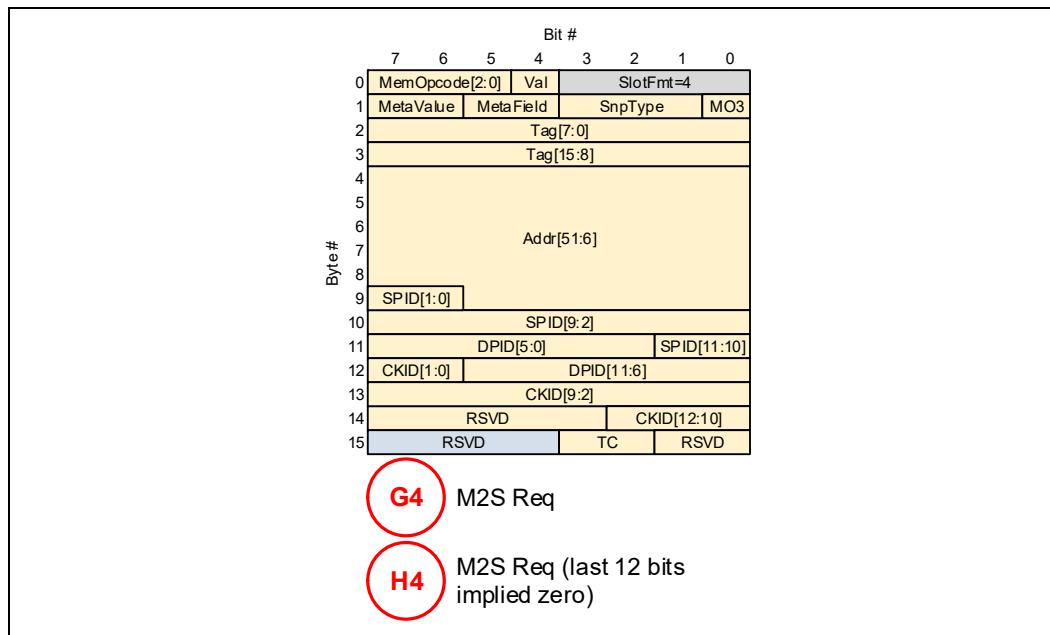


Figure 4-53. 256B Packing: G4/H4 PBR Messages



EVALUATION COPY

Figure 4-54. 256B Packing: G5/H5/HS5 HBR Messages

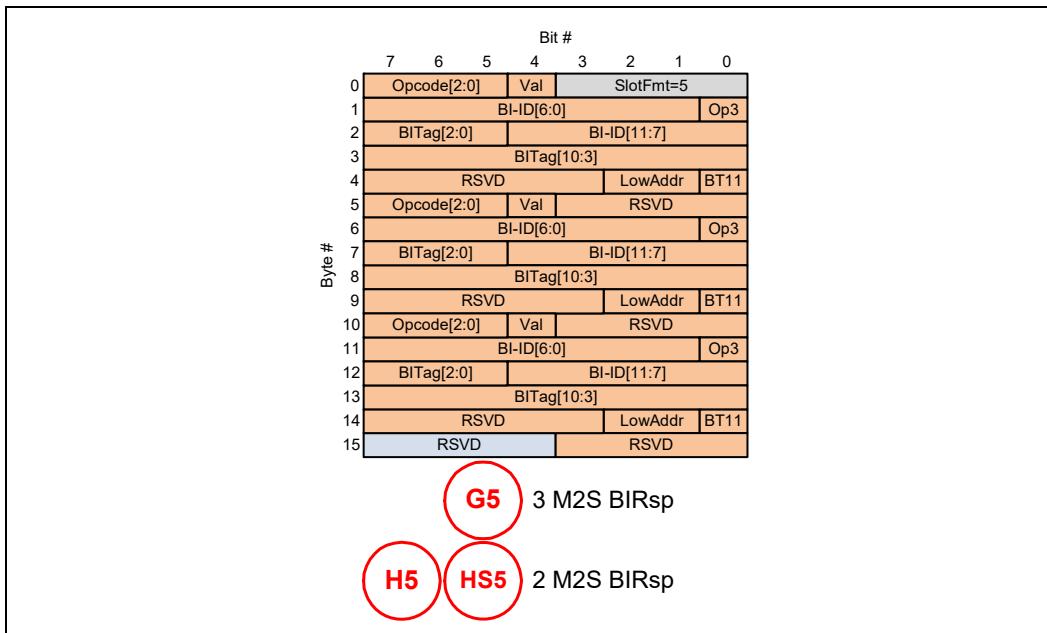
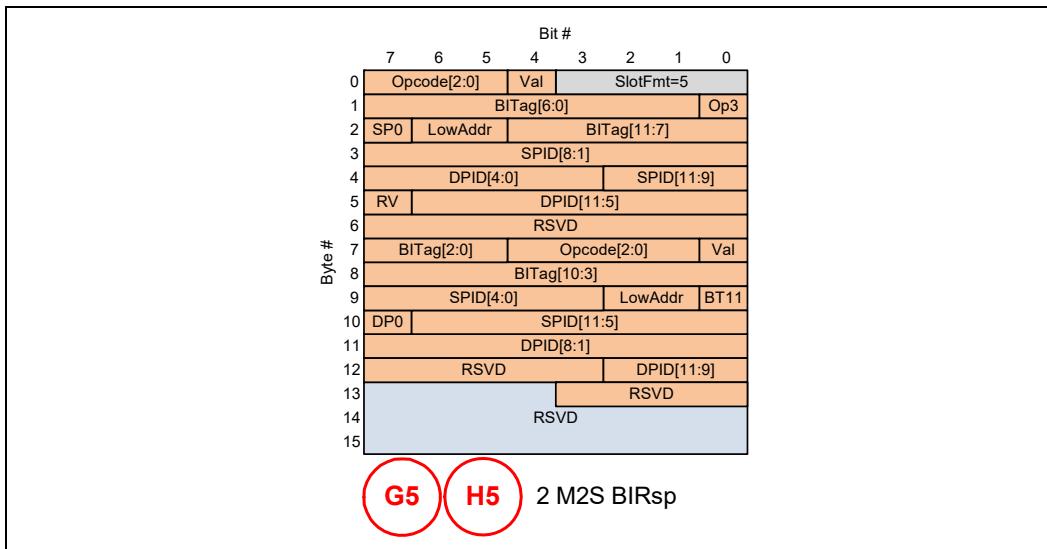


Figure 4-55. 256B Packing: G5/H5 PBR Messages



EVALUATION COPY

Figure 4-56. 256B Packing: G6/H6/HS6 HBR Messages

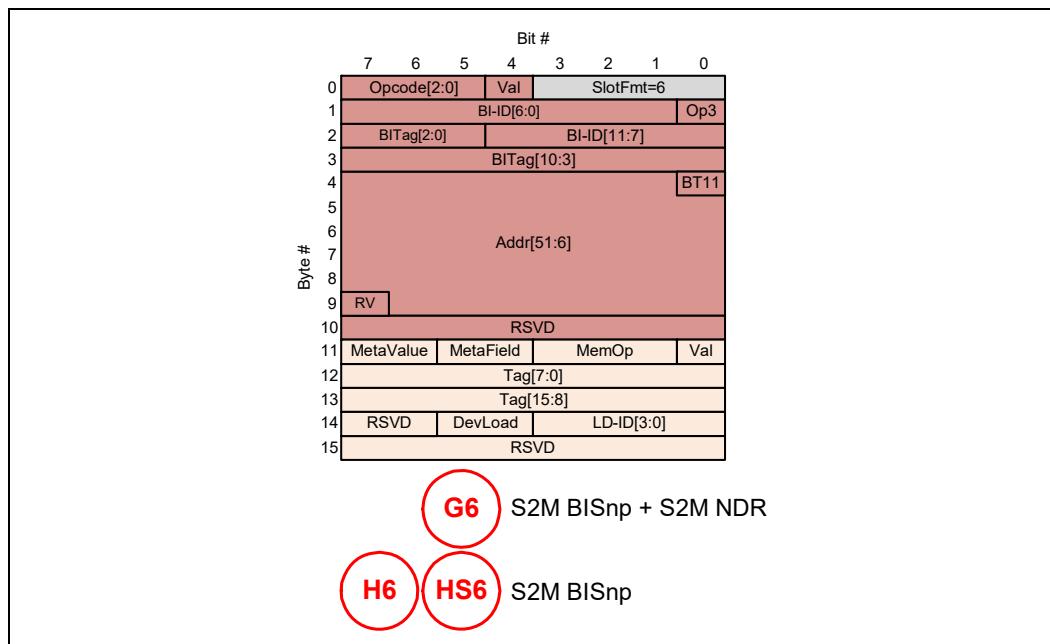
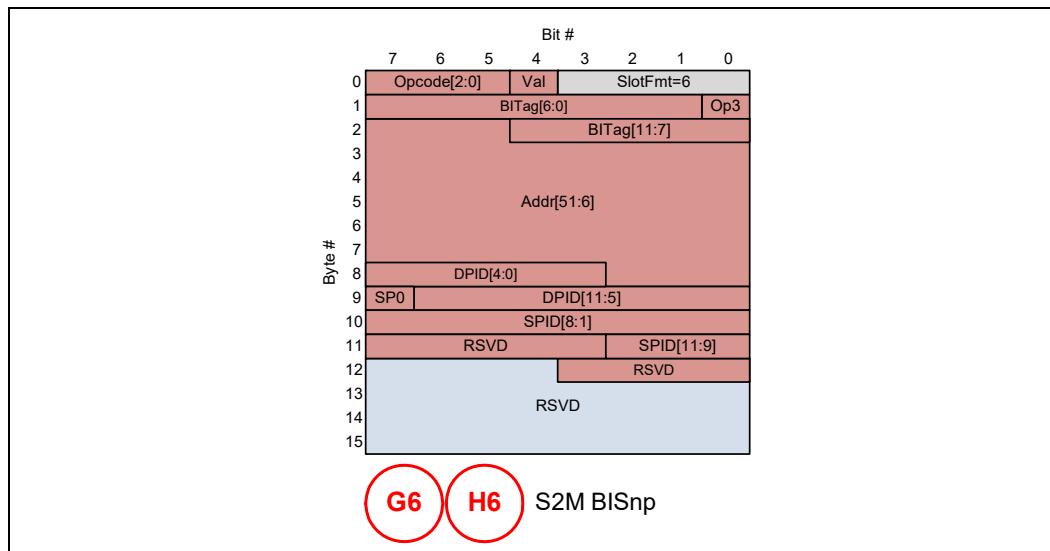


Figure 4-57. 256B Packing: G6/H6 PBR Messages



EVALUATION COPY

Figure 4-58. 256B Packing: G7/H7/HS7 HBR Messages

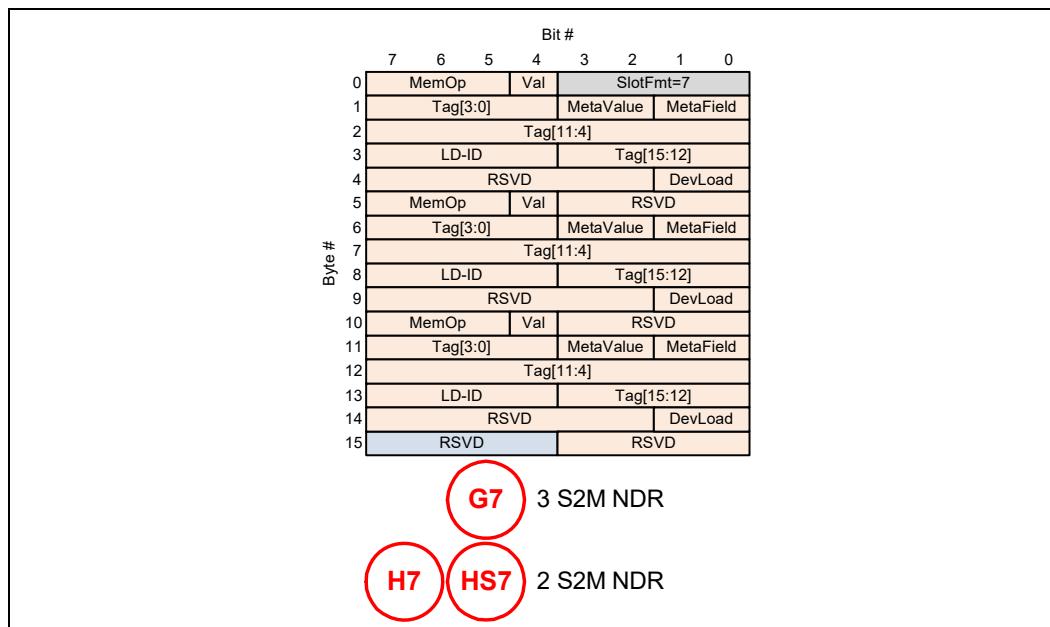
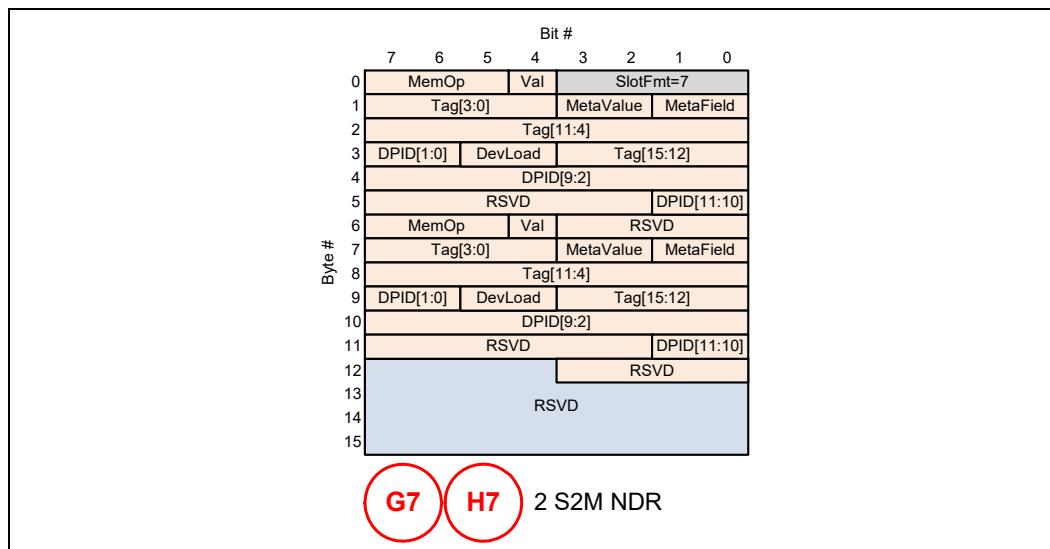


Figure 4-59. 256B Packing: G7/H7 PBR Messages



EVALUATION COPY

Figure 4-60. 256B Packing: G12/H12/HS12 HBR Messages

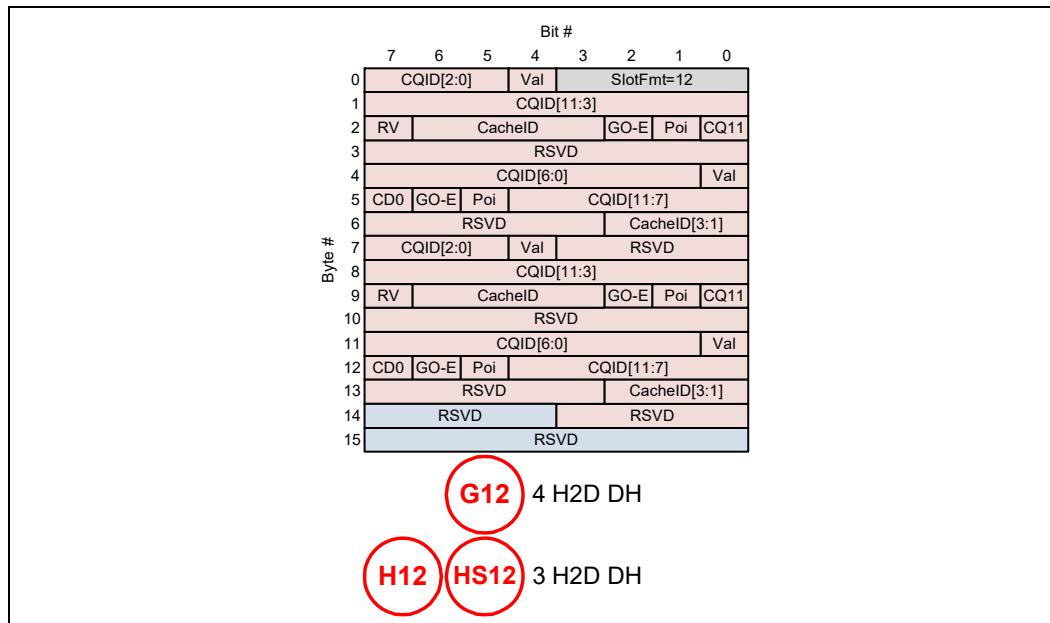
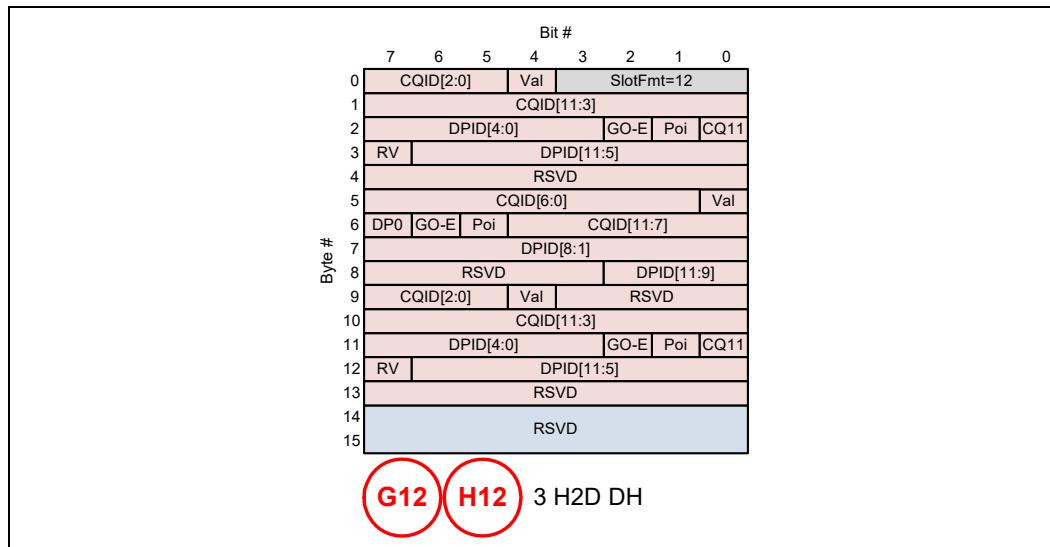


Figure 4-61. 256B Packing: G12/H12 PBR Messages



EVALUATION COPY

Figure 4-62. 256B Packing: G13/H13/HS13 HBR Messages

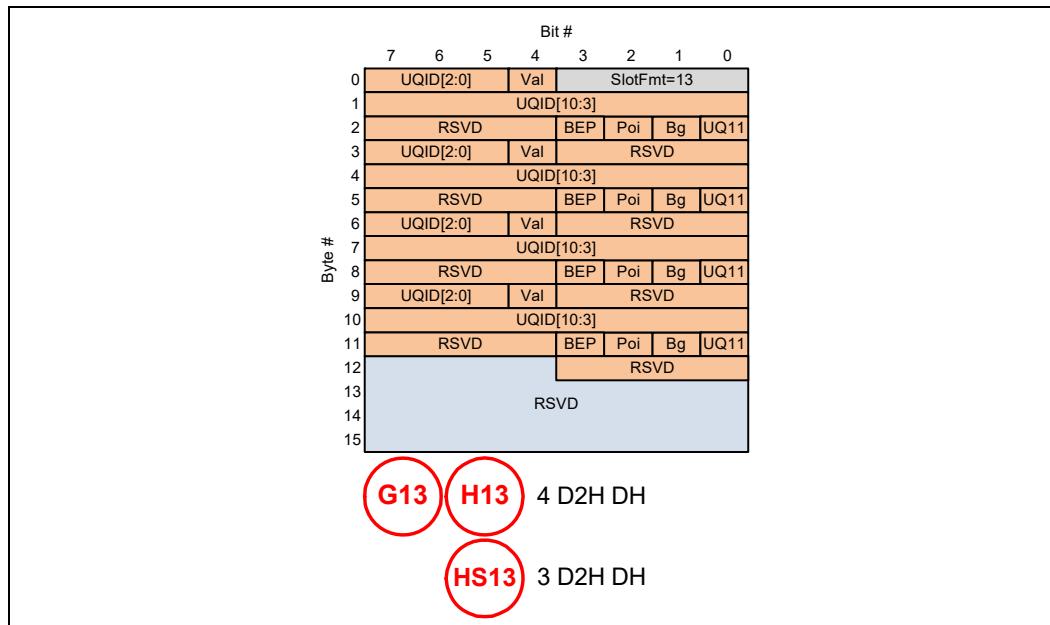
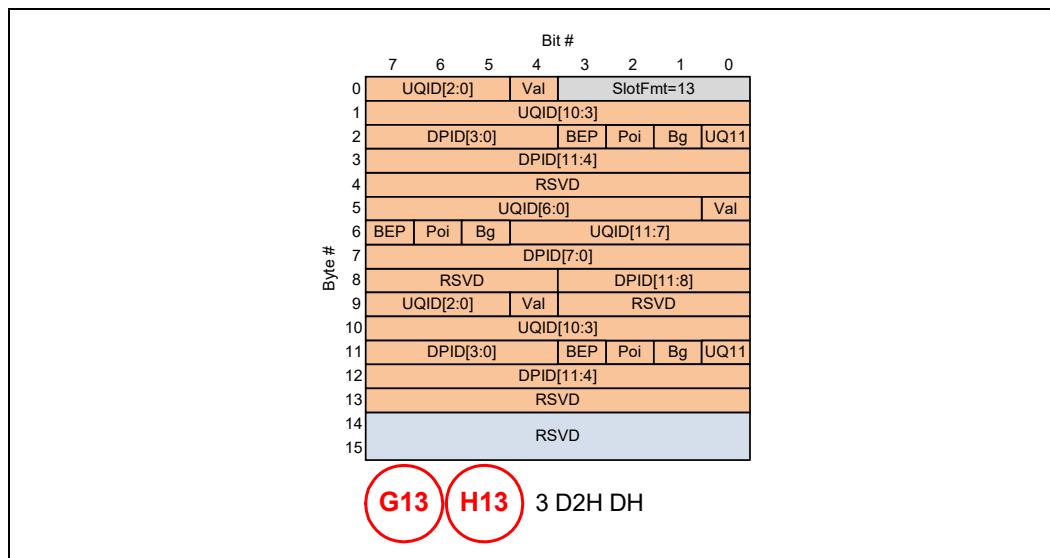


Figure 4-63. 256B Packing: G13/H13 PBR Messages



EVALUATION COPY

Figure 4-64. 256B Packing: G14/H14/HS14 HBR Messages

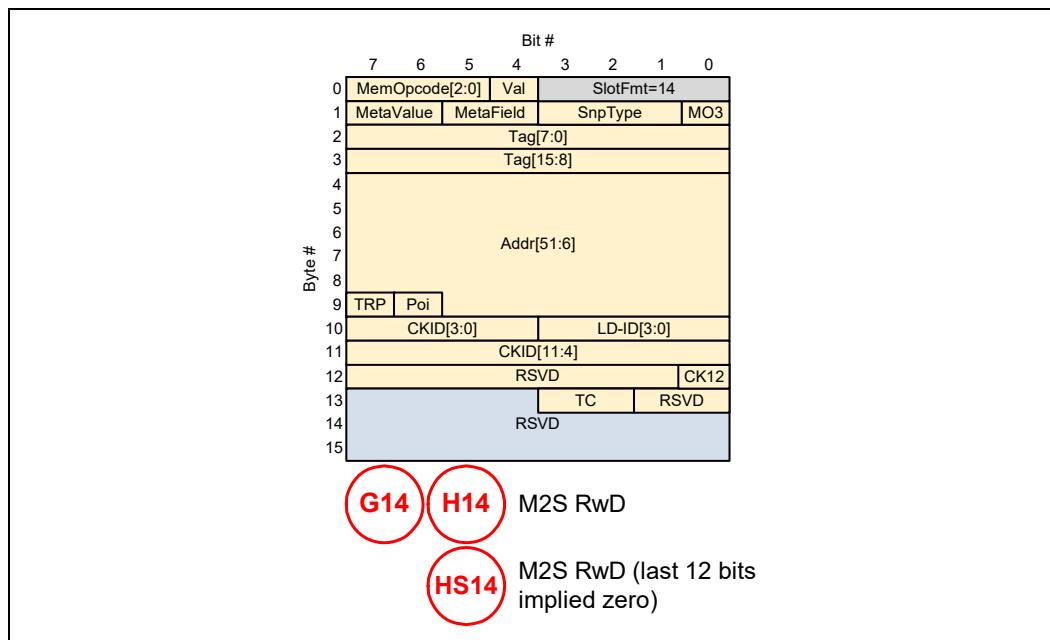
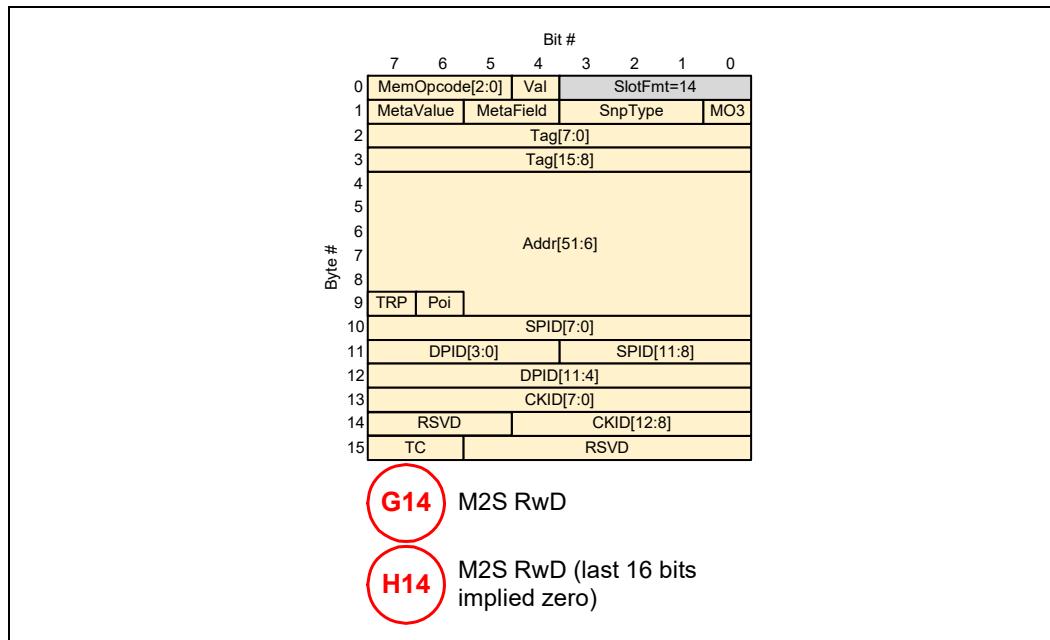


Figure 4-65. 256B Packing: G14/H14 PBR Messages



EVALUATION COPY

Figure 4-66. 256B Packing: G15/H15/HS15 HBR Messages

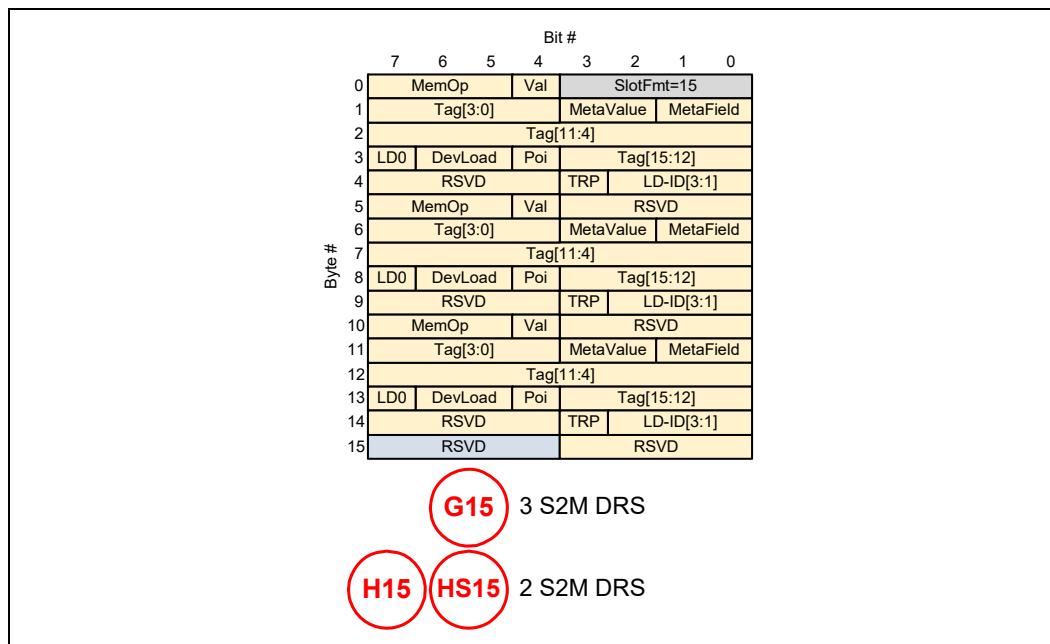
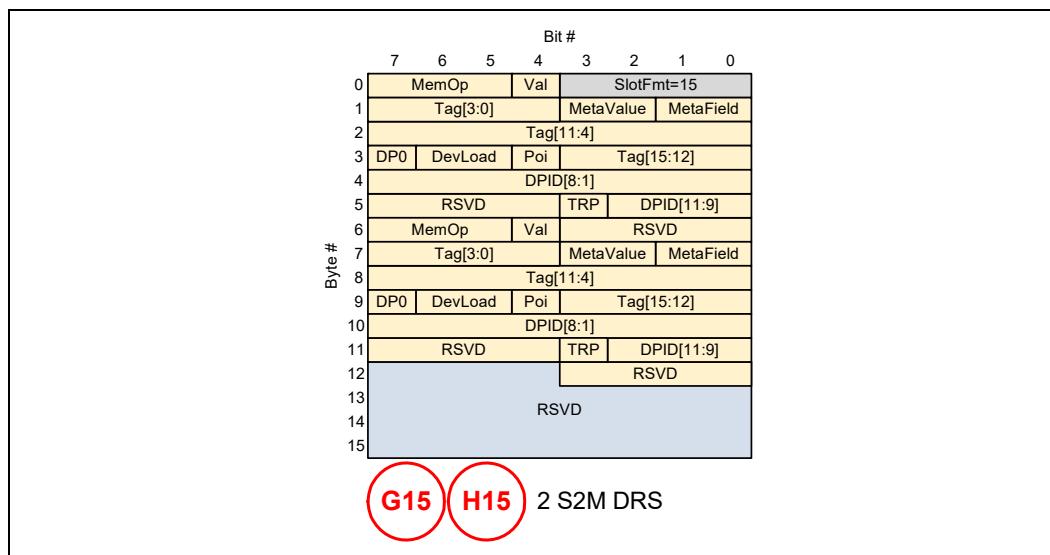


Figure 4-67. 256B Packing: G15/H15 PBR Messages



EVALUATION COPY

Figure 4-68. 256B Packing: Implicit Data

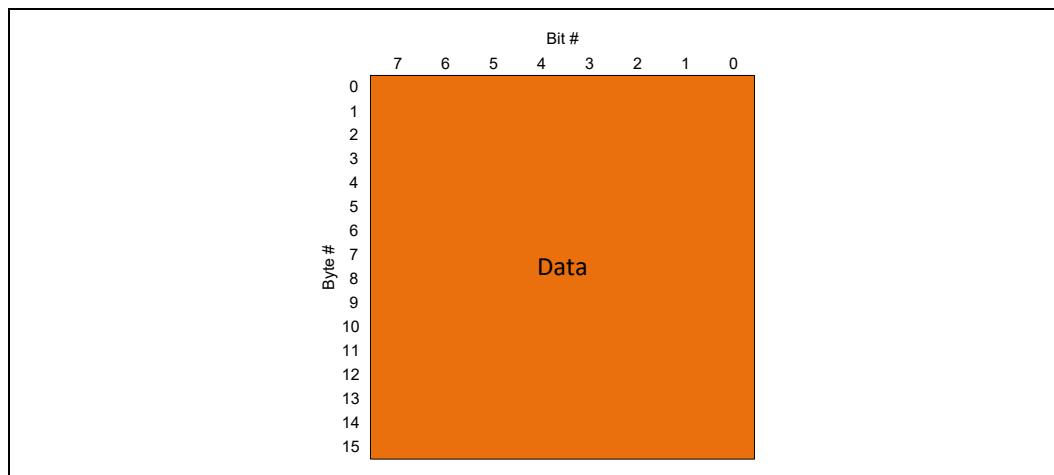


Figure 4-69. 256B Packing: Implicit Trailer RwD

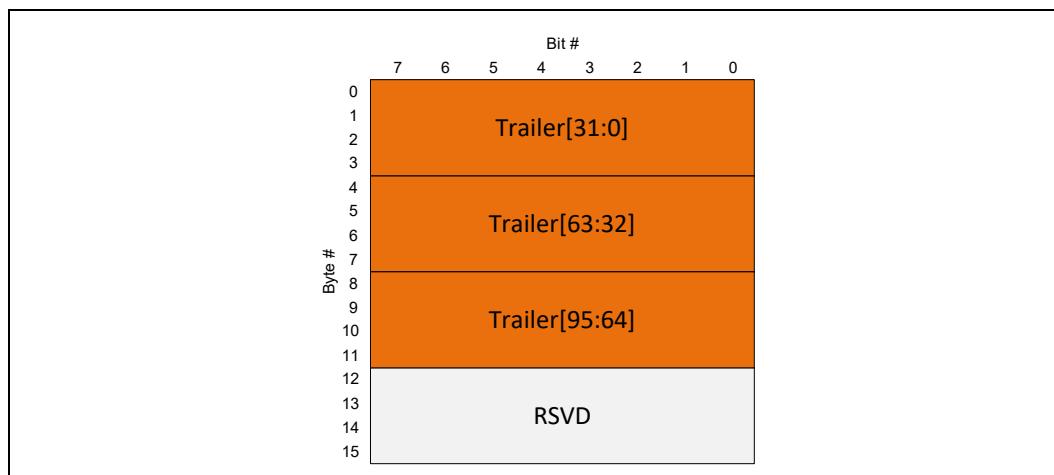


Figure 4-70. 256B Packing: Implicit Trailer DRS

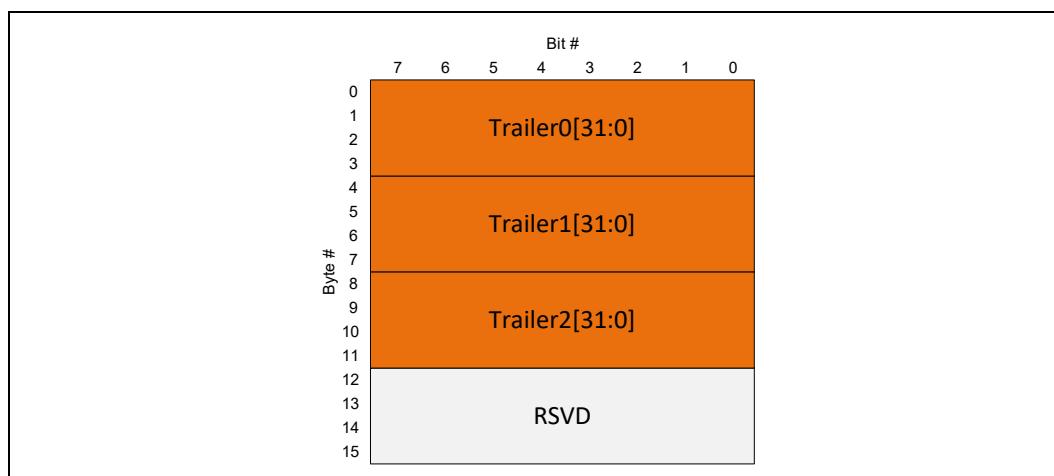
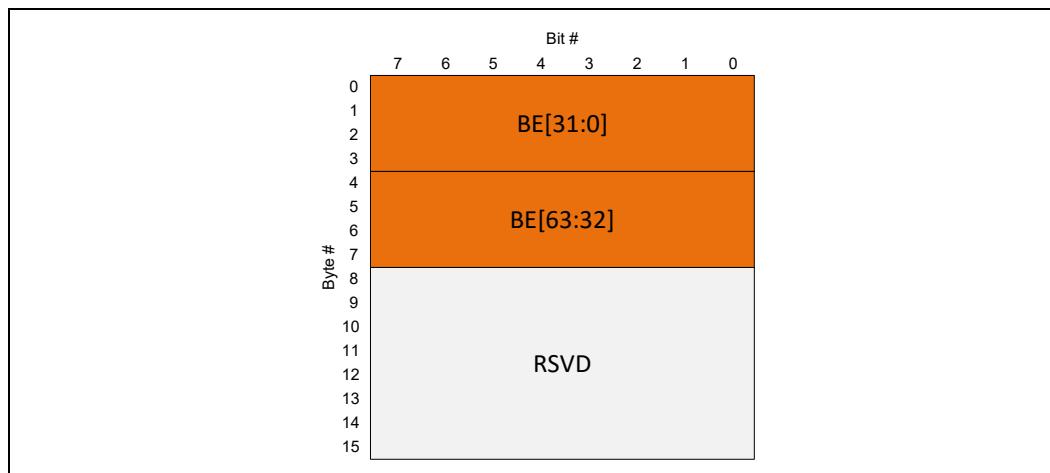


Figure 4-71. 256B Packing: Byte-Enable Trailer for D2H Data

4.3.3.1

Implicit Data Slot Decode

Data and Byte-Enable slots are implicitly known for G-slots based on prior message headers. To simplify decode of the slot format fields, SlotFmt can be used as a quick decode to know if the next 4 G-slots are data slots. Additional G-slots beyond the next 4 may also carry data depending on rollover values, the number of valid Data Headers, and BE bit within headers.

H-slots and HS-slots never carry data, so they always have an explicit 4-bit encoding defining the format.

IMPLEMENTATION NOTE

The quick decode of the current slot is used to determine whether the next 4 G-slots are data slots. The decode required is different for H/HS-slot compared to G-slots. The H/HS slots comparing SlotFmt[3:2] are equal to 11b, and for G slots reduce the compare to only SlotFmt[3] equal to 1. The difference in decode requirement is because the formats H8/HS8 indicates LLCTRL message where G8 is a reserved encoding.

More generally, the optimization for quick decode can be used to limit the logic levels required to determine whether later slots (by number) are data vs. header slots.

IMPLEMENTATION NOTE

With Link Layer data path of 64B wide, only 4 slots are processed per clock, which enables a simplified decode to reduce critical paths in the logic to determine whether a G-slot is a data slot vs. a header slot. All further decode is carried over from the previous clock cycle.

Figure 4-72 shows examples where a quick decode of the SlotFmt field can be used to determine which slots are implicit data slots. The Rollover column is the number of data slots carried over from previous flits. Because H-slots never carry data, their decode can proceed without knowledge of prior headers.

Figure 4-72. Header Slot Decode Example

Roll-Over	Current Slot #	SlotFmt MSB	G-Slots													
			H0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	1 S	D	D	D	D	?	?	?	?	?	?	?	?	?	?
0	1	1	S	D	D	D	D	?	?	?	?	?	?	?	?	?
0	2	1	S	D	D	D	D	?	?	?	?	?	?	?	?	?
1	0	1 S	R	D	D	D	D	?	?	?	?	?	?	?	?	?
4	0	0 S	R	R	R	R	?	?	?	?	?	?	?	?	?	?

Legend

- Data For Current (D)
- Data Roller-Over (R)
- Current Slot (S)
- Further Decode Needed

4.3.3.2 Trailer Decoder

A trailer is defined to be included with data carrying messages when the TRP or BEP bit is set in the header. The trailer size can vary depending on the link's capability. The base functionality requires support of the Byte-Enable use case for trailers. The Extended Metadata (EMD) use of trailers is optional. [Table 4-17](#) defines the use cases that are supported for each data-carrying channel.

Table 4-17. Trailer Size and Modes Supported per Channel

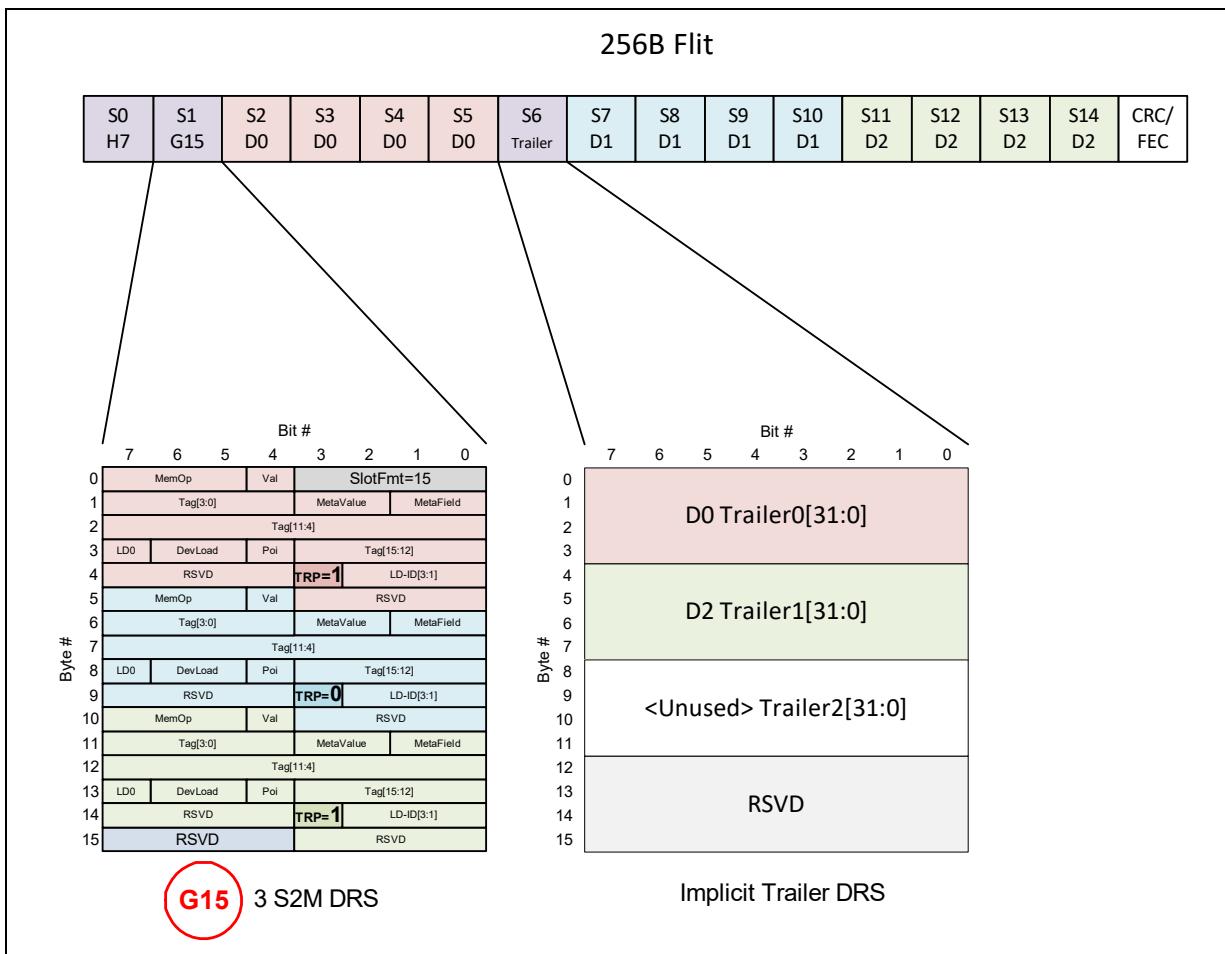
Channel	Trailer Use	Trailer Size Max
M2S Rwd	Byte-Enables (BE) and/or Extended Metadata (EMD)	96-bit if Extended Metadata is supported; otherwise, 64-bit for Byte-Enables.
S2M DRS	Extended Metadata (EMD)	96 bits max (32 bits per DRS message) when EMD is enabled; otherwise, 0-bits.
D2H Data Header (DH)	Byte-Enables	64-bit for Byte-Enables.
Other Channels	None	0 bits

For Rwd and D2H DH messages, the Trailer always follow 4 Data Slots if TRP or BEP is set for the message.

For DRS, the Trailer enables packing of up to 3 trailers together after the first 64B data transfer for Header 0 even when Header 0 does not have an associated trailer. The trailers are tightly packed for each header with TRP bit set. [Figure 4-73](#) illustrates an example case in which the 3 DRS (G15) format is sent where the 1st and 3rd headers have TRP=1 and the 2nd has TRP=0. The trailer comes after the first data transfer (D0) and the valid trailers are tightly packed.

Note:

The “tightly packed” trailer rule is for future extensibility with larger trailers where a complete set of trailers for a multi-data header will not fit into a single slot and sparse use of TRP=1 with tightly packed trailers enables higher efficiency.

Figure 4-73. DRS Trailer Slot Decode Example

4.3.4 256B Flit Packing Rules

Rules for 256B flits follow the same basic requirements as 68B flits, in terms of bit order and tightly packed rules. The tightly packed rules apply within groups of up to 4 slots together instead of across the entire flit. The groups are defined as: 0 to 3, 4 to 7, 8 to 11, and 12 to 14. Note that the final group spans only 3 slots.

- The Tx must not inject data headers in H-slots/HS-slots unless the remaining data slots to send is less than or equal to 16.

Note:

This limits the maximum count of the remaining data to be 36 (16 + 4 (new Data headers) * 5 (4 Data Slots + 1 Byte Enable slot) = 36):

- The MDH disable control bit is used to restrict the number of valid data header bits to one per slot
- If a Data Header slot format is used (G/H/HS 12 to 15) the first message must have the valid bit set

The maximum message rules are applicable on a rolling 128B group in which the groups are A="Slot 0 to 3", B="Slot 4 to 7", C="Slot 8 to 11", D="Slot 12 to 14". Extending these rules to 128B boundary enables the 256B flit slot formats to be fully

utilized. The 256B flit slots often have more messages per slot than the legacy 68B flit message rate would allow. Extending to 128B enables the use of these high message count slots while not increasing the message rate per bandwidth.

The definition of rolling is such that the groups combine into 128B rolling groups: AB (Slot 0 to 7), BC (Slot 4 to 11), CD (Slot 8 to 14), and DA (Slot 12 to 14 in current flit and Slot 0 to 3 in the following flit). The maximum message rates apply to each group. The LOpt 256B flit creates one modification to this rule such that Slot 7 is included in groups B and C: B="Slot 4 to 7" and C="Slot 7 to 11". Sub-Group C has 5 slots with this change. Note that this special case is applicable only to the maximum message rate requirement where CD group considers Slot 7 to 14 instead of Slot 8 to 14.

The maximum message rate per 128B group is defined in [Table 4-18](#), and the 68B flit message rate is included for comparison.

Note:

The term "128B group" is looking at the 128B grouping boundaries of the 256B flit. The actual number of bytes in the combined slots does vary depending on where the alignment is within the 256B flit which has other overhead like CRC, FEC, 2B Hdr.

Note:

The maximum message count was selected based on a worst-case workload requirement for steady-state message requirement in conjunction with the packing rules to achieve the most-efficient operating point. In some cases, this is 2x from the 68B message rate, which is what would be expected, but that is not true in all cases.

Table 4-18. 128B Group Maximum Message Rates

Message Type	Maximum Message Count per 128B Group	Maximum Message Count for Each 68B Flit
D2H Req	4	4
D2H Rsp	4	2
D2H Data Header (DH)	4	4
S2M BISnp	2	N/A
S2M NDR	6	2
S2M DRS-DH	3	3
H2D Req	2	2
H2D Rsp	6	4
H2D Data Header (DH)	4	4
M2S Req	4	2
M2S RwD-DH	2	1
M2S BIRsp	3	N/A

Other 68B rules that do not apply to 256B flits:

- MDH rule that requires >1 valid header per MDH. In 256B slots, only one format is provided for packing each message type, so this rule is not applicable.
- Rules related to BE do not apply because they are handled with a separate message header bit instead of a flit header bit, and because there are no special constraints placed on the number of messages when the TRP or BEP bit is set.
- 32B transfer rules don't apply because only 64B transfers are supported.

IMPLEMENTATION NOTE

Packing choices between H-slot and G-slot can have a direct impact on efficiency in many traffic patterns. Efficiency may be improved if messages that can fully utilize an H-slot (or HS-slot) are prioritized for those slots compared to messages that can better utilize a G-slot.

An example analyzed CXL.mem traffic pattern that sends steady state downstream traffic of MemRd, MemWr, and BIRsp. In this example, MemRd and MemWr can fully utilize an H-slot and do not see a benefit from being packed into a G-slot. The BIRsp packing allows more messages to fit into G-slot (3) compared to an H-slot (2), so prioritizing it for G-slot allows for improvement. In this example, we can see approximately 1.5% bandwidth improvement from prioritizing BIRsp to G-slots as compared to a simple weighted round-robin arbitration.

Prioritizing must be carefully handled to ensure that fairness is provided between each message class.

4.3.5**Credit Return**

Table 4-19 defines the 2-byte credit return encoding in the 256B flit.

Table 4-19. Credit Returned Encoding (Sheet 1 of 3)

Field	Encoding (hex)	Definition				
		Protocol	Channel	Downstream ¹	Upstream ¹	Credit Count
CRD[4:0]	00h	No credit return				
	01h	No Credit Return and the current flit is an Empty flit as defined in Section 4.3.8.1 .				
	02h-03h	Reserved				
	04h	Cache	H2D Request	X		1
	05h					4
	06h					8
	07h					12
	08h					16
	09h	D2H Request		X		1
	0Ah					4
	0Bh					8
	0Ch					12
	0Dh					16
	0Eh-13h	Reserved				
	14h	Memory	M2S Request	X	D	1
	15h					4
	16h					8
	17h					12
	18h		S2M BISnp	D	X	16
	19h					1
	1Ah					4
	1Bh					8
	1Ch					12
	1Dh					16
	1Eh-1Fh	Reserved				

Table 4-19. Credit Returned Encoding (Sheet 2 of 3)

Field	Encoding (hex)	Definition						
		Protocol	Channel	Downstream ¹	Upstream ¹	Credit Count		
CRD[9:5]	00h	No credit return			0			
	01h-03h	Reserved						
	04h	Cache	H2D Data	X		1		
	05h					4		
	06h					8		
	07h					12		
	08h					16		
	09h		D2H Data	X		1		
	0Ah					4		
	0Bh					8		
	0Ch					12		
	0Dh					16		
	0Eh-13h	Reserved						
	14h	Memory	M2S RwD	X	D	1		
	15h					4		
	16h					8		
	17h					12		
	18h		S2M DRS	D	X	16		
	19h					1		
	1Ah					4		
	1Bh					8		
	1Ch							
	1Dh							
	1Eh-1Fh	Reserved						

Table 4-19. Credit Returned Encoding (Sheet 3 of 3)

Field	Encoding (hex)	Definition					
		Protocol	Channel	Downstream ¹	Upstream ¹	Credit Count	
CRD[14:10]	00h	No credit return			0		
	01h-03h	Reserved					
	04h	Cache	H2D Rsp	X		1	
	05h					4	
	06h					8	
	07h					12	
	08h					16	
	09h					1	
	0Ah	Memory	D2H Rsp	X		4	
	0Bh					8	
	0Ch					12	
	0Dh					16	
	0Eh-13h	Reserved					
	14h	M2S BIRsp	X	D	1		
	15h				4		
	16h				8		
	17h				12		
	18h	S2M NDR	D	X	16		
	19h				1		
	1Ah				4		
	1Bh				8		
	1Ch				12		
	1Dh				16		
	1Eh-1Fh			Reserved			
CRD[15]	Reserved						

1. D = Credit channel mapping is applicable only on a Direct P2P CXL.mem link between a device and the switch Downstream Port to which it is attached.

4.3.6

Link Layer Control Messages

In 256B Flit mode, control messages are encoded using the H8 format and sometimes using the HS8 format. Figure 4-74 captures the 256B packing for LLCTRL messages. H8 provides 108 bits to be used to encode the control message after accounting for 4-bit slot format encoding. 8 bits are used to encode LLCTRL/SubType, and 4 bits are kept as reserved, with a 96-bit payload. For HS8, it is limited to 2 bytes less, which cuts the available payload to 80 bits. Table 4-20 captures the defined control messages. In almost all cases, the remaining slots after the control message are considered to be reserved (i.e., cleared to all 0s) and do not carry any protocol information. The exception case is IDE.MAC, which allows for protocol messages in the other slots within the flit. For messages that are injected in the HS slot, the slots prior to the HS slot may carry protocol information but the slots after the HS slot are reserved.

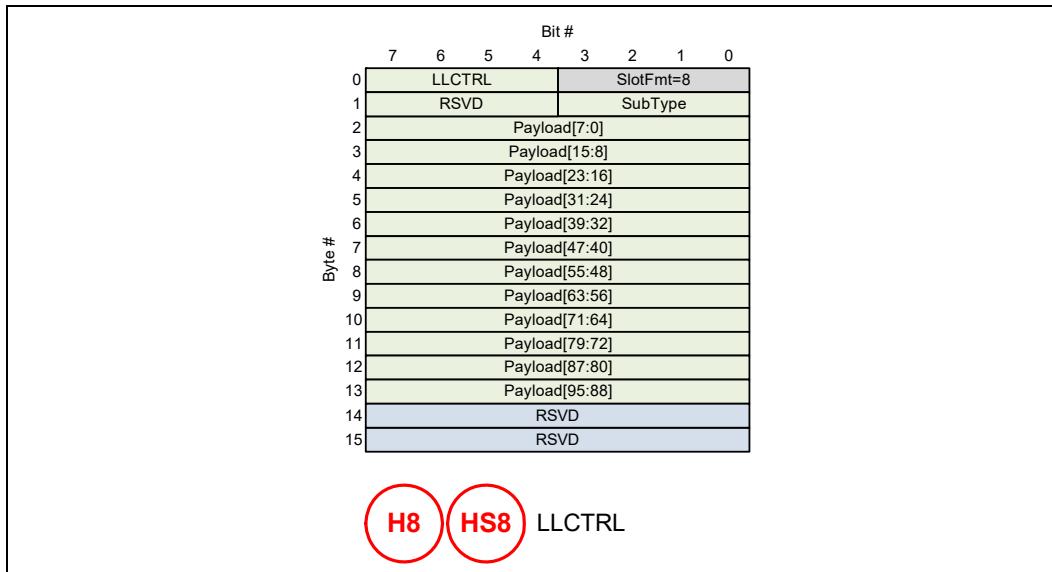
Table 4-20. 256B Flit Mode Control Message Details

Flit Type	LLCTRL	SubType	SubType Description	Payload	Payload Description	Remaining Slots are Reserved? ¹
IDE ²	0010b	0000b	IDE.Idle	95:0	Payload RSVD Message sent as part of IDE flows to pad sequences with idle flits. See 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.	
		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
		0100b	IDE.Stop	95:0	Payload RSVD. Message used to disable IDE. See Chapter 11.0 for details on the use of this message.	Yes
		Others	RSVD	95:0	RSVD	
In-band Error ³	0011b	0000b	Viral	15:0	Viral LD-ID Vector[15:0]: Included for MLD links to indicate which LD-ID is impacted by viral. Bit 0 of the vector encodes LD-ID=0, bit 1 is LD-ID=1, etc. Field is treated as Reserved for ports that do not support LD-ID.	Yes
				79:16	RSVD	
				95:80	RSVD (these bits do not exist in HS format).	
		0001b	Poison	3:0	Poison Message Offset is the encoding of which of the active or upcoming messages will have poison applied. There can be up to 8 active Data carrying messages and up to 4 new data carrying messages where the poison can be applied. <ul style="list-style-type: none"> • 0h = Poison the currently active data message • 1h = Poison the message 1 after the current data message • ... • 7h = Poison the message 7 after the current data message See Section 4.3.6.3 for additional details.	
					79:4	
					95:80	
					Others	
					RSVD	
					RSVD	
INIT ²	1100b	1000b	INIT.Param	0	Direct P2P CXL.mem-capable port. Credits for the channels enabled in this feature are not returned unless both sides support it.	Yes
				95:1	RSVD	
		Others	RSVD	95:0	RSVD	
Reserved	Others	<all>	RSVD	95:0	RSVD	

1. 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).

2. Supported only in H-slot.

3. Supported in either H-slot or HS-slot.

Figure 4-74. 256B Packing: H8/HS8 Link Layer Control Message Slot Format

4.3.6.1

Link Layer Initialization

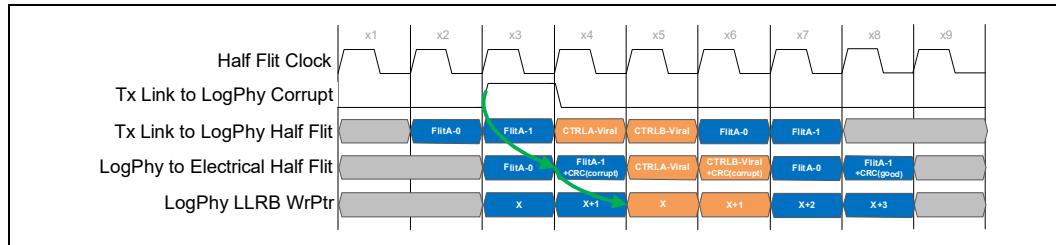
After initial link training (from Link Down), the link layer must send and receive the INIT.Param flit before beginning normal operation. After reaching normal operation, the Link Layer will start by returning all possible credits using the standard credit return mechanism. Normal operation is also required before sending other control messages (IDE, In-band Error).

4.3.6.2

Viral Injection and Containment

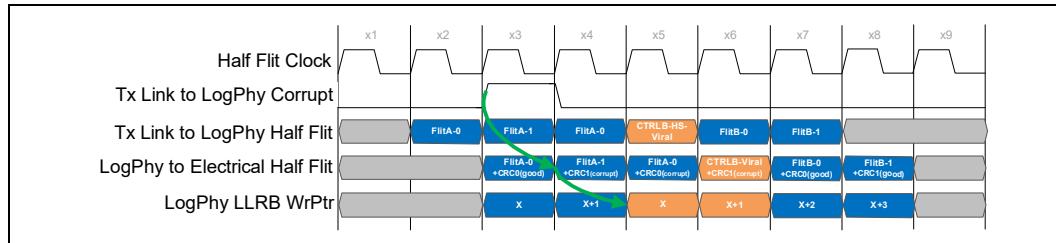
The Viral control flit is injected as soon as possible after the viral condition is observed. For cases in which the error that triggers Viral can impact the current flit, the link layer should signal to the physical layer to stop the currently partially sent CXL.cachemem flit (Flit 0) by injection of a CRC/FEC corruption that ensures a retry condition (note that this does not directly impact CXL.io flits or flits that are being replayed from the Physical Layer retry buffer). Then the Logical Physical Layer will also remove that flit (flit 0) from the retry buffer and replace it with the Viral control flit (flit 1) that must be sent immediately by the link layer. The Link Layer must also resend the flit that was corrupted (flit 0) after the viral flit. Figure 4-75 captures an example of a Link Layer to Logical Physical Layer (LogPhy) with a half-flit interface where CRC is corrupted and Viral is injected. At Cycle "x3", it is signaled to corrupt the current flit (FlitA). At cycle "x4", the CRC(bad) is indicated and the link layer starts sending the Viral control. In Cycle "x5", the retry buffer pointer (WrPtr) is stepped back to ensure the FlitA is removed from the retry buffer and then replaced with the Viral flit sent from the link layer. At Cycle "x6", the CTRL-Viral flit is also sent with corrupted CRC to ensure the full retry flow (disallowing the single flit retry). Also starting at cycle "x6", FlitA is resent from the link layer and forwarded on normally through the LogPhy and retry buffer. FlitA is identical to the flit started in Cycle "x2".

With link IDE enabled this flow works the same and FlitA is retransmitted with the same encryption mask and without altering the integrity state. The control message is not included in link IDE and thus does not impact the IDE requirements.

Figure 4-75. Viral Error Message Injection Standard 256B Flit

The Error signaling with CRC corruption flow requires special handling for LOpt flits. If the link layer is in the first 128B phase of the flit, the flow is identical to Standard Flit mode. However, if the link layer is in the second phase of the 128B flit (when the first 128B was committed), then the flit corruption is guaranteed only on the second half, but the Physical Layer will remove the entire flit from the retry buffer. The link layer will send the first 128B identically to what was sent before, and then the link layer will inject the Viral control message in Slot 8 (HS-format) and Slots 9-14 are considered RSVD and normal operation continues in the next flit. Any data slots and other message encodings are continued in the next flit. [Figure 4-76](#) captures the unique case for the LOpt flit. The difference from the standard 256B flit is in three areas of this flow. First at Cycle "x4", the link layer resends FlitA-0 because this half of the flit may have already been consumed. Then at Cycle "x5", in the second-half of that flit, the link layer injects the control message for Viral (after the final portion of Slot 7). At Cycle "x6", the second half of the original flit (starting with Slot 8) is repacked in the first half of FlitB following the standard packing rules.

This flow cannot be supported with link IDE, thus any error containment must either be detected sufficiently early to corrupt CRC in the first half of the flit or must be injected in the second half without corrupting the CRC.

Figure 4-76. Viral Error Message Injection LOpt 256B Flit

4.3.6.3 Late Poison

Poison can be injected at a point after the header was sent by injecting an Error Control message with the Poison sub-type. The message includes a payload encoding that indicates the data message offset at which the poison applies. It is possible that any one of up to 8 active messages can be targeted. The encoding is an offset that is relative to the data that is yet to be sent, including the currently active data transmission. The poison applies to the entire message payload, just as it does when poison is included in the message header.

If a message is currently active, but not all data slots have been sent, the offset value of zero applies to that message. If a receiver implementation uses "wormhole switching" techniques, where data is forwarded through the on-die fabric before all the data has arrived, then it is possible that data already sent may be consumed. In this case, the only guarantee is that the poison is applied to the remaining data after the poison control message. The following are examples of how this would apply in specific cases.

Example 1:

- Flit 1 - 1st 3 slots of data Message A in Slots 12 to 14.
- Flit 2 - In-band error poison message in Slot 0 with a poison message offset value of 0.
- Flit 3 - 4th slot of data Message A in Slot 1 and data Message B in Slots 2 to 5.
- The poison control message applies to Message A, but is only guaranteed to be applied to the final data slot of that message. But it may also be applied to the entire message.

Example 2:

- Flit 1 - 4 slots of data Message A in Slots 11 to 14 where the message header has Byte-Enables Present (BEP) or Trailer Present (TRP) bit set.
- Flit 2 - In-band error poison message in Slot 0 with a poison message offset value of 0.
- Flit 3 - The Trailer (e.g., Byte enables) for data Message A in Slot 1 and data Message B in Slots 2 to 5.
- The poison control message applies to Message A, but is not guaranteed to be applied to any of the data because it was already sent. Note that the use of Trailer in this example could be any supported trailer (e.g., Extended Meta Data and/or Byte-Enables).

To inject poison on data that is scheduled to be sent in the current flit, and no H-slot/HS-slot exists to interrupt the data transmission, the same CRC corruption flows as described in [Section 4.3.6.2, “Viral Injection and Containment,”](#) are used.

4.3.6.4**Link Integrity and Data Encryption (IDE)**

For the IDE flow, see [Chapter 11.0](#).

4.3.7**Credit Return Forcing**

To avoid starvation, credit return rules ensure that Credits are sent even when there are no protocol messages pending. In 68B Flit mode, this uses a special control message called LLCRD (its algorithm is described in [Section 4.2.8.2](#)). For 256B Flit mode, the same underlying algorithm for forcing is used, but with the following changes:

- Ack forcing is not applicable with 256B flit.
- With 256B flits, CRD is part of standard flit definition, so no special control message is required.
- There is a packing method described in [Section 4.3.8](#). When implementing this algorithm, the end of the flit is tagged as empty if no valid messages or Credit return is included. With this flit packing method, the flit should return a nonzero credit value only if there are other valid messages sent unless the credit forcing algorithm has triggered.
- No requirement to prioritize protocol messages vs. CRD because they are both part of 256B flits.

4.3.8**Latency Optimizations**

To get the best latency characteristics, the 256B flit is expected to be sent with a link layer implementing 64B or 128B pipeline and the Latency-Optimized flit (which is optional). The basic reasoning for these features is self-evident.

Additional latency optimization is possible sending idle slot scheduling of flits to the ARB/MUX which avoids needing to wait for the next start of flit alignment. There are trade-offs between CXL.io vs. empty slots being scheduled, so overall bandwidth should be considered.

IMPLEMENTATION NOTE

A case to consider for idle slot scheduling is with a Link Layer pipeline of 64B in which idle slots allow late-arriving messages to be packed later in the flit. By doing this, the Transmitter can avoid stalls by starting the flit with empty slots. An example case of this is with a x4 port in which a message shows and just misses the first 64B of the flit. In this case, it is necessary to wait an additional 192B before sending the message because the ARB/MUX is injecting an empty flit or a flit from CXL.io. In this example, the observed additional latency in x4 is 6 ns (192 bytes/x4 * 8 bits/byte / 64 GT/s).

4.3.8.1

Empty Flit

As part of the latency optimizations described in this chapter, the Link Layer needs to include a way to indicate that the current flit does not have messages or CRD information. The definition of Empty in this context is that the entire flit can be dropped without side effects in the Link Layer:

- No Data Slots are sent
- No Valid bits are set in any protocol slots
- No control message is sent
- No Credits are returned in the CRD field

A special encoding of the CRD field provides this such that CRD[4:0] = 01h as captured in [Table 4-19](#).

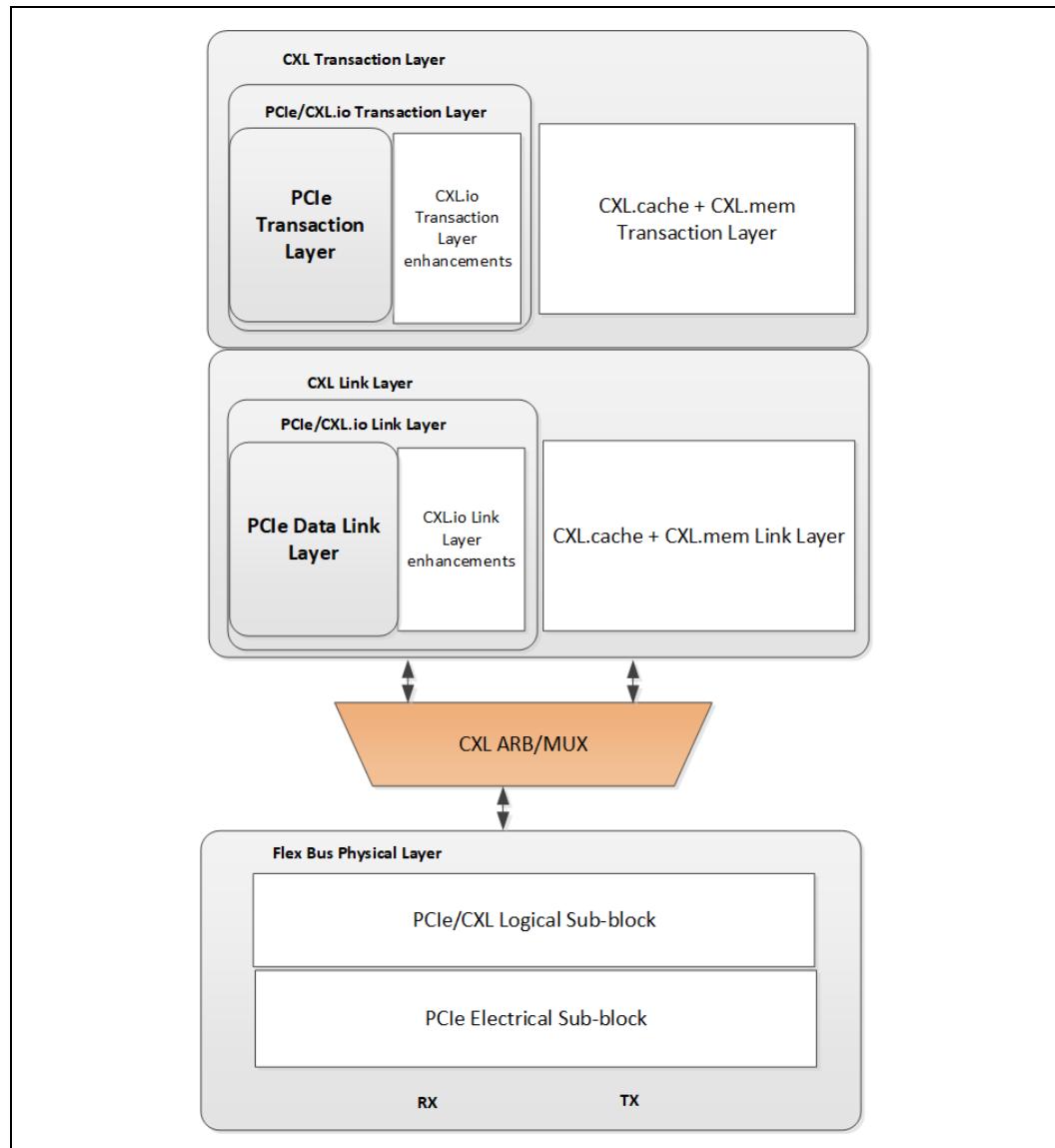
When IDE is enabled, the Empty Encoding shall not be used as all protocol flits are required to be fully processed.

§ §

5.0 CXL ARB/MUX

Figure 5-1 shows where the CXL ARB/MUX exists in the Flex Bus layered hierarchy. The ARB/MUX provides dynamic muxing of the CXL.io and CXL.cache+mem link layer control and data signals to interface with the Flex Bus physical layer.

Figure 5-1. Flex Bus Layers - CXL ARB/MUX Highlighted



In the transmit direction, the ARB/MUX arbitrates between requests from the CXL link layers and multiplexes the data. It also processes power state transition requests from the link layers: resolving them to a single request to forward to the physical layer, maintaining virtual link state machines (vLSMs) for each link layer interface, and generating ARB/MUX link management packets (ALMPs) to communicate the power state transition requests across the link on behalf of each link layer. See [Section 10.3](#), [Section 10.4](#), and [Section 10.5](#) for more details on how the ALMPs are utilized in the overall flow for power state transitions. In PCIe* mode, the ARB/MUX is bypassed, and thus ALMP generation by the ARB/MUX is disabled.

In the receive direction, the ARB/MUX determines the protocol associated with the CXL flit and forwards the flit to the appropriate link layer. It also processes the ALMPs, participating in any required handshakes and updating its vLSMs as appropriate.

For 256B Flit mode, the replay buffer is part of the Physical Layer. ALMPs have a different flit format than in 68B Flit mode, and are protected by forward error correction (FEC) and cyclic redundancy check (CRC). They must also be allocated to the replay buffer in the Physical Layer and follow the replay sequence protocols. Hence, they are guaranteed to be delivered to the remote ARB/MUX error free.

5.1

vLSM States

The ARB/MUX maintains vLSMs for each CXL link layer it interfaces with, transitioning the state based on power state transition requests it receives from the local link layer or from the remote ARB/MUX on behalf of a remote link layer. [Table 5-1](#) lists the different possible states for the vLSMs. PM States and Retrain are virtual states that can differ across interfaces (CXL.io, CXL.cache, and CXL.mem); however, all other states such as LinkReset, LinkDisable, and LinkError are forwarded to the Link Layer and are therefore synchronized across interfaces.

Table 5-1. vLSM States Maintained per Link Layer Interface

vLSM State	Description
Reset	Power-on default state during which initialization occurs
Active	Normal operational state
Active.PMNAK	Substate of Active to indicate unsuccessful ALMP negotiation of PM entry. This is not a state requested by the Link Layer. It is applicable only for Upstream Ports. It is not applicable for 68B Flit mode.
L1.0	Power savings state, from which the link can enter Active via Retrain (maps to PCIe L1)
L1.1	Power savings state, from which the link can enter Active via Retrain (reserved for future use)
L1.2	Power savings state, from which the link can enter Active via Retrain (reserved for future use)
L1.3	Power savings state, from which the link can enter Active via Retrain (reserved for future use)
DAPM	Deepest Allowable PM State (not a resolved state; a request that resolves to an L1 substate)
SLEEP_L2	Power savings state, from which the link must go through Reset to reach Active
LinkReset	Reset propagation state resulting from software-initiated or hardware-initiated reset
LinkError	Link Error state due to hardware-detected errors that cannot be corrected through link recovery (e.g., uncorrectable internal errors or surprise link down)
LinkDisable	Software-controlled link disable state
Retrain	Transitory state that transitions to Active

Note: When the Physical Layer LTSSM enters Hot Reset or Disabled state, that state is communicated to all link layers as LinkReset or LinkDisable, respectively. No ALMPs are exchanged, regardless of who requested them, for these transitions. LinkError must take the LTSSM to Detect or Disabled. For example, it is permitted to map CXL.io Downstream Port Containment to LinkError (when the LTSSM is in Disabled state).

The ARB/MUX looks at the state of each vLSM to resolve to a single state request to forward to the physical layer as specified in [Table 5-2](#). For example, if the current vLSM[0] state is L1.0 (row = L1.0) and the current vLSM[1] state is Active (column = Active), then the resolved request from the ARB/MUX to the Physical layer will be Active.

Table 5-2. ARB/MUX Multiple vLSM Resolution Table

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

Based on the requested state from one or more of the Link Layers, ARB/MUX will change the state request to the physical layer for the desired link state.

For implementations in which the Link Layers support directing the ARB/MUX to LinkReset or LinkError or LinkDisable, the ARB/MUX must unconditionally propagate these requests from the requesting Link Layer to the Physical Layer; this takes priority over [Table 5-2](#).

[Table 5-3](#) describes the conditions under which a vLSM transitions from one state to the next. A transition to the next state occurs after all the steps in the trigger conditions column are complete. Some of the trigger conditions are sequential and indicate a series of actions from multiple sources. For example, on the transition from Active to L1.x state on an Upstream Port, the state transition will not occur until the vLSM has received a request to enter L1.x from the Link Layer followed by the vLSM sending a Request ALMP{L1.x} to the remote vLSM. Next, the vLSM must wait to receive a Status ALMP{L1.x} from the remote vLSM. Once all these conditions are met in sequence, the vLSM will transition to the L1.x state as requested. Certain trigger conditions are applicable only when operating in 68B Flit mode, and these are highlighted in the table "For 68B Flit mode only".

Table 5-3. ARB/MUX State Transition Table (Sheet 1 of 2)

Current vLSM State	Next State	Upstream Port Trigger Condition	Downstream Port Trigger Condition
Active	L1.x	Upon receiving a Request to enter L1.x from Link Layer, the ARB/MUX must initiate a Request ALMP{L1.x} and receive a Status ALMP{L1.x} from the remote vLSM	Upon receiving a Request to enter L1.x from Link Layer and receiving a Request ALMP{L1.x} from the Remote vLSM, the ARB/MUX must send Status ALMP{L1.x} to the remote vLSM
	L2	Upon receiving a Request to enter L2 from Link Layer the ARB/MUX must initiate a Request ALMP{L2} and receive a Status ALMP{L2} from the remote vLSM	Upon receiving a Request to enter L2 from Link Layer and receiving a Request ALMP{L2} from the Remote vLSM the ARB/MUX must send Status ALMP{L2} to the remote vLSM
	Active.PMNAK	For 256B Flit mode: Upon receiving a PMNAK ALMP from the Downstream Port ARB/MUX. This arc is not applicable for 68B Flit mode.	N/A
Active.PMNAK	Active	For 256B Flit mode: Upon receiving a request to enter Active from the Link Layer (see Section 5.1.2.4.2.2). This arc is not applicable for 68B Flit mode.	N/A
L1.x	Retrain	Upon receiving an ALMP Active request from remote ARB/MUX	Upon receiving an ALMP Active request from remote ARB/MUX
Active	Retrain	For 68B Flit mode only: Any of the following two conditions are met: 1) Physical Layer LTSSM enters Recovery. 2) Physical Layer transitions from Recovery to L0 and State Status ALMP synchronization for Recovery exit resolves to Retrain (see Section 5.1.2.3). For 256B Flit mode, this arc is not applicable since the replay buffer is moved to logPHY, there is no reason to expose Active to Retrain arc to protocol layer vLSMs.	For 68B Flit mode only: Physical Layer LTSSM enters Recovery. For 256B Flit mode, this arc is not applicable since the replay buffer is moved to logPHY, there is no reason to expose Active to Retrain arc to protocol layer vLSMs.
Retrain	Active	Link Layer is requesting Active and any of the following conditions are met: 1) For 68B Flit mode only: Physical Layer transitions from Recovery to L0 and State Status ALMP synchronization for Recovery exit resolves to Active. 2) For 68B Flit mode only: Physical Layer transitions from Recovery to L0 and State Status ALMP synchronization for Recovery exit does not resolve to Active. Entry to Active ALMP exchange protocol is complete (see Section 5.1.2.2). 3) Physical Layer has been in L0. Entry to Active ALMP exchange protocol is complete (see Section 5.1.2.2).	Link Layer is requesting Active and any of the following conditions are met: 1) For 68B Flit mode only: Physical Layer transitions from Recovery to L0 and State Status ALMP synchronization for Recovery exit resolves to Active. 2) For 68B Flit mode only: Physical Layer transitions from Recovery to L0 and State Status ALMP synchronization for Recovery exit does not resolve to Active. Entry to Active ALMP exchange protocol is complete (see Section 5.1.2.2). 3) Physical Layer has been in L0. Entry to Active ALMP exchange protocol is complete (see Section 5.1.2.2).
Retrain	Reset	For 68B Flit mode: Physical Layer transitions from Recovery to L0 and State Status ALMP synchronization for Recovery exit resolves to Reset (see Section 5.1.2.3). For 256B Flit mode, this arc is N/A.	N/A
ANY (Except Disable/LinkError)	LinkReset	Physical Layer LTSSM in Hot Reset	Physical Layer LTSSM in Hot Reset

Table 5-3. ARB/MUX State Transition Table (Sheet 2 of 2)

Current vLSM State	Next State	Upstream Port Trigger Condition	Downstream Port Trigger Condition
ANY (Except LinkError)	Disabled	Physical Layer LTSSM in Disabled state	Physical Layer LTSSM in Disabled state
ANY	LinkError	Directed to enter LinkError from Link Layer or indication of LinkError from Physical Layer	Directed to enter LinkError from Link Layer or indication of LinkError from Physical Layer
L2	Reset	Implementation Specific. Refer to rule 3 in Section 5.1.1 .	Implementation Specific. Refer to rule 3 in Section 5.1.1 .
Disabled	Reset	Implementation Specific. Refer to rule 3 in Section 5.1.1 .	Implementation Specific. Refer to rule 3 in Section 5.1.1 .
LinkError	Reset	Implementation Specific. Refer to rule 3 in Section 5.1.1 .	Implementation Specific. Refer to rule 3 in Section 5.1.1 .
LinkReset	Reset	Implementation Specific. Refer to rule 3 in Section 5.1.1 .	Implementation Specific. Refer to rule 3 in Section 5.1.1 .
Reset	Active	Any of the following conditions are met: 1) Link Layer is asking for Active and Entry to Active ALMP exchange protocol is complete (see Section 5.1.2.2). 2) For 68B Flit mode only: Physical Layer transitions from Recovery to L0 and State Status ALMP synchronization for Recovery exit resolves to Active (see Section 5.1.2.3).	Any of the following conditions are met: 1) Link Layer is asking for Active and Entry to Active ALMP exchange protocol is complete (see Section 5.1.2.2). 2) For 68B Flit mode only: Physical Layer transitions from Recovery to L0 and State Status ALMP synchronization for Recovery exit resolves to Active (see Section 5.1.2.3).

5.1.1 Additional Rules for Local vLSM Transitions

- For 68B Flit mode, if any Link Layer requests entry into Retrain to the ARB/MUX, the ARB/MUX must forward the request to the Physical Layer to initiate LTSSM transition to Recovery. In accordance with the Active to Retrain transition trigger condition, after the LTSSM is in Recovery, the ARB/MUX should reflect Retrain to all vLSMs that are in Active state. For 256B Flit mode, there is no Active to Retrain arc in the ARB/MUX vLSM because Physical Layer LTSSM transitions to Recovery do not impact vLSM state.

Note:

For 256B Flit mode: Not exposing the Physical Layer LTSSM transition to Recovery to the Link Layer vLSMs allows for optimizations in which the Rx Retry buffer can drain while the LTSSM is in Recovery. It also avoids corner cases in which the vLSMs become out of sync with the remote Link partner. To handle error conditions such as UpdateFC DLLP timeouts, implementations must have a sideband mechanism from the Link Layers to the Physical Layer for triggering the LTSSM transition to Recovery.

- Once a vLSM is in Retrain state, it is expected that the corresponding Link Layer will eventually request ARB/MUX for a transition to Active.
- If the LTSSM moves to Detect, each vLSM must eventually transition to Reset.

5.1.2 Rules for vLSM State Transitions across Link

This section refers to vLSM state transitions.

5.1.2.1 General Rules

- The link cannot operate for any other protocols if the CXL.io protocol is down (CXL.io operation is a minimum requirement)

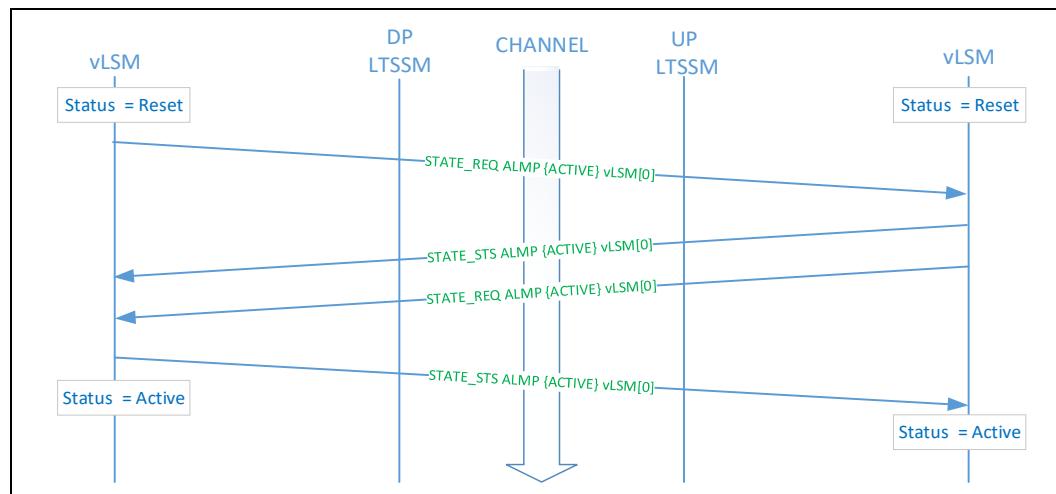
5.1.2.2 Entry to Active Exchange Protocol

The ALMP protocol required for the entry to active consists of 4 ALMP exchanges between the local and remote vLSMs as seen in [Figure 5-2](#). Entry to Active begins with an Active State Request ALMP sent to the remote vLSM which responds with an Active State Status ALMP. The only valid response to an Active State Request is an Active State Status once the corresponding Link Layer is ready to receive protocol flits. The remote vLSM must also send an Active State Request ALMP to the local vLSM which responds with an Active State Status ALMP.

During initial link training, the Upstream Port (UP in [Figure 5-2](#)) must wait for a non-physical layer flit (i.e., a flit that was not generated by the physical layer of the Downstream Port (DP in [Figure 5-2](#))) before transmitting any ALMPs (see [Section 6.4.1](#)). Thus, during initial link training, the first ALMP is always sent from the Downstream Port to the Upstream Port. If additional Active exchange handshakes subsequently occur (e.g., as part of PM exit), the Active request ALMP can be initiated from either side.

Once an Active State Status ALMP has been sent and received by a vLSM, the vLSM transitions to Active State.

Figure 5-2. Entry to Active Protocol Exchange



5.1.2.3 Status Synchronization Protocol

For 256B Flit mode, since the retry buffer is in the physical layer, all ALMPs are guaranteed to be delivered error free to the remote ARB/MUX. Additionally, all ALMPs are guaranteed to get a response. Therefore, there is no scenario where the Upstream Port and Downstream Port vLSMs can get out of sync.

Status Synchronization Protocol is only applicable for 68B Flit mode. The following description and rules are applicable for 68B Flit mode.

After the highest negotiated speed of operation is reached during initial link training, all subsequent LTSSM Recovery transitions must be signaled to the ARB/MUX. vLSM Status Synchronization Protocol must be performed after Recovery exit. A Link Layer cannot conduct any other communication on the link coming out of LTSSM recovery until Status Synchronization Protocol is complete for the corresponding vLSM. [Figure 5-3](#) shows an example of Status Synchronization Protocol.

The Status Synchronization Protocol completion requires the following events in the order listed:

1. Status Exchange: Transmit a State Status ALMP, and receive an error free State Status ALMP. The state indicated in the transmitted State Status ALMP is a snapshot of the vLSM state. Refer to [Section 5.1.2.3.1](#).
2. A corresponding State Status Resolution based on the sent and received State Status ALMPs during the synchronization exchange. See [Table 5-4](#) for determining the resolved vLSM state.
3. New State Request and Status ALMP exchanges when applicable. This occurs if the resolved vLSM state is not the same as the Link Layer requested state.

5.1.2.3.1 vLSM Snapshot Rule

A STATUS_EXCHANGE_PENDING variable is used to determine when a snapshot of the vLSM can be taken. The following rules apply:

- Snapshot of the vLSM is taken before entry to LTSSM Recovery if the STATUS_EXCHANGE_PENDING variable is cleared for that vLSM
- STATUS_EXCHANGE_PENDING variable is set for a vLSM once a snapshot is taken
- STATUS_EXCHANGE_PENDING variable is cleared on reset or on completion of Status Exchange (i.e., Transmit a State Status ALMP, and receive an error free State Status ALMP)

This is to account for situations where a corrupted State Status ALMP during Status Exchange can lead to additional LTSSM transitions through Recovery. See [Figure 5-16](#) for an example of this flow.

Figure 5-3. Example Status Exchange

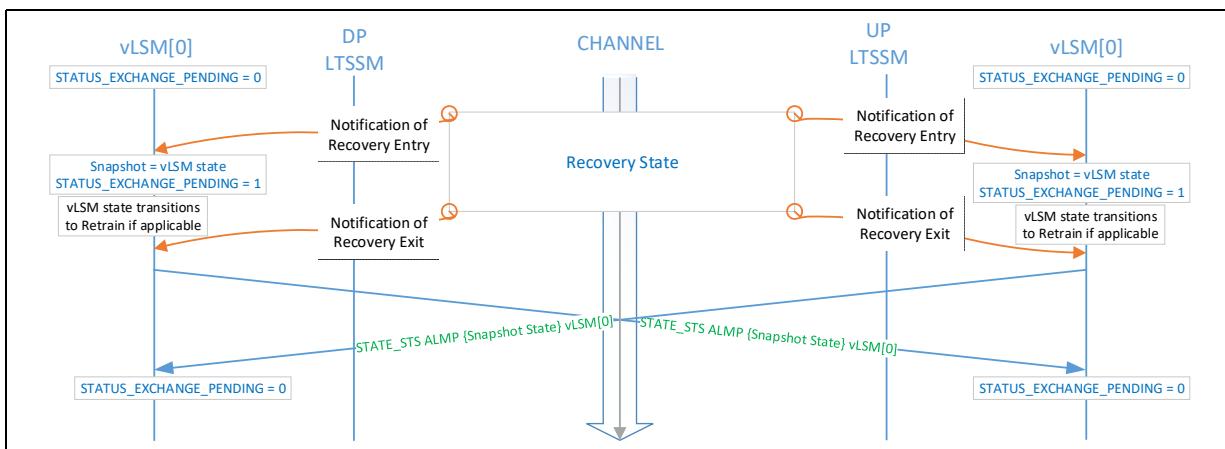


Table 5-4. vLSM State Resolution after Status Exchange (Sheet 1 of 2)

No.	Sent Status ALMP	Received Status ALMP	Resolved vLSM State
1.	Reset	Reset	Reset
2.	Reset	Active	Active
3.	Reset	L2	Reset
4.	Active	Reset	Active
5.	Active	Active	Active
6.	Active	Retrain	Active
7.	Active	L1.x	Retrain
8.	Active	L2	Reset

Table 5-4. vLSM State Resolution after Status Exchange (Sheet 2 of 2)

No.	Sent Status ALMP	Received Status ALMP	Resolved vLSM State
9.	Retrain	Active	Active
10.	Retrain	Retrain	Retrain
11.	Retrain	L1.x	Retrain
12.	L1.x	Active	L1.x
13.	L1.x	Retrain	L1.x
14.	L1.x	L1.x	L1.x
15.	L2	Active	L2
16.	L2	Reset	L2
17.	L2	L2	L2

5.1.2.3.2**Notes on State Resolution after Status Exchange (Table 5-4)**

- For the rows where the resolved state is Active, the corresponding ARB/MUX must ensure that protocol flits received immediately after the State Status ALMP from remote ARB/MUX can be serviced by the Link Layer of the corresponding vLSM. One way to guarantee this is to ensure that for these cases the Link Layer receiver is ready before sending the State Status ALMP during Status Exchange.
- Rows 7 and 11 will result in L1 exit flow following state resolution. The corresponding ARB/MUX must initiate a transition to Active through new State Request ALMPs. Once both the Upstream Port VLSM and Downstream Port vLSM are in Active, the Link Layers can redo PM entry negotiation if required. Similarly, for row 10 if reached during PM negotiation, it is required for both vLSMs to initiate Active request ALMPs.
- When supported, rows 3 and 8 will result in L2 exit flow following state resolution. Since the LTSSM will eventually move to Detect, each vLSM will eventually transition to Reset state.
- Rows 7 and 8 are applicable only for Upstream Ports. Since entry into PM is always initiated by the Upstream Port, and it cannot transition its vLSM to PM unless the Downstream Port has done so, there is no case where these rows can apply for Downstream Ports.
- Behavior is undefined and implementation specific for combinations not captured in Table 5-4.

5.1.2.4**State Request ALMP**

The following rules apply for sending a State Request ALMP. A State Request ALMP is sent to request a state change to Active or PM. For PM, the request can only be initiated by the ARB/MUX on the Upstream Port.

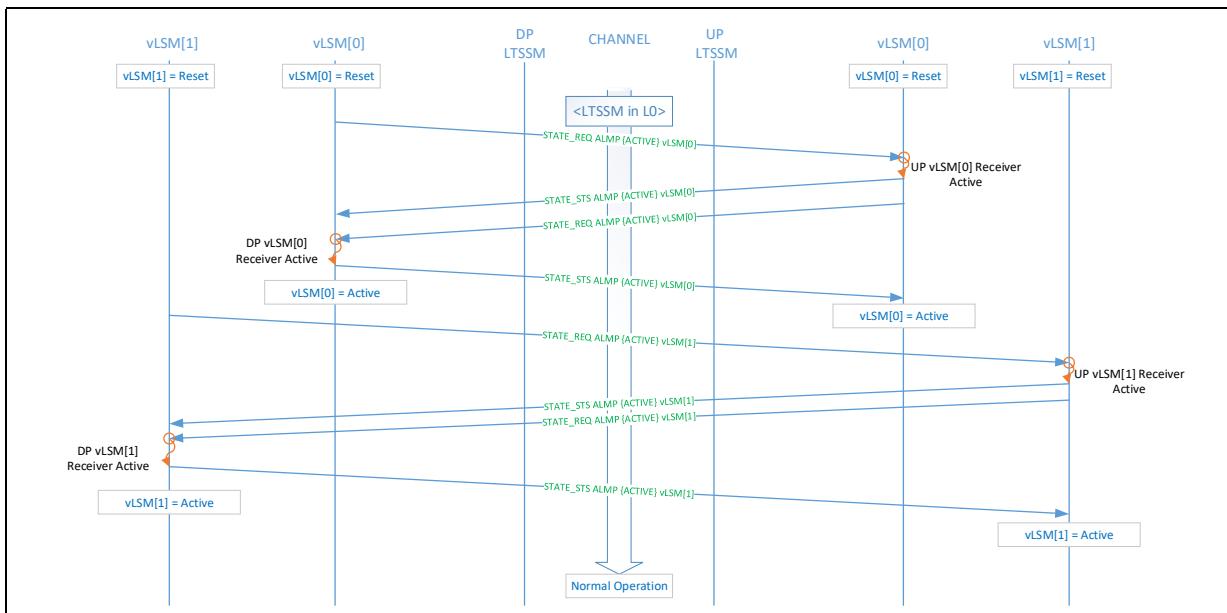
5.1.2.4.1**For Entry into Active**

- All Recovery state operations must complete before the entry to Active sequence starts. For 68B Flit mode, this includes the completion of Status Synchronization Protocol after LTSSM transitions from Recovery to L0.
- An ALMP State Request is sent to initiate the entry into Active State.
- A vLSM must send a Request and receive a Status before the transmitter is considered active. This is not equivalent to vLSM Active state.
- Protocol layer flits must only be transmitted once the vLSM has reached Active state.

Figure 5-4 shows an example of entry into the Active state. The flows in **Figure 5-4** show four independent actions (ALMP handshakes) that may not necessarily occur in the order or small timeframe shown. The vLSM transmitter and receiver may become active independent of one another. Both transmitter and receiver must be active before the vLSM state is Active. The transmitter becomes active after a vLSM has transmitted a Request ALMP{Active} and received a Status ALMP{Active}. The receiver becomes active after a vLSM receives a Request ALMP{Active} and sends a Status ALMP{Active} in response.

Please refer to [Section 5.1.2.2](#) for rules regarding the Active State Request/Status handshake protocol.

Figure 5-4. CXL Entry to Active Example Flow

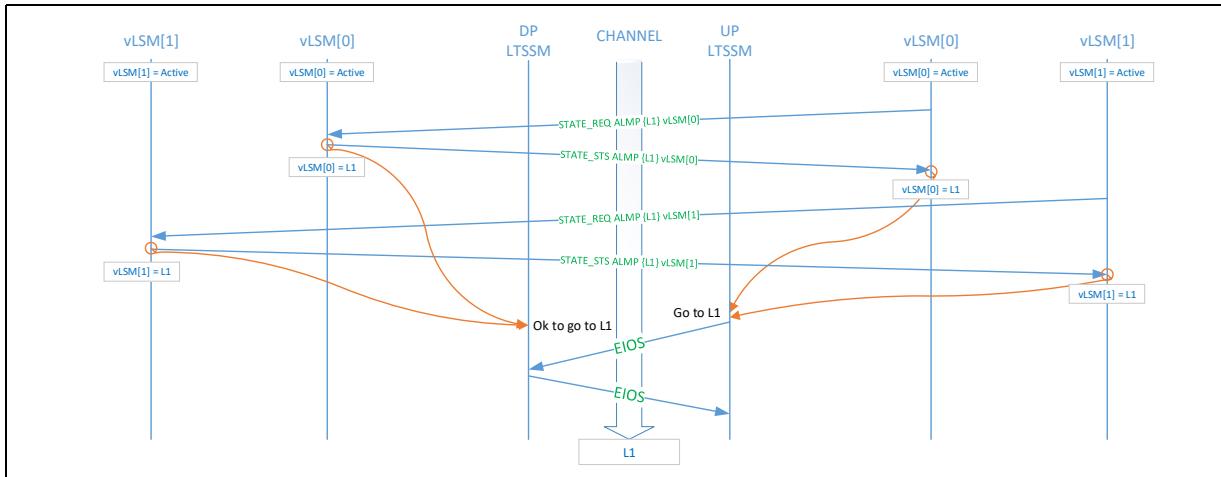


5.1.2.4.2 For Entry into PM State (L1/L2)

- An ALMP State Request is sent to initiate the entry into PM States. Only Upstream Ports can initiate entry into PM states.
- For Upstream Ports, a vLSM must send a Request and receive a Status before the PM negotiation is considered complete for the corresponding vLSM.

Figure 5-5 shows an example of Entry to PM State (L1) initiated by the Upstream Port (UP in the figure) ARB/MUX. Each vLSM will be ready to enter L1 State once the vLSM has sent a Request ALMP{L1} and received a Status ALMP{L1} in return or the vLSM has received a Request ALMP{L1} and sent a Status ALMP{L1} in return. The vLSMs operate independently and actions may not complete in the order or within the timeframe shown. Once all vLSMs are ready to enter PM State (L1), the Channel will complete the EIOS exchange and enter L1.

Figure 5-5. CXL Entry to PM State Example



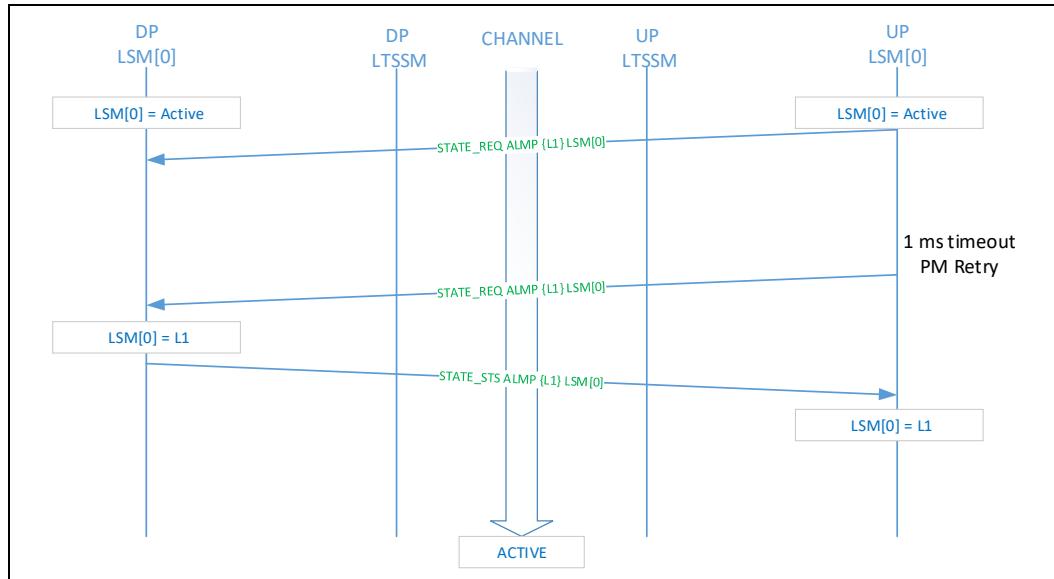
5.1.2.4.2.1 PM Retry and Reject Scenarios for 68B Flit Mode

This section is applicable for 68B Flit mode only. If PM entry is not accepted by the Downstream Port, it must not respond to the PM State Request. In this scenario:

- The Upstream Port is permitted to retry entry into PM with another PM State Request after a 1-ms (not including time spent in recovery states) timeout, when waiting for a response for a PM State Request. Upstream Port must not expect a PM State Status response for every PM State Request ALMP. Even if the Upstream Port has sent multiple PM State Requests because of PM retries, if it receives a single PM State Status ALMP, it must move the corresponding vLSM to the PM state indicated in the ALMP. For a Downstream Port, if the vLSM is Active and it has received multiple PM State Request ALMPs for that vLSM, it is permitted to treat the requests as a single PM request and respond with a single PM State Status only if the vLSM transitions into the PM state. **Figure 5-6** shows an example of this flow.

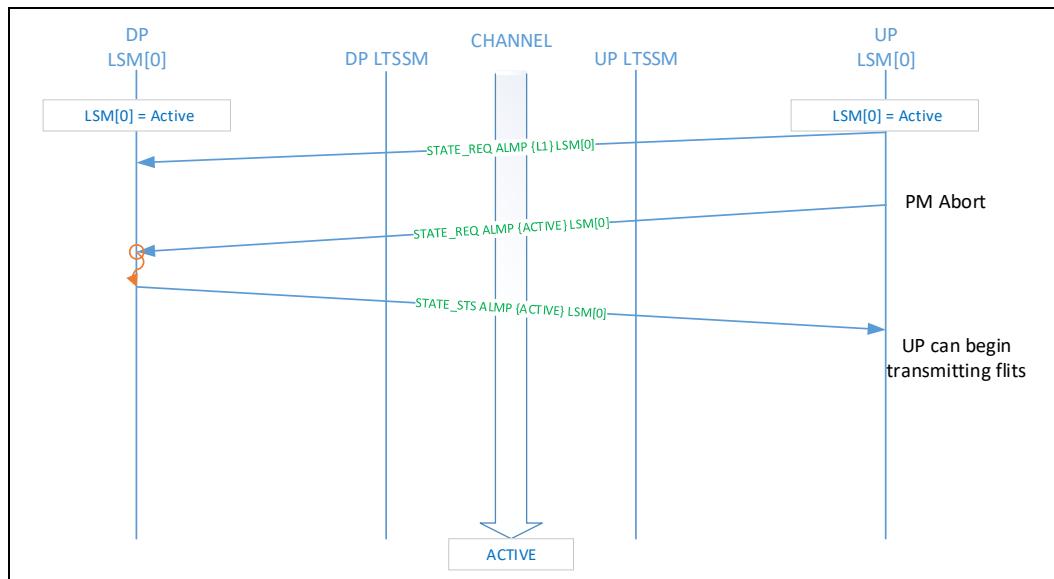
EVALUATION COPY

Figure 5-6. Successful PM Entry following PM Retry



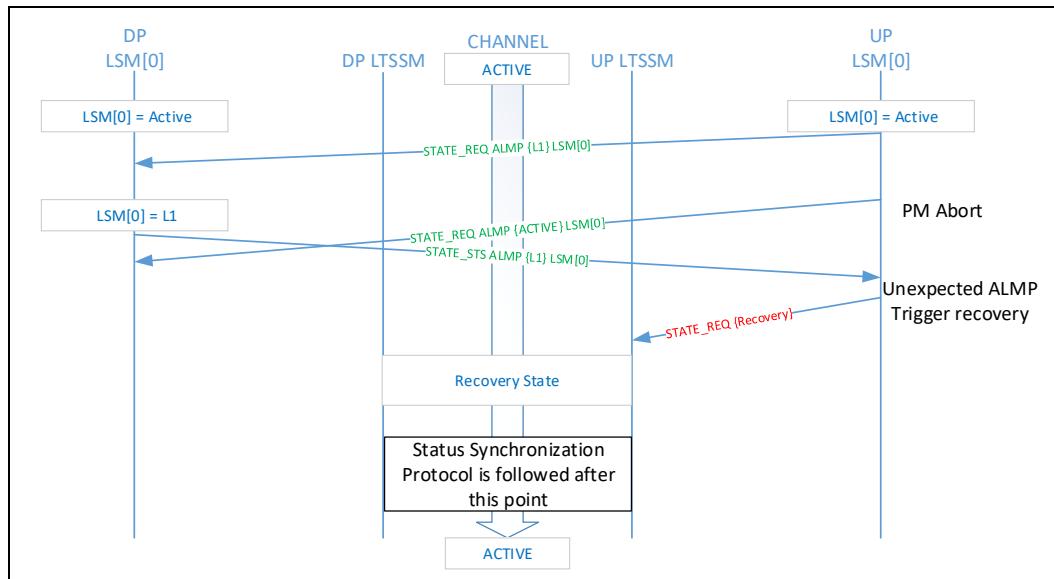
- The Upstream Port is also permitted to abort entry into PM by sending an Active State Request ALMP for the corresponding vLSM. Two scenarios are possible in this case:
 - Downstream Port receives the Active State Request before the commit point of PM acceptance. The Downstream Port must abort PM entry and respond with Active State Status ALMP. The Upstream Port can begin flit transfer toward the Downstream Port once Upstream Port receives Active State Status ALMP. Since the vLSMs are already in Active state and flit transfer was already allowed from the Downstream Port to the Upstream Port direction during this flow, there is no Active State Request ALMP from the Downstream Port-to-Upstream Port direction. [Figure 5-7](#) shows an example of this flow.

Figure 5-7. PM Abort before Downstream Port PM Acceptance



- Downstream Port receives the Active State Request after the commit point of PM acceptance or after its vLSM is in a PM state. The Downstream Port must finish PM entry and send PM State Status ALMP (if not already done so). The Upstream Port must treat the received PM State Status ALMP as an unexpected ALMP and trigger link Recovery. [Figure 5-8](#) shows an example of this flow.

Figure 5-8. PM Abort after Downstream Port PM Acceptance



5.1.2.4.2.2 PM Retry and Reject Scenario for 256B Flit Mode

This section is applicable for 256B Flit mode only. Upon receiving a PM Request ALMP, the Downstream Port must respond to it with either a PM Status ALMP or an Active.PMNAK Status ALMP.

It is strongly recommended for the Downstream Port ARB/MUX to send the response ALMP to the Physical Layer within 10 us of receiving the request ALMP from the Physical Layer (the time is counted only during the L0 state of the physical LTSSM, excluding the time spent in the Downstream Port's Rx Retry buffer for the request, or the time spent in the Downstream Port's Tx Retry buffer for the response). If the Downstream Port does not meet the conditions to accept PM entry within that time window, it must respond with an Active.PMNAK Status ALMP.

The Downstream Port ARB/MUX must wait for at least 1 us after receiving the PM Request ALMP from the Physical Layer before deciding whether to schedule an Active.PMNAK Status ALMP.

Note:

There is no difference between a PM Request ALMP for PCI*-PM vs. ASPM. For both cases on the CXL.io Downstream Port, idle time with respect to lack of TLP flow triggers the Link Layer to request L1 to ARB/MUX. Waiting for at least 1 us on the Downstream Port, the ARB/MUX provides sufficient time for the PCI-PM-related CSR completion from the Upstream Port to the Downstream Port for the write to the non-D0 state to exit the Downstream Port's CXL.io Link Layer, and reduces the likelihood of returning an Active.PMNAK Status ALMP.

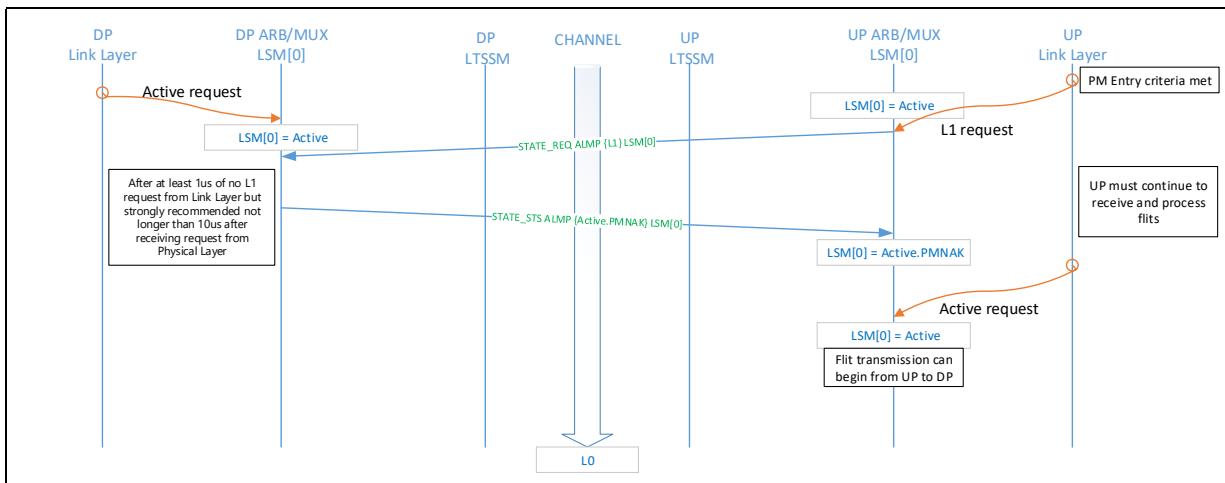
Upon receiving an Active.PMNAK Status ALMP, the Upstream Port must transition the corresponding vLSM to Active.PMNAK state. The Upstream port must continue to receive and process flits while the vLSM state is Active or Active.PMNAK. If PMTimeout

(see [Section 8.2.5.1](#)) is enabled and a response is not received for a PM Request ALMP within the programmed time window, the ARB/MUX must treat this as an uncorrectable internal error and escalate accordingly.

For Upstream Ports, after the Link Layer requests PM entry, the Link Layer must not change this request until it observes the vLSM status change to either the requested state or Active.PMNAK or one of the non-virtual states (LinkError, LinkReset, LinkDisable, or Reset). If Active.PMNAK is observed, the Link Layer must request Active to the ARB/MUX and wait for the vLSM to transition to Active before transmitting flits or re-requesting PM entry (if PM entry conditions are met).

The PM handshakes are reset by any events that cause physical layer LTSSM transitions that result in vLSM states of LinkError, LinkReset, LinkDisable, or Reset; these can occur at any time. Because these are Link down events, no response will be received for any outstanding Request ALMPs.

Figure 5-9. Example of a PMNAK Flow



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 us. (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 24 ms 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 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.

5.1.2.6 State Status ALMP

5.1.2.6.1 When State Request ALMP Is Received

A State Status ALMP is sent after a valid State Request ALMP is received for Active State (if the current vLSM state is already in Active, or if the current vLSM state is not Active and the request is following the entry into Active protocol) or PM States (when entry to the PM state is accepted). For 68B Flit mode, no State Status ALMP is sent if the PM state is not accepted. For 256B Flit mode, an Active.PMNAK State Status ALMP must be sent if the PM state is not accepted.

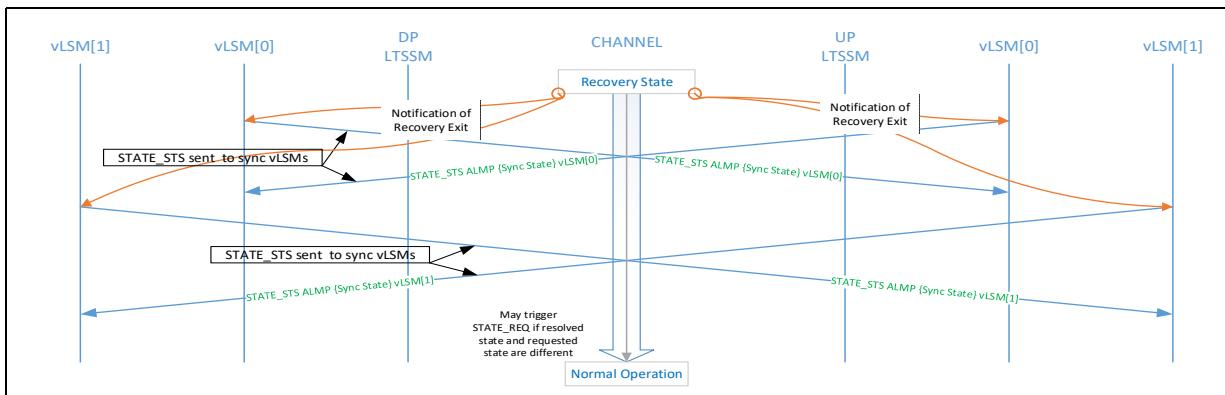
5.1.2.6.2 Recovery State (68B Flit Mode Only)

The rules in this section apply only for 68B Flit mode. For 256B Flit mode, physical layer Recovery does not trigger the Status Synchronization protocol.

- The vLSM will trigger link Recovery if a State Status ALMP is received without a State Request first being sent by the vLSM except when the State Status ALMP is received for synchronization purposes (i.e., after the link exits Recovery).

[Figure 5-10](#) shows a general example of Recovery exit. Please refer to [Section 5.1.2.3](#) for details on the status synchronization protocol.

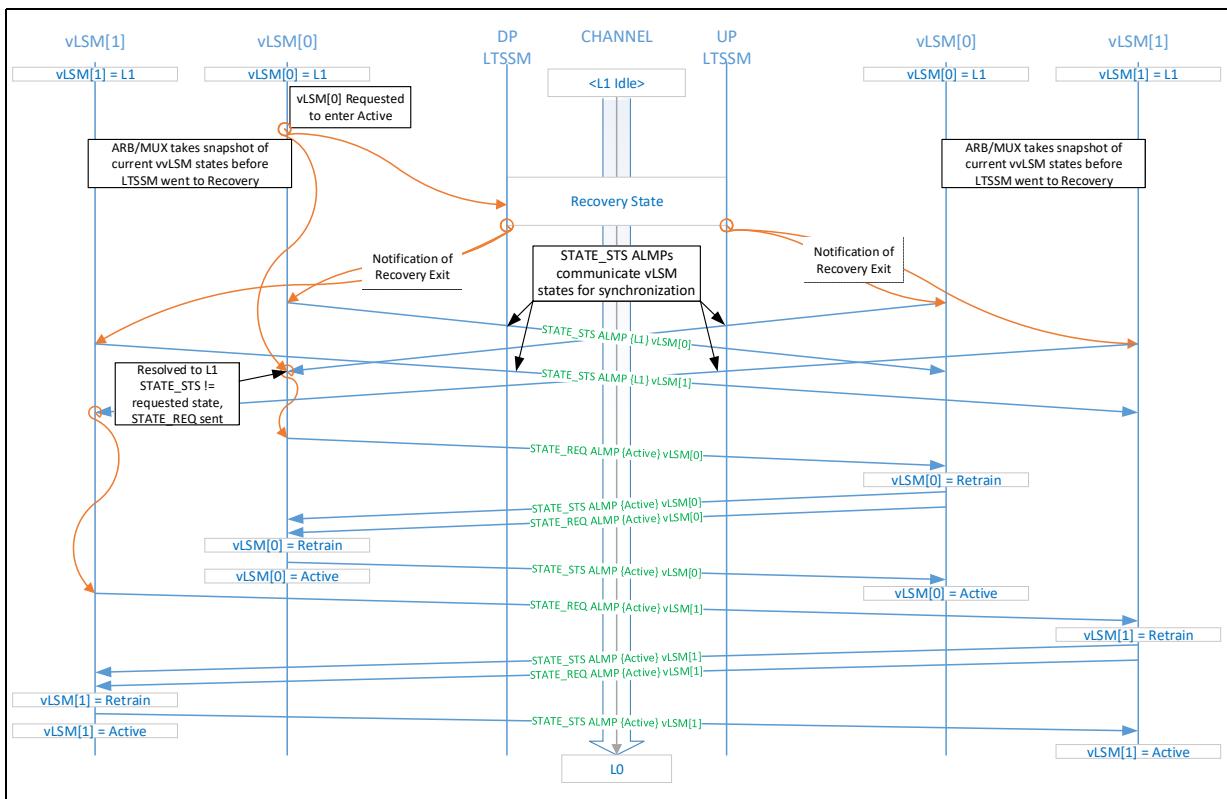
Figure 5-10. CXL Recovery Exit Example Flow



On Exit from Recovery, the vLSMs on either side of the channel will send a Status ALMP to synchronize the vLSMs. The Status ALMPs for synchronization may trigger a State Request ALMP if the resolved state and the Link Layer requested state are not the same, as seen in [Figure 5-11](#). Refer to [Section 5.1.2.3](#) for the rules that apply during state synchronization. The ALMP for synchronization may trigger a re-entry to recovery in the case of unexpected ALMPs. This is explained using the example of initial link training flows in [Section 5.1.3.1](#). If the resolved states from both vLSMs are the same as the Link Layer requested state, the vLSMs are considered to be synchronized and will continue normal operation.

Figure 5-11 shows an example of the exit from a PM State (L1) through Recovery. The Downstream Port (DP in the figure) vLSM[0] in L1 state receives the Active Request, and the link enters Recovery. After the exit from recovery, each vLSM sends Status ALMP{L1} to synchronize the vLSMs. Because the resolved state after synchronization is not equal to the requested state, Request ALMP{Active} and Status ALMP{Active} handshakes are completed to enter Active State.

Figure 5-11. CXL Exit from PM State Example



5.1.2.7 Unexpected ALMPs (68B Flit Mode Only)

Unexpected ALMPs are applicable only for 68B Flit mode. For 256B Flit mode, there are no scenarios that lead to unexpected ALMPs.

The following situations describe circumstances where an unexpected ALMP will trigger link recovery:

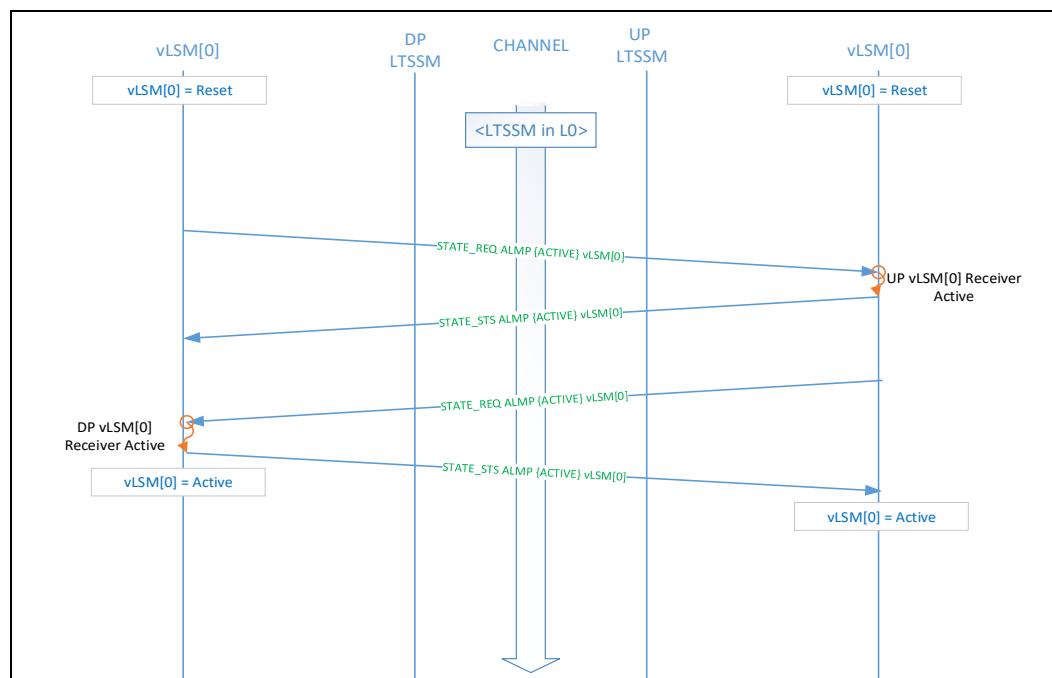
- When performing the Status Synchronization Protocol after exit from recovery, any ALMP other than a Status ALMP is considered an unexpected ALMP and will trigger recovery.
- When an Active Request ALMP has been sent, receipt of any ALMP other than an Active State Status ALMP or an Active Request ALMP is considered an unexpected ALMP and will trigger recovery.
- As outlined in [Section 5.1.2.6.2](#), a State Status ALMP received without a State Request ALMP first being sent is an unexpected ALMP except during the Status Synchronization Protocol.

5.1.3 Applications of the vLSM State Transition Rules for 68B Flit Mode

5.1.3.1 Initial Link Training

As the link trains from 2.5 GT/s speed to the highest supported speed (8.0 GT/s or higher for CXL), the LTSSM may go through several Recovery to L0 to Recovery transitions. Implementations are not required to expose ARB/MUX to all of these Recovery transitions. Depending on whether these initial Recovery transitions are hidden from the ARB/MUX, there are four possible scenarios for the initial ALMP handshakes. In all cases, the vLSM state transition rules guarantee that the situation will resolve itself with the vLSMs reaching Active state. These scenarios are presented in the following figures. Note that the figures are illustrative examples, and implementations must follow the rules outlined in the previous sections. Only one vLSM handshake is shown in the figures, but the similar handshakes can occur for the second vLSM as well. [Figure 5-12](#) shows an example of the scenario where both the Upstream Port and Downstream Port (UP and DP in the figure, respectively) are hiding the initial recovery transitions from ARB/MUX. Since neither of them saw a notification of recovery entry, they proceed with the exchange of Active request and status ALMPs to transition into the Active state. Note that the first ALMP (Active request ALMP) is sent from the Downstream Port to the Upstream Port.

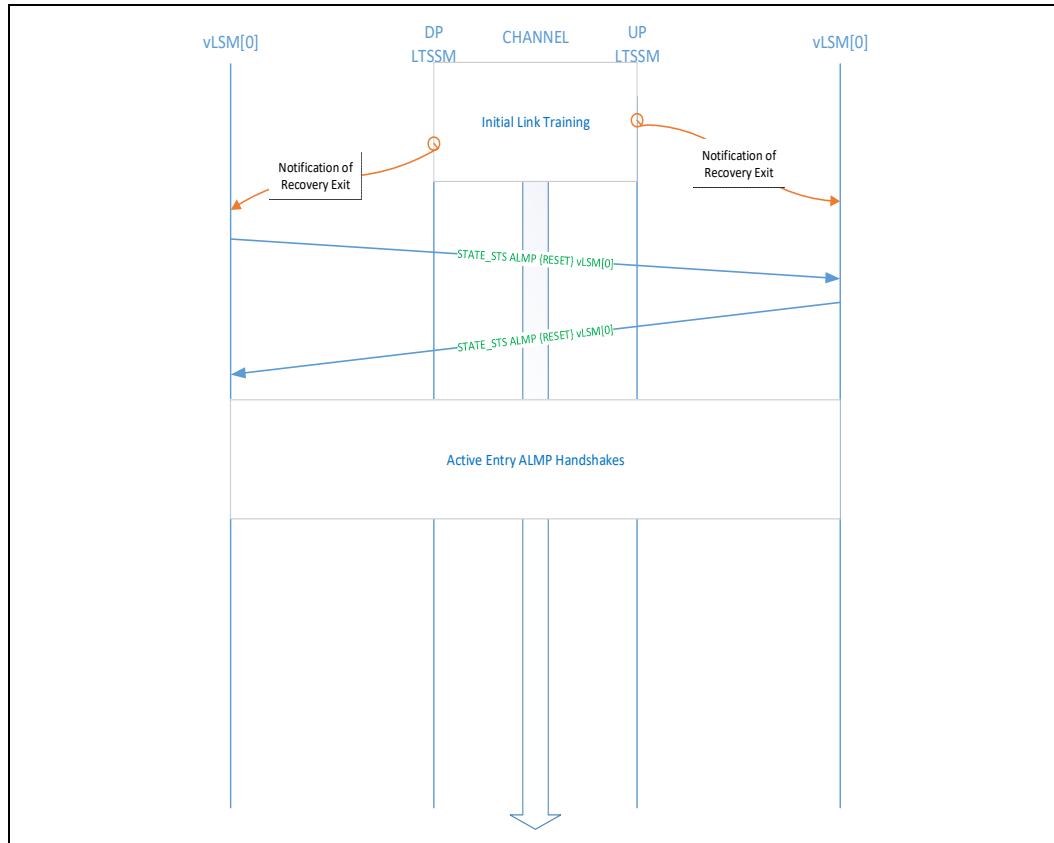
Figure 5-12. Both Upstream Port and Downstream Port Hide Recovery Transitions from ARB/MUX



[Figure 5-13](#) shows an example where both the Upstream Port and Downstream Port (UP and DP in the figure, respectively) notify the ARB/MUX of at least one recovery transition during initial link training. In this case, first state status synchronization ALMPs are exchanged (indicating Reset state), followed by regular exchange of Active request and status ALMPs (not explicitly shown). Note that the first ALMP (Reset status) is sent from the Downstream Port to the Upstream Port.

EVALUATION COPY

Figure 5-13. Both Upstream Port and Downstream Port Notify ARB/MUX of Recovery Transitions



EVALUATION COPY

Figure 5-14 shows an example of the scenario where the Downstream Port (DP in the figure) hides initial recovery transitions from the ARB/MUX, but the Upstream Port (UP in the figure) does not. In this case, the Downstream Port ARB/MUX has not seen recovery transition, so it begins by sending an Active state request ALMP to the Upstream Port. The Upstream Port interprets this as an unexpected ALMP, which triggers link recovery (which must now be communicated to the ARB/MUX because it is after reaching operation at the highest supported link speed). State status synchronization with state=Reset is performed, followed by regular Active request and status handshakes (not explicitly shown).

Figure 5-14. Downstream Port Hides Initial Recovery, Upstream Port Does Not

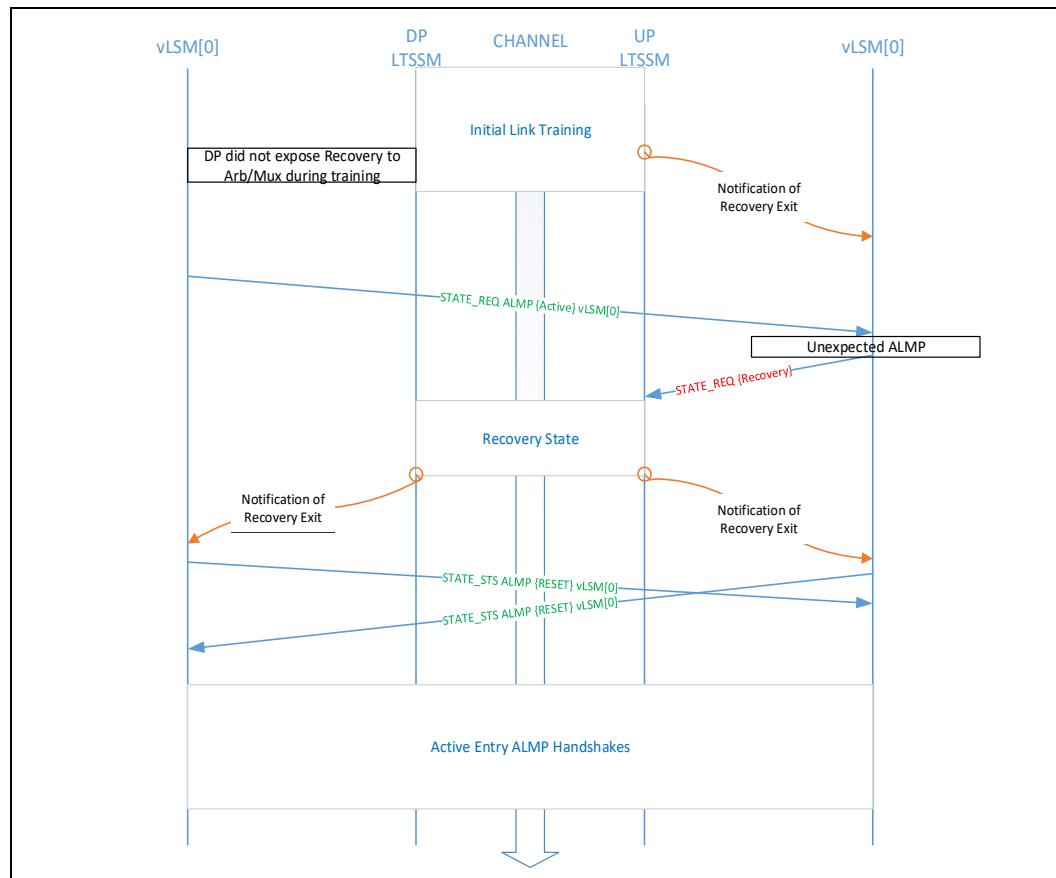
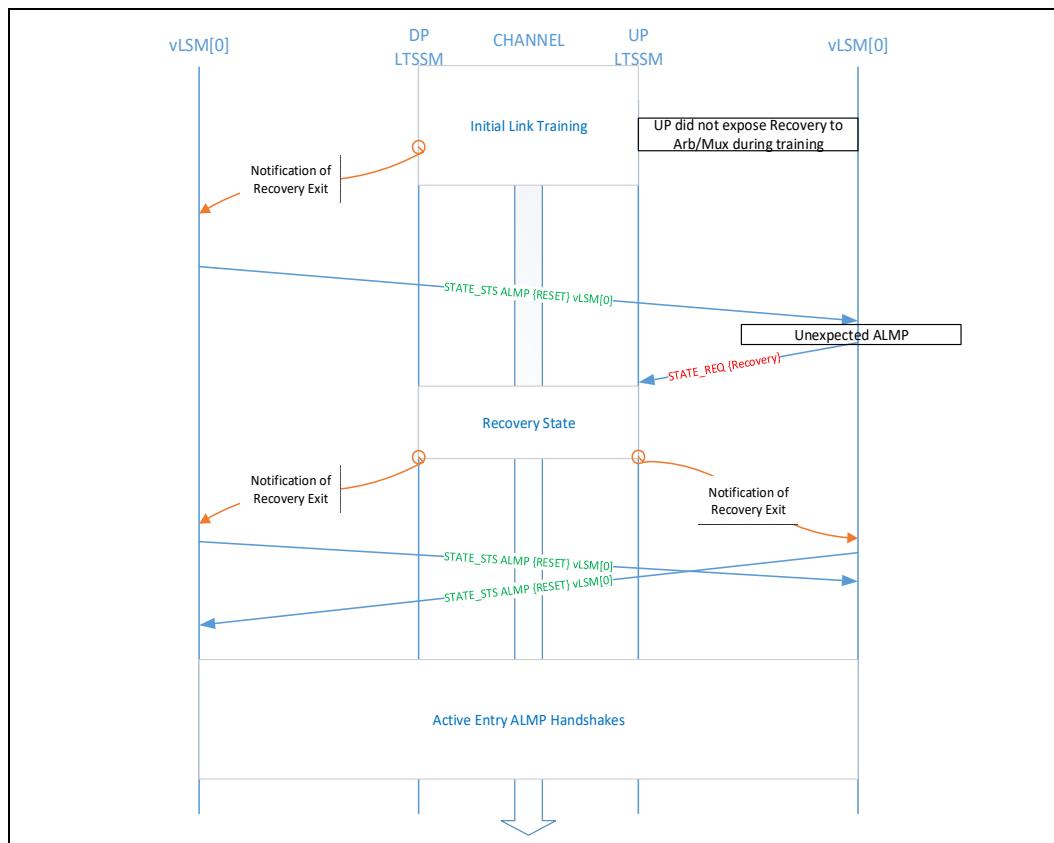


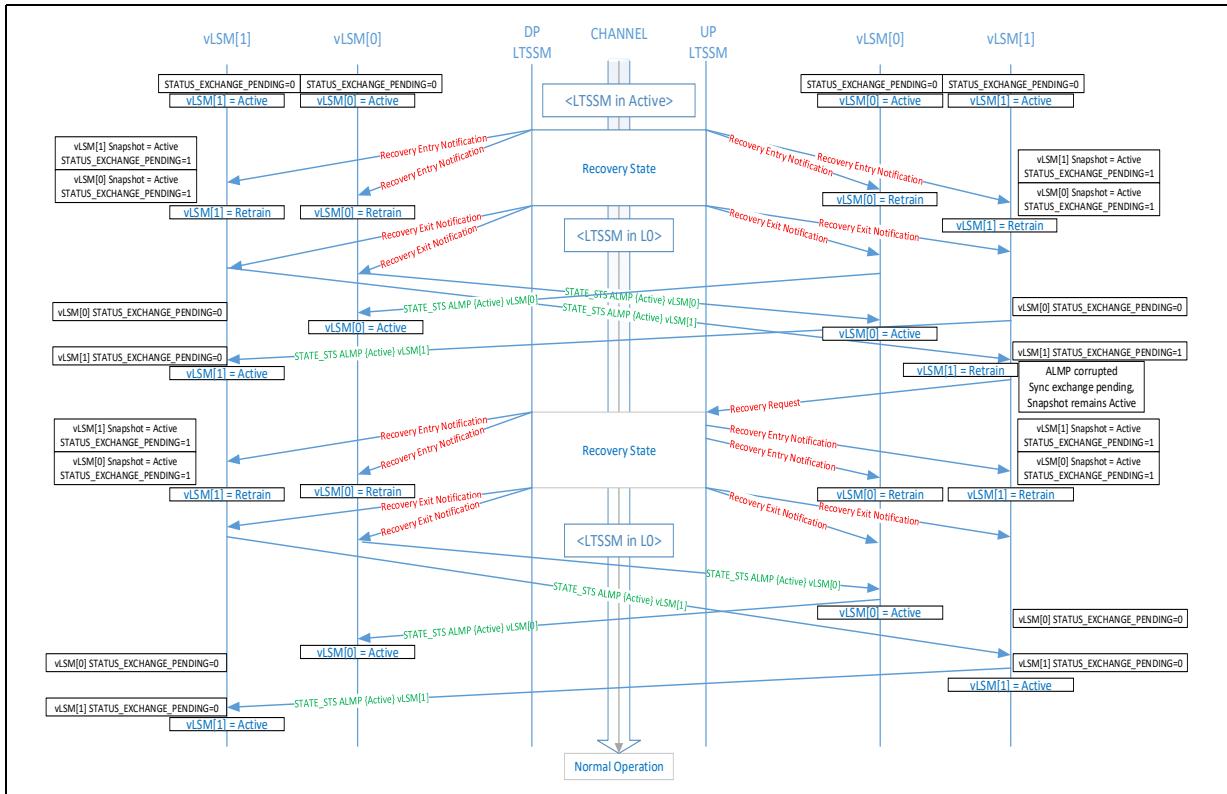
Figure 5-15 shows an example of the scenario where the Upstream Port (UP in the figure) hides initial recovery transitions, but the Downstream Port (DP in the figure) does not. In this case, the Downstream Port first sends a Reset status ALMP. This will cause the Upstream Port to trigger link recovery as a result of the rules in [Section 5.1.2.4.2.1](#) (which must now be communicated to the ARB/MUX because it is after reaching operation at the highest supported link speed). State status synchronization with state=Reset is performed, followed by regular Active request and status handshakes (not explicitly shown).

Figure 5-15. Upstream Port Hides Initial Recovery, Downstream Port Does Not



5.1.3.2 Status Exchange Snapshot Example

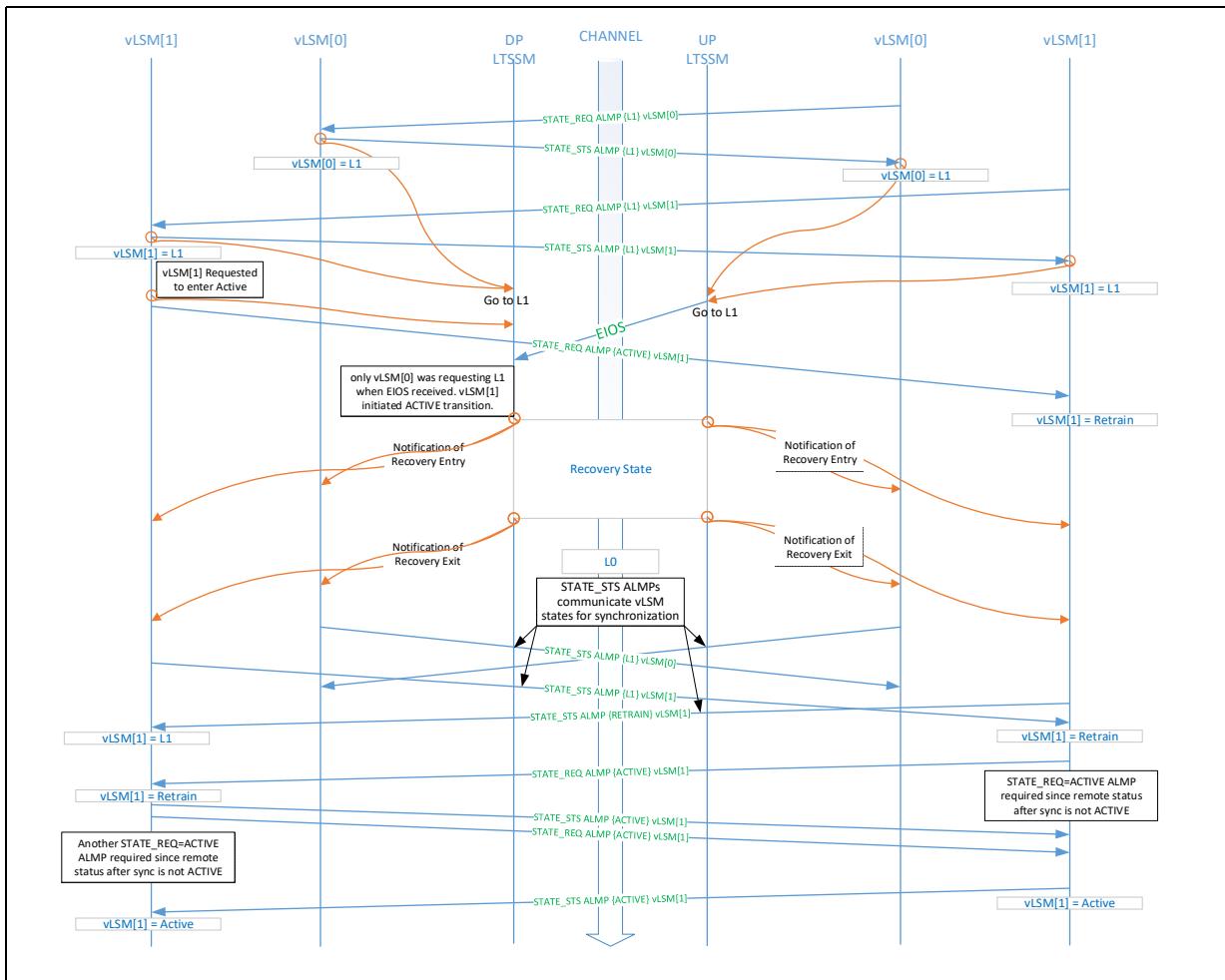
Figure 5-16 shows an example case where a State Status ALMP during Status Exchange gets corrupted for vLSM[1] on the Upstream Port (UP in the figure). A corrupted ALMP is when the lower four DWORDs don't match for a received ALMP; it indicates a bit error on the lower four DWORDs of the ALMP during transmission. The ARB/MUX triggers LTSSM Recovery as a result. When the recovery entry notification is received for the second Recovery entry, the snapshot of vLSM[1] on the Upstream Port is still Active since the status exchanges had not successfully completed.

Figure 5-16. Snapshot Example during Status Synchronization

5.1.3.3 L1 Abort Example

Figure 5-17 shows an example of a scenario that could arise during L1 transition of the physical link. It begins with successful L1 entry by both vLSMs through corresponding PM request and status ALMP handshakes. The ARB/MUX even requests the Physical Layer to take the LTSSM to L1 for both the Upstream Port and Downstream Port (UP and DP in Figure 5-17, respectively). However, there is a race and one of the vLSMs requests Active before EIOS is received by the Downstream Port Physical Layer. This causes the ARB/MUX to remove the request for L1 entry (L1 abort), while sending an Active request ALMP to the Upstream Port. When EIOS is eventually received by the physical layer, since the ARB/MUX on the Downstream Port side is not requesting L1 (and there is no support for L0s in CXL), the Physical Layer must take the LTSSM to Recovery to resolve this condition. On Recovery exit, both the Upstream Port and Downstream Port ARB/MUX send their corresponding vLSM state status as part of the synchronization protocol. For vLSM[1], since the resolved state status (Retrain) is not the same as desired state status (Active), another Active request ALMP must be sent by the Downstream Port to the Upstream Port. Similarly, on the Upstream Port side, the received state status (L1) is not the same as the desired state status (Active since the vLSM moving to Retrain will trigger the Upstream Port link layer to request Active), the Upstream Port ARB/MUX will initiate an Active request ALMP to the Downstream Port. After the Active state status ALMP has been sent and received, the corresponding ARB/MUX will move the vLSM to Active, and the protocol level flit transfer can begin.

Figure 5-17. L1 Abort Example



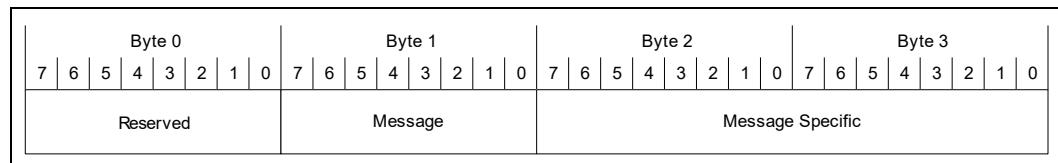
5.2

ARB/MUX Link Management Packets

The ARB/MUX uses ALMPs to communicate virtual link state transition requests and responses associated with each link layer to the remote ARB/MUX.

An ALMP is a 1-DWORD packet with the format shown in Figure 5-18. For 68B Flit mode, this 1-DWORD packet is replicated four times on the lower 16 bytes of a 528-bit flit to provide data integrity protection; the flit is zero-padded on the upper bits. If the ARB/MUX detects an error in the ALMP, it initiates a retrain of the link.

Figure 5-18. ARB/MUX Link Management Packet Format



For 256B Flit mode, Bytes 0, 1, 2, and 3 of the ALMP are placed on Bytes 2, 3, 4, and 5 of the 256B flit, respectively (as defined in [Section 6.2.3.1](#)). There is no replication since the ALMP is now protected through CRC and FEC. [Figure 5-19](#) shows the ALMP byte positions in the Standard 256B flit. [Figure 5-20](#) shows the ALMP byte positions in the Latency-Optimized 256B flit. See [Section 6.2.3.1](#) for definitions of the FlitHdr, CRC, and FEC bytes.

Figure 5-19. ALMP Byte Positions in Standard 256B Flit

FlitHdr (2 bytes)	ALMP Byte 0	ALMP Byte 1	ALMP Byte 2	ALMP Byte 3	122 bytes of 00h		
			114 bytes of 00h				CRC (8 bytes) FEC (6 bytes)

Figure 5-20. ALMP Byte Positions in Latency-Optimized 256B Flit

FlitHdr (2 bytes)	ALMP Byte 0	ALMP Byte 1	ALMP Byte 2	ALMP Byte 3	116 bytes of 00h	CRC (6 bytes)
			116 bytes of 00h			FEC (6 bytes) CRC (6 bytes)

For 256B Flit mode, there are two categories of ALMPs: the vLSM ALMPs and the L0p Negotiation ALMPs. For 68B Flit mode, only vLSM ALMPs are applicable. Byte 1 of the ALMP is shown in [Table 5-5](#).

Table 5-5. ALMP Byte 1 Encoding

Byte 1 Bits	Description
7:0	Message Encoding <ul style="list-style-type: none"> • 0000 0001b = L0p Negotiation ALMP (for 256B Flit mode; reserved for 68B Flit mode) • 0000 1000b = vLSM ALMP is encoded in Bytes 2 and 3 • All other encodings are reserved

Bytes 2 and 3 for vLSM ALMPs are shown in [Table 5-6](#). Bytes 2 and 3 for L0p Negotiation ALMPs are shown in [Table 5-7](#).

Table 5-6. ALMP Byte 2 and 3 Encodings for vLSM ALMP

Byte 2 Bits	Description
3:0	vLSM State Encoding Note: Rx should treat this as reserved for L0p ALMP. <ul style="list-style-type: none"> • 0000b = Reset (for Status ALMP) • 0000b = Reserved (for Request ALMP) • 0001b = Active • 0010b = Reserved (for Request ALMP) • 0010b = Active.PMNAK (for Status ALMP for 256B Flit mode; reserved for 68B Flit mode) • 0011b = DAPM (for Request ALMP) • 0011b = Reserved (for Status ALMP) • 0100b = IDLE_L1.0 (maps to PCIe L1) • 0101b = IDLE_L1.1 (reserved for future use) • 0110b = IDLE_L1.2 (reserved for future use) • 0111b = IDLE_L1.3 (reserved for future use) • 1000b = L2 • 1011b = Retrain (for Status ALMP only) • 1011b = Reserved (for Request ALMP) • All other encodings are reserved
6:4	Reserved
7	Request/Status Type <ul style="list-style-type: none"> • 0 = vLSM Status ALMP • 1 = vLSM Request ALMP
Byte 3 Bits	Description
3:0	Virtual LSM Instance Number: Indicates the targeted vLSM interface when there are multiple vLSMs present. <ul style="list-style-type: none"> • 0001b = ALMP for CXL.io • 0010b = ALMP for CXL.cache and CXL.mem • All other encodings are reserved
7:4	Reserved

Table 5-7. ALMP Byte 2 and 3 Encodings for L0p Negotiation ALMP (Sheet 1 of 2)

Byte 2 Bits	Description
5:0	Reserved
6	<ul style="list-style-type: none"> • 0 = Not an L0p.Priority Request • 1 = L0p.Priority Request
7	Request/Status Type <ul style="list-style-type: none"> • 0 = L0p Response ALMP • 1 = L0p Request ALMP

Table 5-7. ALMP Byte 2 and 3 Encodings for L0p Negotiation ALMP (Sheet 2 of 2)

Byte 3 Bits	Description
3:0	<ul style="list-style-type: none"> 0100b = ALMP for L0p (for 256B Flit mode; reserved for 68B Flit mode) All other encodings are reserved
7:4	<p>L0p Width Note: Encodings 0000b to 0100b are requests for L0p Request ALMP, and imply an ACK for L0p Response ALMP.</p> <ul style="list-style-type: none"> 0000b = x16 0001b = x8 0010b = x4 0011b = x2 0100b = x1 1000b = Reserved for L0p Request ALMP 1000b = L0p NAK for L0p Response ALMP All other encodings are reserved <p>If the width encoding in an ACK does not match the requested L0p width, the ARB/MUX must consider it a NAK. It is permitted to resend an L0p request, if the conditions of entry are still met.</p>

For vLSM ALMPs, the message code used in Byte 1 of the ALMP is 0000 1000b. These ALMPs can be request or status type. The local ARB/MUX initiates transition of a remote vLSM using a request ALMP. After receiving a request ALMP, the local ARB/MUX processes the transition request and returns a status ALMP. For 68B Flit mode, if the transition request is not accepted, a status ALMP is not sent and both local and remote vLSMs remain in their current state. For 256B Flit mode, if the PM transition request is not accepted, an Active.PMNAK Status ALMP is sent.

For L0p Negotiation ALMPs, the message code used in Byte 1 of the ALMP is 0000 0001b. These ALMPs can be of request or response type. See [Section 5.1.2.5](#) for L0p negotiation flow.

5.2.1

ARB/MUX Bypass Feature

The ARB/MUX must disable generation of ALMPs when the Flex Bus link is operating in PCIe mode. Determination of the bypass condition can be via hwinit or during link training.

5.3

Arbitration and Data Multiplexing/Demultiplexing

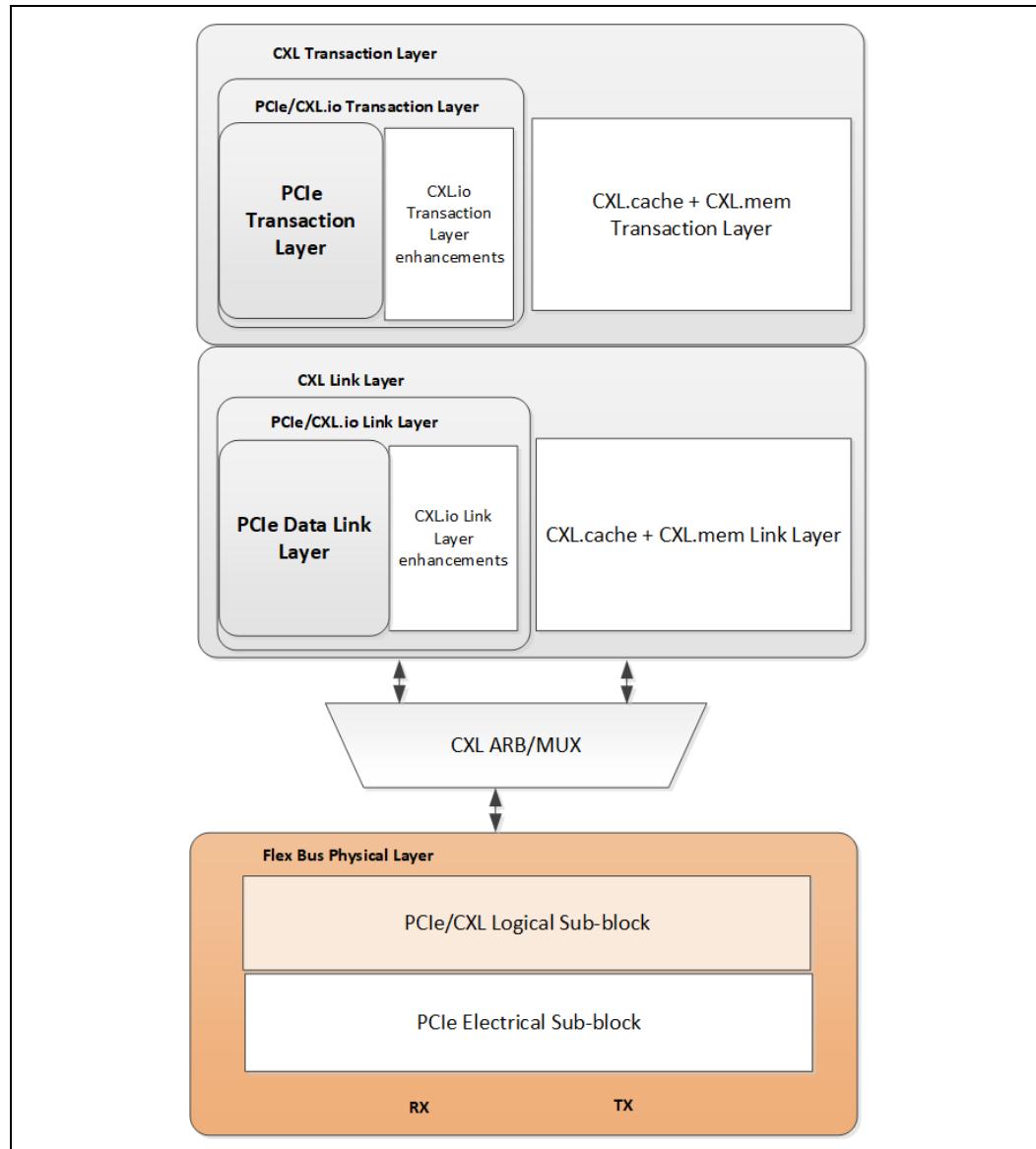
The ARB/MUX is responsible for arbitrating between requests from the CXL link layers and multiplexing the data based on the arbitration results. The arbitration policy is implementation specific as long as it satisfies the timing requirements of the higher-level protocols being transferred over the Flex Bus link. Additionally, there must be a way to program the relative arbitration weightages associated with the CXL.io and CXL.cache + CXL.mem link layers as they arbitrate to transmit traffic over the Flex Bus link. See [Section 8.2.5](#) for more details. Interleaving of traffic between different CXL protocols is done at the 528-bit flit boundary for 68B Flit mode, and at the 256B flit boundary for 256B Flit mode.

§ §

6.0 Flex Bus Physical Layer

6.1 Overview

Figure 6-1. Flex Bus Layers - Physical Layer Highlighted



The figure above shows where the Flex Bus physical layer exists in the Flex Bus layered hierarchy. On the transmitter side, the Flex Bus physical layer prepares data received from either the PCIe* link layer or the CXL ARB/MUX for transmission across the Flex Bus link. On the receiver side, the Flex Bus physical layer deserializes the data received on the Flex Bus link and converts it to the appropriate format to forward to the PCIe link layer or the ARB/MUX. The Flex Bus physical layer consists of a logical sub-block, aka the logical PHY, and an electrical sub-block. The logical PHY operates in PCIe mode during initial link training and switches over to CXL mode, if appropriate, depending on the results of alternate protocol negotiation, during recovery after training to 2.5 GT/s. The electrical sub-block follows PCIe Base Specification.

In CXL mode, normal operation occurs at native link width and 32 GT/s or 64 GT/s link speed. Bifurcation (aka link subdivision) into x8 and x4 widths is supported in CXL mode. Degraded modes of operation include 8 GT/s or 16 GT/s or 32 GT/s link speed and smaller link widths of x2 and x1. **Table 6-1** summarizes the supported CXL combinations. In PCIe mode, the link supports all widths and speeds defined in PCIe Base Specification, as well as the ability to bifurcate.

Table 6-1. Flex Bus.CXL Link Speeds and Widths for Normal and Degraded Mode

Link Speed	Native Width	Degraded Modes Supported
32 GT/s	x16	x16 at 16 GT/s or 8 GT/s; x8, x4, x2, or x1 at 32 GT/s or 16 GT/s or 8 GT/s
	x8	x8 at 16 GT/s or 8 GT/s; x4, x2, or x1 at 32 GT/s or 16 GT/s or 8 GT/s
	x4	x4 at 16 GT/s or 8 GT/s; x2 or x1 at 32 GT/s or 16 GT/s or 8 GT/s
64 GT/s	x16	x16 at 32 GT/s or 16 GT/s or 8 GT/s; x8, x4, x2, or x1 at 64 GT/s or 32 GT/s or 16 GT/s or 8 GT/s
	x8	x8 at 32 GT/s or at 16 GT/s or 8 GT/s; x4, x2, or x1 at 64 GT/s or 32 GT/s or 16 GT/s or 8 GT/s
	x4	x4 at 32 GT/s or at 16 GT/s or 8 GT/s; x2 or x1 at 64 GT/s or 32 GT/s or 16 GT/s or 8 GT/s

This chapter focuses on the details of the logical PHY. The Flex Bus logical PHY is based on the PCIe logical PHY; PCIe mode follows PCIe Base Specification exactly while Flex Bus.CXL mode has deltas from PCIe that affect link training and framing. Please refer to the "Physical Layer Logical Block" chapter of PCIe Base Specification for details on PCIe mode. The Flex Bus.CXL deltas are described in this chapter.

6.2

Flex Bus.CXL Framing and Packet Layout

The Flex Bus.CXL framing and packet layout is described in this section for x16, x8, x4, x2, and x1 link widths.

6.2.1

Ordered Set Blocks and Data Blocks

Flex Bus.CXL uses the PCIe concept of Ordered Set blocks and data blocks. Each block spans 128 bits per lane and potentially two bits of Sync Header per lane.

Ordered Set blocks are used for training, entering and exiting Electrical Idle, transitions to data blocks, and clock tolerance compensation; they are the same as defined in PCIe Base Specification. A 2-bit Sync Header with value 01b is inserted before each 128 bits transmitted per lane in an Ordered Set block when 128b/130b encoding is used; in the Sync Header bypass latency-optimized mode, there is no Sync Header. Additionally, as per PCIe Base Specification, there is no Sync Header when 1b/1b encoding is used.

Data blocks are used for transmission of the flits received from the CXL ARB/MUX. In 68B Flit mode, a 16-bit Protocol ID field is associated with each 528-bit flit payload (512 bits of payload + 16 bits of CRC) received from the link layer, which is striped across the lanes on an 8-bit granularity; the placement of the Protocol ID depends on the width. A 2-bit Sync Header with value 10b is inserted before every 128 bits transmitted per lane in a data block when 128b/130b encoding is used; in the latency-optimized Sync Header Bypass mode, there is no Sync Header. A 528-bit flit may traverse the boundary between data blocks. In 256B Flit mode, the flits are 256 bytes, which includes the Protocol ID information in the Flit Type field.

Transitions between Ordered Set blocks and data blocks are indicated in a couple of ways, only a subset of which may be applicable depending on the data rate and CXL mode. One way is via the 2-bit Sync Header of 01b for Ordered Set blocks and 10b for data blocks. The second way is via the use of Start of Data Stream (SDS) Ordered Sets and End of Data Stream (EDS) tokens. Unlike PCIe where the EDS token is explicit, Flex Bus.CXL encodes the EDS indication in the Protocol ID value in 68B Flit mode; the latter is referred to as an “implied EDS token.” In 256B Flit mode, transitions from Data Blocks to Ordered Set Blocks are permitted to occur at only fixed locations as specified in PCIe Base Specification for PCIe Flit mode.

6.2.2

68B Flit Mode

Selection of 68B Flit mode vs. 256B Flit mode occurs during PCIe link training. The following subsections describe the physical layer framing and packet layout for 68B Flit mode. See [Section 6.2.3](#) for 256B Flit mode.

6.2.2.1

Protocol ID[15:0]

The 16-bit Protocol ID field specifies whether the transmitted flit is CXL.io, CXL.cachemem, or some other payload. [Table 6-2](#) 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.

Table 6-2.

Flex Bus.CXL Protocol IDs

Protocol ID[15:0]	Description
FFFFh	CXL.io
D2D2h	CXL.io with Implied EDS Token
5555h	CXL.cachemem
8787h	CXL.cachemem with Implied EDS Token
9999h	NULL Flit: Null flit generated by the Physical Layer
4B4Bh	NULL flit with Implied EDS Token: Variable length flit containing NULLs that ends exactly at the data block boundary that precedes the Ordered Set block (generated by the Physical Layer)
CCCCh	CXL ARB/MUX Link Management Packets (ALMPs)
1E1Eh	CXL ARB/MUX Link Management Packets (ALMPs) with Implied EDS Token
All other encodings	Reserved

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 exactly at the data block boundary that precedes 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 128b/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 6.8.1](#)), variable length NULL flits will always contain a fixed 528-bit payload when Sync Headers are disabled. See [Section 6.8.1](#) for examples of NULL flit with implied EDS usage scenarios. A NULL flit is composed of an all 0s payload.

An 8-bit encoding with a hamming distance of four is replicated to create the 16-bit encoding for error protection against bit flips. A correctable Protocol ID framing error is logged but no further error handling action is required if only one 8-bit encoding group looks incorrect; the correct 8-bit encoding group is used for normal processing. If both 8-bit encoding groups are incorrect, an uncorrectable Protocol ID framing error is logged, the flit is dropped, and the physical layer enters into recovery to retrain the link.

The physical layer is responsible for dropping any flits it receives with invalid Protocol IDs. This includes dropping any flits with unexpected Protocol IDs that correspond to Flex Bus-defined protocols that have not been enabled during negotiation; Protocol IDs associated with flits generated by physical layer or by the ARB/MUX must not be treated as unexpected. When a flit is dropped due to an unexpected Protocol ID, the physical layer logs an unexpected protocol ID error in the Flex Bus DVSEC Port Status register.

See [Section 6.2.2.8](#) for additional details regarding Protocol ID error detection and handling.

6.2.2.2

x16 Packet Layout

[Figure 6-2](#) shows the x16 packet layout. First, the 16 bits of Protocol ID are transferred, split on an 8-bit granularity across consecutive lanes; this is followed by transfer of the 528-bit flit, striped across the lanes on an 8-bit granularity. Depending on the symbol time, as labeled on the leftmost column in the figure, the Protocol ID plus flit transfer may start on Lane 0, Lane 4, Lane 8, or Lane 12. The pattern of transfer repeats after every 17 symbol times. The two-bit Sync Header shown in the figure, inserted after every 128 bits transferred per lane, is not present for the latency-optimized mode where Sync Header bypass is negotiated.

Flex Bus Physical Layer**Figure 6-2. Flex Bus x16 Packet Layout**

	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15
Sync Hdr	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Symbol0	ProtID[7:0]	ProtID[15:8]	Flit[7:0]	Flit[15:8]	Flit[23:16]	Flit[31:24]	Flit[39:32]	Flit[47:40]	Flit[55:48]	Flit[63:56]	Flit[71:64]	Flit[79:72]	Flit[87:80]	Flit[95:88]	Flit[103:96]	Flit[111:104]
Symbol1	Flit[119:112]	Flit[127:120]	Flit[135:128]	Flit[143:136]	Flit[151:144]	Flit[159:152]	Flit[167:160]	Flit[175:168]	Flit[183:176]	Flit[191:184]	Flit[199:192]	Flit[207:200]	Flit[215:208]	Flit[223:216]	Flit[231:224]	Flit[239:232]
Symbol2	Flit[247:240]	Flit[255:248]	Flit[263:256]	Flit[271:264]	Flit[279:272]	Flit[287:280]	Flit[295:288]	Flit[303:296]	Flit[311:304]	Flit[319:312]	Flit[327:320]	Flit[335:328]	Flit[343:336]	Flit[351:344]	Flit[359:352]	Flit[367:360]
Symbol3	Flit[375:368]	Flit[383:376]	Flit[391:384]	Flit[399:392]	Flit[407:400]	Flit[415:408]	Flit[423:416]	Flit[431:424]	Flit[439:432]	Flit[447:440]	Flit[455:448]	Flit[463:456]	Flit[471:464]	Flit[479:472]	Flit[487:480]	Flit[495:488]
Symbol4	Flit[503:496]	Flit[511:504]	Flit[519:512]	Flit[527:520]	ProtID[7:0]	ProtID[15:8]	Flit[7:0]	Flit[15:8]	Flit[23:16]	Flit[31:24]	Flit[39:32]	Flit[47:40]	Flit[55:48]	Flit[63:56]	Flit[71:64]	Flit[79:72]
Symbol5	Flit[87:80]	Flit[95:88]	Flit[103:96]	Flit[111:104]	Flit[119:112]	Flit[127:120]	Flit[135:128]	Flit[143:136]	Flit[151:144]	Flit[159:152]	Flit[167:160]	Flit[175:168]	Flit[183:176]	Flit[191:184]	Flit[199:192]	Flit[207:200]
Symbol6	Flit[215:208]	Flit[223:216]	Flit[231:224]	Flit[239:232]	Flit[247:240]	Flit[255:248]	Flit[263:256]	Flit[271:264]	Flit[279:272]	Flit[287:280]	Flit[295:288]	Flit[303:296]	Flit[311:304]	Flit[319:312]	Flit[327:320]	Flit[335:328]
Symbol7	Flit[343:336]	Flit[351:344]	Flit[359:352]	Flit[367:360]	Flit[375:368]	Flit[383:376]	Flit[391:384]	Flit[399:392]	Flit[407:400]	Flit[415:408]	Flit[423:416]	Flit[431:424]	Flit[439:432]	Flit[447:440]	Flit[455:448]	Flit[463:456]
Symbol8	Flit[471:464]	Flit[479:472]	Flit[487:480]	Flit[495:488]	Flit[503:496]	Flit[511:504]	Flit[519:512]	Flit[527:520]	ProtID[7:0]	ProtID[15:8]	Flit[7:0]	Flit[15:8]	Flit[23:16]	Flit[31:24]	Flit[39:32]	Flit[47:40]
Symbol9	Flit[55:48]	Flit[63:56]	Flit[71:64]	Flit[79:72]	Flit[87:80]	Flit[95:88]	Flit[103:96]	Flit[111:104]	Flit[119:112]	Flit[127:120]	Flit[135:128]	Flit[143:136]	Flit[151:144]	Flit[159:152]	Flit[167:160]	Flit[175:168]
Symbol10	Flit[183:176]	Flit[191:184]	Flit[199:192]	Flit[207:200]	Flit[215:208]	Flit[223:216]	Flit[231:224]	Flit[239:232]	Flit[247:240]	Flit[255:248]	Flit[263:256]	Flit[271:264]	Flit[279:272]	Flit[287:280]	Flit[295:288]	Flit[303:296]
Symbol11	Flit[311:304]	Flit[319:312]	Flit[327:320]	Flit[335:328]	Flit[343:336]	Flit[351:344]	Flit[359:352]	Flit[367:360]	Flit[375:368]	Flit[383:376]	Flit[391:384]	Flit[399:392]	Flit[407:400]	Flit[415:408]	Flit[423:416]	Flit[431:424]
Symbol12	Flit[439:432]	Flit[447:440]	Flit[455:448]	Flit[463:456]	Flit[471:464]	Flit[479:472]	Flit[487:480]	Flit[495:488]	Flit[503:496]	Flit[511:504]	Flit[519:512]	Flit[527:520]	ProtID[7:0]	ProtID[15:8]	Flit[7:0]	Flit[15:8]
Symbol13	Flit[23:16]	Flit[31:24]	Flit[39:32]	Flit[47:40]	Flit[55:48]	Flit[63:56]	Flit[71:64]	Flit[79:72]	Flit[87:80]	Flit[95:88]	Flit[103:96]	Flit[111:104]	Flit[119:112]	Flit[127:120]	Flit[135:128]	Flit[143:136]
Symbol14	Flit[151:144]	Flit[159:152]	Flit[167:160]	Flit[175:168]	Flit[183:176]	Flit[191:184]	Flit[199:192]	Flit[207:200]	Flit[215:208]	Flit[223:216]	Flit[231:224]	Flit[239:232]	Flit[247:240]	Flit[255:248]	Flit[263:256]	Flit[271:264]
Symbol15	Flit[279:272]	Flit[287:280]	Flit[295:288]	Flit[303:296]	Flit[311:304]	Flit[319:312]	Flit[327:320]	Flit[335:328]	Flit[343:336]	Flit[351:344]	Flit[359:352]	Flit[367:360]	Flit[375:368]	Flit[383:376]	Flit[391:384]	Flit[399:392]
Sync Hdr	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Symbol0	Flit[407:400]	Flit[415:408]	Flit[423:416]	Flit[431:424]	Flit[439:432]	Flit[447:440]	Flit[455:448]	Flit[463:456]	Flit[471:464]	Flit[479:472]	Flit[487:480]	Flit[495:488]	Flit[503:496]	Flit[511:504]	Flit[519:512]	Flit[527:520]
Symbol1	ProtID[7:0]	ProtID[15:8]	Flit[7:0]	Flit[15:8]	Flit[23:16]	Flit[31:24]	Flit[39:32]	Flit[47:40]	Flit[55:48]	Flit[63:56]	Flit[71:64]	Flit[79:72]	Flit[87:80]	Flit[95:88]	Flit[103:96]	Flit[111:104]

Figure 6-3 provides an example where CXL.io and CXL.cachemem traffic is interleaved with an interleave granularity of two flits on a x16 link. The upper part of the figure shows what the CXL.io stream looks like before mapping to the Flex Bus lanes and before interleaving with CXL.cachemem traffic; the framing rules follow the x16 framing rules specified in PCIe Base Specification, as specified in [Section 4.1](#). The lower part of the figure shows the final result when the two streams are interleaved on the Flex Bus lanes. For CXL.io flits, after transferring the 16-bit Protocol ID, 512 bits are used to transfer CXL.io traffic and 16 bits are unused. For CXL.cachemem flits, after transferring the 16-bit Protocol ID, 528 bits are used to transfer a CXL.cachemem flit. See [Chapter 4.0](#) for more details on the flit format. As this example illustrates, the PCIe TLPs and DLLPs encapsulated within the CXL.io stream may be interrupted by non-related CXL traffic if they cross a flit boundary.

Flex Bus Physical Layer

Figure 6-3. Flex Bus x16 Protocol Interleaving Example

	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15				
Sync Hdr	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
Symbol0	PCIe STP Token				PCIe TLP Header DW0				PCIe TLP Header DW1				PCIe TLP Header DW2							
Symbol1	PCIe TLP Data Payload DW0				PCIe TLP Data Payload DW1				PCIe TLP Data Payload DW2				PCIe TLP CRC							
Symbol2	PCIe SDP Token		PCIe DLLP Payload				PCIe DLLP CRC		PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL				
Symbol3	PCIe STP Token				PCIe TLP Header DW0				PCIe TLP Header DW1				PCIe TLP Header DW2							
Symbol4	PCIe TLP Data Payload DW0				PCIe TLP Data Payload DW1				PCIe TLP Data Payload DW2				PCIe TLP Data Payload DW3							
Symbol5	PCIe TLP Data Payload DW4				PCIe TLP Data Payload DW5				PCIe TLP Data Payload DW6				PCIe TLP Data Payload DW7							
Symbol6	PCIe TLP Data Payload DW8				PCIe TLP CRC				PCIe SDP Token	PCIe DLLP Payload				PCIe DLLP CRC						
Symbol7	PCIe STP Token				PCIe TLP Header DW0				PCIe TLP Header DW1				PCIe TLP Header DW2							
Symbol8	PCIe TLP Data Payload DW0				PCIe TLP Data Payload DW1				PCIe TLP Data Payload DW2				PCIe TLP CRC							
Sync Hdr	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
Symbol0	ProtID[7:0]= CXLo	ProtID[15:8] =CXLo	PCIe STP Token				PCIe TLP Header DW0				PCIe TLP Header DW1				PCIe TLP Header DW2[15:0]					
Symbol1	PCIe TLP Header DW2[31:16]				PCIe TLP Data Payload DW0				PCIe TLP Data Payload DW1				PCIe TLP CRC							
Symbol2	PCIe TLP CRC		PCIe SDP Token		PCIe DLLP Payload				PCIe DLLP CRC		PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL				
Symbol3	PCIe IDL	PCIe IDL	PCIe STP Token				PCIe TLP Header DW0				PCIe TLP Header DW1				PCIe TLP Header DW2[15:0]					
Symbol4	PCIe TLP Header DW2[31:16]		Reserved	Reserved	ProtID[7:0]= CXLo	ProtID[15:8] =CXLo	PCIe TLP Data Payload DW0				PCIe TLP Data Payload DW1				PCIe TLP Data Payload DW2[15:0]					
Symbol5	PCIe TLP Data Payload DW2[31:16]		PCIe TLP Data Payload DW3				PCIe TLP Data Payload DW4				PCIe TLP Data Payload DW5				PCIe TLP Data Payload DW2[15:0]					
Symbol6	PCIe TLP Data Payload DW6[31:16]		PCIe TLP Data Payload DW7				PCIe TLP Data Payload DW8				PCIe TLP CRC				PCIe SDP Token					
Symbol7	PCIe DLLP Payload				PCIe DLLP CRC				PCIe STP Token				PCIe TLP Header DW0							
Symbol8	PCIe TLP Header DW1[31:16]				PCIe TLP Header DW2				Reserved	Reserved	ProtID[7:0]= CXLo, camem	ProtID[15:8]= CXLo, camem	Fit[7:0]	Fit[15:8]	Fit[23:16]	Fit[31:24]	Fit[39:32]	Fit[47:40]		
Symbol9	Fit[55:48]	Fit[63:56]	Fit[71:64]	Fit[79:72]	Fit[87:80]	Fit[95:88]	Fit[103:96]	Fit[111:104]	Fit[119:112]	Fit[127:120]	Fit[135:128]	Fit[143:136]	Fit[151:144]	Fit[159:152]	Fit[167:160]	Fit[175:168]				
Symbol10	Fit[183:176]	Fit[191:184]	Fit[199:192]	Fit[207:200]	Fit[215:208]	Fit[223:216]	Fit[231:224]	Fit[239:232]	Fit[247:240]	Fit[255:248]	Fit[263:256]	Fit[271:264]	Fit[279:272]	Fit[287:280]	Fit[295:288]	Fit[303:296]				
Symbol11	Fit[311:304]	Fit[319:312]	Fit[327:320]	Fit[335:328]	Fit[343:336]	Fit[351:344]	Fit[359:352]	Fit[367:360]	Fit[375:368]	Fit[383:376]	Fit[391:384]	Fit[399:392]	Fit[407:400]	Fit[415:408]	Fit[423:416]	Fit[431:424]				
Symbol12	Fit[439:432]	Fit[447:440]	Fit[455:448]	Fit[463:456]	Fit[471:464]	Fit[479:472]	Fit[487:480]	Fit[495:488]	Fit[503:496]	Fit[511:504]	CRC	CRC	ProtID[7:0]= CXLo, camem	ProtID[15:8]= CXLo, camem	Fit[7:0]	Fit[15:8]				
Symbol13	Fit[21:16]	Fit[31:24]	Fit[39:32]	Fit[47:40]	Fit[55:48]	Fit[63:56]	Fit[71:64]	Fit[79:72]	Fit[87:80]	Fit[95:88]	Fit[103:96]	Fit[111:104]	Fit[119:112]	Fit[127:120]	Fit[135:128]	Fit[143:136]				
Symbol14	Fit[151:144]	Fit[159:152]	Fit[167:160]	Fit[175:168]	Fit[183:176]	Fit[191:184]	Fit[199:192]	Fit[207:200]	Fit[215:208]	Fit[223:216]	Fit[231:224]	Fit[239:232]	Fit[247:240]	Fit[255:248]	Fit[263:256]	Fit[271:264]				
Symbol15	Fit[279:272]	Fit[287:280]	Fit[295:288]	Fit[303:296]	Fit[311:304]	Fit[319:312]	Fit[327:320]	Fit[335:328]	Fit[343:336]	Fit[351:344]	Fit[359:352]	Fit[367:360]	Fit[375:368]	Fit[383:376]	Fit[391:384]	Fit[399:392]				
Sync Hdr	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
Symbol0	Fit[407:400]	Fit[415:408]	Fit[423:416]	Fit[431:424]	Fit[439:432]	Fit[447:440]	Fit[455:448]	Fit[463:456]	Fit[471:464]	Fit[479:472]	Fit[487:480]	Fit[495:488]	Fit[503:496]	Fit[511:504]	CRC	CRC				
Symbol1	ProtID[7:0]= CXLo	ProtID[15:8] =CXLo	PCIe TLP Data Payload DW0				PCIe TLP Data Payload DW1				PCIe TLP Data Payload DW2				PCIe TLP CRC[15:0]					

6.2.2.3 x8 Packet Layout

Figure 6-4 shows the x8 packet layout. 16 bits of Protocol ID followed by a 528-bit flit are striped across the lanes on an 8-bit granularity. Depending on the symbol time, the Protocol ID plus flit transfer may start on Lane 0 or Lane 4. The pattern of transfer repeats after every 17 symbol times. The two-bit Sync Header shown in the figure is not present for the Sync Header bypass latency-optimized mode.

Figure 6-4. Flex Bus x8 Packet Layout

	L0	L1	L2	L3	L4	L5	L6	L7
Sync Hdr	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1
Symbol0	ProtID[7:0]	ProtID[15:8]	Flit[7:0]	Flit[15:8]	Flit[23:16]	Flit[31:24]	Flit[39:32]	Flit[47:40]
Symbol1	Flit[55:48]	Flit[63:56]	Flit[71:64]	Flit[79:72]	Flit[87:80]	Flit[95:88]	Flit[103:96]	Flit[111:104]
Symbol2	Flit[119:112]	Flit[127:120]	Flit[135:128]	Flit[143:136]	Flit[151:144]	Flit[159:152]	Flit[167:160]	Flit[175:168]
Symbol3	Flit[183:176]	Flit[191:184]	Flit[199:192]	Flit[207:200]	Flit[215:208]	Flit[223:216]	Flit[231:224]	Flit[239:232]
Symbol4	Flit[247:240]	Flit[255:248]	Flit[263:256]	Flit[271:264]	Flit[279:272]	Flit[287:280]	Flit[295:288]	Flit[303:296]
Symbol5	Flit[311:304]	Flit[319:312]	Flit[327:320]	Flit[335:328]	Flit[343:336]	Flit[351:344]	Flit[359:352]	Flit[367:360]
Symbol6	Flit[375:368]	Flit[383:376]	Flit[391:384]	Flit[399:392]	Flit[407:400]	Flit[415:408]	Flit[423:416]	Flit[431:424]
Symbol7	Flit[439:432]	Flit[447:440]	Flit[455:448]	Flit[463:456]	Flit[471:464]	Flit[479:472]	Flit[487:480]	Flit[495:488]
Symbol8	Flit[503:496]	Flit[511:504]	Flit[519:512]	Flit[527:520]	ProtID[7:0]	ProtID[15:8]	Flit[7:0]	Flit[15:8]
Symbol9	Flit[23:16]	Flit[31:24]	Flit[39:32]	Flit[47:40]	Flit[55:48]	Flit[63:56]	Flit[71:64]	Flit[79:72]
Symbol10	Flit[87:80]	Flit[95:88]	Flit[103:96]	Flit[111:104]	Flit[119:112]	Flit[127:120]	Flit[135:128]	Flit[143:136]
Symbol11	Flit[151:144]	Flit[159:152]	Flit[167:160]	Flit[175:168]	Flit[183:176]	Flit[191:184]	Flit[199:192]	Flit[207:200]
Symbol12	Flit[215:208]	Flit[223:216]	Flit[231:224]	Flit[239:232]	Flit[247:240]	Flit[255:248]	Flit[263:256]	Flit[271:264]
Symbol13	Flit[279:272]	Flit[287:280]	Flit[295:288]	Flit[303:296]	Flit[311:304]	Flit[319:312]	Flit[327:320]	Flit[335:328]
Symbol14	Flit[343:336]	Flit[351:344]	Flit[359:352]	Flit[367:360]	Flit[375:368]	Flit[383:376]	Flit[391:384]	Flit[399:392]
Symbol15	Flit[407:400]	Flit[415:408]	Flit[423:416]	Flit[431:424]	Flit[439:432]	Flit[447:440]	Flit[455:448]	Flit[463:456]
Sync Hdr	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1
Symbol0	Flit[471:464]	Flit[479:472]	Flit[487:480]	Flit[495:488]	Flit[503:496]	Flit[511:504]	Flit[519:512]	Flit[527:520]
Symbol1	ProtID[7:0]	ProtID[15:8]	Flit[7:0]	Flit[15:8]	Flit[23:16]	Flit[31:24]	Flit[39:32]	Flit[47:40]

Figure 6-5 illustrates how CXL.io and CXL.cachemem traffic is interleaved on a x8 Flex Bus link. The same traffic from the x16 example in Figure 6-3 is mapped to a x8 link.

EVALUATION COPY

Flex Bus Physical Layer

Figure 6-5. Flex Bus x8 Protocol Interleaving Example

	L0	L1	L2	L3	L4	L5	L6	L7
Sync Hdr	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1
Symbol0	ProtID[7:0]= CXL.io	ProtID[15:8] =CXL.io	PCIe STP Token					PCIe TLP Header DW0[15:0]
Symbol1	PCIe TLP Header DW0[31:16]					PCIe TLP Header DW1		
Symbol2	PCIe TLP Header DW2[31:16]					PCIe TLP Data Payload DW0		
Symbol3	PCIe TLP Data Payload DW1[31:16]					PCIe TLP Data Payload DW2		
Symbol4	PCIe TLP CRC		PCIe SDP Token			PCIe DLLP Payload		
Symbol5	PCIe DLLP CRC		PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL
Symbol6	PCIe IDL	PCIe IDL	PCIe STP Token					PCIe TLP Header DW0[15:0]
Symbol7	PCIe TLP Header DW0[31:16]					PCIe TLP Header DW1		
Symbol8	PCIe TLP Header DW2[31:16]					PCIe TLP Data Payload DW1		
Symbol9	PCIe TLP Data Payload DW0[31:16]					PCIe TLP Data Payload DW2		
Symbol10	PCIe TLP Data Payload DW2[31:16]					PCIe TLP Data Payload DW3		
Symbol11	PCIe TLP Data Payload DW4[31:16]					PCIe TLP Data Payload DW5		
Symbol12	PCIe TLP Data Payload DW6[31:16]					PCIe TLP Data Payload DW7		
Symbol13	PCIe TLP Data Payload DW8[31:16]					PCIe TLP CRC		
Symbol14	PCIe DLLP Payload					PCIe DLLP CRC		
Symbol15	PCIe STP Token[31:16]			PCIe TLP Header DW0				PCIe TLP Header DW1[15:0]
Sync Hdr	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1
Symbol0	PCIe TLP Header DW1[31:16]					PCIe TLP Header DW2		Reserved
Symbol1	ProtID[7:0]= CXL.camem	ProtID[15:8] =CXL.camem	Flit[7:0]	Flit[15:8]	Flit[23:16]	Flit[31:24]	Flit[39:32]	Flit[47:40]
Symbol2	Flit[55:48]	Flit[63:56]	Flit[7:64]	Flit[79:72]	Flit[87:80]	Flit[95:88]	Flit[103:96]	Flit[111:104]
Symbol3	Flit[119:112]	Flit[127:120]	Flit[135:128]	Flit[143:136]	Flit[151:144]	Flit[159:152]	Flit[167:160]	Flit[175:168]
Symbol4	Flit[183:176]	Flit[191:184]	Flit[199:192]	Flit[207:200]	Flit[215:208]	Flit[223:216]	Flit[231:224]	Flit[239:232]
Symbol5	Flit[247:240]	Flit[255:248]	Flit[263:256]	Flit[271:264]	Flit[279:272]	Flit[287:280]	Flit[295:288]	Flit[303:296]
Symbol6	Flit[311:304]	Flit[319:312]	Flit[327:320]	Flit[335:328]	Flit[343:336]	Flit[351:344]	Flit[359:352]	Flit[367:360]
Symbol7	Flit[375:368]	Flit[383:376]	Flit[391:384]	Flit[399:392]	Flit[407:400]	Flit[415:408]	Flit[423:416]	Flit[431:424]
Symbol8	Flit[439:432]	Flit[447:440]	Flit[455:448]	Flit[463:456]	Flit[471:464]	Flit[479:472]	Flit[487:480]	Flit[495:488]
Symbol9	Flit[503:496]	Flit[511:504]	CRC		ProtID[7:0]= CXL.camem	ProtID[15:8] =CXL.camem	Flit[7:0]	Flit[15:8]
Symbol10	Flit[23:16]	Flit[31:24]	Flit[39:32]	Flit[47:40]	Flit[55:48]	Flit[63:56]	Flit[71:64]	Flit[79:72]
Symbol11	Flit[87:80]	Flit[95:88]	Flit[103:96]	Flit[111:104]	Flit[119:112]	Flit[127:120]	Flit[135:128]	Flit[143:136]
Symbol12	Flit[151:144]	Flit[159:152]	Flit[167:160]	Flit[175:168]	Flit[183:176]	Flit[191:184]	Flit[199:192]	Flit[207:200]
Symbol13	Flit[215:208]	Flit[223:216]	Flit[231:224]	Flit[239:232]	Flit[247:240]	Flit[255:248]	Flit[263:256]	Flit[271:264]
Symbol14	Flit[279:272]	Flit[287:280]	Flit[295:288]	Flit[303:296]	Flit[311:304]	Flit[319:312]	Flit[327:320]	Flit[335:328]
Symbol15	Flit[343:336]	Flit[351:344]	Flit[359:352]	Flit[367:360]	Flit[375:368]	Flit[383:376]	Flit[391:384]	Flit[399:392]
Sync Hdr	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1
Symbol0	Flit[407:400]	Flit[415:408]	Flit[423:416]	Flit[431:424]	Flit[439:432]	Flit[447:440]	Flit[455:448]	Flit[463:456]
Symbol1	Flit[471:464]	Flit[479:472]	Flit[487:480]	Flit[495:488]	Flit[503:496]	Flit[511:504]	CRC	CRC
Symbol2	ProtID[7:0]= CXL.io	ProtID[15:8] =CXL.io	PCIe TLP Data Payload DW0					PCIe TLP Data Payload DW1[15:0]
Symbol3	PCIe TLP Data Payload DW1[31:16]					PCIe TLP Data Payload DW2		

6.2.2.4 x4 Packet Layout

Figure 6-6 shows the x4 packet layout. 16 bits of Protocol ID followed by a 528-bit flit are striped across the lanes on an 8-bit granularity. The Protocol ID plus flit transfer always starts on Lane 0; the entire transfer takes 17 symbol times. The two-bit Sync Header shown in the figure is not present for Latency-optimized Sync Header Bypass mode.

Figure 6-6. Flex Bus x4 Packet Layout

	L0	L1	L2	L3
Sync Hdr	0	0	0	0
	1	1	1	1
Symbol0	ProtID[7:0]	ProtID[15:8]	Flit[7:0]	Flit[15:8]
Symbol1	Flit[23:16]	Flit[31:24]	Flit[39:32]	Flit[47:40]
Symbol2	Flit[55:48]	Flit[63:56]	Flit[71:64]	Flit[79:72]
Symbol3	Flit[87:80]	Flit[95:88]	Flit[103:96]	Flit[111:104]
Symbol4	Flit[119:112]	Flit[127:120]	Flit[135:128]	Flit[143:136]
Symbol5	Flit[151:144]	Flit[159:152]	Flit[167:160]	Flit[175:168]
Symbol6	Flit[183:176]	Flit[191:184]	Flit[199:192]	Flit[207:200]
Symbol7	Flit[215:208]	Flit[223:216]	Flit[231:224]	Flit[239:232]
Symbol8	Flit[247:240]	Flit[255:248]	Flit[263:256]	Flit[271:264]
Symbol9	Flit[279:272]	Flit[287:280]	Flit[295:288]	Flit[303:296]
Symbol10	Flit[311:304]	Flit[319:312]	Flit[327:320]	Flit[335:328]
Symbol11	Flit[343:336]	Flit[351:344]	Flit[359:352]	Flit[367:360]
Symbol12	Flit[375:368]	Flit[383:376]	Flit[391:384]	Flit[399:392]
Symbol13	Flit[407:400]	Flit[415:408]	Flit[423:416]	Flit[431:424]
Symbol14	Flit[439:432]	Flit[447:440]	Flit[455:448]	Flit[463:456]
Symbol15	Flit[471:464]	Flit[479:472]	Flit[487:480]	Flit[495:488]
Sync Hdr	0	0	0	0
	1	1	1	1
Symbol0	Flit[503:496]	Flit[511:504]	Flit[519:512]	Flit[527:520]
Symbol1	ProtID[7:0]	ProtID[15:8]	Flit[7:0]	Flit[15:8]

6.2.2.5 x2 Packet Layout

The x2 packet layout looks similar to the x4 packet layout in that the Protocol ID aligns to Lane 0. 16 bits of Protocol ID followed by a 528-bit flit are striped across two lanes on an 8-bit granularity, taking 34 symbol times to complete the transfer.

6.2.2.6 x1 Packet Layout

The x1 packet layout is used only in degraded mode. The 16 bits of Protocol ID followed by 528-bit flit are transferred on a single lane, taking 68 symbol times to complete the transfer.

6.2.2.7

Special Case: CXL.io – When a TLP Ends on a Flit Boundary

For CXL.io traffic, if a TLP ends on a flit boundary and there is no additional CXL.io traffic to send, the receiver still requires a subsequent EDB indication if it is a nullified TLP or all IDLE flit or a DLLP to confirm it is a good TLP before processing the TLP.

Figure 6-7 illustrates a scenario where the first CXL.io flit encapsulates a TLP that ends at the flit boundary, and the transmitter has no more TLPs or DLLPs to send. To ensure that the transmitted TLP that ended on the flit boundary is processed by the receiver, a subsequent CXL.io flit containing PCIe IDLE tokens is transmitted. The Link Layer generates the subsequent CXL.io flit.

Figure 6-7. CXL.io TLP Ending on Flit Boundary Example

	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15
Sync Hdr	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Symbol0	ProtID[7:0]= CXL.io	ProtID[5:8]= CXL.io	PCIe STP Token		PCIe TLP Header DW0		PCIe TLP Header DW1		PCIe TLP Header DW1		PCIe TLP Header DW15:0]		PCIe TLP Header DW15:0]		PCIe TLP Header DW15:0]	
Symbol1	PCIe TLP Header DW2[31:16]		PCIe TLP Data Payload DW0		PCIe TLP Data Payload DW1		PCIe TLP Data Payload DW2		PCIe TLP Data Payload DW2		PCIe TLP Data Payload DW3[31:16]		PCIe TLP Data Payload DW3[31:16]		PCIe TLP Data Payload DW3[31:16]	
Symbol2	PCIe TLP Data Payload DW3[31:16]		PCIe TLP Data Payload DW4		PCIe TLP Data Payload DW5		PCIe TLP Data Payload DW6		PCIe TLP Data Payload DW6		PCIe TLP Header DW7[15:0]		PCIe TLP Header DW7[15:0]		PCIe TLP Header DW7[15:0]	
Symbol3	PCIe TLP Header DW7[7:1]		PCIe TLP Data Payload DW8		PCIe TLP Data Payload DW9		PCIe TLP Data Payload DW10		PCIe TLP Data Payload DW10		PCIe TLP LRC[15:0]		PCIe TLP LRC[15:0]		PCIe TLP LRC[15:0]	
Symbol4	PCIe TLP CRC[31:16]	Reserved	Reserved	ProtID[7:0]= CXL.io	ProtID[5:8]= CXL.io	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	
Symbol5	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	
Symbol6	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	
Symbol7	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	
Symbol8	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	PCIe IDL	
Symbol9	Flit[55:48]	Flit[63:56]	Flit[71:64]	Flit[79:72]	Flit[87:80]	Flit[85:88]	Flit[103:96]	Flit[111:104]	Flit[119:112]	Flit[127:120]	Flit[135:128]	Flit[143:136]	Flit[151:144]	Flit[159:152]	Flit[167:160]	
Symbol10	Flit[183:176]	Flit[191:184]	Flit[199:192]	Flit[207:200]	Flit[215:208]	Flit[223:216]	Flit[231:224]	Flit[239:232]	Flit[247:240]	Flit[255:248]	Flit[263:256]	Flit[271:264]	Flit[279:272]	Flit[287:280]	Flit[295:288]	Flit[303:296]
Symbol11	Flit[311:304]	Flit[319:312]	Flit[327:320]	Flit[335:320]	Flit[343:330]	Flit[351:344]	Flit[359:352]	Flit[367:360]	Flit[375:356]	Flit[383:376]	Flit[391:384]	Flit[399:392]	Flit[407:400]	Flit[415:408]	Flit[423:416]	Flit[431:424]
Symbol12	Flit[439:432]	Flit[447:440]	Flit[455:448]	Flit[463:456]	Flit[471:464]	Flit[479:472]	Flit[487:480]	Flit[495:488]	Flit[503:496]	Flit[511:504]	CRC	CRC	ProtID[7:0]= CXL.Lamem	ProtID[5:8]= CXL.Lamem	Flit[7:0]	
Symbol13	Flit[23:16]	Flit[31:24]	Flit[39:32]	Flit[47:40]	Flit[55:48]	Flit[63:56]	Flit[71:64]	Flit[79:72]	Flit[87:80]	Flit[85:88]	Flit[103:96]	Flit[111:104]	Flit[119:112]	Flit[127:120]	Flit[135:128]	Flit[143:136]
Symbol14	Flit[151:144]	Flit[159:152]	Flit[167:160]	Flit[175:168]	Flit[183:176]	Flit[191:184]	Flit[199:192]	Flit[207:200]	Flit[215:208]	Flit[223:216]	Flit[231:224]	Flit[239:232]	Flit[247:240]	Flit[255:248]	Flit[263:256]	Flit[271:264]
Symbol15	Flit[279:272]	Flit[287:280]	Flit[295:288]	Flit[303:296]	Flit[311:304]	Flit[319:312]	Flit[327:320]	Flit[335:320]	Flit[343:330]	Flit[351:344]	Flit[359:352]	Flit[367:360]	Flit[375:368]	Flit[383:376]	Flit[391:384]	Flit[399:392]
Sync Hdr	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Symbol0	Flit[407:400]	Flit[415:408]	Flit[423:416]	Flit[431:424]	Flit[439:432]	Flit[447:440]	Flit[455:448]	Flit[463:456]	Flit[471:464]	Flit[479:472]	Flit[487:480]	Flit[495:488]	Flit[503:496]	Flit[511:504]	CRC	CRC

6.2.2.8

Framing Errors

The physical layer is responsible for detecting framing errors and, subsequently, for initiating entry into recovery to retrain the link.

The following are framing errors detected by the physical layer:

- Sync Header errors
- Protocol ID framing errors
- EDS insertion errors
- PCIe framing errors located within the 528-bit CXL.io flit

Protocol ID framing errors are described in [Section 6.2.2](#) and summarized in [Table 6-3](#). A Protocol ID with a value that is defined in the CXL specification is considered a valid Protocol ID. A valid Protocol ID is either expected or unexpected. An expected Protocol ID is one that corresponds to a protocol that was enabled during negotiation. An unexpected Protocol ID is one that corresponds to a protocol that was not enabled during negotiation. A Protocol ID with a value that is not defined in the CXL specification is considered an invalid Protocol ID. Whenever a flit is dropped by the physical layer due to either an Unexpected Protocol ID Framing Error or an

Uncorrectable Protocol ID Framing Error, the physical layer enters LTSSM recovery to retrain the link and notifies the link layers to enter recovery and, if applicable, to initiate link level retry.

Table 6-3. Protocol ID Framing Errors

Protocol ID[7:0]	Protocol ID[15:8]	Expected Action
Invalid	Valid & Expected	Process normally using Protocol ID[15:8]; Log as CXL_Correctable_Protocol_ID_Framing_Error in DVSEC Flex Bus Port Status register.
Valid & Expected	Invalid	Process normally using Protocol ID[7:0]; Log as CXL_Correctable_Protocol_ID_Framing_Error in DVSEC Flex Bus Port Status register.
Valid & Unexpected	Valid & Unexpected & Equal to Protocol ID[7:0]	Drop flit and log as CXL_Unexpected_Protocol_ID_Dropped in DVSEC Flex Bus Port Status register; enter LTSSM recovery to retrain the link; notify link layers to enter recovery and, if applicable, initiate link level retry
Invalid	Valid & Unexpected	Drop flit and log as CXL_Unexpected_Protocol_ID_Dropped in DVSEC Flex Bus Port Status register; enter LTSSM recovery to retrain the link; notify link layers to enter recovery and, if applicable, initiate link level retry
Valid & Unexpected	Invalid	Drop flit and log as CXL_Unexpected_Protocol_ID_Dropped in DVSEC Flex Bus Port Status register; enter LTSSM recovery to retrain the link; notify link layers to enter recovery and, if applicable, initiate link level retry
Valid	Valid & Not Equal to Protocol ID[7:0]	Drop flit and log as CXL_Uncorrectable_Protocol_ID_Framing_Error in DVSEC Flex Bus Port Status register; enter LTSSM recovery to retrain the link; notify link layers to enter recovery and, if applicable, initiate link level retry
Invalid	Invalid	Drop flit and log as CXL_Uncorrectable_Protocol_ID_Framing_Error in DVSEC Flex Bus Port Status register; enter LTSSM recovery to retrain the link; notify link layers to enter recovery and, if applicable, initiate link level retry

6.2.3 256B Flit Mode

256B Flit mode operation relies on support of PCIe Base Specification. Selection of 68B Flit mode or 256B Flit mode occurs during PCIe link training. **Table 6-4** specifies the scenarios in which the link operates in 68B Flit mode and 256B Flit mode. CXL mode is supported at PCIe link rates of 8 GT/s or higher; CXL mode is not supported at 2.5 GT/s or 5 GT/s link rates, regardless of whether PCIe Flit mode is negotiated. If PCIe Flit mode is selected during training, as described in PCIe Base Specification, and the link speed is 8 GT/s or higher, 256B Flit mode is used. If PCIe Flit mode is not selected during training and the link speed is 8 GT/s or higher, 68B Flit mode is used.

Table 6-4. 256B Flit Mode vs. 68B Flit Mode Operation

Data Rate	PCIe Flit Mode	Encoding	CXL Flit Mode
2.5 GT/s, 5 GT/s	No	8b/10b	CXL is not supported
2.5 GT/s, 5 GT/s	Yes	8b/10b	CXL is not supported
8 GT/s, 16 GT/s, 32 GT/s	No	128b/130b	68B flits
8 GT/s, 16 GT/s, 32 GT/s	Yes	128b/130b	256B flits
64 GT/s	Yes	1b/1b	256B flits

6.2.3.1 256B Flit Format

The 256B flit leverages several elements from the PCIe flit. There are two variants of the 256B flit:

- Standard 256B flit
- Latency-optimized 256B flit with 128-byte flit halves

6.2.3.1.1 Standard 256B Flit

The standard 256B flit format is shown in [Figure 6-8](#). The 256-byte flit includes 2 bytes of Flit Header information as specified in [Table 6-5](#). There are 240 bytes of Flit Data, for which the format differs depending on whether the flit carries CXL.io payload, CXL.cachemem payload, or ALMP payload, or whether an IDLE flit is being transmitted. For CXL.io, the Flit Data includes TLP payload and a 4-byte DLLP payload as specified in PCIe Base Specification; the DLLP payload is located at the end of the Flit Data as shown in [Figure 6-9](#). For CXL.cachemem, the Flit Data format is specified in [Chapter 4.0](#). The 8 bytes of CRC protects the Flit Header and Flit Data and is calculated as specified in PCIe Base Specification. The 6 bytes of FEC protects the Flit Header, Flit Data, and CRC, and is calculated as specified in PCIe Base Specification. The application of flit bits to the PCIe physical lanes is shown in [Figure 6-10](#).

Figure 6-8. Standard 256B Flit

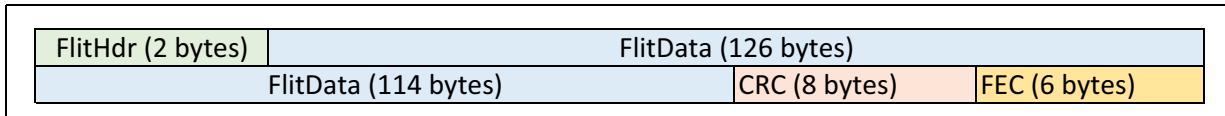
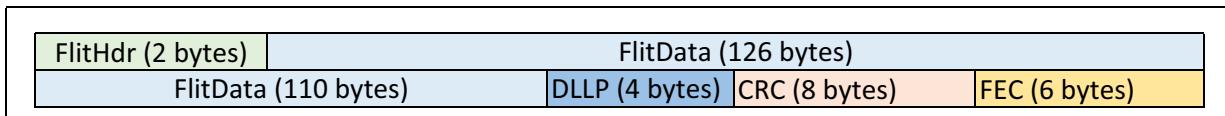


Figure 6-9. CXL.io Standard 256B Flit



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 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.

Table 6-5. 256B Flit Header

Flit Header Field	Flit Header Bit Location	Description
Flit Type[1:0]	[7:6]	<ul style="list-style-type: none"> • 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]	<ul style="list-style-type: none"> • 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]	<ul style="list-style-type: none"> • 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.

6.2.3.1.1.1 256B Flit Type

[Table 6-6](#) specifies the different flit payloads that are associated with each Flit Type encoding.

Table 6-6. 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
01b	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
	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.	Yes, allocate only to the Tx Retry Buffer
11b	ALMP	ARB/MUX	ARB/MUX Link Management Packet.	Yes

Prior to the Sequence Number Handshake upon each entry to L0, as described in PCIe Base Specification, a Flit Type encoding of 00b indicates an IDLE flit. These IDLE flits contain all zeros payload and are generated and consumed by the Physical Layer.

During and after the Sequence Number Handshake in L0, a Flit Type encoding of 00b indicates either a Physical Layer NOP flit or a CXL.io NOP flit. The Physical Layer must insert NOP flits when it backpressures the upper layers due to its Tx retry buffer filling up; it is also required to insert NOP flits when traffic is not generated by the upper layers. These NOP flits must not be allocated into the transmit retry buffer or receive retry buffer. Physical Layer NOP flits carry 0s in the bit positions that correspond to the bit positions in CXL.io flits that are used to carry DLLP payload; the remaining bits in Physical Layer NOP flits are reserved.

CXL.io NOP flits are generated by the CXL.io Link Layer and carry only valid DLLP payload. When a Flit Type of 00b is decoded, the Physical Layer must always check for valid DLLP payload. CXL.io NOP flits must not be allocated into the transmit retry buffer or into the receive buffer.

A Flit Type encoding of 01b indicates CXL.io Payload traffic; these flits can encapsulate both valid TLP payload and valid DLLP payload.

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 retryable and must be allocated in the transmit retry buffer. The Physical Layer must decode the Link Layer CRD[4:0] bits to determine whether the flit carries valid payload or whether the flit is an empty CXL.cachemem Empty flit. See [Table 4-19](#) in [Chapter 4.0](#) for more details.

A Flit Type encoding of 11b indicates an ALMP.

[Figure 6-10](#) shows how the flit is mapped to the physical lanes on the link. The flit is striped across the lanes on an 8-bit granularity starting with 16-bit Flit Header, followed by the 240 bytes of Flit Data, the 8-byte CRC, and finally the 6-byte FEC (3-way interleaved ECC described in PCIe Base Specification).

Figure 6-10. Standard 256B Flit Applied to Physical Lanes (x16)

	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15
Symbol0	FltHd[7:0]	FltHd[15:8]	FltD[7:0]	FltD[15:8]	FltD[23:16]	FltD[31:24]	FltD[39:32]	FltD[47:40]	FltD[55:48]	FltD[63:56]	FltD[71:64]	FltD[79:72]	FltD[87:80]	FltD[95:88]	FltD[103:96]	FltD[111:104]
Symbol1	FltD[119:112]	FltD[127:120]	FltD[135:128]	FltD[145:136]	FltD[151:144]	FltD[159:152]	FltD[167:160]	FltD[175:168]	FltD[183:176]	FltD[191:184]	FltD[199:192]	FltD[207:200]	FltD[215:208]	FltD[223:216]	FltD[231:224]	FltD[239:232]
Symbol2	FltD[247:240]	FltD[255:248]	FltD[263:256]	FltD[271:264]	FltD[279:272]	FltD[287:280]	FltD[295:288]	FltD[303:296]	FltD[311:304]	FltD[319:312]	FltD[327:320]	FltD[335:328]	FltD[343:336]	FltD[351:344]	FltD[359:352]	FltD[367:360]
Symbol3	FltD[375:368]	FltD[383:376]	FltD[391:384]	FltD[399:392]	FltD[407:400]	FltD[415:408]	FltD[423:416]	FltD[431:424]	FltD[439:432]	FltD[447:440]	FltD[455:448]	FltD[463:456]	FltD[471:464]	FltD[479:472]	FltD[487:480]	FltD[495:488]
Symbol4	FltD[503:496]	FltD[511:504]	FltD[519:512]	FltD[527:524]	FltD[535:528]	FltD[543:536]	FltD[551:544]	FltD[559:552]	FltD[567:560]	FltD[575:568]	FltD[583:576]	FltD[591:584]	FltD[599:592]	FltD[607:600]	FltD[615:608]	FltD[623:616]
Symbol5	FltD[631:624]	FltD[639:632]	FltD[647:640]	FltD[655:648]	FltD[663:656]	FltD[671:664]	FltD[679:672]	FltD[687:680]	FltD[695:688]	FltD[703:696]	FltD[711:704]	FltD[719:712]	FltD[727:720]	FltD[735:728]	FltD[743:736]	FltD[751:744]
Symbol6	FltD[759:752]	FltD[767:760]	FltD[775:768]	FltD[783:776]	FltD[791:784]	FltD[799:792]	FltD[807:800]	FltD[815:808]	FltD[823:816]	FltD[831:824]	FltD[839:832]	FltD[847:840]	FltD[855:848]	FltD[863:856]	FltD[871:864]	FltD[879:872]
Symbol7	FltD[887:880]	FltD[895:888]	FltD[903:896]	FltD[911:904]	FltD[919:912]	FltD[927:920]	FltD[935:928]	FltD[943:930]	FltD[951:944]	FltD[959:952]	FltD[967:960]	FltD[975:968]	FltD[983:976]	FltD[991:984]	FltD[999:992]	FltD[1007:1000]
Symbol8	FltD[1015:1008]	FltD[1023:1016]	FltD[1031:1024]	FltD[1039:1032]	FltD[1047:1040]	FltD[1055:1058]	FltD[1063:1064]	FltD[1071:1072]	FltD[1079:1072]	FltD[1087:1080]	FltD[1095:1088]	FltD[1103:1096]	FltD[1119:1112]	FltD[1127:1120]	FltD[1135:1128]	
Symbol9	FltD[1143:1136]	FltD[1151:1144]	FltD[1159:1152]	FltD[1167:1160]	FltD[1175:1168]	FltD[1183:1176]	FltD[1191:1184]	FltD[1199:1192]	FltD[1207:1200]	FltD[1215:1208]	FltD[1223:1216]	FltD[1231:1224]	FltD[1239:1232]	FltD[1247:1240]	FltD[1255:1248]	FltD[1263:1256]
Symbol10	FltD[1271:1264]	FltD[1279:1272]	FltD[1287:1280]	FltD[1295:1288]	FltD[1303:1296]	FltD[1311:1304]	FltD[1319:1312]	FltD[1327:1320]	FltD[1335:1328]	FltD[1343:1336]	FltD[1351:1344]	FltD[1359:1352]	FltD[1367:1360]	FltD[1375:1368]	FltD[1383:1376]	FltD[1391:1384]
Symbol11	FltD[1399:1392]	FltD[1407:1400]	FltD[1415:1408]	FltD[1423:1416]	FltD[1431:1424]	FltD[1439:1432]	FltD[1447:1440]	FltD[1455:1448]	FltD[1463:1456]	FltD[1471:1464]	FltD[1479:1472]	FltD[1487:1480]	FltD[1495:1488]	FltD[1503:1496]	FltD[1511:1504]	FltD[1519:1512]
Symbol12	FltD[1527:1520]	FltD[1535:1528]	FltD[1543:1536]	FltD[1551:1544]	FltD[1559:1552]	FltD[1567:1560]	FltD[1575:1568]	FltD[1583:1576]	FltD[1591:1584]	FltD[1599:1592]	FltD[1607:1600]	FltD[1615:1608]	FltD[1623:1616]	FltD[1631:1624]	FltD[1639:1632]	FltD[1647:1640]
Symbol13	FltD[1655:1648]	FltD[1663:1656]	FltD[1671:1664]	FltD[1679:1672]	FltD[1687:1680]	FltD[1695:1688]	FltD[1703:1696]	FltD[1711:1704]	FltD[1719:1722]	FltD[1727:1728]	FltD[1735:1728]	FltD[1743:1736]	FltD[1751:1744]	FltD[1759:1752]	FltD[1767:1760]	FltD[1775:1768]
Symbol14	FltD[1783:1776]	FltD[1791:1784]	FltD[1799:1792]	FltD[1807:1800]	FltD[1815:1808]	FltD[1823:1816]	FltD[1831:1824]	FltD[1839:1832]	FltD[1847:1840]	FltD[1855:1848]	FltD[1863:1856]	FltD[1871:1864]	FltD[1879:1872]	FltD[1887:1880]	FltD[1895:1888]	FltD[1903:1896]
Symbol15	FltD[1911:1904]	FltD[1919:1912]	CRC0	CRC1	CRC2	CRC3	CRC4	CRC5	CRC6	CRC7	ECC 0A	ECC 0B	ECC 0C	ECC 1A	ECC 1B	ECC 2B

6.2.3.1.2 Latency-Optimized 256B Flit with 128-Byte Flit Halves

Figure 6-11 shows the latency-optimized 256B flit format. This latency-optimized flit format is optionally supported by components that support 256B flits. The decision to operate in standard 256B flit format or the latency-optimized 256B flit format occurs once during CXL alternate protocol negotiation; dynamic switching between the two formats is not supported.

Figure 6-11. Latency-Optimized 256B Flit

FlitHdr (2 bytes)	FlitData (120 bytes)	CRC (6 bytes)
	FlitData (116 bytes)	FEC (6 bytes) CRC (6 bytes)

The latency-optimized flit format organizes the 256-byte flit into 128-byte flit halves. The even flit half consists of the 2-byte Flit Header, 120 bytes of Flit Data, and 6 bytes of CRC that protects the even 128-byte flit half. The odd flit half consists of 116 bytes of Flit Data, 6 bytes of FEC that protects the entire 256 bytes of the flit, and 6 bytes of CRC that protects the 128-byte odd flit half excluding the 6-byte FEC. The benefit of the latency-optimized flit format is reduction of flit accumulation latency. Because each 128-byte flit half is independently protected by CRC, the first half of the flit can be consumed by the receiver if CRC passes without waiting for the second half to be received for FEC decode. The flit accumulation latency savings increases for smaller link widths; for x4 link widths the round trip flit accumulation latency is 8 ns at 64 GT/s link speed. Similarly, the odd flit half can be consumed if CRC passes, without having to wait for the more-complex FEC decode operation to first complete. If CRC fails for either flit half, FEC decode and correct is applied to the entire 256-byte flit. Subsequently, each flit half is consumed if CRC passes, if the flit half was not already previously consumed, and if all data previous to the flit half has been consumed.

Note:

For CXL.io, due to the potential of a Flit Marker, the last TLP of the first 128-byte flit half is not permitted to be consumed until the entire flit is successfully received.

Additionally, the CXL.io Transaction Layer must wait until the second half of the flit is successfully received before responding to a PTM request that was received in the first half of the flit so that it responds with the correct master timestamp.

Note that flits are still retried on a 256-byte granularity even with the latency-optimized 256-byte flit. If either flit half fails CRC after FEC decode and correct, the receiver requests a retry of the entire 256-byte flit. The receiver is responsible for tracking whether it has previously consumed either half during a retry and must drop any flit halves that have been previously consumed.

The following error scenario example illustrates how latency-optimized flits are processed. The even flit half passes CRC check prior to FEC decode and is consumed. The odd flit half fails CRC check. The FEC decode and correct is applied to the 256-byte flit; subsequently, the even flit half now fails CRC and the odd flit half passes. In this scenario, the FEC correction is suspect since a previously passing CRC check now fails. The receiver requests a retry of the 256-byte flit, and the odd flit half is consumed from the retransmitted flit, assuming it passes FEC and CRC checks. Note that even though the even flit half failed CRC post-FEC correction in the original flit, the receiver must not re-consume the even flit half from the retransmitted flit. The expectation is that this scenario occurs most likely due to multiple errors in the odd flit half exceeding FEC correction capability, thus causing additional errors to be injected due to FEC correction.

Table 6-7 summarizes processing steps for different CRC scenarios, depending on results of the CRC check for the even flit half and the odd flit half on the original flit, the post-FEC corrected flit, and the retransmitted flit.

Table 6-7. Latency-Optimized Flit Processing for CRC Scenarios (Sheet 1 of 2)

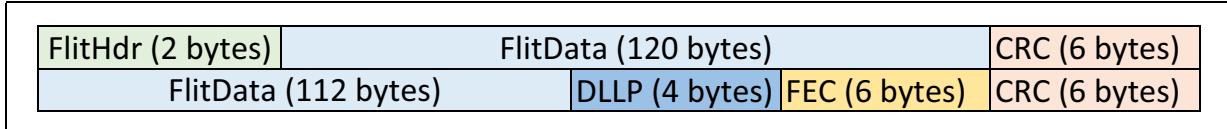
Original Flit			Post-FEC Corrected Flit			Retransmitted Flit			
Even CRC	Odd CRC	Action	Even CRC	Odd CRC	Subsequent Action	Even CRC	Odd CRC	Subsequent Action	
Pass	Pass	Consume Flit	N/A	N/A	N/A	N/A	N/A	N/A	
Pass	Fail	Permitted to consume even flit half; perform FEC decode and correct	Pass	Pass	Consume 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	
					Permitted to consume even flit half if not previously consumed; Request Retry			Permit to consume even flit half if not previously consumed (must drop even flit half if previously consumed); Perform FEC decode and correct	
			Pass	Fail		Pass	Pass		
			Fail	Pass/Fail	Request Retry; Log error for even flit half if previously consumed ¹	Pass	Pass	Permit to consume even flit half if not previously consumed (must drop even flit half if previously consumed); Consume odd flit half	
						Pass	Fail		
						Fail	Pass	Perform FEC decode and correct and evaluate next steps	
Fail	Pass	Perform FEC decode and correct	Pass	Fail	Permit to consume even flit half; Request Retry	Pass	N/A	N/A	
						Pass	Pass	Consume even flit half if not previously consumed (must drop even flit half if previously consumed); Consume odd flit half	
						Pass	Fail	Permit to consume even flit half if not previously consumed; Perform FEC decode and correct and evaluate next steps	
						Fail	Pass	Perform FEC decode and correct and evaluate next steps	
			Fail	Pass/Fail	Request Retry	Pass	Pass	Consume flit	

Table 6-7. Latency-Optimized Flit Processing for CRC Scenarios (Sheet 2 of 2)

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	
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	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	

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.

For CXL.io, the Flit Data includes TLP and DLLP payload; the 4-bytes of DLLP are transferred just before the FEC in the flit as shown in [Figure 6-12](#).

Figure 6-12. CXL.io Latency-Optimized 256B Flit

6.2.3.1.2.1 Latency-Optimized Flit 6-Byte CRC Calculation

The 6-Byte CRC is chosen to optimize the data path as well as allow reuse of the 8-Byte CRC logic from PCIe to save area. The CRC for the even flit half is calculated independent of the calculation for the odd flit half.

A (130, 136) Reed-Solomon code is used, where six bytes of CRC are generated over a 130-byte message to generate a 136-byte codeword. For the even flit half, bytes 0 to 121 of the message are the 122 non-CRC bytes of the flit (with byte 0 of flit mapping to byte 0 of the message, byte 1 of the flit mapping to byte 1 of the message and so on), whereas bytes 122 to 129 are zero (these are not sent on the link, but both transmitter and receiver must zero pad the remaining bytes before computing CRC). For the odd flit half, bytes 0 to 115 of the message are the 116 non-CRC and non-FEC bytes of the flit (with byte 128 of the flit mapping to byte 0 of the message, byte 129 of the flit mapping to byte 1 of the message and so on), whereas bytes 116 to 129 of the message are zero.

The CRC generator polynomial defined over GF(2⁸), is $g(x) = (x + \alpha)(x + \alpha^2)\dots(x + \alpha^6)$, where α is the root of the primitive polynomial of degree 8: $x^8 + x^5 + x^3 + x + 1$. Thus, $g(x) = x^6 + \alpha^{147}x^5 + \alpha^{107}x^4 + \alpha^{250}x^3 + \alpha^{114}x^2 + \alpha^{161}x + \alpha^{21}$.

When reusing the PCIe logic of 8B CRC generation, the first step is to generate the 8-Byte CRC from the PCIe logic. The flit bytes must be mapped to a specific location within the 242 bytes of input to the PCIe logic of 8B CRC generation.

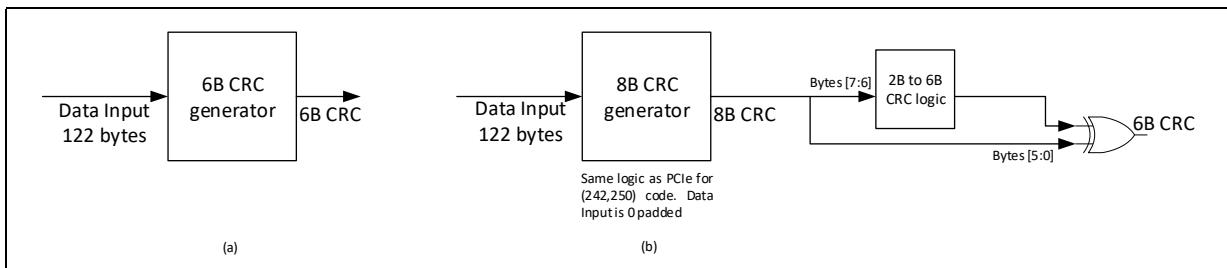
Table 6-8. Byte Mapping for Input to PCIe 8B CRC Generation

PCIe CRC Input Bytes	Even Flit Half Mapping	Odd Flit Half Mapping
Byte 0 to Byte 113	00h for all bytes	00h for all bytes
Byte 114 to Byte 229	Byte 0 to Byte 115 of the flit	Byte 128 to Byte 243 of the flit
Byte 230 to Byte 235	Byte 116 to Byte 121 of the flit	00h for all bytes
Byte 236 to Byte 241	00h for all bytes	00h for all bytes

If the polynomial form of the result is: $r'(x) = r_7x^7 + r_6x^6 + r_5x^5 + r_4x^4 + r_3x^3 + r_2x^2 + r_1x + r_0$, then the 6-Byte CRC can be computed using the following (equation shows the polynomial form of the 6-Byte CRC):

$$\begin{aligned} r(x) = & (r_5 + \alpha^{147}r_6 + \alpha^{90}r_7)x^5 + (r_4 + \alpha^{107}r_6 + \alpha^{202}r_7)x^4 + \\ & (r_3 + \alpha^{250}r_6 + \alpha^{41}r_7)x^3 + (r_2 + \alpha^{114}r_6 + \alpha^{63}r_7)x^2 + \\ & (r_1 + \alpha^{161}r_6 + \alpha^{147}r_7)x + (r_0 + \alpha^{21}r_6 + \alpha^{168}r_7) \end{aligned}$$

Figure 6-13 shows the two concepts of computing the 6-Byte CRC.

Figure 6-13. Different Methods for Generating 6-Byte CRC

The following are provided as attachments to the CXL specification:

- 6B CRC generator matrix (see PCIe Base Specification for the 8B CRC generator matrix).
- 6B CRC Register Transfer Level (RTL) code (see PCIe Base Specification for the 8B CRC RTL code). A single module with 122 bytes of input and 128 bytes of output is provided and can be used for both the even flit half and the odd flit half (by assigning bytes 116 to 121 of the input to be 00h for the odd flit half).
- 8B CRC to 6B CRC converter RTL code. A single module with 122 bytes of input and 128 bytes of output is provided and can be used for both the even flit half and the odd flit half (by assigning bytes 116 to 121 of the input to be 00h for the odd flit half).

6.2.3.2 CRC Corruption for Containment with 256B Flits

CXL has multiple scenarios that require CRC to be intentionally corrupted during transmission of a 256B Flit to force the receiver to reject the Flit and initiate a replay. During the subsequent replay, the transmitter has the opportunity to inject additional information about the Flit. These scenarios include viral containment and late poison and nullify scenarios.

To corrupt the CRC in these scenarios, the transmitter must invert all the bits of the CRC field during transmission. FEC generation must be done using the corrupted CRC. For latency-optimized 256B Flits, the transmitter must invert the CRC bits associated with either the even flit half or the odd flit half.

6.2.3.2.1

CXL.cachemem Viral Injection and Late Poison for 256B Flits

See [Chapter 4.0](#) for details on CXL.cachemem viral injection and late poison scenarios. [Section 4.3.6.2](#) describes the viral injection flow. [Section 4.3.6.3](#) describes the late poison injection flow.

6.2.3.2.2

Late Nullify or Poison for CXL.io

The PCIe specification defines a Flit Marker that is used to nullify or poison the last TLP in the flit. Because the Flit Header is forwarded at the beginning of a flit transmission, a transmitter may not know sufficiently early whether a Flit Marker is required to nullify or poison the last TLP. If the transmitter realizes after the Flit Header has been forwarded that a TLP must be poisoned or nullified, the transmitter must corrupt the CRC by inverting all the CRC bits. When the flit is subsequently replayed, the transmitter must use a Flit Header. For latency-optimized flits, if the last TLP that must be nullified or poisoned is in the even half, the even CRC must be inverted; if the last TLP that must be nullified or poisoned is in the odd half, the odd CRC must be inverted. FEC is calculated on the transmit side using the inverted CRC in these scenarios.

6.3

256B Flit Mode Retry Buffers

Following PCIe Base Specification, in 256B Flit mode, the Physical Layer implements the transmit retry buffer and the optional receive retry buffer. Whereas the retry buffers are managed independently in the CXL.io link layer and the CXL.cachemem link layer in 68B Flit mode, there is a single unified transmit retry buffer that handles all retryable CXL traffic in 256B Flit mode. Similarly, in 256B Flit mode, there is a single unified receive retry buffer that handles all retryable CXL traffic in 256B Flit mode. Retry requests are on a 256-byte flit granularity even when using the latency-optimized 256B flit composed of 128-byte flit halves. Please refer to [Section 6.2.3.1.2](#) for more details.

6.4

Link Training

6.4.1

PCIe Mode vs. Flex Bus.CXL Mode Selection

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), Sync Headers are bypassed as soon as the link has transitioned to a speed of 8 GT/s or higher. For 68B Flit mode, the Flex Bus logical PHY transmits NULL flits after it sends the SDS Ordered Set as soon as it transitions to 8 GT/s or higher link speeds if CXL mode was negotiated earlier in the training process. These NULL flits are used in place of PCIe Idle Symbols to facilitate certain LTSSM transitions to L0 as described in [Section 6.5](#). After the link has transitioned to its final speed, the link can start sending CXL traffic on behalf of the upper layers after the SDS Ordered Set is transmitted if that was what was negotiated earlier in the training process. For Upstream Ports, the physical layer notifies the upper layers that the link is up and available for transmission only after it has received a flit that was not generated by the physical layer of the partner Downstream Port (see [Table 6-2](#) for 68B Flit mode and [Table 6-6](#) for 256B Flit mode). To operate in CXL mode,

the link speed must be at least 8 GT/s. If the link is unable to transition to a speed of 8 GT/s or greater after committing to CXL mode during link training at 2.5 GT/s, the link may ultimately fail to link up even if the device is PCIe capable.

6.4.1.1

Hardware Autonomous Mode Negotiation

Dynamic hardware negotiation of Flex Bus mode occurs during link training in the LTSSM Configuration state before entering L0 at Gen 1 speeds using the alternate protocol negotiation mechanism, facilitated by exchanging modified TS1 and TS2 Ordered Sets as defined by PCIe Base Specification. The Downstream Port initiates the negotiation process by sending TS1 Ordered Sets advertising its Flex Bus capabilities. The Upstream Port responds with a proposal based on its own capabilities and those advertised by the host. The host communicates the final decision of which capabilities to enable by sending modified TS2 Ordered Sets before or during Configuration.Complete.

Please refer to PCIe Base Specification for details on how the various fields of the modified TS1/TS2 OS are set. [Table 6-9](#) shows how the modified TS1/TS2 OS is used for Flex Bus mode negotiation. The “Flex Bus Mode Negotiation Usage” column describes the deltas from the PCIe Base Specification definition that are applicable for Flex Bus mode negotiation. Additional explanation is provided in [Table 6-10](#) and [Table 6-11](#). The presence of Retimer1 and Retimer2 must be programmed into the Flex Bus Port DVSEC by software before the negotiation begins; if retimers are present, the relevant retimer bits in the modified TS1/TS2 OS are used.

Table 6-9. Modified TS1/TS2 Ordered Set for Flex Bus Mode Negotiation (Sheet 1 of 2)

Symbol Number	PCIe Description	Flex Bus Mode Negotiation Usage
0 through 4	See PCIe Base Specification Symbol	
5	Training Control: <ul style="list-style-type: none"> Bits[5:0]: See PCIe Base Specification Bits[7:6]: Modified TS1/TS2 Supported: See PCIe Base Specification for details 	<ul style="list-style-type: none"> Bits[7:6]: Value is 11b
6	<ul style="list-style-type: none"> For Modified TS1: TS1 Identifier, Encoded as D10.2 (4Ah) For Modified TS2: TS2 Identifier, Encoded as D5.2 (45h) 	<ul style="list-style-type: none"> TS1 Identifier during Phase 1 of Flex Bus mode negotiation TS2 Identifier during Phase 2 of Flex Bus mode negotiation
7	<ul style="list-style-type: none"> For Modified TS1: TS1 Identifier, Encoded as D10.2 (4Ah) For Modified TS2: TS2 Identifier, Encoded as D5.2 (45h) 	<ul style="list-style-type: none"> TS1 Identifier during Phase 1 of Flex Bus mode negotiation TS2 Identifier during Phase 2 of Flex Bus mode negotiation
8-9	<ul style="list-style-type: none"> Bits[2:0]: Usage: See PCIe Base Specification Bits[4:3]: Alternate Protocol Negotiation Status: <ul style="list-style-type: none"> Alternate Protocol Negotiation Status when Usage is 010b Otherwise, reserved (see PCIe Base Specification for details) Bits[15:5]: Alternate Protocol Details 	<ul style="list-style-type: none"> Bits[2:0]: Value is 010b (indicating alternate protocols) Bits[4:3]: Alternate Protocol Negotiation Status: See PCIe Base Specification Bits[7:5]: Alternate Protocol ID: <ul style="list-style-type: none"> 000b = Flex Bus Bit[8]: Common Clock Bits[15:9]: Reserved <p>See Table 6-10 for more information.</p>

Table 6-9. Modified TS1/TS2 Ordered Set for Flex Bus Mode Negotiation (Sheet 2 of 2)

Symbol Number	PCIe Description	Flex Bus Mode Negotiation Usage
10-11	Alternate Protocol ID/Vendor ID: <ul style="list-style-type: none"> Alternate Protocol ID/Vendor ID when Usage = 010b See PCIe Base Specification for descriptions that are applicable to other Usage values 	1E98h
12-14	See PCIe Base Specification Specific proprietary usage when Usage = 010b	<ul style="list-style-type: none"> Bits[7:0]: Flex Bus Mode Selection: <ul style="list-style-type: none"> Bit[0]: PCIe Capable/Enable Bit[1]: CXL.io Capable/Enable Bit[2]: CXL.mem Capable/Enable Bit[3]: CXL.cache Capable/Enable Bit[4]: CXL 68B Flit and VH Capable/Enable (formerly known as CXL 2.0 Capable/Enable) Bit[5]: Reserved Bits[23:8]: Flex Bus Additional Info: <ul style="list-style-type: none"> Bit[8]: Multi-Logical Device Capable/Enable Bit[9]: Reserved Bit[10]: Sync Header Bypass Capable/Enable Bit[11]: Latency-Optimized 256B Flit Capable/Enable Bit[12]: Retimer1 CXL Aware¹ Bit[13]: Reserved Bit[14]: Retimer2 CXL Aware² Bit[15]: CXL.io Throttle Required at 64 GT/s Bits[17:16]: CXL NOP Hint Info[1:0] Bit[18]: PBR Flit Capable/Enable Bit[19]: Reserved <p>See Table 6-11 for more information.</p>
15	See PCIe Base Specification	

1. Retimer1 is equivalent to Retimer X or Retimer Z in PCIe Base Specification.

2. Retimer2 is equivalent to Retimer Y in PCIe Base Specification.

Table 6-10. Additional Information on Symbols 8-9 of Modified TS1/TS2 Ordered Set

Bit Field in Symbols 8-9	Description
Bit[8]: Common Clock	The Downstream Port uses this bit to communicate to retimers that there is a common reference clock. Depending on implementation, retimers may use this information to determine which features to enable.

Table 6-11. Additional Information on Symbols 12-14 of Modified TS1/TS2 Ordered Sets (Sheet 1 of 2)

Bit Field in Symbols 12-14	Description
Bit[0]: PCIe Capable/Enable	The Downstream Port and Upstream Port advertise their capability in Phase 1. The Downstream Port communicates the results of the negotiation in Phase 2. ¹
Bit[1]: CXL.io Capable/Enable	The Downstream Port and Upstream Port advertise their capability in Phase 1 as set in the DVSEC Flex Bus Port Control register ² . The Downstream Port communicates the results of the negotiation in Phase 2. This bit must be set to 1 if the CXL 68B Flit and VH Capable/Enable bit is set.
Bit[2]: CXL.mem Capable/Enable	The Downstream Port and Upstream Port advertise their capability in Phase 1 as set in the DVSEC Flex Bus Port Control register ⁴ . The Downstream Port communicates the results of the negotiation in Phase 2.

Table 6-11. Additional Information on Symbols 12-14 of Modified TS1/TS2 Ordered Sets (Sheet 2 of 2)

Bit Field in Symbols 12-14	Description
Bit[3]: CXL.cache Capable/Enable	The Downstream Port and Upstream Port advertise their capability in Phase 1 as set in the DVSEC Flex Bus Port Control register ² . The Downstream Port communicates the results of the negotiation in Phase 2.
Bit[4]: CXL 68B Flit and VH Capable/Enable (formerly known as CXL 2.0 capable/enable)	The Downstream Port and Upstream Port advertise their capability in Phase 1 as set in the DVSEC Flex Bus Port Control register ² . The Downstream Port communicates the results of the negotiation in Phase 2. The Downstream Port must not enable this if PCIe Flit mode is enabled as described in PCIe Base Specification.
Bit[8]: Multi-Logical Device Capable/Enable	The Downstream Port and Upstream Port advertise their capability in Phase 1 as set in the DVSEC Flex Bus Port Control register ² . An Upstream Switch Port must always advertise 0 in this bit. The Downstream Port communicates the results of the negotiation in Phase 2.
Bit[10]: Sync Header Bypass Capable/Enable	<p>The Downstream Port, Upstream Port, and any retimers advertise their capability in Phase 1; the Downstream Port and Upstream Port advertise the value as set in the DVSEC Flex Bus Port Control register². The Downstream Port communicates the results of the negotiation in Phase 2.</p> <p>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.</p> <p>Note: This bit is applicable only at 8 GT/s, 16 GT/s, and 32 GT/s link speeds.</p>
Bit[11]: Latency-Optimized 256B Flit Capable/Enable	The Downstream Port and Upstream Port advertise their capability in Phase 1 as set in the DVSEC Flex Bus Port Control register ² . The Downstream Port communicates the results of the negotiation in Phase 2. See Section 6.2.3.1.2 for details of the latency-optimized 256B flit.
Bit[12]: Retimer1 CXL Aware	Retimer1 advertises whether it is CXL aware in Phase 1. If Retimer1 is CXL aware, it must use the "Sync Header Bypass Capable/Enable" bit. ³
Bit[14]: Retimer2 CXL Aware	Retimer2 advertises whether it is CXL aware in Phase 1. If Retimer2 is CXL aware, it must use the "Sync Header Bypass Capable/Enable" bit. ⁴
Bit[15]: CXL.io Throttle Required at 64 GT/s	During Phase 1, an Upstream Port uses this bit to communicate to the Downstream Port that the Upstream Port does not support receiving consecutive CXL.io flits (including CXL.io NOP flits) when 64 GT/s link speed is negotiated (see Section 6.4.1.3 for more details). Downstream Ports are required to support this feature. The Downstream Port logs the value communicated by the partner Upstream Port in its DVSEC Flex Bus Port Status register (see Section 8.2.1.3.3).
Bits[17:16]: CXL NOP Hint Info[1:0]	<p>During Phase 1, the Downstream Port and Upstream Port advertise whether they support injecting NOP flits in response to receiving NOP hints and also whether they require receiving a single NOP flit or two back-to-back NOP flits to switch over from a higher-latency FEC pipeline to a lower-latency pipeline. This field is encoded as follows:</p> <ul style="list-style-type: none"> • 00b = No support for injecting NOP flits in response to receiving NOP hints. • 01b = Supports injecting NOP flits. Requires receiving a single NOP flit to switch over from a higher-latency FEC pipeline to a lower-latency pipeline. • 10b = Reserved. • 11b = Supports injecting NOP flits. Requires receiving two back-to-back NOP flits to switch over from a higher-latency FEC pipeline to a lower-latency pipeline.
Bit[18]: PBR (Port Based Routing) Flit Capable/Enable	The Upstream Port and Downstream Port advertise that they support PBR flits in Phase 1, as set in the DVSEC Flex Bus Port Control register ² . The Downstream Port communicates the results of the negotiation in Phase 2. The Downstream Port must not enable PBR flits if PCIe Flit mode is not enabled as defined in PCIe Base Specification.

1. PCIe mode and CXL mode are mutually exclusive when the Downstream Port communicates the results of the negotiation in Phase 2.
2. See [Section 8.2.1.3.2](#) for the DVSEC Flex Bus Port Control register definition.
3. Retimer1 is equivalent to Retimer X or Retimer Z in PCIe Base Specification.
4. Retimer2 is equivalent to Retimer Y in PCIe Base Specification.

Hardware autonomous mode negotiation is a two-phase process that occurs while in Configuration.Lanenum.Wait, Configuration.Lanenum.Accept, and Configuration.Complete before entering L0 at Gen 1 speed:

- Phase 1: The Downstream Port sends a stream of modified TS1 Ordered Sets advertising its Flex Bus capabilities; the Upstream Port responds by sending a stream of modified TS1 Ordered Sets indicating which Flex Bus capabilities it wishes to enable. This exchange occurs during Configuration.Lanenum.Wait and/or Configuration.Lanenum.Accept. At the end of this phase, the Downstream Port has enough information to make a final selection of which capabilities to enable. The Downstream Port uses the Flex Bus capabilities information received in the first two consecutively received modified TS1 Ordered Sets in which the Alternate Protocol Negotiation status indicates that the Upstream Port supports the requested protocol.
- Phase 2: The Downstream Port sends a stream of modified TS2 Ordered Sets to the Upstream Port to indicate whether the link should operate in PCIe mode or in CXL mode; for CXL mode, it also specifies which CXL protocols, modes, and features to enable. The Downstream Port must set the Flex Bus enable bits identically in the 16 consecutive modified TS2 Ordered Sets sent before transitioning to Configuration.Idle. The Upstream Port acknowledges the enable request by sending modified TS2 Ordered Sets with the same Flex Bus enable bits set. This exchange occurs during Configuration.Complete. CXL alternate protocol negotiation successfully completes only after the Downstream Port has confirmed that the Flex Bus enable bits reflected in the eight consecutive modified TS2 Ordered Sets it receives that causes the transition to Configuration.Idle match what it transmitted; otherwise, the Downstream Port logs an error in the Flex Bus Port Status register and the physical layer LTSSM returns to Detect. If the Upstream Port receives an enable request in which the Flex Bus enable bits are not a subset of what it advertised in Phase 1, the behavior is undefined.

The Flex Bus negotiation process is complete before entering L0 at 2.5 GT/s. At this point the upper layers may be notified of the decision. If CXL mode is negotiated, the physical layer enables all the negotiated modes and features only after reaching L0 at 8 GT/s or higher speed.

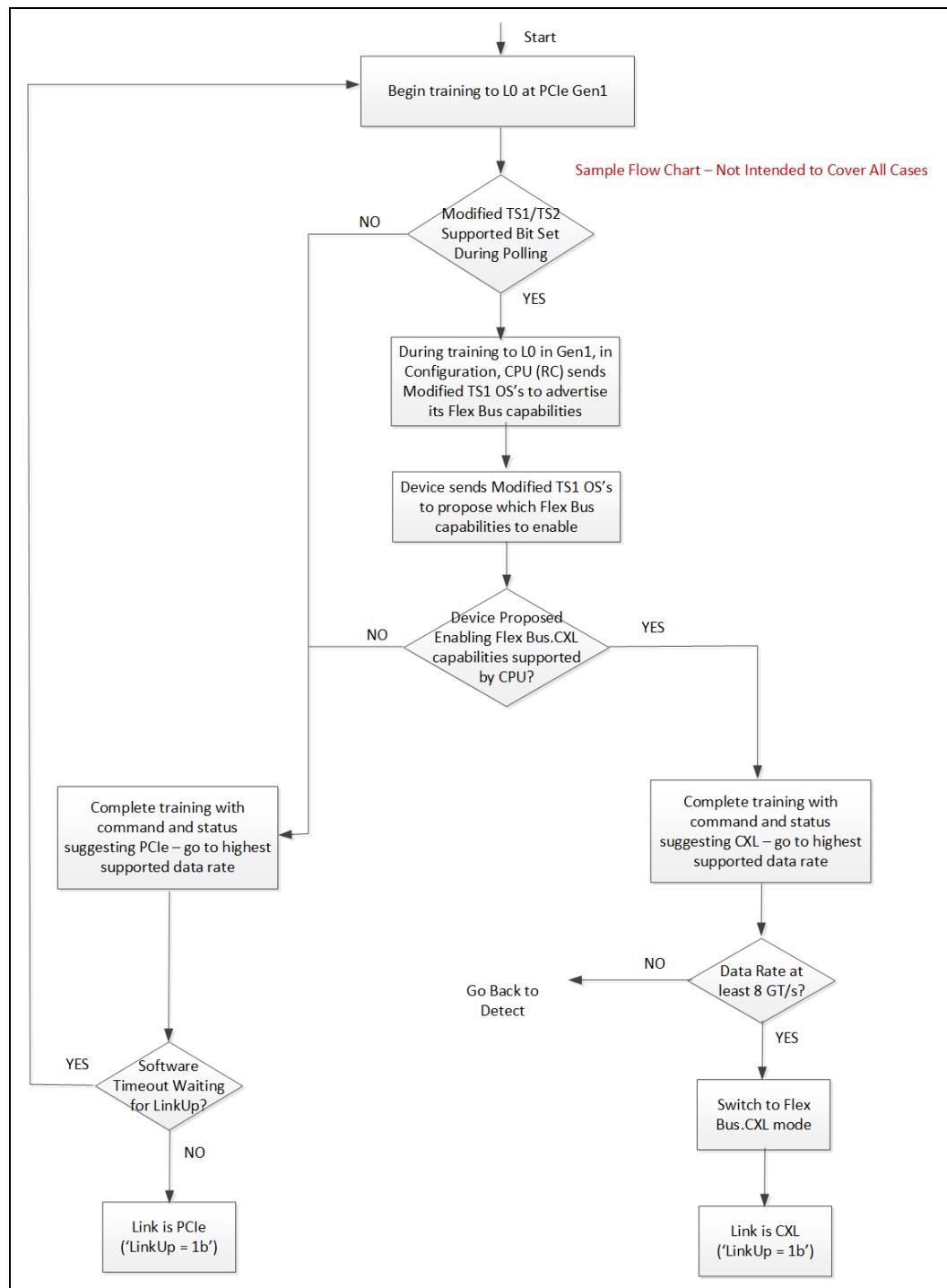
Note:

If CXL is negotiated but the link does not achieve a speed of at least 8 GT/s, the link will fail to link up and go back to LTSSM Detect.

A flow chart describing the mode negotiation process during link training is provided in [Figure 6-14](#). Note that while this flow chart represents the flow for several scenarios, it is not intended to cover all possible scenarios.

EVALUATION COPY

Figure 6-14. Flex Bus Mode Negotiation during Link Training (Sample Flow)



6.4.1.2 Virtual Hierarchy vs. Restricted CXL Device Negotiation

VH-capable devices support switching and hot add, features that are not supported in exclusive Restricted CXL Devices (eRCDs). This difference in supported features impacts the link training behavior. [Table 6-12](#) specifies the Flex Bus physical layer link training result for all possible combinations of upstream and downstream components. The table was constructed based upon the following assumptions:

- VH-capable Endpoints and switches are required to support hot add as a downstream component.
- VH-capable Downstream Ports are not required to support hot add; however, this capability is enforced at the software level. The Flex Bus physical layer will allow the link to train successfully for hot-add scenarios if both the upstream component and downstream component are VH capable.
- For exclusive Restricted CXL Hosts (eRCHs), BIOS prevents CXL hot-add scenarios by disabling CXL alternate protocol negotiation before handing control over to the OS. The Flex Bus physical layer does not have to handle these scenarios.
- For VH-capable Downstream Ports, BIOS sets the Disable_RCD_Training bit in the DVSEC Flex Bus Port Control register before handing control to the OS. For a host, the Flex Bus physical layer uses the Disable_RCD_Training bit to distinguish between initial power-on scenarios and hot-add scenarios to determine appropriate link training behavior with eRCDs.

Note:

In the context of this section, “VH-capable” was previously known as “CXL 2.0 and newer”, “eRCD” was previously known as a “CXL 1.1 capable device”, and “eRCH” was previously known as “CXL 1.1 capable host”.

Table 6-12. VH vs. RCD Link Training Resolution

Upstream Component	Downstream Component	Link Training Result
Host - VH capable	Switch	VH mode
Host - eRCH	Switch	Fail CXL alternate protocol negotiation
Host - VH capable	Endpoint - VH capable	VH mode
Host - VH capable	Endpoint - eRCD	RCD for initial power-on scenario; fail CXL alternate protocol negotiation for hot-add scenario
Host - eRCH	Endpoint - VH capable	RCD - assumes no hot add
Host - eRCH	Endpoint - eRCD	RCD - assumes no hot add
Switch	Endpoint - VH capable	VH mode
Switch	Endpoint - eRCD	RCD for initial power-on scenario; fail CXL alternate protocol negotiation for hot-add scenario

The motivation for forcing the Flex Bus physical layer to fail CXL training for certain combinations of upstream component and downstream component is to avoid unpredictable software behavior if the link were allowed to train. For the specific combination of an eRCH and a switch, the Upstream Switch Port is responsible for ensuring that CXL alternate protocol negotiation fails by returning a value of 01b in the Alternate Protocol Negotiation Status field of the modified TS1 to indicate that it does not support the requested protocol; this must occur during Phase 1 of the alternate protocol negotiation process after the Upstream Switch Port observes that the host is not VH capable.

6.4.1.2.1 Retimer Presence Detection

During CXL alternate protocol negotiation, the presence of a retimer impacts whether the Sync Header bypass optimization can be enabled as described in [Table 6-11](#). While eRCH Downstream Ports rely on BIOS to program the Retimer1_Present and Retimer2_Present bits in the DVSEC Flex Bus Port Control register prior to the start of link training, VH-capable Downstream Ports must ignore those register bits because BIOS is not involved with Hot-Plug scenarios.

VH-capable Downstream Ports must determine retimer presence for CXL alternate-protocol negotiation by sampling the Retimers Present bit and Two Retimers Present bit in the received TS2 Ordered Sets. VH-capable Downstream Ports adhere to the following steps for determining and using retimer presence information:

1. During Polling.Configuration.LTSSM state, the Downstream Port samples the Retimer Present bit and the Two Retimers Present bit for use during CXL alternate protocol negotiation. If the Retimer Present bit is set to 1 in the 8 consecutively received TS2 Ordered Sets that causes the transition to Configuration, then the Downstream Port must assume that a retimer is present for the purposes of CXL alternate protocol negotiation. If the Two Retimers Present bit is set to 1 in the 8 consecutively received TS2 Ordered Sets that causes the transition to Configuration, then the Downstream Port must assume that two retimers are present for the purposes of CXL alternate protocol negotiation.
2. During CXL alternate protocol negotiation, the Downstream Port uses the information sampled in step 1 along with the CXL Alternate Protocol Negotiation Status bits in the modified TS1 Ordered Set to determine whether to enable Sync Header bypass optimization. If a retimer was detected in step 1 on any lane associated with the configured link, then the Downstream Port assumes that a retimer is present. If two retimers were detected in step 1 on any lane associated with the configured link, then the Downstream Port assumes that two retimers are present.
3. During Configuration.Complete, per PCIe Base Specification, the Downstream Port captures "Retimer Present" and "Two Retimers Present" information from the received modified TS2 Ordered Sets into the Link Status 2 register. If the values sampled in this step are inconsistent with the values sampled during Polling.Configuration, then the Downstream Port logs an error in the DVSEC Flex Bus Port Status register, brings the LTSSM to Detect, and then retrains the link with Sync Header bypass optimization disabled.

6.4.1.3 256B Flit Mode

Certain protocol features, such as Back-Invalidate (BI), rely on 256B Flit mode. There is no explicit bit for 256B Flit mode negotiation. The following subsections describe 256B Flit mode negotiation, and [Section 6.4.1.4](#) provides further details of how negotiation results in either 256B Flit mode or 68B Flit mode.

6.4.1.3.1 256B Flit Mode Negotiation

As shown in [Table 6-4](#), 256B Flit mode is implied if PCIe Flit mode is enabled during PCIe link training as described in PCIe Base Specification. PCIe Flit mode is known prior to starting alternate protocol negotiation. No explicit bit is defined in the modified TS1/TS2 Ordered Sets for negotiating 256B Flit mode. 256B Flit mode is supported only at 8 GT/s and higher speeds.

For 256B Flit mode, the Upstream and Downstream Ports additionally negotiate whether to enable the latency-optimized 256B flit format or the standard CXL flit format. See [Section 6.2.3.1.2](#) and [Table 6-9](#).

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 it does not support receiving consecutive CXL.io flits (including CXL.io NOP flits) at a link speed of 64 GT/s. For the purpose of this feature, consecutive CXL.io flits are two flits with Flit Type encoding of 01b that are not separated by either an intervening flit 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.

6.4.1.3.3 NOP Insertion Hint Performance Optimization

Latency-optimized 256B Flit mode enables the Physical Layer to use a lower-latency path that bypasses FEC; however, whenever a CRC error is detected, the Physical Layer must switch over to a higher-latency path that includes the FEC logic. To switch back from the higher-latency FEC path to the lower-latency path, the Physical Layer relies on bubbles in the received data that occur due to SKP OSs or NOPs or flits that can be discarded while waiting for a specific sequence number during a replay or any other gaps. Note that NOPs or any other flits with valid DLLP payload cannot be discarded. Due to the 1E-6 bit error rate, the frequency of SKP OS insertion is insufficient to enable the Physical Layer to spend the majority of its time in the lower-latency path.

To address this, a device that detects a CRC error is permitted to send an NOP Insertion Hint to request the partner device to insert NOPs. The NOP insertion hint is defined as a NAK with a sequence number of 0. Upon receiving an NOP insertion hint, the Physical Layer may schedule a single NOP or two back-to-back NOPs to enable its link partner to switch back over to its low-latency path.

During link training, the Physical Layer communicates to its link partner whether the Physical Layer supports responding to NOP hints by inserting NOPs. The Physical Layer also communicates to its link partner whether the Physical Layer requires only a single NOP or whether it requires two back-to-back NOPs to switch over from its higher-latency FEC path to its lower-latency path.

6.4.1.4 Flit Mode and VH Negotiation

[Table 6-13](#) specifies the negotiation results that the Downstream Port must communicate during Phase 2 depending on the modes advertised by both sides during Phase 1. For example, if both sides advertise PCIe Flit mode and Latency-Optimized 256B Flit mode, then the negotiation results in Latency-Optimized 256B Flit mode.

Table 6-13. Flit Mode and VH Negotiation

Both Components Advertise Support during Phase 1 of Negotiation?				Phase 2 Negotiation Results
PCIe Flit Mode	PBR Flit Mode	Latency-Optimized 256B Flit Mode	68B Flit and VH Capable	
Yes	Yes	Yes/No	Yes	PBR Flit mode, Standard 256B Flit mode
Yes	No	Yes	Yes	Latency-Optimized 256B Flit mode
Yes	No	No	Yes	Standard 256B Flit mode
No	No	No	Yes	68B Flit mode, VH mode
No	No	No	No	See Table 6-12 for VH mode vs. RCD mode; 68B flits

6.4.1.5

Flex Bus.CXL Negotiation with Maximum Supported Link Speed of 8 GT/s or 16 GT/s

If an eRCH or eRCD physical layer implementation supports Flex Bus.CXL operation only at a maximum speed of 8 GT/s or 16 GT/s, it must still advertise support of 32 GT/s speed during link training at 2.5 GT/s to perform alternate protocol negotiation using modified TS1 and TS2 Ordered Sets. After the alternate protocol negotiation is complete, the Flex Bus logical PHY can then advertise the true maximum link speed that it supports as per PCIe Base Specification. It is strongly recommended that VH-capable devices support 32 GT/s link rate; however, a VH-capable device is permitted to use the algorithm described in this section to enable CXL alternate protocol negotiation if it does not support 32 GT/s link rate.

IMPLEMENTATION NOTE

A CXL device that advertises support of 32 GT/s in early training when it does not truly support 32 GT/s link rate may have compatibility issues for Polling.Compliance and Loopback entry from Config.LinkWidthStart. Please see PCIe Base Specification for more details. Devices that do this must ensure that they provide a mechanism to disable this behavior for the purposes of Polling.Compliance and Loopback testing scenarios.

6.4.1.6

Link Width Degradation and Speed Downgrade

If the link is operating in Flex Bus.CXL and degrades to a lower speed or lower link width that is still compatible with Flex Bus.CXL mode, the link should remain in Flex Bus.CXL mode after exiting recovery without having to go through the process of mode negotiation again. If the link drops to a speed or width that is incompatible with Flex Bus.CXL and cannot recover, the link must go down to the LTSSM Detect state; any subsequent action is implementation specific.

6.5

68B Flit Mode: Recovery.Idle and Config.Idle Transitions to L0

PCIe Base Specification requires transmission and receipt of a specific number of consecutive Idle data Symbols on configured lanes to transition from Recovery.Idle to L0 or Config.Idle to L0 (see PCIe Base Specification) while in non-Flit mode. When the Flex Bus logical PHY is in CXL mode operating in 68B Flit mode, it looks for NULL flits instead of Idle Symbols to initiate the transition to L0. When in Recovery.Idle or Config.Idle, the next state is L0 if four consecutive NULL flits are received and eight NULL flits are sent after receiving one NULL flit; all other PCIe Base Specification rules regarding these transitions apply.

6.6

L1 Abort Scenario

Because the CXL ARB/MUX virtualizes the link state that is seen by the link layers and only requests the physical layer to transition to L1 when the link layers are in agreement, there may be a race condition that results in an L1 abort scenario. In this scenario, the physical layer may receive an EIOS or detect Electrical Idle when the ARB/MUX is no longer requesting entry to L1. In this scenario, the physical layer is required to initiate recovery on the link to bring it back to L0.

6.7**68B Flit Mode: Exit from Recovery**

In 68B Flit mode, upon exit from recovery, the receiver assumes that any partial TLPs that were transmitted prior to recovery entry are terminated and must be retransmitted in full via a link-level retry. Partial TLPs include TLPs for which a subsequent EDB, Idle, or valid framing token were not received before entering recovery. The transmitter must satisfy any requirements to enable the receiver to make this assumption.

6.8**Retimers and Low Latency Mode**

The CXL specification supports the following features that can be enabled to optimize latency: bypass of sync header insertion and use of a drift buffer instead of an elastic buffer. Enablement of Sync Header bypass is negotiated during the Flex Bus mode negotiation process described in [Section 6.4.1.1](#). The Downstream Port, Upstream Port, and any retimers advertise their Sync Header bypass capability during Phase 1; and the Downstream Port communicates the final decision on whether to enable Sync Header bypass during Phase 2. Drift buffer mode is decided locally by each component. The rules for enabling each feature are summarized in [Table 6-14](#); these rules are expected to be enforced by hardware.

Table 6-14.**Rules of Enable Low-latency Mode Features**

Feature	Conditions for Enabling	Notes
Sync Header Bypass	<ul style="list-style-type: none"> All components support Common reference clock No retimer is present or retimer cannot add or delete SKPs (e.g., in low-latency bypass mode) Not in loopback mode 	
Drift Buffer L(instead of elastic buffer)	<ul style="list-style-type: none"> Common reference clock 	Each component can independently enable this (i.e., does not have to be coordinated). The physical layer logs in the Flex Bus Port DVSEC when this is enabled.

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 PCIe Base Specification.

Table 6-15.**Sync Header Bypass Applicability and Ordered Set Insertion Rate**

Data Rate	PCIe Flit Mode	Sync Header Bypass Applicable	Ordered Set Insertion Interval while in Data Stream
8 GT/s, 16 GT/s, 32 GT/s	No	Yes (common clock only)	With Sync Header bypassed, after every 340 data blocks ¹ ; With Sync Header enabled, per PCIe Base Specification
8 GT/s, 16 GT/s, 32 GT/s	Yes	Yes (common clock only)	Regardless of whether Sync Header is bypassed, per PCIe Base Specification (where flit interval refers to a 256B flit)
64 GT/s	Yes (required)	No	Per PCIe Base Specification (where flit interval refers to 256B flit)

1. See [Section 6.8.1](#) for details.

6.8.1

68B Flit Mode: SKP Ordered Set Frequency and L1/Recovery Entry

This section is applicable only for 68B Flit mode.

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 OS or a standard SKP OS after every 340 data blocks, unless it is exiting the data stream.

Note:

The control SKP OSs are alternated with standard SKP OSs at 16 GT/s or higher speeds; at 8 GT/s, only standard SKP OSs are scheduled.

- When exiting the data stream, the physical layer must replace the scheduled control SKP OS (or SKP OS) with either an EIOS (for L1 entry) or EIEOS (for all other cases including recovery).

When Sync Header bypass optimization is enabled, retimers rely on the above mechanism to know when L1/recovery entry is occurring. When Sync Header bypass is not enabled, retimers must not rely on the above mechanism.

While the above algorithm dictates the control SKP OS and standard SKP OS frequency within the data stream, it should be noted that CXL devices must still satisfy the PCIe Base Specification requirement of control SKP OS and standard SKP OS insertion, which is at least once every 370 to 375 blocks when not operating in Separate Reference Clocks with Independent Spread Spectrum Clocking (SRIS), as defined in PCIe Base Specification.

[Figure 6-15](#) 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 Header 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. Because 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 0s.

EVALUATION COPY

Figure 6-15. NULL Flit with EDS and Sync Header Bypass Optimization

The diagram illustrates the mapping of two data blocks onto four lanes (Lane 0, Lane 1, Lane 2, Lane 3) during a NULL flit with EDS and Sync Header Bypass Optimization.

340th Data Block: This block is mapped to all four lanes. Lane 0 starts with a ProtID=NULL w/EDS flit, followed by 15 00h flits. Lane 1 contains 15 OS (Ordered Set) flits. Lane 2 contains 15 00h flits. Lane 3 contains 15 00h flits.

Ordered Set Block: This block is also mapped to all four lanes. Lane 0 starts with a ProtID=NULL w/EDS flit, followed by 15 00h flits. Lane 1 contains 15 OS flits. Lane 2 contains 15 00h flits. Lane 3 contains 15 00h flits.

A vertical column on the right lists symbols from 0 to 15, corresponding to the flits in Lane 3.

	Lane 0	Lane 1	Lane 2	Lane 3	
ProtID=NULL w/EDS					Symbol 15
00h	00h	00h	00h	00h	Symbol 0
00h	00h	00h	00h	00h	Symbol 1
00h	00h	00h	00h	00h	Symbol 2
00h	00h	00h	00h	00h	Symbol 3
00h	00h	00h	00h	00h	Symbol 4
00h	00h	00h	00h	00h	Symbol 5
00h	00h	00h	00h	00h	Symbol 6
00h	00h	00h	00h	00h	Symbol 7
00h	00h	00h	00h	00h	Symbol 8
00h	00h	00h	00h	00h	Symbol 9
00h	00h	00h	00h	00h	Symbol 10
00h	00h	00h	00h	00h	Symbol 11
00h	00h	00h	00h	00h	Symbol 12
00h	00h	00h	00h	00h	Symbol 13
00h	00h	00h	00h	00h	Symbol 14
00h	00h	00h	00h	00h	Symbol 15
OS	OS	OS	OS	OS	Symbol 0
OS	OS	OS	OS	OS	Symbol 1
OS	OS	OS	OS	OS	Symbol 2
OS	OS	OS	OS	OS	Symbol 3
OS	OS	OS	OS	OS	Symbol 4
OS	OS	OS	OS	OS	Symbol 5
OS	OS	OS	OS	OS	Symbol 6
OS	OS	OS	OS	OS	Symbol 7
OS	OS	OS	OS	OS	Symbol 8
OS	OS	OS	OS	OS	Symbol 9
OS	OS	OS	OS	OS	Symbol 10
OS	OS	OS	OS	OS	Symbol 11
OS	OS	OS	OS	OS	Symbol 12
OS	OS	OS	OS	OS	Symbol 13
OS	OS	OS	OS	OS	Symbol 14
OS	OS	OS	OS	OS	Symbol 15

Figure 6-16 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 128b/130b encoding is used. In this example, the NULL flit contains only a 16-bit payload of 0s.

Figure 6-16. NULL Flit with EDS and 128b/130b Encoding

The diagram illustrates the mapping of an 'Ordered Set Block' to a data stream. On the left, a box labeled 'Ordered Set Block' is connected by a blue bracket to a 16x4 grid. The grid has columns labeled 'Lane 0', 'Lane 1', 'Lane 2', and 'Lane 3'. The first row contains 'ProtID=NULL w/EDS' and '00h' in Lane 0, and '00h' in Lane 1. Subsequent rows show alternating '01b' and 'OS' values across the lanes. To the right of the grid, each row is labeled with a symbol from 'Symbol 15' down to 'Symbol 0'. Below the grid, two blue double-headed arrows are positioned under the last two rows, indicating they represent the final flits of the data stream.

		Lane 0	Lane 1	Lane 2	Lane 3	
ProtID=NULL w/EDS	00h	00h				Symbol 15
01b	01b	01b	01b			Sync Header
OS	OS	OS	OS			Symbol 0
OS	OS	OS	OS			Symbol 1
OS	OS	OS	OS			Symbol 2
OS	OS	OS	OS			Symbol 3
OS	OS	OS	OS			Symbol 4
OS	OS	OS	OS			Symbol 5
OS	OS	OS	OS			Symbol 6
OS	OS	OS	OS			Symbol 7
OS	OS	OS	OS			Symbol 8
OS	OS	OS	OS			Symbol 9
OS	OS	OS	OS			Symbol 10
OS	OS	OS	OS			Symbol 11
OS	OS	OS	OS			Symbol 12
OS	OS	OS	OS			Symbol 13
OS	OS	OS	OS			Symbol 14
OS	OS	OS	OS			Symbol 15

§ §

7.0 Switching

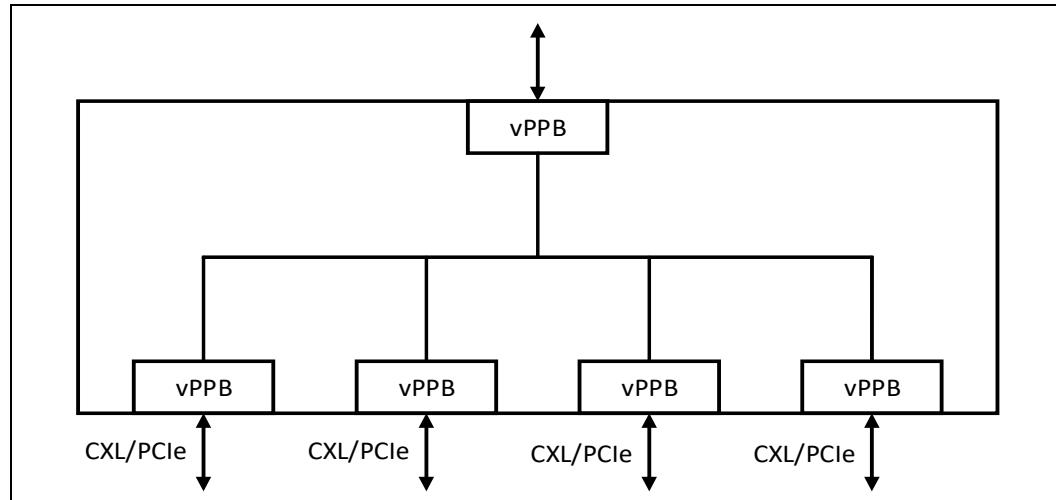
7.1 Overview

This section provides an architecture overview of different CXL switch configurations.

7.1.1 Single VCS

A single VCS consists of a single CXL Upstream Port and one or more Downstream Ports as illustrated in [Figure 7-1](#).

Figure 7-1. Example of a Single VCS



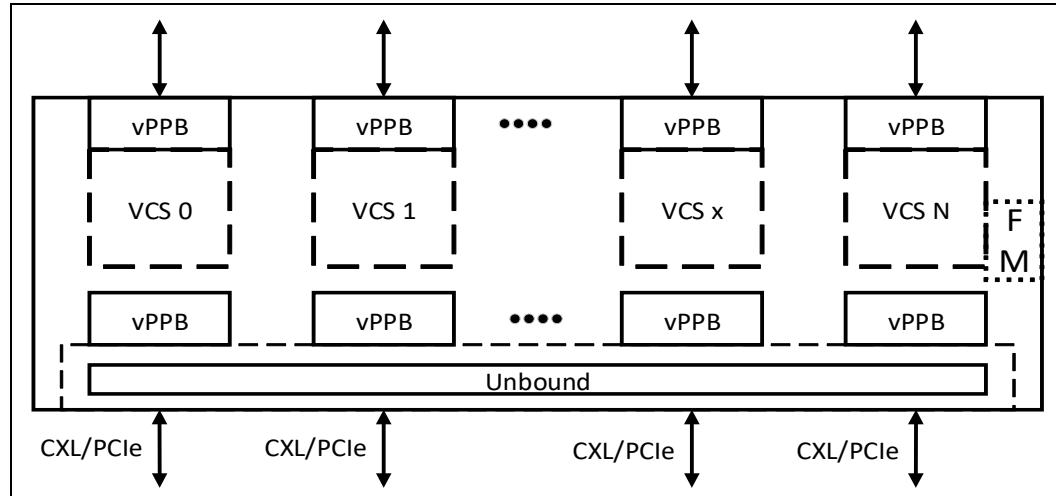
A Single VCS is governed by the following rules:

- Must have a single USP
- Must have one or more DSPs
- DSPs must support operating in CXL mode or PCIe* mode
- All non-MLD (includes PCIe and SLD) ports support a single Virtual Hierarchy below the vPPB
- Downstream Switch Port must be capable of supporting RCD mode
- Must support the CXL Extensions DVSEC for Ports (see [Section 8.1.5](#))
- The DVSEC defines registers to support CXL.io decode to support RCD below the Switch and registers for CXL Memory Decode. The address decode for CXL.io is in addition to the address decode mechanism supported by vPPB.
- Fabric Manager (FM) is optional for a Single VCS

7.1.2 Multiple VCS

A Multiple VCS consists of multiple Upstream Ports and one or more Downstream Ports per VCS as illustrated in Figure 7-2.

Figure 7-2. Example of a Multiple VCS with SLD Ports



A Multiple VCS is governed by the following rules:

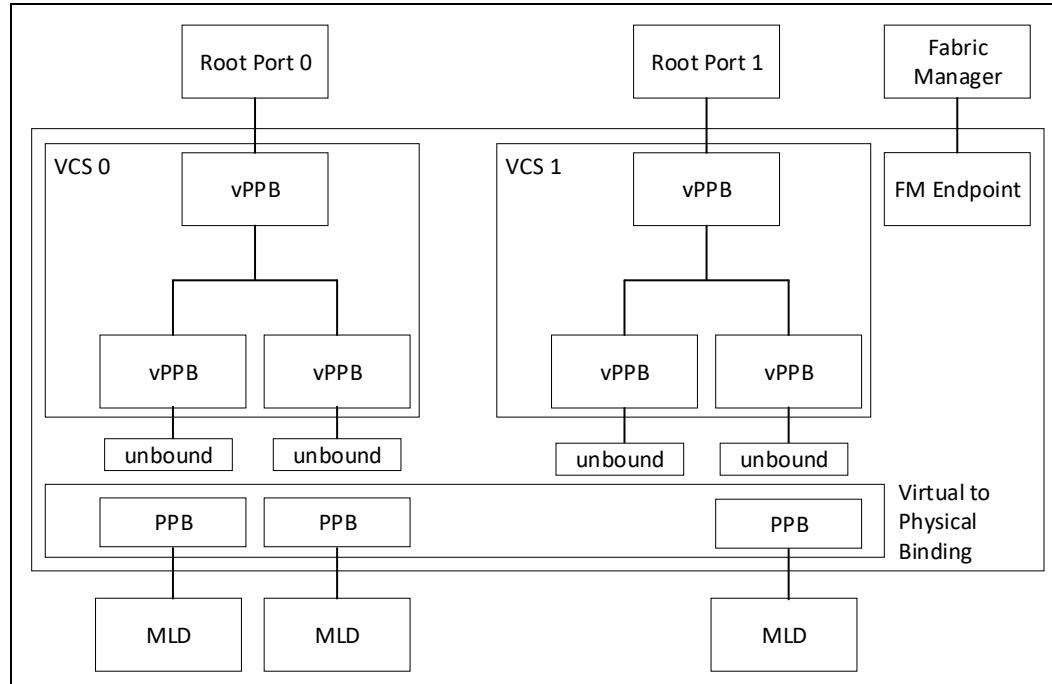
- Must have more than one USP.
- Must have one or more DS vPPBs per VCS.
- The initial binding of upstream (US) vPPB to physical port and the structure of the VCS (including number of vPPBs, the default vPPB capability structures, and any initial bindings of downstream (DS) vPPBs to physical ports) is defined using switch vendor specific methods.
- Each DSP must be bound to a PPB or vPPB.
- FM is optional for Multiple VCS. An FM is required for a Multiple VCS that requires bind/unbind, or that supports MLD ports. Each DSP can be reassigned to a different VCS through the managed Hot-Plug flow orchestrated by the FM.
- When configured, each USP and its associated DS vPPBs form a Single VCS Switch and operate as per the Single VCS rules.
- DSPs must support operating in CXL mode or PCIe mode.
- All non-MLD, non-Fabric, and non-GFD HBR ports support a single Virtual Hierarchy below the Downstream Switch Port.
- DSPs must be capable of supporting RCD mode.

7.1.3

Multiple VCS with MLD Ports

A Multiple VCS with MLD Ports consists of multiple Upstream Ports and a combination of one or more Downstream MLD Ports, as illustrated in [Figure 7-3](#).

Figure 7-3. Example of a Multiple Root Switch Port with Pooled Memory Devices



A Multiple VCS with MLD Ports is governed by the following rules:

- More than one USP.
- One or more Downstream vPPBs per VCS.
- Each SLD DSP can be bound to a Single VCS.
- An MLD-capable DSP can be connected to up to 16 USPs.
- Each of the SLD DSPs can be reassigned to a different VCS through the managed Hot-Plug flow orchestrated by the FM.
- Each of the LD instances in an MLD component can be reassigned to a different VCS through the managed Hot-Plug flow orchestrated by the FM.
- When configured, each USP and its associated vPPBs form a Single VCS, and operate as per the Single VCS rules.
- DSPs must support operating in CXL mode or PCIe mode.
- All non-MLD ports support a single Virtual Hierarchy below the DSP.
- DSPs must be capable of supporting RCD mode.

7.1.4

vPPB Ordering

vPPBs within a VCS are ordered in the following sequence: the USP vPPB, then the DSP vPPBs in increasing Device Number, Function Number order. This means Function 0 of all vPPBs in order of Device Number, then all vPPBs enumerated at Function 1 in order of Device Number, etc.

For a switch with 65 DSP vPPBs whose USP vPPB was assigned a Bus Number of 3, that would result in the following vPPB ordering:

vPPB #	PCIe ID
0	USP 3:0.0
1	DSP 4:0.0
2	DSP 4:1.0
3	DSP 4:2.0
	...
32	DSP 4:31.0
33	DSP 4:0.1
34	DSP 4:1.1
	...
64	DSP 4:31.1
65	DSP 4:0.2

This ordering also applies in cases where multi-function vPPBs exist but not all 32 Device Numbers are assigned. For example, a switch with 8 DSP vPPBs whose USP vPPB was assigned a Bus Number of 3 could present its DSP vPPBs in such a way that the host enumeration would result in the following vPPB ordering:

vPPB #	PCIe ID
0	USP 3:0.0
1	DSP 4:0.0
2	DSP 4:1.0
3	DSP 4:2.0
4	DSP 4:0.1
5	DSP 4:1.1
6	DSP 4:2.1
7	DSP 4:0.2
8	DSP 4:1.2

7.2

Switch Configuration and Composition

This section describes the CXL switch initialization options and related configuration and composition procedures.

7.2.1

CXL Switch Initialization Options

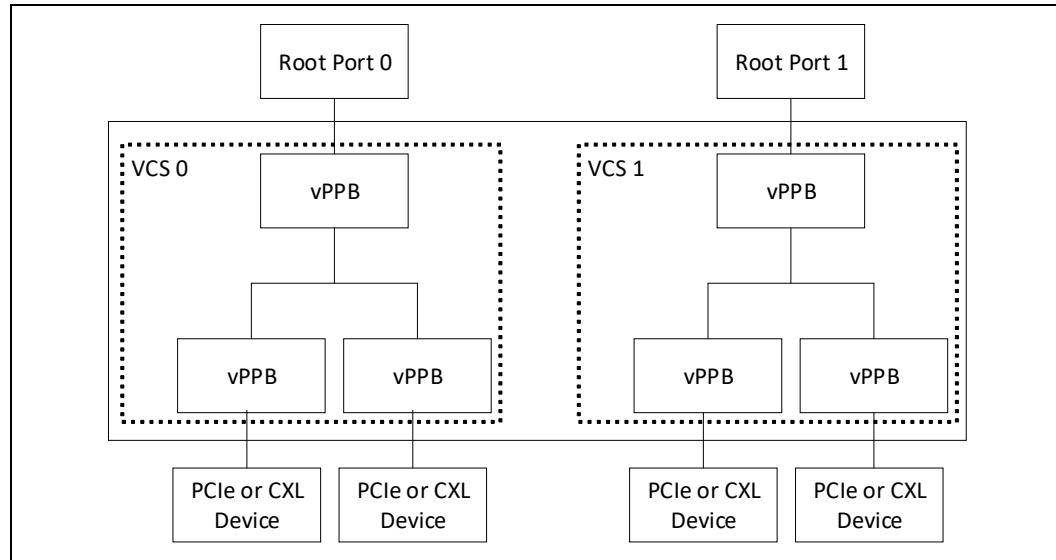
The CXL switch can be initialized using three different methods:

- Static
- FM boots before the host(s)
- FM and host boot simultaneously

7.2.1.1 Static Initialization

Figure 7-4 shows a statically initialized CXL switch with 2 VCSs. In this example, the downstream vPPBs are statically bound to ports and are available to the host at boot. Managed hot-add of Devices is supported using standard PCIe mechanisms.

Figure 7-4. Static CXL Switch with Two VCSs



Static Switch Characteristics:

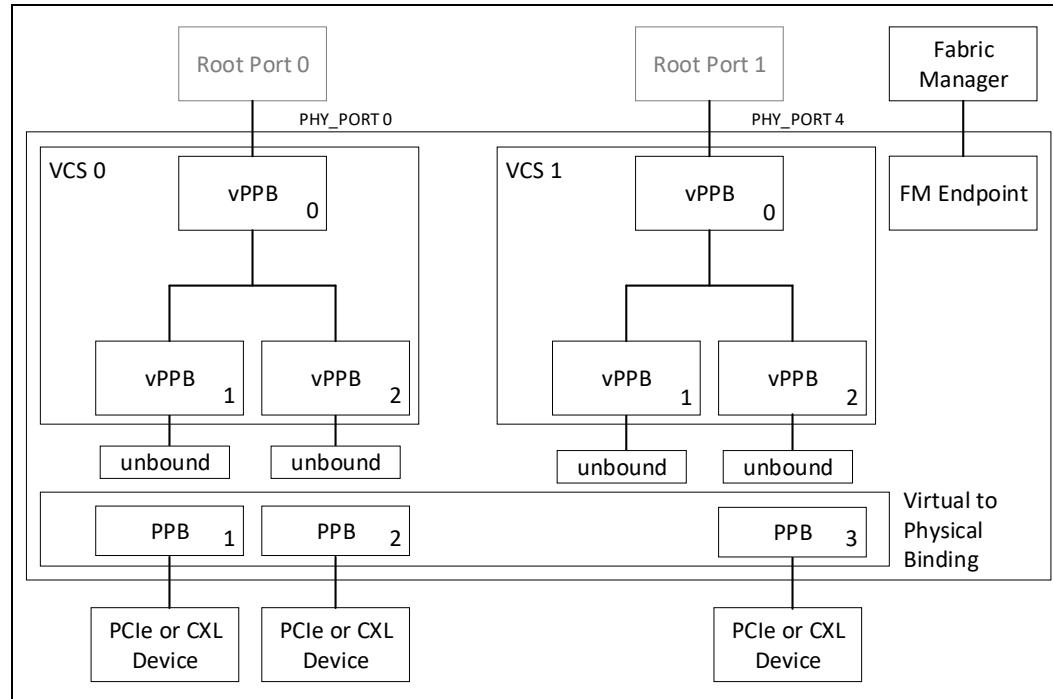
- No support for MLD Ports
- No support for rebinding of ports to a different VCS
- No FM is required
- At switch boot, all VCSs and Downstream Port bindings are statically configured using switch vendor defined mechanisms (e.g., configuration file in SPI Flash)
- Supports RCD mode, CXL VH mode, or PCIe mode
- VCSs, including vPPBs, behave identically to a PCIe switch, along with the addition of supporting CXL protocols
- Each VCS is ready for enumeration when the host boots
- Hot-add and managed hot-remove are supported
- No explicit support for Async removal of CXL devices; Async removal requires that root ports implement CXL Isolation (see [Section 12.3](#))

A switch that provides internal Endpoint functions is beyond the scope of this specification.

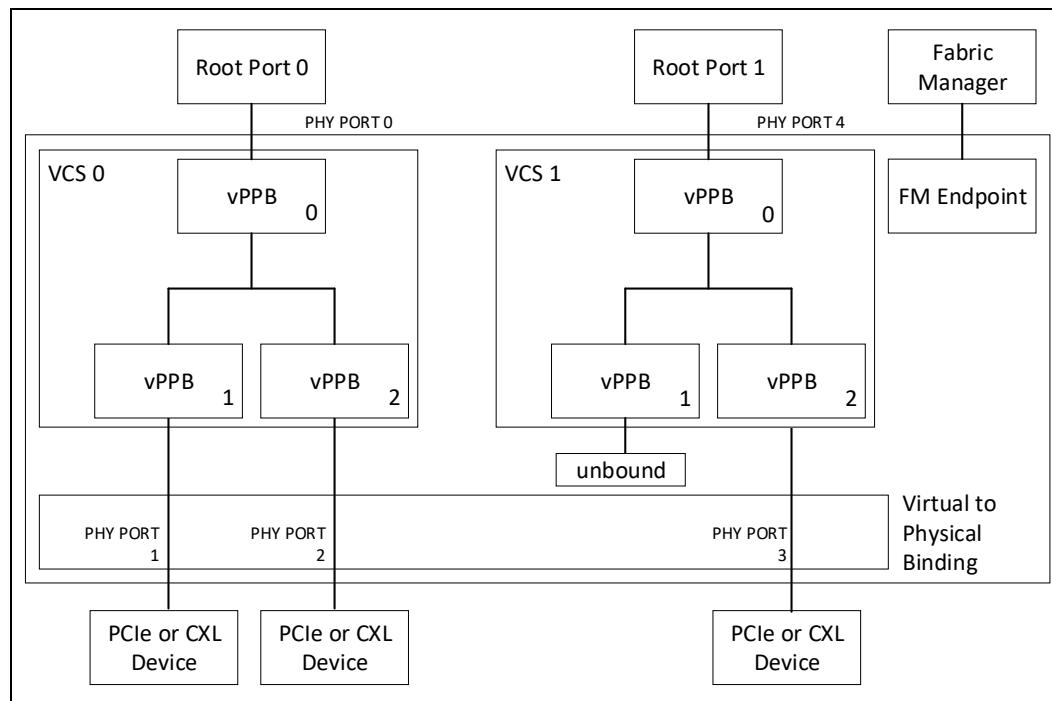
7.2.1.2 Fabric Manager Boots First

In cases where the FM boots first (prior to host(s)), the switch is permitted to be initialized as described in the example shown in [Figure 7-5](#).

Figure 7-5. [Example of CXL Switch Initialization when FM Boots First](#)



1. FM boots while hosts are held in reset.
2. All attached DSPs link up and are bound to FM-owned PPBs.
3. DSPs link up and the switch notifies the FM using a managed hot-add notification.

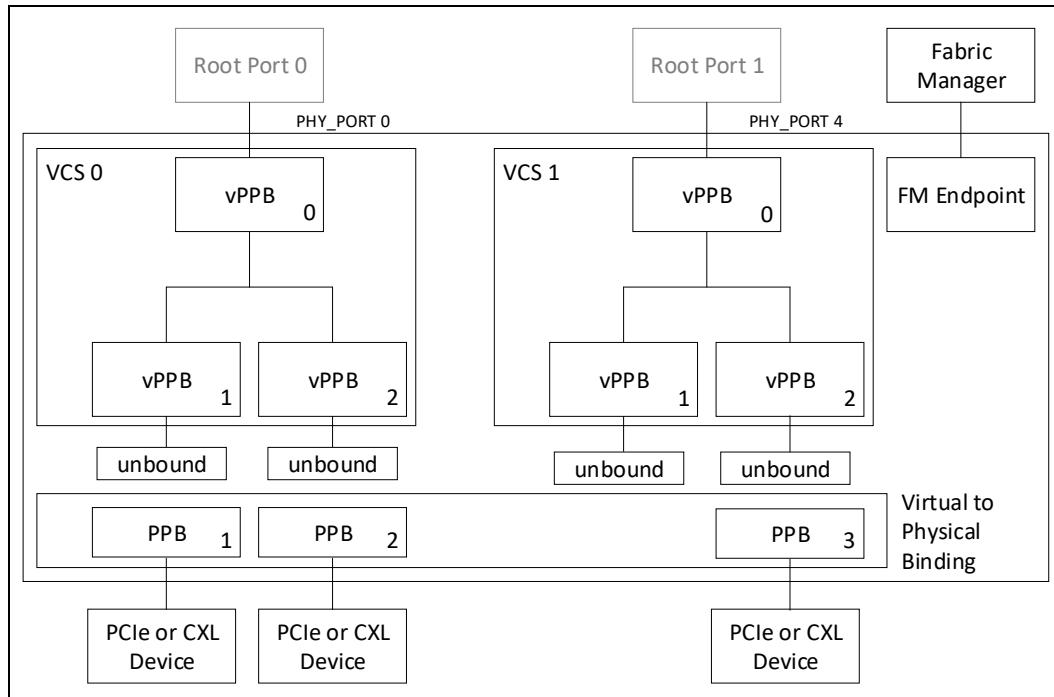
Figure 7-6. Example of CXL Switch after Initialization Completes

As shown in the example above in [Figure 7-6](#), the following steps are taken to configure the switch after initialization completes:

1. FM sends bind command BIND (VCS0, vPPB1, PHY_PORT_ID1) to the switch. The switch then configures virtual to physical binding.
2. Switch remaps vPPB virtual port numbers to physical port numbers.
3. Switch remaps vPPB connector definition (PERST#, PRSNT#) to physical connector.
4. Switch disables the link using PPB Link Disable.
5. At this point, all Physical downstream PPB functionality (e.g., Capabilities, etc.) maps directly to the vPPB including Link Disable, which releases the port for linkup.
6. The FM-owned PPB no longer exists for this port.
7. When the hosts boot, the switch is ready for enumeration.

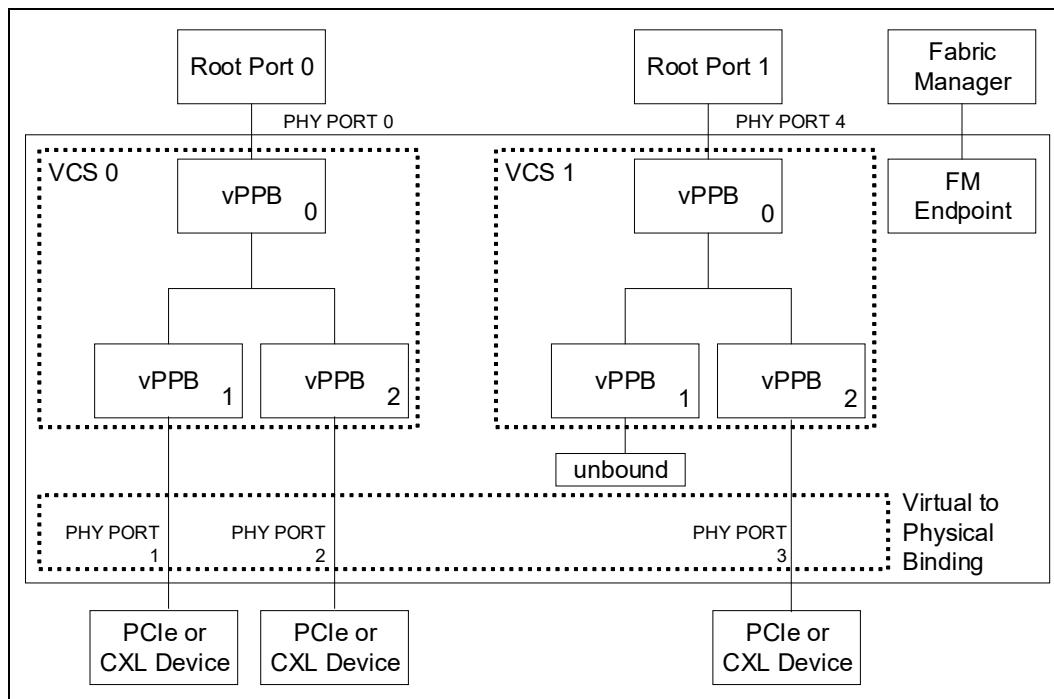
7.2.1.3 Fabric Manager and Host Boot Simultaneously

Figure 7-7. Example of Switch with Fabric Manager and Host Boot Simultaneously



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

1. VCSs are statically defined.
2. 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.
5. FM performs port binding to vPPBs.
6. Switch performs virtual to physical binding.
7. Each bound port results in a hot-add indication to the host.

Figure 7-8. Example of Simultaneous Boot after Binding

7.2.2 Sideband Signal Operation

The availability of slot sideband control signals is decided by the form-factor specifications. Any form factor can be supported, but if the form factor supports the signals listed in [Table 7-1](#), the signals must be driven by the switch or connected to the switch for correct operation.

All other sideband signals have no constraints and are supported exactly as in PCIe.

Table 7-1.**CXL Switch Sideband Signal Requirements**

Signal Name	Signal Description	Requirement
USP PERST#	PCIe Reset provides a fundamental reset to the VCS	This signal must be connected to the switch if implemented
USP ATTN#	Attention button indicates a request to the host for a managed hot-remove of the switch	If hot-remove of the switch is supported, this signal must be generated by the switch
DSP PERST#	PCIe Reset provides a power-on reset to the downstream link partner	This signal must be generated by the switch if implemented
DSP PRSNT#	Out-of-band Presence Detect indicates that a device has been connected to the slot	This signal must be connected to the switch if implemented
DSP ATTN#	Attention button indicates a request to the host for a managed hot-remove of the downstream slot	If managed hot-remove is supported, this signal must be connected to the switch

This list provides the minimum sideband signal set to support managed Hot-Plug. Other optional sidebands signals such as Attention LED, Power LED, Manual Retention Latch, Electromechanical Lock, etc. may also be used for managed Hot-Plug. The behavior of these sideband signals is identical to PCIe.

7.2.3 Binding and Unbinding

This section describes the details of Binding and Unbinding of CXL devices to a vPPB.

7.2.3.1 Binding and Unbinding of a Single Logical Device Port

A Single Logical Device (SLD) port refers to a port that is bound to only one VCS. That port can be linked up with a PCIe device or a CXL Type 1, Type 2, or Type 3 SLD component. In general, the vPPB bound to the SLD port behaves the same as a PPB in a PCIe switch. An exception is that a vPPB can be unbound from any physical port. In this case the vPPB appears to the host as if it is in a Link Down state with no Presence Detect indication. If optional rebinding is desired, this switch must have an FM API support and FM connection. The Fabric Manager can bind any unused physical port to the unbound vPPB. After binding, all the vPPB port settings are applied to that physical port.

Figure 7-9 shows a switch with bound DSPs.

Figure 7-9. Example of Binding and Unbinding of an SLD Port

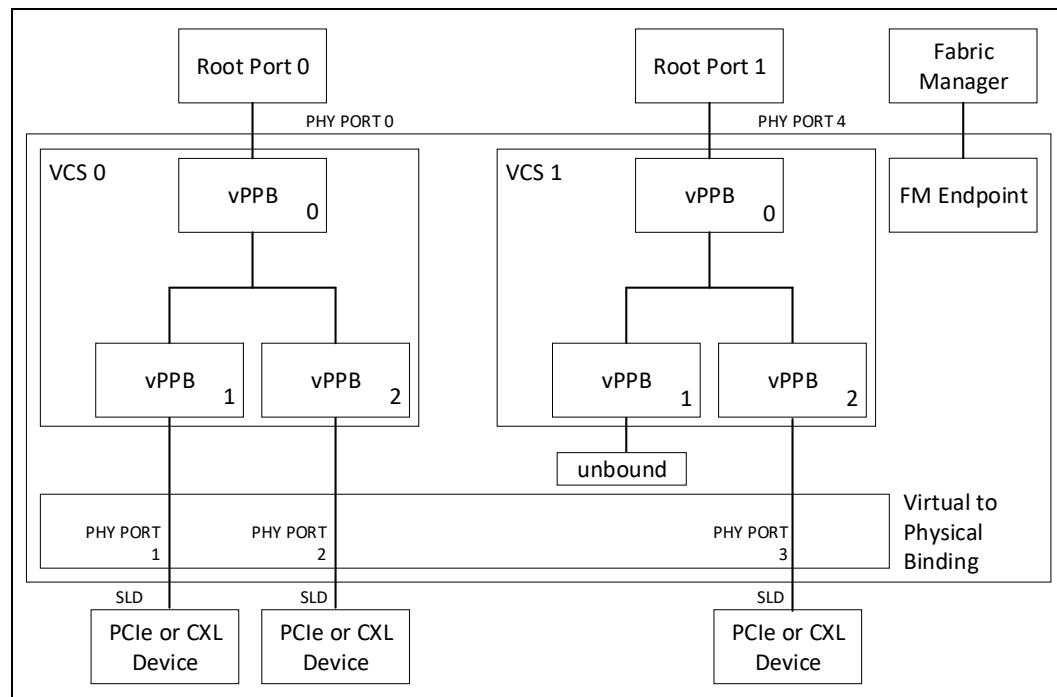


Figure 7-10 shows the state of the switch after the FM has executed an unbind command to vPPB2 in VCS0. Unbind of the vPPB causes the switch to assert Link Disable to the port. The port then becomes FM-owned and is controlled by the PPB settings for that physical port. Through the FM API, the FM has CXL.io access to each FM-owned SLD port or FM-owned LD within an MLD component. The FM can choose to prepare the logical device for rebinding by triggering FLR or CXL Reset. The switch prohibits any CXL.io access from the FM to a bound SLD port and any CXL.io access from the FM to an unbound LD within an MLD component. The FM API does not support FM generation of CXL.cache or CXL.mem transactions to any port.

EVALUATION COPY

Figure 7-10. Example of CXL Switch Configuration after an Unbind Command

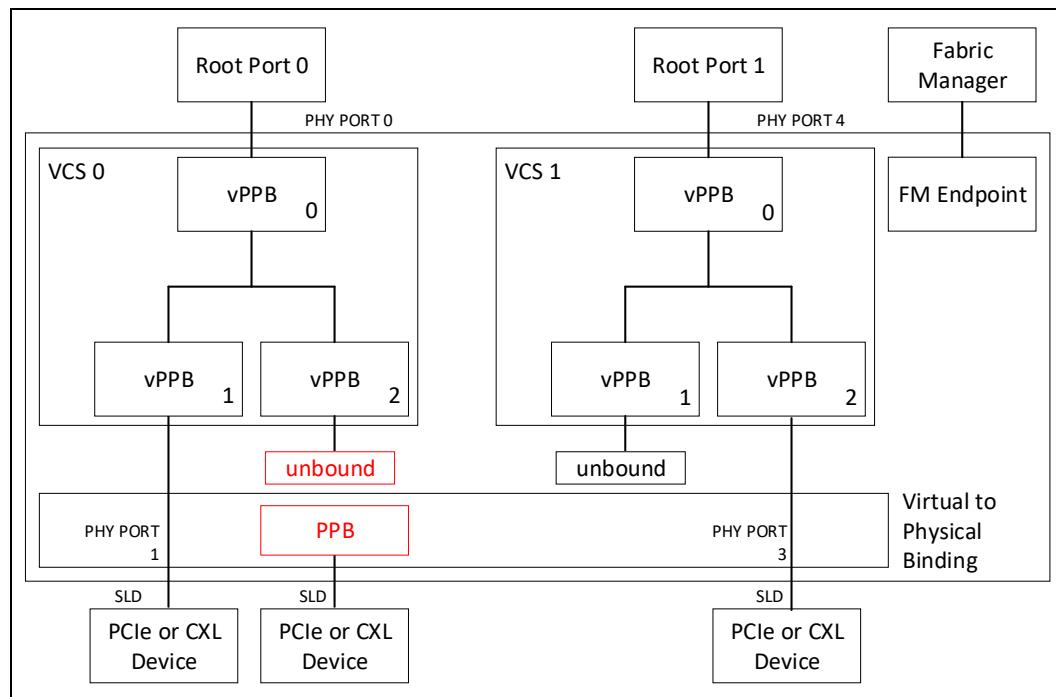
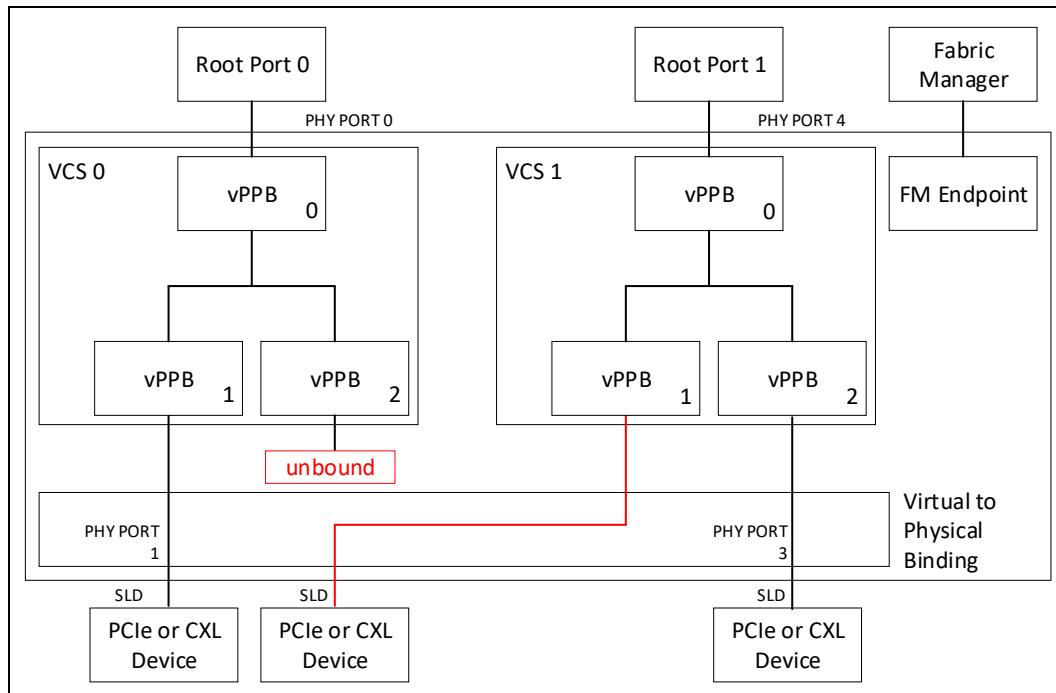


Figure 7-11 shows the state of the switch after the FM executes the bind command to connect VCS1.vPPB1 to the unbound physical port. The successful command execution results in the switch sending a hot-add indication to Host 1. Enumeration, configuration, and operation of the host and Type 3 device is identical to a hot-add of a device.

Figure 7-11. Example of CXL Switch Configuration after a Bind Command

7.2.3.2 Binding and Unbinding of a Pooled Device

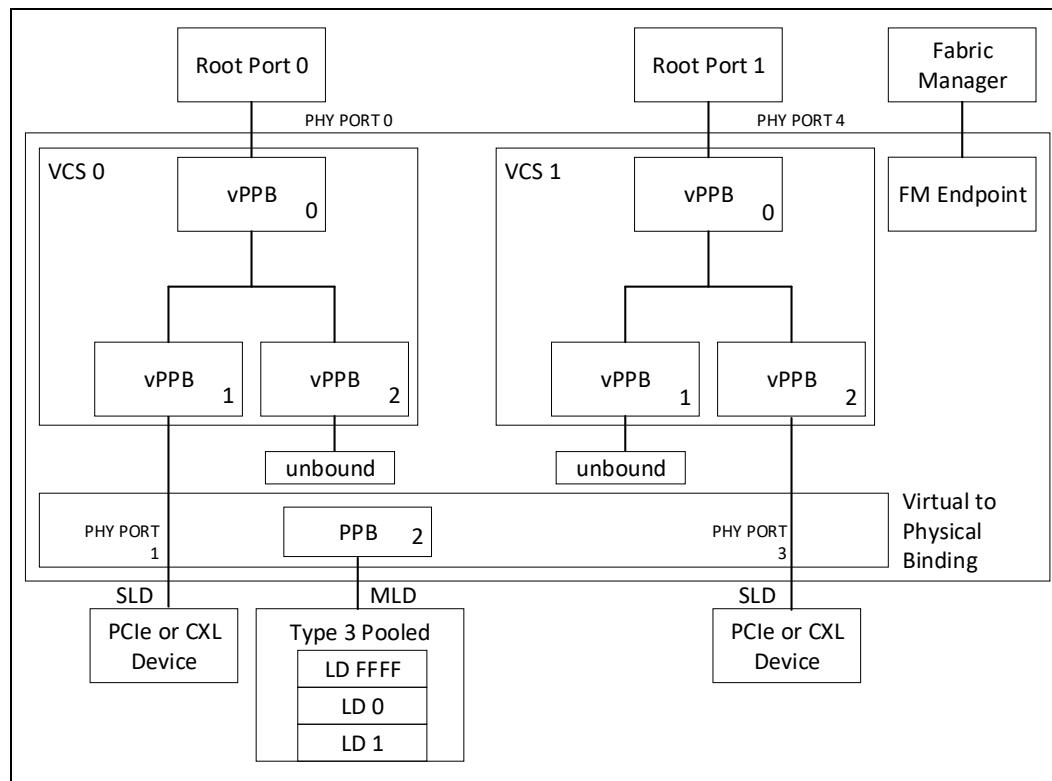
A pooled device contains multiple Logical Devices so that traffic over the physical port can be associated with multiple DS vPPBs. The switch behavior for binding and unbinding of an MLD component is similar to that of an SLD component, but with some notable differences:

1. The physical link cannot be impacted by binding and unbinding of a Logical Device within an MLD component. Thus, PERST#, Hot Reset, and Link Disable cannot be asserted, and there must be no impact to the traffic of other VCSs during the bind or unbind commands.
2. The physical PPB for an MLD port is always owned by the FM. The FM is responsible for port link control, AER, DPC, etc., and manages it using the FM API.
3. The FM may need to manage the pooled device to change memory allocations, enable the LD, etc.

Figure 7-12 shows a CXL switch after boot and before binding of any LDs within the pooled device. Note that the FM is not a PCIe Root Port and that the switch is responsible for enumerating the FMLD after any physical reset since the switch is responsible for proxying commands from FM to the device. The PPB of an MLD port is always owned by the FM since the FM is responsible for configuration and error handling of the physical port. After linkup the FM is notified that it is a Type 3 pooled device.

EVALUATION COPY

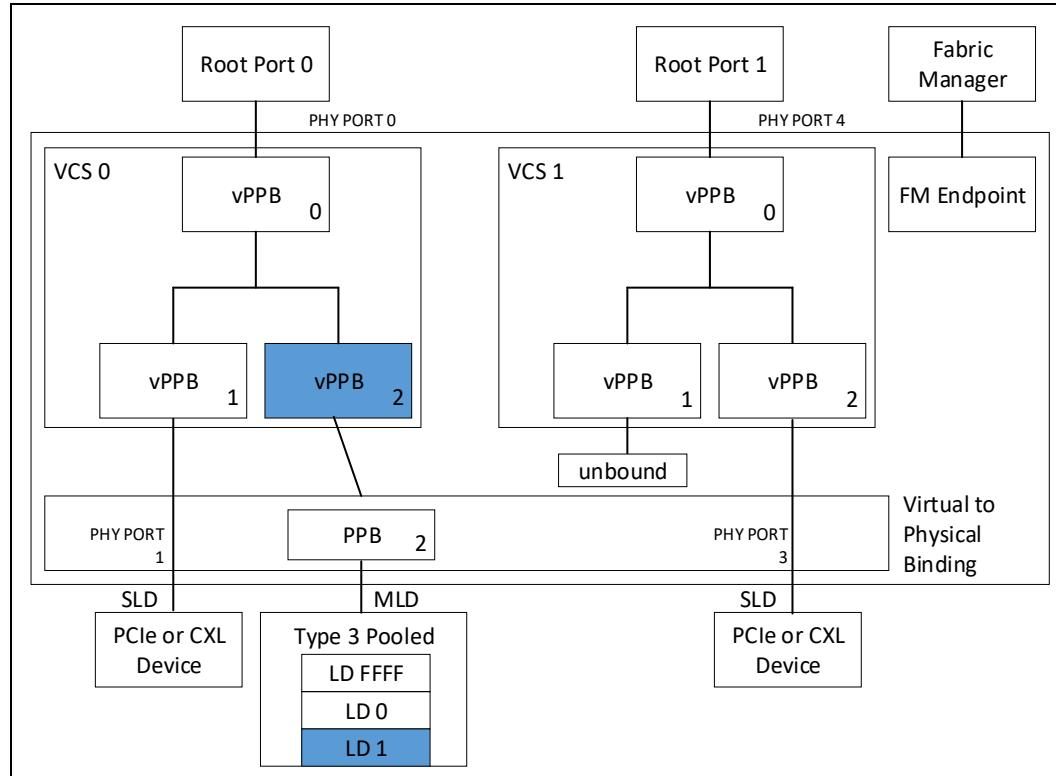
Figure 7-12. Example of a CXL Switch before Binding of LDs within Pooled Device



The FM configures the pooled device for Logical Device 1 (LD 1) and sets its memory allocation, etc. The FM performs a bind command for the unbound vPPB 2 in VCS 0 to LD 1 in the Type 3 pooled device. The switch performs the virtual-to-physical translations such that all CXL.io and CXL.mem transactions that target vPPB 2 in VCS 0 are routed to the MLD port with LD-ID set to 1. Additionally, all CXL.io and CXL.mem transactions from LD 1 in the pooled device are routed according to the host configuration of VCS 0. After binding, the vPPB notifies the VCS 0 host of a hot-add the same as if it were binding a vPPB to an SLD port.

Figure 7-13 shows the state of the switch after binding LD 1 to VCS 0.

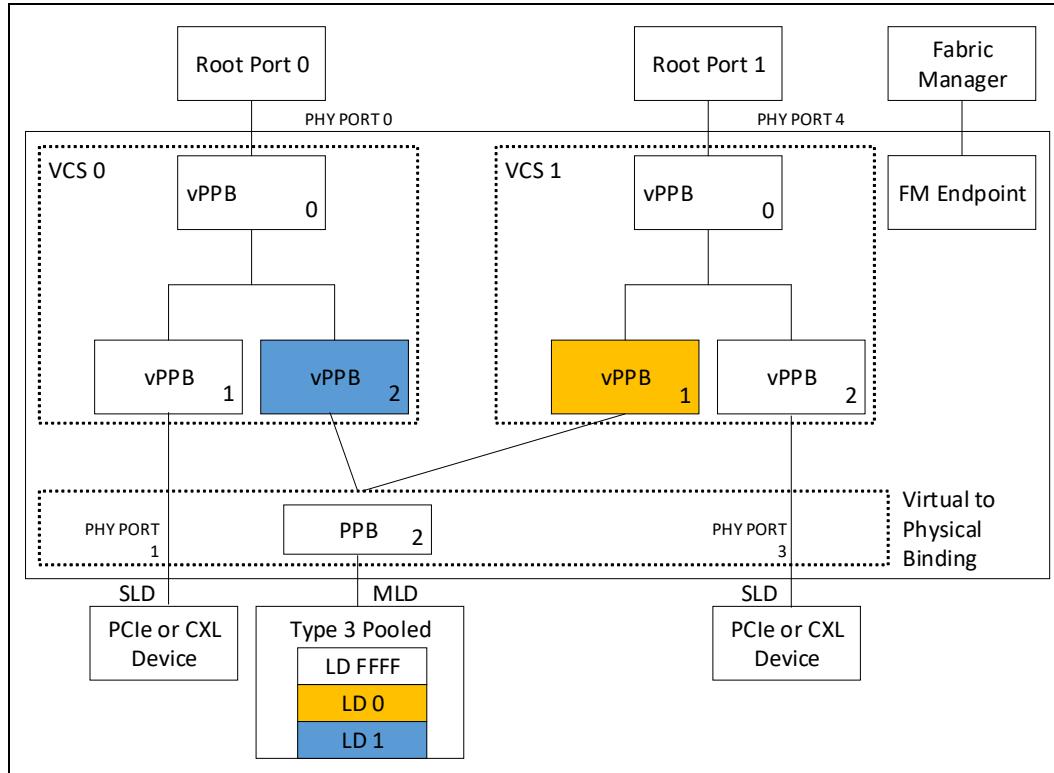
Figure 7-13. Example of a CXL Switch after Binding of LD-ID 1 within Pooled Device



The FM configures the pooled device for Logical Device 0 (LD 0) and sets its memory allocation, etc. The FM performs a bind command for the unbound vPPB 1 in VCS 1 to LD 0 in the Type 3 pooled device. The switch performs the virtual to physical translations such that all CXL.io and CXL.mem transactions targeting the vPPB in VCS 1 are routed to the MLD port with LD-ID set to 0. Additionally, all CXL.io and CXL.mem transactions from LD-ID = 0 in the pooled device are routed to the USP of VCS 1. After binding, the vPPB notifies the VCS 1 host of a hot-add the same as if it were binding a vPPB to an SLD port.

Figure 7-14 shows the state of the switch after binding LD 0 to VCS 1.

Figure 7-14. Example of a CXL Switch after Binding of LD-IDs 0 and 1 within Pooled Device



After binding LDs to vPPBs, the switch behavior is different from a bound SLD Port with respect to control, status, error notification, and error handling. [Section 7.3.4](#) describes the differences in behavior for all bits within each register.

7.2.4

PPB and vPPB Behavior for MLD Ports

An MLD port provides a virtualized interface such that multiple vPPBs can access LDs over a shared physical interface. As a result, the characteristics and behavior of a vPPB that is bound to an MLD port are different than the behavior of a vPPB that is bound to an SLD port. This section defines the differences between them. If not mentioned in this section, the features and behavior of a vPPB that is bound to an MLD port are the same as those for a vPPB that is bound to an SLD port.

This section uses the following terminology:

- Hardwire to 0 refers to status and optional control register bits that are initialized to 0. Writes to these bits have no effect.
- The term ‘Read/Write with no Effect’ refers to control register bits where writes are recorded but have no effect on operation. Reads to those bits reflect the previously written value or the initialization value if it has not been changed since initialization.

7.2.4.1 MLD Type 1 Configuration Space Header

Table 7-2. MLD Type 1 Configuration Space Header

Register	Register Fields	FM-owned PPB	All Other vPPBs
Bridge Control Register	Parity Error Response Enable	Supported	Hardwire to 0s
	SERR# Enable	Supported	Read/Write with no effect
	ISA Enable	Not supported	Not supported
	Secondary Bus Reset (see Section 7.5 for SBR details for MLD ports)	Supported	Read/Write with no effect. Optional FM Event.

7.2.4.2 MLD PCI*-compatible Configuration Registers

Table 7-3. MLD PCI-compatible Configuration Registers

Register/Capability Structure	Capability Register Fields	FM-owned PPB	All vPPBs Bound to the MLD Port
Command Register	I/O Space Enable	Hardwire to 0s	Hardwire to 0s
	Memory Space Enable	Supported	Supported per vPPB
	Bus Master Enable	Supported	Supported per vPPB
	Parity Error Response	Supported	Read/Write with no effect
	SERR# Enable	Supported	Supported per vPPB
	Interrupt Disable	Supported	Hardwire to 0s
Status Register	Interrupt Status	Hardwire to 0 (INTx is not supported)	Hardwire to 0s
	Master Data Parity Error	Supported	Hardwire to 0s
	Signaled System Error	Supported	Supported per vPPB
	Detected Parity Error	Supported	Hardwire to 0s

7.2.4.3 MLD PCIe Capability Structure

Table 7-4. MLD PCIe Capability Structure (Sheet 1 of 3)

Register/Capability Structure	Capability Register Fields	FM-owned PPB	All vPPBs Bound to the MLD Port
Device Capabilities Register	Max_Payload_Size Supported	Configured by the FM to the max value supported by switch hardware and min value configured in all vPPBs	Mirrors PPB
	Phantom Functions Supported	Hardwire to 0s	Hardwire to 0s
	Extended Tag Field Supported	Supported	Mirrors PPB
Device Control Register	Max_Payload_Size	Configured by the FM to Max_Payload Size Supported	Read/Write with no effect
Link Capabilities Register	Link Bandwidth Notification Capability	Hardwire to 0s	Hardwire to 0s

Table 7-4. MLD PCIe Capability Structure (Sheet 2 of 3)

Register/Capability Structure	Capability Register Fields	FM-owned PPB	All vPPBs Bound to the MLD Port
Link Capabilities	ASPM Support	No L0s support	No L0s support
	Clock Power Management	No PM L1 Substates support	No PM L1 Substates support
Link Control	ASPM Control	Supported	Switch only enables ASPM if all vPPBs that are bound to this MLD have enabled ASPM
	Link Disable	Supported	Switch handles it as an unbind by discarding all traffic to/from this LD-ID
	Retrain Link	Supported	Read/Write with no effect
	Common Clock Configuration	Supported	Read/Write with no effect
	Extended Synch	Supported	Read/Write with no effect
	Hardware Autonomous Width Disable	Supported	Read/Write with no effect
	Link Bandwidth Management Interrupt Enable	Supported	Read/Write with no effect
	Link Autonomous Bandwidth Interrupt Enable	Supported	Supported per vPPB. Each host can be notified of autonomous speed change
	DRS Signaling Control	Supported	Switch sends DRS after receiving DRS on the link and after binding of the vPPB to an LD
Link Status register	Current Link Speed	Supported	Mirrors PPB
	Negotiated Link Width	Supported	Mirrors PPB
	Link Training	Supported	Hardwire to 0s
	Slot Clock Configuration	Supported	Mirrors PPB
	Data Link Layer Active	Supported	Mirrors PPB
	Link Autonomous Bandwidth Status	Supported	Supported per vPPB
Slot Capabilities Register	Hot-Plug Surprise	Hardwire to 0s	Hardwired to 0s
	Physical Slot Number	Supported	Mirrors PPB
Slot Status Register	Attention Button Pressed	Supported	Mirrors PPB or is set by the switch on unbind
	Power Fault Detected	Supported	Mirrors PPB
	MRL Sensor Changed	Supported	Mirrors PPB
	Presence Detect Changed	Supported	Mirrors PPB or is set by the switch on unbind
	MRL Sensor State	Supported	Mirrors PPB
	Presence Detect State	Supported	Mirrors PPB or set by the switch on bind or unbind
	Electromechanical Interlock Status	Supported	Mirrors PPB
	Data Link Layer State Changed	Supported	Mirrors PPB or set by the switch on bind or unbind
Device Capabilities 2 Register	OBFF Supported	Hardwire to 0s	Hardwire to 0s

EVALUATION COPY

Table 7-4. MLD PCIe Capability Structure (Sheet 3 of 3)

Register/Capability Structure	Capability Register Fields	FM-owned PPB	All vPPBs Bound to the MLD Port
Device Control 2 Register	ARI Forwarding Enable	Supported	Supported per vPPB
	Atomic Op Egress Blocking	Supported	Mirrors PPB. Read/Write with no effect
	LTR Mechanism Enabled	Supported	Supported per vPPB
	Emergency Power Reduction Request	Supported	Read/Write with no effect. Optional FM notification.
	End-End TLP Prefix Blocking	Supported	Mirrors PPB. Read/Write with no effect
Link Control 2 Register	Target Link Speed	Supported	Read/Write with no effect. Optional FM notification.
	Enter Compliance	Supported	Read/Write with no effect
	Hardware Autonomous Speed Disable	Supported	Read/Write with no effect. Optional FM notification.
	Selectable De-emphasis	Supported	Read/Write with no effect
	Transmit Margin	Supported	Read/Write with no effect
	Enter Modified Compliance	Supported	Read/Write with no effect
	Compliance SOS	Supported	Read/Write with no effect
	Compliance Preset/De-emphasis	Supported	Read/Write with no effect
Link Status 2 Register	Current De-emphasis Level	Supported	Mirrors PPB
	Equalization 8.0 GT/s Complete	Supported	Mirrors PPB
	Equalization 8.0 GT/s Phase 1 Successful	Supported	Mirrors PPB
	Equalization 8.0 GT/s Phase 2 Successful	Supported	Mirrors PPB
	Equalization 8.0 GT/s Phase 3 Successful	Supported	Mirrors PPB
	Link Equalization Request 8.0 GT/s	Supported	Read/Write with no effect
	Retimer Presence Detected	Supported	Mirrors PPB
	Two Retimers Presence Detected	Supported	Mirrors PPB
	Crosslink Resolution	Hardwire to 0s	Hardwire to 0s
	Downstream Component Presence	Supported	Reflects the binding state of the vPPB
	DRS Message Received	Supported	Switch sends DRS after receiving DRS on the link and after binding of the vPPB to an LD

7.2.4.4 MLD PPB Secondary PCIe Capability Structure

All fields in the Secondary PCIe Capability Structure for a Virtual PPB shall behave identically to PCIe except the following:

Table 7-5.

MLD Secondary PCIe Capability Structure

Register/Capability Structure	Capability Register Fields	FM-owned PPB	All vPPBs Bound to the MLD Port
Link Control 3 Register	Perform Equalization	Supported	Read/Write with no effect
	Link Equalization Request Interrupt Enable	Supported	Read/Write with no effect
	Enable Lower SKP OS Generation Vector	Supported	Read/Write with no effect
Lane Error Status Register	All fields	Supported	Mirrors PPB
Lane Equalization Control Register	All fields	Supported	Read/Write with no effect
Data Link Feature Capabilities Register	All fields	Supported	Hardwire to 0s
Data Link Feature Status Register	All fields	Supported	Hardwire to 0s

7.2.4.5 MLD Physical Layer 16.0 GT/s Extended Capability

All fields in the Physical Layer 16.0 GT/s Extended Capability Structure for a Virtual PPB shall behave identically to PCIe except the following:

Table 7-6.

MLD Physical Layer 16.0 GT/s Extended Capability

Register/Capability Structure	Capability Register Fields	FM-owned PPB	All vPPBs Bound to the MLD Port
16.0 GT/s Status Register	All fields	Supported	Mirrors PPB
16.0 GT/s Local Data Parity Mismatch Status Register	Local Data Parity Mismatch Status Register	Supported	Mirrors PPB
16.0 GT/s First Retimer Data Parity Mismatch Status Register	First Retimer Data Parity Mismatch Status	Supported	Mirrors PPB
16.0 GT/s Second Retimer Data Parity Mismatch Status Register	Second Retimer Data Parity Mismatch Status	Supported	Mirrors PPB
16.0 GT/s Lane Equalization Control Register	Downstream Port 16.0 GT/s Transmitter Preset	Supported	Mirrors PPB

7.2.4.6 MLD Physical Layer 32.0 GT/s Extended Capability

Table 7-7. MLD Physical Layer 32.0 GT/s Extended Capability

Register/Capability Structure	Capability Register Fields	FM-owned PPB	All vPPBs Bound to the MLD Port
32.0 GT/s Capabilities Register	All fields	Supported	Mirrors PPB
32.0 GT/s Control Register	All fields	Supported	Read/Write with no effect
32.0 GT/s Status Register	Link Equalization Request 32.0 GT/s	Supported	Read/Write with no effect
	All fields except Link Equalization Request 32.0 GT/s	Supported	Mirrors PPB
Received Modified TS Data 1 Register	All fields	Supported	Mirrors PPB
Received Modified TS Data 2 Register	All fields	Supported	Mirrors PPB
Transmitted Modified TS Data 1 Register	All fields	Supported	Mirrors PPB
32.0 GT/s Lane Equalization Control Register	Downstream Port 32.0 GT/s Transmitter Preset	Supported	Mirrors PPB

7.2.4.7 MLD Lane Margining at the Receiver Extended Capability

Table 7-8. MLD Lane Margining at the Receiver Extended Capability

Register/Capability Structure	Capability Register Fields	FM-owned PPB	All vPPBs Bound to the MLD Port
Margining Port Status Register	All fields	Supported	Always indicates Margining Ready and Margining Software Ready
Margining Lane Control Register	All fields	Supported	Read/Write with no effect

7.2.5 MLD ACS Extended Capability

CXL.io Requests and Completions are routed to the USP.

Table 7-9. MLD ACS Extended Capability

Register/Capability Structure	Capability Register Fields	FM-owned PPB	All vPPBs Bound to the MLD Port
ACS Capability Register	All fields	Supported	Supported because a vPPB can be bound to any port type
ACS Control Register	ACS Source Validation Enable	Hardwire to 0	Read/Write with no effect
	ACS Translation Blocking Enable	Hardwire to 0	Read/Write with no effect
	ACS P2P Request Redirect Enable	Hardwire to 1	Read/Write with no effect
	ACS P2P Completion Redirect Enable	Hardwire to 1	Read/Write with no effect
	ACS Upstream Forwarding Enable	Hardwire to 0	Read/Write with no effect
	ACS P2P Egress Control Enable	Hardwire to 0	Read/Write with no effect
	ACS Direct Translated P2P Enable	Hardwire to 0	Read/Write with no effect
	ACS I/O Request Blocking Enable	Hardwire to 0	Read/Write with no effect
	ACS DSP Memory Target Access Control	Hardwire to 0s	Read/Write with no effect
	ACS Unclaimed Request Redirect Control	Hardwire to 0	Read/Write with no effect

7.2.6 MLD PCIe Extended Capabilities

All fields in the PCIe Extended Capability structures for a vPPB shall behave identically to PCIe.

7.2.7 MLD Advanced Error Reporting Extended Capability

AER in an MLD port is separated into Triggering, Notifications, and Reporting. Triggering and AER Header Logging is handled at switch ingress and egress using switch-vendor-specific means. Notification is also switch-vendor specific, but it results in the vPPB logic for all vPPBs that are bound to the MLD port being informed of the AER errors that have been triggered. The vPPB logic is responsible for generating the AER status and error messages for each vPPB based on the AER Mask and Severity registers.

vPPBs that are bound to an MLD port support all the AER Mask and Severity configurability; however, some of the Notifications are suppressed to avoid confusion.

The PPB has its own AER Mask and Severity registers and the FM is notified of error conditions based on the Event Notification settings.

Errors that are not vPPB specific are provided to the host with a header log containing all 1s data. The hardware header log is provided only to the FM through the PPB.

Table 7-10 lists the AER Notifications and their routing indications for PPBs and vPPBs.

Table 7-10. MLD Advanced Error Reporting Extended Capability

Hardware Triggers	AER Error	FM-owned PPB	All vPPBs Bound to the MLD Port
AER Notifications	Data Link Protocol Error	Supported	Supported per vPPB
	Surprise Down Error	Supported	Supported per vPPB
	Poisoned TLP Received	Supported	Hardwire to 0
	Flow Control Protocol Error	Supported	Supported per vPPB
	Completer Abort	Supported	Supported to the vPPB that generated it
	Unexpected Completion	Supported	Supported to the vPPB that received it
	Receiver Overflow	Supported	Supported per vPPB
	Malformed TLP	Supported	Supported per vPPB
	ECRC Error	Supported	Hardwire to 0
	Unsupported Request	Supported	Supported per vPPB
	ACS Violation	Supported	Hardwire to 0
	Uncorrectable Internal Error	Supported	Supported per vPPB
	MC ¹ Blocked	Supported	Hardwire to 0
	Atomic Op Egress Block	Supported	Hardwire to 0
	E2E TLP Prefix Block	Supported	Hardwire to 0
	Poisoned TLP Egress block	Supported	Hardwire to 0
	Bad TLP (correctable)	Supported	Supported per vPPB
	Bad DLLP (correctable)	Supported	Supported per vPPB
	Replay Timer Timeout (correctable)	Supported	Supported per vPPB
	Replay Number Rollover (correctable)	Supported	Supported per vPPB
	Other Advisory Non-Fatal (correctable)	Supported	Supported per vPPB
	Corrected Internal Error Status (correctable)	Supported	Supported per vPPB
	Header Log Overflow Status (correctable)	Supported	Supported per vPPB

1. Refers to Multicast.

7.2.8 MLD DPC Extended Capability

Downstream Port Containment has special behavior for an MLD Port. The FM configures the AER Mask and Severity registers in the PPB and also configures the AER Mask and Severity registers in the FMLD in the pooled device. As in an SLD port an unmasked uncorrectable error detected in the PPB and an ERR_NONFATAL and/or ERR_FATAL received from the FMLD can trigger DPC.

Continuing the model of the ultimate receiver being the entity that detects and reports errors, the ERR_FATAL and ERR_NONFATAL messages sent by a Logical Device can trigger a virtual DPC in the PPB. When a virtual DPC is triggered, the switch discards all traffic received from and transmitted to that specific LD. The LD remains bound to the vPPB and the FM is also notified. Software triggered DPC also triggers virtual DPC on a vPPB.

When the DPC trigger is cleared the switch autonomously allows passing of traffic to/from the LD. Reporting of the DPC trigger to the host is identical to PCIe.

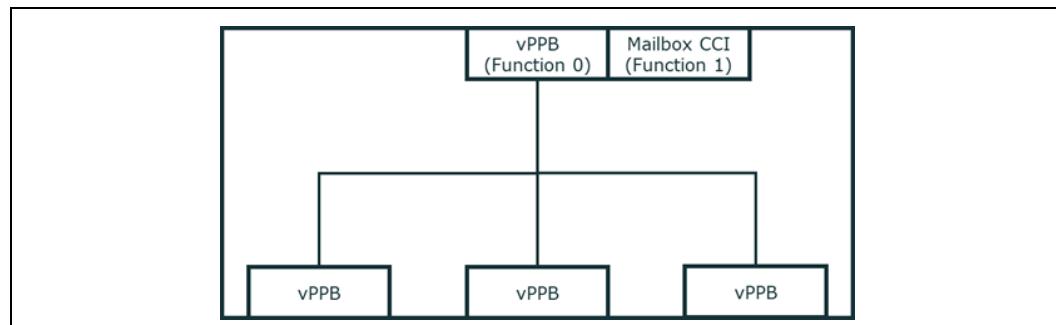
Table 7-11. MLD PPB DPC Extended Capability

Register/ Capability Structure	Capability Register Fields	FM-owned PPB	All vPPBs Bound to the MLD Port
DPC Control Register	DPC Trigger Enable	Supported	Switch internally detected unmasked uncorrectable errors do not trigger virtual DPC
	DPC Trigger Reason	Supported	Unmasked uncorrectable error is not a valid value

7.2.9 Switch Mailbox CCI

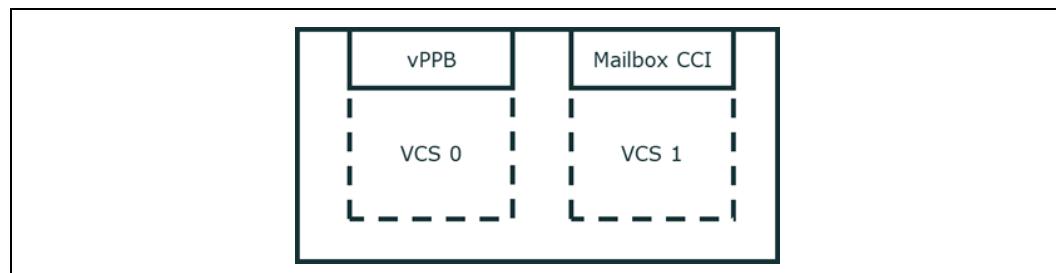
CXL Switch Mailbox CCIs optional. They are exposed as PCIe Endpoints with a Type 0 configuration space. In Single VCS and Multiple VCS, the Mailbox CCI is optional. If implemented, the Mailbox CCI shall be exposed to the Host in one of two possible configurations. In the first, it is exposed as an additional PCIe function in the Upstream Switch Port, as illustrated in [Figure 7-15](#).

Figure 7-15. Multi-function Upstream vPPB



Switch Mailbox CCIs may also be exposed in a VCS with no vPPBs. In this configuration, the Mailbox CCI device is the only PCIe function that is present in the Upstream Port, as illustrated in [Figure 7-16](#).

Figure 7-16. Single-function Mailbox CCI



7.3 CXL.io, CXL.cachemem Decode and Forwarding

7.3.1 CXL.io

Within a VCS, the CXL.io traffic must obey the same request, completion, address decode, and forwarding rules for a Switch as defined in PCIe Base Specification. There are additional decode rules that are defined to support an eRCD connected to a switch (see [Section 9.12.4](#)).

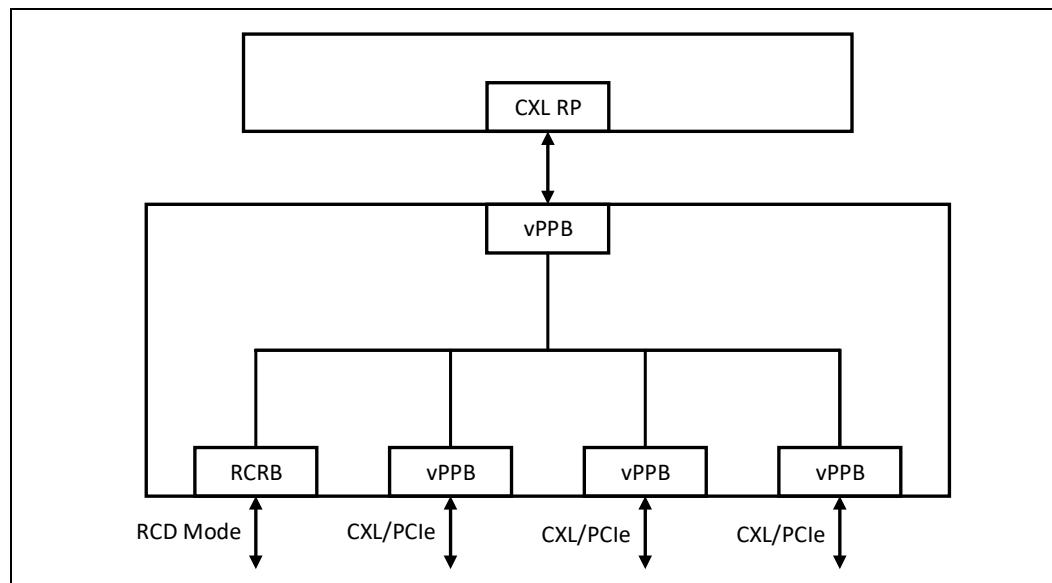
7.3.1.1 CXL.io Decode

When a TLP is decoded by a PPB, it determines the destination PPB to route the TLP based on the rules defined in PCIe Base Specification. Unless specified otherwise, all rules defined in PCIe Base Specification apply for routing of CXL.io TLPs. TLPs must be routed to PPBs within the same VCS. Routing of TLPs to and from an FM-owned PPB need to follow additional rules as defined in [Section 7.2.3](#). P2P inside a Switch complex is limited to PPBs within a VCS.

7.3.1.2 RCD Support

RCDs are not supported behind ports that are configured to operate as FM-owned PPBs. When connected behind a switch, RCDs must appear to software as RCiEP devices. The mechanism defined in this section enables this functionality.

Figure 7-17. CXL Switch with a Downstream Link Auto-negotiated to Operate in RCD Mode



The CXL Extensions DVSEC for Ports (see [Section 8.1.5](#)) defines the alternate MMIO and Bus Range Decode windows for forwarding of requests to eRCDs connected behind a Downstream Port.

7.3.2 CXL.cache

If the switch does not support CXL.cache protocol enhancements that enable multi-device scaling (as described in [Section 8.2.4.28](#)), only one of the CXL SLD ports in the VCS is allowed to be enabled to support Type 1 devices or Type 2 devices. Requests and Responses received on the USP are routed to the associated DSP and vice-versa. Therefore, additional decode registers are not required for CXL.cache for such switches.

If the switch supports CXL.cache protocol enhancements that enable multi-device scaling, more than one of the CXL SLD ports in the VCS can be configured to support Type 1 devices or Type 2 devices. [Section 9.15.2](#) and [Section 9.15.3](#) describe how such a CXL switch routes CXL.cache traffic.

CXL.cache is not supported over FM-owned PPBs.

7.3.3

CXL.mem

The HDM Decode DVSEC capability contains registers that define the Memory Address Decode Ranges for Memory. CXL.mem requests originate from the Host/RP and flow downstream to the Devices through the switch. CXL.mem responses originate from the Device and flow upstream to the RP.

7.3.3.1

CXL.mem Request Decode

All CXL.mem Requests received by the USP target one of the Downstream PPBs within the VCS. The address decode registers in the VCS determine the downstream VCS PPB to route the request. The VCS PPB may be a VCS-owned PPB or an FM-owned PPB. See [Section 7.3.4](#) for additional routing rules.

7.3.3.2

CXL.mem Response Decode

CXL.mem Responses received by the DSP target one and only one Upstream Port. For VCS-owned PPB the responses are routed to the Upstream Port of that VCS. Responses received on an FM-owned PPB go through additional decode rules to determine the VCS ID to route the requests to. See [Section 7.3.4](#) for additional routing rules.

7.3.4

FM-owned PPB CXL Handling

All PPBs are FM-owned. A PPB can be connected to a port that is disconnected, 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.

For all CXL.io transactions driven by the FM API, the switch acts as a virtual Root Complex for PPBs and Endpoints. The switch is responsible for enumerating the functions associated with that port and sending/receiving CXL.io traffic.

7.4

CXL Switch PM

7.4.1

CXL Switch ASPM L1

ASPM L1 for switch Ports is as defined in [Chapter 10.0](#).

7.4.2

CXL Switch PCI-PM and L2

A vPPB in a VCS operates the same as a PCIe vPPB for handling of PME messages.

7.4.3

CXL Switch Message Management

CXL VDMs are of the “Local - Terminate at Receiver” type. When a switch is present in the hierarchy, the switch implements the message aggregation function and therefore all Host-generated messages terminate at the switch. The switch aggregation function is responsible for regenerating these messages on the Downstream Port. All messages

and responses generated by the directly attached CXL components are aggregated and consolidated by the DSP and consolidated messages or responses are generated by the USP.

The PM message credit exchanges occur between the Host and Switch Aggregation port, and separately between the Switch Aggregation Port and device.

Table 7-12. CXL Switch Message Management

Message Type	Type	Switch Message Aggregation and Consolidation Responsibility
PM Reset Messages	Host Initiated	Host-generated requests terminate at Upstream Port, broadcast messages to all ports within VCS hierarchy
Sx Entry		
GPF Phase 1 Request		
GPF Phase 2 Request		
PM Reset Acknowledge	Device Responses	Device-generated responses terminate at Downstream Port within VCS hierarchy. Switch aggregates responses from all other connected ports within VCS hierarchy.
Sx Entry		
GPF Phase 1 Response		
GPF Phase 2 Response		

7.5 CXL Switch RAS

Table 7-13. CXL Switch RAS

Host 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# assert	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 port reset	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 received 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

Because the MLD DVSEC only exists in the FMLD, the switch must use the FM LD-ID in the CXL.io configuration write transaction when triggering LD reset.

7.6 Fabric Manager Application Programming Interface

This section describes the Fabric Manager Application Programming Interface.

7.6.1

CXL Fabric Management

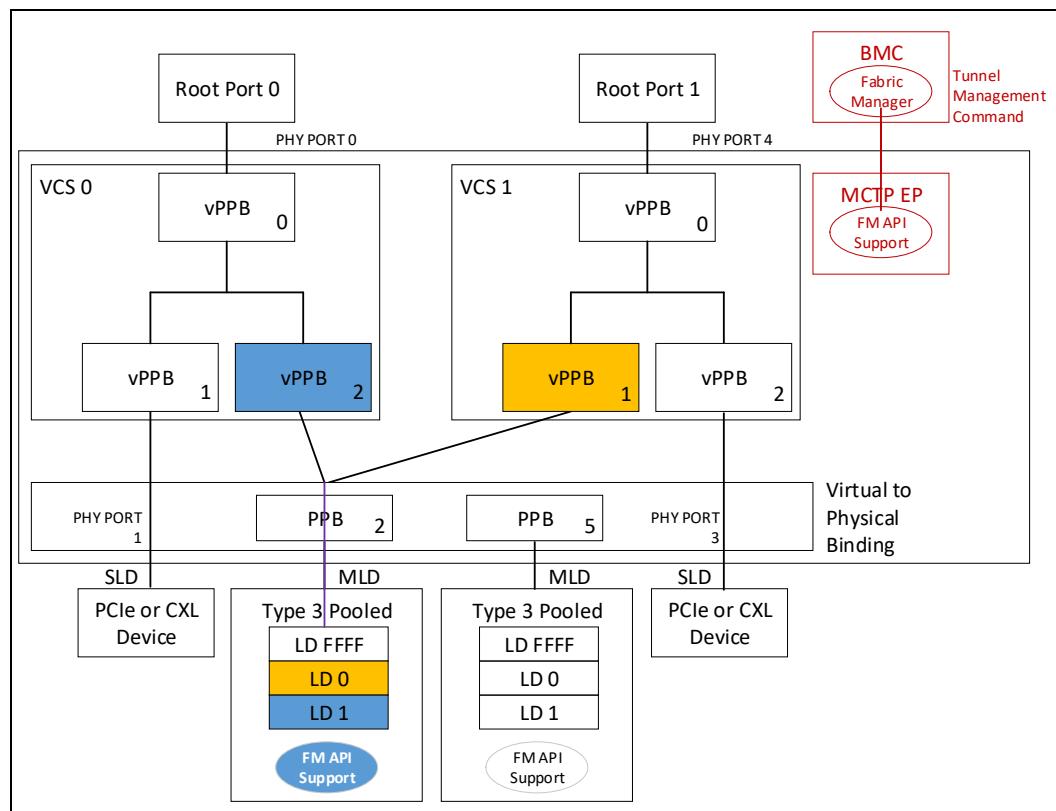
CXL devices can be configured statically or dynamically via a Fabric Manager (FM), an external logical process that queries and configures the system's operational state using the FM commands defined in this specification. The FM is defined as the logical process that decides when reconfiguration is necessary and initiates the commands to perform configurations. It can take any form, including, but not limited to, software running on a host machine, embedded software running on a BMC, embedded firmware running on another CXL device or CXL switch, or a state machine running within the CXL device itself.

7.6.2

Fabric Management Model

CXL devices are configured by FMs through the Fabric Manager Application Programming Interface (FM API) command sets, as defined in [Section 8.2.9.10](#), through a CCI. A CCI is exposed through a device's Mailbox registers (see [Section 8.2.8.4](#)) or through an MCTP-capable interface. See [Section 9.19](#) for details on the CCI processing of these commands.

Figure 7-18. Example of Fabric Management Model



FMs issue request messages and CXL devices issue response messages. CXL components may also issue the "Event Notification" request if notifications are supported by the component and the FM has requested notifications from the component using the Set MCTP Event Interrupt Policy command. See [Section 7.6.3](#) for transport protocol details.

The following list provides a number of examples of connectivity between an FM and a component's CCI, but should not be considered a complete list:

- An FM directly connected to a CXL device through any MCTP-capable interconnect can issue FM commands directly to the device. This includes delivery over MCTP-capable interfaces such as SMBus as well as VDM delivery over a standard PCIe tree topology where the responder is mapped to a CXL attached device.
- An FM directly connected to a CXL switch may use the switch to tunnel FM commands to MLD components directly attached to the switch. In this case, the FM issues the “Tunnel Management Command” command to the switch specifying the switch port to which the device is connected. Responses are returned to the FM by the switch. In addition to MCTP message delivery, the FM command set provides the FM with the ability to have the switch proxy config cycles and memory accesses to a Downstream Port on the FM’s behalf.
- An FM or part of the overall FM functionality may be embedded within a CXL component. The communication interface between such an embedded FM FW module and the component hardware is considered a vendor implementation detail and is not covered in this specification.

7.6.3 CCI Message Format and Transport Protocol

CCI commands are transmitted across MCTP-capable interconnects as MCTP messages using the format defined in [Figure 7-19](#) and listed in [Table 7-14](#).

Figure 7-19. CCI Message Format

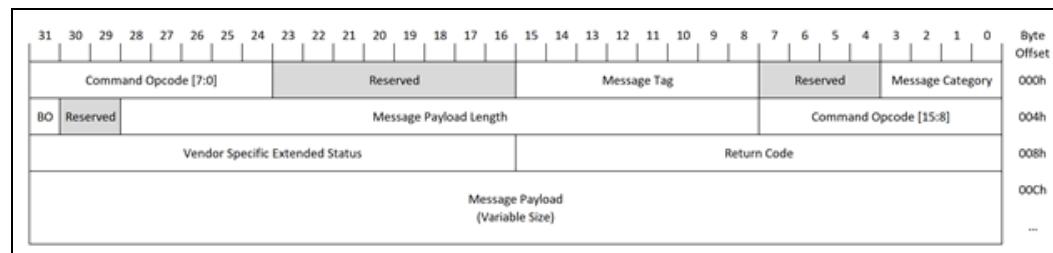


Table 7-14. CCI Message Format (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
0h	1	<ul style="list-style-type: none"> Bits[3:0]: Message Category: Type of CCI message: <ul style="list-style-type: none"> — 0h = Request — 1h = Response — All other encodings are reserved Bits[7:4]: Reserved
1h	1	Message Tag : Tag number assigned to request messages by the Requester used to track response messages when multiple request messages are outstanding. Response messages shall use the tag number from the corresponding Request message.
2h	1	Reserved
3h	2	Command Opcode[15:0] : As defined in Table 8-37 , Table 8-126 , and Table 8-215 .
5h	2	Message Payload Length[15:0] : Expressed in bytes. As defined in Table 8-37 , Table 8-126 , and Table 8-215 .
7h	1	<ul style="list-style-type: none"> Bits[4:0]: Message Payload Length[20:16]: Expressed in bytes. As defined in Table 8-37, Table 8-126, and Table 8-215. Bits[6:5]: Reserved. Bit[7]: Background Operation (BO): As defined in Section 8.2.8.4.6.

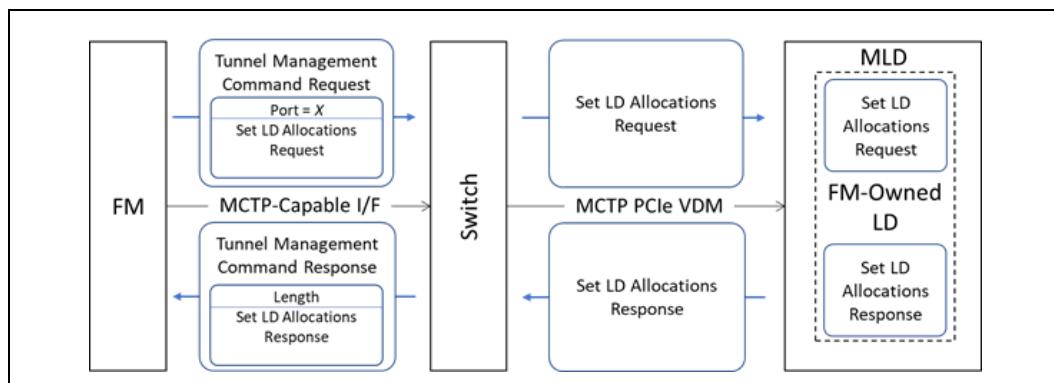
Table 7-14. CCI Message Format (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
8h	2	Return Code[15:0] : As defined in Table 8-34 . Must be 0 for Request messages.
Ah	2	Vendor Specific Extended Status[15:0] : As defined in Section 8.2.8.4.6 . Must be 0 for Request messages.
Ch	Varies	Message Payload : Variably sized payload for message in little-endian format. The length of this field is specified in the Message Payload Length[20:0] fields above. The format depends on Opcode and Message Category , as defined in Table 8-37 , Table 8-126 , and Table 8-215 .

Commands from the FM API Command Sets may be transported as MCTP messages as defined in CXL Fabric Manager API over MCTP Binding Specification (DSP0234). All other CCI commands may be transported as MCTP messages as defined by the respective binding specification, such as CXL Type 3 Component Command Interface over MCTP Binding (DSP0281).

7.6.3.1 Transport Details for MLD Components

MLD components that do not implement an MCTP-capable interconnect other than their CXL interface shall expose a CCI through their CXL interface(s) using MCTP PCIe VDM Transport Binding Specification (DSP0238). FMs shall use the Tunnel Management Command to pass requests to the FM-owned LD, as illustrated in [Figure 7-20](#).

Figure 7-20. Tunneling Commands to an MLD through a CXL Switch

7.6.4 CXL Switch Management

Dynamic configuration of a switch by an FM is not required for basic switch functionality, but is required to support MLDs or CXL fabric topologies.

7.6.4.1 Initial Configuration

The non-volatile memory of the switch stores, in a vendor-specific format, all necessary configuration settings that are required to prepare the switch for initial operation. This includes:

- Port configuration, including direction (upstream or downstream), width, supported rates, etc.
- Virtual CXL Switch configuration, including number of vPPBs for each VCS, initial port binding configuration, etc.
- CCI access settings, including any vendor-defined permission settings for management.

7.6.4.2 Dynamic Configuration

After initial configuration is complete and a CCI on the switch is operational, an FM can send Management Commands to the switch.

An FM may perform the following dynamic management actions on a CXL switch:

- Query switch information and configuration details
- Bind or Unbind ports
- Register to receive and handle event notifications from the switch (e.g., Hot-Plug, surprise removal, and failures)

When a switch port is connected to a downstream PCIe switch, and that port is bound to a vPPB, the management of that PCIe switch and its downstream device will be handled by the VCS's host, not the FM.

7.6.4.3 MLD Port Management

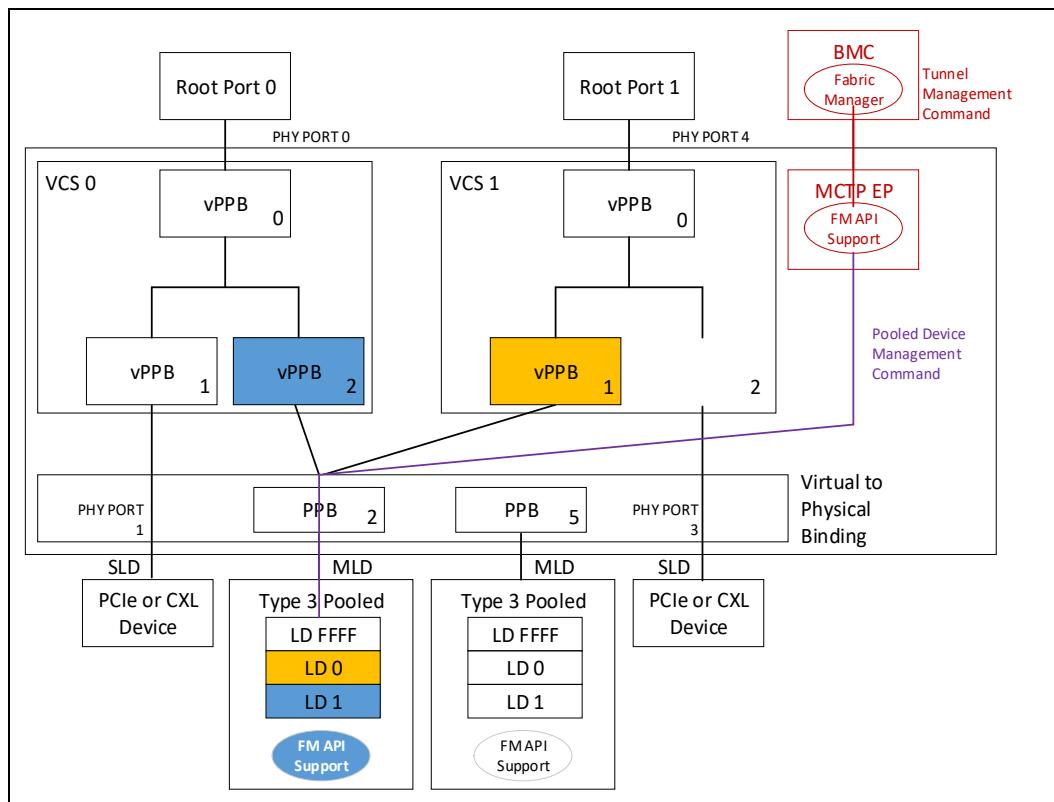
A switch with MLD Ports requires an FM to perform the following management activities:

- MLD discovery
- LD binding/unbinding
- Management Command Tunneling

7.6.5 MLD Component Management

The FM can connect to an MLD over a direct connection or by tunneling its management commands through the CCI of the CXL switch to which the device is connected. The FM can perform the following operations:

- Memory allocation and QoS Telemetry management
- Security (e.g., LD erasure after unbinding)
- Error handling

Figure 7-21. Example of MLD Management Requiring Tunneling

7.6.6

Management Requirements for System Operations

This section presents examples of system use cases to highlight the role and responsibilities of the FM in system management. These use case discussions also serve to itemize the FM commands that CXL devices must support to facilitate each specific system behavior.

7.6.6.1

Initial System Discovery

As the CXL system initializes, the FM can begin discovering all direct attached CXL devices across all supported media interfaces. Devices supporting the FM API may be discovered using transport specific mechanisms such as the MCTP discovery process, as defined in MCTP Base Specification (DSP0236).

When a component is discovered, the FM shall issue the Identify command (see [Section 8.2.9.1.1](#)) prior to issuing any other commands to check the component's type and its maximum supported command message size. A return of "Retry Required" indicates that the component is not yet ready to accept commands. After receiving a successful response to the Identify request, the FM may issue the Set Response Message Limit command (see [Section 8.2.9.1.4](#)) to limit the size of response messages from the component based on the size of the FM's receive buffer. The FM shall not issue any commands with input arguments such that the command's response message exceeds the FM's maximum supported message size. Finally, the FM issues Get Log, as defined in [Section 8.2.9.5.2.1](#), to read the Command Effects Log to determine which command opcodes are supported.

7.6.6.2 CXL Switch Discovery

After a CXL switch is released from reset (i.e., PERST# has been deasserted), it loads its initial configuration from non-volatile memory. Ports configured as DS PPBs will be released from reset to link up. Upon detection of a switch, the FM will query its configuration, capabilities, and connected devices. The **Physical Switch Command Set** is required for all switches implementing FM API support. The **Virtual Switch Command Set** is required for all switches that support multiple host ports.

An example of an FM Switch discovery process is as follows:

1. FM issues **Identify Switch Device** to check switch port count, enabled port IDs, number of supported LDs, and enabled VCS IDs.
2. FM issues **Get Physical Port State** for each enabled port to check port configuration (US or DS), link state, and attached device type. This allows the FM to check for any port link-up issues and create an inventory of devices for binding. If any MLD components are discovered, the FM can begin MLD Port management activities.
3. If the switch supports multiple host ports, FM issues **Get Virtual CXL Switch Info** for each enabled VCS to check for all bound vPPBs in the system and create a list of binding targets.

7.6.6.3 MLD and Switch MLD Port Management

MLDs must be connected to a CXL switch to share their LDs among VCSs. If an MLD is discovered in the system, the FM will need to prepare it for binding. A switch must support the **MLD Port Command Set** to support the use of MLDs. All MLD components shall support the MLD Component Command Set.

1. FM issues management commands to the device's LD FFFFh using **Tunnel Management Command**.
2. FM can execute advanced or vendor-specific management activities, such as encryption or authentication, using the **Send LD CXL.io Configuration Request** and **Send LD CXL.io Memory Request** commands.

7.6.6.4 Event Notifications

Events can occur on both devices and switches. The event types and records are listed in [Section 7.6.8](#) for FM API events and in [Section 8.2.9.2](#) for component events. The Event Records framework is defined in [Section 8.2.9.2.1](#) to provide a standard event record format that all CXL components shall use when reporting events to the managing entity. The managing entity specifies the notification method, such as MSI/MSI-X, EFN VDM, or MCTP Event Notification. The Event Notification message can be signaled by a device or by a switch; the notification always flows toward the managing entity. An Event Record is not sent with the Event Notification message. After the managing entity knows that an event has occurred, the entity can use component commands to read the Event Record.

1. To facilitate some system operations, the FM requires event notifications so it can execute its role in the process in a timely manner (e.g., notifying hosts of an asserted Attention Button on an MLD during a Managed Hot-Removal). If supported by the device, the FM can check and modify the current event notification settings with the Events command set.
2. If supported by the device, the event logs can be read with the **Get Event Records** command to check for any error events experienced by the device that might impact normal operation.

7.6.6.5

Binding Ports and LDs on a Switch

Once all devices, VCSs, and vPPBs have been discovered, the FM can begin binding ports and LDs to hosts as follows:

1. FM issues the **Bind vPPB** command specifying a physical port, VCS ID and vPPB index to bind the physical port to the vPPB. An LD-ID must also be specified if the physical port is connected to an MLD. The switch is permitted to initiate a Managed Hot-Add if the host has already booted, as defined in [Section 9.9](#).
2. Upon completion of the binding process, the switch notifies the FM by generating a **Virtual CXL Switch Event Record**.

7.6.6.6

Unbinding Ports and LDs on a Switch

The FM can unbind devices or LDs from a VCS with the following steps:

1. FM issues the **Unbind vPPB** command specifying a VCS ID and vPPB index to unbind the physical port from the vPPB. The switch initiates a Managed Hot-Remove or Surprise Hot-Remove depending on the command options, as defined in PCIe Base Specification.
2. Upon completion of the unbinding process, the switch will generate a **Virtual CXL Switch Event Record**.

7.6.6.7

Hot-Add and Managed Hot-Removal of Devices

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 and the port links up.
2. FM issues the **Get Physical Port State** command for the physical port that has linked up to discover the connected device type. The FM can now bind the physical port to a vPPB. If it's an MLD, then the FM can proceed with MLD Port management activities; otherwise, the device is ready for binding.

When a device is Hot-Removed from an unbound port on a switch, the FM receives a notification. The switch notifies the FM by generating **Physical Switch Event Records** as the Presence Detect sideband is deasserted and the associated port links down.

1. The switch notifies the FM by generating **Physical Switch Event Records** as the Presence Detect sideband is deasserted and the associated port links down.

When an SLD or PCIe device is Hot-Added to a bound port, the FM can be notified but is not involved. When a Surprise or Managed Hot-Removal of an SLD or PCIe device occurs on a bound port, the FM can be notified but is not involved.

A bound port will not advertise support for MLDs during negotiation, so MLD components will link up as an SLD.

The FM manages managed hot-removal of MLDs as follows:

1. When the Attention Button sideband is asserted on an MLD port, the Attention state bit is updated in the corresponding PPB and vPPB CSRs and the switch notifies the FM and hosts with LDs that are bound and below that MLD port. The hosts are notified with the MSI/MSI-X interrupts assigned to the affected vPPB and a **Virtual CXL Switch Event Record** is generated.
2. As defined in PCIe Base Specification, hosts will read the Attention State bit in their vPPB's CSR and prepare for removal of the LD. When a host is ready for the LD to be removed, it will set the Attention LED bit in the associated vPPB's CSR. The

switch records these CSR updates by generating **Virtual CXL Switch Event Records**. The FM unbinds each assigned LD with the **Unbind vPPB** command as it receives notifications from each host.

- When all host handshakes are complete, the MLD is ready for removal. The FM uses the **Send PPB CXL.io Configuration Request** command to set the Attention LED bit in the MLD port PPB to indicate that the MLD can be physically removed. The timeout value for the host handshakes to complete is implementation specific. There is no requirement for the FM to force the unbind operation, but it can do so using the “Simulate Surprise Hot-Remove” unbinding option in the **Unbind vPPB** command.

7.6.6.8 Surprise Removal of Devices

There are two kinds of surprise removals: physical removal of a device, and surprise Link Down. The main difference between the two is the state of the presence pin, which will be deasserted after a physical removal but will remain asserted after a surprise Link Down. The switch notifies the FM of a surprise removal by generating **Virtual CXL Switch Event Records** for the change in link status and Presence Detect, as applicable.

Three cases of Surprise Removal are described below:

- When a Surprise Removal of a device occurs on an unbound port, the FM must be notified.
- When a Surprise Removal of an SLD or PCIe device occurs on a bound port, the FM is permitted to be notified but must not be involved in any error handling operations.
- When a Surprise Removal of an MLD component occurs, the FM must be notified. The switch will automatically unbind any existing LD bindings. The FM must perform error handling and port management activities, the details of which are considered implementation specific.

7.6.7 Fabric Management Application Programming Interface

The FM manages all devices in a CXL system via the sets of commands defined in the FM API. This specification defines the minimum command set requirements for each device type.

Note: CXL switches and MLDs require FM API support to facilitate the advanced system

Table 7-15. FM API Command Sets

Command Set Name	HBR Switch FM API Requirement ¹	MLD FM API Requirement ¹
Physical Switch (Section 7.6.7.1)	M	P
Virtual Switch (Section 7.6.7.2)	O	P
MLD Port (Section 7.6.7.3)	O	P
MLD Component (Section 7.6.7.4)	P	M
Multi-Headed Device (Section 7.6.7.5)	P	P
DCD Management (Section 7.6.7.6)	P	O
PBR Switch (Section 7.7.13)	P	P
Global Memory Access Endpoint (Section 7.7.14)	P	P

1. M = Mandatory, O = Optional, P = Prohibited.

capabilities outlined in [Section 7.6.6](#). FM API is optional for all other CXL device types.

Command opcodes are listed in [Table 8-215](#). [Table 8-215](#) also identifies the minimum command sets and commands that are required to implement defined system capabilities. The following subsections define the commands grouped in each command set. Within each command set, commands are marked as mandatory (M) or optional (O). If a command set is supported, the required commands within that set must be implemented, but only if the Device supports that command set. For example, the Get Virtual CXL Switch Information command is required in the Virtual Switch Command Set, but that set is optional for switches. This means that a switch does not need to support the Get Virtual CXL Switch Information command if it does not support the Virtual Switch Command Set.

All commands have been defined as stand-alone operations; there are no explicit dependencies between commands, so optional commands can be implemented or not implemented on a per-command basis. Requirements for the implementation of commands are driven instead by the desired system functionality.

7.6.7.1

Physical Switch Command Set

This command set is only supported by and must be supported by CXL switches that have FM API support.

7.6.7.1.1

Identify Switch Device (Opcode 5100h)

This command retrieves information about the capabilities and configuration of a CXL switch.

Possible Command Return Codes:

- Success
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-16. Identify Switch Device Response Payload

Byte Offset	Length in Bytes	Description
00h	1	Ingress Port ID: Ingress CCI port index of the received request message. For CXL/PCIe ports, this corresponds to the physical port number. For non-CXL/PCIe, this corresponds to a vendor-specific index of the buses that the device supports, starting at 0. For example, a request received on the second of 2 SMBuses supported by a device would return a 1.
01h	1	Reserved
02h	1	Number of Physical Ports: Total number of physical ports in the CXL switch, including inactive/disabled ports.
03h	1	Number of VCSs: Maximum number of virtual CXL switches that are supported by the CXL switch.
04h	20h	Active Port Bitmask: Bitmask that defines whether a physical port is enabled (1) or disabled (0). Each bit corresponds 1:1 with a port, with the least significant bit corresponding to Port 0.
24h	20h	Active VCS Bitmask: Bitmask that defines whether a VCS is enabled (1) or disabled (0). Each bit corresponds 1:1 with a VCS ID, with the least significant bit corresponding to VCS 0.
44h	2	Total Number of vPPBs: The number of virtual PPBs that are supported by the CXL switch across all VCSs.
46h	2	Number of Bound vPPBs: Total number of vPPBs, across all VCSs, that are bound.
48h	1	Number of HDM Decoders: Number of HDM decoders available per USP.

7.6.7.1.2 Get Physical Port State (Opcode 5101h)

This command retrieves the physical port information.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-17. Get Physical Port State Request Payload

Byte Offset	Length in Bytes	Description
0h	1	Number of Ports: Number of ports requested.
1h	Varies	Port ID List: 1-byte ID of requested port, repeated Number of Ports times.

Table 7-18. Get Physical Port State Response Payload

Byte Offset	Length in Bytes	Description
0h	1	Number of Ports: Number of port information blocks returned.
1h	3	Reserved
4h	Varies	Port Information List: Port information block as defined in Table 7-19 , repeated Number of Ports times.

Table 7-19. Get Physical Port State Port Information Block Format (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
0h	1	Port ID
1h	1	<ul style="list-style-type: none"> Bits[3:0]: Current Port Configuration State: <ul style="list-style-type: none"> 0h = Disabled 1h = Bind in progress 2h = Unbind in progress 3h = DSP 4h = USP 5h = Fabric Port Fh = Invalid Port_ID; all subsequent field values are undefined All other encodings are reserved Bit[4]: GAE Support: Indicates whether GAE support is present (1) or not present (0) on a port. Valid only for PBR switches if Current Port Configuration State is 4h (USP). Bits[7:5]: Reserved.
2h	1	<ul style="list-style-type: none"> Bits[3:0]: Connected Device Mode: Formerly known as Connected Device CXL Version. This field is reserved for all values of Current Port Configuration State except DSP. <ul style="list-style-type: none"> 0h = Connection is not CXL or is disconnected 1h = RCD mode 2h = CXL 68B Flit and VH mode 3h = Standard 256B Flit mode 4h = CXL Latency-Optimized 256B Flit mode 5h = PBR mode All other encodings are reserved Bits[7:4]: Reserved.
3h	1	Reserved
4h	1	<p>Connected Device Type</p> <ul style="list-style-type: none"> 00h = No device detected 01h = PCIe Device 02h = CXL Type 1 device 03h = CXL Type 2 device or HBR switch 04h = CXL Type 3 SLD 05h = CXL Type 3 MLD 06h = PBR component All other encodings are reserved <p>This field is reserved if Supported CXL Modes is 00h. This field is reserved for all values of Current Port Configuration State except 3h (DSP) or 5h (Fabric Port).</p>
5h	1	<p>Supported CXL Modes: Formerly known as Connected CXL Version. Bitmask that defines which CXL modes are supported (1) or not supported (0) by this port:</p> <ul style="list-style-type: none"> Bit[0]: RCD Mode Bit[1]: CXL 68B Flit and VH Capable Bit[2]: 256B Flit and CXL Capable Bit[3]: CXL Latency-Optimized 256B Flit Capable Bit[4]: PBR Capable Bits[7:5]: Reserved for future CXL use <p>Undefined when the value is 00h.</p>

Table 7-19. Get Physical Port State Port Information Block Format (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
6h	1	<ul style="list-style-type: none"> Bits[5:0]: Maximum Link Width: Value encoding matches the Maximum Link Width field in the PCIe Link Capabilities register in the PCIe Capability structure. Bits[7:6]: Reserved.
7h	1	<ul style="list-style-type: none"> Bits[5:0]: Negotiated Link Width: Value encoding matches the Negotiated Link Width field in PCIe Link Capabilities register in the PCIe Capability structure. Bits[7:6]: Reserved.
8h	1	<ul style="list-style-type: none"> Bits[5:0]: Supported Link Speeds Vector: Value encoding matches the Supported Link Speeds Vector field in the PCIe Link Capabilities 2 register in the PCIe Capability structure. Bits[7:6]: Reserved.
9h	1	<ul style="list-style-type: none"> Bits[5:0]: Max Link Speed: Value encoding matches the Max Link Speed field in the PCIe Link Capabilities register in the PCIe Capability structure. Bits[7:6]: Reserved.
Ah	1	<ul style="list-style-type: none"> Bits[5:0]: Current Link Speed: Value encoding matches the Current Link Speed field in the PCIe Link Status register in the PCIe Capability structure. Bits[7:6]: Reserved.
Bh	1	<p>LTSSM State: Current link LTSSM Major state:</p> <ul style="list-style-type: none"> 00h = Detect 01h = Polling 02h = Configuration 03h = Recovery 04h = L0 05h = L0s 06h = L1 07h = L2 08h = Disabled 09h = Loopback 0Ah = Hot Reset All other encodings are reserved <p>Link substates should be reported through vendor-defined diagnostics commands.</p>
Ch	1	First Negotiated Lane Number
Dh	2	<p>Link State Flags</p> <ul style="list-style-type: none"> Bit[0]: Lane Reversal State: <ul style="list-style-type: none"> 0 = Standard lane ordering 1 = Reversed lane ordering Bit[1]: Port PCIe Reset State (PERST#): <ul style="list-style-type: none"> 0 = Not in reset 1 = In reset Bit[2]: Port Presence Pin State (PRSNT#): <ul style="list-style-type: none"> 0 = Not present 1 = Present Bit[3]: Power Control State: <ul style="list-style-type: none"> 0 = Power on 1 = Power off Bits[15:4]: Reserved
Fh	1	Supported LD Count : Number of additional LDs supported by this port. All ports must support at least one LD.

7.6.7.1.3 Physical Port Control (Opcode 5102h)

This command is used by the FM to control unbound ports and MLD ports, including issuing resets and controlling sidebands.

Possible Command Return Codes:

- Success

- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-20. Physical Port Control Request Payload

Byte Offset	Length in Bytes	Description
0h	1	PPB ID: Physical PPB ID, which corresponds 1:1 to associated physical port number.
1h	1	Port Opcode: Code that defines which operation to perform: <ul style="list-style-type: none"> • 00h = Assert PERST# • 01h = Deassert PERST# • 02h = Reset PPB • All other encodings are reserved

7.6.7.1.4 Send PPB CXL.io Configuration Request (Opcode 5103h)

This command sends CXL.io Config requests to the specified physical port's PPB. This command is only processed for unbound ports and MLD ports.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-21. Send PPB CXL.io Configuration Request Input Payload

Byte Offset	Length in Bytes	Description
0h	1	PPB ID: Target PPB's physical port.
1h	3	<ul style="list-style-type: none"> • Bits[7:0]: Register Number: As defined in PCIe Base Specification • Bits[11:8]: Extended Register Number: As defined in PCIe Base Specification • Bits[15:12]: First Dword Byte Enable: As defined in PCIe Base Specification • Bits[22:16]: Reserved • Bit[23]: Transaction Type: <ul style="list-style-type: none"> — 0 = Read — 1 = Write
4h	4	Transaction Data: Write data. Valid only for write transactions.

Table 7-22. Send PPB CXL.io Configuration Request Output Payload

Byte Offset	Length in Bytes	Description
0h	4	Return Data: Read data. Valid only for read transactions.

7.6.7.1.5 Get Domain Validation SV State (Opcode 5104h)

This command is used by the Host to check the state of the secret value.

Possible Command Return Codes:

- Success
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-23. Get Domain Validation SV State Response Payload

Byte Offset	Length in Bytes	Description
0h	1	Secret Value State: State of the secret value: • 00h = Not set • 01h = Set • All other encodings are reserved

7.6.7.1.6 Set Domain Validation SV (Opcode 5105h)

This command is used by the Host to set the secret value of its VCS. The secret value can be set only once. This command will fail with Invalid Input if it is called more than once.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-24. Set Domain Validation SV Request Payload

Byte Offset	Length in Bytes	Description
0h	10h	Secret Value: UUID used to uniquely identify a host hierarchy.

7.6.7.1.7 Get VCS Domain Validation SV State (Opcode 5106h)

This command is used by the FM to check the state of the secret value in a VCS.

Possible Command Return Codes:

- Success
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-25. Get VCS Domain Validation SV State Request Payload

Byte Offset	Length in Bytes	Description
0h	1	VCS ID: Index of VCS to query.

Table 7-26. Get VCS Domain Validation SV State Response Payload

Byte Offset	Length in Bytes	Description
0h	1	Secret Value State: State of the secret value: <ul style="list-style-type: none"> • 00h = Not set • 01h = Set • All other encodings are reserved

7.6.7.1.8 Get Domain Validation SV (Opcode 5107h)

This command is used by the FM to retrieve the secret value from a VCS.

Possible Command Return Codes:

- Success
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-27. Get Domain Validation SV Request Payload

Byte Offset	Length in Bytes	Description
0h	1	VCS ID: Index of VCS to query.

Table 7-28. Get Domain Validation SV Response Payload

Byte Offset	Length in Bytes	Description
0h	10h	Secret Value: UUID used to uniquely identify a host hierarchy.

7.6.7.2 Virtual Switch Command Set

This command set is supported only by the CXL switch. It is required for switches that support more than one VCS. The following commands are defined:

Table 7-29. Virtual Switch Command Set Requirements

Command Name	Requirement ¹
Get Virtual CXL Switch Info	M
Bind vPPB	O
Unbind vPPB	O
Generate AER Event	O

1. M = Mandatory, O = Optional.

7.6.7.2.1 Get Virtual CXL Switch Info (Opcode 5200h)

This command retrieves information on a specified number of VCSs in the switch. Because of the possibility of variable numbers of vPPBs within each VCS, the returned array has variably sized elements.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 7-30. Get Virtual CXL Switch Info Request Payload

Byte Offset	Length in Bytes	Description
0h	1	Start vPPB: Specifies the ID of the first vPPB for each VCS to include in the vPPB information list in the response (bytes 4 – 7 in Table 7-32). This enables compatibility with devices that have small maximum command message sizes.
1h	1	vPPB List Limit: The maximum number of vPPB information entries to include in the response (bytes 4 – 7 in Table 7-32). This enables compatibility with devices that have small maximum command message sizes. This field shall have a minimum value of 1.
2h	1	Number of VCSs: Number of VCSs requested. This field shall have a minimum value of 1.
3h	Number of VCSs	VCS ID List: 1-byte ID of requested VCS, repeated Number of VCSs times.

Table 7-31. Get Virtual CXL Switch Info Response Payload

Byte Offset	Length in Bytes	Description
0h	1	Number of VCSs: Number of VCS information blocks returned.
1h	3	Reserved
4h	Varies	VCS Information List: VCS information block as defined in Table 7-32 , repeated Number of VCSs times.

Table 7-32. Get Virtual CXL Switch Info VCS Information Block Format

Byte Offset	Length in Bytes	Description
0h	1	Virtual CXL Switch ID
1h	1	VCS State: Current state of the VCS: <ul style="list-style-type: none"> • 00h = Disabled • 01h = Enabled • FFh = Invalid VCS ID; all subsequent field values are invalid • All other encodings are reserved
2h	1	USP ID: Physical port ID of the local Upstream Port, or the local fabric physical port ID of a Downstream ES VCS. Valid only when the VCS is enabled.
3h	1	Number of vPPBs: Total number of vPPBs in the VCS. This value may be larger than the vPPB List Limit field specified in the request. In this case, the length of vPPB information list, starting at byte 4, is defined by 'vPPB List Limit', not by this field. vPPB information list consists of vPPB List Entry Count number of entries and each entry is 4B in length. vPPB List Entry Count=min(vPPB List Limit, Number of vPPBs).
4h	1	vPPB[Start vPPB] Binding Status <ul style="list-style-type: none"> • 00h = Unbound • 01h = Bind or unbind in progress • 02h = Bound Physical Port • 03h = Bound LD • 04h = Bound PID • All other encodings are reserved
5h	2	For PBR Switches when Binding Status is 02h or 03h and for HBR Switches: <ul style="list-style-type: none"> • Bits[7:0]: vPPB[Start vPPB] Bound Port ID: Physical port number of the bound port. Valid only when Binding Status is 02h or 03h. • Bits[15:8]: vPPB[Start vPPB] Bound LD ID: ID of the LD that is bound to the port from the MLD on an associated physical port. Valid only when vPPB[Start vPPB] Binding Status is 03h; otherwise, the value is FFh. For PBR Switches when Binding Status is 04h: <ul style="list-style-type: none"> • Bits[11:0]: vPPB[Start vPPB] Bound PID: PID of the bound vPPB, as defined in Section 7.7.12.3. • Bits[15:12]: Reserved.
7h	1	Reserved
...		...
4 + (vPPB List Entry Count - 1) * 4	1	vPPB[Start vPPB + vPPB List Entry Count¹ - 1] Binding Status: As defined above. <ul style="list-style-type: none"> •
5 + (vPPB List Entry Count - 1) * 4	1	vPPB[Start vPPB + vPPB List Entry Count¹ - 1] Bound Port ID: As defined above.
6 + (vPPB List Entry Count - 1) * 4	1	vPPB[Start vPPB + vPPB List Entry Count¹ - 1] Bound LD ID: As defined above.
7 + (vPPB List Entry Count - 1) * 4	1	Reserved

1. The vPPB information list length is defined by the lesser of the vPPB List Limit field in the command request and the Number of vPPBs field in the command response.

7.6.7.2.2 Bind vPPB (Opcode 5201h)

This command performs a binding operation on the specified vPPB. If the bind target is a physical port connected to a Type 1, Type 2, Type 3, or PCIe device or a physical port whose link is down, the specified physical port of the CXL switch is fully bound to the vPPB. If the bind target is a physical port connected to an MLD, then a corresponding LD-ID must also be specified.

All binding operations are executed as background commands. The switch notifies the FM of binding completion through the generation of event records, as defined in [Section 7.6.6](#).

Possible Command Return Codes:

- Background Command Started
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Busy

Command Effects:

- Background Operation

Table 7-33. Bind vPPB Request Payload

Byte Offset	Length in Bytes	Description
0h	1	Virtual CXL Switch ID
1h	1	vPPB ID: Index of the vPPB within the VCS specified in the VCS ID.
2h	1	Physical Port ID
3h	1	Reserved
4h	2	LD ID: LD-ID if the target port is an MLD port. Must be FFFF for other EP types.

7.6.7.2.3 Unbind vPPB (Opcode 5202h)

This command unbinds the physical port or LD from the virtual hierarchy vPPB. All unbinding operations are executed as background commands. The switch notifies the FM of unbinding completion through the generation of event records, as defined in [Section 7.6.6](#).

Possible Command Return Codes:

- Unsupported
- Background Command Started
- Invalid Input
- Internal Error
- Retry Required
- Busy

Command Effects:

- Background Operation

Table 7-34. **Unbind vPPB Request Payload**

Byte Offset	Length in Bytes	Description
0h	1	Virtual CXL Switch ID
1h	1	vPPB ID: Index of the vPPB within the VCS specified in the VCS ID.
2h	1	<ul style="list-style-type: none"> • Bits[3:0]: Unbind Option: <ul style="list-style-type: none"> — 0h = Wait for port Link Down before unbinding — 1h = Simulate Managed Hot-Remove — 2h = Simulate Surprise Hot-Remove — All other encodings are reserved • Bits[7:4]: Reserved

7.6.7.2.4 Generate AER Event (Opcode 5203h)

This command generates an AER event on a specified VCS's vPPB (US vPPB or DS vPPB). The switch must respect the Host's AER mask settings.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-35. **Generate AER Event Request Payload**

Byte Offset	Length in Bytes	Description
0h	1	Virtual CXL Switch ID
1h	1	vPPB Instance: The value of 0 represents USP. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
2h	2	Reserved
4h	4	AER Error <ul style="list-style-type: none"> • Bits[4:0]: <ul style="list-style-type: none"> — If Severity=0, bit position of the error type in the AER Correctable Error Status register, as defined in PCIe Base Specification — If Severity=1, bit position of the error type in the AER Uncorrectable Error Status register, as defined in PCIe Base Specification • Bits[30:5]: Reserved • Bit[31]: Severity <ul style="list-style-type: none"> — 0 = Correctable Error — 1 = Uncorrectable Error
8h	20h	AER Header: TLP Header to place in AER registers, as defined in PCIe Base Specification.

7.6.7.3 MLD Port Command Set

This command set is applicable to CXL switches and MLDs. The following commands are defined:

Table 7-36. MLD Port Command Set Requirements

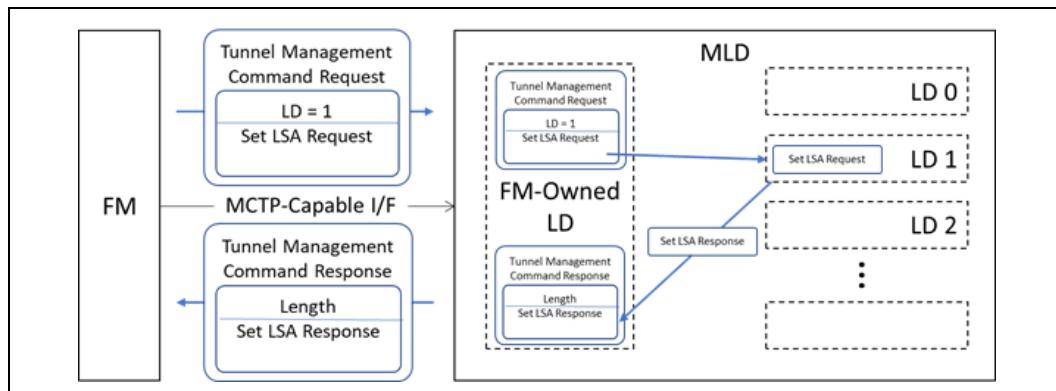
Command Name	Requirement	
	Switches ¹	MLDs ¹
Tunnel Management Command	M	O
Send LD CXL.io Configuration Request	M	P
Send LD CXL.io Memory Request	M	P

1. M = Mandatory, O = Optional, P = Prohibited.

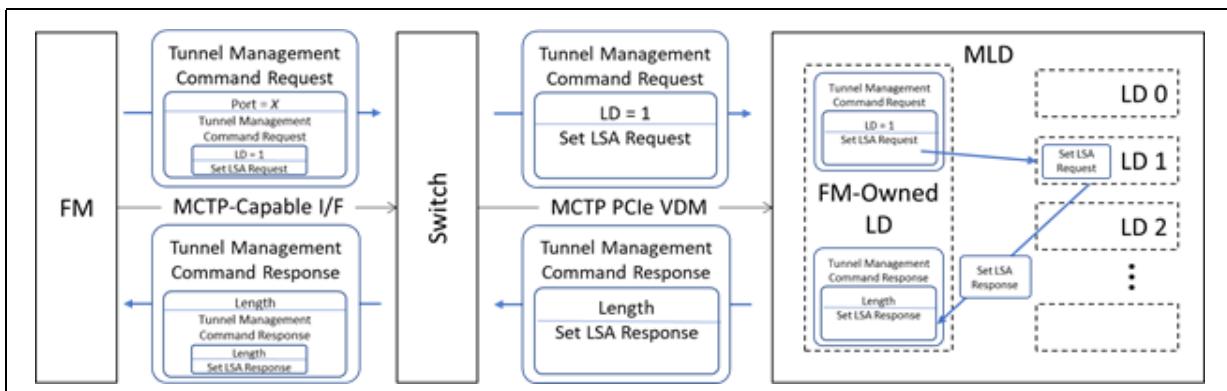
7.6.7.3.1 Tunnel Management Command (Opcode 5300h)

This command tunnels the provided command to LD FFFFh of the MLD on the specified port, using the transport defined in [Section 7.6.3.1](#).

When sent to an MLD, this provided command is tunneled by the FM-owned LD to the specified LD, as illustrated in [Figure 7-22](#).

Figure 7-22. Tunnelling Commands to an LD in an MLD

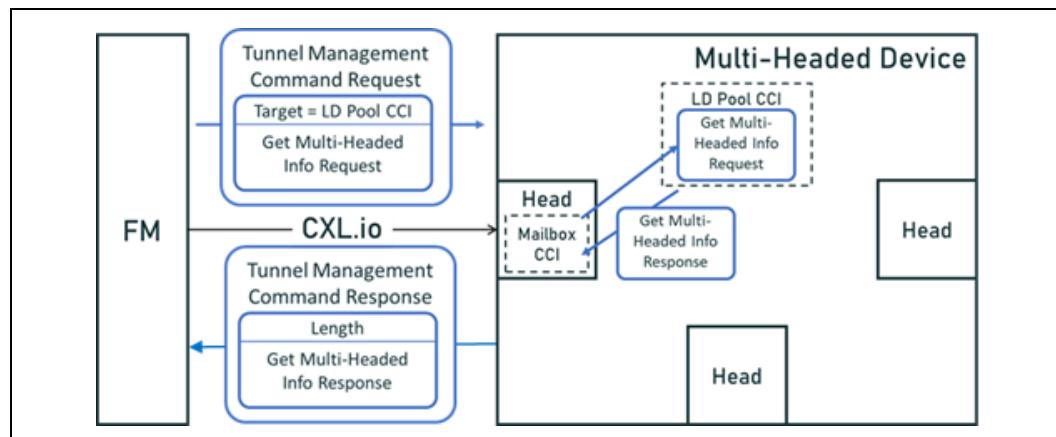
The Management Command input payload field includes the tunneled command encapsulated in the CCI Message Format, as defined in [Figure 7-19](#). This can include an additional layer of tunneling for commands issued to LDs in an MLD that is accessible only through a CXL switch's MLD Port, as illustrated in [Figure 7-23](#).

Figure 7-23. Tunnelling Commands to an LD in an MLD through a CXL Switch

Response size varies, based on the tunneled FM command's definition. Valid targets for the tunneled commands include switch MLD Ports, valid LDs within an MLD, and the LD Pool CCI in a Multi-Headed device. Tunneled commands sent to any other targets shall be discarded and this command shall return an "Invalid Input" return code. The FM-owned LD (LD=FFFFh) is an invalid target in MLDs.

The LD Pool CCI in Multi-Headed devices is targeted using the "Target Type" field, as illustrated in [Figure 7-24](#). This command shall return an "Invalid Input" return code failure if tunneling to the LD Pool CCI is not permitted on the CCI that receives the request.

Figure 7-24. Tunneling Commands to the LD Pool CCI in a Multi-Headed Device



A Multi-Headed device shall terminate the processing of a request that includes more than 3 layers of tunneling and return the Unsupported return code.

The Tunnel Management Command itself does not cause any Command Effects, but the Management Command provided in the request will cause Command Effects as per its definition.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-37. Tunnel Management Command Request Payload

Byte Offset	Length in Bytes	Description
0h	1	Port or LD ID: Egress port ID for commands sent to a switch, or LD-ID for commands sent to an MLD. Valid only when Target Type is 0.
1h	1	<ul style="list-style-type: none"> Bits[3:0]: Target Type: Specifies the type of tunneling target for this command: <ul style="list-style-type: none"> 0h = Port or LD based. Indicates that the “Port or LD ID” field is used to determine the target. 1h = LD Pool CCI. Indicates that the tunneling target is the LD Pool CCI of a Multi-Headed device. All other encodings are reserved. Bits[7:4]: Reserved
2h	2	Command Size: Number of valid bytes in Management Command .
4h	Varies	Management Command: Request message formatted in the CCI Message Format as defined in Figure 7-19 .

Table 7-38. Tunnel Management Command Response Payload

Byte offset	Length in Bytes	Description
0h	2	Response Length: Number of valid bytes in Response Message .
2h	2	Reserved
4h	Varies	Response Message: Response message formatted in the CCI Message Format as defined in Figure 7-19 .

7.6.7.3.2 Send LD CXL.io Configuration Request (Opcode 5301h)

This command allows the FM to read or write the CXL.io Configuration Space of an unbound LD or FMLD. The switch will convert the request into CfgRd/CfgWr TLPs to the target device. Invalid Input Return Code shall be generated if the requested LD is bound.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-39. Send LD CXL.io Configuration Request Payload

Byte Offset	Length in Bytes	Description
0h	1	PPB ID: Target PPB's physical port.
1h	3	<ul style="list-style-type: none"> Bits[7:0]: Register Number: As defined in PCIe Base Specification Bits[11:8]: Extended Register Number: As defined in PCIe Base Specification Bits[15:12]: First Dword Byte Enable: As defined in PCIe Base Specification Bits[22:16]: Reserved Bit[23]: Transaction Type: <ul style="list-style-type: none"> 0 = Read 1 = Write
4h	2	LD ID: Target LD-ID.
6h	2	Reserved
8h	4	Transaction Data: Write data. Valid only for write transactions.

Table 7-40. Send LD CXL.io Configuration Response Payload

Byte Offset	Length in Bytes	Description
0h	4	Return Data: Read data. Valid only for read transactions.

7.6.7.3.3 Send LD CXL.io Memory Request (Opcode 5302h)

This command allows the FM to batch read or write the CXL.io Memory Space of an unbound LD or FMLD. The switch will convert the request into MemRd/MemWr TLPs to the target device. Invalid Input Return Code shall be generated if the requested LD is bound.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-41. Send LD CXL.io Memory Request Payload

Byte Offset	Length in Bytes	Description
00h	1	Port ID: Target MLD port.
01h	2	<ul style="list-style-type: none"> Bits[11:0]: Reserved Bits[15:12]: First Dword Byte Enable: As defined in PCIe Base Specification Bits[19:16]: Last Dword Byte Enable: As defined in PCIe Base Specification Bits[22:20]: Reserved Bit[23]: Transaction Type: <ul style="list-style-type: none"> 0 = Read 1 = Write
04h	2	LD ID: Target LD-ID.
06h	2	Transaction Length: Transaction length in bytes, up to a maximum of 4 KB (1000h).
08h	8	Transaction Address: The target HPA that points into the target device's MMIO Space.
10h	Varies	Transaction Data: Write data. Valid only for write transactions.

Table 7-42. Send LD CXL.io Memory Request Response Payload

Byte Offset	Length in Bytes	Description
0h	2	Return Size: Number of successfully transferred bytes.
2h	2	Reserved
4h	Varies	Return Data: Read data. Valid only for read transactions.

7.6.7.4

MLD Component Command Set

This command set is only supported by, and must be supported by, MLD components implementing FM API support. These commands are processed by MLDs. When an FM is connected to a CXL switch that supports the FM API and does not have a direct connection to an MLD, these commands are passed to the MLD using the **Tunnel Management Command**. The following commands are defined:

Table 7-43. MLD Component Command Set Requirements

Command Name	Requirement ¹
Get LD Info	M
Get LD Allocations	M
Set LD Allocations	O
Get QoS Control	M
Set QoS Control	M
Get QoS Status	O
Get QoS Allocated BW	M
Set QoS Allocated BW	M
Get QoS BW Limit	M
Set QoS BW Limit	M

1. M = Mandatory, O = Optional.

7.6.7.4.1 Get LD Info (Opcode 5400h)

This command retrieves the configurations of the MLD.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-44. Get LD Info Response Payload

Byte Offset	Length in Bytes	Description
0h	8	Memory Size: Total device memory capacity.
8h	2	LD Count: Number of logical devices supported.
Ah	1	QoS Telemetry Capability: Optional QoS Telemetry for memory MLD capabilities for management by an FM (see Section 3.3.4). <ul style="list-style-type: none"> • Bit[0]: Egress Port Congestion Supported: When set, the associated feature is supported and the Get QoS Status command must be implemented (see Section 3.3.4.3.9). • Bit[1]: Temporary Throughput Reduction Supported: When set, the associated feature is supported (see Section 3.3.4.3.5). • Bits[7:2]: Reserved.

7.6.7.4.2 Get LD Allocations (Opcode 5401h)

This command retrieves the memory allocations of the MLD.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 7-45. Get LD Allocations Request Payload

Byte Offset	Length in Bytes	Description
0h	1	Start LD ID: ID of the first LD in the LD Allocation List.
1h	1	LD Allocation List Limit: Maximum number of LD information blocks returned. This field shall have a minimum value of 1.

Table 7-46. Get LD Allocations Response Payload

Byte Offset	Length in Bytes	Description
0h	1	Number of LDs: Number of LDs enabled in the device.
1h	1	Memory Granularity: This field specifies the granularity of the memory sizes configured for each LD: <ul style="list-style-type: none"> • 0h = 256 MB • 1h = 512 MB • 2h = 1 GB • All other encodings are reserved
2h	1	Start LD ID: ID of the first LD in the LD Allocation List.
3h	1	LD Allocation List Length: Number of LD information blocks returned. This value is the lesser of the request's 'LD Allocation List Limit' and response's 'Number of LDs'.
4h	Varies	LD Allocation List: LD Allocation blocks for each LD, as defined in Table 7-47 , repeated LD Allocation List Length times.

Table 7-47. LD Allocations List Format

Byte Offset	Length in Bytes	Description
0h	8	Range 1 Allocation Multiplier: Memory Allocation Range 1 for LD. This value is multiplied with Memory Granularity to calculate the memory allocation range in bytes.
8h	8	Range 2 Allocation Multiplier: Memory Allocation Range 2 for LD. This value is multiplied with Memory Granularity to calculate the memory allocation range in bytes.

7.6.7.4.3 Set LD Allocations (Opcode 5402h)

This command sets the memory allocation for each LD. This command will fail if the device fails to allocate any of the allocations defined in the request. The allocations provided in the response reflect the state of the LD allocations after the command is processed, which allows the FM to check for partial success.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- Configuration Change after Cold Reset
- Configuration Change after Conventional Reset
- Configuration Change after CXL Reset
- Immediate Configuration Change
- Immediate Data Change

Table 7-48. Set LD Allocations Request Payload

Byte Offset	Length in Bytes	Description
0h	1	Number of LDs: Number of LDs to configure. This field shall have a minimum value of 1.
1h	1	Start LD ID: ID of the first LD in the LD Allocation List.
2h	2	Reserved
4h	Varies	LD Allocation List: LD Allocation blocks for each LD, starting at Start LD ID, as defined in Table 7-47 , repeated Number of LDs times.

Table 7-49. Set LD Allocations Response Payload

Byte Offset	Length in Bytes	Description
0h	1	Number of LDs: Number of LDs configured.
1h	1	Start LD ID: ID of the first LD in the LD Allocation List.
2h	2	Reserved
4h	Varies	LD Allocation List: Updated LD Allocation blocks for each LD, starting at Start LD ID, as defined in Table 7-47 , repeated Number of LDs times.

7.6.7.4.4 Get QoS Control (Opcode 5403h)

This command retrieves the MLD's QoS control parameters.

Possible Command Return Codes:

- Success
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 7-50. Payload for Get QoS Control Response, Set QoS Control Request, and Set QoS Control Response (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
0h	1	QoS Telemetry Control: Default is 00h. <ul style="list-style-type: none"> • Bit[0]: Egress Port Congestion Enable: See Section 3.3.4.3.9 • Bit[1]: Temporary Throughput Reduction Enable: See Section 3.3.4.3.5 • Bits[7:2]: Reserved
1h	1	Egress Moderate Percentage: Threshold in percent for Egress Port Congestion mechanism to indicate moderate congestion. Valid range is 1-100. Default is 10.
2h	1	Egress Severe Percentage: Threshold in percent for Egress Port Congestion mechanism to indicate severe congestion. Valid range is 1-100. Default is 25.

Table 7-50. **Payload for Get QoS Control Response, Set QoS Control Request, and Set QoS Control Response (Sheet 2 of 2)**

Byte Offset	Length in Bytes	Description
3h	1	Backpressure Sample Interval: Interval in ns for Egress Port Congestion mechanism to take samples. Valid range is 0-15. Default is 8 (800 ns of history for 100 samples). Value of 0 disables the mechanism. See Section 3.3.4.3.4 .
4h	2	ReqCmpBasis: Estimated maximum sustained sum of requests and recent responses across the entire device, serving as the basis for QoS Limit Fraction. Valid range is 0-65,535. Value of 0 disables the mechanism. Default is 0. See Section 3.3.4.3.7 .
6h	1	Completion Collection Interval: Interval in ns for Completion Counting mechanism to collect the number of transmitted responses in a single counter. Valid range is 0-255. Default is 64 (1.024 us of history, given 16 counters). See Section 3.3.4.3.10 .

7.6.7.4.5 Set QoS Control (Opcode 5404h)

This command sets the MLD's QoS control parameters, as defined in [Table 7-50](#). The device must complete the set operation before returning the response. The command response returns the resulting QoS control parameters, as defined in the same table. This command will fail, returning Invalid Input, if any of the parameters are outside their valid range.

Possible Command Codes:

- Success
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- Immediate Policy Change

Payload for Set QoS Control Request and Response is documented in [Table 7-50](#).

7.6.7.4.6 Get QoS Status (Opcode 5405h)

This command retrieves the MLD's QoS Status. This command is mandatory if the Egress Port Congestion Supported bit is set (see [Table 7-44](#)).

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 7-51. Get QoS Status Response Payload

Byte Offset	Length in Bytes	Description
0h	1	Backpressure Average Percentage: Current snapshot of the measured Egress Port average congestion. See Section 3.3.4.3.4 .

7.6.7.4.7**Get QoS Allocated BW (Opcode 5406h)**

This command retrieves the MLD's QoS allocated bandwidth on a per-LD basis (see [Section 3.3.4.3.7](#)).

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 7-52. Payload for Get QoS Allocated BW Request

Byte Offset	Length in Bytes	Description
0h	1	Number of LDs: Number of LDs queried. This field shall have a minimum value of 1.
1h	1	Start LD ID: ID of the first LD in the QoS Allocated BW List.

Table 7-53. Payload for Get QoS Allocated BW Response

Byte Offset	Length in Bytes	Description
0h	1	Number of LDs: Number of LDs queried.
1h	1	Start LD ID: ID of the first LD in the QoS Allocated BW List.
2h	Number of LDs	QoS Allocation Fraction: Byte array of allocated bandwidth fractions for LDs, starting at Start LD ID. The valid range of each array element is 0-255. Default value is 0. Value in each byte is the fraction multiplied by 256.

7.6.7.4.8**Set QoS Allocated BW (Opcode 5407h)**

This command sets the MLD's QoS allocated bandwidth on a per-LD basis, as defined in [Section 3.3.4.3.7](#). The device must complete the set operation before returning the response. The command response returns the resulting QoS allocated bandwidth, as defined in the same table. This command will fail, returning Invalid Input, if any of the parameters are outside their valid range.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error

- Retry Required
- Invalid Payload Length

Command Effects:

- Configuration Change after Cold Reset
- Configuration Change after Conventional Reset
- Configuration Change after CXL Reset
- Immediate Configuration Change
- Immediate Data Change

Table 7-54. **Payload for Set QoS Allocated BW Request, and Set QoS Allocated BW Response**

Byte Offset	Length in Bytes	Description
0h	1	Number of LDs: Number of LDs configured.
1h	1	Start LD ID: ID of the first LD in the QoS Allocated BW List.
2h	Number of LDs	QoS Allocation Fraction: Byte array of allocated bandwidth fractions for LDs, starting at Start LD ID. The valid range of each array element is 0-255. Default value is 0. Value in each byte is the fraction multiplied by 256.

7.6.7.4.9 Get QoS BW Limit (Opcode 5408h)

This command retrieves the MLD’s QoS bandwidth limit on a per-LD basis (see [Section 3.3.4.3.7](#)).

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 7-55. **Payload for Get QoS BW Limit Request**

Byte Offset	Length in Bytes	Description
0h	1	Number of LDs: Number of LDs queried. This field shall have a minimum value of 1.
1h	1	Start LD ID: ID of the first LD in the QoS BW Limit List.

Table 7-56. **Payload for Get QoS BW Limit Response**

Byte Offset	Length in Bytes	Description
0h	1	Number of LDs: Number of LDs queried.
1h	1	Start LD ID: ID of the first LD in the QoS BW Limit List.
2h	Number of LDs	QoS Limit Fraction: Byte array of allocated bandwidth limit fractions for LDs, starting at Start LD ID. The valid range of each array element is 0-255. Default value is 0. Value in each byte is the fraction multiplied by 256.

7.6.7.4.10 Set QoS BW Limit (Opcode 5409h)

This command sets the MLD's QoS bandwidth limit on a per-LD basis, as defined in [Section 3.3.4.3.7](#). The device must complete the set operation before returning the response. The command response returns the resulting QoS bandwidth limit, as defined in the same table. This command will fail, returning Invalid Input, if any of the parameters are outside their valid range. This command will fail, returning Internal Error, if the device was able to set the QoS BW Limit for some of the LDs in the request, but not all the LDs.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- Configuration Change after Cold Reset
- Configuration Change after Conventional Reset
- Configuration Change after CXL Reset
- Immediate Configuration Change
- Immediate Data Change

Table 7-57. Payload for Set QoS BW Limit Request, and Set QoS BW Limit Response

Byte Offset	Length in Bytes	Description
0h	1	Number of LDs: Number of LDs configured.
1h	1	Start LD ID: ID of the first LD in the QoS BW Limit List.
2h	Number of LDs	QoS Limit Fraction: Byte array of allocated bandwidth limit fractions for LDs, starting at Start LD ID. The valid range of each array element is 0-255. Default value is 0. Value in each byte is the fraction multiplied by 256.

7.6.7.5

Multi-Headed Device Command Set

The Multi-Headed device command set includes commands for querying the Head-to-LD mapping in a Multi-Headed device. Support for this command set is required on the LD Pool CCI of a Multi-Headed device.

7.6.7.5.1

Get Multi-Headed Info (Opcode 5500h)

This command retrieves the number of heads, number of supported LDs, and Head-to-LD mapping of a Multi-Headed device.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-58. Get Multi-Headed Info Request Payload

Byte Offset	Length in Bytes	Description
0h	1	Start LD ID: ID of the first LD in the LD Map.
1h	1	LD Map List Limit: Maximum number of LD Map entries returned. This field shall have a minimum value of 1.

Table 7-59. Get Multi-Headed Info Response Payload

Byte Offset	Length in Bytes	Description
0h	1	Number of LDs: Total number of LDs in the LD Pool. This field shall have a minimum value of 1.
1h	1	Number of Heads: Total number of CXL heads. This field shall have a minimum value of 1.
2h	2	Reserved
4h	1	Start LD ID: ID of the first LD in the LD Map.
5h	1	LD Map Length: Number of LD Map entries returned. LD Map Length = Min (LD Map List Limit, (Number of LDs - Start LD ID))
6h	2	Reserved
8h	LD Map Length	LD Map: Port number of the head to which each LD is assigned, starting at Start LD ID, repeated LD Map Length times. A value of FFh indicates that LD is not currently assigned to a head.

7.6.7.5.2 Get Head Info (Opcode 5501h)

This command retrieves information for one or more heads.

This command fails with the Invalid Input return code if the values of the Start Head and Number of Heads fields request the information for a non-existent head.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-60. Get Head Info Request Payload

Byte Offset	Length in Bytes	Description
0h	1	Start Head: Specifies the ID of the first head information block requested.
1h	1	Number of Heads: Number of head information blocks requested.

Table 7-61. Get Head Info Response Payload

Byte Offset	Length in Bytes	Description
0h	1	Number of Heads: Number of head information blocks returned.
1h	3	Reserved
4h	Varies	Head Information List: Head information block as defined in Table 7-62 , repeated Number of Heads times.

Table 7-62. Get Head Info Head Information Block Format

Byte Offset	Length in Bytes	Description
0h	1	Port Number: Value encoding matches the Port Number field in the PCIe Link Capabilities register in the PCIe Capability structure.
1h	1	<ul style="list-style-type: none"> Bits[5:0]: Maximum Link Width: Value encoding matches the Maximum Link Width field in the PCIe Link Capabilities register in the PCIe Capability structure Bits[7:6]: Reserved
2h	1	<ul style="list-style-type: none"> Bits[5:0]: Negotiated Link Width: Value encoding matches the Negotiated Link Width field in the PCIe Link Capabilities register in the PCIe Capability structure Bits[7:6]: Reserved
3h	1	<ul style="list-style-type: none"> Bits[5:0]: Supported Link Speeds Vector: Value encoding matches the Supported Link Speeds Vector field in the PCIe Link Capabilities 2 register in the PCIe Capability structure Bits[7:6]: Reserved
4h	1	<ul style="list-style-type: none"> Bits[5:0]: Max Link Speed: Value encoding matches the Max Link Speed field in the PCIe Link Capabilities register in the PCIe Capability structure Bits[7:6]: Reserved
5h	1	<ul style="list-style-type: none"> Bits[5:0]: Current Link Speed: Value encoding matches the Current Link Speed field in the PCIe Link Status register in the PCIe Capability structure Bits[7:6]: Reserved
6h	1	LTSSM State: Current link LTSSM Major state: <ul style="list-style-type: none"> 00h = Detect 01h = Polling 02h = Configuration 03h = Recovery 04h = L0 05h = L0s 06h = L1 07h = L2 08h = Disabled 09h = Loopback 0Ah = Hot Reset All other encodings are reserved Link substates should be reported through vendor-defined diagnostics commands.
7h	1	First Negotiated Lane Number
8h	1	Link State Flags <ul style="list-style-type: none"> Bit[0]: Lane Reversal State: <ul style="list-style-type: none"> 0 = Standard lane ordering 1 = Reversed lane ordering Bit[1]: Port PCIe Reset State (PERST#): <ul style="list-style-type: none"> 0 = Not in reset 1 = In reset Bits[7:2]: Reserved

7.6.7.6 DCD Management Command Set

The DCD Management command set includes commands for querying and configuring Dynamic Capacity. It is used by the FM to manage memory assignment within a DCD. Memory management for GFDs is defined in [Section 8.2.9.9.10](#).

7.6.7.6.1 Get DCD Info (Opcode 5600h)

This command retrieves the number of supported hosts, total Dynamic Capacity of the device, and supported region configurations. To retrieve the corresponding DCD info for a GFD, see [Section 8.2.9.9.10.1](#).

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-63. Get DCD Info Response Payload (Sheet 1 of 2)

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 host. DCDs shall report between 1 and 8 regions. All other encodings are reserved.
02h	2	Reserved
04h	2	<ul style="list-style-type: none"> • Bits[3:0]: Supported Add Capacity Selection Policies: Bitmask that specifies the selection policies, as defined in Section 7.6.7.6.5, that the device supports when capacity is added. At least one policy shall be supported. A value of 1 indicates that a policy is supported, and a value of 0 indicates that a policy is not supported: <ul style="list-style-type: none"> — Bit[0]: Free — Bit[1]: Contiguous — Bit[2]: Prescriptive — Bit[3]: Must be 0 • Bits[15:4]: Reserved
06h	2	Reserved
08h	2	<ul style="list-style-type: none"> • Bits[1:0]: Supported Release Capacity Removal Policies: Bitmask that specifies the removal policies, as defined in Section 7.6.7.6.6, that the device supports when capacity is released. At least one policy shall be supported. A value of 1 indicates that a policy is supported, and a value of 0 indicates that a policy is not supported: <ul style="list-style-type: none"> — Bit[0]: Tag-based — Bit[1]: Prescriptive • Bits[15:2]: Reserved
0Ah	1	Sanitize on Release Configuration Support Mask: Bitmask, where bit position corresponds to region number, indicating whether the Sanitize on Release capability is configurable (1) or not configurable (0) for that region.
0Bh	1	Reserved
0Ch	8	Total Dynamic Capacity: Total memory media capacity of the device available for dynamic assignment to any host in multiples of 256 MB.

Table 7-63. Get DCD Info Response Payload (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
14h	8	Region 0 Supported Block Size Mask: Indicates the block sizes that the region supports. Each bit indicates a power of 2 supported block size, where bit n being set indicates that block size 2^n is supported. Bits[5:0] and bits[63:52] shall be 0. At least one block size shall be supported.
1Ch	8	Region 1 Supported Block Size Mask: As defined in Region 0 Supported Block Size Mask. Valid only if Number of Supported Regions > 1.
24h	8	Region 2 Supported Block Size Mask: As defined in Region 0 Supported Block Size Mask. Valid only if Number of Supported Regions > 2.
2Ch	8	Region 3 Supported Block Size Mask: As defined in Region 0 Supported Block Size Mask. Valid only if Number of Supported Regions > 3.
34h	8	Region 4 Supported Block Size Mask: As defined in Region 0 Supported Block Size Mask. Valid only if Number of Supported Regions > 4.
3Ch	8	Region 5 Supported Block Size Mask: As defined in Region 0 Supported Block Size Mask. Valid only if Number of Supported Regions > 5.
44h	8	Region 6 Supported Block Size Mask: As defined in Region 0 Supported Block Size Mask. Valid only if Number of Supported Regions > 6.
4Ch	8	Region 7 Supported Block Size Mask: As defined in Region 0 Supported Block Size Mask. Valid only if Number of Supported Regions > 7.

7.6.7.6.2 Get Host DC Region Configuration (Opcode 5601h)

This command retrieves the Dynamic Capacity configuration for a specified host.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-64. Get Host DC Region Configuration Request Payload

Byte Offset	Length in Bytes	Description
0h	2	Host ID: For an LD-FAM device, the LD-ID of the host interface configuration to query.
2h	1	Region Count: The maximum number of region configurations to return in the output payload.
3h	1	Starting Region Index: Index of the first requested region.

Table 7-65. Get Host DC Region Configuration Response Payload (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
0h	2	Host ID: For an LD-FAM device, the LD-ID of the host interface configuration returned.
2h	1	Number of Available Regions: As defined in Get Dynamic Capacity Configuration Output Payload.

Table 7-65. Get Host DC Region Configuration Response Payload (Sheet 2 of 2)

3h	1	Number of Regions Returned: The number of entries in the Region Configuration List.
4h	Varies	Region Configuration List: DC Region Info for region specified via Starting Region Index input field. The format of each entry is defined in Table 7-66 .
Varies	4	Total Number of Supported Extents: Total number of extents that the device supports on this LD.
Varies	4	Number of Available Extents: Remaining number of extents that the device supports, as defined in Section 9.13.3.3 .
Varies	4	Total Number of Supported Tags: Total number of Tag values that the device supports on this LD.
Varies	4	Number of Available Tags: Remaining number of Tag values that the device supports, as defined in Section 9.13.3.3 .

Table 7-66. DC Region Configuration

Byte Offset	Length in Bytes	Description
00h	8	Region Base: As defined in Table 8-165 .
08h	8	Region Decode Length: As defined in Table 8-165 .
10h	8	Region Length: As defined in Table 8-165 .
18h	8	Region Block Size: As defined in Table 8-165 .
20h	1	<p>Note: More than one bit may be set at a time.</p> <ul style="list-style-type: none"> • Bits[1:0]: Reserved • Bit[2]: NonVolatile: As defined in the Flags field of Device Scoped Memory Affinity Structure defined in Coherent Device Attribute Table (CDAT) Specification • Bit[3]: Sharable: As defined in the Flags field of Device Scoped Memory Affinity Structure defined in CDAT Specification • Bit[4]: Hardware Managed Coherency: As defined in the Flags field of Device Scoped Memory Affinity Structure defined in CDAT Specification • Bit[5]: Interconnect specific Dynamic Capacity Management: As defined in the Flags field of Device Scoped Memory Affinity Structure defined in CDAT Specification • Bit[6]: Read-Only: As defined in the Flags field of Device Scoped Memory Affinity Structure defined in CDAT Specification • Bit[7]: Reserved
21h	3	Reserved
24h	1	<ul style="list-style-type: none"> • Bit[0]: Sanitize on Release: As defined in Table 8-165 • Bits[7:1]: Reserved
25h	3	Reserved

7.6.7.6.3

Set DC Region Configuration (Opcode 5602h)

This command sets the configuration of a DC Region. This command shall be processed only when all capacity has been released from the region on all LDs. The device shall generate an Event Record of type Region Configuration Updated upon successful processing of this command.

This command shall fail with Unsupported under the following conditions:

- When all capacity has been released from the DC Region on all hosts, and one or more blocks are allocated to the specified region
- When the Sanitize on Release field does not match the region's configuration, as reported from the Get Host DC Region Configuration, and the device does not support reconfiguration of the Sanitize on Release setting, as advertised by the

Sanitize on Release Configuration Support Mask in the Get DCD Info response payload

This command shall fail with Invalid Security State under the following condition:

- In support of confidential computing, if the device has been locked while utilizing secure CXL TSP interfaces, the device shall reject any attempts to change the DCD configuration by returning Invalid Security State status. See [Section 11.5](#) for details on locking a device and locked device behavior.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Security State

Command Effects:

- Configuration Change after Cold Reset
- Configuration Change after Conventional Reset
- Configuration Change after CXL Reset
- Immediate Configuration Change
- Immediate Data Change

Table 7-67. Set DC Region Configuration Request and Response Payload

Byte Offset	Length in Bytes	Description
0h	1	Region ID: Specifies which region to configure. Valid range is from 0 to 7.
1h	3	Reserved
4h	8	Region Block Size: As defined in Table 8-165 .
Ch	1	<ul style="list-style-type: none"> • Bit[0]: Sanitize on Release: As defined in Table 8-165 • Bits[7:1]: Reserved
Dh	3	Reserved

7.6.7.6.4 Get DC Region Extent Lists (Opcode 5603h)

This command retrieves the Dynamic Capacity Extent List for a specified host.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-68. Get DC Region Extent Lists Request Payload

Byte Offset	Length in Bytes	Description
0h	2	Host ID: For an LD-FAM device, the LD-ID of the host interface.
2h	2	Reserved
4h	4	Extent Count: The maximum number of extents to return in the output response. The device may not return more extents than requested; however, it can return fewer extents. 0 is valid and allows the FM to retrieve the Total Extent Count and Extent List Generation Number without retrieving any extent data.
8h	4	Starting Extent Index: Index of the first requested extent. A value of 0 will retrieve the first extent in the list.

Table 7-69. Get DC Region Extent Lists Response Payload

Byte Offset	Length in Bytes	Description
00h	2	Host ID: For an LD-FAM device, the LD-ID of the host interface query.
02h	2	Reserved
04h	4	Starting Extent Index: Index of the first extent in the list.
08h	4	Returned Extent Count: The number of extents returned in Extent List[].
0Ch	4	Total Extent Count: The total number of extents in the list.
10h	4	Extent List Generation Number: A device-generated value that is used to indicate that the list has changed.
14h	4	Reserved
18h	Varies	Extent List[]: Extent list for the specified host as defined in Table 8-51 .

7.6.7.6.5 Initiate Dynamic Capacity Add (Opcode 5604h)

This command initiates the addition of Dynamic Capacity to the specified region on a host. This command shall complete when the device initiates the Add Capacity procedure, as defined in [Section 8.2.9.2.2](#). The processing of the actions initiated in response to this command may or may not result in a new entry in the Dynamic Capacity Event Log. To perform Dynamic Capacity Add on a GFD, see [Section 8.2.9.9.10.7](#).

A Selection Policy is specified to govern the device's selection of which memory resources to add:

- **Free:** Unassigned extents are selected by the device, with no requirement for contiguous blocks
- **Contiguous:** Unassigned extents are selected by the device and shall be contiguous
- **Prescriptive:** Extent list of capacity to assign is included in the request payload
- **Enable Shared Access:** Enable access to extent(s) previously added to another host in a DC Region that reports the "Sharable" flag, as designated by the specified tag value

See [Section 9.13.3.2](#) for examples of how this command may be used to set up different types of sharing arrangements.

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 Selection Policy
- When the Length field is not a multiple of the Block size and the Selection Policy is either Free or Contiguous

The command, with selection policy Enable Shared Access, shall also fail with Invalid Input under the following conditions:

- When the specified region is not Sharable
- When the tagged capacity is already mapped to any Host ID via a non-Sharable region
- When the tagged capacity cannot be added to the requested region due to device-imposed restrictions
- When the same tagged capacity is currently accessible by the same LD

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, or if there is insufficient contiguous space to satisfy a request with Selection Policy set to Contiguous.

The command shall fail with Invalid Extent List under the following conditions:

- When the Selection Policy is set to Prescriptive and the Extent Count is invalid
- When the Selection Policy is set to Prescriptive and any of the DPAs are already accessible to the same LD

The command shall fail with Resources Exhausted if the Extent List would cause the device to exceed its extent or tag tracking ability.

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

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Extent List
- Resources Exhausted

Command Effects:

- Configuration Change after Cold Reset
- Configuration Change after Conventional Reset
- Configuration Change after CXL Reset
- Immediate Configuration Change
- Immediate Data Change

Table 7-70. Initiate Dynamic Capacity Add Request Payload

Byte Offset	Length in Bytes	Description
00h	2	Host ID: For an LD-FAM device, the LD-ID of the host interface to which the capacity is being added.
02h	1	<ul style="list-style-type: none"> • Bits[3:0]: Selection Policy: Specifies the policy to use for selecting which extents comprise the added capacity: <ul style="list-style-type: none"> — 0h = Free — 1h = Contiguous — 2h = Prescriptive — 3h = Enable Shared Access — All other encodings are reserved • Bits[7:4]: Reserved
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.
04h	8	Length: The number of bytes of capacity to add. Always a multiple of the configured Region Block Size returned in Get DCD Info. Shall be > 0. This field is reserved when the Selection Policy is set to Prescriptive or Enable Shared Access.
0Ch	10h	Tag: Context field utilized by implementations that make use of the Dynamic Capacity feature. This field is reserved when the Selection Policy is set to Prescriptive.
1Ch	4	Extent Count: The number of extents in the Extent List. Present only when the Selection Policy is set to Prescriptive.
20h	Varies	Extent List: Extent list of capacity to add as defined in Table 8-51 . Present only when the Selection Policy is set to Prescriptive.

7.6.7.6.6**Initiate Dynamic Capacity Release (Opcode 5605h)**

This command initiates the release of Dynamic Capacity from a host. This command shall complete when the device initiates the Remove Capacity procedure, as defined in [Section 8.2.9.9.9](#). The processing of the actions initiated in response to this command may or may not result in a new entry in the Dynamic Capacity Event Log. To perform Dynamic Capacity removal on a GFD, see [Section 8.2.9.9.10.8](#).

A removal policy is specified to govern the device's selection of which memory resources to remove:

- **Tag-based:** Extents are selected by the device based on tag, with no requirement for contiguous extents
- **Prescriptive:** Extent list of capacity to release is included in request payload

To remove a host's access to the shared extent, the FM issues Initiate Dynamic Capacity Release Request with Selection Policy=Tag-Based with the Host ID associated with that host. The Tag field must match the Tag value used during Capacity Add. The host access can be removed in any order. The physical memory resources and tag associated with a shared extent shall remain assigned and unavailable for re-use until that extent has been released from all hosts that have been granted access.

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 Tag represents sharable capacity, and the Extent List covers only a portion of the capacity associated with the Tag

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 occurs regardless of whether an Event is logged.

The command shall fail with Resources Exhausted if the Extent List would cause the device to exceed its extent or tag tracking ability.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Extent List
- Resources Exhausted

Command Effects:

- Configuration Change after Cold Reset
- Configuration Change after Conventional Reset
- Configuration Change after CXL Reset
- Immediate Configuration Change
- Immediate Data Change

Table 7-71. Initiate Dynamic Capacity Release Request Payload (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	2	Host ID: For an LD-FAM device, the LD-ID of the host interface from which the capacity is being released.
02h	1	Flags <ul style="list-style-type: none"> • Bits[3:0]: Removal Policy: Specifies the policy to use for selecting which extents comprise the released capacity: <ul style="list-style-type: none"> — 0h = Tag-based — 1h = Prescriptive — All other encodings are reserved • Bit[4]: Forced Removal: <ul style="list-style-type: none"> — 1 = Device does not wait for a Release Dynamic Capacity command from the host. Host immediately loses access to released capacity. • Bit[5]: Sanitize on Release: <ul style="list-style-type: none"> — 1 = Device shall sanitize all released capacity as a result of this request using the method described in Section 8.2.9.9.5.1. If this is a shared capacity, the sanitize operation shall be performed after the last host has released the capacity. • Bits[7:6]: Reserved
03h	1	Reserved
04h	8	Length: The number of bytes of capacity to remove. Always a multiple of the configured Region Block Size returned in Get DCD Info. Shall be > 0. This field is reserved when the Removal Policy is set to Prescriptive.

Table 7-71. **Initiate Dynamic Capacity Release Request Payload (Sheet 2 of 2)**

Byte Offset	Length in Bytes	Description
0Ch	10h	Tag: Optional opaque context field utilized by implementations that make use of the Dynamic Capacity feature. This field is reserved when the Removal Policy is set to Prescriptive.
1Ch	4	Extent Count: The number of extents in the Extent List. Present only when the Removal Policy is set to Prescriptive.
20h	Varies	Extent List: Extent list of capacity to release as defined in Table 8-51 . Present only when the Removal Policy is set to Prescriptive.

7.6.7.6.7**Dynamic Capacity Add Reference (Opcode 5606h)**

This command prevents the tagged sharable capacity from being sanitized, freed, and/or reallocated, regardless of whether it is currently visible to any hosts via extent lists. The tagged capacity will remain allocated, and contents will be preserved even if all DCD Extents that reference it are removed.

This command has no effect and will return Success if the FM has already added a reference to the tagged capacity.

This command shall return Invalid Input if the Tag in the payload does not match an existing sharable tag.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- Configuration Change after Cold Reset
- Configuration Change after Conventional Reset
- Configuration Change after CXL Reset
- Immediate Configuration Change

Table 7-72.**Dynamic Capacity Add Reference Request Payload**

Byte Offset	Length in Bytes	Description
00h	10h	Tag: Tag that is associated with the memory capacity to be preserved.

7.6.7.6.8**Dynamic Capacity Remove Reference (Opcode 5607h)**

This command removes a reference to tagged sharable capacity that was previously added via Dynamic Capacity Add Reference (see [Section 7.6.7.6.7](#)). If there are no remaining extent lists that reference the tagged capacity, the memory will be freed and sanitized if appropriate.

This command shall return Invalid Input if the Tag in the payload does not match an existing sharable tag.

Possible Command Return Codes:

- Success

- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- Configuration Change after Cold Reset (if freed)
- Configuration Change after Conventional Reset (if freed)
- Configuration Change after CXL Reset (if freed)
- Immediate Configuration Change (if freed)

Table 7-73. Dynamic Capacity Remove Reference Request Payload

Byte Offset	Length in Bytes	Description
00h	10h	Tag: Tag that is associated with the memory capacity.

7.6.7.6.9 Dynamic Capacity List Tags (Opcode 5608h)

This command allows an FM to re-establish context by receiving a list of all existing tags, with bitmaps indicating which LDs have access, and a flag indicating whether the FM holds a reference.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error

Command Effects:

- None

Table 7-74. Dynamic Capacity List Tags Request Payload

Byte Offset	Length in Bytes	Description
00h	04h	Starting Index: Index of the first tag to return.
04h	04h	Max Tags: Maximum number of tags to return in the response payload. If Max Tags is 0, no tags list will be returned; however, the Generation Number shall be valid.

Table 7-75. Dynamic Capacity List Tags Response Payload (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	4	Generation Number: Generation number of the tags list. This number shall change every time the remainder of the command's payload would change.
04h	4	Total Number of Tags: Maximum number of tags to return in the response payload.
08h	4	Number of Tags Returned: Number of tags returned in the Tags List.

Table 7-75. Dynamic Capacity List Tags Response Payload (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
0Ch	1	Validity Bitmap <ul style="list-style-type: none"> Bit[0]: Reference Bitmaps Valid: A value of 1 indicates that the Reference Bitmap fields in the Tags List are valid. This bit shall be 0 for GFDs and 1 for all other device types. Bit[1]: Pending Reference Bitmaps Valid: A value of 1 indicates that the Pending Reference Bitmap fields in the Tags List are valid. This bit shall be 0 for GFDs and 1 for all other device types. Bits[7:2]: Reserved.
0Dh	3	Reserved
10h	Varies	Tags List : List of Dynamic Capacity Tag Information structures. The format of each entry is defined in Table 7-76 .

Table 7-76. Dynamic Capacity Tag Information

Byte Offset	Length in Bytes	Description
00h	10h	Tag : Tag that is associated with the memory capacity.
10h	1	Flags <ul style="list-style-type: none"> Bit[0]: FM Holds Reference: When set, this bit indicates that the FM holds a reference on this Tag. Bits[7:1]: Reserved.
11h	3	Reserved
14h	20h	Reference Bitmap : Each 1 indicates an LD that has accepted the capacity associated with this tag. Bit 0 of the first byte represents LD 0, and bit 7 of the last byte represents LD 255. This field is reserved if the Reference Bitmaps Valid bit is not set in the Dynamic Capacity List Tags Response Payload (see Table 7-75).
34h	20h	Pending Reference Bitmap : Each 1 indicates an LD for which the tagged capacity has been added with no host response yet. Bit 0 of the first byte represents LD 0, and bit 7 of the last byte represents LD 255. This field is reserved if the Pending Reference Bitmaps Valid bit is not set in the Dynamic Capacity List Tags Response Payload (see Table 7-75).

7.6.8 Fabric Management Event Records

The FM API uses the Event Records framework defined in [Section 8.2.9.2.1](#). This section defines the format of event records specific to Fabric Management activities.

7.6.8.1 Physical Switch Event Records

Physical Switch Event Records define events that are related to physical switch ports, as defined in [Table 7-77](#).

Table 7-77. Physical Switch Events Record Format

Byte Offset	Length in Bytes	Description
00h	30h	Common Event Record: See corresponding common event record fields defined in Section 8.2.9.2.1 . The Event Record Identifier field shall be set to 77cf9271-9c02-470b-9fe4-bc7b75f2da97, which identifies a Physical Switch Event Record.
30h	1	Physical Port ID: Physical Port that is generating the event.
31h	1	Event Type: Identifies the type of event that occurred: <ul style="list-style-type: none"> 00h = Link State Change 01h = Slot Status Register Updated
32h	2	Slot Status Register: As defined in PCIe Base Specification.
34h	1	Reserved
35h	1	<ul style="list-style-type: none"> Bits[3:0]: Current Port Configuration State: See Table 7-19 Bits[7:4]: Reserved
36h	1	<ul style="list-style-type: none"> Bits[3:0] Connected Device Mode: See Table 7-19 Bits[7:4]: Reserved
37h	1	Reserved
38h	1	Connected Device Type: See Table 7-19
39h	1	Supported CXL Modes: See Table 7-19
3Ah	1	<ul style="list-style-type: none"> Bits[5:0]: Maximum Link Width: Value encoding matches the Maximum Link Width field in the PCIe Link Capabilities register in the PCIe Capability structure Bits[7:6]: Reserved
3Bh	1	<ul style="list-style-type: none"> Bits[5:0]: Negotiated Link Width: Value encoding matches the Negotiated Link Width field in the PCIe Link Capabilities register in the PCIe Capability structure Bits[7:6]: Reserved
3Ch	1	<ul style="list-style-type: none"> Bits[5:0]: Supported Link Speeds Vector: Value encoding matches the Supported Link Speeds Vector field in the PCIe Link Capabilities 2 register in the PCIe Capability structure Bits[7:6]: Reserved
3Dh	1	<ul style="list-style-type: none"> Bits[5:0]: Max Link Speed: Value encoding matches the Max Link Speed field in the PCIe Link Capabilities register in the PCIe Capability structure Bits[7:6]: Reserved
3Eh	1	<ul style="list-style-type: none"> Bits[5:0]: Current Link Speed: Value encoding matches the Current Link Speed field in the PCIe Link Status register in the PCIe Capability structure Bits[7:6]: Reserved
3Fh	1	LTSSM State: See Section 7.6.7.1 .
40h	1	First Negotiated Lane Number: Lane number of the lowest lane that has negotiated.
41h	2	Link state flags: See Section 7.6.7.1 .
43h	3Dh	Reserved

7.6.8.2 Virtual CXL Switch Event Records

Virtual CXL Switch Event Records define events that are related to VCSs and vPPBs, as defined in [Table 7-78](#).

Table 7-78. Virtual CXL Switch Event Record Format

Byte Offset	Length in Bytes	Description
00h	30h	Common Event Record: See corresponding common event record fields defined in Section 8.2.9.2.1 . The Event Record Identifier field shall be set to 40d26425-3396-4c4d-a5da-3d47263af425, which identifies a Virtual Switch Event Record.
30h	1	VCS ID
31h	1	vPPB ID
32h	1	Event Type: Identifies the type of event that occurred: <ul style="list-style-type: none"> • 00h = Binding Change • 01h = Secondary Bus Reset • 02h = Link Control Register Updated • 03h = Slot Control Register Updated
33h	1	vPPB Binding Status: Current vPPB binding state, as defined in Table 7-32 . If Event Type is 00h, this field contains the updated binding state of a vPPB following the binding change. Successful bind and unbind operations generate events to the Informational Event Log. Failed bind and unbind operations generate events to the Warning Event Log.
34h	1	vPPB Port ID: Current vPPB bound port ID, as defined in Table 7-32 . If Event Type is 00h, this field contains the updated binding state of a vPPB following the binding change. Successful bind and unbind operations generate events to the Informational Event Log. Failed bind and unbind operations generate events to the Warning Event Log.
35h	1	vPPB LD ID: Current vPPB bound LD-ID, as defined in Table 7-32 . If Event Type is 00h, this field contains the updated binding state of a vPPB following the binding change. Successful bind and unbind operations generate events to the Informational Event Log. Failed bind and unbind operations generate events to the Warning Event Log.
36h	2	Link Control Register Value: Current Link Control register value, as defined in PCIe Base Specification.
38h	2	Slot Control Register Value: Current Slot Control register value, as defined in PCIe Base Specification.
3Ah	46h	Reserved

7.6.8.3 MLD Port Event Records

MLD Port Event Records define events that are related to switch ports connected to MLDs, as defined in [Table 7-79](#).

Table 7-79. MLD Port Event Records Payload

Byte Offset	Length in Bytes	Description
00h	30h	Common Event Record: See corresponding common event record fields defined in Section 8.2.9.2.1 . The Event Record Identifier field shall be set to 8dc44363-0c96-4710-b7bf-04bb99534c3f, which identifies an MLD Port Event Record.
30h	1	Event Type: Identifies the type of event that occurred: <ul style="list-style-type: none"> • 00h = Error Correctable Message Received. Events of this type shall be added to the Warning Event Log. • 01h = Error Non-Fatal Message Received. Events of this type shall be added to the Failure Event Log. • 02h = Error Fatal Message Received. Events of this type shall be added to the Failure Event Log.
31h	1	Port ID: ID of the MLD port that is generating the event.
32h	2	Reserved
34h	8	Error Message: The first 8 bytes of the PCIe error message (ERR_COR, ERR_NONFATAL, or ERR_FATAL) that is received by the switch.
3Ch	44h	Reserved

7.7

CXL Fabric Architecture

The CXL fabric architecture adds new features to scale from a node to a rack-level interconnect to service the growing computational needs in many fields. Machine learning/AI, drug discovery, agricultural and life sciences, materials science, and climate modeling are some of the fields with significant computational demand. The computation density required to meet the demand is driving innovation in many areas, including near and in-memory computing. CXL Fabric features provide a robust path to build flexible, composable systems at rack scale that are able to capitalize on simple load/store memory semantics or Unordered I/O (UIO).

CXL fabric extensions allow for topologies of interconnected fabric switches using 12-bit PIDs (SPIDs/DPIDs) to uniquely identify up to 4096 Edge Ports. The following are the main areas of change to extend CXL as an interconnect fabric for server composability and scale-out systems:

- Expand the size of CXL fabric using Port Based Routing and 12-bit PIDs.
- Enable support for G-FAM devices (GFDs). A GFD is a highly scalable memory resource that is accessible by all hosts and all peer devices.
- Host and device peer communication may be enabled using UIO. A future ECN is planned to complete the definition for this use case.

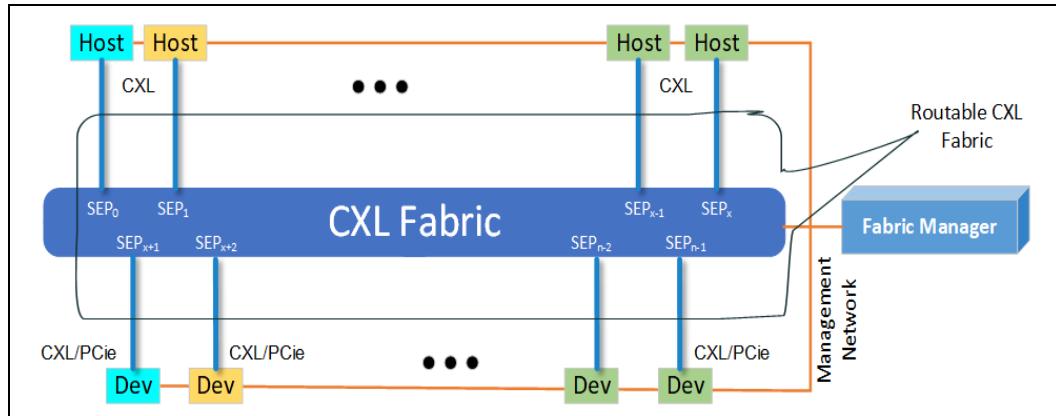
Figure 7-25. High-level CXL Fabric Diagram

Figure 7-25 is a high-level illustration of a routable CXL Fabric. The fabric consists of one or more interconnected fabric switches. In this figure, there are “n” Switch Edge Ports (SEP_i) on the Fabric where each Edge Port can connect to a CXL host root port or a CXL/PCIe device (Dev). As shown, a Fabric Manager (FM) connects to the CXL Fabric and may connect to selected endpoints over an out-of-band management network. The management network may be a simple 2-wire interface, such as SMBus, I2C, I3C, or a complex fabric such as Ethernet. The FM is responsible for the initialization and setup of the CXL Fabric and the assignment of devices to different Virtual Hierarchies. Extensions to FM API (see [Section 7.6](#)) to handle cross-domain traffic will be taken up as a future ECN.

Initially, the FM binds a set of devices to the host’s Virtual Hierarchies, essentially composing a system. After the system has booted, the FM may add or remove devices from the system using fabric bind and unbind operations. These system changes are presented to the hosts by the fabric switches as managed Hot-Add and Hot-Remove events as described in [Section 9.9](#). This allows for dynamic reconfiguration of systems that are composed of hosts and devices.

Root ports on the CXL Fabric may be part of the same or different domains. If the root ports are in different domains, hardware coherency across those root ports is not a requirement. However, devices that support sharing (including MLDs, Multi-Headed devices, and GFDs) may support hardware-managed cache coherency across root ports in multiple domains.

7.7.1

CXL Fabric Use Case Examples

Following are a few examples of systems that may benefit from using CXL-switched Fabric for low-latency communication.

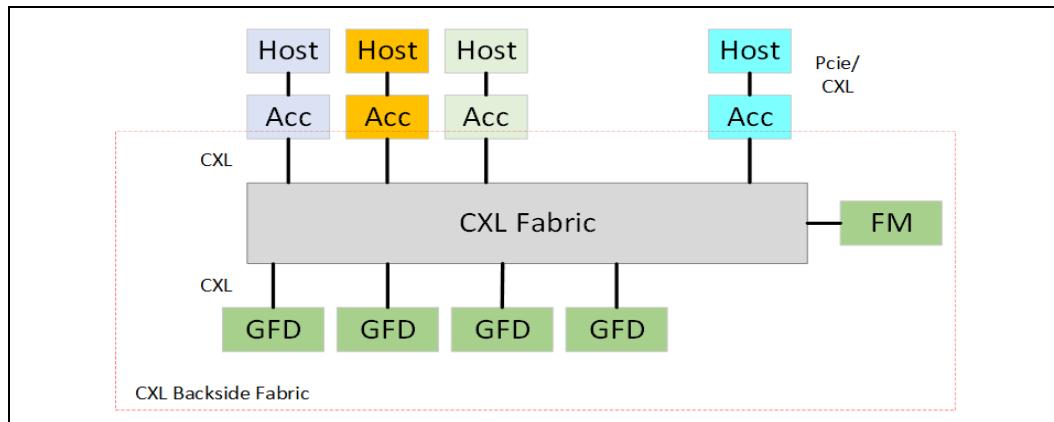
7.7.1.1

Machine-learning Accelerators

Accelerators used for machine-learning applications may use a dedicated CXL-switched Fabric for direct communication between devices in different domains. The same Fabric may also be used for sharing GFDs among accelerators. Each host and accelerator of same color shown in [Figure 7-26](#) (basically, those that are directly above and below one another) belongs to a single domain. Accelerator devices can use UIO transactions to access memory on other accelerator and GFDs. In such a system, each accelerator is attached to a host and expected to be hardware-cache coherent with the host when using a CXL link. Communication between accelerators across domains is via the I/O coherency model. Device caching of data from another device memory (HDM or PDM) requires software-managed coherency with appropriate cache flushes and barriers. A

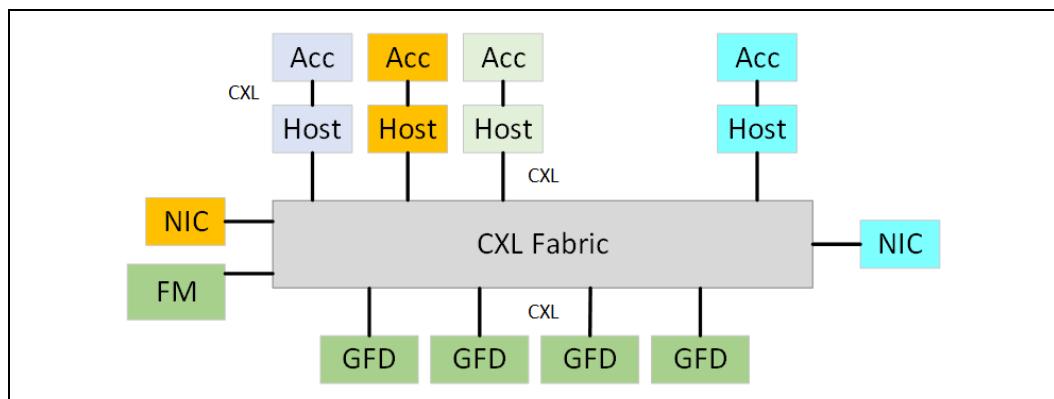
Switching

Switch Edge ingress port is expected to implement a common set of address decoders that is to be used for Upstream Ports and Downstream Ports. Implementations may enable a dedicated CXL Fabric for accelerators using features available in this revision. However, it is not fully defined by the specification. Peer communication is defined in [Section 7.7.9](#).

Figure 7-26. ML Accelerator Use Case**7.7.1.2****HPC/Analytics Use Case**

High-performance computing and Big Data Analytics are two areas that may also benefit from a dedicated CXL Fabric for host-to-host communication and sharing of G-FAM. CXL.mem or UIO may be used to access GFDs. Some G-FAM implementations may enable cross-domain hardware cache coherency. Software cache coherency may still be used for shared-memory implementations. Host-to-host communication is defined in [Section 7.7.3](#).

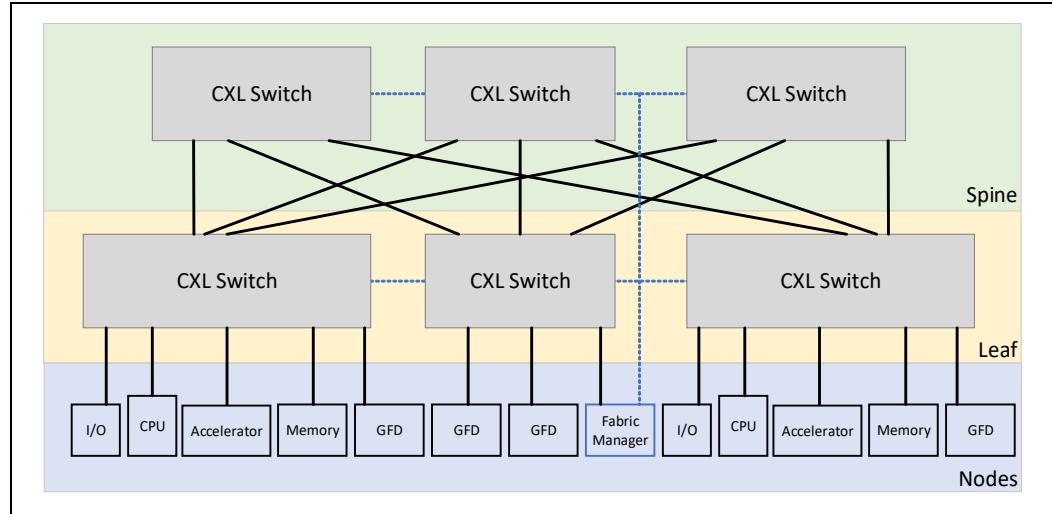
NICs may be used to directly move data from network storage to G-FAM devices, using the UIO traffic class. CXL.mem and UIO use fabric address decoders to route to target GFDs that are members of many domains.

Figure 7-27. HPC/Analytics Use Case**7.7.1.3****Composable Systems**

Support for multi-level switches with PBR fabric extensions provides additional capabilities for building software-composable systems. In [Figure 7-28](#), a leaf/spine switch architecture is shown in which all resources are attached to the leaf switches. Each domain may span multiple switches. All devices must be bound to a host or an FM. Cross-domain traffic is limited to CXL.mem and UIO transactions.

Composing systems from resources within a single leaf switch allows for low-latency implementations. In such implementations, a spine switch is used only for cross-domain and G-FAM accesses.

Figure 7-28. Sample System Topology for Composable Systems



7.7.2 Global-Fabric-Attached Memory (G-FAM)

7.7.2.1 Overview

G-FAM provides a highly scalable memory resource that is accessible by all hosts and peer devices within a CXL fabric. G-FAM ranges can be assigned exclusively to a single host/peer requester or can be shared by multiple hosts/peers. When shared, multi-requester cache coherency can be managed by either software or hardware. Access rights to G-FAM ranges are enforced by decoders in Requester Edge ports and the target GFD.

GFD HDM space can be accessed by hosts/peers from multiple domains using CXL.mem, and by peer devices from multiple domains using CXL.io UIO. GFDs implement no PCI configuration space, and they are configured and managed instead via Global Memory Access Endpoints (GAEs) in Edge USPs or via out-of-band mechanisms.

Unlike an MLD, which has a separate Device Physical Address (DPA) space for each host/peer interface (LD), a GFD has one DPA space that is common across all hosts and peer devices. The GFD translates the Host Physical Address (HPA)¹ in each incoming request into a DPA, using per-requester translation information that is stored within the GFD Decoder Table. To create shared memory, two or more HPA ranges (each from a different requester) are mapped to the same DPA range. When the GFD needs to issue a BISnp, the GFD translates the DPA into an HPA for the associated host using the same GFD decoder information.

When a GFD receives a request, the requester is identified by the SPID in the request, which is referred to as the Requester PID or RPID. Using this term avoids confusion when describing messages that the GFD sends to the requester, where the RPID is used for the DPID, and the GFD PID is used for the SPID.

1. "HPA" is used for peer device requests in addition to host requests, even though "HPA" is a misnomer for some peer-device use cases.

All memory capacity on a GFD is managed by the Dynamic Capacity (DC) mechanisms, as defined in [Section 8.2.9.9.9](#). A GFD allows each requester to access up to 8 RPID non-overlapping decoders, where the maximum number of decoders per SPID is implementation dependent. Each decoder has a translation from HPA space to the common DPA space, a flag that indicates whether cache coherency is maintained by software or hardware, and information about multi-GFD interleaving, if used. For each requester, the FM may define DC Regions in DPA space and convey this information to the host via a GAE. It is expected that the host will program the Fabric Address Segment Table (FAST) decoders and GFD decoders for all RPIDs in its domain to map the entire DPA range of each DC Region that needs to be accessed by the host or by one of its associated accelerators.

G-FAM memory ranges can be interleaved across any power-of-two number of GFDs from 2 to 256, with an Interleave Granularity of 256B, 512B, 1 KB, 2 KB, 4 KB, 8 KB, or 16 KB. GFDs that are located anywhere within the CXL fabric, as defined in [Section 2.7](#), may be used to contribute memory to an Interleave Set.

If a GFD supports UIO Direct P2P to HDM (see [Section 7.7.9.1](#)), all GFD ports shall support UIO, and for each GFD link whose link partner also supports UIO, VC3 shall be auto-enabled by the ports (see [Section 7.7.11.5.1](#)).

7.7.2.2

Host Physical Address View

Hosts that access G-FAM shall allocate a contiguous address range for Fabric Address space within their Host Physical Address (HPA) space, as shown in [Figure 7-29](#). The Fabric Address range is defined by the FabricBase and FabricLimit registers. All host requests that fall within the Fabric Address range are routed to a selected CXL port. Hosts that use multiple CXL ports for G-FAM may either address interleave requests across the ports or may allocate a Fabric Address space for each port.

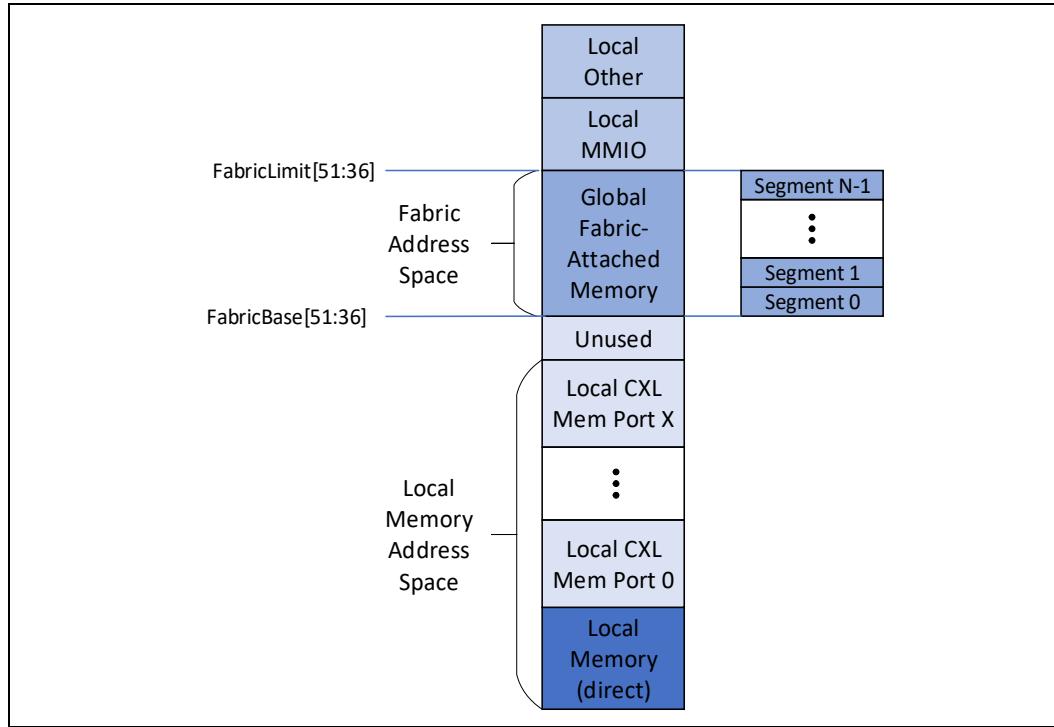
G-FAM requests from a host flow to a PBR Edge USP. In the USP, the Fabric Address range is divided into N equal-sized segments. A segment may be any power-of-two size from 64 GB to 8 TB, and must be naturally aligned. The number of segments implemented by a switch is implementation dependent. Host software is responsible for configuring the segment size so that the number of segments times the segment size fully spans the Fabric Address space. The FabricBase and FabricLimit registers can be programmed to any multiple of the segment size.

Each segment has an associated GFD or Interleave Set of GFDs. Requests whose HPA falls anywhere within the segment are routed to the specified GFD or to a GFD within the Interleave Set. Segments are used only for request routing and may be larger than the accessible portion of a GFD. When this occurs, the accessible portion of the GFD starts at address offset zero within the segment. Any requests within the segment that are above the accessible portion of the GFD will fail to positively decode in the GFD and will be handled as described in [Section 8.2.4.20](#).

Host interleaving across root ports is entirely independent from GFD interleaving. Address bits that are used for root port interleaving and for GFD interleaving may be fully overlapping, partially overlapping, or non-overlapping. When the host uses root port interleaving, FabricBase, FabricLimit, and segment size in the corresponding PBR Edge USPs must be identically configured.

7.7.2.3 G-FAM Capacity Management

Figure 7-29. Example Host Physical Address View



GFDs are managed using CCIs like all other classes of CXL components. A GFD requires support for the PBR Link CCI message format, as defined in [Section 7.7.11.6](#), on its CXL link and may optionally implement additional MCTP-based CCIs (e.g., SMBus).

G-FAM relies exclusively on the Dynamic Capacity (DC) mechanism for capacity management, as described in [Section 8.2.9.9.9](#). GFDs have no “legacy” static capacity as shown in the left side of [Figure 9-24](#) in [Chapter 9.0](#). Dynamic Capacity for G-FAM has much in common with the Dynamic Capacity for LD-FAM:

- Both have identical concepts for DC Regions, Extents, and Blocks
- Both support up to 8 DC Regions per host/peer interface
- DC-related parameters in the CDAT for each are identical
- Mailbox commands for each are highly similar; however, the specific Mailbox access methods are considerably different
 - For LD-FAM, the Mailbox for each host’s LD is accessed via LD structures
 - For G-FAM, management for each host is defined in [Section 7.7.2.6](#)

An LD-FAM DCD (i.e., DCD-capable SLDs or MLDs) allocates memory capacity and binds it to a specific Host ID in one operation. A GFD allocates Dynamic Capacity to a named Memory Group in one operation and binds specific Host IDs to named Memory Groups in a separate operation. Thus, the GFD requires different DCD Management commands than LD-FAM DCDs.

In contrast to LD-FAM, each GFD has a single DPA space instead of a separate DPA space per host. G-FAM DPA space is organized by Device Media Partitions (DMPs), as shown in [Figure 7-30](#). DMPs are DPA ranges with certain attributes. A fundamental DMP

attribute is the media type (e.g., DRAM or PM). A DMP attribute that is configured by the FM is the DC Block size. DMPs expose all GFD memory that is assignable for host use.

The rules for DMPs are as follows:

- Each GFD contains 1-4 DMPs, whose size is configured by the FM.
- Each DC Region consists of part or all of one DMP assigned to a host/peer. Each DC Region can be mapped into an RPID's HPA space using the GFD Decoder Table.
- Each DC Region inherits associated DMP attributes.

Figure 7-30. Example HPA Mapping to DMPs

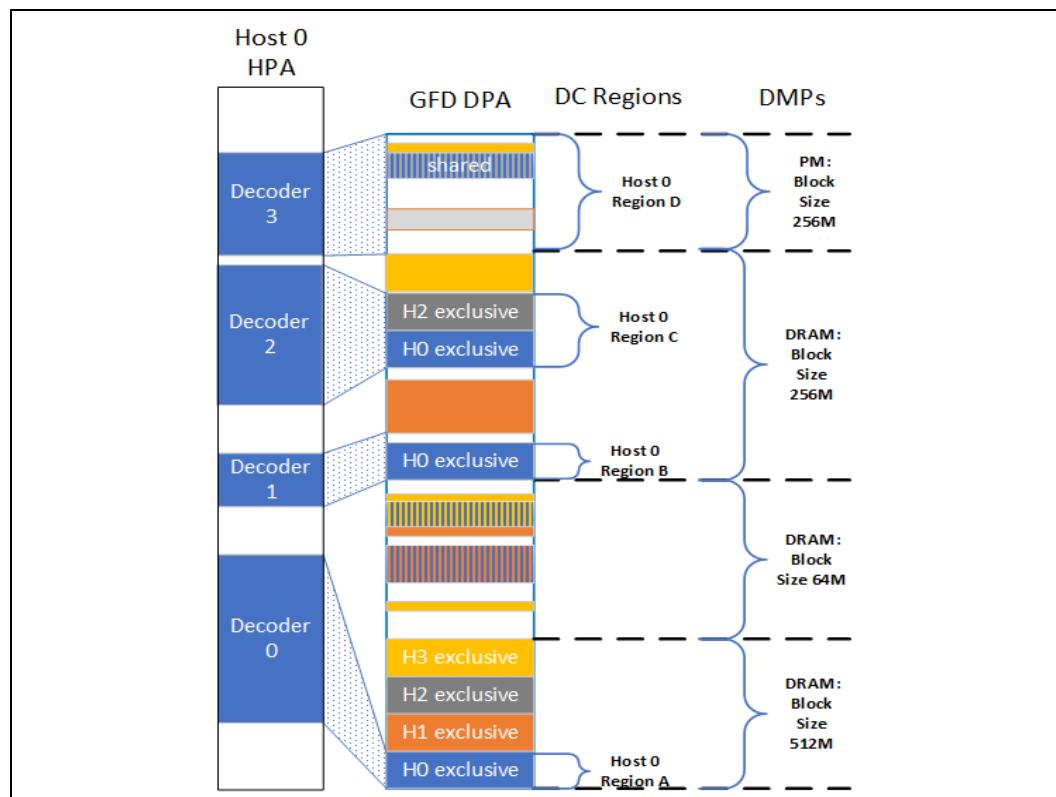


Table 7-80 lists the key differences between LD-FAM and G-FAM.

Table 7-80.

Differences between LD-FAM and G-FAM (Sheet 1 of 2)

Feature or Attribute	LD-FAM	G-FAM
Number of supported hosts	16 max	1000s architecturally; 100s more realistic
Support for DMPs	No	Yes
Architected FM API support for DMP configuration by the FM	N/A	Yes

Table 7-80. Differences between LD-FAM and G-FAM (Sheet 2 of 2)

Feature or Attribute	LD-FAM	G-FAM
Routing and decoders used for HDM addresses	Interleave RP routing by host HDM Decoder Interleave VH routing in USP HDM Decoder or LDST decoder 1–10 HDM Decoders in each LD	Interleave RP routing by host HDM Decoder Interleave fabric routing in USP FAST and Interleave DPID Table (IDT) 1–8 GFD Decoders per RPID in the GFD
Interleave Ways (IW)	$\frac{1}{2}/4/8/16$ plus $3/6/12$	2–256 in powers of 2
DC Block Size	Powers of 2, as indicated by Region * Supported Block Size Mask	64 MB and up in powers of 2

Additional differences exist in how MLDs and GFDs process requests. An MLD has three types of decoders that operate sequentially on incoming requests:

- Per-LD HDM decoders translate from HPA space to a per-LD DPA space, removing the interleaving bits
- Per-LD decoders determine within which per-LD DC Region the DPA resides, and then whether the addressed DC block within the Region is accessible by the LD
- Per-LD implementation-dependent decoders translate from the DPA to the media address

A GFD has two types of decoders that operate sequentially on incoming requests:

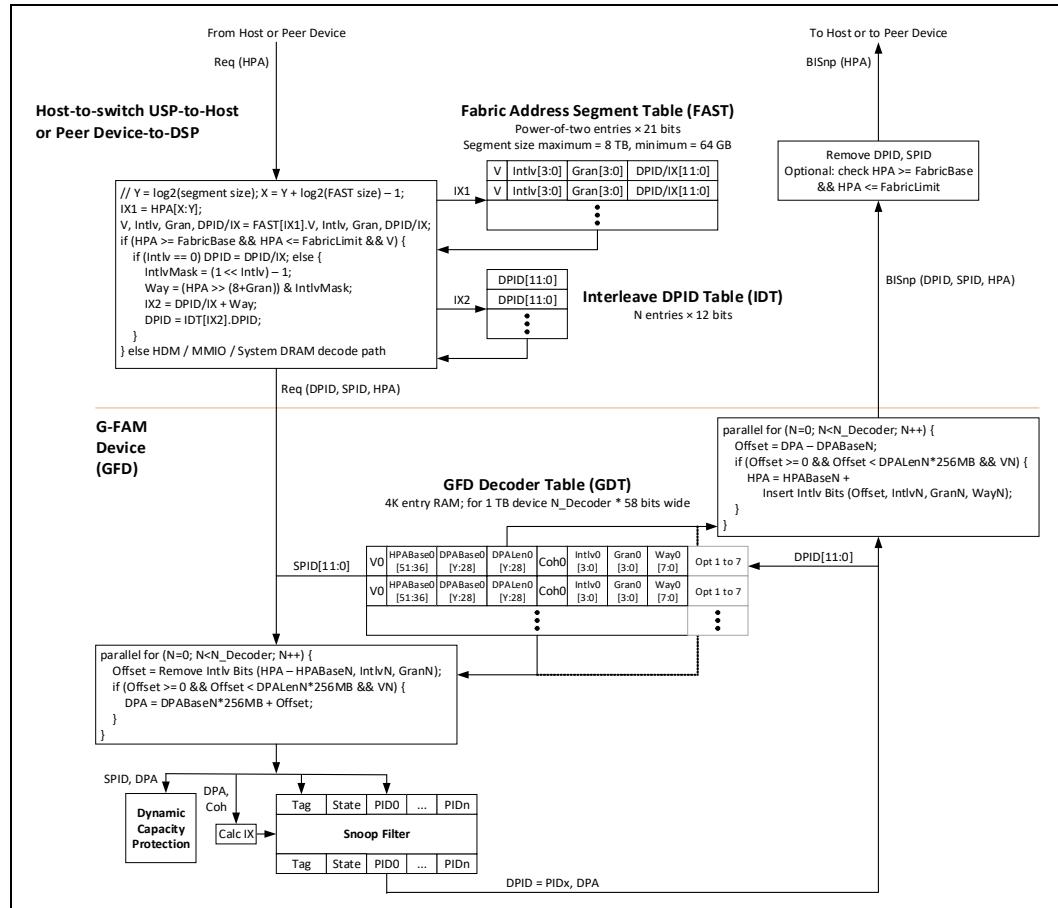
- Per-RPID GFD decoders translate from HPA space to a common DPA space, removing the interleaving bits. This DPA may be used as the media address directly or via a simple mapping.
- A common decoder determines within which Device Media Partition (DMP) the DPA is located, and then whether the block that is addressed within the DMP is accessible by the RPID.

7.7.2.4

G-FAM Request Routing, Interleaving, and Address Translations

The mechanisms for GFD request routing, interleaving, and address translations within both the Edge ingress port and the GFD are shown in [Figure 7-31](#). GFD requests may arrive either at an Edge USP from a host or at an Edge DSP from a peer device. This is referred to as the Edge request port.

Figure 7-31. G-FAM Request Routing, Interleaving, and Address Translations



The Edge request port shall decode the request HPA to determine the DPID of the target GFD using the FAST¹ and the Interleave DPID Table (IDT). The FAST contains one entry per segment. The FAST depth must be a power-of-two but is implementation dependent. The segment size is specified by the FSegSz[2:0] register as defined in [Table 7-81](#). The FAST entry accessed is determined by bits X:Y of the request address, where Y = log₂ of the segment size in bytes and X = Y + log₂ of the FAST depth in entries. The maximum Fabric Address space and the HPA bits that are used to address the FAST are shown in [Table 7-81](#) for all supported segment sizes for some example FAST depths. For a host with a 52-bit HPA, the maximum Fabric Address space is 4 PB minus one segment each above and below the Fabric Address space for local memory and for MMIO, as shown in [Figure 7-29](#).

1. This section covers using FAST decoders with G-FAM. The LD-FAM Segment Table (LDST) decoders used with LD-FAM have identical functionality with few exceptions. [Table 7-111](#), [Table 7-112](#), and [Table 7-113](#) apply to LD-FAM as well as to G-FAM.

Table 7-81. Fabric Segment Size Table¹

FSegSz[2:0]	Fabric Segment Size	FAST Depth (Entries)			
		256	1K	4K	16K
000b	64 GB	16 TB HPA[43:36]	64 TB HPA[45:36]	256 TB HPA[47:36]	1 PB HPA[49:36]
001b	128 GB	32 TB HPA[44:37]	128 TB HPA[46:37]	512 TB HPA[48:37]	2 PB HPA[50:37]
010b	256 GB	64 TB HPA[45:38]	256 TB HPA[47:38]	1 PB HPA[49:38]	4 PB – 512 GB HPA[51:38]
011b	512 GB	128 TB HPA[46:39]	512 TB HPA[48:39]	2 PB HPA[50:39]	
100b	1 TB	256 TB HPA[47:40]	1 PB HPA[49:40]	4 PB – 2 TB HPA[51:40]	
101b	2 TB	512 TB HPA[48:41]	2 PB HPA[50:41]		
110b	4 TB	1 PB HPA[49:42]	4 PB – 8 TB HPA[51:42]		
111b	8 TB	2 PB HPA[50:43]			

1. LDST Segment Size (LSegSz) uses the same encodings as those defined for FSegSz.

Each FAST entry contains a valid bit (V), the number of interleaving ways (Intlv), the interleave granularity (Gran), and a DPID or IDT index (DPID/IX). The encodings for the Intlv and Gran fields are defined in [Table 7-82](#) and [Table 7-83](#), respectively. If the HPA is between FabricBase and FabricLimit inclusive and the FAST entry valid bit is set, then there is a FAST hit, and the FAST is used to determine the DPID. Otherwise, the target device is determined by other architected decoders.

Table 7-82. Segment Table Intlv[3:0] Field Encoding

Intlv[3:0]	GFD Interleaving Ways
0h	Interleaving is disabled
1h	2-way interleaving
2h	4-way interleaving
3h	8-way interleaving
4h	16-way interleaving
5h	32-way interleaving
6h	64-way interleaving
7h	128-way interleaving
8h	256-way interleaving
9h – Fh	Reserved

Table 7-83. Segment Table Gran[3:0] Field Encoding

Gran [3:0]	GFD Interleave Granularity
0h	256B
1h	512B
2h	1 KB
3h	2 KB
4h	4 KB
5h	8 KB
6h	16 KB
7h – Fh	Reserved

Note that FabricBase and FabricLimit may be used to restrict the amount of the FAST used. For example, for a host with a 52-bit HPA space, if the FAST is accessed using HPA[51:40] without restriction, then it would consume the entire HPA space. In this case, FabricBase and FabricLimit must be set to restrict the Fabric Address space to the desired range of HPA space. This has the effect of reducing the number of entries in the FAST that are being used.

FabricBase and FabricLimit may also be used to allow the FAST to start at an HPA that is not a multiple of the FAST depth. For example, for a host with a 52-bit HPA space, if 2 PB of Fabric Address space is needed to start at an HPA of 1 PB, then a 4K entry FAST with 512 GB segments can be accessed using HPA[50:39] with FabricBase set to 1 PB and FabricLimit set to 3 PB. HPAs 1 PB to 2 PB-1 will then correspond to FAST entries 2048 to 4095, while HPAs 2 PB to 3 PB-1 will wrap around and correspond to FAST entries 0 to 2047. When programming FabricBase, FabricLimit, and segment size, care must be taken to ensure that a wraparound does not occur that would result in aliasing multiple HPAs to the same segment.

On a FAST hit, if the FAST Intlv field is 0h, then GFD interleaving is not being used for this segment and the DPID/IX field contains the GFD's DPID. If the Intlv field is nonzero, then the Interleave Way is selected from the HPA using the Gran and Intlv fields, and then added to the DPID/IX field to generate an index into the IDT. The IDT defines the set of DPIDs for each Interleave Set that is accessible by the Edge request port. For an N-way Interleave Set, the set of DPIDs is determined by N contiguous entries in the IDT, with the first entry pointed to by DPID/IX which may be anywhere in the IDT. The IDT depth is implementation dependent.

After the GFD's DPID is determined, a request that contains the SPID of the Edge request port and the unmodified HPA is sent to the target GFD. The GFD shall then use the SPID to access the GFD Decoder Table (GDT) to select the decoders that are associated with the requester. Note that a host and its associated CXL devices will each have a unique RPID, and therefore each will use a different entry in the GDT. The GDT provides up to 8 decoders per RPID. Each decoder within a GFD Decoder Table entry contains structures defined in [Section 8.2.9.9.10.19](#).

The GFD shall then compare, in parallel, the request HPA against all decoders to determine whether the request hits any decoder's HPA range. To accomplish this, for each decoder, a DPA offset is calculated by first subtracting HPABase from HPA and then removing the interleaving bits. The LSB of the interleaving bits to remove is determined by the interleave granularity and the number of bits to remove is determined by the interleave ways. If $\text{offset} \geq 0$, $\text{offset} < \text{DPALen}$, and the Valid bit is set, then the request hits within that decoder. If only one decoder hits, then the DPA is calculated by adding DPABase to the offset. If zero or multiple decoders hit, then an access error is returned.

After the request HPA is translated to DPA, the RPID and the DPA are used to perform the Dynamic Capacity access check, as described in [Section 7.7.2.5](#), and to access the GFD snoop filter. The design of the snoop filter is beyond the scope of this specification.

When the snoop filter needs to issue a back-invalidate to a host/peer, the DPA is translated to an HPA by performing the HPA-to-DPA steps in reverse. The RPID is used to access the GDT to select the decoders for the requester, which may be the host itself or one of its devices that performs Direct P2P. The GFD shall then compare, in parallel, the DPA against all selected decoders to determine whether the back-invalidate hits any decoder's DPA range.

This is accomplished by first calculating DPA offset = DPA – DPABase, and then testing whether offset ≥ 0 , offset $< \text{DPALen}$, and the decoder is valid. If only one decoder hits, then the HPA is calculated by inserting the interleaving bits into the offset and then adding it to HPABase. When inserting the interleaving bits, the LSB is determined by interleave granularity, the number of bits is determined by the interleaving ways, and the value of the bits is determined by the way within the interleave set. If zero or multiple decoders hit, then an internal snoop filter error has occurred which will be handled as defined in a future specification update.

After the HPA is calculated, a BISnp with the GFD's SPID and HPA is issued to the Edge Port containing the FAST decoder of the host/peer that owns this HDM-DB Region, using the PID stored in the snoop filter as the DPID. The FAST decoder then optionally checks whether the HPA is located within the FAST decoder's Fabric Address space. The DPID and SPID are then removed, and the BISnp is then issued to the host/peer in HBR format.

IMPLEMENTATION NOTE

It is recommended that a PBR switch size structures to support the typical to full scale of a PBR fabric.

It is recommended that the FAST have 4K to 16K entries.

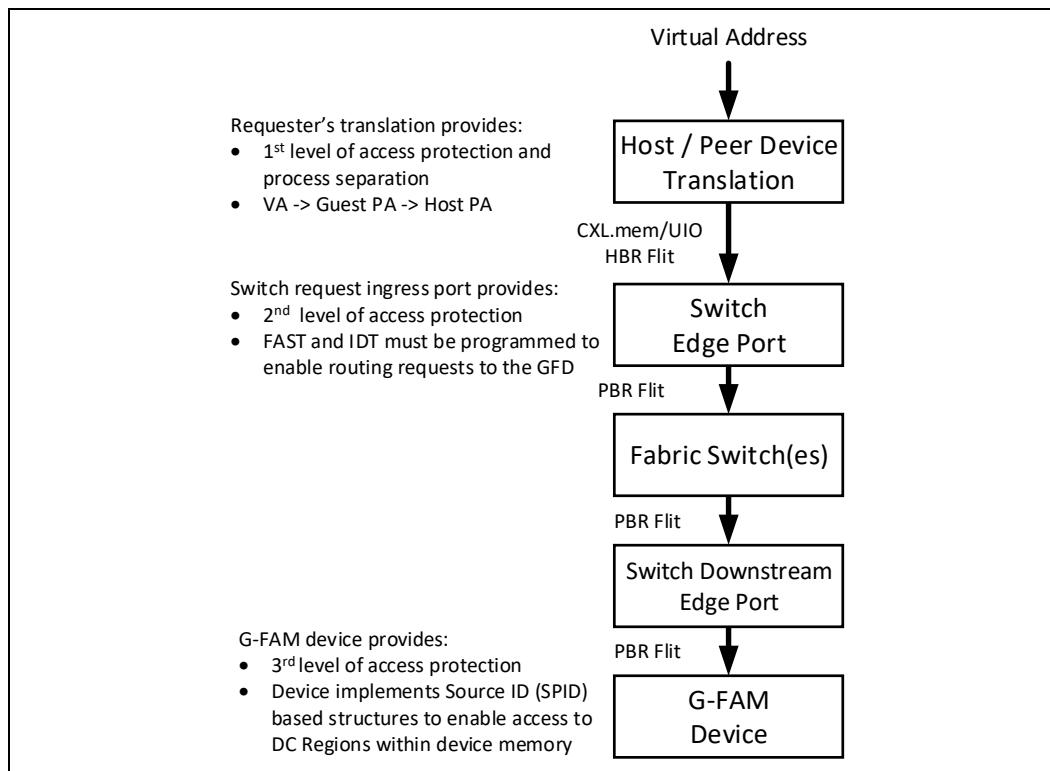
It is recommended that the IDT have 4K to 16K entries to support a sufficient number of interleave groups and interleave ways to cover all GFDs in a system.

7.7.2.5

G-FAM Access Protection

G-FAM access protection is available at three levels of the hierarchy (see [Figure 7-32](#)):

- The first level of protection is through the host's (or peer device's) page tables. This fine-grained protection is used to restrict the Fabric Address space that is accessible by each process to a subset of that which is accessible by the host/peer.
- The second level of protection is described in the GAE in the form of the Global Memory Mapping Vector (GMV), described in [Section 7.7.2.6](#).
- The third level of protection is at the target GFD itself and is fine grained. This section describes this third level of GFD protection.

Figure 7-32. Memory Access Protection Levels

The GFD's DPA space is divided into one or more Device Media Partitions (DMPs). Each DMP is defined by a base address within DPA space (DMPBase), a length (DMPLength), and a block size (DMPBlockSize). DMPBase and DMPLength must be a multiple of 256 MB, while DMPBlockSize must be a power-of-two size in bytes. The DMPBlockSize values that are supported by a device are device dependent and are defined in the GFD Supported Block Size Mask register. Each GFD decoder targets the DPA range of a DC Region within a single DMP (i.e., must not straddle DMP boundaries). The DC Region's block size is determined by the associated DMP's block size. The number of DMPs is device-implementation dependent. Unique DMPs are typically used for different media types (e.g., DRAM, NVM, etc.) and to provide sufficient DC block sizes to meet customer needs.

The GFD Dynamic Capacity protection mechanism is shown in [Figure 7-33](#). To support scaling to 4096 CXL requesters, the GFD DC protection mechanism uses a concept called Memory Groups. A Memory Group is a set of DMP blocks that can be accessed by the same set of requesters. The maximum number of Memory Groups (NG) that are supported by a GFD is implementation dependent. Each DMP block is assigned a Memory Group ID (GrpID), using a set of Memory Group Tables (MGTs). There is one MGT per DMP. Each MGT has one entry per DMP block within the DMP, with entry 0 in the MGT corresponding to Block 0 within the DMP. The depth of each MGT is implementation dependent. DPA is decoded to determine within which DMP a request falls, and then that DMP's MGT is used to determine the GrpID. The GrpID width is $X = \text{ceiling}(\log_2(NG))$ bits. For example, a device with 33 to 64 groups would require 6-bit GrpIDs.

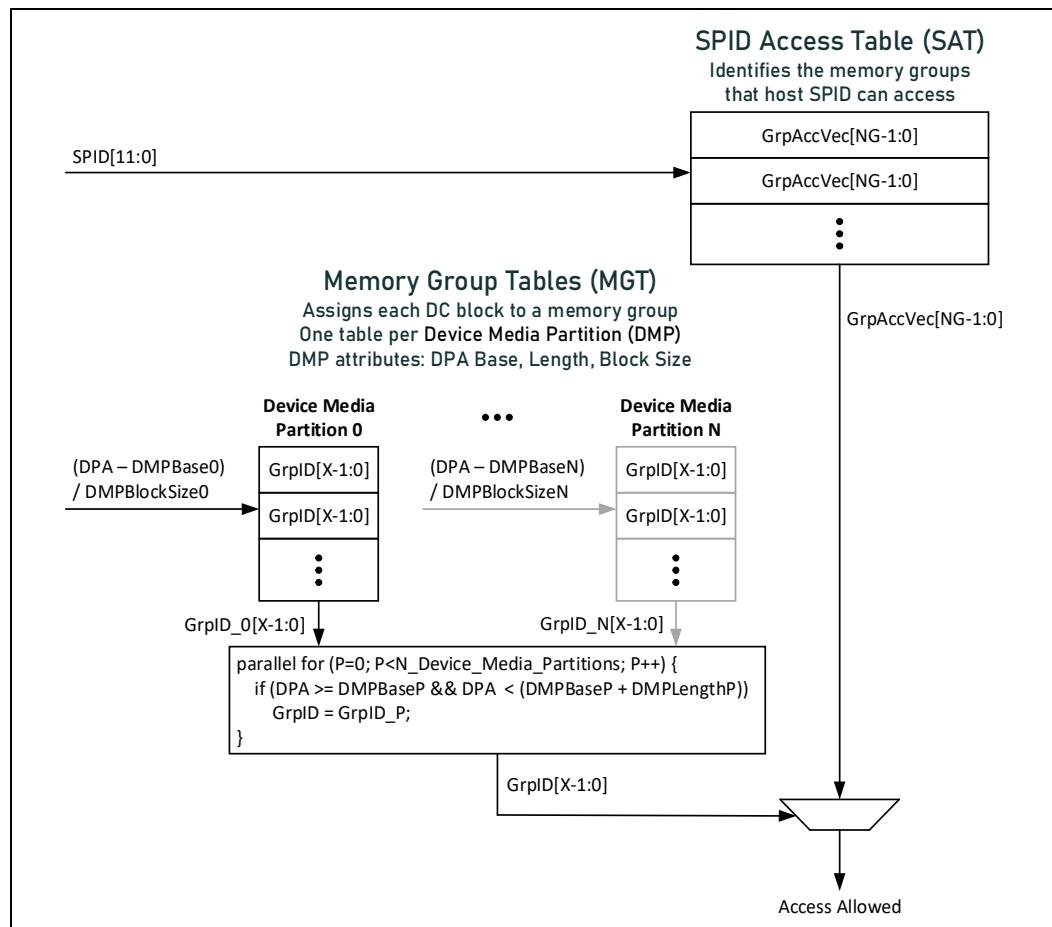
In parallel with determining the GrpID for a request, the Request SPID is used to index the SPID Access Table (SAT) to produce a vector that identifies which Memory Groups the SPID is allowed to access (GrpAccVec). After the GrpID for a request is determined, the GrpID is used to select a GrpAccVec bit to determine whether access is allowed.

IMPLEMENTATION NOTE

To support allocation of GFD capacity to hosts in sufficiently small percentages of the GFD, it is recommended that devices implement a minimum of 1K entries per MGT. Implementations may choose to use a separate RAM per MGT, or may use a single partitioned RAM for all MGTs.

To support a sufficient number of memory ranges with different host access lists, it is recommended that devices implement a minimum of 64 Memory Groups.

Figure 7-33. GFD Dynamic Capacity Access Protections



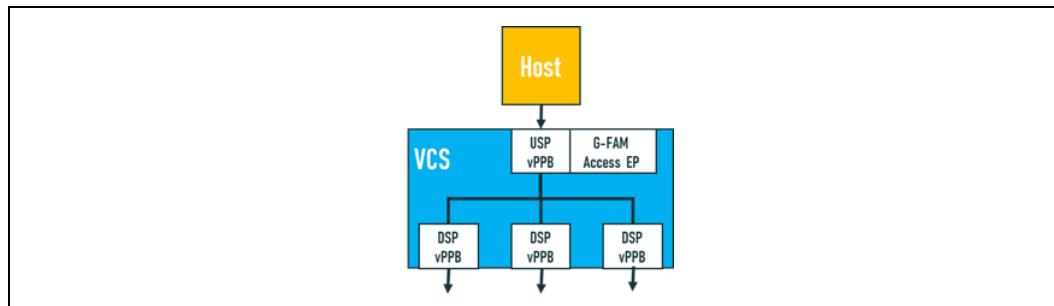
7.7.2.6 Global Memory Access Endpoint

Access to G-FAM/GIM resources and configuration of the FAST through a PBR fabric edge switch is facilitated by a Global Memory Access Endpoint (GAE) which is a Mailbox CCI that includes support for the Global Memory Access Endpoint Command set and the opcodes required to configure and enable FAST use, including **Get PID Access Vectors** and **Configure FAST**.

The GAE is presented to the host as a PCIe Endpoint with a Type 0 configuration space as defined in [Section 7.2.9](#).

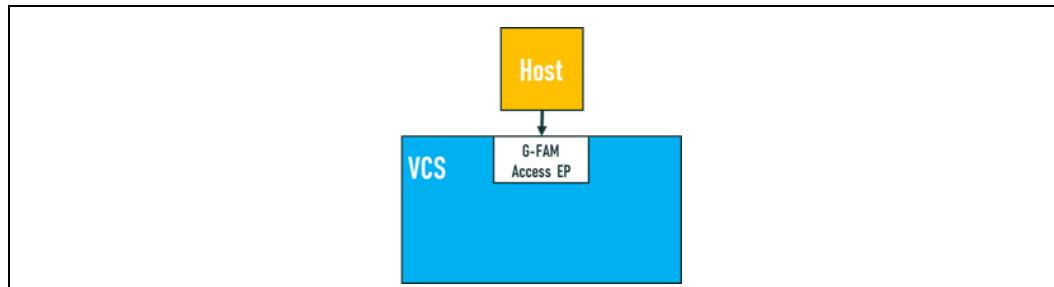
There are two configurations under which a host edge port USP will expose a GAE. The first configuration, illustrated in [Figure 7-34](#), provides LD-FAM and G-FAM/GIM resources to a host. In this configuration, the GAE Mailbox CCI is used to configure G-FAM/GIM access for the USP and any DSPs connected to EPs. It may also include support for opcodes necessary to manage the CXL switch capability providing LD-FAM resources.

Figure 7-34. PBR Fabric Providing LD-FAM and G-FAM Resources



The second configuration, illustrated in [Figure 7-35](#), only provides access to G-FAM/GIM resources. In this configuration, there is no CXL switch instantiated in the VCS and the GAE is the only PCIe function presented to the host.

Figure 7-35. PBR Fabric Providing Only G-FAM Resources



A GAE is also required in the vUSP of a Downstream ES VCS. This GAE is used for configuring that VCS, including configuring the FAST and LDST in the Edge DSPs and providing CDAT information, as described in [Section 7.7.12.4](#).

Each GAE maintains two access vectors, which are used to control whether the host has access to a particular PID:

- **Global Memory Mapping Vector (GMV):** 4k bitmask indicating which PIDs have been enabled for G-FAM or GIM access
- **VendPrefixL0 Target Vector (VTV):** 4k bitmask indicating which PIDs have been enabled for VendPrefixL0

7.7.2.7

Event Notifications from GFDs

GFDs do not maintain individual logs for every requester. Instead, events of interest are reported using the Enhanced Event Notifications defined in [Section 8.2.9.2.9](#) and [Section 8.2.9.2.10](#). These notifications are transported across the fabric using GAM VDMs, as defined in [Section 3.1.11.6](#).

For event notifications sent to a host, the GAM VDM's DPID is the PID of the host's GAE. When received by the GAE, the GAM VDM's 32B payload is written into the host's GAM Buffer. All GAM VDMs that are received by the GAE are logged into the same GAM Buffer, regardless of their SPID.

The GAM Buffer is a circular buffer in host memory that is configured for 32B entries. Its location in host memory is configured with the Set GAM Buffer request. The GAE writes received GAM VDM payloads into the buffer offset that is specified by the head index reported by the Get GAM Buffer request (see [Section 8.2.9.2.11](#)). As the host reads entries, the host increments the tail index using the Set GAM Buffer request (see [Section 8.2.9.2.12](#)). Head and tail indexes wrap to the beginning of the buffer when they increment beyond the buffer size.

The buffer is empty when the head index and tail index are equal. The buffer is full when the head index is immediately before the tail index. Old entries are not overwritten by the GAE until the host removes them from the buffer by incrementing the tail index. The GAE will report a buffer overflow condition if a GAM VDM is received when the buffer is full.

GAM VDMs are not forwarded to peer devices and are instead silently dropped by the peer's edge switch.

7.7.3

Global Integrated Memory (GIM)

A host domain may include multiple tiers of memory:

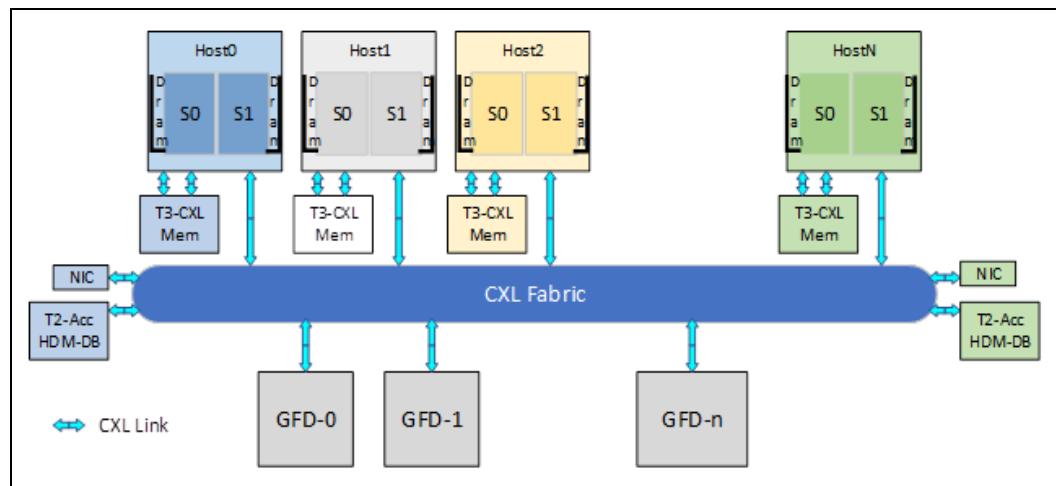
- Memory natively attached to a host (e.g., DDR, HBM, etc.)
- Device memory attached to a host CXL link
- Device memory attached to a host through CXL switches

All the memory tiers listed above are managed by a host operating system. CXL devices may be a Type 2 device or Type 3 device and may optionally support back-invalidate channels. A CXL Fabric may be composed of many host domains and G-FAM devices (GFD) as shown in [Figure 7-36](#). GFD is a scalable memory resource that is accessible by all hosts and peer devices within a CXL Fabric.

Each host domain may allow other host domains within the CXL Fabric to access locally managed memory at any tier. Global Integrated Memory (GIM) refers to the memory in remote host domains that is mapped into local host physical address space. Hosts and devices are allowed to initiate cross-domain accesses to GIM, utilizing Unordered I/O (UIO) transactions. CXL.mem or CXL.cache must not be used for GIM accesses.

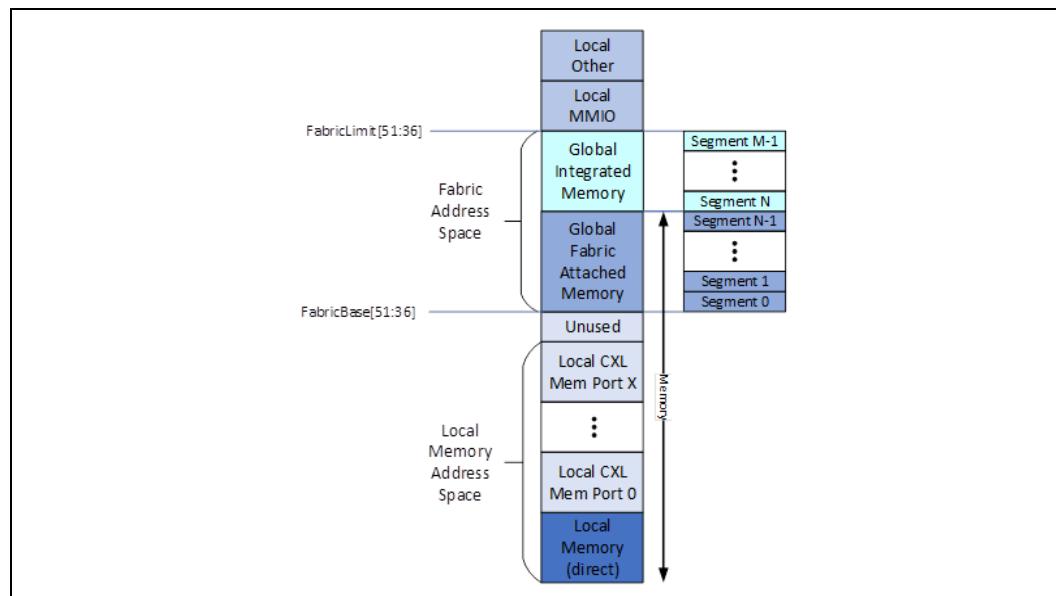
Cross-domain accesses are considered I/O coherent — data is coherent at the time of access. Remote domains may either mark this memory as uncacheable or manage caches with SW mechanisms.

GIM is primarily used for enabling remote DMA and messaging across domains. It is not intended for memory pooling or borrowing use cases.

Figure 7-36. CXL Fabric Example with Multiple Host Domains and Memory Types**7.7.3.1****Host GIM Physical Address View**

Hosts and devices may use proprietary decode mechanisms to identify the target DPID and may bypass address decoders in the switch ingress port. Hosts and devices are typically limited to access between homogeneous peers. See [Section 7.7.3.2](#) for ways by which hosts/devices can access Global Integrated Memory (GIM) without using the FAST decoders. This section covers the decode path that uses the FAST decoders.

Hosts that access GIM and rely on address decoders in the switch must map this range in the Fabric Address Space. Hosts that access GIM and GFD must include both ranges in the Fabric Address Space and must use a contiguous address range within the Host Physical Address (HPA) space as shown in [Figure 7-37](#).

Figure 7-37. Example Host Physical Address View with GFD and GIM

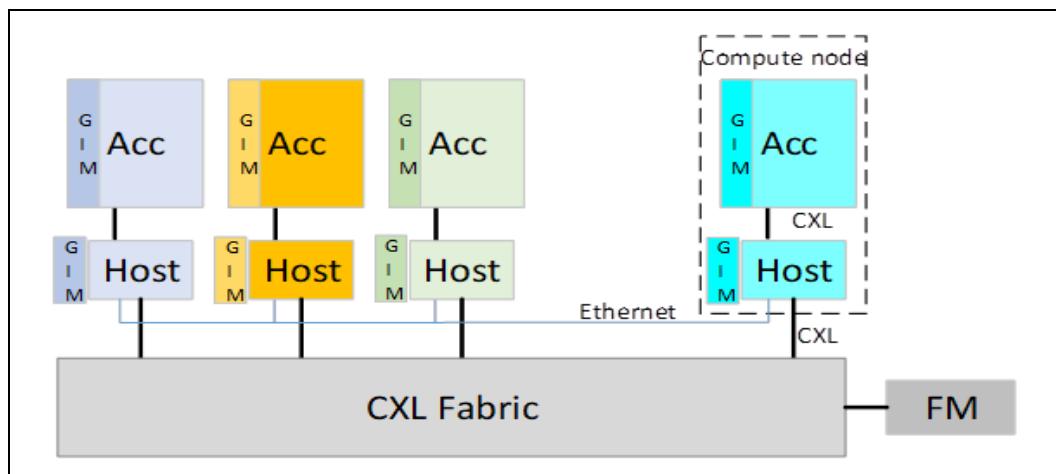
All accesses to GIM regions must only use UIO. It is recommended to map GIM as MMIO instead of a normal write back memory type to avoid potential deadlock. However, implementations may use proprietary methods to guarantee UIO use even when internally using a cacheable memory type. Thus, MMIO mapping of GIM is only a recommendation and not a requirement.

Host and device accesses to GFD and GIM are decoded using a common FAST decoder to determine the target's DPID.

7.7.3.2 Use Cases

ML and HPC applications are typically distributed across many compute nodes and need a scalable and efficient network for low-latency communication and synchronization. [Figure 7-38](#) is an example of a system with a compute node composed of a Host, an Accelerator, and a cluster of nodes connected through a CXL switch fabric. Each host may expose a region or all available memory to other compute nodes.

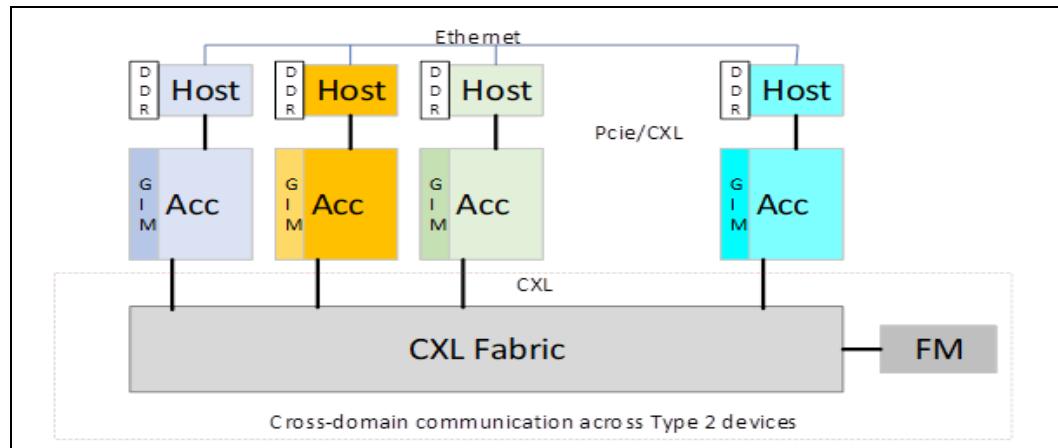
Figure 7-38. Example Multi-host CXL Cluster with Memory on Host and Device Exposed as GIM



A second example in [Figure 7-39](#) shows a CXL Fabric that connects all the accelerators. In this example, only the memory attached to the device is exposed to other devices as GIM. UIO allows flexible implementation options to enable RDMA semantics between devices. Software and security requirements are beyond the scope of this specification. GIM builds a framework for using the same set of capabilities for host-to-host communication, device-to-device communication, host-to-device communication, and device-to-host communication.

EVALUATION COPY

Figure 7-39. Example ML Cluster Supporting Cross-domain Access through GIM



7.7.3.3 Transaction Flows and Rules for GIM

The flow in [Figure 7-40](#) describes how a host can access GIM in another host, using the fabric address model described earlier in this chapter. While [Figure 7-40](#) uses host-to-host as the example, the same model works for host-to-device, device-to-device and device-to-host as well. A device that implements GIM as target is expected to have the required functionality that translates the combination of <Address: PID> in the incoming UIO TLP to a local memory address and to provide the required security on cross-domain accesses. This functionality can also use more information than just <Address:PID> from the TLP (e.g., PASID) for additional functionality/security. Designs can choose to reuse the GFD architecture for defining this translation/protection functionality or can implement a proprietary IOMMU-like logic. Details of this functionality are beyond the scope of this Specification.

Figure 7-40. GIM Access Flows Using FASTs

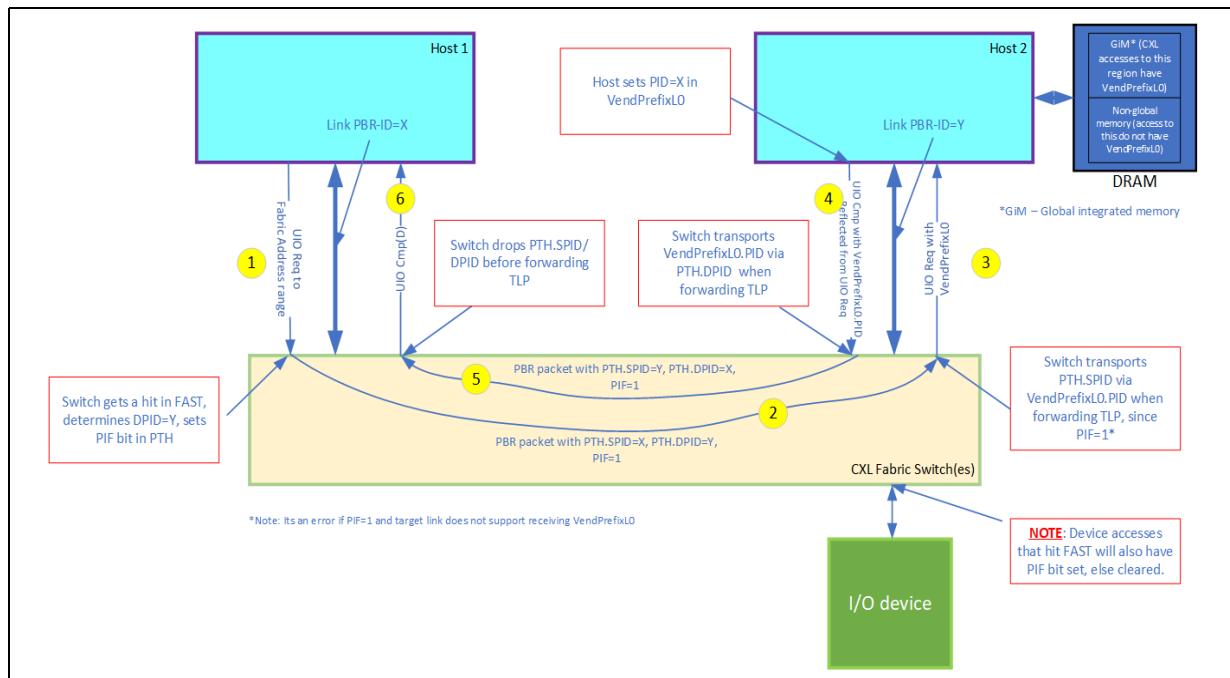
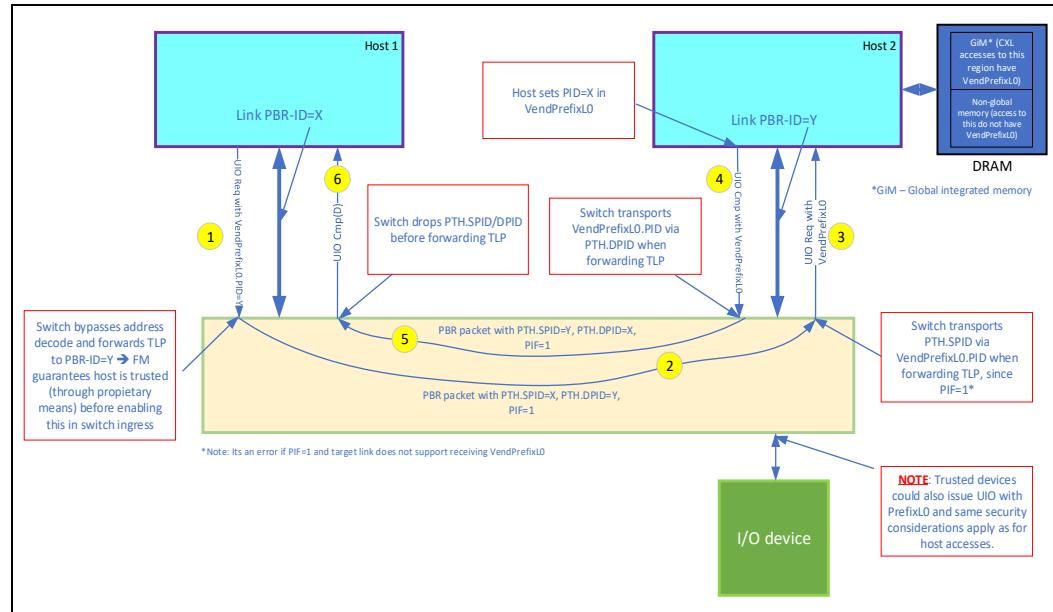


Figure 7-41. GIM Access Flows without FASTs



Although the flows described in [Figure 7-40](#) and [Figure 7-41](#) are self-explanatory, here are the key rules for PBR switches/Hosts/Devices that support the GIM flows:

- FM enables usage of VendPrefixL0 on non-PBR edge ports, using the FM API discussed in [Table 7-186](#). By default, VendPrefixL0 usage is disabled on edge ports.

The mechanism that the FM uses to determine on which ports to enable this functionality is beyond the scope of this specification.

7.7.3.3.1 GIM Rules for PBR Switch Ingress Port

- GIM flows are supported only via UIO transactions in this version of the specification. At this time, GIM flows are NOT supported via CXL.cachemem transactions or Non-UIO TLPs.
 - If switch ingress port receives a Non-UIO request with VendPrefixL0, it treats it as a UR.
- At the Non-PBR edge ingress port, for UIO request TLPs that do not have VendPrefixL0 and that are decoded via the FASTs, the switch sets the PTH.PIF bit when forwarding the request into the PBR fabric.
 - For UIO request TLPs that are not decoded via the FASTs, this bit is cleared when forwarded to the PBR fabric.
- At the Non-PBR edge ingress port, if the port is enabled for Ingress Request VendPrefixL0 usage and UIO request TLP has VendPrefixL0 and VendPrefixL0.PID matches one of the allowed PIDs in VTV (see [Section 7.7.2.6](#)), the switch bypasses all decode, sets PTH.DPID=VendPrefixL0.PID, PTH.SPID=Ingress Port PID, and PTH.PIF=1 when forwarding the request to the PBR fabric.
 - If a UIO request TLP is received with VendPrefixL0 but the port is not enabled for Ingress Request VendPrefixL0 usage or if the PID in the prefix does not match any of the allowed PIDs in VTV, the switch treats the request as a UR.
- At the Non-PBR edge ingress port, for UIO completion TLPs, the switch forwards the received VendPrefixL0.PID on PTH.DPID when forwarding the packet to the PBR fabric, if Ingress Completion VendPrefixL0 usage is enabled on the port (see

[Section 7.7.15.5](#)) and VendPrefixL0.PID matches one of the allowed PIDs in VTV (see [Section 7.7.2.6](#)). PTH.SPID on the completion TLP is set to the PID of the ingress port.

- if a UIO completion TLP is received on a Non-PBR edge ingress port when Ingress Completion VendPrefixL0 usage is disabled on the port or if the PID in the prefix does not match any of the allowed PIDs in VTV, the switch must drop the packet and treat it as an Unexpected Completion.
- Switch sets the PIF bit whenever it successfully forwards the received completion TLP to the PBR fabric.

7.7.3.3.2 GIM Rules for PBR Switch Egress Port

- At the Non-PBR edge egress port, for UIO request TLPs with the PTH.PIF bit set, the switch forwards the PTH.SPID field of the request TLP on the VendPrefixL0.PID field if the egress port is enabled for Egress Request VendPrefixL0 usage.
 - If the PTH.PIF bit is set but the egress port is not enabled for Egress Request VendPrefixL0 usage, the switch should treat the request as a UR.
 - If the PTH.PIF bit is cleared in the UIO request TLP, the request TLP is forwarded to the egress link without VendPrefixL0, regardless of whether the port is enabled for Egress Request VendPrefixL0 usage.
- At the Non-PBR edge egress port, the switch does not send VendPrefixL0 on completion TLPs.
- Switch forwards the PTH.PIF bit as-is on edge PBR links

7.7.3.3.3 GIM Rules for Host/Devices

- Host/Devices that support VendPrefixL0 semantics and receive a UIO Request TLP with VendPrefixL0 must return the received PID value in the associated completion's VendPrefixL0.
- Host/Devices must always return a value of 0 for Completer ID in the UIO completions.

7.7.3.3.4 Other GIM Rules

- VendPrefixL0 must never be sent on edge PBR links, such as the links connecting to a GFD
- GFD must ignore the PTH.PIF bit on TLPs that the GFD receives
- GFD is permitted to set the PTH.PIF bit on CXL.io request TLPs that the GFD sources and always sets this bit on CXL.io completion TLPs that the GFD sources

Note:

If setting the PTH.PIF bit on request TLPs, the GFD must do so only if it is sure that the ultimate destination (e.g., GIM) needs to be aware of the PID of the source agent that is generating the request (such as for functional/security reasons); otherwise, the GFD should not set the bit.

7.7.3.4 Restrictions with Host-to-Host UIO Usages

Host-to-Host UIO usages can result in deadlock when mixed with UIO traffic going to the host that can route back in the host. To avoid such deadlocks:

- Systems that support Host-to-Host UIO must use a separate VC for Host-to-Host UIO traffic vs. remainder of UIO, on host edge links.

(OR)

- Minimally avoid usages that can cause loopback traffic, either in the host or in switches. Generically, this restriction could mean that UIO accesses do not target MMIO space.

A detailed analysis of restrictions that are needed to make a specific system configuration to work with Host-to-Host UIO enabled is beyond the scope of this specification.

A future ECN may be considered that allows for more deadlock avoidance options beyond the two listed above.

7.7.4

Non-GIM Usages with VendPrefixL0

When Hosts/Devices initiate UIO requests with VendPrefixL0, address decoding is bypassed in the Switch ingress port. This allows for proprietary implementations in which the address/data information in the TLP can potentially be vendor-defined. Such usages are beyond the scope of this specification; however, GIM-related rules enumerated in [Section 7.7.3.3](#) allow such implementations as well.

7.7.5

HBR and PBR Switch Configurations

CXL supports two types of switches: HBR (Hierarchy Based Routing) and PBR (Port Based Routing). "HBR" is the shorthand name for the CXL switches introduced in the CXL 2.0 specification and enhanced in subsequent CXL ECNs and specifications. In this section, the interaction between the two will be discussed.

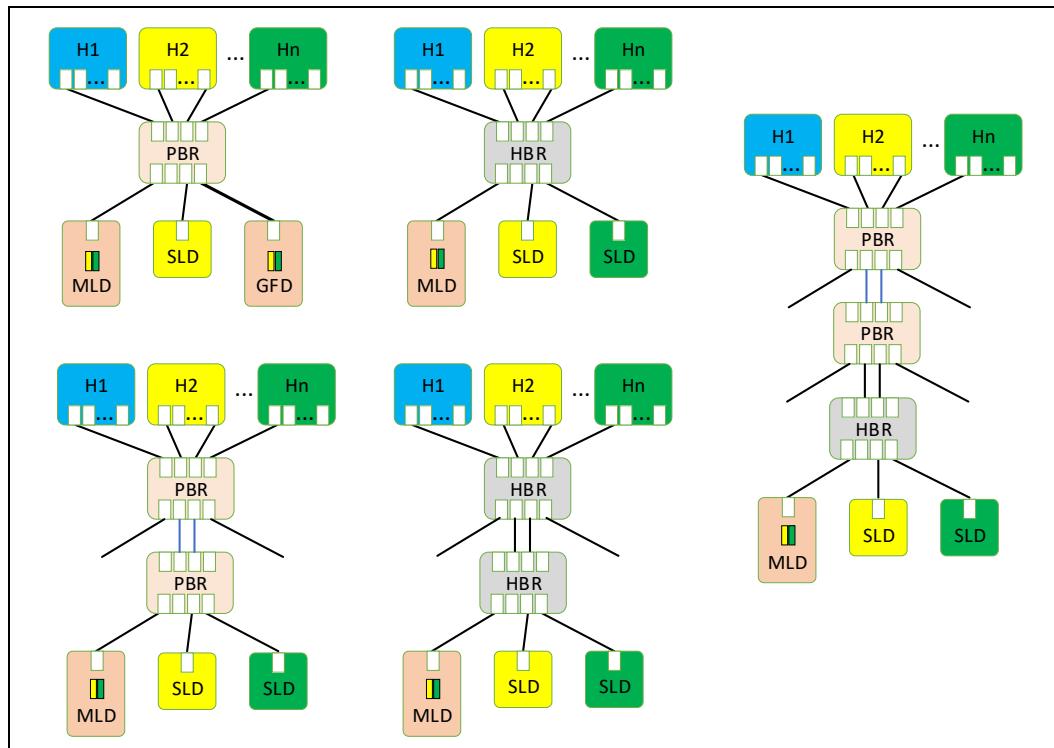
A variety of HBR/PBR switch combinations are supported. The basic rules are as follows:

- Host RP must be connected to an HBR USP, PBR USP, or a non-GFD
- Non-GFD must be connected to an HBR DSP, a PBR DSP, or a Host RP
- PBR USP may be connected only to a host RP; connecting it to an HBR DSP is not supported
- HBR USP may be connected to a host RP, a PBR DSP, or an HBR DSP
- GFD may be connected only to a PBR DSP
- PBR FPort may be connected only to a PBR FPort of a different PBR switch

[Figure 7-42](#) illustrates some example supported switch configurations, but should not be considered a complete list.

EVALUATION COPY

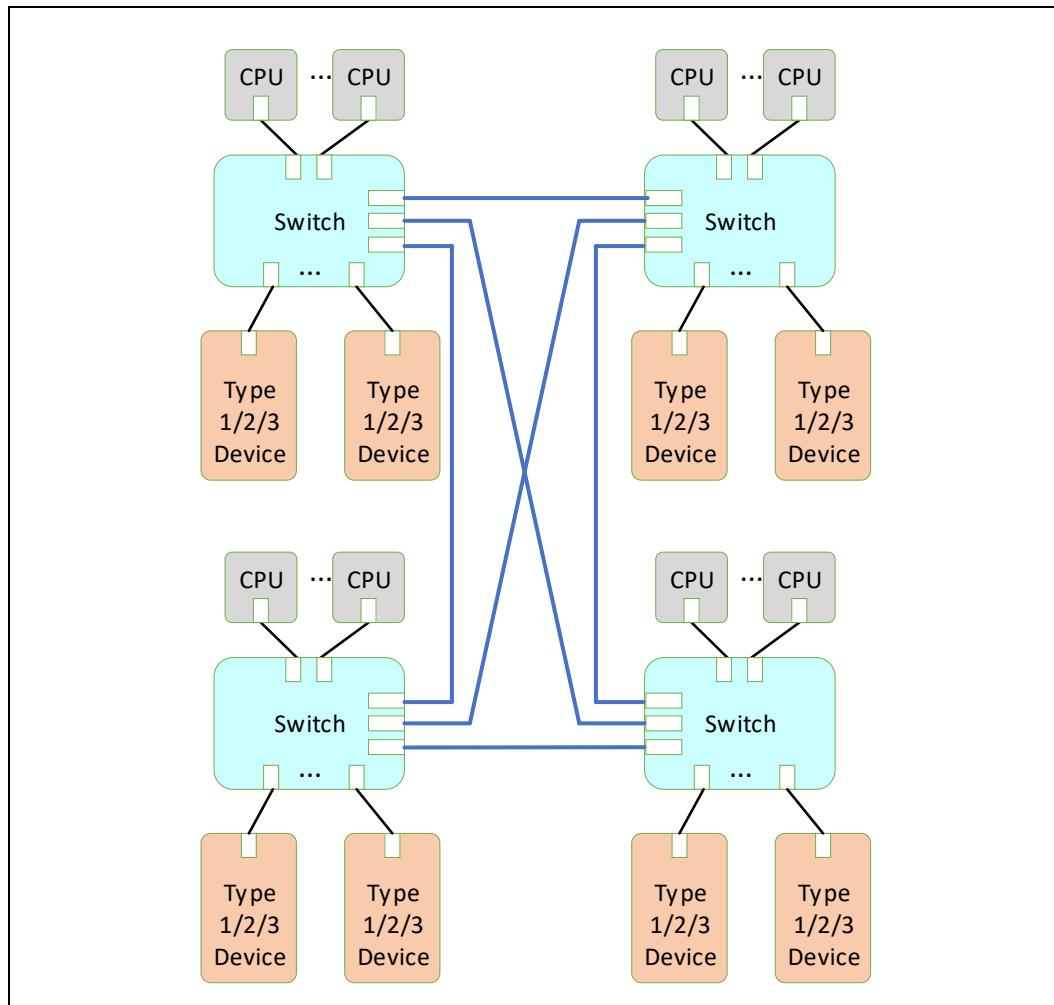
Figure 7-42. Example Supported Switch Configurations



CXL fabric topology is non-prescriptive when using PBR switches. There is no predefined list of supported topologies. PID-based routing combined with flexible routing tables enables a high degree of freedom in choosing a topology. The PBR portion of the fabric may freely use any topology for which deadlock-free routing can be found.

To name a few examples, a PBR fabric might implement a simple PCIe-like tree topology, more-complex tree topologies such as fat tree (aka folded Clos), or non-tree topologies such as mesh, ring, star, linear, butterfly, or HyperX, as well as hybrids and multi-dimensional variants of these topologies.

[Figure 7-43](#) illustrates an example of fully connected mesh topology (aka 1-dimensional HyperX). It has the notable ability to connect a relatively large number of components while still limiting the number of switch traversals. A direct link exists between each pair of switches, so it is possible for the FM to set up routing tables such that all components connected to the same switch can reach one another with a single switch traversal, and all components connected to different switches can reach one another with two switch traversals.

Figure 7-43. Example PBR Mesh Topology

7.7.5.1

PBR Forwarding Dependencies, Loops, and Deadlocks

When messages are forwarded through PBR switches from one Fabric Port to another, a dependency is created — acceptance of arriving messages into one PBR Fabric Port is conditional upon the ability to transmit messages out of another PBR Fabric Port. Other arriving traffic commingled on the same inbound link is also affected by the dependency. Thus, traffic waiting to be forwarded can block traffic that needs to exit the PBR portion of the fabric via a USP or DSP of the PBR switch.

Some topologies, such as PCIe tree or fat tree, are inherently free of loops. Thus, the resulting Fabric Port-forwarding dependencies are inherently non-circular. However, in topologies that contain loops, dependencies can form a closed loop, thereby resulting in a deadlock.

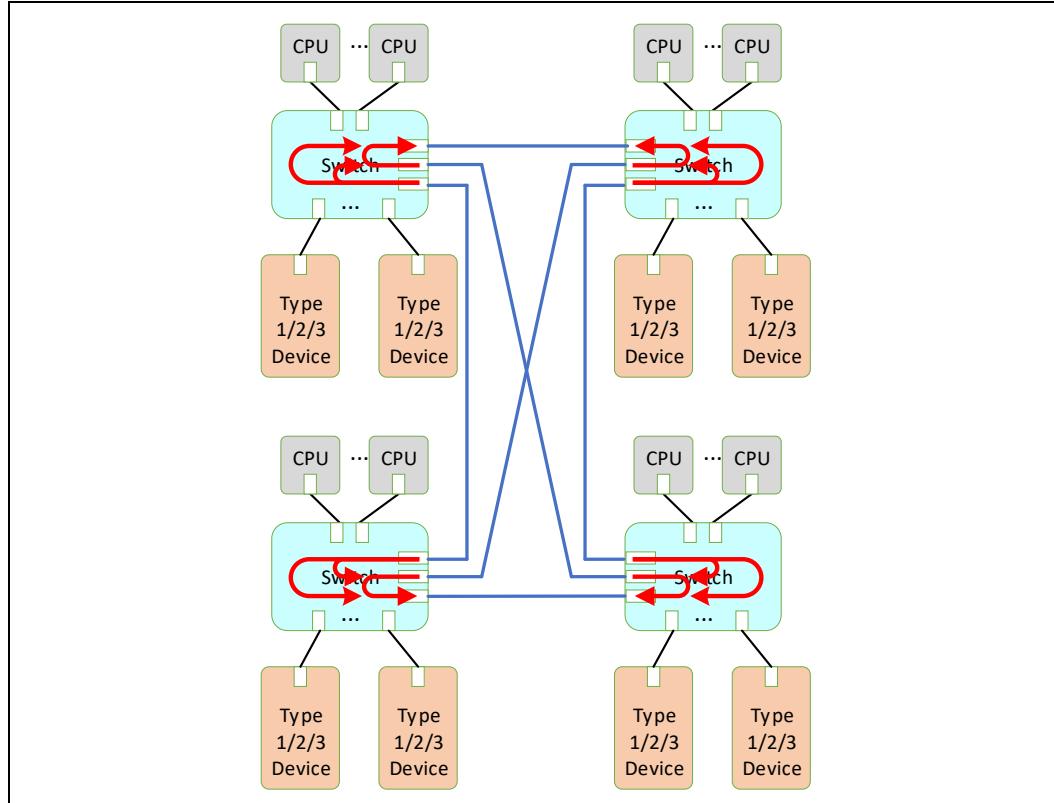
The routing table programming in the PBR switches, performed by the FM, must take potential deadlock into account. The dependencies must not be allowed to form a closed loop.

EVALUATION COPY

Switching

This can be illustrated using the mesh topology presented in [Figure 7-44](#).

Figure 7-44. Example Routing Scheme for a Mesh Topology



One simplistic approach for the mesh topology would be to support only minimal routes. Messages traverse at most one inter-switch PBR link en route from any source host or device to any destination host or device. This simplistic solution is deadlock-free because no message forwarding occurs between PBR Fabric Ports of any switch, and thus there are no forwarding dependencies created from which loops may form. The single route choice, however, limits bandwidth.

[Figure 7-44](#) illustrates a more-sophisticated routing scheme applied to the same mesh topology as [Figure 7-43](#). Each PBR switch is programmed to support three forwarding paths out of the 6 possible pairings. The arrows show permitted forwarding between Fabric Ports. For example, a message traveling from the lower-left switch to the upper-right switch has two route choices:

- Via the direct link
- Indirectly via the upper-left switch

Note that the message cannot travel via the lower-right switch because that switch has no forwarding arrow shown between those Fabric Ports.

The forwarding arrows do not form closed loops; thus, there are no circular dependencies that could lead to deadlock.

This approach to mesh routing (i.e., restricting the choice of intermediate nodes to avoid circular dependencies) can also be applied to larger 1D-HyperX topologies. For a fully connected mesh that contains N switches, there are N-2 potential intermediate

switches to consider for possible indirect routes between any pair of switches. However, this deadlock-avoidance restriction limits the usable intermediate switch choices to one-half of that number ($(N-2)/2$), rounding down if N is odd.

Multi-dimensional HyperX topologies can be routed deadlock-free by using this technique within each dimension, and implementing dimension-ordered routing.

Although this section covers some cases for circular dependency avoidance, fully architected deadlock dependency avoidance with topologies that contain fabric loops is beyond the scope of this specification.

7.7.6

7.7.6.1

PBR Switching Details

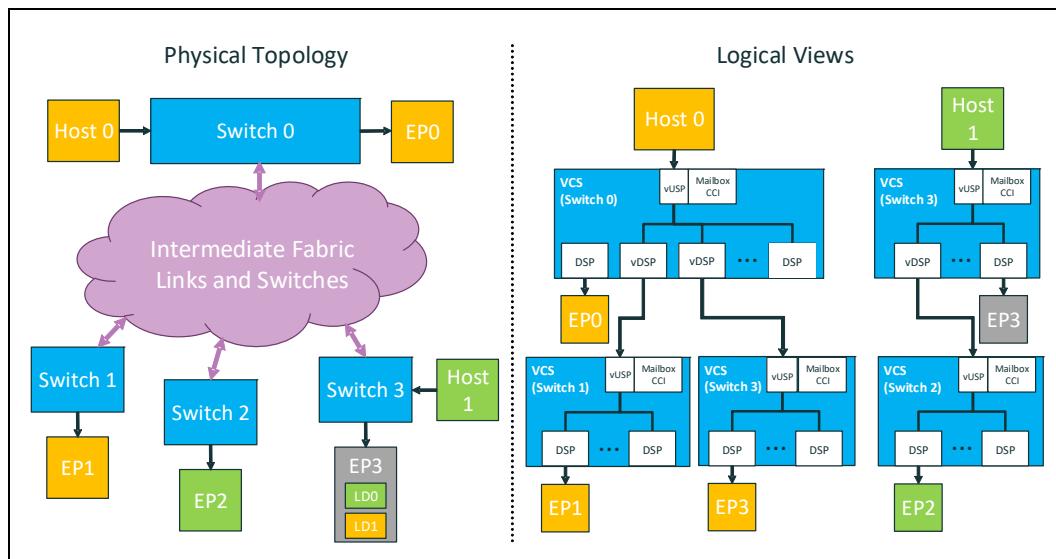
Virtual Hierarchies Spanning a Fabric

Hosts connected to CXL Fabrics (composed of PBR switches) do not require special, fabric-specific discovery mechanisms. The fabric complexities are abstracted, and the host is presented with a simple switching topology that is compliant with PCIe Base Specification. All intermediate Fabric switches are obscured from host view. At most, two layers of Edge Switches (ESs) are presented:

- Host ES: The host discovers a single switch representative of the edge to which it is connected. Any EPs also physically connected to this PBR switch and bound to the host's VH are seen as being directly connected to PPBs within the VCS.
- Downstream ES: As desired, the FM may establish binding connections between the Host ES VCS and one or more remote PBR switches within the Fabric. When such a binding connection is established, the remote switch presents a VCS that is connected to one of the Host ES vPPBs. The Host discovers a single link between a virtualized DSP (vDSP) in the Host ES and a virtualized USP (vUSP) in the Downstream ES, regardless of the number of intermediate fabric switches, if any. The link state is virtualized by the Host ES and is representative of the routing path between the two ESs; if any intermediate ISLs go down, the Host ES will report a surprise Link Down error on the corresponding vPPB.
- If an HBR switch is connected to a PBR DSP, that HBR switch and any HBR switches below it will be visible to the host. HBR switches are not Fabric switches.

A PBR switch's operation as a "Host ES" or a "Downstream ES" per the above descriptions is relative to each host's VH. A PBR switch may simultaneously support Host ES Ports and Downstream ES Ports for different VHs. ISLs within the Fabric are capable of carrying bidirectional traffic for more than one VH at the same time. Edge DSPs support PCIe devices, SLDs, MLDs, GFDs, PCIe switches, and CXL HBR switches.

A Mailbox CCI is required in the vUSP of a Downstream ES VCS for management purposes.

Figure 7-45. Physical Topology and Logical View

7.7.6.2 PBR Message Routing across the Fabric

PBR switches can support both static and dynamic routing for each DPID, as determined by message class.

With static routing, messages of a given message class use a single fixed path between source and destination Edge Ports. Messages that use a vDSP/vUSP binding (see [Section 7.7.6.4](#)) always use static routing as well, though the vUSP as a source or destination is always associated with an FPort instead of an Edge Port.

With dynamic routing, messages of a given message class can use different paths between source and destination Edge Ports, dynamically determined by factors such as congestion avoidance, algorithms to distribute traffic across multiple links, or changes with link connectivity. Each DPID supports static routing for those message classes that require it, and it can support either static or dynamic routing for the other message classes.

Dynamic routing is generally preferred when suitable, but in certain cases static routing must be used to ensure in-order delivery of messages as required by ordering rules. Due to its ability to distribute traffic across multiple links, dynamic routing is especially preferred for messages that carry payload data, as indicated in [Table 7-84](#).

Somewhat orthogonal to dynamic vs. static routing, PBR switches support hierarchical and edge-to-edge decoding and routing. With hierarchical routing, a message is decoded and routed within each ES using HBR mechanisms and statically routed between ESs, using vDSP/vUSP bindings. With edge-to-edge routing, a message is routed from a source Edge Port to a destination Edge Port, using a DPID determined at the source Edge Port or GFD. Edge-to-edge routing uses either dynamic or static routing, as determined by the message class.

[Table 7-84](#) summarizes the type of PBR decoding and routing used, by message class.

Table 7-84. PBR Fabric Decoding and Routing, by Message Class

Message Class ** Payload Data		Ordering Rules	Preferred Routing ¹	Decoding and Routing Mechanism
CXL.cache	D2H Req		Dynamic	Edge-to-edge routing using the Cache ID lookups or vPPB bindings
	H2D Rsp	I11a: Snoop (H2D Req) push GO (H2D Rsp)	Static	
	H2D DH **		Dynamic	
	H2D Req	I11a: Snoop (H2D Req) push GO (H2D Rsp)	Static	
	D2H Rsp		Dynamic	
	D2H DH **		Dynamic	
CXL.mem	M2S Req	G8a (HDM-D to Type 2): MemRd*/MemInv* push Mem*Fwd	HDM-H: Dynamic HDM-D: Static HDM-DB: Dynamic	LD-FAM: Edge-to-edge routing if using LDST ² Hierarchical routing if using HDM Decoder G-FAM: edge-to-edge routing using FAST
	M2S RWD **	-	Dynamic	LD-FAM: Edge-to-edge routing if using LDST ² Hierarchical routing if using HDM Decoder G-FAM: Edge-to-edge routing using FAST
	S2M NDR	E6a: BI-ConflictAck pushes Cmp*	Static	Edge-to-edge routing using vPPB bindings or BI-ID lookups
	S2M DRS **	-	Dynamic	
	S2M BISnp	-	Dynamic	
	M2S BIRsp	-	Dynamic	
CXL.io	All CXL.io TLPs ** except next row	PCIe (many)	Static	Hierarchical decoding within each ES vDSP/vUSP between Host ES and each Downstream ES
	UIO Direct P2P to HDM TLPs **	-	Dynamic	Edge-to-edge routing using FAST or LDST decoder

1. When dynamic routing is preferred, static routing is still permitted.

2. LDST decoders do not support HDM-D.

The Ordering Rules column primarily covers a few special cases with CXL.cachemem messages in which the fabric is required to enforce ordering within a single message class or between two message classes. The alphanumeric identifier refers to ordering summary table entries in [Table 3-56](#) and [Table 3-57](#).

With LD-FAM, host software may use either HDM Decoders or LDST decoders, though LDST decoders do not support HDM-D. Host software implemented solely against the CXL 2.0 Specification comprehends only HDM Decoders, and such host software may continue to use them with PBR Fabrics. Newer host software that comprehends and uses LDST decoders can benefit from edge-to-edge routing, which uses dynamic routing for suitable message classes.

For CXL.io TLPs, the PTH.Hie (hierarchical) bit determines when intermediate PBR switches must use static routing. When the PTH.Hie bit is 1, intermediate PBR switches shall use static routing for the TLP; otherwise, such switches are permitted to use dynamic routing for the TLP. When a PTH is pre-pended to a TLP, the Hie bit shall be 1 if the TLP is a vDSP/vUSP message; otherwise, the Hie bit shall be 0.

7.7.6.3 PBR Message Routing within a Single PBR Switch

A message received or converted to PBR format at a PBR switch ingress port is routed to one of the switch's egress ports, as determined by the ingress port's DPID Routing Table (DRT) and its associated Routing Group Table (RGT). Their structures are described in detail in [Section 7.7.13.10](#) and [Section 7.7.13.12](#), respectively, and this section provides a high-level summary.

A DRT has 4096 entries and is indexed by a DPID. Each DRT entry contains a 2-bit entry type field that indicates whether the entry is valid, and whether the entry contains a single physical port number or an RGT index.

DRT entries that contain an RGT index are required when multiple egress ports need to be specified for use with dynamic routing. An RGT is a power-of-2-sized table with up to 256 entries. Each RGT entry contains an ordered list of up to eight physical port numbers, along with two 3-bit fields that indicate how many in the list are valid and how many of those are primary vs. secondary. This allows one or more primary and zero or more secondary egress ports to be listed. Cases that require static routing must always use the first list entry. The RGT entry also contains a 3-bit dynamic routing mode and 3-bit mix setting. The distinction between primary vs. secondary varies by dynamic routing mode and mix setting.

In routing modes that utilize the mix setting, its value determines the mix of the primary and secondary egress port group usage, assuming that one or more secondary egress ports are specified. Using the mix setting supports egress port selection based on known bandwidth differences that exist elsewhere in the fabric or based on preferred vs. overflow routing paths. Secondary egress ports should be specified only when there are significant differences with primary egress ports; otherwise, all suitable egress ports should be specified as primary. When no secondary egress ports have been specified, the mix setting shall be ignored.

Mix Setting	% Primary	% Secondary
0	87.5	12.5
1	75	25
2	62.5	37.5
3	50	50
4	37.5	62.5
5	25	75
6	12.5	87.5
7	Preferred	Overflow

Mix setting 7 is intended for use in cases where primary and secondary egress port groups represent preferred and overflow ports, respectively. Mix setting 7 mandates the choice of a primary (preferred) path route whenever flow-control conditions and link state permit.

The term candidate egress port refers to a port that is present in the appropriate RGT entry, where the message can be queued or internally routed immediately. The egress port need not have link credits to send the packet immediately. An implementation may optionally base part of the candidate selection on the egress port state (e.g., link-up or containment states).

The mix dynamic routing mode descriptions that follow describe routing outcomes in terms of probability, consistent with a weighted (pseudo) random implementation. Random selection has the advantage that each routing decision is stateless and independent of one another, and it has high immunity to hot-route problems that might

otherwise arise from repetitive patterns in packet arrivals. The specific random routing implementation is not prescribed. Implementations that achieve the specified mix by deterministic means, such as by weighted round-robin, are permitted.

The architected dynamic routing modes include the optional modes listed in [Table 7-85](#).

Table 7-85. Optional Architected Dynamic Routing Modes

Mode	Description
Mix with Random	<p>The candidate list is first narrowed to select either the primary or the secondary group based on the configured mix. A random selection is then made within that group. A message class shall stall when the selected subset is empty due to flow-control conditions.</p> <p>The FM may choose to select this mode (if supported) as an alternative to Mix with Congestion Avoidance if the latter is not supported.</p>
Mix with Congestion Avoidance	<p>The candidate list is first narrowed to select either the primary or the secondary group based on the configured mix. A local congestion-avoiding selection is then made within that group. A message class shall stall when the selected subset is empty due to flow-control conditions. Congestion-avoiding candidate selection is based on vendor-specific congestion metrics, favoring the selection of less-congested egress ports. For example, the congestion metric might be a measure of egress port backlog, considering all queued traffic for that egress port across the entire switch.</p> <p>The FM may choose to select this mode (if supported) when Advanced Congestion Avoidance mode is inappropriate or not supported, or if fixed-traffic ratio apportionment or preferred/overflow behavior is needed.</p>
Advanced Congestion Avoidance	<p>A congestion-avoiding selection is made considering both primary and secondary candidate egress ports, ignoring the mix setting value. Egress ports with the minimal remaining hop count should be specified as primary; any suitable egress ports that have higher remaining hop counts should be specified as secondary. Candidate selection is based on vendor-specific metrics, favoring less-congested egress ports in general, and especially avoiding secondary candidates that are already heavily scheduled with primary traffic, regardless of the target DPID.</p> <p>An example congestion metric might be backlog-based, but with different weightings for primary vs. secondary backlogs. Congestion metric values for primary backlogs should be higher than secondary backlogs when assessing the congestion level of a secondary candidate egress port. This discourages the use of secondary candidate ports that have a high primary backlog. In congestion metrics, messages that are queued or internally routed via the physical port number in a DRT or via dynamic routing modes other than Advanced Congestion Avoidance should be considered primary backlog.</p> <p>The FM may choose to select this mode (if supported) for routing egress ports that carry commingled minimal and non-minimal traffic.</p>

PBR switches that implement RGTs shall support at least one of the three architected dynamic routing modes (those listed in [Table 7-85](#)) within each RGT.

DRT entries that contain a single physical port instead of an RGT index are useful when there is only one reasonable egress port choice (e.g., routing to an Edge Port). This avoids an RGT look-up and additional processing to determine which egress port to use. This may also help reduce the number of entries that need to be implemented in the associated RGT.

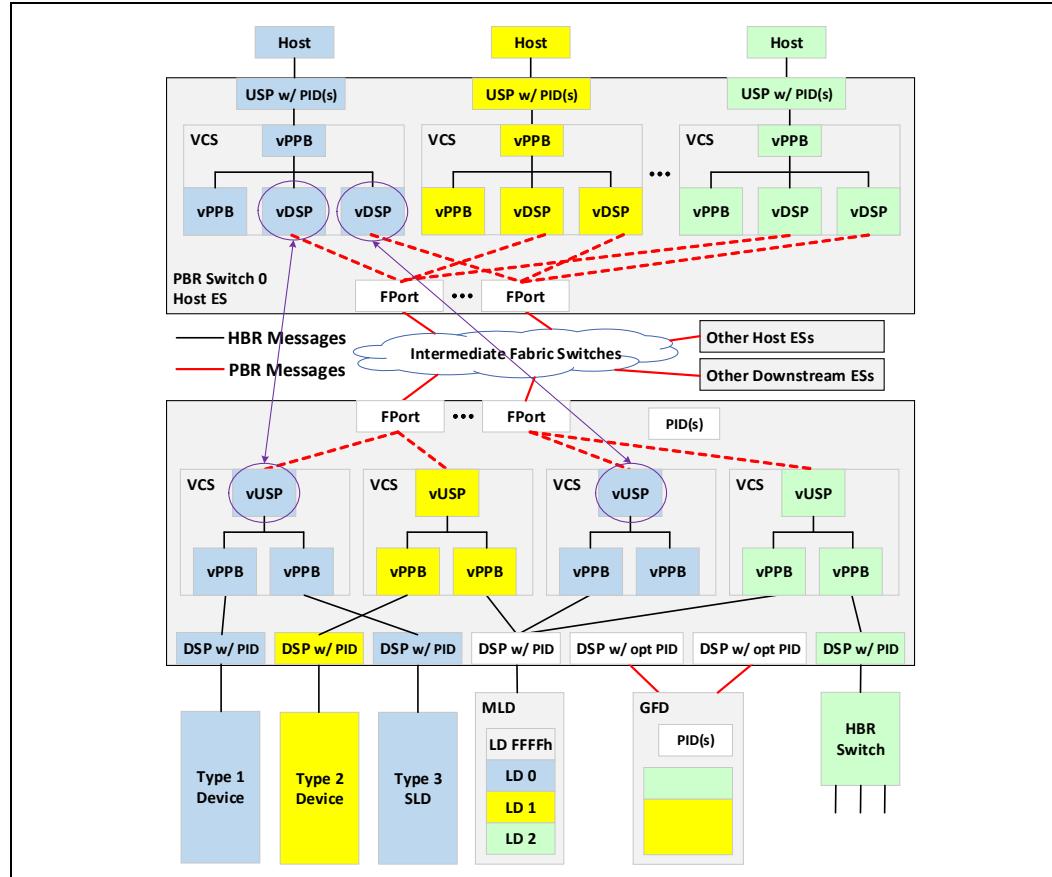
7.7.6.4

PBR Switch vDSP/vUSP Bindings and Connectivity

Within the context of a single VH, the virtual connection between a VCS in the Host ES and a VCS in a Downstream ES is accomplished with a vDSP/vUSP binding. A vDSP is a vPPB in the Host ES VCS that the host sees as a DSP. A vUSP is a vPPB in the Downstream ES VCS that the host sees as a USP. Host software always sees a single virtual link connecting the vDSP and vUSP, even though one or more intermediate Fabric switches may be physically present.

Figure 7-46 shows an example PBR Fabric that consists of one Host ES, one Downstream ES, and an unspecified number of intermediate Fabric switches connecting the two.

Figure 7-46. Example PBR Fabric



The rules for vDSP/vUSP bindings are as follows:

- Each active Host ES vDSP is bound to one Host ES FPort and one Downstream ES vUSP
- Each active Downstream ES vUSP is bound to one Downstream ES FPort and one Host ES vDSP
- All messages routed using a vDSP/vUSP binding must contain both a DPID and an SPID
- vDSPs and vUSPs are never assigned PIDs
- Each PID used for vDSP/vUSP bindings may support both static and dynamic routing; however, vDSP/vUSP traffic always uses static routing
- Each vDSP/vUSP binding has a single host USP PID that determines which Host ES FPort will be used to route from vUSP to vDSP
- Each vDSP/vUSP binding has a single Downstream ES PID that determines which Downstream ES FPort will be used to route from vDSP to vUSP

When a Host ES FPort transmits a vDSP/vUSP message downstream in a PBR flit, the message contains the DPID and SPID taken from the vDSP's binding. Assuming no errors, the message traverses any intermediate Fabric switches that are present and is received by an FPort that is bound to the Downstream ES vUSP. A vUSP there claims the message by matching both the DPID and SPID from its binding.

Similarly, when a Downstream ES FPort transmits a vDSP/vUSP message upstream in a PBR flit, the message contains the DPID and SPID taken from the vUSP's binding. Assuming no errors, the message traverses any intermediate Fabric switches that are present and is received by an FPort that is bound to the Host ES vDSP. A vDSP there claims the message by matching both the DPID and SPID from its binding.

7.7.6.5

PID Use Models and Assignments

The example PBR Fabric illustrated in [Figure 7-46](#) illustrates key aspects of how PIDs can be assigned and used. PIDs are either assigned by the FM or by static fabric initialization (see [Section 7.7.12.1.1](#)).

A Host ES USP often has one PID but may have multiple PIDs assigned to support multiple vDSP/vUSP bindings in the same Downstream ES. Each vDSP/vUSP binding may use a different Host ES FPort and/or Downstream ES FPort, providing traffic isolation for differentiated quality of service. If multiple vDSP bindings use the same PID for the Downstream ES, different PIDs for the USP can distinguish their bindings.

The Downstream ES FPorts may have one or more PIDs assigned, where each PID can be associated with a different set of FPorts. In an example scenario, there might be one PID for the left set of FPorts for multipathing and another PID for the right set. For a PID assigned to an FPort set for multipathing, DRTs in different USPs can specify different egress ports for static routing, distributing the static routing traffic for certain topologies without requiring additional DS_ES PIDs.

A DSP may be assigned multiple PIDs, one PID, or no PIDs. A DSP above a non-GFD usually has one PID, but may be assigned multiple PIDs for isolating traffic from multiple senders or for associating a unique PID for each caching or HDM-DB-capable device attached to one or more HBR switches below an Edge Port. DSPs above a multi-ported GFD may not require dedicated assigned PIDs, relying instead on one or more PIDs assigned to the GFD itself.

A GFD may have one or more PIDs assigned. A multi-ported GFD may have multiple PIDs assigned for differentiated quality of service, though a single PID may be sufficient for congestion avoidance.

As mentioned in the previous section, each vDSP/vUSP binding has two PIDs assigned. For downstream vDSP/vUSP messages that use a given binding, the SPID is a PID associated with the host Edge USP, and the DPID is a PID associated with the Downstream ES FPort. Such messages are always transmitted by the same Host ES FPort and received by the same Downstream ES FPort. Then, the FPort uses various vUSP info decoding mechanisms to route the message to the appropriate Downstream ES vPPB using PBR mechanisms or HBR mechanisms, depending upon the message class. See CXL Switch Message Conversion (see [Section 7.7.6.6](#)). If there are any intermediate Fabric switches, such messages always take a single static path.

Upstream vDSP/vUSP messages are handled in a similar manner, but only involve CXL.io message classes. On a given binding, the SPID is the PID associated with the Downstream ES FPort, and the DPID is the PID associated with the host Edge USP. Such messages are always transmitted by the same Downstream ES FPort and received by the same Host ES FPort. Then, the receiving FPort uses the associated vDSP context to identify the appropriate target using HBR mechanisms. If the target is an egress port, the message is routed there for transmission. If the target is another vDSP, that vDSP

converts the PIDs to its bound PIDs and transmits it from its associated FPort, which may be the same FPort on which it arrived or on a different FPort. If there are any intermediate Fabric switches, such messages always take a single static path.

A PBR switch requires an assigned PID to send and receive management requests, responses, and notifications. Transactions that target this PID are processed by central logic or by FW within the switch.

FMs connected to a PBR switch via an MCTP-based CCI also consume a PID. This PID is communicated to the PBR switch when the FM claims ownership of the device. The PID is used to direct transactions to the FM, such as Event Notifications generated by components owned by the FM.

PID FFFh is reserved and is used to indicate that a transaction should be processed locally. It allows FMs to target devices before they have had a valid PID assigned and when they have an assigned PID of which the FM is unaware.

7.7.6.6

CXL Switch Message Format Conversion

A PBR switch converts messages received from HBR hosts, devices, and switches to the PBR message format for routing across a PBR Fabric. In addition, messages received from the PBR fabric that target the HBR hosts, devices, and switches are converted to messages using the non-PID spaces (i.e., CacheID, BI-ID, and LD-ID). The following subsections provide the conversion flow for each message class.

The FM assigns PIDs to various PBR switch ports, as described in [Section 7.7.6.5](#). The DPID value for request messages is determined by a variety of ways, including HDM Decoders, vPPB bindings, and lookup tables or CAMs using non-PID spaces. The DPID value for a response message is often the SPID value from the associated request message but is sometimes determined by one of the ways mentioned for request messages.

With HBR format messages, MLDs support a 4-bit LD-ID field in CXL.mem protocol for selection and routing of MLD messages, and CXL.cache includes a 4-bit CacheID field that is used to allow up to 16 Type 1 Devices or Type 2 Devices below an RP. PBR format messages use 12-bit PIDs to support large Fabrics. This section describes the support required in PBR switches for routing messages from non-fabric-aware hosts and devices that support the 4-bit LD-ID and 4-bit CacheID fields. It also covers BI-ID-based routing.

Considering the wide range of supported PBR/HBR switch topologies, the variety of specific routing techniques for the many different cases of port connectivity is quite complex. Below is a general description for the HBR and PBR switch routing mechanisms that are used by key message classes, followed by port processing tables with more-specific details for both classes of switches.

7.7.6.6.1

CXL.io, Including UIO

An HBR switch routes most CXL.io TLPs between its ports using standard mechanisms defined by PCIe Base Specification. A DSP above an MLD uses LD-ID Prefixes to identify which LD a downstream TLP is targeting or from which LD an upstream TLP came.

UIO Requests that directly target HDM ranges can use enhanced UIO-capable HDM Decoders for their routing. This includes UIO Requests from the host that target devices with HDM, as well as “Direct P2P” cases where UIO Requests from one device target other devices with HDM. UIO Direct P2P to HDM traffic goes upstream, P2P, and downstream along different portions of its path.

A PBR switch converts PCIe-format TLPs or CXL.io HBR-format TLPs to PBR-format TLPs by pre-pending to each TLP a 4B CXL PBR TLP Header (PTH), which includes an SPID and DPID. Conversion from PBR format to HBR format or PCIe format consists of stripping the CXL PTH from the TLP.

7.7.6.6.2 CXL.cache

A number of CXL.cache messages in 256B HBR format have a 4-bit CacheID field that enables up to 16 caching devices below a single RP. CXL.cache messages in 68B HBR format do not support this feature, and thus never carry a CacheID field. CXL.cache messages in PBR format do support this feature, but convey the necessary information via PIDs instead of a CacheID field. [Table 7-86](#) summarizes which message classes contain the CacheID field.

Table 7-86. Summary of CacheID Field

Msg Class	CacheID Field		
	68B HBR	256B HBR	256B PBR
D2H Req	No	Yes	No
H2D Rsp	No	Yes	No
H2D DH	No	Yes	No
H2D Req	No	Yes	No
D2H Rsp	No	No	No
D2H DH	No	No	No

For HBR format messages that contain a CacheID field, in some cases an HBR or PBR DSP needs to know whether to propagate or assign the CacheID. This information is configured by host software and is contained in the CXL Cache ID Decoder Capability Structure (see [Section 8.2.4.29](#)).

[Table 7-87](#) summarizes the HBR switch routing for CXL.cache message classes. [Table 7-88](#) summarizes the PBR switch routing for CXL.cache message classes.

Table 7-87. Summary of HBR Switch Routing for CXL.cache Message Classes

Message Class	Switch Routing
D2H Request	For HBR switch routing of D2H requests upstream to the bound host, the D2H request to the USP relies on the DSP's vPPB binding at each switch level. CacheID is added to the message by the DSP above the device to enable routing of the H2D response.
H2D Response or Data Header	For HBR switch routing of H2D responses or data headers downstream to the DSP, the USP at each switch level looks up the PCIe-defined PortID from the Cache ID Route Table.
H2D Request	For HBR switch routing of H2D requests downstream to the DSP, the USP at each switch level looks up the PCIe-defined PortID from the Cache ID Route Table.
D2H Response or Data Header	For HBR switch routing of D2H responses or data headers upstream to the bound host, the D2H response or data header to the USP relies upon the DSP's vPPB binding at each switch level.

Within a PBR fabric, all CXL.cache messages are routed edge-to-edge, and they never use vDSP/vUSP bindings.

In contrast to most 256B HBR-format CXL.cache messages, PBR-format cache messages never contain a CacheID field, thus the equivalent information when needed must be conveyed via PIDs.

When multiple caching devices are attached to an HBR switch below a PBR fabric, the FM must allocate and assign a unique PID for each such caching device. This enables PBR switches to convert between a caching device's unique PID and CacheID when needed.

Table 7-88. Summary of PBR Switch Routing for CXL.cache Message Classes

Message Class	Switch Routing
D2H Request	For PBR switch routing of these messages upstream to the host, Edge DSPs get the Host USP DPID from their vPPB. Those above an SLD get their SPID from their vPPB. Those above an HBR USP look up the SPID from the Cache ID Route Table using the CacheID contained in the HBR-format message. For converting to HBR format at the Edge USP, the USP derives the CacheID from a 16-entry CAM using the SPID.
H2D Response or Data Header	For PBR switch routing of these messages downstream to the Edge DSP, the Edge USP looks up the DPID from the Cache ID Route Table using the CacheID in the HBR-format message. For converting to HBR format at the Edge DSP, above an SLD the CacheID is unused, and above an HBR USP the Cache ID is derived from a 16-entry CAM match using the DPID.
H2D Request	For PBR switch routing of these messages downstream to the Edge DSP, the Edge USP looks up the DPID from the CacheID Route Table using the CacheID. The USP gets the SPID from its vPPB. For converting to HBR format at the Edge DSP, above an SLD the CacheID is unused, and above an HBR USP the CacheID is derived from a 16-entry CAM match using the DPID.
D2H Response or Data Header	For PBR switch routing of these messages upstream to the host, Edge DSPs get the DPID from their vPPB. For converting to HBR format at the Edge USP, the CacheID field is not present in the message.

At an Edge DSP, when converting a downstream CXL.cache message from PBR to HBR format, if the CacheID field is unused, its value shall be cleared to 0.

7.7.6.6.3 CXL.mem

Several CXL.mem message classes in HBR format have a 4-bit LD-ID field that is used by Type 3 MLDS for determining the targeted LD. This feature is supported by both 68B and 256B HBR formats. PBR format conveys the necessary information via PIDs instead of an LD-ID field. [Table 7-89](#) summarizes which message classes contain the LD-ID field.

Table 7-89. Summary of LD-ID Field

Msg Class	LD-ID Field		
	68B HBR	256B HBR	256B PBR
M2S Req	Yes	Yes	No
M2S RwD	Yes	Yes	No
S2M NDR	Yes	Yes	No
S2M DRS	Yes	Yes	No
S2M BISnp	N/A	In BI-ID	No
M2S BIRsp	N/A	In BI-ID	No

CXL.mem BISnp/BIRsp messages support the Back-Invalidate feature in 256B HBR format via a 12-bit BI-ID field, which determines the routing for BIRsp. This feature and its associated field are not supported in 68B HBR format. PBR format supports this feature and conveys the necessary information via 12-bit PIDs. [Table 7-90](#) summarizes which message classes contain the BI-ID field.

In 256B HBR format over an MLD link, the 12-bit BI-ID field in BISnp/BIRsp carries the 4-bit LD-ID value, and the remaining 8 bits are all 0s. In 256B HBR format over non-MLD links, the 12-bit BI-ID field carries the 8-bit Bus Number of the HDM-DB device, and the remaining 4 bits are all 0s.

Table 7-90. Summary of BI-ID Field

Msg Class	BI-ID Field		
	68B HBR	256B HBR	256B PBR
S2M BISnp	N/A	Yes	No
M2S BIRsp	N/A	Yes	No

For messages that contain a BI-ID field, in some cases an HBR or PBR DSP needs to know whether to propagate or assign the BI-ID. This information is configured by host software and is contained in the CXL BI Decoder Capability Structure (see [Section 8.2.4.27](#)).

The Direct P2P CXL.mem to Accelerators use case, supported only by PBR fabrics, is not covered in this section; see [Section 7.7.10](#).

[Table 7-91](#) summarizes the HBR switch routing for CXL.mem message classes.

[Table 7-92](#) summarizes the PBR switch routing for CXL.mem message classes.

Table 7-91. Summary of HBR Switch Routing for CXL.mem Message Classes

Message Class	Switch Routing
M2S Request	For HBR switch routing of M2S requests downstream toward the device, the HDM Decoder at the USP determines the PCIe-defined PortID of the DSP at each switch level. For a DSP above an MLD, there is a vPPB for each LD, which provides the LD-ID to insert in the request message.
S2M Response	For HBR switch routing of S2M responses upstream to the USP, the DSP relies on its vPPB binding at each switch level. For a DSP immediately above an MLD, there is a vPPB for each LD, and the LD-ID in the response message identifies the associated vPPB.
S2M BISnp	For HBR switch routing of S2M BISnp requests upstream to the USP, the DSP relies on its vPPB binding at each switch level. For a DSP immediately above an MLD, there is a vPPB for each LD, and the BI-ID in the response message carries an LD-ID that identifies the associated vPPB. The DSP then looks up the BusNum associated with its vPPB, places the BusNum in the BI-ID for later use in routing the associated BIRsp back to the DSP.
M2S BIRsp	For HBR switch routing of M2S BIRsp messages downstream to the DSP immediately above the device, the USP at each switch level relies on the BI-ID that carries the BusNum of the target DSP. The HBR switch then uses BusNum routing.

In an HBR switch, when filling in a subset of the bits in the BI-ID field with a value, the remaining bits in the BI-ID field shall be cleared to 0.

Within a PBR fabric, most CXL.mem message classes are routed edge-to-edge and do not use vDSP/vUSP bindings. The exceptions are M2S Req/RwD message classes with LD-FAM when host software has configured HDM Decoders in the Host ES USP to route them, in which case vDSP/vUSP bindings are used. See details regarding PBR Message Routing across the Fabric in [Section 7.7.6.2](#).

When HDM-DB devices are attached to an HBR switch below a PBR fabric, the FM must allocate and assign a unique PID for each HDM-DB device. This enables PBR switches to convert between an HDM-DB device's unique PID and Bus Number when needed.

Table 7-92. Summary of PBR Switch Routing for CXL.mem Message Classes

Message Class	Switch Routing
M2S Request	<p>FAST/LDST Decoder Case: For Host ES routing of M2S requests downstream to the Edge DSP, the FAST/LDST decoder at the USP determines the DPID for routing the message edge-to-edge.</p> <p>HDM Decoder Case: For hierarchical routing of M2S requests downstream toward the Edge DSP, the HDM Decoder at the USP of each ES determines the egress vPPB (EvPPB), which contains an appropriate DPID. A vDSP in the Host ES contains the DPID/SPID that is used for targeting its partner Downstream ES vUSP. A DSP vPPB contains its dedicated DPID. Both host and Downstream ESs use PBR routing locally because a DSP above an MLD relies on the request having a valid SPID.</p> <p>For a DSP immediately above an MLD, a 16-entry CAM match using the SPID returns the associated LD-ID, which determines the LD-specific vPPB to use and is also inserted in the request message. For a DSP above a GFD, the message remains in PBR format.</p>
S2M Response	<p>For Edge DSP routing of S2M responses upstream to the Edge USP, the Edge DSP's vPPB contains the DPID for routing the message edge-to-edge. For a DSP immediately above an MLD, there is a vPPB for each LD, and the LD-ID in the response message identifies the associated vPPB. For a DSP above a GFD, the message is already in PBR format and remains so.</p>
S2M BISnp	<p>For Edge DSP routing of S2M BISnp messages upstream to the Edge USP, the Edge DSP's vPPB contains the DPID for routing the message edge-to-edge. For an Edge DSP immediately above an MLD, there is a vPPB for each LD, and the BI-ID in the BISnp carries an LD-ID that identifies the associated vPPB. The Edge DSP uses its vPPB's PID for the SPID.</p> <p>For an Edge DSP above an HBR USP, the BI-ID contains the BusNum associated with the HDM-DB device. The Edge DSP uses the BusNum to look up the associated SPID from a 256-entry table.</p> <p>At the Edge USP, the USP converts the BISnp to HBR format, copying the SPID value into the BI-ID.</p>
M2S BIRsp	<p>For Edge USP routing of M2S BIRsp messages downstream to the Edge DSP above the HDM-DB device, the Edge USP converts the BIRsp to PBR format, using the BI-ID value as the DPID, and then routes the BIRsp edge-to-edge.</p> <p>For an Edge DSP immediately above an MLD, a 16-entry CAM match using the SPID returns the associated LD-ID, which determines the LD-specific vPPB to use and is also inserted in the BI-ID field of the BIRsp. For an Edge DSP above an HBR switch USP, the DSP converts the BIRsp to HBR format, looking up the target BusNum in a 4k-entry table using the DPID, then copying it to the BI-ID. For a DSP above a GFD, the message remains in PBR format.</p>

At an Edge DSP, when converting a downstream CXL.mem message from PBR to HBR format, if an LD-ID or BI-ID field is unused, its value shall be cleared to 0. Also, when filling in a subset of the bits in the BI-ID field with a value, the remaining bits in the BI-ID field shall be cleared to 0.

7.7.6.7

HBR Switch Port Processing of CXL Messages

Table 7-93, **Table 7-94**, and **Table 7-95** summarize how HBR switches perform port processing of CXL.io, CXL.cache, and CXL.mem messages, respectively. A USP is below either an RP, a PBR DSP, or an HBR DSP. A USP can be in only one Virtual Hierarchy. A DSP is above either an HBR switch USP, an SLD, or an MLD.

For conciseness, there are many abbreviations within the tables. US stands for upstream. DS stands for downstream. P2P stands for peer-to-peer. DMA stands for direct memory access. Direct P2P stands for UIO Direct P2P to HDM. BusNum stands for Bus Number. “←” stands for assignment (e.g., “LD-ID Prefix ← vPPB context” means “the LD-ID prefix is assigned a value from the associated vPPB context”). Text beginning with “PCIe” (also shown in gold) means that the routing is defined in PCIe Base Specification.

In the CXL.io table (see [Table 7-93](#)), not all TLP types are explicitly covered; however, those not listed are usually handled by standard PCIe routing mechanisms (e.g., PCIe Messages are not explicitly covered, but ID-routed Messages are handled by PCIe ID routing, and address-routed Messages are handled by PCIe Memory Address routing).

Table 7-93. HBR Switch Port Processing Table for CXL.io

Message Class and Direction	HBR USP below RP or PBR/HBR DSP	HBR DSP		
		Above HBR USP	Above SLD	Above MLD
Cfg Req DS	PCIe ID routing	PCIe ID routing		PCIe ID routing LD-ID Prefix↔vPPB context
Mem Req DS/US/P2P Incl UIO DMA Excl HDM UIO	PCIe Mem addr routing	PCIe Mem addr routing		PCIe Mem addr routing US: LD-ID Prefix identifies vPPB DS: LD-ID Prefix↔vPPB context
HDM UIO Req Direct P2P and Host Requester	US: PCIe Mem addr routing DS: HDM Decoder routing	US: PCIe Mem addr routing DS/Direct P2P: USP HDM Decoder		US: PCIe Mem addr routing DS/Direct P2P: USP HDM Decoder US: LD-ID Prefix identifies vPPB DS: LD-ID Prefix↔vPPB context
Cpl US	PCIe ID routing	PCIe ID routing		LD-ID Prefix identifies vPPB PCIe ID routing
Cpl DS	PCIe ID routing	PCIe ID routing		PCIe ID routing LD-ID Prefix↔vPPB context

Table 7-94. HBR Switch Port Processing Table for CXL.cache

Message Class and Direction	HBR USP below RP or PBR/HBR DSP	HBR DSP		
		Above HBR USP	Above SLD	Above MLD
D2H Req US	Propagate CacheID	Propagate CacheID vPPB binding routing to USP	CacheID↔Local Cache ID field vPPB binding routing to USP	
H2D Rsp/DH DS	Propagate CacheID PortID↔Cache ID Route Table		Propagate Cache ID	
H2D Req DS	PortID routing to DSP OS must handle multi-level HBR	Propagate CacheID	(SLD should ignore it)	
D2H Rsp/DH US	-	vPPB binding routing to USP		

Table 7-95. HBR Switch Port Processing Table for CXL.mem

Message Class and Direction	HBR USP below RP or PBR/HBR DSP	HBR DSP		
		Above HBR USP	Above SLD	Above MLD
M2S Req DS	PortID←HDM Decoder (HPA) Routing to DSP uses PortID	Propagate LD-ID (not used by these receivers)		LD-ID←vPPB context
S2M Rsp US	Propagate LD-ID (not used by these receivers)	vPPB binding routing to USP Propagate LD-ID (not used for internal switch routing)		LD-ID identifies vPPB vPPB binding routing to USP
S2M BISnsp US	BI-ID[7:0] contains BusNum Propagate BI-ID	Propagate BI-ID vPPB binding routing to USP	Received BI-ID is ignored BI-ID[7:0]←BusNum(vPPB) vPPB binding routing to USP	BI-ID[3:0] contains LD-ID LD-ID identifies vPPB BI-ID[7:0]←BusNum(vPPB) vPPB binding routing to USP
M2S BIRrsp DS	Target BusNum←BI-ID[7:0] PCIe BusNum routing to DSP	Propagate BI-ID	Propagate BI-ID (SLD should ignore it)	BI-ID[3:0]←LD-ID(vPPB)

7.7.6.8 PBR Switch Port Processing of CXL Messages

[Table 7-96](#), [Table 7-97](#), and [Table 7-98](#) summarize how PBR switches perform port processing of CXL.io, CXL.cache, and CXL.mem messages, respectively. A PBR USP must be below an RP and can be in only one Virtual Hierarchy. A PBR DSP is above either an SLD, an MLD, a GFD, or an HBR switch USP. A PBR FPort can only be connected to another PBR FPort in a different PBR switch.

For conciseness, there are many abbreviations within the tables. US stands for upstream. DS stands for downstream. P2P stands for peer-to-peer. DMA stands for direct memory access. Direct P2P stands for UIO Direct P2P to HDM. EvPPB stands for Egress vPPB. BusNum stands for Bus Number. RT stands for the CacheID Route Table. “←” stands for assignment (e.g., “LD-ID Prefix ← vPPB context” means “the LD-ID prefix is assigned a value from the associated vPPB context”). Also referring to a vPPB context, vPPB.root.PID stands for the PID of the associated Edge USP, and vPPB.self.PID stands for the PID of the vPPB itself. Eg2Eg means Edge-to-Edge. Text beginning with “PCIe” (also shown in gold) means that the routing is defined in PCIe Base Specification.

In the CXL.io table (see [Table 7-96](#)), not all TLP types are explicitly covered; however, those not listed are usually handled by standard PCIe routing mechanisms (e.g., PCIe Messages are not explicitly covered, but ID-routed Messages are handled by PCIe ID routing, and address-routed Messages are handled by PCIe Memory Address routing). Also, the UIO Direct P2P to HDM use case is not covered; see [Section 7.7.7](#).

In the CXL.mem table (see [Table 7-98](#)) the Direct P2P CXL.mem to Accelerators use case is not covered; see [Section 7.7.10](#).

Table 7-96. PBR Switch Port Processing Table for CXL.io

Message Class and Direction	Edge USP	Edge DSP in Either Host ES or Downstream ES			
	Always Below an RP	Above HBR Switch USP	Above SLD	Above MLD	Above GFD
Cfg Req DS	PCIe ID routing	PCIe ID routing		PCIe ID routing LD-ID Prefix<vPPB LD-ID	N/A
Mem Req DS/US/P2P Incl UIO DMA Excl HDM UIO	PCIe Mem addr routing	PCIe Mem addr routing		PCIe Mem addr routing DS: LD-ID Prefix< vPPB LD-ID US incl P2P: LD-ID Prefix identifies vPPB	N/A
Cpl US Excl HDM UIO	PCIe ID routing	PCIe ID routing		LD-ID Prefix identifies vPPB PCIe ID routing	N/A
Cpl DS Excl HDM UIO	PCIe ID routing	PCIe ID routing		PCIe ID routing LD-ID Prefix<vPPB LD-ID	N/A
HDM UIO Req Direct P2P and Host Requester	Direct P2P: N/A Host Requester (DS): Either use FAST/LDST to convert to PBR and route Eg2Eg, or use USP's HDM decoder routing within this switch DS: Convert to HBR US/DS above MLD: Handle LD-ID Prefix the same way as handled by US and DS mem requests	US: Either use FAST/LDST to convert to PBR and route Eg2Eg, or use USP's HDM decoder routing within this switch DS: Convert to HBR US/DS above MLD: Handle LD-ID Prefix the same way as handled by US and DS mem requests			US: N/A DS: Keep in PBR
HDM UIO Cpl Direct P2P and Host Requester	Direct P2P: N/A Host Requester (US): Convert to HBR	US: Either use UIO ID-based Re-Router to convert to PBR and route Eg2Eg, or use PCIe ID routing DS: Convert to HBR US/DS above MLD: Handle LD-ID Prefix the same way as handled by US and DS Cpls			US: Keep in PBR and route Eg2Eg DS: N/A

Table 7-97. PBR Switch Port Processing Table for CXL.cache

Message Class and Direction	PBR Edge USP	PBR Edge DSP in Either Host ES or Downstream ES			
	Always Below an RP	Above HBR Switch USP	Above SLD	Above MLD	Above GFD
D2H Req US	Convert to HBR fmt CacheID<CAM ₁₆ (SPID)	Convert to PBR fmt DPID<vPPB.root.PID SPID<RT(CacheID)	Convert to PBR fmt DPID<vPPB.root.PID SPID<vPPB.self.PID		
H2D Rsp/ DH DS	Convert to PBR fmt DPID<RT(CacheID)	Convert to HBR fmt 256B: CacheID< CAM ₁₆ (DPID) 68B: Has no CacheID	Convert to HBR fmt 256B: CacheID<0 68B: Has no CacheID		
H2D Req DS	Convert to PBR fmt DPID<RT(CacheID) SPID<vPPB.self.PID				
D2H Rsp/ DH US	Convert to HBR fmt	Convert to PBR fmt DPID<vPPB.root.PID	Convert to PBR fmt DPID<vPPB.root.PID		

Table 7-98. PBR Switch Port Processing Table for CXL.mem

Message Class and Direction	PBR Edge USP	PBR Edge DSP in Either Host ES or Downstream ES			
	Always Below an RP	Above HBR Switch USP	Above SLD	Above MLD	Above GFD
M2S Req/Rwd DS	FAST or LDST: Convert to PBR fmt DPID←xxST(HPA) SPID←vPPB.self.PID	Convert to HBR fmt LD-ID is unused	LD-ID←CAM ₁₆ (SPID) Convert to HBR MLD fmt	(no LD-ID field) Keep in PBR fmt	N/A
	HDM Decoder: Convert to PBR fmt EvPPB←HDM-Dec(HPA) DPID←EvPPB.bndg. DPID SPID←vPPB.self.PID				
S2M NDR/DRS US	Convert to HBR fmt LD-ID is unused	LD-ID is ignored Convert to PBR fmt DPID←vPPB.root.PID		LD-ID identifies vPPB Convert to PBR fmt DPID←vPPB.root.PID	Keep in PBR fmt (no LD-ID field)
S2M BISnp US	Convert to HBR fmt BI-ID[11:0]←SPID	Convert to PBR fmt DPID←vPPB.root.PID BusNum←BI-ID[7:0] SPID←RAM ₂₅₆ (BusNum)	Convert to PBR fmt DPID←vPPB.root.PID SPID←vPPB.self.PID	BI-ID[3:0] contains LD-ID LD-ID identifies vPPB Convert to PBR fmt DPID←vPPB.root.PID SPID←vPPB.self.PID	Keep in PBR fmt
M2S BIRsp DS	Convert to PBR fmt DPID←BI-ID[11:0]	Convert to HBR fmt BusNum←RAM _{4k} (DPID) BI-ID[7:0]←BusNum	Convert to HBR fmt BI-ID is unused	Convert to HBR fmt LD-ID←CAM ₁₆ (SPID) BI-ID[3:0]←vPPB.LD-ID	Keep in PBR fmt

7.7.6.9 PPB and vPPB Behavior of PBR Link Ports

A PBR Link port has two varieties: an Inter-Switch Link (ISL) and a GFD Link.

The ISL case is a downstream-to-downstream crosslink. The DSP on each side of the link is managed by the FM with assistance from switch firmware. The full PCIe capabilities of a DSP shall be available. Bus master enable, AER, DPC, and other capabilities that an host typically controls will be controlled by the FM and/or switch firmware.

Other users of an ISL can be any number of VHs. The ISL (and as many switch hops and additional ISLs as it takes) functions as a single link between vDSP and vUSP. Any one ISL can potentially be shared by multiple VHs. Because a VH shares the link with other VHs, there is no way for a VH to control any of the link physical characteristics. However, the Host ES vDSP shall reflect the physical link settings for the fabric port DSP to which it is bound (e.g., link speed, lane margining, etc.).

A GFD PBR link is similar to an ISL in that many VH can share it. It is different however in that no vDSP nor vUSP is associated with it. The link itself is a simple up-down link, with the switch having an (FM-owned) DSP leading, via the PBR link, to the USP of a GFD. A switch DSP should have full PCIe capabilities, just like for an ISL or any other DSP.

The remainder of this section focuses on the vDSP and vUSP perspective, from the PCIe configuration space, for a variety of capabilities:

- “Supported” means that the PCIe register is available to be read and written by the host
- “Not supported” means that the register is either read-only or the capability is unavailable

- “Mirrors DSP” means that the values reflect the (typically physical link) value in the DSP
- “Read/Write with no effect” implies that the vDSP/vUSP register will be unaffected by reads and writes

It is expected that a fabric port DSP supports all PCIe capabilities required by the PCIe spec for a downstream port. DPC, which is optional for PCIe, is required for CXL for a DSP that is a fabric port.

7.7.6.9.1 ISL Type 1 Configuration Space Header

Table 7-99. ISL Type 1 Configuration Space Header

Register	Register Fields	FM-owned DSP	vDSP	vUSP
Bridge Control Register	Parity Error Response Enable	Supported	Supported	Supported
	SERR# Enable	Supported	Supported	Supported
	ISA Enable	Not Supported	Not Supported	Not Supported
	Secondary Bus Reset	Supported	Supported	Supported

7.7.6.9.2 ISL PCI-compatible Configuration Register

Table 7-100. ISL PCI-compatible Configuration Space Header

Register	Register Fields	FM-owned DSP	vDSP	vUSP
Command	I/O Space Enable	Hardwire to 0	Supported	Supported
Command	Memory Space Enable	Supported	Supported	Supported
	Bus Master Enable	Not Supported	Supported	Supported
	Parity Error Response	Supported	Supported	Supported
	SERR# Enable	Supported	Supported	Supported
	Interrupt Disable	Supported	Supported	Supported
Status	Interrupt Status	Hardwire to 0	Supported	Supported
	Master Data Parity Error	Supported	Supported	Supported
	Signaled System Error	Supported	Supported	Supported
	Detected Parity Error	Supported	Supported	Supported

7.7.6.9.3 ISL PCIe Capability Structure

Table 7-101. ISL PCIe Capability Structure (Sheet 1 of 3)

Register	Register Fields	FM-owned DSP	vDSP	vUSP
Device Capabilities	Max Payload Size Supported	FM Configured	Mirrors DSP	Mirrors DSP
	Phantom Functions Supported	0	0	0
	Extended Tag Field Supported	Supported	Supported	Supported
Device Control	Max Payload Size	FM Configured	Mirrors DSP	Mirrors DSP
Link Capabilities	Link Bandwidth Notification Capability	0	0	0
	ASPM Support	No L0s	no L0s	no L0s
	Clock Power Management	No PM L1 Substates	Mirrors DSP	Mirrors DSP

Table 7-101. ISL PCIe Capability Structure (Sheet 2 of 3)

Register	Register Fields	FM-owned DSP	vDSP	vUSP
Link Control	ASPM Control	Supported	Not Supported	Not Supported
	Link Disable	Supported	Supported	Not Supported
	Retrain Link	Supported	Read/Write with no effect	Not Supported
	Common Clock Configuration	Supported	Read/Write with no effect	
	Extended Synch	Supported	Read/Write with no effect	
	Hardware Autonomous Width Disable	Supported	Read/Write with no effect	
	Link Bandwidth Management Interrupt Enable	Supported	Read/Write with no effect	Not Supported
	Link Autonomous Bandwidth Interrupt Enable	Supported	Supported	Not Supported
	Flit Mode Disable	0	0	0
	DRS Signaling Control	Supported	Supported	Not Supported
Link Status	Current Link Speed	Supported	Mirrors DSP	Mirrors DSP
	Negotiated Link Speed	Supported	Mirrors DSP	Mirrors DSP
	Link Training	Supported	0	0
	Slot Clock Configuration	Supported	Mirrors DSP	Mirrors DSP
	Data Link Layer Active	Supported	Mirrors DSP	0
	Link Bandwidth Management Status	Supported	Mirrors DSP	0
	Link Autonomous Bandwidth Status	Supported	Mirrors DSP	0
Slot Capabilities	Hot-Plug Surprise	Supported	Mirrors DSP	0
	Physical Slot Number	Supported	Supported	0
Slot Status	Attention Button Pressed	Supported	Supported	0
	Power Fault Detected	Supported	Mirrors DSP	0
	MRL Sensor Changed	Supported	Mirrors DSP	0
	Presence Detect Changed	Supported	Supported	0
	MRL Sensor State	Supported	Mirrors DSP	0
	Presence Detect State	Supported	Supported	0
	Electromechanical Interlock Status	Supported	Mirrors DSP	0
	Data Link Layer State Changed	Supported	Supported	0
Device Capabilities 2	All bits	Supported	Supported	0
Device Control 2	ARI Forwarding Enable	Supported	Supported	0
	Atomic Op Egress Blocking	Supported	Supported	0
	LTR Mechanism Enabled	Supported	Supported	0
	Emergency Power Reduction Request	Supported	Read/Write with no effect	0
	End-End TLP Prefix Blocking	Supported	Mirrors DSP. Read/Write with no effect	0

Table 7-101. ISL PCIe Capability Structure (Sheet 3 of 3)

Register	Register Fields	FM-owned DSP	vDSP	vUSP
Link Control 2	Target Link Speed	Supported	Read/Write with no effect	Read/Write with no effect
	Enter Compliance	Supported	Read/Write with no effect	Read/Write with no effect
	Hardware Autonomous Speed Disable	Supported	Read/Write with no effect	Read/Write with no effect
	Selectable De-emphasis	Supported	Read/Write with no effect	Read/Write with no effect
	Transmit Margin	Supported	Read/Write with no effect	Read/Write with no effect
	Enter Modified Compliance	Supported	Read/Write with no effect	Read/Write with no effect
	Compliance SOS	Supported	Read/Write with no effect	Read/Write with no effect
	Compliance Preset/De-emphasis	Supported	Read/Write with no effect	Read/Write with no effect
Link Status 2	Current De-emphasis Level	Supported	Mirrors DSP	Mirrors DSP
	Equalization 8.0 GT/s Complete	Supported	Mirrors DSP	Mirrors DSP
	Equalization 8.0 GT/s Phase 1 Successful	Supported	Mirrors DSP	Mirrors DSP
	Equalization 8.0 GT/s Phase 2 Successful	Supported	Mirrors DSP	Mirrors DSP
	Equalization 8.0 GT/s Phase 3 Successful	Supported	Mirrors DSP	Mirrors DSP
	Link Equalization Request 8.0 GT/s	Supported	Read/Write with no effect	Read/Write with no effect
	Retimer Presence Detected	Supported	Mirrors DSP	Mirrors DSP
	Two Retimers Presence Detected	Supported	Mirrors DSP	Mirrors DSP
	Crosslink Resolution	Supported	All 0s	All 0s
	Flit Mode Status	Supported	Supported	Supported
	Downstream Component Presence	Supported	Supported	0
	DRS Message Received	Supported	Supported	0

7.7.6.9.4 ISL Secondary PCIe Capability Structure

All fields in the Secondary PCIe Capability Structure for a Virtual PPB shall behave identically to PCIe except the following:

Table 7-102. ISL Secondary PCIe Extended Capability (Sheet 1 of 2)

Register	Register Fields	FM-owned DSP	vDSP	vUSP
Link Control 3	Perform Equalization	Supported	Read/Write with no effect	Read/Write with no effect
	Link Equalization Request Interrupt Enable	Supported	Read/Write with no effect	Read/Write with no effect
	Enable Lower SKP OS Generation Vector	Supported	Read/Write with no effect	Read/Write with no effect

Table 7-102. ISL Secondary PCIe Extended Capability (Sheet 2 of 2)

Register	Register Fields	FM-owned DSP	vDSP	vUSP
Lane Error Status	All fields	Supported	Mirrors DSP	Mirrors DSP
Lane Equalization Control	All fields	Supported	Read/Write with no effect	Read/Write with no effect
Data Link Features Capabilities	All fields	Supported	Mirror DSP	Mirror DSP

7.7.6.9.5 ISL Physical Layer 16.0 GT/s Extended Capability

All fields in the Physical Layer 16.0 GT/s Extended Capability Structure for a Virtual PPB shall behave identically to PCIe except the following:

Table 7-103. ISL Physical Layer 16.0 GT/s Extended Capability

Register	Register Fields	FM-owned DSP	vDSP	vUSP
16.0 GT/s Status	All fields	Supported	Mirrors DSP	Mirrors DSP
16.0 GT/s Local Data Parity Mismatch Status	Local Data Parity Mismatch Status	Supported	Mirrors DSP	Mirrors DSP
16.0 GT/s First Retimer Data Parity Mismatch Status	First Retimer Data Parity Mismatch Status	Supported	Mirrors DSP	Mirrors DSP
16.0 GT/s Second Retimer Data Parity Mismatch Status	Second Retimer Data Parity Mismatch Status	Supported	Mirrors DSP	Mirrors DSP
16.0 GT/s Lane Equalization Control	Downstream Port 16.0 GT/s Transmitter Preset	Supported	Mirrors DSP	Mirrors DSP

7.7.6.9.6 ISL Physical Layer 32.0 GT/s Extended Capability

All fields in the Physical Layer 32.0 GT/s Extended Capability Structure for a Virtual PPB shall behave identically to PCIe except the following:

Table 7-104. ISL Physical Layer 32.0 GT/s Extended Capability

Register	Register Fields	FM-owned DSP	vDSP	vUSP
32.0 GT/s Control Register	All fields	Supported	Read/Write with no effect	Read/Write with no effect
32.0 GT/s Status Register	Link Equalization Request 32.0 GT/s	Supported	Read/Write with no effect	Read/Write with no effect
	All fields except Link Equalization Request 32.0 GT/s	Supported	Mirrors DSP	Mirrors DSP
Received Modified TS Data 1 Register	All fields	Supported	Mirrors DSP	Mirrors DSP
Received Modified TS Data 2	All fields	Supported	Mirrors DSP	Mirrors DSP
Transmitted Modified TS Data 1	All fields	Supported	Mirrors DSP	Mirrors DSP
32.0 GT/s Lane Equalization Control	Downstream Port 32.0 GT/s Transmitter Preset	Supported	Mirrors DSP	Mirrors DSP

7.7.6.9.7 ISL Physical Layer 32.0 GT/s Extended Capability

All fields in the Physical Layer 64.0 GT/s Extended Capability Structure for a Virtual PPB shall behave identically to PCIe except the following:

Table 7-105. ISL Physical Layer 64.0 GT/s Extended Capability

Register	Register Fields	FM-owned DSP	vDSP	vUSP
64.0 GT/s Control Register	All fields	Supported	Read/Write with no effect	Read/Write with no effect
64.0 GT/s Status Register	Link Equalization Request 64.0 GT/s	Supported	Read/Write with no effect	Read/Write with no effect
	All fields except Link Equalization Request 64.0 GT/s	Supported	Mirrors DSP	Mirrors DSP
Received Modified TS Data 1 Register	All fields	Supported	Mirrors DSP	Mirrors DSP
Received Modified TS Data 2	All fields	Supported	Mirrors DSP	Mirrors DSP
Transmitted Modified TS Data 1	All fields	Supported	Mirrors DSP	Mirrors DSP
64.0 GT/s Lane Equalization Control	Downstream Port 64.0 GT/s Transmitter Preset	Supported	Mirrors DSP	Mirrors DSP

7.7.6.9.8 ISL Lane Margining at the Receiver Extended Capability

All fields in the ISL Lane Margining at the Receiver for a Virtual PPB shall behave identically to PCIe except the following:

Table 7-106. ISL Lane Margining at the Receiver Extended Capability

Register	Register Fields	FM-owned DSP	vDSP	vUSP
Margining Port Status Register	All fields	Supported	Mirrors DSP	Mirrors DSP
Margining Lane Control Register	All fields	Supported	Read/Write with no effect	Read/Write with no effect

7.7.6.9.9 ISL ACS Extended Capability

ACS applies only to a Downstream Port which, for a PBR link, applies to either a DSP above a GFD, a DSP connected to a crosslink, or a vDSP in a VH. All fields in the ISL ACS at the Receiver for a Virtual PPB shall behave identically to PCIe.

7.7.6.9.10 ISL Advanced Error Reporting Extended Capability

AER can apply to a vPPB on any side of a link. FM-owned DSPs, vDSPs, and vUSPs support all AER fields.

7.7.6.9.11 ISL DPC Extended Capability

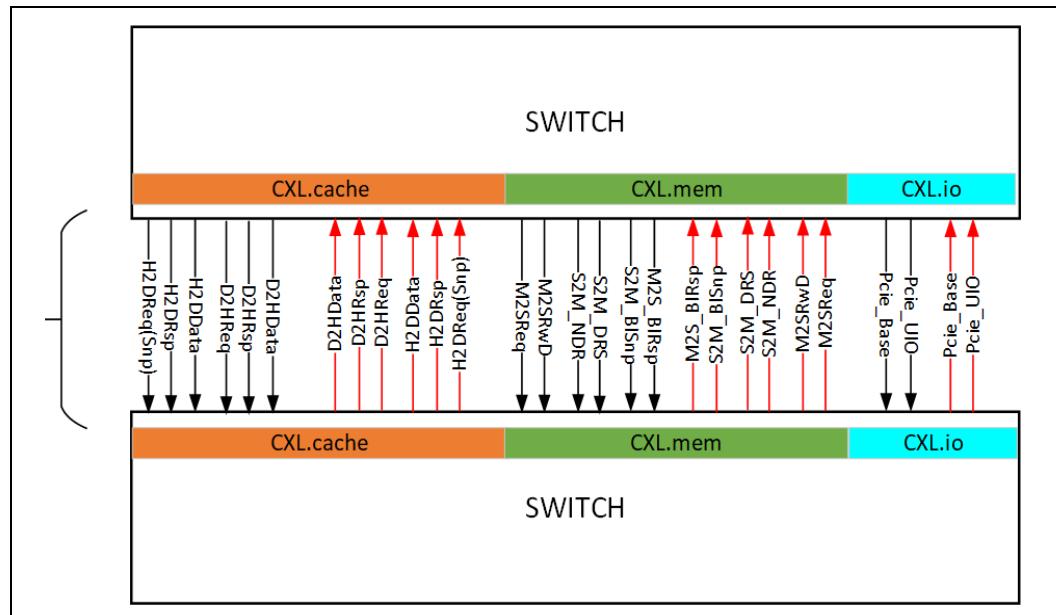
DPC for both vDSP and vUSP is supported for all fields. The FM-owned DSP above an ISL must have DPC. DPC on the DSP above an ISL shall always be enabled by FM. DPC support is required to provide sufficient delay so that the various software entities — switch firmware, host software, fabric manager — are able to complete DPC event processing at their own pace.

7.7.7

Inter-Switch Links (ISLs)

Inter-Switch Links (ISLs) carry PBR-format flits and must support all message classes and associated sub-channels, including one UIO VC. It is also additionally required that these message classes come up enabled automatically at power on, including the default UIO VC (VC3).

Figure 7-47. ISL Message Class Sub-channels



7.7.7.1

.io Deadlock Avoidance on ISLs/PBR Fabric

ISLs and PBR switches carry CXL.io Upstream traffic and CXL.io Downstream traffic from different hosts in the same physical direction/queues. To avoid deadlocks, these two traffic types need to be kept independent on ISLs and internally through PBR switches. To assist in maintaining the required independence, each TLP inside the PBR fabric is tagged with a DSAR (Downstream Acceptance Rules) bit. Here are the rules for setting the value of the DSAR bit within the PTH:

- When an Edge DSP converts a received TLP from HBR to PBR format, the Edge DSP shall clear the DSAR bit
- When an Edge USP converts a received TLP from HBR to PBR format, the Edge USP shall set the DSAR bit
- When a Host ES vDSP forwards a TLP P2P, it shall set the DSAR bit
- When a GFD sends a TLP (which is always in PBR format), the GFD shall clear the DSAR bit
- When an Edge DSP above a GFD forwards a TLP to the GFD, the Edge DSP shall set the DSAR bit

For the remainder of this section, traffic with DSAR=0 is referred to as USAR (Upstream Acceptance Rules) traffic, and DSAR=1 traffic is referred to as DSAR (Downstream Acceptance Rules) traffic. On an ISL, this bit is carried in the PTH. Traffic within each VC is required to follow the ordering rules specified in [Table 7-107](#) and [Table 7-108](#).

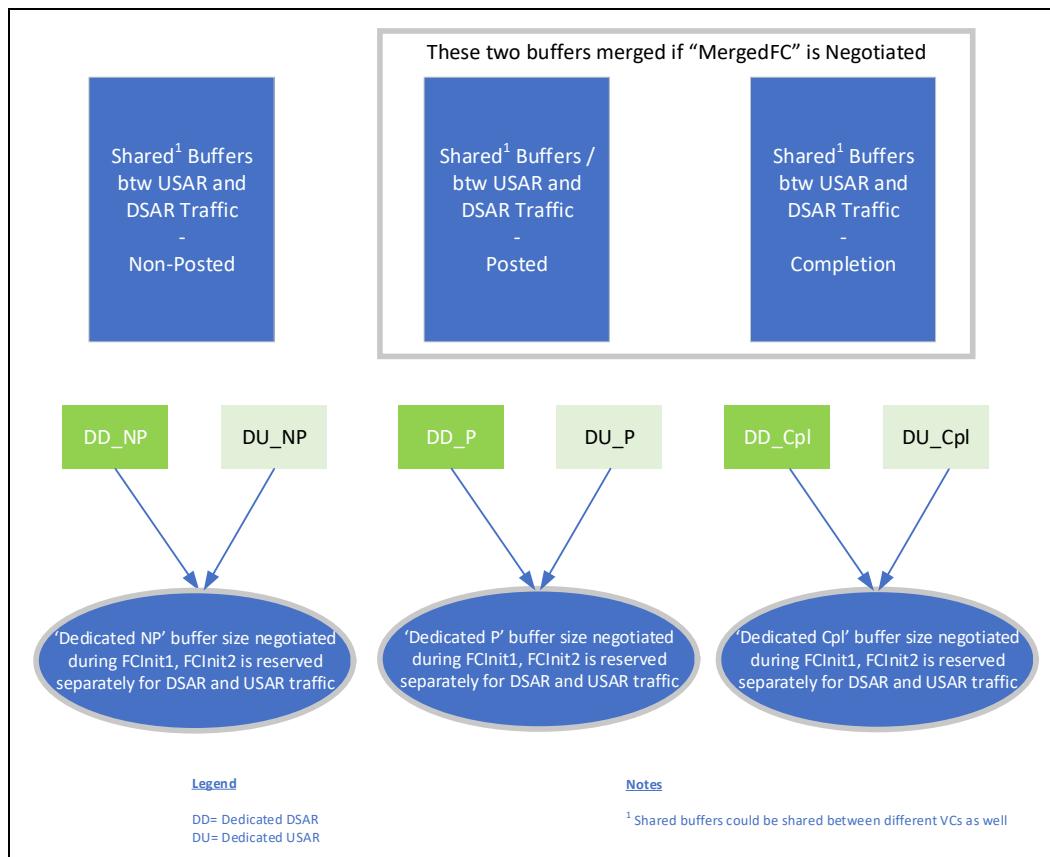
Table 7-107. PBR Fabric .io Ordering Table, Non-UIO

Row Pass Column?			DSAR			USAR								
			Posted Request	Non-Posted Request		Completion	Posted Request	Non-Posted Request		Completion				
DSAR	Posted Request			Read Request	NP Request with Data			Read Request	NP Request with Data					
	Non-Posted Request	Read Request	Per PCIe Base Specification						Yes	Yes	Yes	Yes		
		NP Request with data							Yes/No	Yes	Yes	Yes/No		
		Completion							Yes/No	Yes	Yes	Yes/No		
USAR	Posted Request			Yes/No	Yes	Yes	Yes/No	Per PCIe Base Specification						
	Non-Posted Request	Read Request	Yes/No	Yes/No	Yes/No	Yes/No								
		NP Request with data	Yes/No	Yes/No	Yes/No	Yes/No								
	Completion			Yes/No	Yes	Yes	Yes/No							

Table 7-108. PBR Fabric .io Ordering Table, UIO

Row Pass Column?			DSAR			USAR											
			UIO PR-FC TLP	UIO NPR-FC TLP	UIO Completion	UIO PR-FC TLP	UIO NPR-FC TLP	UIO Completion									
DSAR	UIO PR-FC TLP		Per PCIe Base Specification						Yes	Yes	Yes/No						
	UIO NPR-FC TLP								Yes	Yes	Yes/No						
	UIO Completion								Yes	Yes	Yes						
USAR	UIO PR-FC TLP		Per PCIe Base Specification	Yes/No	Yes/No	Yes/No	Per PCIe Base Specification										
	UIO NPR-FC TLP			Yes/No	Yes/No	Yes/No											
	UIO Completion			Yes	Yes	Yes/No											

To support the additional ordering requirements stated above, the following rules apply on ISL (also pictorially depicted in Figure 7-48):

Figure 7-48. Deadlock Avoidance Mechanism on ISL

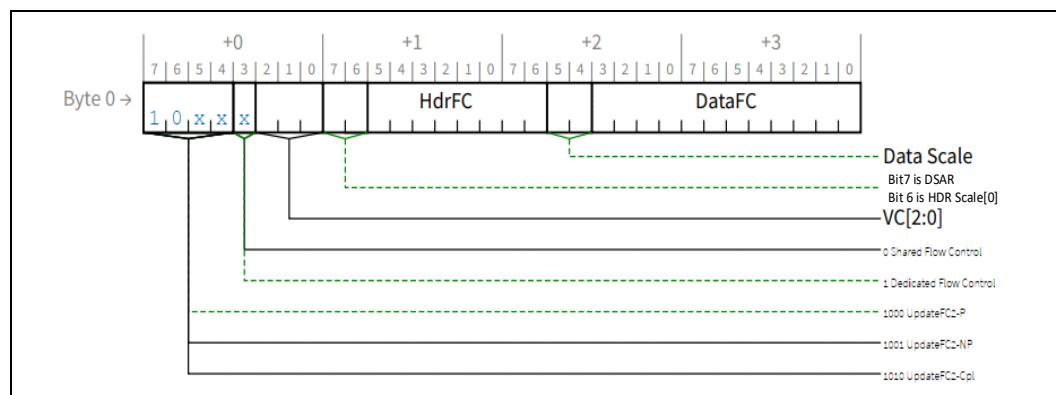
- PBR Fabric .io ordering rules apply independently within each VC implemented
- On edge HBR/PCIe links and on edge PBR links, PBR Fabric ordering rules do not apply
 - On edge PBR links, PTH bit can be ignored for ordering purposes and only the regular CXL.io ordering rules from PCIe Base Specification apply.
- Nonzero dedicated credits are always required on ISL for each VC, regardless of whether multiple VCs are enabled
- Baseline Shared and Merged FC initialization and usage rules, as described in PCI Base Specification, apply on ISLs as well, with some new rules/exceptions as noted below:
 - Dedicated buffers are required separately per FC class for DSAR and USAR traffic and they are both the same value as negotiated during FC initialization.
 - As an example, if one Posted HDR and one Posted DATA credit were exchanged for Dedicated buffers during InitFC1/2, the transmitter assumes there is 1 Posted data credit for DSAR traffic and one Posted data credit for USAR traffic and similarly for Posted HDR Credit as well.
 - Shared buffers can be shared between DSAR and USAR traffic.
- Update-FC DLLP is modified as shown in [Figure 7-49](#), to indicate release of DSAR or USAR buffers. Transmitters can use this information on shared credits to implement QoS limiting between DSAR and USAR traffic.
 - This modification is implicitly enabled on ISLs and requires no negotiation

Note:

To aid debug, Switches are recommended to capture the Hdr and data_Scale values negotiated at initialization so that debug software can access the values.

- Optimized_Update_FC DLLP applies to USAR traffic only and it is implicit on ISLs. All DSAR traffic's shared buffer credit return occurs only via Update-FC DLLP.

Figure 7-49. Update-FC DLLP Format on ISL



7.7.8

PBR TLP Header (PTH) Rules

For the purposes of this discussion, a "PBR link" is a link that negotiated to PBR flit format via the physical layer TS "PBR Flit bit" (see [Section 6.4](#)). See [Section 3.1.8](#) for details of PTH format.

- A PTH is inserted (via an appropriate decode mechanism) on CXL.io TLPs by an Edge Switch or the PTH is directly generated by devices (e.g., GFD) that natively support PBR link
- A PTH is forwarded as-is (unless explicitly noted otherwise as in handling PTH.DSAR bit on an edge PBR link) on a CXL.io TLP if the egress port is connected to a PBR link
- A PTH is removed when its CXL.io TLP exits to an edge non-PBR link
 - Note that some contents of PTH could be transferred to VendPrefixL0 if the egress port is an HBR link and the VendPrefixL0 is supported and enabled on the link. See [Section 7.7.3](#) and [Section 7.7.4](#).
- A PTH is included in link-IDE Integrity protection, if supported and enabled, when the PTH traverses PBR links.
- PTH is not included in .io selective IDE protection.

7.7.9

PBR Support for UIO Direct P2P to HDM

PBR switches support special routing mechanisms to enable the UIO Direct P2P to HDM use case with edge-to-edge routing, which often can be much more efficient compared to the hierarchical routing used in HBR switches. For backward compatibility, legacy software unaware of these special PBR routing mechanisms can continue to use HDM decoders, providing limited UIO Direct P2P capability.

An enhanced version of the FAST decoder as defined in [Section 7.7.2.4](#) can be implemented in the Edge DSP above the UIO requester, providing edge-to-edge routing for UIO requests that target GFDs.

Another instance of the FAST decoder hardware can provide edge-to-edge routing for UIO requests that target LD-FAM devices. This instance is referred to as an LD-FAM Segment Table (LDST), and it is usually configured with a different segment size and amount of mapped HDM space from any FAST decoders in use.

With LD-FAM devices, UIO completions can be routed edge-to-edge with an ID-Based Re-Router mechanism, which can be implemented in the Edge DSP above each LD-FAM device. The Re-Router matches against the destination ID of the UIO completion to determine the DPID for edge-to-edge routing. G-FAM devices automatically use edge-to-edge routing for UIO completions without this mechanism.

FAST decoders, LDST decoders, and ID-Based Re-Routers are each configured by host software using CCI command sets, as documented in [Section 7.7.15](#).

GFDs are not associated with any VH, thus they have no PCI ID (segment, bus, device, function number) assigned by any host. When a GFD sends a UIO completion, the completer segment field (if present) and the completer ID field in the completion are reserved and shall be 0.

7.7.9.1

FAST Decoder Use for UIO Direct P2P to G-FAM

FAST decoder instances in Edge USPs and DSPs have several similarities:

- Both convert requests from HBR format to PBR format, and route edge-to-edge to target GFDs.
- For the SPID, each uses the PID associated with its port.
- Both support CXL.mem and (CXL.io) UIO requests.
- A USP FAST decoder receives HBR format downstream requests coming from the RP. CXL.mem requests result from host accesses to GFDs.
- A DSP FAST decoder receives HBR format upstream requests coming from the requester device. UIO requests result from UIO Direct P2P traffic, where the UIO requester may be directly connected to an Edge DSP, or it may be connected via one or more HBR switches below the Edge DSP. CXL.mem requests result from the Direct P2P CXL.mem for accelerators use case, covered in [Section 7.7.10](#).

A DSP FAST decoder can be configured with a segment size different from the host's USP FAST decoder(s), but it is recommended for all FAST decoders to use the same segment size to avoid software complexity.

A DSP FAST decoder may need to be configured with a different number of segments from the host's USP FAST decoder(s) (e.g., a requester device may not need access to the entire Fabric Address space mapped by the USP FAST decoder). On the other hand, a requester device may need to access the Fabric Address space associated with an entire host Domain, not just a single RP within a host domain.

7.7.9.2

LDST Decoder Use for UIO Direct P2P to LD-FAM

LDST decoder instances in Edge USPs and DSPs have several similarities:

- Both convert requests from HBR format to PBR format, and route edge-to-edge to target LD-FAM devices.
- For the SPID, each uses the PID associated with its port.
- Both support CXL.mem and (CXL.io) UIO requests.
- A USP LDST decoder receives HBR format downstream requests coming from the RP. CXL.mem requests result from host accesses to LD-FAM devices. UIO requests currently have no architected use cases, but they are not prohibited.
- Host software determines whether host accesses to LD-FAM devices use LDST decoders versus HDM Decoders in Edge USPs. For backward compatibility, legacy software that's unaware of LDST decoders can continue to use HDM decoders. For overcoming scaling limitations with the number of HDM decoders supported by Edge USPs, LDST-aware software can use LDST decoders, though LDST decoders do not support HDM-D.

- A DSP LDST decoder receives HBR format upstream requests coming from the requester device. UIO requests result from UIO Direct P2P traffic. CXL.mem requests result from the Direct P2P CXL.mem for accelerators use case, covered in [Section 7.7.10](#).

A DSP LDST decoder can be configured with a segment size different from the host's USP LDST decoder(s), but it is recommended for all LDST decoders to use the same segment size to avoid software complexity.

A DSP LDST decoder may need to be configured with a different number of segments from the host's USP LDST decoder(s) (e.g., a requester device may not need access to the entire LD-FAM HDM space mapped by the USP LDST decoder). On the other hand, an accelerator may need to access the LD-FAM HDM space associated with the entire host Domain, not a single RP in the host Domain.

When any LDST decoders are in use, host SW needs to configure any HDM decoders mapping the same LD-FAM HDM ranges with decoder characteristics compatible with LDST decoders. This applies to HDM decoders present in the host, PBR switches, HBR switches, or LD-FAM devices. These decoder characteristics include:

- Minimum decoder granularity: 64 GB for LDST
- Interleave Ways (IW): neither HBR nor PBR switches have the special logic required to support 3/6/12, but LDST supports the other architected IW values.

Note that Dynamic Capacity (DC) Block Sizes are not visible to either type of decoder.

LDST decoders insert a requester segment field in UIO requests when necessary. This is described in [Section 7.7.9.3](#).

7.7.9.3

ID-Based Re-Router for UIO Completions with LD-FAM

For UIO Direct P2P to LD-FAM devices, UIO completions by default are routed using hierarchical PCI ID-based routing, and the ID may include a PCI segment number in addition to bus, device, and function numbers. If present in the Edge DSP above an LD-FAM device, the ID-Based Re-Router does a CAM match using the PCI ID, returning the DPID needed for edge-to-edge routing. This mechanism efficiently handles intra-VH cases, and it is especially efficient for cross-VH cases by avoiding P2P through the Root Complex.

PCI segment numbers in TLPs is a feature added in PCI Express Base Specification 6.0, and PCI segments should not be confused with "segments" in the context of FAST/LDST decoders. LDST decoders support the PCIe convention that requesters generally don't include PCI segment numbers in requests¹ but rely instead on routing mechanisms to add PCI segment number fields when needed for cross-segment routing. Host software configures LDST decoders to add² the requester segment field in the request when it targets a different PCI segment. When the LD-FAM device responds to the UIO request with a UIO completion, it automatically includes segment fields when necessary in the Destination ID and Completer ID. Host software shall configure the ID-Based Re-Router with the PCI segment number in entries that need it.

-
1. With Selective IDE non-configuration requests, the requester is required to include the requester segment field in the request because a routing element inserting the field would cause an integrity violation with Selective IDE.
 2. Although PCIe Base Specification forbids PCIe switches from inserting a Requester Segment field, the CXL UIO Direct P2P to HDM mechanisms in CXL switches are beyond the scope of PCIe Base Specification and do not violate the underlying architecture principles.

7.7.9.4

LDST and ID-Based Re-Router Access Protection

LDST and ID-Based Re-Router use is protected by the LDST Access Vector (LAV) to ensure that only valid PIDs are programmed by the host into the LDST and ID-Based Re-Router structures. The LAV is a 4k-bit vector with a similar functionality as the GMVs and VTVs.

The FM is responsible for enabling access to PIDs in the LAV before the host can program those PIDs into the LDST or ID-Based Re-Router structures. For cross-VH use cases, the FM is also responsible for using the Domain Validation SV mechanism, when available, to confirm that every VH that is enabled for cross-VH access belongs to the same host domain.

7.7.10

PBR Support for Direct P2P CXL.mem for Accelerators

Direct P2P CXL.mem provides the ability for an accelerator to access peer Type 3 memory devices using CXL.mem. PBR switches require special routing mechanisms to support this, specifically the FAST and LDST decoders. For Direct P2P CXL.mem, these decoders function essentially the same as they do for supporting the UIO Direct P2P to HDM use case, with the following exceptions:

- They intercept and forward upstream.mem requests instead of UIO requests
- They target only Type 3 (HDM) devices, not Type 2 devices
- The accelerator (requester device) and Type 3 device must each be directly connected to an Edge DSP
- With an MLD (Type 3 device), each accelerator must be assigned a dedicated LD distinct from the host's LD

Note that both types of decoders support .mem requests when they are implemented in Edge USPs, so .mem support is not unique to the Direct P2P CXL.mem use case.

Same as with the UIO Direct P2P use case, a FAST decoder can be implemented in the Edge DSP above an accelerator, providing edge-to-edge routing for .mem requests that target G-FAM devices (GFDs). The same FAST decoder instance can support either the UIO Direct P2P or Direct P2P CXL.mem use case.

Similarly, an LDST decoder can be implemented in the Edge DSP above an accelerator, providing edge-to-edge routing for .mem requests that target LD-FAM devices. The same LDST decoder instance can support either the UIO Direct P2P or Direct P2P CXL.mem use case.

Type 3 devices used with Direct P2P CXL.mem can be mapped under either HDM-H or HDM-DB coherency ranges. If mapped under HDM-DB, peer devices other than the associated accelerator can access the HDM-DB memory using UIO Direct P2P to HDM, in which case the associated accelerator serves the role of the host participating in BI protocol (i.e., the HDM-DB device directs BISnps to the accelerator).

Direct P2P CXL.mem traffic going to or from an MLD (directly connected to an Edge DSP) works essentially the same as with host .mem traffic, as documented in [Section 7.7.6.6.3](#) and [Section 7.7.6.8](#).

CXL.mem responses for the Direct P2P CXL.mem use case require no special routing mechanism. For S2M responses from G-FAM, the GFD's RPID context for the accelerator contains the DPID needed for edge-to-edge routing back to the accelerator. For S2M responses from LD-FAM, the vPPB in the Edge DSP above the Type 3 device contains the DPID needed for edge-to-edge routing back to the accelerator.

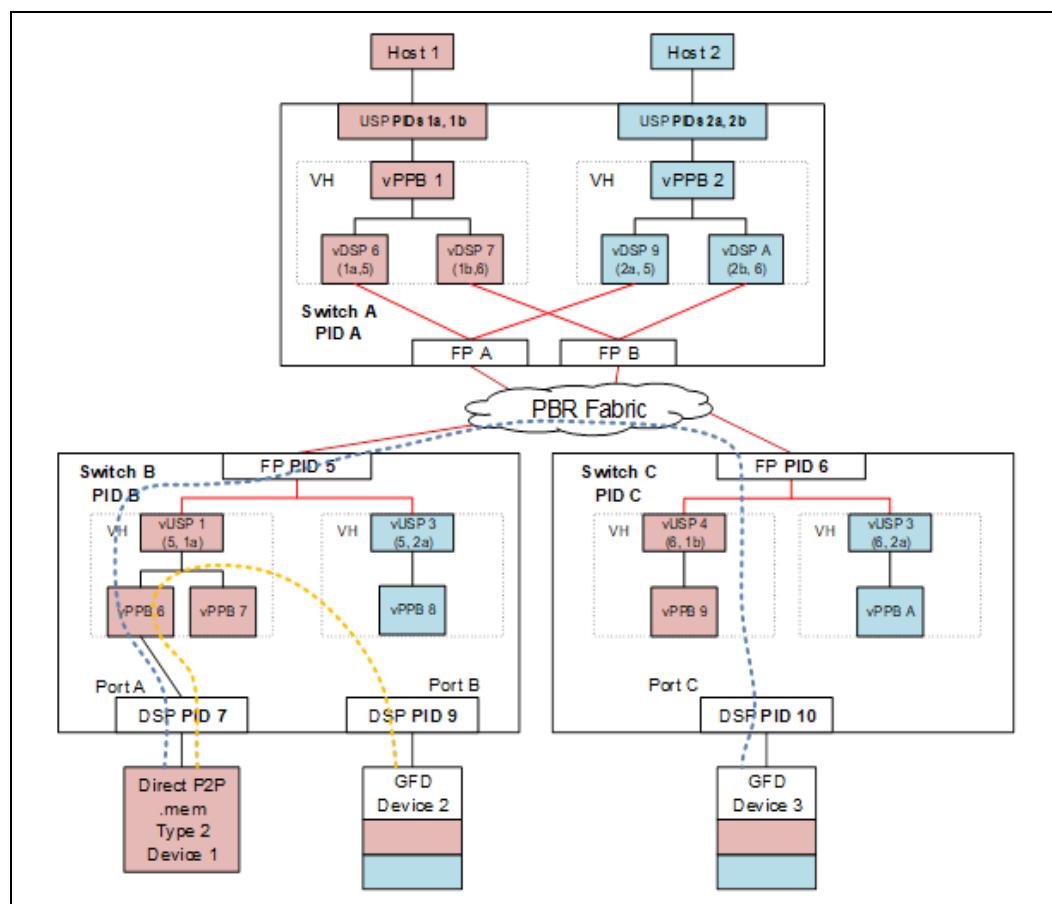
Same as with the UIO Direct P2P use case, FAST decoders and LDST decoders are each configured by host software using CCI command sets, as documented in [Section 7.7.15](#) for FAST decoders and [Section 7.7.13](#) for LDST decoders.

7.7.10.1 Message Routing for Direct P2P CXL.mem Accesses with GFD

Direct P2P CXL.mem messages are routed using standard PBR mechanisms. [Figure 7-50](#) illustrates an example PBR Fabric with a Direct P2P CXL.mem enabled Type 2 accelerator and two peer GFDs accessible to it. The dashed lines indicate the paths taken by the Direct P2P CXL.mem messages. Upstream .mem requests from the accelerator are routed edge-to-edge to the appropriate GFD by the FAST decoder in vPPB 6. Upstream .mem responses from either GFD are routed edge-to-edge back to the accelerator by standard PBR routing.

For an HDM-DB GFD sending a BISnp, the GFD's RPID context for the accelerator contains the DPID that is needed for edge-to-edge routing to the accelerator.

Figure 7-50. Example Topology with Direct P2P CXL.mem with GFD

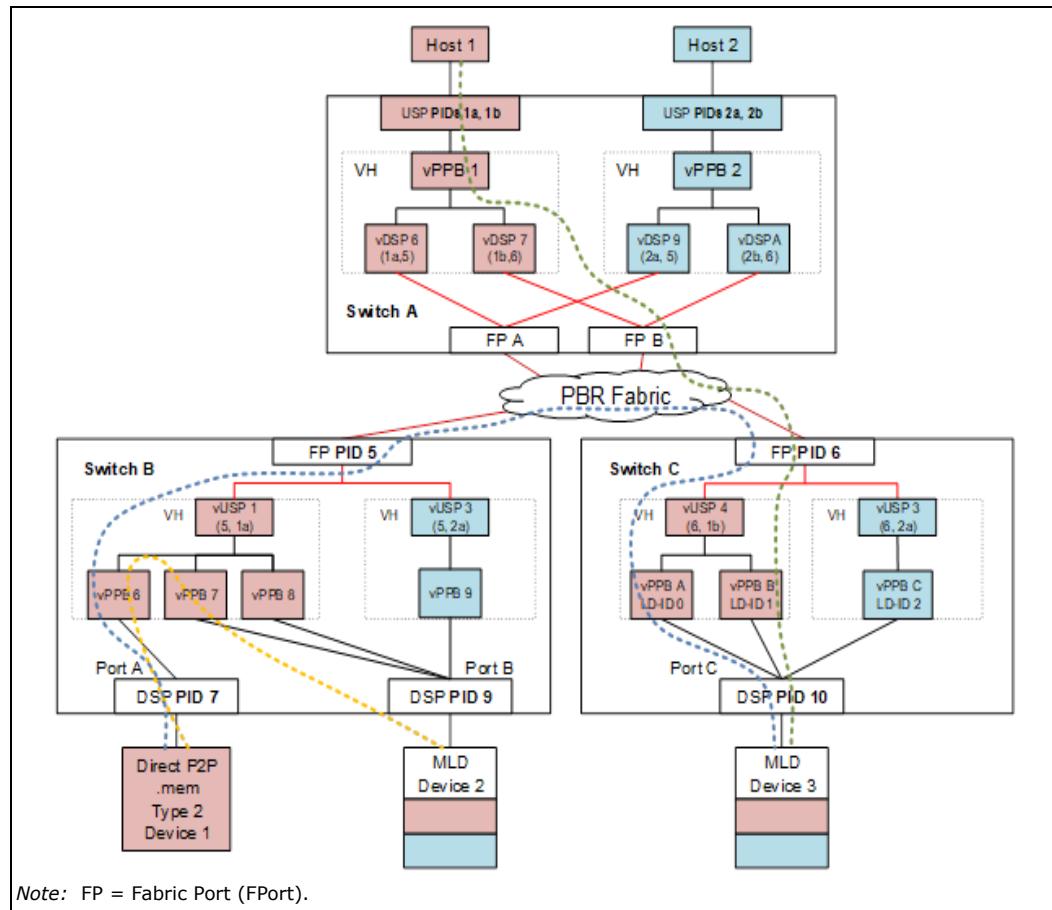


7.7.10.2 Message Routing for Direct P2P CXL.mem Accesses with MLD

Direct P2P CXL.mem accesses to an MLD require a distinct LD and associated peer requester LD-ID on the link between the MLD and the Edge DSP to which it is attached. This is accomplished by assigning a vPPB in the DSP in the same Domain as the host that owns the requester. The host and any peer accelerators will each have their own vPPB bound to them, which utilize their individual LD-IDs.

Figure 7-51 illustrates an example PBR Fabric with a Direct P2P CXL.mem enabled Type 2 accelerator and two peer MLDs accessible to it. The dashed lines indicate the paths taken by the Direct P2P CXL.mem messages. Upstream CXL.mem requests from the accelerator are routed edge-to-edge to the appropriate MLD by the LDST decoder in vPPB 6. Upstream CXL.mem responses from either MLD are routed edge-to-edge back to the accelerator by standard PBR routing using the accelerator's PID, which in each case is retrieved from the accelerator's vPPB in the DSP above the MLD.

Figure 7-51. Example Topology with Direct P2P CXL.mem with MLD



In this example, the path taken by CXL.mem messages between the host and one MLD is also shown. Downstream CXL.mem requests from the host are routed edge-to-edge to the appropriate MLD by the LDST decoder in vPPB 1. Upstream CXL.mem responses from the MLD are routed edge-to-edge back to the host by standard PBR routing using the host's PID contained in vPPB B.

For an HDM-DB LD-FAM device sending a BISnp, the Edge DSP above the LD-FAM device contains the DPID that is needed for edge-to-edge routing to the accelerator.

7.7.11

PBR Link Events and Messages

A PBR link can carry traffic from many different VH at the same time. Some events may occur that only affect a single VH, while other events need to apply to all VH sharing the link.

Basic PBR link requirements are discussed in [Section 7.7.11.1](#).

A summary of all the CXL Vendor Defined Messages (VDMs) that are PTH routed to the destination is provided in [Section 7.7.11.2](#).

PCIe events for a single VH are discussed in [Section 7.7.11.3](#).

PCIe events for multiple VH sharing a link are discussed in [Section 7.7.11.4](#).

Events that occur outside PCIe are discussed in [Section 7.7.11.5](#).

Messaging to and from a host to a GFD is discussed in [Section 3.1.11.1](#).

7.7.11.1

PBR Link Fundamentals

CXL defines two types of PBR links:

- Inter-Switch Link (ISL)
- GFD link

All PBR links must support PBR Flit mode. Because PBR Flit mode relies on PCIe Flit mode, all host-OS-visible DSPs should report PCIe Flit mode as enabled. The DSPs include both a Host Edge Switch vDSP and a DSP above a PBR link that leads to a GFD.

The owner of a PBR link is an FM-managed DSP. Switch firmware may assist the FM in managing the DSP. An ISL is a downstream-to-downstream crosslink and thus has an FM-managed DSP on each side of the link. A GFD link has only one DSP and thus has only one FM-managed DSP. The speed and width of a PBR link is solely controlled by the FM-managed DSP(s) on the link and not by any vDSPs that share the link.

Each side of an ISL is managed separately. Each DSP above an ISL must support DPC, to allow firmware on each side of the link an independent amount of time to process fabric port events. DPC shall be enabled for all cases on ISL except when the ISL is the only path to the FM, in which case the DSP furthest from the FM shall not have DPC enabled.

FM-initiated CXL.io traffic sent across a PBR link shall be limited to DMTF-format VDMs. The PTH.DPID is used to indicate whether the PBR Link Partner should sink the TLP or forward the TLP. If the PTH.DPID = FFFh, the PBR Link Partner must sink the VDM because that is how the initial device discovery occurs and how PIDs are assigned. If the PTH.DPID = the device's PID, then the device must also sink the VDM because that is how the device is accessed by the FM.

All VH users of a PBR link have their functionality ride on top of the FM-managed link. For example, a VH's DSP cannot see a Link Up if the fabric link is not up. A VH cannot change the width or speed of its shared link, rather it will inherit the setting of the FM-managed DSP.

To manage different software response times to events, every vDSP for every VH must support DPC. DPC allows a host to keep its Link Down from its (VH) perspective until it is ready to re-enable it, having cleaned up all the side effects of a Link Down. A Host may or may not choose to enable DPC.

L0p is optional on a PBR link. The FM-managed DSP initiates any L0p transitions via a mechanism that is beyond this specification.

Every CXL.io TLP on a PBR link will carry a 4B PTH. The VDMs described in this section follow the same rule. See [Section 3.1.8](#). There are three fields of note in the PTH that are required for the VDMs described in this section:

- SPID: Source PID
 - From a vDSP: Use vDSP's USP PID
 - From a vUSP: Use vUSP's FPort PID

- From a switch: Use switch's PID
- From a downstream edge: Use DSP's PID
- From a host edge: Use USP's PID
- DPID: Destination PID
 - To a vDSP: Use vDSP's USP PID
 - To a vUSP: Use vUSP's FPort PID
 - To a switch: Use switch's PID
 - To a downstream edge: Use DSP's PID
 - To host edge: Use USP's PID
- DSAR flag

7.7.11.2 CXL VDMs

See [Section 3.1.11](#) for a list of VDMs that are used in the PBR fabric.

7.7.11.3 Single VH Events

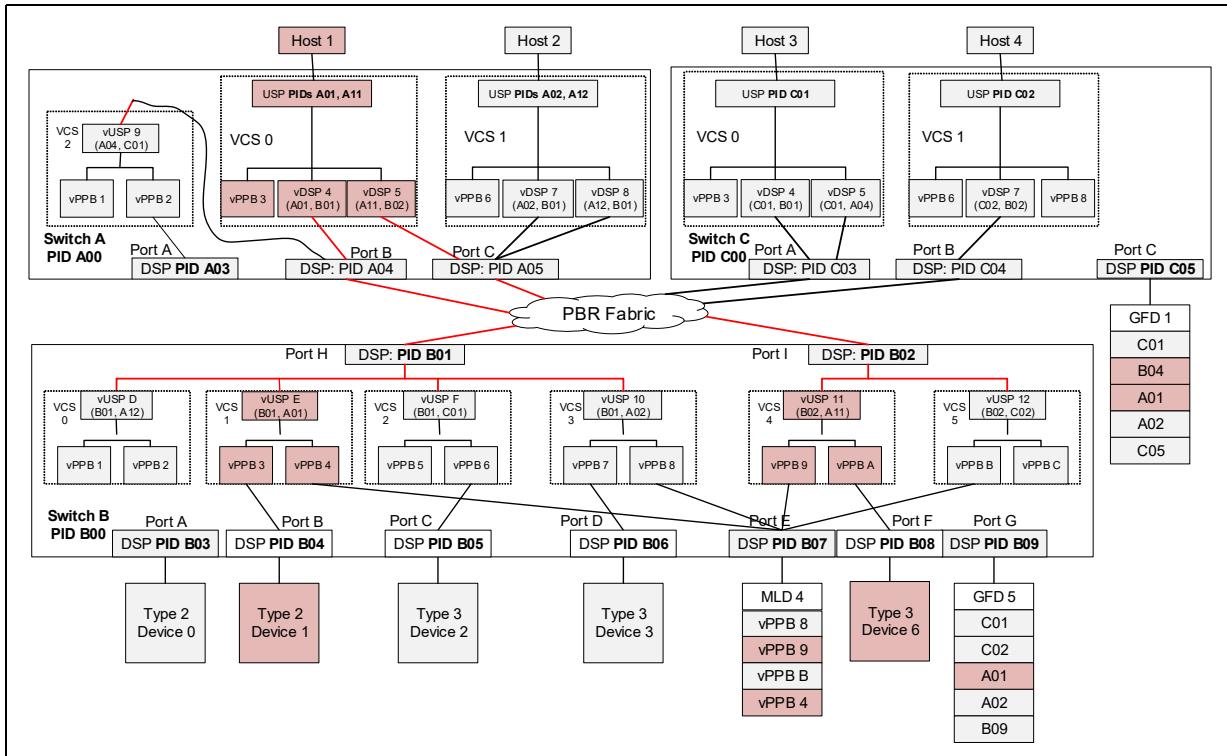
Events that are contained within a single VH should not affect other VHs that share an ISL.

PCIe visible events that are contained within a single VH include:

- Assert Reset
- Deassert Reset
- Link Up

[Figure 7-52](#) shows the virtual hierarchy from a Host 1 perspective (other hierarchies are grayed out). In Switch A, Host 1 finds only a single switch VCS 0. However, in Switch B, two switches VCS 1 and VCS 4 are in the Host 1 hierarchy. VCS 1 has Switch B vUSP E connected below Switch A vDSP 4, and VCS 4 has Switch B vUSP 11 below Switch A vDSP 7. Switch C has a GFD with that is accessible by Host 1 devices, but the GFD is not visible to the Host 1 PCI hierarchy. See [Section 7.7.14](#) for more details on control of the GFD.

Figure 7-52. Single VH



7.7.11.3.1 Assert Reset VDM

Every PCIe hierarchy supports three levels of Conventional Reset:

- Fundamental cold reset (PERST#): Input pin
- Fundamental warm reset (PERST#): Input pin
- Hot reset due to Link Down, in-band hot reset, USP secondary bus reset, DSP secondary bus reset, or link disabled

CXL Fabric links support propagation of these resets. The ISL link state is not affected by any VH's Assert Reset or Assert PERST# VDM. Assertion of reset is accomplished using one of two different VDM opcodes:

- **Assert PERST#:** Used for fundamental reset assertion for that VH, Opcode 0
- **Assert Reset:** Used for hot reset assertion for that VH, Opcode 1

The separate PERST# message allows for fundamental reset functionality without the need for extra pins between switches.

Assert PERST# should be triggered whenever a VH has its input fundamental reset asserted on a Host ES. Assert Reset should be triggered whenever the Host ES:

- Receives a hot reset input
- Has a secondary bus reset on its USP
- Has a secondary bus reset on its VDSP
- Has a link disable on its vDSP

The Assert Reset VDMs all are sent from a vDSP to its paired vUSP. The VDM sent will have a PTH with:

- SPID = vDSP's host PID
- DPID = vUSP's FPort PID
- DSAR flag = 1

VDM header fields for Assert Reset VDMs:

- CXL VDM code of 80h
- PBR Opcode 0 or 1 indicates which Assert PERST# or Assert Reset message

It is expected that the Assert Reset VDM will reach a vUSP uniquely identified by the SPID and DPID at the destination switch.

A vDSP, upon sending Assert Reset VDM, will have its link state transition to Hot Reset.

A vUSP, upon receiving an Assert Reset VDM, will have its link state transition to Hot Reset. While in Hot Reset, all Port non-sticky registers and state machines that belong to the VH must return to their initialized state.

A vUSP, upon receiving an Assert PERST# VDM, shall have its link state transition to Hot Reset and also shall clear any sticky bits as outlined by PCIe Base Specification for PERST# behavior.

It is possible to send any number of Assert Reset VDMs or Assert PERST# VDMs.

In [Figure 7-53](#), if Host 1 asserts its PERST#, then both vDSP 4 and vDSP 5 shall issue an AssertPERST# VDM. The format of the PTH would be (SPID=A01, DPID=B01) for vDSP 4 and (SPID=A11, DPID=B02) for vDSP 5. If Host 1 instead asserted vDSP 4 secondary bus reset, then only vDSP 4 would send an AssertReset VDM with (SPID=A01, DPID=B01).

7.7.11.3.2 Deassert Reset VDM

A Deassert Reset VDM signals a release of reset and an exiting of the Hot Reset state to enter Detect for that VH. This VDM shall be sent from the Host Edge Switch due to a deassertion of the PERST# input resulting from an exit from Hot Reset.

If DSP is enabled the DPC trigger status must be cleared before a Deassert Reset VDM can be sent because DPC triggered prevents any TLPs from egressing that port.

Propagation of reset deassertion over an ISL is enabled via a Deassert Reset VDM, which is used for hot reset deassertion for that VH, Opcode 3.

A Deassert Reset VDM is used to instruct the vUSP to exit Hot Reset and enter Detect. The Deassert Reset VDM sent will have a PTH with:

- SPID = vDSP's host PID
- DPID = vUSP's FPort PID
- DSAR flag = 1

VDM header fields for Deassert Reset VDMs:

- CXL VDM code of 80h
- PBR Opcode 3

A vDSP, upon sending a Deassert Reset VDM, will have its link state transition from Hot Reset to Detect. A vUSP, upon receiving a Deassert Reset VDM, will have its link state transition from Hot Reset to Detect. If the link state is not in Hot Reset, a link state change will not occur.

The link for that VH will remain in Detect until the vUSP sends a Link Up VDM and the vDSP receives a Link Up VDM. If a Link Up VDM is not received within 10 ms, a subsequent Deassert Reset VDM shall be sent. This can repeat until 10 Deassert Reset VDMs have been sent. After a tenth Deassert Reset VDM is sent, if a Link Up VDM is still not received within 10 ms, the reset deassertion failed and the FM shall be notified.

In [Figure 7-53](#), if Host 1 clears the secondary bus reset in Switch A vDSP 4, then vDSP 4 would send a Deassert Reset VDM with (SPID=A01, DPID=B01). Switch B vUSP E would exit the hot reset state. As part of the exit from LTSSM Detect and due to the shared link nature of an ISL, vUSP E will bypass the PCIe LTSSM states of Polling and Configuration and transition the vDSP-to-vUSP link back to L0 (Link Up) by sending a Response Link Up VDM.

7.7.11.3.3 Link Up VDM

A Link Up VDM signals a transition to L0 active for that VH's link. The Link Up VDM is sent by a vUSP to its paired vDSP to convey a post-Detect state across the shared ISL.

The vUSP sends a Link Up VDM after receiving a Deassert Reset VDM. The vUSP can perform any required post-reset initialization before sending the Link Up VDM. The vUSP may take as long as it needs after Deassert Reset to send the Link Up VDM. Any number of Deassert Reset VDMs may be received by the vUSP; for each Deassert Reset VDM received, a Link Up VDM shall be sent.

The vUSP, after sending a Link Up VDM, shall have its link state transition to L0 from Detect. Polling and Configuration link states are bypassed by the Link Up VDM because the required TS1 and TS2 Ordered Sets cannot be sent over a shared ISL.

A vDSP, after receiving a Link Up VDM, shall have its link state transition to L0 from Detect. If not in Detect, there is no state change. Any number of Link Up VDMs may be received. Polling and Configuration link states are bypassed by the Link Up VDM, with the link directly transitioning from Detect to L0.

Neither a vDSP nor vUSP should ever have their link state reach Polling or Configuration state.

The VDM sent will have a PTH with:

- SPID = vUSP's FPort PID
- DPID = vDSP's host PID
- DSAR flag = 1

VDM header fields for LinkUp VDMs:

- CXL VDM code of 80h
- PBR Opcode 4

7.7.11.3.4 Dynamic vDSP-to-vUSP Bind

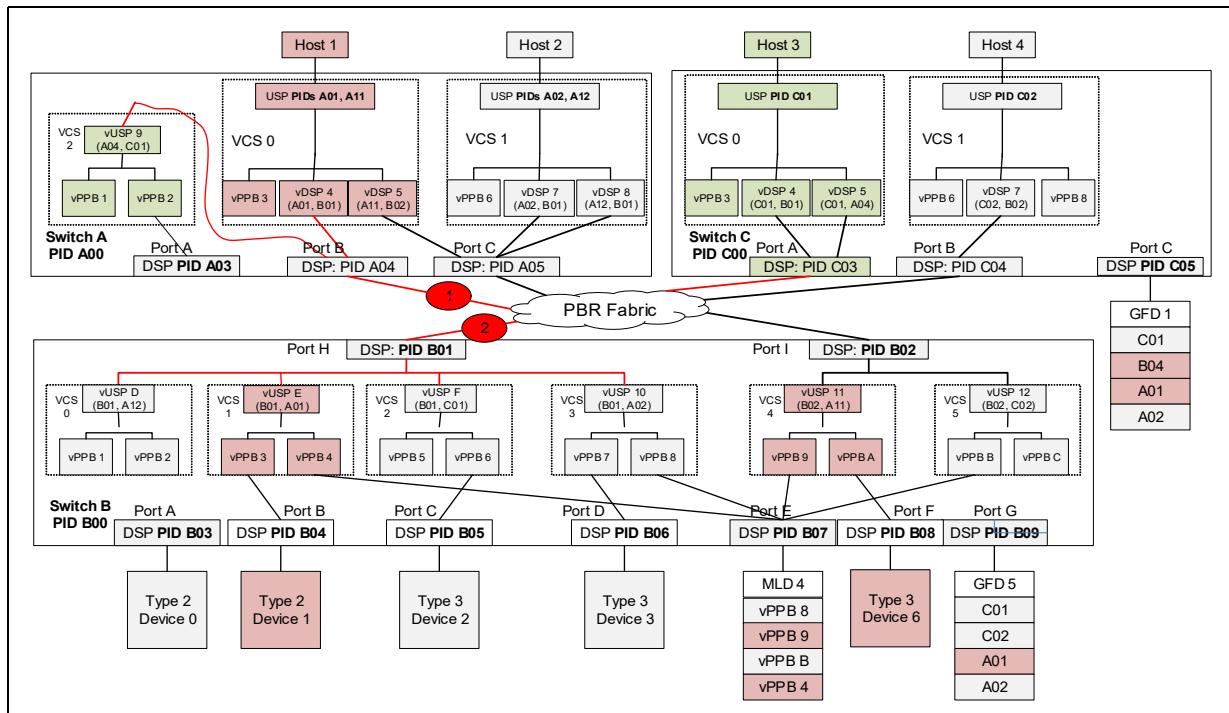
See [Section 7.7.12.3](#) for more details on the Configure PID Binding API sequence. After Configure PID Bind, the vDSP or vUSP shall be in a Hot Reset state. A vDSP may issue an Assert Reset VDM or a Deassert Reset VDM from the reset state, as dictated by its VH. A vUSP shall remain in Hot Reset until the vUSP receives a Deassert Reset VDM, upon which, after processing the necessary post-reset tasks, the vUSP will send a Link Up VDM.

7.7.11.4 Shared Link Events

Events that affect multiple VHs on the same link need to be reported to the FM. The FM shall take any necessary action.

The FM is required to keep an inventory for each ISL. Figure 7-53 shows how the link from Switch A Port B (indicated by an oval with 1) is shared by both a Host 1 hierarchy and a Host 3 hierarchy. Events on this link will affect both hierarchies. The oval with 2 is another shared link used by multiple hierarchies, of which only a Host 1 hierarchy is colored in but the ISL also includes Host 3 (VCS 2) and two hierarchies of Host 2 (VCS 0 and VCS 3).

Figure 7-53. Shared Link Events



7.7.11.4.1 Inter-Switch Link (ISL) Down

An ISL going down may affect one or more VHs.

A switch on each side of the ISL knows if the link had any issues. The fabric port's DPC is used to handle link issues. If DPC triggers, switch firmware will be notified. DPC may trigger due to Link Down or due to other reasons, such as software trigger; the net result is that the ISL will go down. Once the link goes down the switch reports the event to its primary FM. The FM is responsible for resolving the ISL Down event for all involved VHs.

The fabric port's DPC should remain triggered until switch firmware can resolve the side effects of an ISL Down event. When the FM has finished its resolution tasks, the FM will instruct the switch to clear the DPC trigger on the fabric port DSP. DPC trigger clear indicates resolution of the event and also allows the ISL to come back up.

The FM requires an inventory of users of an ISL to correctly resolve an ISL Down event. FM tasks for the resolution of an ISL Down event involves the following:

- Unbinding any affected VHs' vDSP

- Unbinding any affected VHs' vUSP
- Clearing any affected multi-path in a switch's RGT
- Clearing any affected GFD Access Vector in a switch's GAE

For example, if the link at Oval #1 in [Figure 7-53](#) breaks, Switch A and an unlabeled PBR fabric switch will both notify their primary FM. The FM will then unbind the following affected vDSPs and vUSPs:

- Switch A vDSP 4 and vUSP 9
- Switch B vUSP E
- Switch C vDSP 4

As another example, if the link at Oval #2 in [Figure 7-53](#) breaks, Switch B and an unlabeled PBR fabric switch will both notify their primary FM. The FM will then unbind the following affected vDSPs and vUSPs:

- Switch A vDSP 4 and vDSP 8
- Switch B vUSP D, vUSP E, vUSP F, and vUSP 10
- Switch C vDSP 4 and vDSP 7

In addition to the unbinding of the vDSP and vUSP pair affected by an ISL Down event, the RGT and GAE GFD access vectors may be updated by the FM. The RGT would be updated to avoid the path leading to the fault. The GFD Access Vector may be updated to remove a GFD that is no longer reachable.

7.7.11.5 Switch Reported Events

Some events are switch specific or are outside normal PCIe reporting methods and thus require switch-specific intervention. These include:

- Link Partner Info

7.7.11.5.1 Link Partner Info VDM

A Link Partner Info VDM is sent on all PBR links immediately after the InitFC process finishes for VC0. Each side of the link will send a Link Partner Info VDM. This is a message with payload. For CXL 3.1, the payload is a fixed size of 16 DWORDs.

There are two types of PBR links: ISL and GFD. Both have same Link Partner Info, but have a different value for the type.

The Link Partner Info payload includes the following fields about the far end of the link:

- 16B Device UUID
- 1B Physical Port
- 2B {4b Device Type (0 = PBR switch, 1 = GFD, all other encodings are reserved), 12b PID (0xFFFF is initialized)}
- 1B Standard FC VC list, default {8'bxxxx_0xx1}
- 1B UIO FC VC list, default {8'bxxxx_Pxx0}
- 16B FM Primary UUID
- 16B FM Secondary UUID

Table 7-109. Link Partner Info Payload

Byte	+3								+2								+1								+0																					
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0														
0	DevType[3:0]				PID[11:0]								Far End PortID[7:0]								PortID[7:0]																									
4	Device UUID[127:0]																																													
8																																														
12																																														
16																																														
20	Primary FM UUID[15:0]												UIO FC VC List[7:0]				Standard FC VC List[7:0]																													
24	Primary FM UUID[111:16]																																													
28																																														
32	Secondary FM UUID[15:0]												Primary FM UUID[127:112]																																	
36																					Secondary FM UUID[111:16]																									
40																																														
44																																														
48																					Secondary FM UUID[127:112]																									
52																																														
56																																														
60																					Reserved																									

The Link Partner Info VDM.PTH fields are as listed below. This VDM will terminate at the Receiver.

- SPID = Originator's (switch's/GFD's) PID, FFFh means uninitialized
- DPID = FFFh (not used)
- DSAR flag = 1

VDM header fields for LinkPartnerInfo VDMs:

- Type 74h (Message with Data, terminate at Receiver)
- CXL VDM code of 90h
- PBR Opcode 0

A single message is sufficient to carry all the link info for CXL r3.1.

An ISL must have UIO FC VC list bit 3 set.

If both sides of a PBR link have UIO FC VC list bit 3 set, then UIO VC3 should be auto enabled by hardware.

7.7.11.6 PBR Link CCI Message Format and Transport Protocol

CCI commands are transported on PBR links as defined in [Section 7.6.3](#) and its associated binding specifications (see DSP0234, DSP0238, and DSP0281) with some notable caveats and clarifications:

- As with all .io traffic across PBR links, MCTP PCIe VDMs include a PTH whose SPID and DPID define the routing of the message
- PCIe enumeration is not required for ISL PPBs and GFDs
- GFDs do not implement a PCIe Physical Function

- “Requester ID” and “Target ID” fields in the VDM’s TLP header are reserved because IDs are not assigned to many elements within the fabric (e.g., FM, ISL PPBs, Switch Management FW, GFDs, etc.)

7.7.12 PBR Fabric Management

7.7.12.1 Fabric Boot and Initialization

Much like as outlined for HBR switches in [Section 7.2.1](#), PBR switches may be initialized in one of three different ways:

- Statically
- FM boots before the host(s)
- FM and host boot simultaneously

7.7.12.1.1 Static Fabric Initialization

A static fabric deployment uses statically predefined configuration data to define the fabric configuration settings typically assigned dynamically by an FM.

Static Fabric Characteristics:

- No support for G-FAM or MLD
- No support for dynamic binding changes or DCD
- No FM is required, but may be needed for error handling
- At switch boot, all ports have a PID assigned, DRT and RGT tables are pre-populated, and EP and PID binding settings are predefined as defined by vendor-specific switch configuration data (e.g., configuration file in SPI Flash)
- Each VH is ready for enumeration when the host boots
- Hot-add and managed hot-remove are supported on Downstream Edge Ports

7.7.12.1.2 Fabric Manager Boots First

With this method, the FM configures the fabric binding relationships and access permissions before the host boots and enumerates its VH.

- FM boots while hosts are held in reset
- All attached ISLs and DSPs link up and, when negotiated in PBR mode, exchange the PBR Link Information VDM
- FM discovers fabric topology, claims ownership of all components under its management, and assign PIDs
- FM binds EPs to VCSs and configures GFDs
- FM configures GMV and VTV to enable G-FAM, GIM and Edge-to-edge P2P, as required when available

7.7.12.1.3 Fabric Manager and Host Boot Simultaneously

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

- VCSs, PID assignment, GFD configuration, and bindings between Host ES to Downstream ES VCSs are statically defined
- Edge vPPBs within each VCS are unbound and presented to the host as Link Down
- Switch discovers downstream devices and presents them to the FM
- Host enumerates the VH and configures the DVSEC registers

- FM performs port binding to edge vPPBs
- Switch performs virtual to physical binding
- Each bound port results in a Presence Detect Change or Link State Change notification to the host
- For G-FAM access, FM updates GMV and VTV access vectors for hosts

7.7.12.2 PBR Fabric Discovery

To effectively manage a PBR fabric, the FM must understand the physical topology through a fabric discovery process. A typical fabric discovery may proceed as follows.

1. FM discovers the component to which it is directly connected and claims primary FM ownership.

Management of a PBR device requires that a primary FM is registered. A PBR device shall accept only the following commands from an FM that is not registered as the primary FM:

- **Identify**
- **Get Supported Logs**
- **Get Log**
- **Identify PBR Component**
- **Claim Ownership**

All other commands shall fail with "Unsupported Request". A PBR device shall only advertise support for the CEL and the CEL shall only advertise the commands in the above list when the supported logs or CEL contents are queried by an FM that is not registered as the primary FM.

If the FM is connected to a switch, crawl out and discovery of the fabric continues.

2. FM explores all switch ports.

As primary FM, the switch capabilities and switch port status can be queried. The **Get Physical Port State** and **Get PBR Link Partner Info** commands provide information on the devices connected to each port.

PBR switches can determine the type of device present at the far end of a link after negotiation using the link state information provided in [Table 7-110](#).

Table 7-110. Far End Device Type Detection

Device Type	Negotiated Link Direction	Negotiated PBR-Enabled	Negotiated MLD-Enabled	Received "Link Partner Info" Type
Host	USP	N	N	N/A
PBR Switch	DSP-DSP Crosslink	Y	N	Switch
GFD	DSP	Y	N	GFD
MLD	DSP	N	Y	N/A
SLD, PCIe EP, or HBR Switch	DSP	N	N	N/A

3. FM may choose to first continue discovery of any connected switches or to manage devices on the far end of all switch ports.

PBR switch PPBs connected as ISLs are configured by the FM with the **Send PPB CXL.io Configuration Request** command.

The FM uses the **Fabric Crawl Out** command, as defined in [Section 7.7.13.2](#), using switch port number as the target to manage the devices on the far end of each switch port. The FM claims ownership and assigns a PID to each defined as covered in step 1.

Once the far end device has been assigned a PID, the FM must program the PBR switch's DRT to enable routing of that PID to the appropriate switch port. The FM can now use this new assigned PID as the target for subsequent **Fabric Crawl Out** requests.

Steps 1 – 3 are repeated for all PBR switches discovered.

7.7.12.3

Assigning and Binding PIDs

As defined in [Section 7.7.6.5](#), there are many entities within a fabric that require PIDs to be assigned. GFDs and PBR switches are assigned a PID for device management purposes when the FM registers with these devices using the **Claim Ownership** command. A PBR switch reports all additional possible PID assignments with the **Get PID Target List** command.

The FM may start performing binding operations after all required PIDs have been assigned using the **Configure PID Assignment** commands. There are two methods for binding, depending on the location of the source and target of the operation. The **Bind vPPB** command is used to bind a direct attached device or LD to a switch's VCS.

The **Configure PID Binding** command is used to bind Downstream ES VCS vUSPs to Host ES vDSPs in a two-step operation. First, a binding command is sent to the Downstream ES, assigning the PID of the Host edge port to a Downstream ES VCS. Assignment of this PID allows the Downstream ES FPorts to select appropriate decoding and routing logic based on the SPID of incoming transactions. As detailed in [Section 7.7.12.4](#), latency and BW values are configured with this binding so that CDAT information can be generated in the Downstream ES.

A binding command is also sent to the Host ES, assigning the PID of the desired Downstream ES FPort and associating the binding with a specified vDSP. The Host ES uses this as the DPID for downstream transactions.

7.7.12.4

Reporting Fabric Route Performance via CDAT

Hosts require CDAT information that defines the attributes and performance characteristics of regions of memory for all memory interconnect configurations, including PBR fabrics. Special mechanisms are defined for determining and reporting this information in a PBR fabric because hosts have no visibility of intermediate ISLs, as outlined in [Section 7.7.6.1](#). The mechanisms used for LD-FAM differ from those used for G-FAM.

7.7.12.4.1

Accessing CDAT Information for LD-FAM

There are up to three components involved in the path to LD-FAM in a PBR fabric: a Host ES, a Downstream ES, and an LD-FAM device. The Host ES and LD-FAM devices require no special handling and report CDAT information covering their own characteristics as they would in an HBR system deployment. The Downstream ES, however, is required to report CDAT information that covers its own device-level performance factoring in the impact of the fabric routing path, as described below.

Latency and BW values are provided when the binding between a Host ES VCS and Downstream ES VCS is configured with the **Configure PID Binding** command. Routes through a fabric are expected to have symmetric performance characteristics. As such, only one latency and BW value is provided to define the fabric routing path. The Downstream ES adds the latency of the routing path to its own latency and uses the lesser of the BW values.

Hosts access CDAT information for Downstream ES VCSs from a DOE instance present in the vUSP.

7.7.12.4.2 Accessing CDAT Information for G-FAM

The access mechanism for CDAT from G-FAM is necessarily different from LD-FAM as a result of 2 key architectural differences: G-FAM is presented through the FAST, not a switch-based topology, and GFDs do not implement nor expose a DOE instance to the host. CDAT access for G-FAM instead relies on the use of CCI opcodes.

The GAE providing G-FAM access is responsible for producing the CDAT for each segment of the FAST. Latency and BW values are provided when PID access is enabled with the **Configure PID Access** command. The CDAT information is queried by the host using the **Read CDAT** command.

GFDs are responsible for providing CDAT information covering their own characteristics. The host queries CDAT information from GFDs using the **Proxy GFD Management** Command request to initiate the **Read CDAT** command.

7.7.12.5 Configuring CacheID in PBR Fabric

From the host's perspective, configuration of CacheID for VHs spanning a PBR Fabric is performed identically to such configuration in an exclusively HBR topology. PBR switches automatically exchange ID configuration information in the following manner:

1. The Downstream ES presents ID route table capabilities in its vPPBs (see [Section 8.2.4.28](#) for details on the CacheID Route Table).
2. The host will enumerate and assign all IDs and program the route table capability, triggering the Commit bit to complete the configuration.
3. The setting of the Commit bit triggers the Downstream ES to generate one or more RTUpdate VDMs, as defined in [Section 3.1.11.7](#), targeted at the Host PID. The Host ES will intercept this VDM based on its PBR opcode.
4. Upon receipt of the VDM, the Host ES programs the necessary ID to PID translation logic in the Host edge port.
5. The Host ES acknowledges successful programming of the ID translation logic with an RTUpdateAck VDM, as defined in [Section 3.1.11.8](#), sent to the Downstream ES for each RTUpdate VDM that was received and successfully processed.
6. Upon receipt of the VDM, the Downstream ES sets the corresponding 'RT Committed' bit in the vUSP.

A downstream HBR switch topology requires PIDs for each unique potential target so that IDs can be translated between CacheID and PID at the fabric edges. For CacheID, the ID is valid if the Valid bit is set in a Cache ID Target entry in the Cache ID Route Table Capability Structure. The corresponding PID used is the PID of the DSP to which the Route Table entry has been configured to map. Multiple PIDs must be assigned to a DSP if multiple IDs map to that DSP.

7.7.12.6 Dynamic Fabric Changes

This section outlines how FMs and PBR switches handle various changes to the system configuration during runtime.

7.7.12.6.1 Hot-Add and Link Up Events

A new Link Up on an unbound edge port is indicated to the FM via a Physical Switch Event Record. The FM uses the **Get Physical Port State** and **Get PBR Link Partner Info** commands to query information on the device connected to the port.

When an SLD or PCIe device is Hot-Added to a bound port, the FM can be notified but is not involved.

7.7.12.6.2 Dynamic Configuration Changes

There are many runtime configuration changes that an FM can trigger on a fabric:

- Binding/Unbinding: New bindings are presented to hosts as hot-add operations. Unbinding an EP is presented as a hot-remove operation.
- Updates to GMV/VTv: The GAE generates a notification to the host when changes are made to the GMV or VTV enabling or disabling access to a particular PID.
- GFD DCD changes: GFDs generate notifications to all impacted GAEs when updates are made to a host group's extent list.

7.7.12.6.3 Hot/Surprise Remove and Link Down Events

The FM is responsible for managing a Link Down event:

- The PBR switch that experienced the Link Down notifies the FM with a Physical Switch Event Record
- EP Link Down events are represented as surprise removes to the host
- The FM manages any required topology changes associated with an ISL Link Down event, including clearing the PID binding between the Upstream ES and Downstream ES VCSs, which is presented to the host as a hot-remove of the Downstream ES VCS
- GFD Link Down events prompt the FM to disable access to the corresponding PID in all impacted hosts' GAE GMV and VTV
- PBR switches drop unrouteable transactions

7.7.13 PBR Switch Command Set

This command set is only supported by, and must be supported by, PBR switches to facilitate the discovery of a PBR fabric and configuration of routing and bindings.

7.7.13.1 Identify PBR Switch (Opcode 5700h)

This command provides information to the FM about a PBR switch's fabric capabilities.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-111. Identify PBR Switch Response Payload

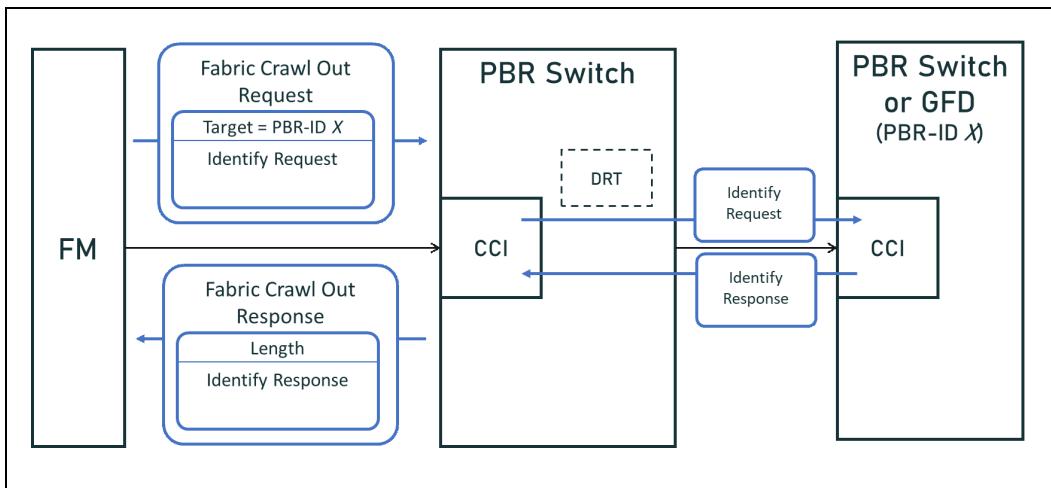
Byte Offset	Length in Bytes	Description
0h	8	GAE Support Map: Bitmask indicating whether a VCS includes (1) or does not include (0) a GAE instance in the host edge port where bit position corresponds to VCS ID.
8h	1	Number of DRTs: Total number of DRTs supported by the switch. This value shall be greater than 0.
9h	1	Number of RGTs: Total number of RGTs supported by the switch.
Ah	1	Reserved
Bh	1	<ul style="list-style-type: none"> Bit[0]: Random Supported: Indicates whether "Random" dynamic routing mode is supported (1) or not supported (0) Bit[1]: Congestion Avoidance Supported: Indicates whether "Mix with CA" dynamic routing mode is supported (1) or not supported (0) Bit[2]: Advanced Congestion Avoidance Supported: Indicates whether "Advanced CA" dynamic routing mode is supported (1) or not supported (0) Bits[5:3]: Reserved Bit[6]: Vendor-specific Routing Mode 1 Supported: Indicates whether the vendor-specific routing mode configured by dynamic routing mode value 6 is supported (1) or not supported (0) Bit[7]: Vendor-specific Routing Mode 2 Supported: Indicates whether the vendor-specific routing mode configured by mode value 7 is supported (1) or not supported (0)

7.7.13.2 Fabric Crawl Out (Opcode 5701h)

This command is used to tunnel management commands at components in a PBR fabric in two scenarios:

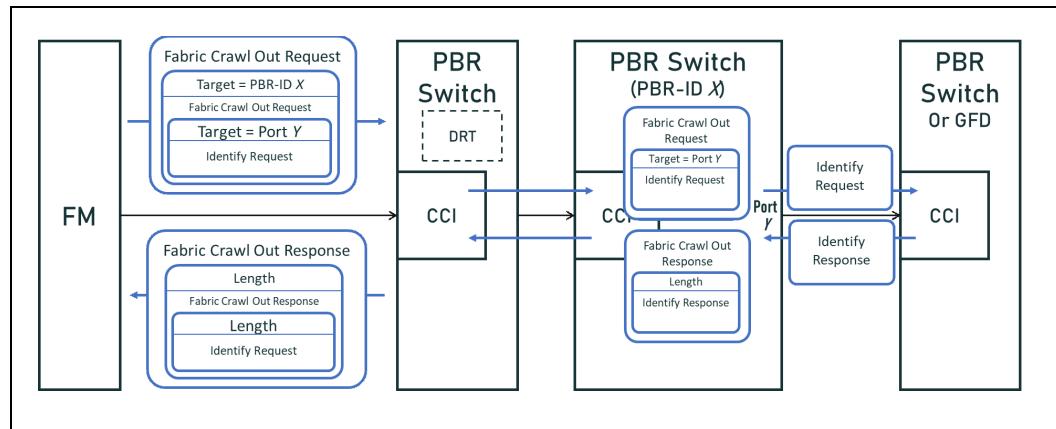
- PBR devices with no assigned PID: Tunneled command is sent to the PBR switch to which the PBR device is attached with a target specifying the PBR switch port to which the PBR device is connected. The receiving switch will transmit the command out the specified port using the reserved DPID FFFh.
- PBR devices with an assigned PID: Tunnel command is sent to a PBR switch with a target specifying the PID assigned to the PBR device.

The transport of these commands across PBR links is defined in [Section 7.7.11.6](#).

Figure 7-54. Tunneling Commands to Remote Devices

The Management Command input payload field includes the tunneled command encapsulated in the CCI Message Format, as defined in [Figure 7-19](#). This can include an additional layer of tunneling for commands issued to components with no assigned PID, as illustrated in [Figure 7-55](#).

Figure 7-55. Tunneling Commands to Remote Devices with No Assigned PID



Response size varies, based on the tunneled command's definition. Valid targets for the tunneled commands include PBR switch ports, and PBR devices within a fabric.

This command fails with "Invalid Input" if the target specifies a non-existent switch port or a PID with no valid entry in the DRT.

Components shall terminate the processing of a request that includes more than 2 layers of tunneling and provide an "Unsupported" return code.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-112. Fabric Crawl Out Request Payload

Byte Offset	Length in Bytes	Description
0h	2	<p>Target: Encoding depends on Target Type:</p> <ul style="list-style-type: none"> • Target Type = 0: <ul style="list-style-type: none"> — Bits[7:0]: Port Number: Switch shall transmit command out specified egress port. — Bits[15:8]: Reserved. • Target Type = 1: <ul style="list-style-type: none"> — Bits[11:0]: PBR-ID: Target PID. Switch shall determine egress port using DRT. — Bits[15:12]: Reserved. • All other encodings are reserved
2h	1	<ul style="list-style-type: none"> • Bits[3:0]: Target Type: Specifies the type of tunneling target for this command: <ul style="list-style-type: none"> — 0h = Port Number: Indicates that the tunneling target is a component on the far end of a switch port — 1h = PBR-ID: Indicates that the tunneling target is a component in the PBR fabric address by a PID — All other encodings are reserved • Bits[7:4]: Reserved
3h	1	Reserved
4h	2	Command Size: Number of valid bytes in Management Command.
6h	Varies	Management Command: Request message formatted in the CCI Message Format as defined in Figure 7-19 .

Table 7-113. Fabric Crawl Out Response Payload

Byte Offset	Length in Bytes	Description
0h	2	Response Length: Number of valid bytes in Response Message.
2h	2	Reserved
4h	Varies	Response Message: Response message formatted in the CCI Message Format as defined in Figure 7-19 .

7.7.13.3 Get PBR Link Partner Info (Opcode 5702h)

This command reads the data received from the latest “Link Partner Info” VDM on a PBR link.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-114. Get PBR Link Partner Info Request Payload

Byte Offset	Length in Bytes	Description
0h	1h	Number of Ports: Number of ports requested.
1h	Varies	Port ID List: 1-byte ID of requested port, repeated Number of Ports times.

Table 7-115. Get PBR Link Partner Info Response Payload

Byte Offset	Length in Bytes	Description
0h	1	Number of Ports: Number of port information blocks returned.
1h	3	Reserved
4h	Varies	Link Partner Info List: Link Partner Info block as defined in Table 7-116 , repeated Number of Ports times.

Table 7-116. Link Partner Info Format

Byte Offset	Length in Bytes	Description
00h	1	Port ID: ID of requested port.
01h	1	Far End Port ID: As reported in Link Partner Info VDM.
02h	2	<ul style="list-style-type: none"> Bits[11:0]: PID: As reported in Link Partner Info VDM Bits[15:12]: Device Type: As reported in Link Partner Info VDM
04h	10h	Device UUID: As reported in Link Partner Info VDM.
14h	1	Standard FC VC List: As reported in Link Partner Info VDM.
15h	1	UIO FC VC List: As reported in Link Partner Info VDM.
16h	10h	Primary FM UUID: As reported in Link Partner Info VDM.
26h	10h	Secondary FM UUID: As reported in Link Partner Info VDM.

7.7.13.4 Get PID Target List (Opcode 5703h)

This command retrieves the list of targets within a PBR switch to which a PID may be assigned. This does not include the PID assigned to the switch itself as part of the **Claim FM Ownership** command. As outlined in [Section 7.7.6.5](#), the following restrictions apply when assigning PIDs:

- A fabric port may be assigned one PID that can be shared among multiple fabric ports
- A Downstream Edge Port may be assigned one PID that must be unique
- A Host Edge Port may be assigned more than one PID, each of which must be unique

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-117. Get PID Target List Request Payload

Byte Offset	Length in Bytes	Description
0h	2	Start Index: Index of first target to return.
2h	2	Number of Targets: Maximum number of targets to return.

Table 7-118. Get PID Target List Response Payload

Byte Offset	Length in Bytes	Description
0h	2	Total Number of Targets: Total number of PID targets supported by the device.
2h	2	Number of Targets: Number of targets returned in Target List .
4h	Varies	Target List: List of PID target as defined in Table 7-119 .

Table 7-119. Target List Format

Byte Offset	Length in Bytes	Description
0h	2	Target ID: ID of PID Target for use in Configure PID Assignment.
2h	1	<ul style="list-style-type: none"> • Bits[2:0]: Target Type: <ul style="list-style-type: none"> — 000b = Fabric Port — 001b = Host Edge Port (USP/GAE) — 010b = Downstream Edge Port — All other encodings are reserved • Bits[7:3]: Reserved
3h	1	Instance ID: Index of PID for targets that can support multiple PIDs.
4h	1	VCS ID: ID of associated VCS. Valid only when Target Type is 1 (Host Edge Port).
5h	1	Physical Port ID: Physical port ID of the target.
6h	2	<ul style="list-style-type: none"> • Bits[11:0]: PID: Current PID assignment. FFFh if unassigned. • Bits[15:12]: Reserved.

7.7.13.5 Configure PID Assignment (Opcode 5704h)

This command is used to assign PIDs to targets within a PBR switch.

Note:

This command does not update the corresponding DRT entries for assigned or cleared PIDs. The DRT must be updated separately, using the **Set DRT** command as necessary.

This command shall return Invalid Input under the following conditions:

- Specified target is invalid
- PID has already been assigned to another target within the switch

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error

- Retry Required

Command Effects:

- None

Table 7-120. Configure PID Assignment Request Payload

Byte Offset	Length in Bytes	Description
0h	1	<ul style="list-style-type: none"> Bits[2:0]: Operation: Specifies the PID assignment operation: <ul style="list-style-type: none"> — 000b = Assign PID — 001b = Clear PID — All other encodings are reserved Bits[7:3]: Reserved
2h	2	Number of Targets : Number of entries in PID Assignment List.
4h	Varies	PID Assignment List : List of PID assignments as defined in Table 7-121 .

Table 7-121. PID Assignment

Byte Offset	Length in Bytes	Description
0h	2	<ul style="list-style-type: none"> Bits[11:0]: PID: PID to assign to the specified target Bits[15:12]: Reserved
2h	2	Target ID : Index of PID target, as reported in Get PID Target List response.
4h	1	Instance ID : Index of PID for targets that can support multiple PIDs.

7.7.13.6 Get PID Binding (Opcode 5705h)

This command reads the binding of Downstream ES PIDs to Upstream ES vDSPs or Upstream ES USP PIDs to Downstream ES vUSPs. The output also includes latency and BW values for the fabric routing path for use in generating associated CDAT information.

Possible Command Return Codes:

- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Busy

Command Effects:

- Background Operation

Table 7-122. Get PID Binding Request Payload

Byte Offset	Length in Bytes	Description
0h	1	Target VCS : ID of the VCS to query.
1h	1	Target vPPB : Index of the vPPB to query. Reserved when the binding target is a Host ES VCS.

Table 7-123. Get PID Binding Response Payload

Byte Offset	Length in Bytes	Description
00h	2	<ul style="list-style-type: none"> Bits[11:0]: PID: PID of the remote binding target. FFFh if unbound. Bits[15:12]: Reserved.
02h	2	<ul style="list-style-type: none"> Bits[11:0]: Host ES Management PID: Switch management PID of the Host ES. Valid only when the binding target is a Downstream ES VCS. Bits[15:12]: Reserved.
04h	8	Latency Entry Base Unit : Latency Entry Base Unit for path between host and target device, as defined in ACPI HMAT System Locality Latency and Bandwidth Information Structure. Valid only when the binding target is a Downstream ES VCS.
0Ch	2	Latency Entry : Latency Entry for path between host and target device, as defined in ACPI HMAT System Locality Latency and Bandwidth Information Structure. Valid only when the binding target is a Downstream ES VCS.
0Eh	8	BW Entry Base Unit : Bandwidth Entry Base Unit for path between host and target device, as defined in ACPI HMAT System Locality Latency and Bandwidth Information Structure. Valid only when the binding target is a Downstream ES VCS.
16h	2	BW Entry : Bandwidth Entry for path between host and target device, as defined in ACPI HMAT System Locality Latency and Bandwidth Information Structure. Valid only when the binding target is a Downstream ES VCS.

7.7.13.7 Configure PID Binding (Opcode 5706h)

This command configures the binding of a PID to a target. It is used to bind:

- Downstream ES PIDs to Upstream ES vDSPs
- Upstream ES USP PIDs to Downstream ES vUSPs

The command input includes latency and BW values for the fabric routing path for use in generating associated CDAT information.

Possible Command Return Codes:

- Unsupported
- Background Command Started
- Invalid Input
- Internal Error
- Retry Required
- Busy

Command Effects:

- Background Operation

Table 7-124. Configure PID Binding Request Payload (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	1	<ul style="list-style-type: none"> Bits[2:0]: Operation: <ul style="list-style-type: none"> — 000b = Bind — 001b = Unbind — All other encodings are reserved Bits[7:3]: Reserved
01h	1	Target VCS : ID of the VCS to which the PID is being bound.
02h	1	Target vPPB : Index of the vPPB to which the PID is being bound. Valid only when the binding target is a Host ES VCS.

Table 7-124. Configure PID Binding Request Payload (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
03h	1	Reserved
04h	2	<ul style="list-style-type: none"> Bits[11:0]: PID: PID of the remote binding target Bits[15:12]: Reserved
06h	2	Reserved
08h	8	Latency Entry Base Unit : Latency Entry Base Unit for path between host and target device, as defined in ACPI HMAT System Locality Latency and Bandwidth Information Structure. Valid only when the binding target is a Downstream ES VCS.
10h	2	Latency Entry : Latency Entry for path between host and target device, as defined in ACPI HMAT System Locality Latency and Bandwidth Information Structure. Valid only when the binding target is a Downstream ES VCS.
12h	8	BW Entry Base Unit : Bandwidth Entry Base Unit for path between host and target device, as defined in ACPI HMAT System Locality Latency and Bandwidth Information Structure. Valid only when the binding target is a Downstream ES VCS.
1Ah	2	BW Entry : Bandwidth Entry for path between host and target device, as defined in ACPI HMAT System Locality Latency and Bandwidth Information Structure. Valid only when the binding target is a Downstream ES VCS.

7.7.13.8 Get Table Descriptors (Opcode 5707h)

This command reads descriptors of the DPID Routing Tables and Routing Group Tables in a PBR Switch.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-125. Get Table Descriptors Request Payload

Byte Offset	Length in Bytes	Description
0h	2	Start Index : Starting index into list of descriptors.
2h	2	Number of Descriptors : Number of descriptors to read.

Table 7-126. Get Table Descriptors Response Payload

Byte Offset	Length in Bytes	Description
0h	2	Start Index : Starting index into list of descriptors.
2h	2	Number of Descriptors : Number of table descriptors.
4h	Varies	Get Table Descriptors List : List of table descriptors as defined in Table 7-127 .

Table 7-127. Get Table Descriptor Format

Byte Offset	Length in Bytes	Description
0h	1	<ul style="list-style-type: none"> Bits[1:0]: Table Type: <ul style="list-style-type: none"> — 00b = DRT — 01b = RGT — All other encodings are reserved Bits[7:2]: Reserved
1h	2	Table Index: Index of table.
3h	20h	Active Port Mask: Bitmask defining which ports actively use (1) or do not actively use (0) this table. Bit position corresponds to physical port number.
23h	4	Reserved

7.7.13.9 Get DRT (Opcode 5708h)

This command reads the DPID Routing Tables in a PBR Switch.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-128. Get DRT Request Payload

Byte Offset	Length in Bytes	Description
0h	1	DRT Index: Index of DRT to read.
1h	1	Reserved
2h	2	Number of Entries: Number of DRT entries to read.
4h	2	Start Entry: Starting index into DRT entries.

Table 7-129. Get DRT Response Payload

Byte Offset	Length in Bytes	Description
0h	1	DRT Index: Index of DRT.
1h	1	Reserved
2h	2	Number of Entries: Number of DRT entries.
4h	2	Start Entry: Starting index into DRT entries.
6h	1	Associated RGT Index: Index of RGT used by this DRT.
7h	1	Reserved
8h	Varies	DRT Entry List: List of DRT entry values as defined in Table 7-130 .

Table 7-130. DRT Entry Format

Byte Offset	Length in Bytes	Description
0h	1	<ul style="list-style-type: none"> Bits[1:0]: Entry Type: Type of routing target specifier and M2S Req routing: <ul style="list-style-type: none"> — 00b = Invalid — 01b = Physical Port number — 10b = RGT index — 11b = Reserved Bits[7:2]: Reserved
1h	1	Routing Target: Encoding depends on Entry Type: <ul style="list-style-type: none"> • 00h = Reserved • 01h = Physical port number • 02h = RGT entry index

7.7.13.10 Set DRT (Opcode 5709h)

This command sets the DPID Routing Tables in a PBR Switch.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-131. Set DRT Request Payload

Byte Offset	Length in Bytes	Description
0h	1	DRT Index: Index of DRT to configure.
1h	1	Reserved
2h	2	Number of Entries: Number of DRT entries to configure.
4h	2	Start Entry: Starting index into DRT entries.
6h	Varies	DRT Entry List: List of DRT entry values as defined in Table 7-130 .

7.7.13.11 Get RGT (Opcode 570Ah)

This command reads the Routing Group Tables in a PBR Switch.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-132. Get RGT Request Payload

Byte Offset	Length in Bytes	Description
0h	1	RGT Index: Index of RGT.
1h	1	Reserved
2h	2	Number of Entries: Number of RGT entries.
4h	2	Start Entry: Starting index into RGT entries.

Table 7-133. Get RGT Response Payload

Byte Offset	Length in Bytes	Description
0h	1	RGT Index: Index of RGT.
1h	1	Reserved
2h	2	Number of Entries: Number of RGT entries.
4h	2	Start Entry: Starting index into RGT entries.
6h	Varies	RGT Entry List: List of RGT entry values as defined in Table 7-134 .

Table 7-134. RGT Entry Format

Byte Offset	Length in Bytes	Description
0h	1	Egress Port[0]: Physical port number.
1h	1	Egress Port[1]: Physical port number.
2h	1	Egress Port[2]: Physical port number.
3h	1	Egress Port[3]: Physical port number.
4h	1	Egress Port[4]: Physical port number.
5h	1	Egress Port[5]: Physical port number.
6h	1	Egress Port[6]: Physical port number.
7h	1	Egress Port[7]: Physical port number.
8h	1	<ul style="list-style-type: none"> • Bits[2:0]: Highest Valid Entry: Highest index in the Egress Port list that is valid. • Bits[5:3]: Highest Primary Entry: Highest index in the Egress Port list that specifies a primary routing path. Subsequent valid egress ports are considered secondary paths. • Bits[7:6]: Reserved.
9h	1	<ul style="list-style-type: none"> • Bits[2:0]: Dynamic Routing Mode: Specifies the dynamic routing mode to be used for this entry: <ul style="list-style-type: none"> — 000b = Random — 001b = Congestion Avoidance — 010b = Advanced Congestion Avoidance — 011b, 101b = Reserved — 110b, 111b = Vendor-specific • Bits[5:3]: Mix Setting: Specifies the mix used for dynamic routing mode, as defined in Section 7.7.6.3 • Bits[7:6]: Reserved
Ah	2	Reserved

7.7.13.12 Set RGT (Opcode 570Bh)

This command configures the Routing Group Tables in a PBR switch.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-135. Set RGT Request Payload

Byte Offset	Length in Bytes	Description
0h	1	RGT Index: Index of RGT to configure.
1h	1	Reserved
2h	2	Number of Entries: Number of RGT entries to configure.
4h	2	Start Entry: Starting index into RGT entries.
6h	Varies	RGT Entry List: List of RGT entry values as defined in Table 7-133 .

7.7.13.13 Get LDST/IDT Capabilities (Opcode 570Ch)

This command retrieves a vPPB's LDST and IDT Capabilities, per [Section 7.7.9](#).

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-136. Get LDST/IDT Capabilities Request Payload

Byte Offset	Length in Bytes	Description
0h	1	vPPB Instance: The value of 0 represents USP. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .

Table 7-137. Get LDST/IDT Capabilities Response Payload

Byte Offset	Length in Bytes	Description
0h	1	vPPB Instance: The value of 0 represents USP. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
1h	2	Number of Segments: Number of LDST segments that are supported by this LDST/IDT. The number of entries must be 0 or a power of 2.
3h	1	LDST Segment Size <ul style="list-style-type: none"> • Bits[2:0]: LSegSz per the FSegSz encoding defined in Table 7-81 • Bits[7:3]: Reserved The device shall return 0h if this value has not been initialized.
4h	2	Number of IDT: Number of Interleave Device Table entries supported by this LDST/IDT.
6h	2	Number of Completer ID-Based Re-Routers: Number of Completer ID-Based Re-Router entries supported by this LDST/IDT.
8h	2	<ul style="list-style-type: none"> • Bits[11:0]: Local PID: PID assigned to this vPPB. FFFh if unassigned. • Bits[15:12]: Reserved.
Ah	8	Fabric Base: Base HPA of this LDST. FabricBase shall be aligned to the programmed LDST Segment Size. The device shall return 0h if this value has not been initialized.
12h	8	Fabric Limit: Upper HPA of this LDST. Shall be greater than FabricBase. Shall be aligned to the programmed LDST Segment Size. The device shall return 0h if this value has not been initialized.

7.7.13.14 Set LDST/IDT Configuration (Opcode 570Dh)

This command sets the GAE's LDST and IDT Capabilities, per [Section 7.7.9](#). Because the FabricBase and FabricLimit values must be aligned to the programmed LDST Segment Size, all three Host-chosen values are configured in one request.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- Immediate Configuration Change

Table 7-138. Set LDST/IDT Configuration Request Payload

Byte Offset	Length in Bytes	Description
0h	1	vPPB Instance: The value of 0 represents USP. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
1h	1	LDST Segment Size <ul style="list-style-type: none"> Bits[2:0]: LSegSz per the FSegSz encoding defined in Table 7-81 Bits[7:3]: Reserved
2h	8	FabricBase: Base HPA of this LDST. FabricBase shall be aligned to the programmed LDST Segment Size. The value 0h will disable this LDST/IDT decoder.
Ah	8	FabricLimit: Upper HPA of this LDST. Shall be greater than FabricBase. Shall be aligned to the programmed LDST Segment Size. The value 0h will disable this LDST/IDT decoder.

7.7.13.15 Get LDST Segment Entries (Opcode 570Eh)

This command reads the configuration of LDST Segment entries. The Host is responsible for mapping the LD-FAM range of HPAs to the appropriate number of available Segment Entries. Should the Host or the GAE have limited message payload capacity, the Host shall be responsible for breaking up the configuration operation into suitably sized requests.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-139. Get LDST Segment Entries Request Payload

Byte Offset	Length in Bytes	Description
0h	1	vPPB Instance: The value of 0 represents USP. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
1h	1	Reserved
2h	2	Seg Count: Number of LDST Segment Entries requested. Value should be >0 and not more than the lesser of the total Segment table entries available, or that number of entries that can be contained in the maximum message size handled by the host and the target GAE.
4h	2	Starting Segment Index: Index of the first segment being requested. An index of 0 shall designate the configuration of the 1 st Segment, corresponding to HPA = FabricBase. The starting index given shall not be larger than the maximum segment entry number supported. The starting index Plus the Seg Count value shall not be larger than the maximum segment entry number supported.

Table 7-140. Get LDST Segment Entries Response Payload

Byte Offset	Length in Bytes	Description
0h	2	Seg Count: Number of LDST Segment Entries described in the Seg Entry_List[]. Value should be >0 and not more than the lesser of the total Segment table entries available, or that number of entries that can be contained in the maximum message size handled by the host and the target GAE.
2h	2	Starting Segment Index: Index of the first segment being returned. An index of 0 shall designate the configuration of the 1 st Segment, corresponding to HPA = FabricBase. The starting index given shall not be larger than the maximum segment entry number supported. The starting index Plus the Seg Count value shall not be larger than the maximum segment entry number supported.
4h	Varies	Segment List[]: List of Segment Entries as defined in Table 7-141 .

Table 7-141. LDST Segment Entry Format

Byte Offset	Length in Bytes	Description
0h	1	Valid <ul style="list-style-type: none"> Bit[0]: Valid Entry: As per Figure 7-31 Bit[1]: Enable PCIe Segment: Indicates that the target is in a separate PCIe segment, thus the request will include the requester's segment number Bits[7:2]: Reserved
1h	1	Intlv <ul style="list-style-type: none"> Bits[3:0]: Interleave Mode: As per Table 7-82 Bits[7:4]: Reserved
2h	1	Gran <ul style="list-style-type: none"> Bits[3:0]: Interleave Granularity: As per Table 7-83 Bits[7:4]: Reserved
3h	1	Reserved
4h	2	DPID/IX: DPID or IDT Index, depending on Intlv field value: <ul style="list-style-type: none"> Bits[11:0]: <ul style="list-style-type: none"> If Intlv == 0, this is the actual DPID to which the LD-FAM request is to be sent. Else, this is Index of the IDT entry that contains the DPID of the first EP in the interleave set. See Figure 7-31 and the description of interleaving in Section 7.7.2.4. Bits[15:12]: Reserved
6h	2	Reserved

7.7.13.16 Set LDST Segment Entries (Opcode 570Fh)

This command is used by the Host to set the configuration of LDST Segment entries. The Host is responsible for mapping the LD-FAM range of HPAs to the appropriate number of available Segment Entries, per [Section 7.7.2.4](#). Should the Host or the GAE have limited message payload capacity, the Host shall be responsible for breaking up the configuration operation into suitably sized requests.

This command fails with Invalid Input if access to the specified DPID is not enabled in the LAV.

Possible Command Return Codes:

- Success
- Unsupported

- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- Immediate Configuration Change

Table 7-142. Set LDST Segment Entries Request Payload

Byte Offset	Length in Bytes	Description
0h	1	vPPB Instance: The value of 0 represents USP. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
1h	1	Reserved
2h	2	Seg Count: Number of LDST Segment Entries described in the Seg Entry_List[]. Value should be >0 and not more than the lesser of the total Segment table entries available, or that number of entries that can be contained in the maximum message size handled by the host and the target GAE.
4h	2	Starting Segment Index: Index of the first segment being configured. An index of 0 shall designate the configuration of the 1 st Segment, corresponding to HPA = FabricBase. The starting index given shall not be larger than the maximum segment entry number supported. The starting index Plus the Seg Count value shall not be larger than the maximum segment entry number supported.
6h	Varies	Segment List[]: List of Segment Entries as defined in Table 7-141 .

7.7.13.17 Get LDST IDT DPID Entries (Opcode 5710h)

This command reads the configuration of IDT entries that are used by the LDST. The Host is responsible for mapping the capacity of specific devices targeted by LDST into interleaved regions of HPA. Should the Host or the switch mailbox have limited message payload capacity, the Host shall be responsible for breaking up the configuration operation into suitably sized requests.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-143. Get LDST IDT DPID Entries Request Payload

Byte Offset	Length in Bytes	Description
0h	1	vPPB Instance: The value of 0 represents USP. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
1h	1	Reserved
2h	2	LDST IDT Entry Count: Number of LDST IDT Entries requested. Value should be >0 and not more than the lesser of the total LDST IDT table entries available, or that number of entries that can be contained in the maximum message size handled by the host and the target switch mailbox.
4h	2	Starting LDST IDT Entry Index: Index of the first LDST IDT entry being requested. An index of 0 shall designate the configuration of the 1 st entry. The starting index Plus the LDST IDT Entry Count value shall not be larger than the maximum LDST IDT entry number supported.

Table 7-144. Get LDST IDT DPID Entries Response Payload

Byte Offset	Length in Bytes	Description
0h	2	LDST IDT Entry Count: Number of LDST IDT Entries returned. Value should be >0 and not more than the lesser of the total LDST IDT table entries available, or that number of entries that can be contained in the maximum message size handled by the host and the target switch mailbox.
2h	2	Starting LDST IDT Entry Index: Index of the first LDST IDT entry being returned. An index of 0 shall designate the configuration of the 1 st entry. The starting index Plus the LDST IDT Entry Count value shall not be larger than the maximum LDST IDT entry number supported.
4h	Varies	LDST IDT DPID[]: DPID of the target device for the LDST IDT entry. See Figure 7-31 and the description of interleaving in Section 7.7.2.4 . Repeats LDST IDT Entry Count number of times. <ul style="list-style-type: none">• Bits[11:0]: PID of the target device• Bits[15:12]: Reserved

7.7.13.18 Set LDST IDT DPID Entries (Opcode 5711h)

This command sets the configuration of IDT entries that are used by the LDST. The Host is responsible for mapping the capacity of specific devices targeted by LDST into interleaved regions of HPA. Should the Host or the switch mailbox have limited message payload capacity, the Host shall be responsible for breaking up the configuration operation into suitably sized requests.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- Immediate Configuration Change

Table 7-145. Set LDST IDT DPID Entries Request Payload

Byte Offset	Length in Bytes	Description
0h	1	vPPB Instance: The value of 0 represents USP. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
1h	1	Reserved
2h	2	LDST IDT Entry Count: Number of LDST IDT Entries being configured. Value should be >0 and not more than the lesser of the total LDST IDT table entries available, or that number of entries that can be contained in the maximum message size handled by the host and the target switch mailbox.
4h	2	Starting LDST IDT Entry Index: Index of the first LDST IDT entry being configured. An index of 0 shall designate the configuration of the 1 st entry. The starting index Plus the LDST IDT Entry Count value shall not be larger than the maximum LDST IDT entry number supported.
6h	2	Reserved
8h	Varies	LDST IDT DPID: DPID of the device for the LDST IDT entry. See Figure 7-31 and the description of interleaving in Section 7.7.2.4 . Repeats LDST IDT Entry Count number of times. <ul style="list-style-type: none">• Bits[11:0]: PID of the target device• Bits[15:12]: Reserved

7.7.13.19 Get Completer ID-Based Re-Router Entries (Opcode 5712h)

This command reads the configuration of Completer ID-Based Re-Router entries.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-146. Get Completer ID-Based Re-Router Entries Request Payload

Byte Offset	Length in Bytes	Description
0h	1	vPPB Instance: The value of 0 represents USP. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
1h	1	Reserved
2h	2	Completer ID-Based Re-Router Entry Count: Number of Completer ID-Based Re-Router Entries requested. Value should be >0 and not more than the lesser of the total Completer ID-Based Re-Router table entries available, or that number of entries that can be contained in the maximum message size handled by the host and the target GAE.
4h	2	Starting Completer ID-Based Re-Router Entry Index: Index of the first Completer ID-Based Re-Router entry being requested. An index of 0 shall designate the configuration of the 1 st entry. The starting index Plus the Completer ID-Based Re-Router Entry Count value shall not be larger than the maximum Completer ID-Based Re-Router entry number supported.

Table 7-147. Get Completer ID-Based Re-Router Entries Response Payload

Byte Offset	Length in Bytes	Description
0h	2	Completer ID-Based Re-Router Entry Count: Number of Completer ID-Based Re-Router Entries returned. Value should be >0 and not more than the lesser of the total Completer ID-Based Re-Router table entries available, or that number of entries that can be contained in the maximum message size handled by the host and the target GAE.
2h	2	Starting Completer ID-Based Re-Router Entry Index: Index of the first Completer ID-Based Re-Router entry being returned. An index of 0 shall designate the configuration of the 1 st entry. The starting index Plus the Completer ID-Based Re-Router Entry Count value shall not be larger than the maximum Completer ID-Based Re-Router entry number supported.
4h	Varies	Completer ID-Based Re-Router Entry List[]: As defined in Table 7-148 . Repeats Completer ID-Based Re-Router Entry Count number of times.

Table 7-148. Completer ID-Based Re-Router Entry

Byte Offset	Length in Bytes	Description
0h	2	Completer ID-Based Re-Router DPID[]: DPID of the requester for the Completer ID-Based Re-Router entry. <ul style="list-style-type: none"> • Bits[11:0]: PID of the requester • Bit[12]: Enable PCIe Segment: Indicates that the requester is in a separate PCIe segment, so the request will include the requester's and completer's segment numbers • Bits[15:13]: Reserved
2h	1	Requester PCIe Segment: PCIe Segment number for the requester. Valid only if Enable PCIe Segment is set.
3h	1	Requester Bus Number: PCIe Bus number for requester.
4h	1	<ul style="list-style-type: none"> • Bits[2:0]: Requester Function Number: PCIe Function number for requester • Bits[7:3]: Requester Device Number: PCIe Device number for requester

7.7.13.20 Set Completer ID-Based Re-Router Entries (Opcode 5713h)

This command sets the configuration of Completer ID-Based Re-Router entries.

This command fails with Invalid Input if access to the specified DPID is not enabled in the LAV.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- Immediate Configuration Change

Table 7-149. Set Completer ID-Based Re-Router Entries Request Payload

Byte Offset	Length in Bytes	Description
0h	1	vPPB Instance: The value of 0 represents USP. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
1h	1	Reserved
2h	2	Completer ID-Based Re-Router Entry Count: Number of Completer ID-Based Re-Router Entries being configured. Value should be >0 and not more than the lesser of the total Completer ID-Based Re-Router table entries available, or that number of entries that can be contained in the maximum message size handled by the host and the target GAE.
4h	2	Starting Completer ID-Based Re-Router Entry Index: Index of the first Completer ID-Based Re-Router entry being configured. An index of 0 shall designate the configuration of the 1 st entry. The starting index given shall not be larger than the maximum Completer ID-Based Re-Router entry number supported. The starting index Plus the Completer ID-Based Re-Router Entry Count value shall not be larger than the maximum Completer ID-Based Re-Router entry number supported.
6h	Varies	Completer ID-Based Re-Router Entry List[]: As defined in Table 7-148 . Repeats Completer ID-Based Re-Router Entry Count number of times.

7.7.13.21 Get LDST Access Vector (Opcode 5714h)

This command is used by the host to query its current LAV.

This command will return Invalid Input when the requested byte range exceeds the size of the access vector buffer, as defined in [Table 7-163](#).

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-150. Get LDST Access Vector Request Payload

Byte Offset	Length in Bytes	Description
0h	4	Start Byte: Offset in bytes into Vector Data .
4h	4	Number of Bytes: Size in bytes of Vector Data requested.

Table 7-151. Get LDST Access Vector Response Payload

Byte Offset	Length in Bytes	Description
0h	4	Number of Bytes: Size in bytes of Vector Data returned.
4h	Varies	Vector Data: Excerpt of data from LDST Access Vector , defined in Table 7-152 . Excerpt begins a Start Byte and is Number of Bytes long.

Table 7-152. LDST Access Vector

Byte Offset	Length in Bytes	Description
000h	200h	LDST Access Vector: 4-Kb vector in which each bit corresponds to the associated PID (i.e., bit n represents PID n). A value of 1 in a bit position indicates that LDST and ID-Based Re-Router access to the corresponding PID is enabled. A value of 0 in a bit position indicates that access to the corresponding PID is blocked.

7.7.13.22 Get VCS LDST Access Vector (Opcode 5715h)

This command is used by the FM to query a VCS's current LAV.

This command will return Invalid Input when the requested byte range exceeds the size of the access vector buffer, as defined in [Table 7-163](#).

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-153. Get VCS LDST Access Vector Request Payload

Byte Offset	Length in Bytes	Description
0h	1	VCS ID: ID of VCS to query.
1h	3	Reserved
4h	4	Start Byte: Offset in bytes into Vector Data .
8h	4	Number of Bytes: Size in bytes of Vector Data requested.

The Get VCS LDST Access Vector Response Payload is defined in [Table 7-151](#).

7.7.13.23 Configure VCS LDST Access (Opcode 5716h)

This command is used by the FM to control access to a specified PID as reported in the LAV.

Possible Command return codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-154. Configure VCS LDST Access Request Payload

Byte Offset	Length in Bytes	Description
0h	1	VCS ID: ID of VCS to configure.
1h	1	Reserved
2h	2	<ul style="list-style-type: none"> • Bits[11:0]: PID: PID of LDST or Completer ID-Based Re-Router target • Bits[14:12]: Operation: Specifies which configuration to perform: <ul style="list-style-type: none"> — 000b = Enable PID access in the LAV — 001b = Disable PID access in the LAV — All other encodings are reserved • Bit[15]: Reserved

7.7.14**Global Memory Access Endpoint Command Set**

This command set is used by a host to discover and manage the structures and devices involved in providing access to G-FAM and GIM resources.

7.7.14.1**Identify GAE (Opcode 5800h)**

This command is used by the Host to query a GAE's capabilities, including maximum number of supported enabled PIDs and maximum number of simultaneous outstanding proxy operations and VendPrefixL0 support. It also reports the remaining number of proxy threads currently available.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-155. Identify GAE Request Payload

Byte Offset	Length in Bytes	Description
0h	1	Start vPPB Instance: Index of vPPB whose FAST Segment Info should be provided in the first entry in vPPB Global Memory Support Info List . The value of 0 represents the GAE. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
1h	1	Number of vPPBs: Number of vPPBs in vPPB Global Memory Support Info List .

Table 7-156. Identify GAE Response Payload

Byte Offset	Length in Bytes	Description
0h	2	<ul style="list-style-type: none"> Bits[11:0]: Total Number of Supported Enabled PIDs: Maximum number of PIDs that can be enabled for concurrent use with the Configure PID Access command Bit[12]: Egress Request/Ingress Completion VendPrefixL0 Supported: Indicates whether VendPrefixL0 is supported (1) or not supported (0) for Egress UIO Requests and Ingress UIO completions, as configured by the FM for this host with the Set VendPrefixL0 State command Bit[13]: Ingress Request VendPrefixL0 Supported: Indicates whether VendPrefixL0 is supported (1) or not supported (0) for Ingress UIO requests, as configured by the FM for this host with the Set VendPrefixL0 State command Bit[14]: G-FAM/GIM Configuration Supported: Indicates whether the switch supports (1) or does not support (0) re-configuration of the GIM Support bit with the Set FAST Segment Entry command Bit[15]: Reserved
2h	2	Total Number of Supported Threads: Maximum number of simultaneous proxy operations supported by the GAE.
4h	2	Number of Available Threads: Remaining number of simultaneous proxy operations supported by the GAE.
6h	1	Start vPPB Instance: Index of vPPB whose FAST Segment Info is provided in the first entry in vPPB Global Memory Support Info List .
7h	1	Number of vPPBs: Number of vPPBs whose FAST Segment Info is provided in the first entry in vPPB Global Memory Support Info List .
8h	Varies	vPPB Global Memory Support Info List: List of vPPB Global Memory Support Info, as defined in Table 7-157 , for the vPPBs identified with Start vPPB Instance and Number of vPPBs .

Table 7-157. vPPB Global Memory Support Info

Byte Offset	Length in Bytes	Description
0h	2	Number of FAST Segments: Total number of segments in the FAST for the specified GAE/vPPB.
2h	2	Reserved

7.7.14.2 Get PID Interrupt Vector (Opcode 5801h)

This command queries a GAE's PID interrupt vector.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-158. Get PID Interrupt Vector Request Payload

Byte Offset	Length in Bytes	Description
0h	4	Start Byte: Offset in bytes into PID Interrupt Vector.
4h	4	Number of Bytes: Size in bytes of PID Interrupt Vector requested.
8h	1	<ul style="list-style-type: none"> Bit[0]: Clear on Read: A value of 1 indicates that the PID Interrupt Vector should be cleared to all 0s when this command completes. A GAE must ensure that no interrupts are lost in between capturing the current PID Interrupt Vector value for the response payload and clearing the vector's contents. Bits[7:1]: Reserved.

Table 7-159. Get PID Interrupt Vector Response Payload

Byte Offset	Length in Bytes	Description
0h	4	Number of Bytes: Size in bytes of PID Interrupt Vector returned.
4h	Varies	PID Interrupt Vector Data: Excerpt of data from PID Interrupt Vector , defined in Table 7-160 . Excerpt begins a Start Byte and is Number of Bytes long.

Table 7-160. PID Interrupt Vector

Byte Offset	Length in Bytes	Description
000h	200h	PID Interrupt Vector: 4-Kb vector in which each bit corresponds to the associated PID (i.e., bit n represents PID n). A value of 1 in a bit position indicates that the GAE has received an interrupt VDM from the corresponding PID since the PID Interrupt Vector was last cleared.

7.7.14.3 Get PID Access Vectors (Opcode 5802h)

This command is used by the Host to query a GAE's current GFD Mapping Vector and VendPrefixL0 Target Vector.

This command will return Invalid Input when the requested byte range exceeds the size of the access vector buffer, as defined in [Table 7-163](#).

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-161. Get PID Access Vectors Request Payload

Byte Offset	Length in Bytes	Description
0h	4	Start Byte: Offset in bytes into Vector Data .
1h	4	Number of Bytes: Size in bytes of Vector Data requested.

Table 7-162. Get PID Access Vectors Response Payload

Byte Offset	Length in Bytes	Description
0h	4	Number of Bytes: Size in bytes of Vector Data returned.
4h	Varies	Vector Data: Excerpt of data from PID Access Vector , defined in Table 7-163 . Excerpt begins a Start Byte and is Number of Bytes long.

Table 7-163. PID Access Vector

Byte Offset	Length in Bytes	Description
000h	200h	GFD Mapping Vector: 4-Kb vector in which each bit corresponds to the associated PID (i.e., bit n represents PID n). A value of 1 in a bit position indicates that FAST access to the corresponding PID is enabled. A value of 0 in a bit position indicates that access to the corresponding PID is blocked.
200h	200h	VendPrefixL0 Target Vector: 4-Kb vector in which each bit corresponds to the associated PID (i.e., bit n represents PID n). A value of 1 in a bit position indicates that VendPrefixL0 access to the corresponding PID is enabled. A value of 0 in a bit position indicates that access to the corresponding PID is blocked.

7.7.14.4 Get FAST/IDT Capabilities (Opcode 5803h)

This command is used by the Host to retrieve the GAE's FAST and IDT Capabilities, per [Section 7.7.2.4](#).

The host should re-discover the FAST/IDT Capabilities of a vPPB after a Presence Detect Changed notification has been received indicating that an adapter is present if the vPPB supports Presence Detect, or when a Link Up is detected if the vPPB does not support Presence Detect.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-164. Get FAST/IDT Capabilities Request Payload

Byte Offset	Length in Bytes	Description
0h	1	vPPB Instance: The value of 0 represents GAE. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
1h	3	Reserved

Table 7-165. Get FAST/IDT Capabilities Response Payload

Byte Offset	Length in Bytes	Description
0h	1	vPPB Instance: The value of 0 represents USP. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
1h	2	Number of Segments: Number of FAST segments that are supported by this FAST/IDT. The number of entries must be 0 or a power of 2.
3h	1	FAST Segment Size <ul style="list-style-type: none"> Bits[2:0]: FSegSz per Table 7-81 Bits[7:3]: Reserved The device shall return 0h if this value has not been initialized.
4h	2	Number of IDT: Number of Interleave Device Table entries supported by this FAST/IDT.
6h	1	vPPB PID List Length: Number of PIDs assigned to this vPPB, as reported in vPPB PID List . Shall be 0 for vDSPs and vUSPs.
7h	1	<ul style="list-style-type: none"> Bit[0]: Egress Request/Ingress Completion VendPrefixL0 Enabled: Indicates whether VendPrefixL0 is enabled (1) or disabled (0) for Egress UIO Requests and Ingress UIO completions Bit[1]: Ingress Request VendPrefixL0 Enabled: Indicates whether VendPrefixL0 is enabled (1) or disabled (0) for Ingress UIO requests Bit[7:2]: Reserved
8h	2	Reserved
Ah	8	Fabric Base: Base HPA of this FAST. The device shall return 0h if this value has not been initialized.
12h	8	Fabric Limit: Upper HPA of this FAST. The device shall return 0h if this value has not been initialized.
1Ah	Varies	vPPB PID: List of PIDs assigned to this vPPB, as defined in Table 7-166 .

Table 7-166. vPPB PID List Entry Format

Byte Offset	Length in Bytes	Description
0h	2	<ul style="list-style-type: none"> Bits[11:0]: vPPB PID: PID assigned to the vPPB Bits[15:12]: Reserved

7.7.14.5 Set FAST/IDT Configuration (Opcode 5804h)

This command is used by the Host to set the GAE's FAST and IDT Capabilities, per [Section 7.7.2.4](#). Because the FabricBase and FabricLimit values must be aligned to the programmed FAST Segment Size, all three Host-chosen values are configured in one request.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- Immediate Configuration Change

Table 7-167. Set FAST/IDT Configuration Request Payload

Byte Offset	Length in Bytes	Description
0h	1	vPPB Instance: The value of 0 represents USP. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
1h	1	FAST Segment Size <ul style="list-style-type: none"> • Bits[2:0]: FSegSz per Table 7-81 • Bits[7:3]: Reserved
2h	1	<ul style="list-style-type: none"> • Bit[0]: Enable Egress Request/Ingress Completion VendPrefixL0: Configures whether VendPrefixL0 is enabled (1) or disabled (0) for Egress UIO Requests and Ingress UIO completions • Bit[1]: Enable Ingress Request VendPrefixL0: Configures whether VendPrefixL0 is enabled (1) or disabled (0) for Ingress UIO requests • Bit[7:2]: Reserved
3h	1	Reserved
4h	8	FabricBase: Base HPA of this FAST. FabricBase shall be aligned to the programmed FAST Segment Size. The value 0h will disable this FAST/IDT decoder.
Ch	8	FabricLimit: Upper HPA of this FAST. Shall be greater than FabricBase. Shall be aligned to the programmed FAST Segment Size. The value 0h will disable this FAST/IDT decoder.

7.7.14.6 Get FAST Segment Entries (Opcode 5805h)

This command reads the configuration of FAST Segment entries. The Host is responsible for mapping the GFAM range of HPAs to the appropriate number of available Segment Entries. Should the Host or the GAE have limited message payload capacity, the Host shall be responsible for breaking up the configuration operation into suitably sized requests.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-168. Get FAST Segment Entries Request Payload

Byte Offset	Length in Bytes	Description
0h	1	vPPB Instance: The value of 0 represents USP. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
1h	1	Reserved
2h	2	Seg Count: Number of FAST Segment Entries requested. Value should be >0 and not more than the lesser of the total Segment table entries available, or that number of entries that can be contained in the maximum message size handled by the host and the target GAE.
4h	2	Starting Segment Index: Index of the first segment being requested. An index of 0 shall designate the configuration of the 1 st Segment, corresponding to HPA = FabricBase. The starting index Plus the Seg Count value shall not be larger than the maximum segment entry number supported.

Table 7-169. Get FAST Segment Entries Response Payload

Byte Offset	Length in Bytes	Description
0h	2	Seg Count: Number of FAST Segment Entries described in the Seg Entry_List[]. Value should be >0 and not more than the lesser of the total Segment table entries available, or that number of entries that can be contained in the maximum message size handled by the host and the target GAE.
2h	2	Starting Segment Index: Index of the first segment being returned. An index of 0 shall designate the configuration of the 1 st Segment, corresponding to HPA = FabricBase. The starting index Plus the Seg Count value shall not be larger than the maximum segment entry number supported.
4h	Varies	Segment List[]: List of Segment Entries as defined in Table 7-170 .

Table 7-170. FAST Segment Entry Format

Byte Offset	Length in Bytes	Description
0h	1	Valid <ul style="list-style-type: none"> Bit[0]: Valid Entry: As per Figure 7-31 Bit[1]: GIM Segment: Segment used for GIM access Bits[7:2]: Reserved
1h	1	Intlv <ul style="list-style-type: none"> Bits[3:0]: Interleave Mode: As per Table 7-82 Bits[7:4]: Reserved
2h	1	Gran <ul style="list-style-type: none"> Bits[3:0]: Interleave Granularity: As per Table 7-83 Bits[7:4]: Reserved
3h	1	Reserved
4h	2	DPID/IX: DPID or IDT Index, depending on Intlv field value: <ul style="list-style-type: none"> Bits[11:0]: <ul style="list-style-type: none"> If Intlv == 0, this is the actual DPID to which the GFAM request is to be sent. Else, this is Index of the IDT entry that contains the DPID of the first GFD in the interleave set. See Figure 7-31 and the description of interleaving in Section 7.7.2.4. Bits[15:12]: Reserved
6h	2	Reserved

7.7.14.7 Set FAST Segment Entries (Opcode 5806h)

This command is used by the Host to set the configuration of FAST Segment entries. The Host is responsible for mapping the GFAM range of HPAs to the appropriate number of available Segment Entries, per [Section 7.7.2.4](#). Should the host or the GAE have limited message payload capacity, the Host shall be responsible for breaking up the configuration operation into suitably sized requests.

There are two types of segments: those that access G-FAM, and those that access GIM. Valid PID targets for G-FAM segments are defined in the GMV. Valid targets for GIM segments are defined in the VTV.

This command will complete with an Invalid Input status if the requester is not authorized to access the specified ID, as advertised by the GMV or VTV.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- Immediate Configuration Change

Table 7-171. Set FAST Segment Entries Request Payload

Byte Offset	Length in Bytes	Description
0h	1	vPPB Instance: The value of 0 represents USP. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
1h	1	Reserved
2h	2	Seg Count: Number of FAST Segment Entries described in the Seg Entry_List[]. Value should be >0 and not more than the lesser of the total Segment table entries available, or that number of entries that can be contained in the maximum message size handled by the host and the target GAE.
4h	2	Starting Segment Index: Index of the first segment being configured. An index of 0 shall designate the configuration of the 1 st Segment, corresponding to HPA = FabricBase. The starting index Plus the Seg Count value shall not be larger than the maximum segment entry number supported.
6h	Varies	Segment List[]: List of Segment Entries as defined in Table 7-170 .

7.7.14.8 Get IDT DPID Entries (Opcode 5807h)

This command reads the configuration of IDT entries. The Host is responsible for mapping the capacity of specific GFDs into interleaved regions of HPA. Should the Host or the GAE have limited message payload capacity, the Host shall be responsible for breaking up the configuration operation into suitably sized requests.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error

- Retry Required

Command Effects:

- None

Table 7-172. Get IDT DPID Entries Request Payload

Byte Offset	Length in Bytes	Description
0h	1	vPPB Instance: The value of 0 represents USP. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
1h	1	Reserved
2h	2	IDT Entry Count: Number of IDT Entries requested. Value should be >0 and not more than the lesser of the total IDT table entries available, or that number of entries that can be contained in the maximum message size handled by the host and the target GAE.
4h	2	Starting IDT Entry Index: Index of the first IDT entry being requested. An index of 0 shall designate the configuration of the 1 st entry. The starting index Plus the IDT Entry Count value shall not be larger than the maximum IDT entry number supported.

Table 7-173. Get IDT DPID Entries Response Payload

Byte Offset	Length in Bytes	Description
0h	2	IDT Entry Count: Number of IDT Entries returned. Value should be >0 and not more than the lesser of the total IDT table entries available, or that number of entries that can be contained in the maximum message size handled by the host and the target GAE.
2h	2	Starting IDT Entry Index: Index of the first IDT entry being returned. An index of 0 shall designate the configuration of the 1 st entry. The starting index Plus the IDT Entry Count value shall not be larger than the maximum IDT entry number supported.
4h	Varies	IDT DPID[]: DPID of the GFD for the IDT entry. See Figure 7-31 and the description of interleaving in Section 7.7.2.4 . Repeats IDT Entry Count number of times. <ul style="list-style-type: none"> • Bits[11:0]: PID of the target GFD • Bits[15:12]: Reserved

7.7.14.9 Set IDT DPID Entries (Opcode 5808h)

This command sets the configuration of IDT entries. The Host is responsible for mapping the capacity of specific GFDs into interleaved regions of HPA. Should the Host or the GAE have limited message payload capacity, the Host shall be responsible for breaking up the configuration operation into suitably sized requests.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- Immediate Configuration Change

Table 7-174. Set IDT DPID Entries Request Payload

Byte Offset	Length in Bytes	Description
0h	1	vPPB Instance: The value of 0 represents USP. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
1h	1	Reserved
2h	2	IDT Entry Count: Number of IDT Entries being configured. Value should be >0 and not more than the lesser of the total IDT table entries available, or that number of entries that can be contained in the maximum message size handled by the host and the target GAE.
4h	2	Starting IDT Entry Index: Index of the first IDT entry being configured. An index of 0 shall designate the configuration of the 1 st entry. The starting index Plus the IDT Entry Count value shall not be larger than the maximum IDT entry number supported.
6h	2	Reserved
8h	Varies	IDT DPID: DPID of the GFD for the IDT entry. See Figure 7-31 and the description of interleaving in Section 7.7.2.4 . Repeats IDT Entry Count number of times. <ul style="list-style-type: none">• Bits[11:0]: PID of the target GFD• Bits[15:12]: Reserved

7.7.14.10 Proxy GFD Management Command (Opcode 5809h)

This command is used to initiate the transfer of a management command to a GFD, as defined in [Section 3.1.11.1](#).

Only one proxy request may be outstanding per target PID regardless of the number of available proxy threads. A proxy request that targets a PID with an existing outstanding proxy request shall fail with ‘Invalid Input’. The command shall fail with ‘Resources Exhausted’ if there are no available proxy operation threads.

The GAE increments and tracks Command Sequence Number on a per-Target PID basis.

This command will complete with an Invalid Input status if the requester is not authorized to access the specified ID, as advertised by the GMV.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Resources Exhausted

Command Effects:

- None

Table 7-175. Proxy GFD Management Command Request Payload

Byte Offset	Length in Bytes	Description
00h	2	<ul style="list-style-type: none"> Bits[11:0]: PBR-ID: Target PID for the management command Bits[15:12]: Reserved
02h	8	Request Address : Pointer to request message in Host memory that is formatted in the CCI Message Format as defined in Figure 7-19 .
0Ah	2	Request Size : Size of the request at Request Address in bytes.
0Ch	8	Response Address : Pointer in Host memory at which the response should be written. The response shall be formatted in the CCI Message Format as defined in Figure 7-19 .
14h	2	Maximum Response Size : Size of the response at Request Address in bytes.

Table 7-176. Proxy GFD Management Command Response Payload

Byte Offset	Length in Bytes	Description
0h	1	<ul style="list-style-type: none"> Bits[2:0]: Command Sequence Number: Proxy thread identifier for use with Get Proxy Thread Status request Bits[7:3]: Reserved
1h	2	Number of Available Threads : Remaining number of simultaneous proxy operations supported by the GAE.

7.7.14.11 Get Proxy Thread Status (Opcode 580Ah)

This command queries whether the GAE is tracking the specified Command Sequence Number and Target PID as 'In Progress'.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-177. Get Proxy Thread Status Request Payload

Byte Offset	Length in Bytes	Description
0h	2	<ul style="list-style-type: none"> Bits[11:0]: PBR-ID: Target PID for the management command Bits[14:12]: Command Sequence Number: Proxy thread identifier returned by Proxy GFD Management Command request Bit[15]: Reserved

Table 7-178. Get Proxy Thread Status Response Payload

Byte Offset	Length in Bytes	Description
0h	1	<ul style="list-style-type: none"> Bit[0]: In Progress: A value of 1 indicates that the GAE is tracking the specified Thread Handle. A value of 0 indicates that the GAE is not tracking the specified Thread Handle. Bits[7:1]: Reserved.
1h	1	Reserved
2h	2	Number of Available Threads : Remaining number of simultaneous proxy operations supported by the GAE.

7.7.14.12 Cancel Proxy Thread (Opcode 580Bh)

This command effectively cancels a proxy thread that is in progress by instructing the GAE to no longer track the specified thread handle as 'In Progress'. The GAE shall discard any transactions associated with threads that are not being tracked as 'In Progress'.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-179. Cancel Proxy Thread Request Payload

Byte Offset	Length in Bytes	Description
0h	2	<ul style="list-style-type: none"> Bits[11:0]: PBR-ID: Target PID for the management command Bits[14:12]: Command Sequence Number: Proxy thread identifier returned by Proxy GFD Management Command request Bit[15]: Reserved

Table 7-180. Cancel Proxy Thread Response Payload

Byte Offset	Length in Bytes	Description
0h	2	Number of Available Threads : Remaining number of simultaneous proxy operations supported by the GAE.

7.7.15 Global Memory Access Endpoint Management Command Set

This command set is used by the FM to discover and manage the structures and devices involved in providing access to G-FAM and GIM resources.

7.7.15.1 Identify VCS GAE (Opcode 5900h)

This command is used by the FM to query a GAE's capabilities, including maximum number of supported enabled PIDs and maximum number of simultaneous outstanding proxy operations and VendPrefixL0 support. It also reports the remaining number of proxy threads currently available.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-181. Identify VCS GAE Request Payload

Byte Offset	Length in Bytes	Description
0h	1	VCS ID: ID of VCS to query.
1h	1	Start vPPB Instance: Index of vPPB whose FAST Segment Info should be provided in the first entry in vPPB Global Memory Support Info List . The value of 0 represents the GAE. The values of 1 and above represent the DSP vPPBs in increasing Device Number, Function Number order, as defined in Section 7.1.4 .
2h	1	Number of vPPBs: Number of vPPBs whose FAST Segment Info should be provided in vPPB Global Memory Support Info List .

The Identify VCS GAE Response Payload is defined in [Table 7-156](#).

7.7.15.2 Get VCS PID Access Vectors (Opcode 5901h)

This command is used by the FM to query a GAE's current GFD Mapping Vector and VendPrefixL0 Target Vector.

This command will return Invalid Input under the following conditions:

- The requested byte range exceeds the size of the access vector buffer, as defined in [Table 7-163](#)
- The specified VCS does not include a GAE

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-182. Get VCS PID Access Vectors Request Payload

Byte Offset	Length in Bytes	Description
0h	1	VCS ID: ID of VCS to query.
1h	3	Reserved
4h	4	Start Byte: Offset in bytes into Vector Data .
8h	4	Number of Bytes: Size in bytes of Vector Data requested.

The Get VCS PID Access Vectors Response Payload is defined in [Table 7-162](#).

7.7.15.3 Configure VCS PID Access (Opcode 5902h)

This command is used by the FM to control access to a specified PID as reported in the GFD Mapping Vector or VendPrefixL0 Target Vector. It is used by the FM to enable or disable access to a PID from a GAE.

Possible Command return codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-183. Configure VCS PID Access Request Payload

Byte Offset	Length in Bytes	Description
0h	1	VCS ID: ID of VCS to configure.
1h	1	Reserved
2h	2	<ul style="list-style-type: none"> • Bits[11:0]: PID: PID of GFD or VendPrefixL0 target • Bits[14:12]: Operation: Specifies which configuration to perform: <ul style="list-style-type: none"> — 000b = Enable PID access in the GMV — 001b = Disable PID access in the GMV — 010b = Enable PID access in the VTV — 011b = Disable PID access in the VTV — 100b = Update Latency/BW — All other encodings are reserved • Bit[15]: Reserved
4h	8	Latency Entry Base Unit: Latency Entry Base Unit for path between host and target device, as defined in ACPI HMAT System Locality Latency and Bandwidth Information Structure. Reserved when Operation is 001b or 011b.
Ch	2	Latency Entry: Latency Entry for path between host and target device, as defined in ACPI HMAT System Locality Latency and Bandwidth Information Structure. Reserved when Operation is 001b or 011b.
Eh	8	BW Entry Base Unit: Bandwidth Entry Base Unit for path between host and target device, as defined in ACPI HMAT System Locality Latency and Bandwidth Information Structure. Reserved when Operation is 001b or 011b.
16h	2	BW Entry: Bandwidth Entry for path between host and target device, as defined in ACPI HMAT System Locality Latency and Bandwidth Information Structure. Reserved when Operation is 001b or 011b.

7.7.15.4 Get VendPrefixL0 State (Opcode 5903h)

This command is used by the FM to query the enable state of VendPrefixL0 in a VCS. Support for this command indicates whether a PBR switch supports VendPrefixL0. The **Get VendPrefixL0 State** command shall only be implemented by PBR switches that support VendPrefixL0.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-184. Get VendPrefixL0 State Request Payload

Byte Offset	Length in Bytes	Description
0h	1	VCS ID: ID of the VCS to which the GAE or vPPB belongs.

Table 7-185. Get VendPrefixL0 State Response Payload

Byte Offset	Length in Bytes	Description
0h	1	<ul style="list-style-type: none"> • Bit[0]: Egress Request/Ingress Completion VendPrefixL0 Enabled: Indicates whether support for VendPrefixL0 is enabled (1) or disabled (0) for Egress UIO Requests and Ingress UIO completions in the specified VCS • Bit[1]: Ingress Request VendPrefixL0 Enabled: Indicates whether support for VendPrefixL0 is enabled (1) or disabled (0) for Ingress UIO requests in the specified VCS • Bit[7:2]: Reserved

7.7.15.5 Set VendPrefixL0 State (Opcode 5904h)

This command is used by the FM to enable or disable support for VendPrefixL0 in a VCS. Support for this command indicates whether a PBR switch supports VendPrefixL0; it shall be implemented by and shall only be implemented by PBR switches that support VendPrefixL0.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 7-186. Set VendPrefixL0 State Request Payload

Byte Offset	Length in Bytes	Description
0h	1	VCS ID: ID of the VCS to configure.
1h	1	Reserved
2h	1	<ul style="list-style-type: none">• Bit[0]: Enable Egress Request/Ingress Completion VendPrefixL0: Enables (1) or disables (0) support for VendPrefixL0 for Egress UIO Requests and Ingress UIO completions in the specified VCS• Bit[1]: Enable Ingress Request VendPrefixL0: Enables (1) or disables (0) support for VendPrefixL0 for Ingress UIO requests in the specified VCS• Bit[7:2]: Reserved

§ §

8.0

Control and Status Registers

The CXL component control and status registers are mapped into separate spaces:

- Configuration Space: Registers are accessed using configuration reads and configuration writes
- Memory mapped space: Registers are accessed using memory reads and memory writes

Table 8-1 summarizes the attributes for the register bits defined in this chapter. Unless specified otherwise, the definition of these attributes is consistent with PCIe® Base Specification.

All numeric values in various registers and data structures are always encoded in little-endian format. All UUIDs in this section follow the format defined in the IETF RFC 4122 specification.

CXL components have the same requirements as PCIe with respect to hardware initializing the register fields to their default values, with notable exceptions for system-integrated devices. See PCIe Base Specification for details.

Table 8-1. Register Attributes

Attribute	Description
RO	Read Only
ROS	Read Only Sticky : Not affected by CXL Reset. Otherwise, the behavior follows PCIe Base Specification.
RW	Read-Write
RWS	Read-Write-Sticky : Not affected by CXL Reset. Otherwise, the behavior follows PCIe Base Specification.
RWO	Read-Write-One-To-Lock : This attribute is not defined in PCIe Base Specification and is unique to CXL. Field becomes RO after writing 1 to it. Cleared by a hot reset, a warm reset, or a cold reset. Not affected by CXL Reset.
RWL	Read-Write-Lockable : This attribute is not defined in PCIe Base Specification and is unique to CXL. These bits follow RW behavior until they are locked. After the bits are locked, the value cannot be altered by software until the next hot reset, warm reset, or cold reset. Upon hot reset, warm reset, or cold reset, the behavior reverts back to RW. Not affected by CXL Reset after the bits are locked. The locking condition associated with each RWL field is specified as part of the field definition.
RW1C	Read-Write-One-To-Clear
RW1CS	Read-Write-One-To-Clear-Sticky : Not affected by CXL Reset. Otherwise, the behavior follows PCIe Base Specification.
HwInit	Hardware Initialized
RsvdP	Reserved and Preserved
RsvdZ	Reserved and Zero

8.1 Configuration Space Registers

This section describes the Configuration Space registers that may be used to discover and configure CXL functionality. RCH Downstream Port does not map any registers into Configuration Space.

8.1.1

PCIe Designated Vendor-Specific Extended Capability (DVSEC) ID Assignment

The CXL specification-defined Configuration Space registers are grouped into blocks, and each block is enumerated as a PCIe Designated Vendor-Specific Extended Capability (DVSEC) structure. The DVSEC Vendor ID field is set to 1E98h to indicate that these Capability structures are defined by the CXL specification.

The DVSEC Revision field represents the version of the DVSEC structure. The DVSEC Revision is incremented whenever the structure is extended to add more functionality. Backward compatibility shall be maintained during this process. For all values of n, a DVSEC Revision n+1 structure may extend Revision n by replacing fields that are marked as reserved in Revision n, but must not redefine the meaning of existing fields. In addition, Revision n+1 may append new registers to Revision n structure and thereby increasing the DVSEC Length field. Software that was written for a lower Revision may continue to operate on CXL DVSEC structures with a higher Revision, but will not be able to take advantage of new functionality.

The following values of DVSEC ID, as listed in [Table 8-2](#), are defined by the CXL specification.

[Table 8-2](#) in this version of the specification does not define the behavior of the CXL fabric switches (see [Section 2.7](#)) and G-FAM devices (see [Section 2.8](#)).

Table 8-2.

CXL DVSEC ID Assignment (Sheet 1 of 2)

CXL Capability (see Section 8.1.3)	DVSEC ID	Highest DVSEC Revision	Mandatory ¹	Not Permitted ¹	Optional ¹
PCIe DVSEC for CXL Devices (see Section 8.1.3)	0000h	3	D1, D2, LD, FMLD	P, UP1, DP1, R, USP, DSP	
Non-CXL Function Map DVSEC (see Section 8.1.4)	0002h	0		P, UP1, DP1, R, DSP	D1, D2, LD, FMLD, USP ²
CXL Extensions DVSEC for Ports (formerly known as CXL 2.0 Extensions DVSEC for Ports; see Section 8.1.5)	0003h	0	R, USP, DSP	P, D1, D2, LD, FMLD, UP1, DP1	
GPF DVSEC for CXL Ports (see Section 8.1.6)	0004h	0	R, DSP	P, D1, D2, LD, FMLD, UP1, DP1, USP	
GPF DVSEC for CXL Devices (see Section 8.1.7)	0005h	0	D2, LD	P, UP1, DP1, R, USP, DSP, FMLD	D1
PCIe DVSEC for Flex Bus Port (see Section 8.1.8)	0007h	2	D1, D2, LD, FMLD, UP1, DP1, R, USP, DSP	P	

Table 8-2. CXL DVSEC ID Assignment (Sheet 2 of 2)

CXL Capability	DVSEC ID	Highest DVSEC Revision	Mandatory ¹	Not Permitted ¹	Optional ¹
Register Locator DVSEC (see Section 8.1.9)	0008h	0	D2, LD, FMLD, R, USP, DSP	P	D1, UP1, DP1
MLD DVSEC (see Section 8.1.10)	0009h	0	FMLD	P, D1, D2, LD, UP1, DP1, R, USP, DSP	
PCIe DVSEC for Test Capability (see Section 14.16.1)	000Ah	0	D1	P, LD, FMLD, DP1, UP1, R, USP, DSP	D2

1. P - PCIe device, D1 - RCD, D2 - SLD, LD - Logical Device, FMLD - Fabric Manager owned LD FFFFh, UP1 - RCD Upstream Port, DP1 - RCH Downstream Port, R - CXL root port, USP - CXL Upstream Switch Port, DSP - CXL Downstream Switch Port. A physical component may be capable of operating in multiple modes. For example, a CXL device may operate either as an RCD or SLD based on the link training. In such cases, these definitions refer to the current mode of operation.
2. Non-CXL Function Map DVSEC is mandatory for CXL USPs that include a Switch Mailbox CCI as an additional Function.

8.1.2

CXL Data Object Exchange (DOE) Type Assignment

Data Object Exchange (DOE) is a PCI-SIG*-defined mechanism for the host to perform data object exchanges with a PCIe Function.

The following values of DOE Type are defined by the CXL specification. The CXL specification-defined DOE Messages use Vendor ID 1E98h.

[Table 8-3](#) in this version of the specification does not define the behavior of CXL fabric switches (see [Section 2.7](#)) and G-FAM devices (see [Section 2.8](#)).

Table 8-3. CXL DOE Type Assignment

CXL Capability	DOE Type	Mandatory ¹	Not Permitted ¹	Optional ¹
Compliance (see Chapter 14.0) ²	0	LD, FMLD	P, UP1, DP1, R, USP, DSP	D1, D2
Reserved	1			
Table Access (Coherent Device Attributes; see Section 8.1.11)	2	D2, LD, USP	FMLD, P, UP1, DP1, R, DSP	D1

1. P - PCIe device, D1 - RCD, D2 - SLD, LD - Logical Device, FMLD - Fabric Manager owned LD FFFFh, UP1 - RCD Upstream Port, DP1 - RCH Downstream Port, R - CXL root port, USP - CXL Upstream Switch Port, DSP - CXL Downstream Switch Port. A physical component may be capable of operating in multiple modes. For example, a CXL device may operate either as an RCD or SLD based on the link training. In such cases, these definitions refer to the current mode of operation.
2. eRCDs are required to implement PCIe DVSEC for Test Capability (see [Section 14.16.1](#)). For all other Devices, support for the Compliance DOE Type is highly recommended and PCIe DVSEC for Test Capability is not required if the Compliance DOE Type is implemented. If Compliance DOE Type is not implemented by a device, the device shall implement PCIe DVSEC for Test Capability (see [Section 14.16.1](#)).

8.1.3

PCIe DVSEC for CXL Devices

Note:

The CXL 1.1 specification referred to this DVSEC as “PCIe DVSEC for Flex Bus Device” and used the term “Flex Bus” while referring to various register names and fields. The CXL 2.0 specification renamed the DVSEC and the register/field names by replacing the term “Flex Bus” with the term “CXL” while retaining the functionality.

An RCD creates a new PCIe enumeration hierarchy. As such, it spawns a new Root Bus and can expose one or more PCIe device numbers and function numbers at this bus number. These are exposed as Root Complex Integrated Endpoints (RCiEP). The PCIe Configuration Space of Device 0, Function 0 shall include the CXL PCIe DVSEC as shown in [Figure 8-1](#).

A non-RCD is enumerated like a standard PCIe Endpoint and appears below a CXL Root Port or a CXL Switch. A non-RCD shall expose one PCIe device number and one or more function numbers at the parent Port's secondary bus number. These devices set PCI Express Capabilities Register.Device/Port Type=PCI Express* Endpoint and thus appear as standard PCIe Endpoints (EP). The PCIe Configuration Space of Device 0, Function 0 shall include the CXL PCIe DVSEC as shown in [Figure 8-1](#).

In either case, the capability, status, and control fields in Device 0, Function 0 DVSEC control the CXL functionality of the entire device.

Software may use the presence of this DVSEC to differentiate between a CXL device and a PCIe device. As such, a standard PCIe device must not expose this DVSEC. See [Table 8-2](#) for the complete listing.

See PCIe Base Specification for a description of the standard DVSEC register fields.

Figure 8-1.

PCIe DVSEC for CXL Devices

31	16	15	0
		PCI Express Extended Capability Header	00h
		Designated Vendor-specific Header 1	04h
	CXL Capability	Designated Vendor-specific Header 2	08h
	CXL Status	CXL Control	0Ch
	CXL Status2	CXL Control2	10h
	CXL Capability2	CXL Lock	14h
	Range 1 Size High		18h
	Range 1 Size Low		1Ch
	Range 1 Base High		20h
	Range 1 Base Low		24h
	Range 2 Size High		28h
	Range 2 Size Low		2Ch
	Range 2 Base High		30h
	Range 2 Base Low		34h
Reserved		CXL Capability3	38h

To advertise this CXL capability, the standard DVSEC register fields shall be set to the values shown in [Table 8-4](#). The DVSEC Length field is set to 03Ch bytes to accommodate the registers included in the DVSEC. The DVSEC ID is cleared to 0h to advertise that this is a PCIe DVSEC for the CXL Device structure. An RCD may implement a DVSEC Revision of 0h or higher. Devices that are not RCDs must implement a DVSEC Revision of 1h or higher.

Table 8-4.

PCIe DVSEC CXL Devices - Header

Register	Bit Location	Field	Value
Designated Vendor-Specific Header 1 (Offset 04h)	15:0	DVSEC Vendor ID	1E98h
	19:16	DVSEC Revision	3h
	31:20	DVSEC Length	03Ch
Designated Vendor-Specific Header 2 (Offset 08h)	15:0	DVSEC ID	0000h

The CXL device-specific registers are described in the following subsections.

8.1.3.1 DVSEC CXL Capability (Offset 0Ah)

Bit	Attributes	Description
0	RO	Cache_Capable: If set, indicates that the CXL.cache protocol is supported when operating in Flex Bus.CXL mode. This must be 0 for all LDs of an MLD.
1	RO	IO_Capable: If set, indicates that the CXL.io protocol is supported when operating in Flex Bus.CXL mode. Must be 1.
2	RO	Mem_Capable: If set, indicates that the CXL.mem protocol is supported when operating in Flex Bus.CXL mode. This must be 1 for all LDs of an MLD.
3	RO	Mem_HwInit_Mode: If set, indicates that this CXL.mem-capable device initializes memory with assistance from hardware and firmware located on the device. If cleared, indicates that memory is initialized by host software such as a device driver. This bit must be ignored when Mem_Capable=0. Functions that implements the Class Code specified in Section 8.1.12.1 shall set this bit to 1.
5:4	RO	HDM_Count: Number of HDM ranges implemented by the CXL device and reported through this function. This field must return 00b if Mem_Capable=0. <ul style="list-style-type: none"> • 00b = Zero ranges. This setting is illegal when Mem_Capable=1. • 01b = One HDM range. • 10b = Two HDM ranges. • 11b = Reserved.
6	RO	Cache Writeback and Invalidate Capable: If set, indicates that the device implements the Disable Caching and Initiate Cache Write Back and Invalidation control bits in the DVSEC CXL Control2 register, and the Cache Invalid status bit in the DVSEC CXL Status2 register. All devices that are not RCDs shall set this capability bit when Cache_Capable=1. ¹
7	RO	CXL Reset Capable: If set, indicates that the device supports CXL Reset and implements the CXL Reset Timeout field in this register, the Initiate CXL Reset bit in the DVSEC CXL Control2 register, and the DVSEC CXL Reset Complete status bit in the DVSEC CXL Status2 register. ¹ This bit must report the same value for all LDs of an MLD.
10:8	RO	CXL Reset Timeout: If the CXL Reset Capable bit in this register is set, this field indicates the maximum time that the device may take to complete the CXL Reset. If the CXL Reset Mem Clr Capable bit in this register is 1, this time also accounts for the time that is needed for clearing or randomizing of volatile HDM Ranges. If the CXL Reset Complete status bit in the DVSEC CXL Status2 register is not set after the passage of this time duration, software may assume that CXL Reset has failed. This value must be the same for all LDs of an MLD. ¹ <ul style="list-style-type: none"> • 000b = 10 ms • 001b = 100 ms • 010b = 1 second • 011b = 10 second • 100b = 100 second • All other encodings are reserved
11	HwInit	CXL Reset Mem Clr Capable: When set, the Device is capable of clearing or randomizing volatile HDM Ranges during CXL Reset. ¹
12	HwInit	TSP Capable: When set, the Device is capable of supporting TSP and shall support TSP requests (see Section 11.5.5) and MemRdFill (see Table 3-41). ²

Bit	Attributes	Description
13	HwInit	Multiple Logical Device: If set, indicates that the Device is a Logical Device (which could be an FM-owned LD) within an MLD. If cleared, indicates that the Device is an SLD or an RCD. ¹
14	RO	Viral_Capable: If set, indicates that the CXL device supports Viral handling. This value must be 1 for all devices.
15	HwInit	PM Init Completion Reporting Capable: If set, indicates that the CXL device is capable of supporting the Power Management Initialization Complete flag. All devices that are not RCDs shall set this capability bit. RCDs may implement this capability. ¹ This capability is not applicable to switches and root ports. Switches and root ports shall hardwire this bit to 0.

1. This bit/field was introduced as part of DVSEC Revision=1.
2. This bit/field was introduced as part of DVSEC Revision=3.

8.1.3.2 DVSEC CXL Control (Offset 0Ch)

Bit	Attributes	Description
0	RWL	Cache_Enable: When set to 1, enables CXL.cache protocol operation when in Flex Bus.CXL mode. Locked by the CONFIG_LOCK bit ¹ . If this bit is 0, the component is permitted to silently drop all CXL.cache transactions. Default value of this bit is 0.
1	RO	IO_Enable: When set to 1, enables CXL.io protocol operation when in Flex Bus.CXL mode. This bit always returns 1.
2	RWL	Mem_Enable: When set to 1, enables CXL.mem protocol operation when in Flex Bus.CXL mode. Locked by the CONFIG_LOCK bit ¹ . If this bit is 0, the component is permitted to silently drop all CXL.mem transactions. Default value of this bit is 0.
7:3	RWL	Cache_SF_Coverage: Performance hint to the device. Locked by the CONFIG_LOCK bit ¹ . <ul style="list-style-type: none"> • 00h = Indicates no Snoop Filter coverage on the host • For all other values of N = Indicates Snoop Filter coverage on the host of $2^{(N+15d)}$ bytes (e.g., value of 5h indicates 1-MB snoop filter coverage) Default value of this field is 00h.
10:8	RWL	Cache_SF_Granularity: Performance hint to the device. Locked by the CONFIG_LOCK bit ¹ . <ul style="list-style-type: none"> • 000b = Indicates 64B granular tracking on the host • 001b = Indicates 128B granular tracking on the host • 010b = Indicates 256B granular tracking on the host • 011b = Indicates 512B granular tracking on the host • 100b = Indicates 1KB granular tracking on the host • 101b = Indicates 2KB granular tracking on the host • 110b = Indicates 4KB granular tracking on the host • 111b = Reserved Default value of this field is 000b.
11	RWL	Cache_Clean_Eviction: Performance hint to the device. Locked by the CONFIG_LOCK bit ¹ . <ul style="list-style-type: none"> • 0 = Indicates clean evictions from device caches are needed for best performance • 1 = Indicates clean evictions from device caches are NOT needed for best performance Default value of this bit is 0.

Bit	Attributes	Description
12	RWL/RsvdP	Direct P2P Mem Enable: This bit must be RWL if the Direct P2P Mem Capable bit in the DVSEC CXL Capability3 register is set; otherwise, this bit is permitted to be hardwired to 0. Software must not set this bit unless the Direct P2P Mem Capable bit is set. ² When set, enables Direct P2P CXL.mem protocol operation. If this bit is 0, the component is not permitted to initiate Direct P2P CXL.mem transactions. Default value of this bit is 0. Locked by the CONFIG_LOCK bit ¹ .
13	RsvdP	Reserved
14	RWL	Viral_Enable: When set, enables Viral handling in the CXL device. Locked by the CONFIG_LOCK bit ¹ . If 0, the CXL device may ignore the viral that it receives. Default value of this bit is 0.
15	RsvdP	Reserved

1. CONFIG_LOCK bit in the DVSEC CXL Lock register.
2. This bit was introduced as part of DVSEC Revision=3.

8.1.3.3 DVSEC CXL Status (Offset 0Eh)

Bit	Attributes	Description
13:0	RsvdZ	Reserved
14	RW1CS	Viral_Status: When set, indicates that the CXL device has encountered a Viral condition. This bit does not indicate that the device is currently in Viral condition. See Section 12.4 for more details.
15	RsvdZ	Reserved

8.1.3.4 DVSEC CXL Control2 (Offset 10h)

Bit	Attributes	Description
0	RW	Disable Caching: When set to 1, device shall no longer cache new modified lines in its local cache. Device shall continue to correctly respond to CXL.cache transactions. ¹ Default value of this bit is 0.
1	RW	Initiate Cache Write Back and Invalidations: When set to 1, the device shall write back all modified lines in the local cache and then invalidate all lines. The device shall send a CacheFlushed message to the host, as required by CXL.cache protocol, to indicate that the device does not hold any modified lines. ¹ If this bit is set when Disable Caching=0, the device behavior is undefined. This bit always returns the value of 0 when read by the software. A write of 0 is ignored.
2	RW	Initiate CXL Reset: When set to 1, the device shall initiate CXL Reset as defined in Section 9.7. This bit always returns the value of 0 when read by the software. A write of 0 is ignored. ¹ If Software sets this bit while the previous CXL Reset is in progress, the results are undefined.
3	RW	CXL Reset Mem Clr Enable: When set, and the CXL Reset Mem Clr Capable bit in the DVSEC CXL Capability register returns 1, the device shall clear or randomize volatile HDM ranges as part of the CXL Reset operation. When the CXL Reset Mem Clr Capable bit is cleared, this bit is ignored and volatile HDM ranges may or may not be cleared or randomized during CXL Reset. ¹ Default value of this bit is 0.

Bit	Attributes	Description
4	RWS/RO	<p>Desired Volatile HDM State after Hot Reset: This bit must be RWS if the Volatile HDM State after Hot Reset - Configurability bit in the DVSEC CXL Capability3 register is set; otherwise, this bit is permitted to be hardwired to 0. Software must not set this bit unless the Volatile HDM State after Hot Reset - Configurability bit is set.²</p> <p>The reset default is 0.</p> <ul style="list-style-type: none"> • 0 = Follow the Default Volatile HDM State after the Hot Reset bit in the DVSEC CXL Capability3 register • 1 = Device shall preserve the Volatile HDM content across Hot Reset
5	RW/RO	<p>Modified Completion Enable: This bit must be RW if the Modified Completion Capable bit in the DVSEC CXL Capability2 register is set; otherwise, this bit is permitted to be hardwired to 0. Software must not set this bit unless the Modified Completion Capable bit is set.³</p> <p>The reset default is 0.</p> <ul style="list-style-type: none"> • 0 = This device is not permitted to return modified data • 1 = This device is permitted to return modified data using the Cmp-M response
15:6	RsvdP	Reserved

1. This bit was introduced as part of DVSEC Revision=1.
2. This bit was introduced as part of DVSEC Revision=2.
3. This bit was introduced as part of DVSEC Revision=3.

8.1.3.5 DVSEC CXL Status2 (Offset 12h)

Bit	Attributes	Description
0	RO	<p>Cache Invalid: When set, the device guarantees that it does not hold any valid lines and Disable Caching=1¹. This bit shall read as 0 when Disable Caching=0.²</p>
1	RO	<p>CXL Reset Complete: When set, the device has successfully completed CXL Reset as defined in Section 9.7.²</p> <p>Device shall clear this bit upon transition of Initiate CXL Reset bit¹ from 0 to 1, prior to initiating the CXL Reset flow.</p>
2	RO	<p>CXL Reset Error: When set, the device has completed CXL Reset with errors. Additional information may be available in device error records (see Section 8.2.9.2.1). Host software or Fabric Manager may optionally reissue CXL Reset.²</p> <p>Device shall clear this bit upon transition of the Initiate CXL Reset bit¹ from 0 to 1, prior to initiating the CXL Reset flow.</p>
3	RW1CS/ RsvdZ	<p>Volatile HDM Preservation Error: This bit shall be set if the Software requested the device to preserve Volatile HDM content across a Hot Reset but the device failed to do so.³</p> <p>RW1CS if the Volatile HDM State after Hot Reset - Configurability bit in the DVSEC CXL Capability3 register is set; otherwise, it is RsvdZ.</p>
14:4	RsvdZ	Reserved
15	RO	<p>Power Management Initialization Complete: When set, indicates that the device has successfully completed the Power Management Initialization flow described in Figure 3-4 and is ready to process various Power Management messages.²</p> <p>If this bit is not set within 100 ms of link-up, software may conclude that Power Management initialization has failed and may then issue a Secondary Bus Reset to force link re-initialization and Power Management re-initialization.</p>

1. Bit in the DVSEC CXL Control2 register.
2. This bit was introduced as part of DVSEC Revision=1.
3. This bit was introduced as part of DVSEC Revision=2.

8.1.3.6 DVSEC CXL Lock (Offset 14h)

Bit	Attributes	Description
0	RWO	<p>CONFIG_LOCK: When set, all register fields in the PCIe DVSEC for CXL Devices Capability with the RWL attribute become read only. Consult individual register fields for details.</p> <p>This bit is cleared upon device Conventional Reset. This bit and all the fields that are locked by this bit are unaffected by CXL Reset.</p> <p>Default value of this bit is 0.</p>
15:1	RsvdP	Reserved

8.1.3.7 DVSEC CXL Capability2 (Offset 16h)

Bit	Attributes	Description
3:0	RO	<p>Cache Size Unit: A CXL device that is not CXL.cache-capable shall return the value of 0h.¹</p> <ul style="list-style-type: none"> • 0h = Cache size is not reported • 1h = 64 KB • 2h = 1 MB • All other encodings are reserved
5:4	HwInit	<p>Fallback Capability: Defines the fallback operation mode of a Type 2 Device. Fallback operation mode is where the device does not appear as a Type 2 CXL device, yet provides useful functionality. This field is not intended for advertising debug modes of operation.²</p> <ul style="list-style-type: none"> • 00b = Device either does not support fallback mode or does not advertise fallback mode • 01b = PCIe • 10b = CXL Type 1 • 11b = CXL Type 3
6	HwInit	<p>Modified Completion Capable: When set to 1, it indicates that this device is capable of returning modified data using the Cmp-M response.³</p>
7	HwInit	<p>No Clean Writeback: Specifies that a device shall not issue clean writebacks. This bit shall be set to 1 if the device does not support CXL.cache and does not support Direct P2P CXL.mem as a requester. For DVSEC Revisions = 1h or 2h, software can consider the device 'No Clean Writeback' capable if Cache_Capable is not set.³</p> <ul style="list-style-type: none"> • 0 = Device may or may not generate clean writebacks • 1 = Device guarantees to never generate clean writebacks at the device's cacheline granularity
15:8	RO	<p>Cache Size: Expressed in multiples of Cache Size Unit. If Cache Size=4 and Cache Size Unit=1h, the device has a 256-KB cache.¹</p> <p>A CXL device that is not CXL.cache-capable shall return the value of 00h.</p>

1. This field was introduced as part of DVSEC Revision=1.

2. This field was introduced as part of DVSEC Revision=2.

3. This bit was introduced as part of DVSEC Revision=3.

8.1.3.8 DVSEC CXL Range Registers

These registers are not applicable to an FM-owned LD.

The DVSEC CXL Range 1 register set must be implemented if Mem_Capable=1 in the DVSEC CXL Capability register. The DVSEC CXL Range 2 register set must be implemented if (Mem_Capable=1 and HDM_Count=10b in the DVSEC CXL Capability register). Each set contains 4 registers - Size High, Size Low, Base High, and Base Low.

A CXL.mem-capable device is permitted to report zero memory size.

8.1.3.8.1 DVSEC CXL Range 1 Size High (Offset 18h)

Bit	Attributes	Description
31:0	RO	Memory_Size_High: Corresponds to bits 63:32 of the CXL Range 1 memory size regardless of whether the device implements CXL HDM Decoder Capability registers.

8.1.3.8.2 DVSEC CXL Range 1 Size Low (Offset 1Ch)

Bit	Attributes	Description
0	RO	Memory_Info_Valid: When set, indicates that the CXL Range 1 Size High and Size Low registers are valid regardless of whether the device implements CXL HDM Decoder Capability registers. Must be set within 1 second of reset deassertion to the CXL device.
1	RO	Memory_Active: When set, indicates that the CXL Range 1 memory is fully initialized and available for software use regardless of whether the device implements CXL HDM Decoder Capability registers. When cleared, indicates that the CXL Range 1 memory may be unavailable for software use regardless of whether the device implements CXL HDM Decoder Capability registers. Must be set within Range 1 Memory_Active_Timeout of reset deassertion to the CXL device when Mem_HwInit_Mode=1 in the DVSEC CXL Capability register.
4:2	RO	Media_Type: Indicates the memory media characteristics regardless of whether the device implements CXL HDM Decoder Capability registers. All CXL.mem devices that are not eRCDs shall set this field to 010b. <ul style="list-style-type: none"> • 000b = Volatile memory. This setting is deprecated starting with the CXL 2.0 specification. • 001b = Non-volatile memory. This setting is deprecated starting with the CXL 2.0 specification. • 010b = Memory characteristics are communicated via CDAT (see Section 8.1.11) and not via this field.¹ • All other encodings are reserved.
7:5	RO	Memory_Class: Indicates the class of memory regardless of whether the device implements CXL HDM Decoder Capability registers. All CXL.mem devices that are not eRCDs shall set this field to 010b. <ul style="list-style-type: none"> • 000b = Memory Class (e.g., normal DRAM). This setting is deprecated starting with the CXL 2.0 specification. • 001b = Storage Class. This setting is deprecated starting with the CXL 2.0 specification. • 010b = Memory characteristics are communicated via CDAT (see Section 8.1.11) and not via this field.¹ • All other encodings are reserved.

Bit	Attributes	Description
12:8	RO	<p>Desired_Interleave: If a CXL.mem-capable eRCD is connected to a single CPU via multiple CXL links, this field represents the memory interleaving desired by the device. BIOS will configure the CPU to interleave accesses to this HDM range across links at this granularity or to the closest possible value that the host supports.</p> <p>In all other cases, this field represents the minimum desired interleave granularity for optimal device performance regardless of whether the device implements CXL HDM Decoder Capability registers. Software should program the Interleave Granularity (IG) field in the HDM Decoder Control registers (see Section 8.2.4.20.7) to be an exact match or any larger granularity than the device advertises via the CXL HDM Decoder Capability register (see Section 8.2.4.20.1). This field is treated as a hint. The device shall function correctly if the actual value that is programmed in the Interleave Granularity (IG) field in the HDM Decoder Control registers is less than what is reported via this field.</p> <ul style="list-style-type: none"> • 00h = No Interleave • 01h = 256-Byte Granularity • 02h = 4-KB Interleave • 03h = 512 Bytes¹ • 04h = 1024 Bytes¹ • 05h = 2048 Bytes¹ • 06h = 8192 Bytes¹ • 07h = 16384 Bytes¹ • All other encodings are reserved <p>Note: If a CXL device has different desired interleave values for DPA ranges that are covered by this CXL Range 1, the device should report a value that best fits the requirements for all such ranges (e.g., the maximum of the values).</p> <p>Note: If CXL devices in an Interleave Set advertise different values for this field, Software may choose the smallest value that best fits the set.</p>
15:13	HwInit	<p>Memory_Active_Timeout: For devices that advertise Mem_HwInit_Mode=1 in the DVSEC CXL Capability register, this field indicates the maximum time that the device is permitted to take to set the Memory_Active bit in this register after a hot reset, a warm reset, or a cold reset regardless of whether the device implements CXL HDM Decoder Capability registers. If the Memory_Active bit is not set after the passage of this time duration, software may assume that the HDM reported by this range has failed. This value must be the same for all LDs of an MLD.¹</p> <ul style="list-style-type: none"> • 000b = 1 second • 001b = 4 seconds • 010b = 16 seconds • 011b = 64 seconds • 100b = 256 seconds • All other encodings are reserved
16	RO	<p>Memory_Active_Degraded: When set, indicates that the CXL Range 1 memory is initialized and available for software use regardless of whether the device implements CXL HDM Decoder Capability registers. When set, it also signifies a reduction in capacity or performance relative to what is expected.²</p> <p>If this bit is 1, the Memory_Active flag in this register shall be 0. If the Memory_Active flag in this register is 1, this bit shall be 0.</p> <p>Either Memory_Active or Memory_Active_Degraded shall be set within Range_1 Memory_Active_Timeout of reset deassertion to the CXL device when Mem_HwInit_Mode=1 in the DVSEC CXL Capability register.</p>
27:17	RsvdP	Reserved
31:28	RO	Memory_Size_Low: Corresponds to bits 31:28 of the CXL Range 1 memory size regardless of whether the device implements CXL HDM Decoder Capability registers.

1. Introduced as part of DVSEC Revision=1.
2. This bit was introduced as part of DVSEC Revision=3.

8.1.3.8.3 DVSEC CXL Range 1 Base High (Offset 20h)

Bit	Attributes	Description
31:0	RWL	<p>Memory_Base_High: Corresponds to bits 63:32 of CXL Range 1 base in the host address space. Locked by the CONFIG_LOCK bit in the DVSEC CXL Lock register.</p> <p>If a device implements CXL HDM Decoder Capability registers and software has enabled the HDM Decoder by setting the HDM Decoder Enable bit in the CXL HDM Decoder Global Control register, the value of this register is not used during address decode. It is recommended that software program this to match CXL HDM Decoder 0 Base High register for backward compatibility.</p> <p>Default value of this register is 0h.</p>

8.1.3.8.4 DVSEC CXL Range 1 Base Low (Offset 24h)

Bit	Attributes	Description
27:0	RsvdP	Reserved.
31:28	RWL	<p>Memory_Base_Low: Corresponds to bits 31:28 of the CXL Range 1 base in the host address space. Locked by the CONFIG_LOCK bit in the DVSEC CXL Lock register.</p> <p>If a device implements CXL HDM Decoder Capability registers and software has enabled the HDM Decoder by setting the HDM Decoder Enable bit in the CXL HDM Decoder Global Control register, the value of this field is not used during address decode. It is recommended that software program this to match CXL HDM Decoder 0 Base Low register for backward compatibility.</p> <p>Default value of this field is 0h.</p>

A CXL.mem-capable device that does not implement CXL HDM Decoder Capability registers directs host accesses to an Address A within its local HDM if the following two equations are satisfied:

Equation 8-1.

$$\text{Memory_Base}[63:28] \leq (\text{A} \gg 28) < \text{Memory_Base}[63:28]+\text{Memory_Size}[63:28]$$

Equation 8-2.

$$\text{Memory_Active AND DVSEC CXL Mem_Enable}=1$$

where \gg represents a bitwise right-shift operation.

A CXL.mem-capable device that implements CXL HDM Decoder Capability registers follows the above behavior as long as the HDM Decoder Enable bit in the CXL HDM Decoder Global Control register (see [Section 8.2.4.20.2](#)) is 0.

Note:

Software is required to set HDM Decoder Enable bit in the CXL HDM Decoder Global Control register to enable the device to generate a BISnP request or allow UIO access to its HDM. Under these scenarios, the DVSEC CXL Range 1 Base Low register, DVSEC CXL Range 1 Base High register, DVSEC CXL Range 2 Base Low register, and DVSEC CXL Range 2 Base High register do not participate in CXL.mem address decode.

If Address A is not backed by real memory (e.g., a device with less than 256 MB of memory), a device that does not implement CXL HDM Decoder Capability registers must gracefully handle those accesses (i.e., return all 1s on reads and drop writes).

Aliasing (mapping more than one Host Physical Address (HPA) to a single Device Physical Address) is forbidden.

8.1.3.8.5 DVSEC CXL Range 2 Size High (Offset 28h)

Bit	Attributes	Description
31:0	RO	Memory_Size_High: Corresponds to bits 63:32 of the CXL Range 2 memory size regardless of whether the device implements CXL HDM Decoder Capability registers.

8.1.3.8.6 DVSEC CXL Range 2 Size Low (Offset 2Ch)

Bit	Attributes	Description
0	RO	Memory_Info_Valid: When set, indicates that the CXL Range 2 Size High and Size Low registers are valid regardless of whether the device implements CXL HDM Decoder Capability registers. Must be set within 1 second of reset deassertion to the CXL device.
1	RO	Memory_Active: When set, indicates that the CXL Range 2 memory is fully initialized and available for software use, regardless of whether the device implements CXL HDM Decoder Capability registers. When cleared, indicates that the CXL Range 2 memory may be unavailable for software use regardless of whether the device implements CXL HDM Decoder Capability registers. Must be set within Range 2 Memory_Active_Timeout of reset deassertion to the CXL device when Mem_HwInit_Mode=1 in the DVSEC CXL Capability register.
4:2	RO	Media_Type: Indicates the memory media characteristics regardless of whether the device implements CXL HDM Decoder Capability registers. All CXL.mem devices that are not eRCDs shall set this field to 010b. <ul style="list-style-type: none"> 000b = Volatile memory. This setting is deprecated starting with the CXL 2.0 specification. 001b = Non-volatile memory. This setting is deprecated starting with the CXL 2.0 specification. 010b = Memory characteristics are communicated via CDAT (see Section 8.1.11) and not via this field.¹ 111b = Not Memory. This setting is deprecated starting with the CXL 2.0 specification. All other encodings are reserved.
7:5	RO	Memory_Class: Indicates the class of memory regardless of whether the device implements CXL HDM Decoder Capability registers. All CXL.mem devices that are not eRCDs shall set this field to 010b. <ul style="list-style-type: none"> 000b = Memory Class (e.g., normal DRAM), This setting is deprecated starting with the CXL 2.0 specification. 001b = Storage Class. This setting is deprecated starting with the CXL 2.0 specification. 010b = Memory characteristics are communicated via CDAT (see Section 8.1.11) and not via this field.¹ All other encodings are reserved.
12:8	RO	Desired_Interleave: See the Desired_Interleave field definition in the DVSEC CXL Range 1 Size Low register (see Section 8.1.3.8.2).

Bit	Attributes	Description
15:13	HwInit	<p>Memory_Active_Timeout: For devices that advertises Mem_HwInit_Mode=1 in the DVSEC CXL Capability register, this field indicates the maximum time that the device is permitted to take to set the Memory_Active bit in this register after a Conventional Reset regardless of whether the device implements CXL HDM Decoder Capability registers. If the Memory_Active bit is not set after the passage of this time duration, software may assume that the HDM reported by this range has failed. This value must be the same for all LDs of an MLD.¹</p> <ul style="list-style-type: none"> • 000b = 1 second • 001b = 4 seconds • 010b = 16 seconds • 011b = 64 seconds • 100b = 256 seconds • All other encodings are reserved
16	RO	<p>Memory_Active_Degraded: When set, indicates that the CXL Range 2 memory is initialized and available for software use regardless of whether the device implements CXL HDM Decoder Capability registers. When set, it also signifies a reduction in capacity or performance relative to what is expected.²</p> <p>If this bit is 1, the Memory_Active flag in this register shall be 0. If the Memory_Active flag in this register is 1, this bit shall be 0.</p> <p>Either Memory_Active or Memory_Active_Degraded shall be set within Range_2 Memory_Active_Timeout of reset deassertion to the CXL device when Mem_HwInit_Mode=1 in the DVSEC CXL Capability register.</p>
27:17	RsvdP	Reserved
31:28	RO	Memory_Size_Low: Corresponds to bits 31:28 of the CXL Range 2 memory size regardless of whether the device implements CXL HDM Decoder Capability registers.

1. Introduced as part of DVSEC Revision=1.
2. This bit was introduced as part of DVSEC Revision=3.

8.1.3.8.7 DVSEC CXL Range 2 Base High (Offset 30h)

Bit	Attributes	Description
31:0	RWL	<p>Memory_Base_High: Corresponds to bits 63:32 of CXL Range 2 base in the host address space. Locked by the CONFIG_LOCK bit in the DVSEC CXL Lock register.</p> <p>If a device implements CXL HDM Decoder Capability registers and software has enabled the HDM Decoder by setting the HDM Decoder Enable bit in the CXL HDM Decoder Global Control register, the value of this register is not used during address decode. It is recommended that software program this to match the corresponding CXL HDM Decoder Base High register for backward compatibility.</p> <p>Default value of this register is 0000 0000h.</p>

8.1.3.8.8 DVSEC CXL Range 2 Base Low (Offset 34h)

Bit	Attributes	Description
27:0	RsvdP	Reserved
31:28	RWL	<p>Memory_Base_Low: Corresponds to bits 31:28 of the CXL Range 2 base in the host address space. Locked by the CONFIG_LOCK bit in the DVSEC CXL Lock register.</p> <p>If a device implements CXL HDM Decoder Capability registers and software has enabled the HDM Decoder by setting the HDM Decoder Enable bit in the CXL HDM Decoder Global Control register, the value of this field is not used during address decode. It is recommended that software program this to match the corresponding CXL HDM Decoder Base Low register for backward compatibility.</p> <p>Default value of this field is 0h.</p>

8.1.3.9 DVSEC CXL Capability3 (Offset 38h)

Bit	Attributes	Description ¹
0	HwInit	Default Volatile HDM State after Cold Reset² <ul style="list-style-type: none"> • 0 = The Volatile HDM content after a Cold Reset is undefined. The content may or may not be cleared. The content may or may not be randomized. • 1 = The device shall clear or randomize the volatile HDM content after a Cold reset. The clear or randomize operation shall be completed before Memory_Active is set.
1	HwInit	Default Volatile HDM State after Warm Reset² <ul style="list-style-type: none"> • 0 = The Volatile HDM content after a Warm Reset is undefined. The content may or may not be cleared. The content may or may not be randomized. • 1 = The device shall clear or randomize the volatile HDM content after a Warm Reset. The clear or randomize operation shall be completed before Memory_Active is set.
2	HwInit	Default Volatile HDM State after Hot Reset² <ul style="list-style-type: none"> • 0 = The Volatile HDM content after a Hot Reset is undefined. The content may or may not be cleared. The content may or may not be randomized. • 1 = The device shall clear or randomize the volatile HDM content after a Hot Reset. The clear or randomize operation shall be completed before Memory_Active is set. <p>If the Volatile HDM State after Hot Reset - Configurability bit is set, the software is permitted to override the Default State and request that the memory be preserved across a Hot Reset.</p>
3	HwInit	Volatile HDM State after Hot Reset - Configurability² <ul style="list-style-type: none"> • 0 = The device does not support preservation of Volatile HDM State across Hot Reset • 1 = The device supports preservation of Volatile HDM State across a Hot Reset. The Software may request the device to preserve Volatile HDM content across a Hot Reset by setting the Desired Volatile HDM State after Hot Reset bit prior to the Hot Reset event.
4	HwInit	Direct P2P Mem Capable: If set, indicates that the Direct P2P CXL.mem protocol is supported. ³
15:5	RsvdP	Reserved

1. This register was added as part of DVSEC Revision=2.

2. This bit was introduced as part of DVSEC Revision=2.

3. This bit was introduced as part of DVSEC Revision=3.

8.1.4 Non-CXL Function Map DVSEC

Figure 8-2. Non-CXL Function Map DVSEC

	31	16	15	0
PCI Express Extended Capability Header				00h
Designated Vendor-specific Header 1				04h
Reserved		Designated Vendor-specific Header 2		08h
				0Ch
Non-CXL Function Map Register 0				10h
Non-CXL Function Map Register 1				14h
Non-CXL Function Map Register 2				18h
Non-CXL Function Map Register 3				1Ch
Non-CXL Function Map Register 4				20h
Non-CXL Function Map Register 5				24h
Non-CXL Function Map Register 6				28h
Non-CXL Function Map Register 7				

This DVSEC capability identifies the list of device and function numbers associated with non-virtual functions (i.e., functions that are not a Virtual Function) implemented by CXL device that are not capable of participating in CXL.cache/CXL.mem protocol. The PCIe Configuration Space of Device 0, Function 0 of a CXL device may include Non-CXL Function Map DVSEC as shown in [Figure 8-2](#). See [Table 8-2](#) for the complete listing. To advertise this capability, the standard DVSEC register fields must be set to the values shown in [Table 8-5](#). The DVSEC Length field must be set to 02Ch bytes to accommodate the registers included in the DVSEC. The DVSEC ID must be set to 0002h to advertise that this is a Non-CXL Function Map DVSEC capability structure for CXL ports.

If this DVSEC capability is present, it must be included in Device 0, Function 0 of a CXL device.

Absence of Non-CXL Function Map DVSEC indicates that PCIe DVSEC for CXL devices ([Section 8.1.3](#)) located on Device 0, Function 0 governs whether all Functions participate in CXL.cache and CXL.mem protocol.

Table 8-5. Non-CXL Function Map DVSEC - Header

Register	Bit Location	Field	Value
Designated Vendor-Specific Header 1 (Offset 04h)	15:0	DVSEC Vendor ID	1E98h
	19:16	DVSEC Revision	0h
	31:20	DVSEC Length	02Ch
Designated Vendor-Specific Header 2 (Offset 08h)	15:0	DVSEC ID	0002h

8.1.4.1 Non-CXL Function Map Register 0 (Offset 0Ch)

Bit	Attributes	Description
31:0	HwInit	<p>Non CXL Function: Each bit represents a non-virtual function number implemented by the device on the same bus as the physical function that carries PCIe DVSEC for CXL devices.</p> <p>When a bit is set, the corresponding Device/Function number or Function number (ARI device) is not capable of participating in CXL.cache or CXL.mem protocol. Bits corresponding to Non-existent Device/Function or Function numbers shall always return 0.</p> <p>If the device does not support ARI, bit x in this register maps to Device x, Function 0. If the device supports ARI, bit x in this register maps to Function x.</p> <p>Bit 0 of this register shall always be cleared to 0 since PCIe DVSEC for CXL devices declares whether Device 0, Function 0 participates in CXL.cache and CXL.mem protocol.</p>

8.1.4.2 Non-CXL Function Map Register 1 (Offset 10h)

Bit	Attributes	Description
31:0	HwInit	<p>Non CXL Function: Each bit represents a non-virtual function number implemented by the device on the same bus as the physical function that carries PCIe DVSEC for CXL devices.</p> <p>When a bit is set, the corresponding Device/Function number or Function number (ARI device) is not capable of participating in CXL.cache or CXL.mem protocol. Bits corresponding to Non-existent Device/Function or Function numbers shall always return 0.</p> <p>If the device does not support ARI, bit x in this register maps to Device x, Function 1. If the device supports ARI, bit x in this register maps to Function x+32.</p>

8.1.4.3 Non-CXL Function Map Register 2 (Offset 14h)

Bit	Attributes	Description
31:0	HwInit	<p>Non CXL Function: Each bit represents a non-virtual function number implemented by the device on the same bus as the physical function that carries PCIe DVSEC for CXL devices.</p> <p>When a bit is set, the corresponding Device/Function number or Function number (ARI device) is not capable of participating in CXL.cache or CXL.mem protocol. Bits corresponding to Non-existent Device/Function or Function numbers shall always return 0.</p> <p>If the device does not support ARI, bit x in this register maps to Device x, Function 2. If the device supports ARI, bit x in this register maps to Function (x+64).</p>

8.1.4.4 Non-CXL Function Map Register 3 (Offset 18h)

Bit	Attributes	Description
31:0	HwInit	<p>Non CXL Function: Each bit represents a non-virtual function number implemented by the device on the same bus as the physical function that carries PCIe DVSEC for CXL devices.</p> <p>When a bit is set, the corresponding Device/Function number or Function number (ARI device) is not capable of participating in CXL.cache or CXL.mem protocol. Bits corresponding to Non-existent Device/Function or Function numbers shall always return 0.</p> <p>If the device does not support ARI, bit x in this register maps to Device x, Function 3. If the device supports ARI, bit x in this register maps to Function (x+96).</p>

8.1.4.5 Non-CXL Function Map Register 4 (Offset 1Ch)

Bit	Attributes	Description
31:0	HwInit	<p>Non CXL Function: Each bit represents a non-virtual function number implemented by the device on the same bus as the physical function that carries PCIe DVSEC for CXL devices.</p> <p>When a bit is set, the corresponding Device/Function number or Function number (ARI device) is not capable of participating in CXL.cache or CXL.mem protocol. Bits corresponding to Non-existent Device/Function or Function numbers shall always return 0.</p> <p>If the device does not support ARI, bit x in this register maps to Device x, Function 4. If the device supports ARI, bit x in this register maps to Function (x+128).</p>

8.1.4.6 Non-CXL Function Map Register 5 (Offset 20h)

Bit	Attributes	Description
31:0	HwInit	<p>Non CXL Function: Each bit represents a non-virtual function number implemented by the device on the same bus as the physical function that carries PCIe DVSEC for CXL devices.</p> <p>When a bit is set, the corresponding Device/Function number or Function number (ARI device) is not capable of participating in CXL.cache or CXL.mem protocol. Bits corresponding to Non-existent Device/Function or Function numbers shall always return 0.</p> <p>If the device does not support ARI, bit x in this register maps to Device x, Function 5. If the device supports ARI, bit x in this register maps to Function (x+160).</p>

8.1.4.7 Non-CXL Function Map Register 6 (Offset 24h)

Bit	Attributes	Description
31:0	HwInit	<p>Non CXL Function: Each bit represents a non-virtual function number implemented by the device on the same bus as the physical function that carries PCIe DVSEC for CXL devices.</p> <p>When a bit is set, the corresponding Device/Function number or Function number (ARI device) is not capable of participating in CXL.cache or CXL.mem protocol. Bits corresponding to Non-existent Device/Function or Function numbers shall always return 0.</p> <p>If the device does not support ARI, bit x in this register maps to Device x, Function 6. If the device supports ARI, bit x in this register maps to Function (x+192).</p>

8.1.4.8 Non-CXL Function Map Register 7 (Offset 28h)

Bit	Attributes	Description
31:0	HwInit	<p>Non CXL Function: Each bit represents a non-virtual function number implemented by the device on the same bus as the physical function that carries PCIe DVSEC for CXL devices.</p> <p>When a bit is set, the corresponding Device/Function number or Function number (ARI device) is not capable of participating in CXL.cache or CXL.mem protocol. Bits corresponding to Non-existent Device/Function or Function numbers shall always return 0.</p> <p>If the device does not support ARI, bit x in this register maps to Device x, Function 7. If the device supports ARI, bit x in this register maps to Function (x+224).</p>

8.1.5 CXL Extensions DVSEC for Ports

Figure 8-3. CXL Extensions DVSEC for Ports

31	16	15	0
PCI Express Extended Capability Header			
Designated Vendor-specific Header 1			
CXL Port Extension Status		Designated Vendor-specific Header 2	
Alt Bus Limit	Alt Bus Base	Port Control Extensions	
Alternate Memory Limit		Alternate Memory Base	
Alt Prefetch Memory Limit		Alt Prefetch Memory Base	
Alternate Prefetchable Memory Base High			
Alternate Prefetchable Memory Limit High			
CXL RCRB Base			
CXL RCRB Base High			

The PCIe Configuration Space of a CXL root port, CXL Downstream Switch Port, and CXL Upstream Switch Port must implement this DVSEC capability as shown in [Figure 8-3](#). See [Table 8-2](#) for the complete listing. To advertise this capability, the standard DVSEC register fields must be set to the values shown in [Table 8-6](#). The DVSEC Length field must be set to 028h bytes to accommodate the registers included in the DVSEC. The DVSEC ID must be set to 0003h to advertise that this is a CXL Extension DVSEC capability structure for CXL ports.

Table 8-6.

CXL Extensions DVSEC for Ports - Header

Register	Bit Location	Field	Value
Designated Vendor-Specific Header 1 (Offset 04h)	15:0	DVSEC Vendor ID	1E98h
	19:16	DVSEC Revision	0h
	31:20	DVSEC Length	028h
Designated Vendor-Specific Header 2 (Offset 08h)	15:0	DVSEC ID	0003h

8.1.5.1 CXL Port Extension Status (Offset 0Ah)

Bit	Attributes	Description
0	RO	Port Power Management Initialization Complete: When set, it indicates that the root port, the Upstream Switch Port or the Downstream Switch Port has successfully completed the Power Management Initialization Flow as described in Figure 3-4 and is ready to process various Power Management events. If this bit is not set within 100 ms of link-up, software may conclude that the Power Management initialization has failed and may issue Secondary Bus Reset to force link re-initialization and Power Management re-initialization.
13:1	RsvdP	Reserved
14	RW1CS	Viral Status: When set, indicates that the Upstream Switch Port or the Downstream Switch Port has entered Viral (see Section 12.4 for more details). This bit is not applicable to Root Ports, and reads shall return the value of 0.
15	RsvdP	Reserved

8.1.5.2 Port Control Extensions (Offset 0Ch)

Bit	Attributes	Description
0	RW	<p>Unmask SBR: When 0, SBR bit in Bridge Control register of this Port has no effect. When 1, the Port shall generate hot reset when SBR bit in Bridge Control gets set to 1.</p> <p>Default value of this bit is 0.</p> <p>When the Port is operating in PCIe mode or RCD mode, this field has no effect on SBR functionality and Port shall follow PCIe Base Specification.</p>
1	RW	<p>Unmask Link Disable: When 0, Link Disable bit in Link Control register of this Port has no effect.</p> <p>When 1, the Port shall disable the CXL Link when Link Disable bit in Link Control gets set to 1 and Link is re-enabled when Link Disable bit in Link control is cleared to 0.</p> <p>Default value of this bit is 0.</p> <p>When the Port is operating in PCIe mode or RCD mode, this field has no effect on Link Disable functionality and the Port shall follow PCIe Base Specification.</p>
2	RW	<p>Alt Memory and ID Space Enable: When set to 1, the Port positively decodes downstream transactions to ranges specified in Alternate Memory Base/Limit registers, Alternate Prefetchable Memory Base/Limit, Alternate Prefetchable Base/Limit Upper 32 Bits and Alternate Bus Base/Limit registers regardless of the Memory Space Enable bit in the PCI* Command register.</p> <p>When cleared to 0, the Port does not decode downstream transactions to ranges specified in Alternate Memory Base/Limit registers, Alternate Prefetchable Memory Base/Limit, Alternate Prefetchable Base/Limit Upper 32 Bits and Alternate Bus Base/Limit registers.</p> <p>Default value of this bit is 0.</p> <p>Firmware/Software must ensure this bit is 0 when the Port is operating in PCIe mode.</p>
3	RW	<p>Alt BME: This bit overrides the state of BME bit in Command register if the requester's bus number is within the range specified by Alternate Bus Base and Alternate Bus Limit range.</p> <p>This bit alone controls forwarding of Memory or I/O Requests by a Port in the Upstream direction if the requester's bus number is within the range specified by Alternate Bus Base and Alternate Bus Limit range.</p> <p>If the requester's bus number is within the range specified by Alternate Bus Base and Alternate Bus Limit range and this bit is 0, Memory and I/O Requests received at a Root Port or the Downstream side of a Switch Port must be handled as Unsupported Requests (UR), and for Non-Posted Requests a Completion with UR Completion Status must be returned. This bit does not affect forwarding of Completions in either the Upstream or Downstream direction.</p> <p>Default value of this bit is 0.</p> <p>Firmware/Software must ensure this bit is 0 when the Port is operating in PCIe mode.</p>
4	RW/RsvdP	<p>UIO To HDM Enable</p> <ul style="list-style-type: none"> DSP that is capable of UIO Direct P2P accesses to HDM: This bit is RW. If 0, return Completer Abort to UIO accesses with Complete of Partial Match. See Table 9-18 for details. The default value of this bit is 0. All others: This bit is RsvdP. It is permitted to be hardwired to 0 and software must not set this bit.
13:5	RsvdP	Reserved
14	RW	<p>Viral Enable: When set, enables Viral generation functionality of the Upstream Switch Port or the Downstream Switch Port. See Section 12.4 for more details.</p> <p>If 0, the port shall not generate viral.</p> <p>Default value of this bit is 0.</p> <p>Regardless of the state of this bit, a switch shall always forward viral as described in Section 12.4.</p> <p>This bit is not applicable to root ports, and reads shall return the value of 0. Viral behavior of a Root Port may be controlled by a host specific configuration mechanism.</p>
15	RsvdP	Reserved

8.1.5.3 Alternate Bus Base (Offset 0Eh)

Alternate Bus Base Number and Alternate Bus Limit Number registers define a bus range that is decoded by the Port in addition to the standard Secondary Bus Number to Subordinate Bus Number range. An ID-routed TLP transaction received from the primary interface is forwarded to the secondary interface if the bus number is not less than the Alternate Bus Base and not greater than the Alternate Bus Limit. See [Figure 9-11](#).

Bit	Attributes	Description
7:0	RW	Alt Bus Base: The lowest bus number that is positively decoded by this Port as part of alternate decode path. Default value of this field is 0.

8.1.5.4 Alternate Bus Limit (Offset 0Fh)

See [Section 8.1.5.3, “Alternate Bus Base \(Offset 0Eh\).”](#)

Bit	Attributes	Description
7:0	RW	Alt Bus Limit: The highest bus number that is positively decoded by this Port as part of alternate decode path. Default value of this field is 0. Alternate bus decoder is disabled if Alt Memory and ID Space Enable=0.

8.1.5.5 Alternate Memory Base (Offset 10h)

Alternate Memory Base and Alternate Memory Limit registers define a memory mapped address range that is in addition to the standard Memory Base and Memory Limit registers. Alternate Memory Base and Alternate Memory Limit registers are functionally equivalent to PCIe-defined Memory Base and Memory Limit registers. These are used by the Port to determine when to forward memory transactions from one interface to the other. See [Figure 9-10](#).

Bit	Attributes	Description
3:0	RsvdP	Reserved
15:4	RW	Alt Mem Base: Corresponds to A[31:20] of the CXL.io Alternate memory base address. See definition of Memory Base register in PCIe Base Specification. Default value of this field is 000h.

8.1.5.6 Alternate Memory Limit (Offset 12h)

See [Section 8.1.5.5, “Alternate Memory Base \(Offset 10h\).”](#)

Bit	Attributes	Description
3:0	RsvdP	Reserved
15:4	RW	Alt Mem Limit: Corresponds to A[31:20] of the CXL.io Alternate memory limit address. See definition of Memory Limit register in PCIe Base Specification. Default value of this field is 000h.

8.1.5.7**Alternate Prefetchable Memory Base (Offset 14h)**

Alternate Prefetchable Memory Base, Alternate Prefetchable Memory Base High, Alternate Prefetchable Memory Limit, and Alternate Prefetchable Memory Limit High registers define a 64-bit memory mapped address range that is in addition to the one defined by the PCIe standard Prefetchable Memory Base, Prefetchable Base Upper 32 bits, Prefetchable Memory Limit, and Prefetchable Limit Upper 32 bits registers.

Alternate Prefetchable Memory registers are functionally equivalent to PCIe-defined Prefetchable Memory registers. These are used by the Port to determine when to forward Prefetchable memory transactions from one interface to the other.

Bit	Attributes	Description
3:0	RsvdP	Reserved
15:4	RW	Alt Prefetch Mem Base: Corresponds to A[31:20] of the CXL.io Alternate Prefetchable memory base address. See definition of Prefetchable Memory Base register in PCIe Base Specification. Default value of this field is 000h.

8.1.5.8**Alternate Prefetchable Memory Limit (Offset 16h)**

See Section 8.1.5.7, "Alternate Prefetchable Memory Base (Offset 14h)."

Bit	Attributes	Description
3:0	RsvdP	Reserved
15:4	RW	Alt Prefetch Mem Limit: Corresponds to A[31:20] of the CXL.io Alternate Prefetchable memory limit address. See definition of Prefetchable memory limit register in PCIe Base Specification. Default value of this field is 000h.

8.1.5.9**Alternate Memory Prefetchable Base High (Offset 18h)**

See Section 8.1.5.7, "Alternate Prefetchable Memory Base (Offset 14h)."

Bit	Attributes	Description
31:0	RW	Alt Prefetch Base High: Corresponds to A[63:32] of the CXL.io Alternate Prefetchable memory base address. See definition of Prefetchable Base Upper 32 Bits register in PCIe Base Specification. Default value of this register is 0000 0000h.

8.1.5.10**Alternate Prefetchable Memory Limit High (Offset 1Ch)**

See Section 8.1.5.7, "Alternate Prefetchable Memory Base (Offset 14h)."

Bit	Attributes	Description
31:0	RW	Alt Prefetch Limit High: Corresponds to A[63:32] of the CXL.io Alternate Prefetchable memory limit address. See definition of Prefetchable Limit Upper 32 Bits register in PCIe Base Specification. Default value of this register is 0000 0000h.

8.1.5.11**CXL RCRB Base (Offset 20h)**

This register is only relevant to CXL root ports and Downstream Switch Ports. Software programs this register to transition a Port to operate using RCD addressing. Software may take this step upon determining that the Port is connected to an eRCD.

System Firmware must ensure CXL RCRB Enable is 0, whenever the Port is operating in PCIe mode.

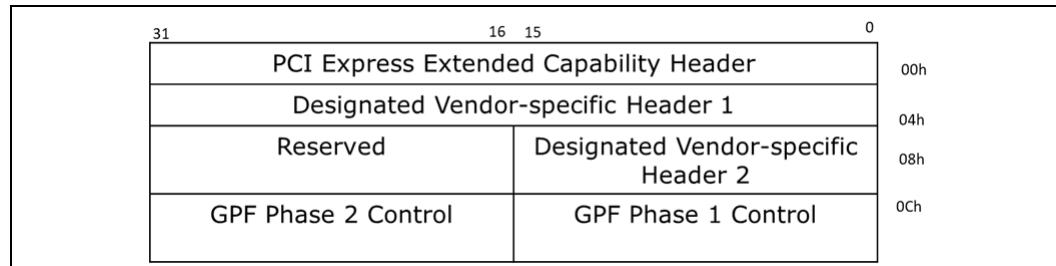
Bit	Attributes	Description
0	RW	CXL RCRB Enable: When set, the RCRB region is enabled and the registers belonging to this Port can be accessed via RCH Downstream Port RCRB. After this write is complete, the Port registers shall no longer appear in Configuration Space, but rather in MMIO space starting at RCRB Base. Once a Port is transitioned to use RCD addressing, the software is responsible for ensuring it remains in that mode until the next Conventional Reset and RCRB Base Address is not modified; otherwise, the hardware behavior is undefined. Default value of this bit is 0.
12:1	RsvdP	Reserved
31:13	RW	CXL RCRB Base Address Low: This points to the address bits[31:13] of an 8-KB memory region where the lower 4-KB hosts the RCH Downstream Port RCRB and the upper 4-KB hosts the RCD Upstream Port RCRB. Default value of this field is 0 0000h.

8.1.5.12 CXL RCRB Base High (Offset 24h)

Bit	Attributes	Description
31:0	RW	CXL RCRB Base Address High: This points to the address bits [63:32] of an 8-KB memory region where the lower 4-KB hosts the RCH Downstream Port RCRB and the upper 4-KB hosts the RCD Upstream Port RCRB. Default value of this register is 0000 0000h.

8.1.6 GPF DVSEC for CXL Port

Figure 8-4. GPF DVSEC for CXL Port



The PCIe Configuration Space of CXL Downstream Switch Ports and CXL root ports must implement this DVSEC capability as shown in Figure 8-4. See Table 8-2 for the complete listing.

To advertise this capability, the standard DVSEC register fields must be set to the values shown in Table 8-7. The DVSEC Length field must be set to 010h bytes to accommodate the registers included in the DVSEC. The DVSEC ID must be set to 0004h to advertise that this is an GPF DVSEC capability structure for CXL ports.

Table 8-7. GPF DVSEC for CXL Port - Header

Register	Bit Location	Field	Value
Designated Vendor-Specific Header 1 (Offset 04h)	15:0	DVSEC Vendor ID	1E98h
	19:16	DVSEC Revision	0h
	31:20	DVSEC Length	010h
Designated Vendor-Specific Header 2 (Offset 08h)	15:0	DVSEC ID	0004h

8.1.6.1 GPF Phase 1 Control (Offset 0Ch)

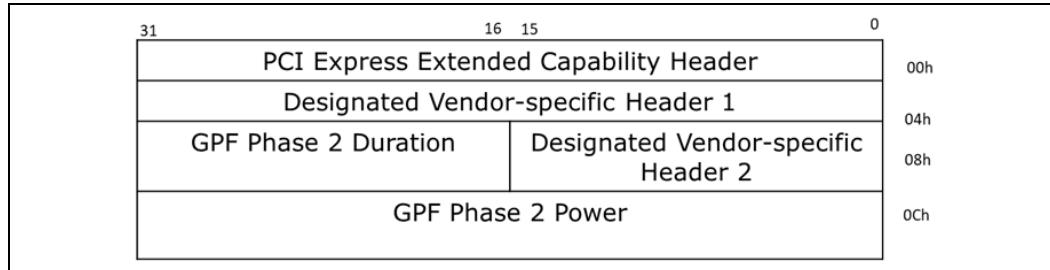
Bit	Attributes	Description
3:0	RW	Port GPF Phase 1 Timeout Base: This field determines the GPF Phase 1 timeout. The timeout duration is calculated by multiplying the Timeout Base with the Timeout Scale.
7:4	RsvdP	Reserved
11:8	RW	Port GPF Phase 1 Timeout Scale: This field specifies the time scale associated with GPF Phase 1 Timeout. <ul style="list-style-type: none"> • 0h = 1 us • 1h = 10 us • 2h = 100 us • 3h = 1 ms • 4h = 10 ms • 5h = 100 ms • 6h = 1 s • 7h = 10 s • All other encodings are reserved
15:12	RsvdP	Reserved

8.1.6.2 GPF Phase 2 Control (Offset 0Eh)

Bit	Attributes	Description
3:0	RW	Port GPF Phase 2 Timeout Base: This field determines the GPF Phase 2 timeout. The timeout duration is calculated by multiplying the Timeout Base with the Timeout Scale.
7:4	RsvdP	Reserved
11:8	RW	Port GPF Phase 2 Timeout Scale: This field specifies the time scale associated with GPF Phase 2 Timeout. <ul style="list-style-type: none"> • 0h = 1 us • 1h = 10 us • 2h = 100 us • 3h = 1 ms • 4h = 10 ms • 5h = 100 ms • 6h = 1 s • 7h = 10 s • All other encodings are reserved
15:12	RsvdP	Reserved

8.1.7 GPF DVSEC for CXL Device

Figure 8-5. GPF DVSEC for CXL Device



Device 0, Function 0 of CXL.mem-capable devices must implement this DVSEC capability (see [Figure 8-5](#)) if the device supports GPF (see [Table 8-2](#) for the complete listing). A device that does not support CXL.mem must not implement DVSEC Revision 0 capability. To advertise this capability, the standard DVSEC register fields must be set to the values shown in [Table 8-8](#). The DVSEC Length field must be set to 010h bytes to accommodate the registers included in the DVSEC. The DVSEC ID must be set to 0005h to advertise that this is an GPF DVSEC structure for CXL devices.

Table 8-8. GPF DVSEC for CXL Device - Header

Register	Bit Location	Field	Value
Designated Vendor-Specific Header 1 (Offset 04h)	15:0	DVSEC Vendor ID	1E98h
	19:16	DVSEC Revision	0h
	31:20	DVSEC Length	010h
Designated Vendor-Specific Header 2 (Offset 08h)	15:0	DVSEC ID	0005h

8.1.7.1 GPF Phase 2 Duration (Offset 0Ah)

Bit	Attributes	Description
3:0	RO	Device GPF Phase 2 Time Base: This field reports the maximum amount of time this device would take to complete GPF Phase 2. The time duration is calculated by multiplying the Time Base with the Time Scale.
7:4	RsvdP	Reserved
11:8	RO	Device GPF Phase 2 Time Scale: This field specifies the time scale associated with Device GPF Phase 2 Time. <ul style="list-style-type: none"> • 0h = 1 us • 1h = 10 us • 2h = 100 us • 3h = 1 ms • 4h = 10 ms • 5h = 100 ms • 6h = 1 s • 7h = 10 s • All other encodings are reserved
15:12	RsvdP	Reserved

8.1.7.2 GPF Phase 2 Power (Offset 0Ch)

Bit	Attributes	Description
31:0	RO	GPF Phase 2 Active Power: Active power consumed by the device during GPF Phase 2. Expressed in multiples of mW.

8.1.8 PCIe DVSEC for Flex Bus Port

See [Section 8.2.1.3](#) for the register layout.

In RCHs and RCDs that implement RCRB, this DVSEC is accessed via RCRB.

The DVSEC associated with all other CXL devices shall be accessible via Device 0, Function 0 of the device. Upstream Switch Ports, Downstream Switch Ports, and CXL root ports shall implement this DVSEC in the Configuration Space associated with the Port. See [Table 8-2](#) for the complete listing.

8.1.9 Register Locator DVSEC

The PCIe Configuration Space of a CXL root port, CXL Downstream Switch Port, CXL Upstream Switch Port, and non-RCDs must implement this DVSEC capability. If a CXL device implements Register Locator DVSEC, it must appear in Device 0, Function 0 of the device. This requirement does not apply to CXL Switches.

This DVSEC capability contains one or more Register Block entries. [Figure 8-6](#) illustrates a DVSEC Capability with 3 Register Block Entries. See [Table 8-2](#) for the complete listing.

Figure 8-6. Register Locator DVSEC with 3 Register Block Entries

31	16 15	0
	PCI Express Extended Capability Header	00h
	Designated Vendor-specific Header 1	04h
	Reserved	08h
	Designated Vendor-specific Header 2	
	Register Block 1 - Register Offset Low	0Ch
	Register Block 1 - Register Offset High	10h
	Register Block 2 - Register offset Low	14h
	Register Block 2 - Register offset High	18h
	Register Block 3 - Register Offset Low	1Ch
	Register Block 3 - Register offset High	20h

Each register block included in the Register Locator DVSEC has an Offset Low and an Offset High register to specify the location of the registers within the Memory Space. The Offset Low register includes an identifier which specifies the type of CXL registers. Each register block identifier shall only occur once in the Register Locator DVSEC structure, except for the Designated Vendor Specific register block identifier or the CPMU register block identifier where multiple instances are allowed. Each register block must be contained within the address range covered by the associated BAR.

To advertise this capability, the standard DVSEC register fields must be set to the values shown in [Table 8-9](#). The DVSEC Length field must be set to $(0Ch + n * 8)$ bytes to accommodate the registers included in the DVSEC, where n is the number of Register Blocks described by this Capability. The DVSEC ID must be set to 0008h to advertise that this is a CXL Register Locator DVSEC capability structure.

Table 8-9. Register Locator DVSEC - Header

Register	Bit Location	Field	Value
Designated Vendor-Specific Header 1 (Offset 04h)	15:0	DVSEC Vendor ID	1E98h
	19:16	DVSEC Revision	0h
	31:20	DVSEC Length	varies
Designated Vendor-Specific Header 2 (Offset 08h)	15:0	DVSEC ID	0008h

8.1.9.1 Register Offset Low (Offset: Varies)

This register reports the BAR Indicator Register (BIR), the Register Block Identifier, and the lower address bits of the BAR offset associated with the Register Block.

Bit	Attributes	Description
2:0	HwInit	<p>Register BIR: Indicates which one of a Function's BARs, located beginning at Offset 10h in Configuration Space, or entry in the Enhanced Allocation capability with a matching BAR Equivalent Indicator (BEI), is used to map the CXL registers into Memory Space.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> • 000b = Base Address Register 10h • 001b = Base Address Register 14h • 010b = Base Address Register 18h • 011b = Base Address Register 1Ch • 100b = Base Address Register 20h • 101b = Base Address Register 24h • All other encodings are reserved <p>The Register block must be contained within the specified BAR. The specified BAR must be associated with the Function that implements the Register Locator DVSEC. For a 64-bit BAR, the Register BIR indicates the lower DWORD.</p>
7:3	RsvdP	Reserved
15:8	HwInit	<p>Register Block Identifier: Identifies the type of CXL registers.</p> <p>Defined encodings are:</p> <ul style="list-style-type: none"> • 00h = Indicates the register block entry is empty and the Register BIR, Register Block Offset Low, and Register Block Offset High fields are invalid. • 01h = Component Registers. The format of the Component Register block is defined in Section 8.2.3. • 02h = BAR Virtualization ACL Registers. The format of the BAR Virtualization ACL Register Block is defined in Section 8.2.6. • 03h = CXL Device Registers. The format of the CXL Device Register block is defined in Section 8.2.8. • 04h = CPMU Registers. More than one instance per Register Locator DVSEC instance is permitted. The CPMU Register format is defined in Section 8.2.7. • FFh = Designated Vendor Specific Registers. The format of the designated vendor specific register block starts with the header defined in Table 8-10. • All other encodings are reserved.
31:16	HwInit	Register Block Offset Low: A[31:16] byte offset from the starting address of the Function's BAR associated with the Register BIR field to point to the base of the Register Block. Register Block Offset is 64-KB aligned. Hence A[15:0] is 0000h.

Table 8-10. Designated Vendor Specific Register Block Header

Offset	Bit Location	Attributes	Description
00h	15:0	RO	Vendor ID: The PCI-SIG assigned Vendor ID for the organization that defined the layout and controls the specification for this register block.
	31:16	RO	Vendor Register Block ID: Value defined by the Vendor ID in bits 15:0 that indicates the nature and format of the vendor specific registers.
	35:32	RO	Vendor Register Block Revision: Version number defined by the Vendor ID in bits 15:0 that indicates the version of the register block.
	63:36	RsvdP	Reserved
08h	31:0	RO	Vendor Register Block Length: The number of bytes in the register block, including the Designated Vendor Specific Register Block Header and the vendor specific registers.
	63:32	RsvdP	Reserved

8.1.9.2 Register Offset High (Offset: Varies)

This register reports the higher address bits of the BAR offset associated with the Register Block. Zeroed if the register block entry in the Register Locator DVSEC is empty.

Bit	Attributes	Description
31:0	HwInit	Register Block Offset High: A[63:32] byte offset from the starting address of the Function's BAR associated with the Register BIR field to point to the base of the Register Block.

8.1.10 MLD DVSEC

The MLD DVSEC (see [Figure 8-7](#)) applies only to FM-owned LDs and must not be implemented by any other functions. See [Table 8-2](#) for the complete listing.

To advertise this capability, the standard DVSEC register fields must be set to the values shown in [Table 8-11](#). The DVSEC Length field must be set to 010h bytes to accommodate the registers included in the DVSEC. The DVSEC ID must be set to 0009h to advertise that this is an MLD DVSEC capability structure.

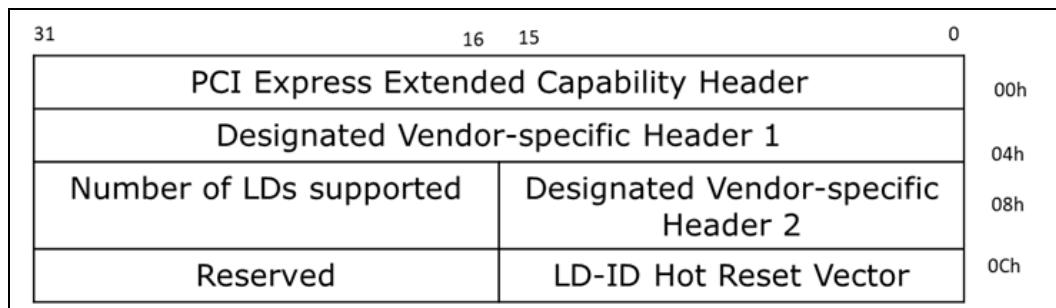
Figure 8-7. MLD DVSEC

Table 8-11. MLD DVSEC - Header

Register	Bit Location	Field	Value
Designated Vendor-Specific Header 1 (Offset 04h)	15:0	DVSEC Vendor ID	1E98h
	19:16	DVSEC Revision	0h
	31:20	DVSEC Length	010h
Designated Vendor-Specific Header 2 (Offset 08h)	15:0	DVSEC ID	0009h

8.1.10.1 Number of LD Supported (Offset 0Ah)

This register is used by an MLD to advertise the number of LDs supported.

Bit	Attributes	Description
15:0	HwInit	Number of LDs Supported: This field indicates the number of LDs (not counting FM-owned LDs) that are supported. An MLD must be associated with at least one LD. As such, 0000h is an illegal value for this field. Up to 16 LDs are supported; encodings greater than 16 are reserved.

8.1.10.2 LD-ID Hot Reset Vector (Offset 0Ch)

This register is used by the switch to trigger hot reset of the logical device or devices associated with LD-ID Hot Reset Vector bit positions that are set to a value of 1.

Bit	Attributes	Description
15:0	RW	LD-ID Hot Reset Vector: Each bit position in this vector represents an LD-ID. Up to 16 LD-IDs are supported. Setting any bit position to 1 triggers a hot reset of the associated logical device. Multiple bits can be set simultaneously to trigger hot reset of multiple logical devices. Read of this register returns a value of 0000h.

8.1.11 Table Access DOE

Coherent Device Attributes Table (CDAT) allows a device or a switch to expose its performance attributes such as latency and bandwidth characteristics and other attributes of the device or the switch. A CXL Upstream Switch Port or Device 0, Function 0 of a CXL device may implement Table Access DOE capability, which can be used to read out CDAT, one entry at a time. See [Table 8-3](#) for the complete listing.

A device may interrupt the host when CDAT content changes using the MSI associated with this DOE Capability instance. A device may share the instance of this DOE mailbox with other Data Objects.

This type of Data Object is identified as shown below. The Vendor ID must be set to the CXL Vendor ID to indicate that this Object Type is defined by the CXL specification. The Data Object Type must be set to 02h to advertise that this is a Table Access type of data object.

Table 8-12. Coherent Device Attributes - Data Object Header

Bit Location	Field	Value
15:0	Vendor ID	1E98h
23:16	Data Object Type	02h

8.1.11.1 Read Entry

Read the specified entry from the specified table within the device or the switch. For CXL, the table type is always CDAT. If the HDM_Count field in DVSEC CXL Capability is 01b, CDAT content is valid only when the Memory_Info_Valid flag in DVSEC CXL Range 1 Size Low (see [Section 8.1.3.8.2](#)) is 1. If the HDM_Count field in DVSEC CXL Capability is 10b, CDAT content is valid only when Memory_Info_Valid flag in DVSEC CXL Range 1 Size Low and DVSEC CXL Range 2 Size Low (see [Section 8.1.3.8.6](#)) are both 1.

Table 8-13. Read Entry Request

Data Object Byte Location	Length in Bytes	Description
00h	8	Standard DOE Request Header: See PCIe Base Specification.
08h	1	Table Access Request Code: 0 to indicate this is a request to read an entry. All other values are reserved.
09h	1	Table Type <ul style="list-style-type: none"> • 0 = CDAT • All other types are reserved
0Ah	2	EntryHandle: Handle value associated with the entry being requested. For Table Type = 0, EntryHandle = 0 specifies that the request is for the CDAT header and EntryHandle>0 indicates the request is for the CDAT Structure[EntryHandle - 1].

Table 8-14. Read Entry Response

Data Object Byte Location	Length in Bytes	Description
00h	8	Standard DOE Request Header: See PCIe Base Specification.
08h	1	Table Access Response Code: 0 to indicate this is a response to read entry request
09h	1	Table Type: <ul style="list-style-type: none"> • 0 = CDAT • All other types are reserved Shall match the input supplied during the matching Read Entry Request.
0Ah	2	EntryHandle: EntryHandle value associated with the next entry in the Table. EntryHandle=FFFFh represents the last entry in the table and thus the end of the table.
0Ch	Variable	The table entry that corresponds to the EntryHandle field in the Read Entry Request (see Table 8-13).

8.1.12 Memory Device Configuration Space Layout

This section defines the Configuration Space registers required for CXL memory devices to advertise support for the memory device capabilities (see [Section 8.2.8.5](#)) and memory device command sets (see [Section 8.2.9.9](#)).

8.1.12.1 PCI Header - Class Code Register (Offset 09h)

The PCI Header, Class Code register (Offset 09h) shall be implemented as follows, indicating the Function is a “CXL Memory Device following the CXL 2.0 or later specification”. Such a CXL device shall advertise a Register Locator DVSEC entry with Register Block Identifier=03h.

Bit	Attributes	Description
7:0	RO	Programming Interface (PI) : Shall be set to 10h.
15:8	RO	Sub Class Code (SCC) : Indicates the sub class code as CXL memory device. Shall be set to 02h.
23:16	RO	Base Class Code (BCC) : Indicates the base class code as a memory controller. Shall be set to 05h.

8.1.12.2 Memory Device PCIe Capabilities and Extended Capabilities

The optional PCI and PCIe capabilities described in this section are required for a CXL memory device that implements the Class Code specified in [Section 8.1.12.1](#). See PCIe Base Specification for definitions of the associated registers.

Table 8-15. Memory Device PCIe Capabilities and Extended Capabilities

PCIe Capabilities and Extended Capabilities	Exceptions	Notes
Device Serial Number Extended Capability		Uniquely identifies the CXL memory device.

8.1.13 Switch Mailbox CCI Configuration Space Layout

This section defines the Configuration Space registers that are required for CXL Switch Mailbox CCI (see [Section 8.2.8.6](#)) and FM API command sets (see [Section 8.2.9.9](#)).

8.1.13.1 PCI Header - Class Code Register (Offset 09h)

To advertise Switch Mailbox CCI support, the PCI Header, Class Code register (Offset 09h) shall be implemented as indicated in [Table 8-16](#), indicating the Function is a "CXL Fabric Management Host Interface controller". Such a CXL Function shall advertise a Register Locator DVSEC entry with Register Block Identifier=03h.

Table 8-16. PCI Header - Class Code Register (Offset 09h) for Switch Mailbox CCI

Bit	Attributes	Description
7:0	RO	Programming Interface (PI) : Shall be cleared to 00h.
15:8	RO	Sub Class Code (SCC) : Shall be set to 0Bh.
23:16	RO	Base Class Code (BCC) : Shall be set to 0Ch.

8.2

Memory Mapped Registers

CXL memory mapped registers are located in six general regions as specified in [Table 8-17](#). Notably, the RCH Downstream Port and RCD Upstream Port are not discoverable through PCIe Configuration Space. Instead, the RCH Downstream and RCD Upstream Port registers are implemented using PCIe Root Complex register blocks (RCRBs). Additionally, the RCH Downstream Ports and RCD Upstream Ports each implement an MEMBAR0 region (also known as Component registers) to host registers for configuring the CXL subsystem components associated with the respective Port. MEMBAR0 register ([Figure 8-9](#)) holds the address of Component registers.

The RCH Downstream Port and RCD Upstream Port memory mapped register regions appear in memory space as shown in [Figure 8-8](#). Note that the RCRBs do not overlap with the MEMBAR0 regions. Also, note that the RCD Upstream Port's MEMBAR0 region

must fall within the range specified by the RCH Downstream Port's memory base and limit register. As long as these requirements are satisfied, the details of how the RCRBs are mapped into memory space are implementation specific.

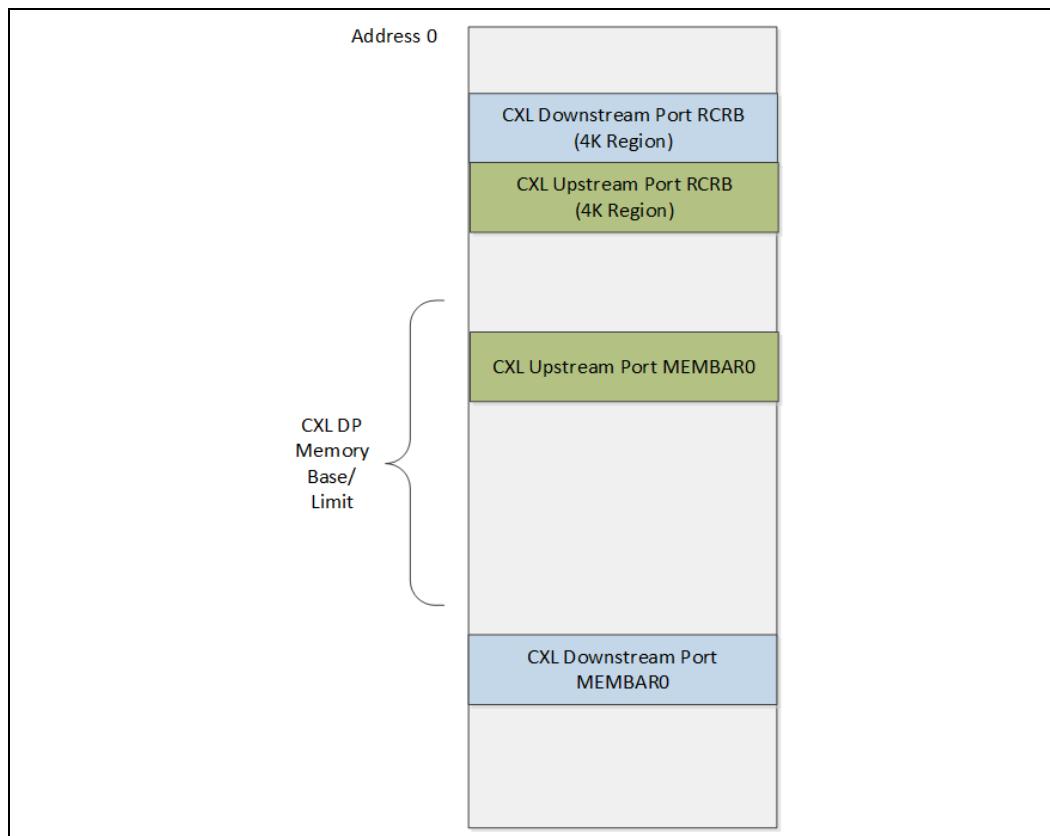
Software shall use CXL.io Memory Read and Write to access memory mapped register defined in this section. Unless specified otherwise, software shall restrict the accesses width based on the following:

- A 32-bit register shall be accessed as a 1-byte, 2-byte, or 4-byte quantity.
- A 64-bit register shall be accessed as a 1-byte, 2-byte, 4-byte, or 8-byte quantity.
- The address shall be a multiple of the access width (e.g., when accessing a register as a 4-byte quantity, the address shall be a multiple of 4).
- The accesses shall map to contiguous bytes.

If these rules are not followed, the behavior is undefined.

Table 8-17. CXL Memory Mapped Register Regions

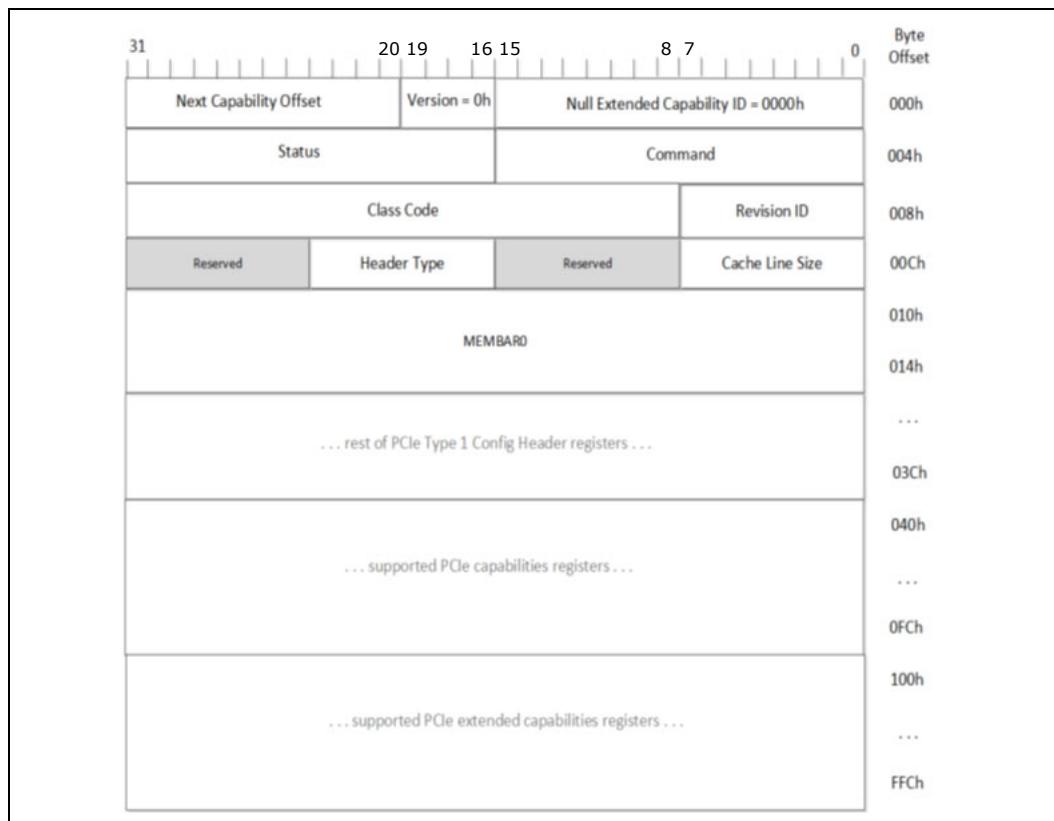
Memory Mapped Region	Description	Location
RCH Downstream Port RCRB	This is a 4-KB region with registers based upon PCIe defined registers for a root port with deltas listed in this chapter. Includes registers from PCIe Type 1 Config Header and PCIe capabilities and extended capabilities.	This is a contiguous 4-KB memory region relocatable via an implementation specific mechanism. This region is located outside the Downstream Port's MEMBAR0 region. Note: The combined Downstream and Upstream Port RCRBs are a contiguous 8-KB region.
RCD Upstream Port RCRB	This is a 4-KB region with registers based upon PCIe defined registers for an Upstream Port with deltas listed in this chapter. Includes 64B Config Header and PCIe capabilities and extended capabilities.	This is a contiguous 4-KB memory region relocatable via an implementation specific mechanism. This region is located outside the Upstream Port's MEMBAR0 region. This region may be located within the range specified by the Downstream Port's memory base/limit registers, but that is not a requirement. Note: The combined Downstream and Upstream Port RCRBs are a contiguous 8-KB region. The RCD Upstream Port captures the base of its RCRB from the Address field of the first MMIO Read (MRd) request received after the Conventional Reset.
RCH Downstream Port Component Registers	This memory region hosts registers that allow software to configure CXL Downstream Port subsystem components, such as the CXL protocol, link, and physical layers and the CXL ARB/MUX.	The location of this region is specified by a 64-bit MEMBAR0 register located at Offsets 10h and 14h of the Downstream Port's RCRB.
RCD Upstream Port Component Registers	This memory region hosts registers that allow software to configure CXL Upstream Port subsystem components, such as CXL protocol, link, and physical layers and the CXL ARB/MUX.	The location of this region is specified by a 64-bit MEMBAR0 register located at Offsets 10h and 14h of the Upstream Port's RCRB. This region is located within the range specified by the Downstream Port's memory base/limit registers.
Component Registers for All Other CXL Components	This memory region hosts registers that allow software to configure CXL Port subsystem components, such as CXL protocol, link, and physical layers and the CXL ARB/MUX. These are located in CXL root ports, CXL DSPs, CXL USPs, and CXL devices that do not have a CXL RCRB.	The CXL Port specific component registers are mapped in memory space allocated via a standard PCIe BAR associated with the appropriate PCIe non-virtual Function. Register Locator DVSEC structure (see Section 8.1.9) describes the BAR number and the offset within the BAR where these registers are mapped.
CXL CHBCR (CXL Host Bridge Component Registers)	This memory region hosts registers that allow software to configure CXL functionality that affects multiple root ports such as Memory Interleaving.	These registers are mapped in memory space, but the base address is discovered via ACPI CEDT (see Section 9.18.1).

Figure 8-8. RCD and RCH Memory Mapped Register Regions

8.2.1 RCD Upstream Port and RCH Downstream Port Registers

8.2.1.1 RCH Downstream Port RCRB

The RCH Downstream Port RCRB is a 4-KB memory region that contains registers based upon the PCIe-defined registers for a root port. [Figure 8-9](#) illustrates the layout of the CXL RCRB for a Downstream Port. With the exception of the first DWORD, the first 64 bytes of the RCH Downstream Port RCRB implement the registers from a PCIe Type 1 Configuration Header. The first DWORD of the RCRB contains a NULL Extended Capability ID with a Version of 0h and a Next Capability Offset pointer. A 64-bit MEMBAR0 is implemented at Offsets 10h and 14h; this points to a private memory region that hosts registers for configuring Downstream Port subsystem components as specified in [Table 8-17](#). The supported PCIe capabilities and extended capabilities are discovered by following the linked lists of pointers. Supported PCIe capabilities are mapped into the offset range from 040h to OFFh. Supported PCIe extended capabilities are mapped into the offset range from 100h to FFFh. The RCH Downstream Port supported PCIe capabilities and extended capabilities are listed in [Table 8-18](#); please refer to PCIe Base Specification for definitions of the associated registers.

Figure 8-9. RCH Downstream Port RCRB**Table 8-18.** RCH Downstream Port PCIe Capabilities and Extended Capabilities (Sheet 1 of 2)

PCIe Capabilities and Extended Capabilities	Exceptions ¹	Notes
PCIe Capability	Slot Capabilities, Slot Control, Slot Status, Slot Capabilities 2, Slot Control 2, and Slot Status 2 registers are not applicable.	N/A
PCI Power Management Capability	N/A. Software should ignore.	N/A
MSI Capability	N/A. Software should ignore.	N/A
Advanced Error Reporting Extended Capability	N/A. Software should ignore.	Required for CXL device despite being optional for PCIe. Downstream Port is required to forward ERR_ messages.
ACS Extended Capability	None	N/A
Multicast Extended Capability	N/A. Software should ignore.	N/A
Downstream Port Containment Extended Capability	Use with care. DPC trigger will bring down physical link, reset device state, disrupt CXL.cache and CXL.mem traffic.	N/A
Designated Vendor-Specific Extended Capability (DVSEC)	None	See Section 8.2.1.3 for Flex Bus Port DVSEC definition.
Secondary PCIe Extended Capability	None	None

Table 8-18. RCH Downstream Port PCIe Capabilities and Extended Capabilities (Sheet 2 of 2)

PCIe Capabilities and Extended Capabilities	Exceptions ¹	Notes
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

1. It is the responsibility of software to be aware of the registers within the capabilities that are not applicable in CXL mode in case designs choose to use a common code base for PCIe mode and CXL mode.

8.2.1.2 RCD Upstream Port RCRB

The RCD Upstream Port RCRB is a 4-KB memory region that contains registers based upon the PCIe Base Specification-defined registers. The Upstream Port captures the upper address bits [63:12] of the first memory read received after link initialization as the base address for the Upstream Port RCRB. [Figure 8-10](#) illustrates the layout of the RCRB for an RCD Upstream Port. With the exception of the first DWORD, the first 64 bytes of the RCD Upstream Port RCRB implement the registers from a PCIe Type 0 Configuration Header. The first DWORD of the RCRB contains a NULL Extended Capability ID with a Version of 0h and a Next Capability Offset pointer. A 64-bit BAR (labeled MEMBAR0) is implemented at Offsets 10h and 14h; this points to a memory region that hosts registers for configuring the Upstream Port subsystem CXL.mem as specified in [Table 8-17](#). The supported PCIe capabilities and extended capabilities are discovered by following the linked lists of pointers. Supported PCIe capabilities are mapped into the offset range from 040h to OFFh. Supported PCIe extended capabilities are mapped into the offset range from 100h to FFFh. The CXL Upstream Port-supported PCIe capabilities and extended capabilities are listed in [Table 8-19](#); see PCIe Base Specification for definitions of the associated registers.

The following standard registers that are part of the PCI Type 0 header definition are considered reserved and have no effect on the behavior of an RCD Upstream Port:

- Command register (Offset 04h)
- Status register (Offset 06h)

Per PCIe Base Specification, the following registers in the PCIe Capability are marked reserved for an RCiEP and shall not be implemented by the Device 0, Function 0 of the RCD:

- Link Registers - Link Capabilities, Link Control, Link Status, Link Capabilities 2, Link Control 2, and Link Status 2
- Slot Registers - Slot Capabilities, Slot Control, Slot Status, Slot Capabilities 2, Slot Control 2, and Slot Status 2
- Root Port Registers - Root Capabilities, Root Control, and Root Status

Software must reference the Link registers in the Upstream Port RCRB PCIe capability structure to discover the link capabilities and link status, and to configure the link properties. These registers shall follow the PCIe Base Specification definition of an Upstream Switch Port. Software must set the ASPM Control field in the Link Control register if it wishes to enable CXL.io L1.

All fields in the Upstream Port's Device Capabilities register, Device Control register, Device Status register, Device Capabilities 2 register, Device Control 2 register, and Device Status 2 register are reserved.

The Device/Port Type, Slots Implemented and Interrupt Message Number fields in the Upstream Port's Capability register are reserved.

Figure 8-10. RCD Upstream Port RCRB

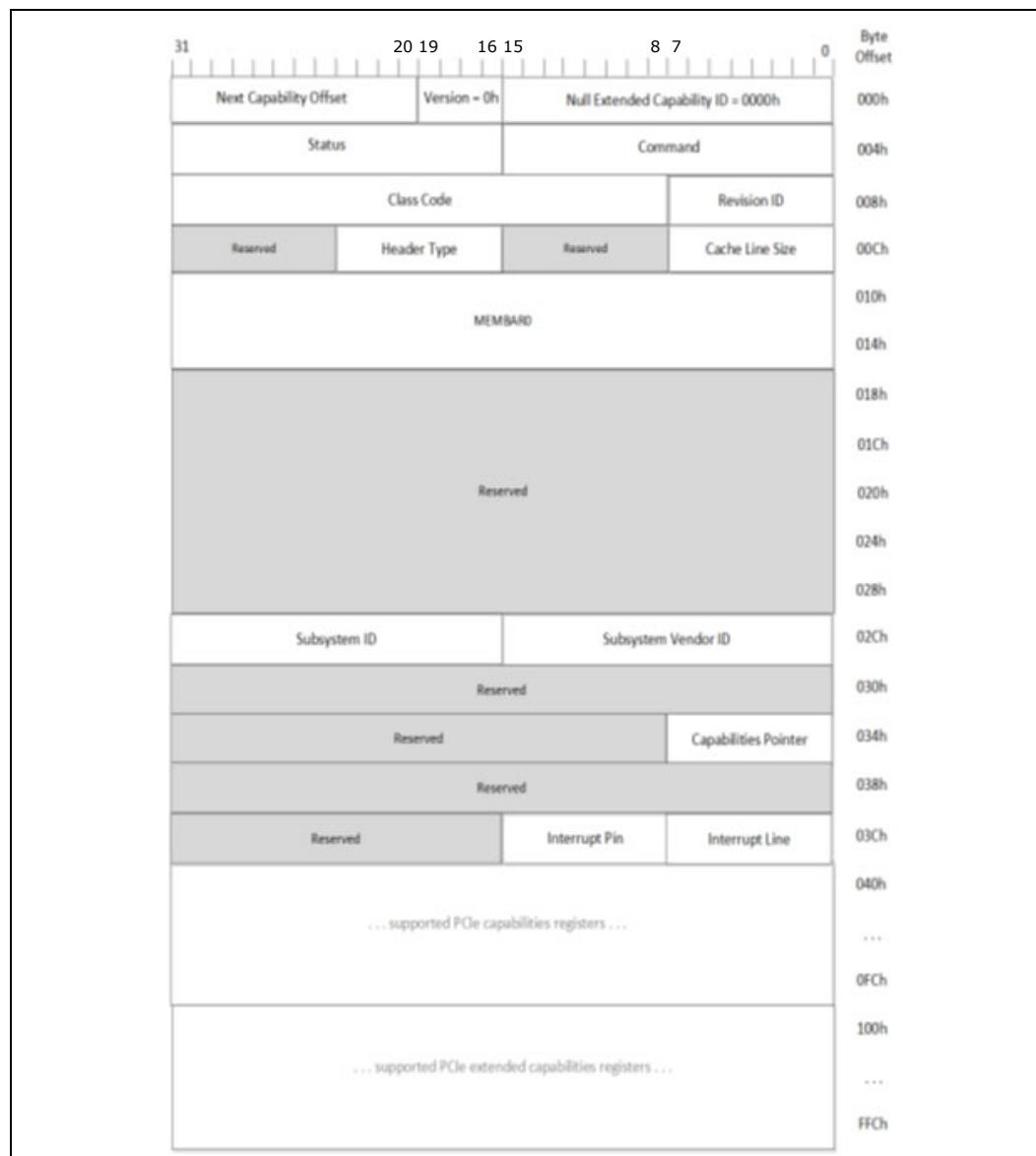


Table 8-19. RCD Upstream Port PCIe Capabilities and Extended Capabilities

PCIe Capabilities and Extended Capabilities	Exceptions ¹	Notes
PCIe Capability	See Section 8.2.1.2 .	None
Advanced Error Reporting Extended Capability	N/A. Software should ignore.	Required for CXL devices despite being optional for PCIe. Link/Protocol errors detected by Upstream Port are logged/reported via RCiEP.
Virtual Channel Extended Capability	None	VC0 and VC1
Designated Vendor-Specific Extended Capability (DVSEC)	None	See Section 8.2.1.3 for Flex Bus Port DVSEC definition.
Secondary PCIe 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

1. It is the responsibility of software to be aware of the registers within the capabilities that are not applicable in CXL mode in case designs choose to use a common code base for PCIe mode and CXL mode.

8.2.1.3 Flex Bus Port DVSEC

All CXL ports implement a Flex Bus Port DVSEC. This DVSEC is located in the RCRBs of the RCD Upstream Ports and RCH Downstream Ports. RCD and RCH ports may implement DVSEC Revision = 0, 1, or 2 of this DVSEC. See [Table 8-2](#) for the complete listing.

This DVSEC is also located in the Configuration Space of CXL root ports, Upstream Switch Ports, Downstream Switch Port, and CXL device's primary function (Device 0, Function 0) if the device does not implement CXL RCRB. A CXL component that is neither an RCD nor an RCH shall report DVSEC Revision greater than or equal to 1. Revision 2 introduces 3 new registers.

[Figure 8-11](#) shows the layout of the Flex Bus Port DVSEC and [Table 8-20](#) shows how the Header 1 and Header 2 registers shall be set. The following subsections give details of the registers defined in the Flex Bus Port DVSEC.

Figure 8-11. PCIe DVSEC for Flex Bus Port

31	16 15	0
PCI Express Extended Capability Header		00h
Designated Vendor-specific Header 1		04h
DVSEC Flex Bus Port Capability	Designated Vendor-specific Header 2	08h
DVSEC Flex Bus Port Status	DVSEC Flex Bus Port Control	0Ch
DVSEC Flex Bus Port Received Modified TS Data Phase1		10h
DVSEC Flex Bus Port Capability2		14h
DVSEC Flex Bus Port Control2		18h
DVSEC Flex Bus Port Status2		1Ch

Table 8-20. PCIe DVSEC Header Register Settings for Flex Bus Port

Register	Bit Location	Field	Value
Designated Vendor-Specific Header 1 (Offset 04h)	15:0	DVSEC Vendor ID	1E98h
	19:16	DVSEC Revision	2h
	31:20	DVSEC Length	020h
Designated Vendor-Specific Header 2 (Offset 08h)	15:0	DVSEC ID	0007h

8.2.1.3.1 DVSEC Flex Bus Port Capability (Offset 0Ah)

Note:

The Mem_Capable, IO_Capable, and Cache_Capable fields are also present in the Flex Bus DVSEC for the device. This allows for future scalability where multiple devices, each with potentially different capabilities, may be populated behind a single Port.

Bit	Attributes	Description
0	HwInit	Cache_Capable: If set, indicates CXL.cache protocol support when operating in Flex Bus.CXL mode. This should be cleared to 0 for all LDs of an MLD.
1	HwInit	IO_Capable: If set, indicates CXL.io protocol support when operating in Flex Bus.CXL mode. Must be 1.
2	HwInit	Mem_Capable: If set, indicates CXL.mem protocol support when operating in Flex Bus.CXL mode. This must be 1 for all LDs of an MLD.
4:3	RsvdP	Reserved
5	HwInit	CXL 68B Flit and VH Capable: Formerly known as CXL2p0_Capable. If set, indicates CXL VH functionality support with 68B flits is available when operating in Flex Bus.CXL mode. This must be 1 for all LDs of an MLD. ¹
6	HwInit	CXL_Multi-Logical_Device_Capable: If set, indicates Multi-Logical Device support available when operating in Flex Bus.CXL mode. This bit must be cleared to 0 on CXL host Downstream Ports. The value must be the same for all LDs of an MLD. ¹
12:7	RsvdP	Reserved
13	HwInit	CXL_Latency_Optimized_256B_Flit_Capable: If set, indicates support for latency-optimized 256B flits as described in Section 6.2.3.1.2 when operating in Flex Bus.CXL mode. The value must be the same for all LDs of an MLD. ²

Bit	Attributes	Description
14	HwInit	CXL PBR Flit Capable: If set, indicates support for PBR flits as described in Table 6-11 when operating in Flex Bus.CXL mode. ²
15	RsvdP	Reserved

1. Introduced as part of DVSEC Revision=1.
 2. Introduced as part of DVSEC Revision=2.

8.2.1.3.2 DVSEC Flex Bus Port Control (Offset 0Ch)

The Flex Bus physical layer uses the values that software sets in this register as a starting point for alternate protocol negotiation as long as the corresponding bit in the Flex Bus Port Capability register is set. The Flex Bus physical layer shall sample the values in this register only during exit from the Detect LTSSM state; the physical layer shall ignore any changes to this register in all other LTSSM states.

Bit	Attributes	Description
0	RW if Downstream Port; otherwise, HwInit	Cache_Enable: When set, enables CXL.cache protocol operation when in Flex Bus.CXL mode. Default value of this bit is 0.
1	RO	IO_Enable: When set, enables CXL.io protocol operation when in Flex Bus.CXL mode. Must always be set to 1.
2	RW if Downstream Port; otherwise HwInit	Mem_Enable: When set, enables CXL.mem protocol operation when in Flex Bus.CXL mode. Default value of this bit is 0.
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.
4	HwInit	Drift_Buffer_Enable: When set, enables drift buffer (instead of elastic buffer) if there is a common reference clock.
5	RW if Downstream Port; otherwise HwInit	CXL 68B Flit and VH Enable: Formerly known as CXL2p0_Enable. When set, enables CXL VH operation with 68B flits when in Flex Bus.CXL mode. This bit is reserved if CXL 68B Flit and VH Capable=0. ¹ Default value of this bit is 0.
6	RW if Downstream Port; otherwise HwInit	CXL_Multi-Logical_Device_Enable: When set, enable Multi-Logical Device operation when in Flex Bus.CXL mode. This bit shall always be cleared to 0 for CXL root ports and RCH Downstream Ports. ¹ Default value of this bit is 0.
7	RW if Downstream Port; otherwise HwInit	Disable_RCD_Training: Formerly known as Disable_CXL1p1_Training. When set, RCD mode is disabled. Typical usage model is that System Firmware will use this bit to disable Hot-Plug of an eRCD below a CXL root port or DSP. This bit is reserved on all RCD and RCH Upstream Ports. ¹ Default value of this bit is 0.
8	RW if Downstream Port; otherwise, RsvdP	Retimer1_Present: When set, indicates presence of Retimer1. This bit is defined only for a Downstream Port. This bit is reserved for an Upstream Port. Default value of this bit is 0. This bit is only used by RCH Downstream Ports. All other ports shall ignore this bit.
9	RW if Downstream Port; otherwise, RsvdP	Retimer2_Present: When set, indicates presence of Retimer2. This bit is defined only for a Downstream Port. This bit is reserved for an Upstream Port. Default value of this bit is 0. This bit is only used by RCH Downstream Ports. All other ports shall ignore this bit.
12:10	RsvdP	Reserved

Bit	Attributes	Description
13	RW if Downstream Port; otherwise HwInit	CXL Latency_Optimized_256B_Flit_Enable: When set, enables latency-optimized 256B flits when in Flex Bus.CXL mode. This bit is reserved on components that do not support 256B Flit mode. ² Default value of this bit is 0.
14	RW if Downstream Port; otherwise, HwInit	CXL PBR Flit Enable: When set, enables PBR flits when in Flex Bus.CXL mode. This bit is reserved on components that do not support PBR Flit mode. See Table 6-11 . ² Default value of this bit is 0.
15	RsvdP	Reserved

1. Introduced as part of DVSEC Revision=1.
2. Introduced as part of DVSEC Revision=2.

8.2.1.3.3 DVSEC Flex Bus Port Status (Offset 0Eh)

The Flex Bus physical layer reports the results of alternate protocol negotiation in this register.

Bit	Attributes	Description
0	RO	Cache_Enabled: When set, indicates that CXL.cache protocol operation has been enabled as a result of PCIe alternate protocol negotiation for Flex Bus.
1	RO	IO_Enabled: When set, indicates that CXL.io protocol operation has been enabled as a result of PCIe alternate protocol negotiation for Flex Bus.
2	RO	Mem_Enabled: When set, indicates that CXL.mem protocol operation has been enabled as a result of PCIe alternate protocol negotiation for Flex Bus.
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.
4	RO	Drift_Buffer_Enabled: When set, indicates that the physical layer has enabled its drift buffer instead of its elastic buffer.
5	RO	CXL 68B Flit and VH Enabled: Formerly known as CXL2p0_Enabled. When set, indicates that CXL VH operation with 68B Flit mode has been enabled as a result of PCIe alternate protocol negotiation for Flex Bus. ¹
6	RO	CXL_Multi-Logical_Device_Enabled: When set, indicates that CXL Multi-Logical Device operation has been negotiated. ¹
7	RW1CS	Even Half Failed: When set, indicates the Physical Layer detected a CRC error on the even flit half of a post-FEC corrected flit; however, even flit half was previously consumed because the even half passed CRC in the original flit. This bit is reserved in 68B Flit mode. This error is also logged as a Receiver Error in the AER Correctable Status register by the associated root port. ²
8	RW1CS	CXL_Correctable_Protocol_ID_Framing_Error: See Section 6.2.2 for more details. This bit is reserved in 256B Flit mode. It is recommended that this error also be logged as a Receiver Error in the AER Correctable Status register by the associated root port.
9	RW1CS	CXL_Uncorrectable_Protocol_ID_Framing_Error: See Section 6.2.2 for more details. This bit is reserved in 256B Flit mode. It is recommended that this error also be logged as a Receiver Error in the AER Correctable Status register by the associated root port.
10	RW1CS	CXL_Unexpected_Protocol_ID_Dropped: When set, indicates that the physical layer dropped a flit with an unexpected Protocol ID that is not the result of an Uncorrectable Protocol ID Framing Error. See Section 6.2.2 for more details. This bit is reserved in 256B Flit mode. It is recommended that this error also be logged as a Receiver Error in the AER Correctable Status register by the associated root port.

Bit	Attributes	Description
11	RW1CS	CXL_Retimers_Present_Mismatched: When set, indicates that the Downstream Port physical layer detected an inconsistency in the “Retimers Present” or “Two Retimers Present” bits in the received TS2 Ordered Sets during Polling.Config vs. Config.Complete LTSSM states. The physical layer will force disable of the sync header bypass optimization when this error condition has been detected. See Section 6.4.1.2.1 for more details. This bit is reserved on Upstream Ports.
12	RW1CS	FlexBusEnableBits_Phase2_Mismatch: When set, indicates that the Downstream Port physical layer detected that the Upstream Port did not exactly reflect the Flex Bus enable bits located in symbols 12-14 of the modified TS2 during Phase 2 of the negotiation. See Section 6.4.1.1 for more details. This bit is reserved on Upstream Ports.
13	RO	CXL_Latency_Optimized_256B_Flit_Enabled: When set, indicates that latency-optimized 256B flits have been enabled as a result of PCIe alternate protocol negotiation for Flex Bus. ²
14	RO	CXL_PBR_Flit_Enabled: When set, indicates that PBR flits have been enabled as a result of PCIe alternate protocol negotiation for Flex Bus. See Table 6-11 . ²
15	RO	CXL.io_Throttle_Required_at_64GT/s: When set, indicates that the partner Upstream Port does not support receiving consecutive CXL.io flits at 64 GT/s (see Section 6.4.1.3). This bit is only defined for Downstream Ports; this bit is reserved on Upstream Ports. ²

1. Introduced as part of DVSEC Revision=1.
2. Introduced as part of DVSEC Revision=2.

8.2.1.3.4 DVSEC Flex Bus Port Received Modified TS Data Phase1 (Offset 10h)

If CXL alternate protocol negotiation is enabled and the Modified TS Received bit is set in the 32.0 GT/s Status register (see PCIe Base Specification), then this register contains the values received in Symbols 12 through 14 of the Modified TS1 Ordered Set during Phase 1 of CXL alternate protocol negotiation.

Bit	Attributes	Description ¹
23:0	RO	Received_Flex_Bus_Data_Phase_1: This field contains the values received in Symbols 12 through 14 of the Modified TS1 Ordered Set during Phase 1 of CXL alternate protocol negotiation. ²
31:24	RsvdZ	Reserved

1. This register was introduced as part of DVSEC Revision=1.
2. This field was introduced as part of DVSEC Revision=1.

8.2.1.3.5 DVSEC Flex Bus Port Capability2 (Offset 14h)

Bit	Attributes	Description ¹
0	RO	NOP_Hint_Capable: If set, indicates support for sending and processing NOP hints when operating with latency-optimized 256B flits in Flex Bus.CXL mode. ²
31:1	RsvdP	Reserved

1. This register was introduced as part of DVSEC Revision=2.
2. This bit was introduced as part of DVSEC Revision = 2.

8.2.1.3.6 DVSEC Flex Bus Port Control2 (Offset 18h)

Bit	Attributes	Description ¹
0	RW	NOP_Hint_Enable: If set, enables sending and processing NOP hints when operating with latency-optimized 256B flits in Flex Bus.CXL mode. ² The default value of this field is 0.
31:1	RsvdP	Reserved

1. This register was introduced as part of DVSEC Revision=2.
2. This bit was introduced as part of DVSEC Revision = 2.

8.2.1.3.7 DVSEC Flex Bus Port Status2 (Offset 1Ch)

Bit	Attributes	Description ¹
1:0	RO	NOP_Hint_Info: The Physical Layer captures what the remote link partner advertises during Phase 1 of link training. ²
31:2	RsvdP	Reserved

1. This register was introduced as part of DVSEC Revision=2.
2. This field was introduced as part of DVSEC Revision = 2.

8.2.2 Accessing Component Registers

The RCD Upstream Port maps the Component registers in memory space that are allocated via the MEMBAR0 register of the RCD RCRB if the RCD implements RCRB. Similarly, the RCH Downstream Port maps the Component registers in memory space that are allocated via the MEMBAR0 register of the RCH RCRB. [Section 8.2.3](#) defines the architected registers. [Table 8-21](#) lists the relevant offset ranges from MEMBAR0 for CXL.io, CXL.cache, CXL.mem, and CXL ARB/MUX registers.

For an RCD Upstream Port that does not implement RCRB and for CXL components that are part of a CXL VH, the Component registers are mapped in memory space allocated via a standard PCIe BAR. The Register Locator DVSEC structure (see [Section 8.1.9](#)) describes the BAR number and the offset within the BAR where these registers are mapped.

A CXL Host Bridge contains Component registers that control the functionality of one or more CXL root ports. These are labeled CHBCR. These registers are also mapped in memory space, and the base address is discovered via ACPI CEDT (see [Section 9.18.1.2](#)).

For register layout, see [Figure 9-14](#) and [Figure 9-15](#).

8.2.3 Component Register Layout and Definition

The layout and discovery mechanism of the Component register is identical for all CXL Components and CXL Host Bridges (CHBCR). [Table 8-21](#) lists the relevant offset ranges from the Base of the Component register block for CXL.io, CXL.cache, CXL.mem, and CXL ARB/MUX registers.

Software shall use CXL.io Memory Reads and Writes to access CXL Component registers defined in [Section 8.2.4](#) and [Section 8.2.5](#). Software shall restrict the access width based on the following rules:

- A 32-bit register shall be accessed as a 4-byte quantity. Partial reads are not permitted.

- A 64-bit register shall be accessed as an 8-byte quantity. Partial reads are not permitted.
- Accesses shall map to contiguous bytes.

If these rules are not followed, the behavior is undefined. Note that these rules are more stringent than the general rules for the memory mapped registers that are specified in [Section 8.2](#).

This section and [Table 8-22](#) in this version of the specification do not define the behavior of CXL fabric switches (see [Section 2.7](#)) and G-FAM devices (see [Section 2.8](#)).

Table 8-21. CXL Subsystem Component Register Ranges

Range	Size	Description
0000 0000h - 0000 0FFFh	4 KB	Reserved for CXL.io registers. This specification does not define any CXL.io registers, hence the entire range is considered reserved.
0000 1000h - 0000 1FFFh	4 KB	CXL.cachemem Primary Range
0000 2000h - 0000 DFFFh	48 KB	Implementation specific. May host zero or more instances of CXL.cachemem Extended Ranges.
0000 E000h - 0000 E3FFh	1 KB	CXL ARB/MUX registers
0000 E400h - 0000 FFFFh	7 KB	Reserved. The range F000-FFFFh may host CXL.cachemem Extended Range.

8.2.4 CXL.cache and CXL.mem Registers

CXL.cache and CXL.mem registers are located in the CXL.cachemem Primary Range (Offset 1000h-1FFFh) or one of the CXL.cachemem Extended Ranges. Within each of the 4-KB region of memory space assigned to CXL.cache and CXL.mem, the location of architecturally specified registers is described using an array of pointers. The array, described in [Table 8-23](#), is located starting at Offset 00h of this 4-KB region. The first element of the array will declare the version of CXL.cache and CXL.mem protocols, as well as the size of the array. Each subsequent element will then host the pointers to capability-specific register blocks within the 4-KB region. [Table 8-24](#) and [Table 8-25](#) illustrate this concept with an example.

Structures with Capability ID of 1 through 0Ah are not permitted to be part of the CXL.cachemem Extended Ranges. Capability ID 0Ah structure identifies the CXL.cachemem Extended Ranges. Structures with Capability ID 0 or Capability ID greater than 0Ah are permitted to be part of the CXL.cachemem Primary Range or any of the CXL.cachemem Extended Ranges.

For each capability ID, CXL_Capability_Version field is incremented whenever the structure is extended to add more functionality. Backward compatibility shall be maintained during this process. For all values of n, CXL_Capability_Version=n+1 structure may extend CXL_Capability_Version=n by replacing fields that are marked as reserved in CXL_Capability_Version=n, but shall not redefine the meaning of existing fields. In addition, CXL_Capability_Version n+1 may append new registers to the CXL_Capability_Version n structure. Software that was written for a lower CXL_Capability_Version may continue to operate on structures with a higher CXL_Capability_Version, but will not be able to take advantage of new functionality.

CXL_Capability_ID field represents the functionality and CXL_Capability_Version represents the version of the structure. The following values of CXL_Capability_ID are defined by CXL specification.

Table 8-22. CXL_Capability_ID Assignment

Capability	ID	Highest Version	Mandatory ¹	Not Permitted ¹	Optional ¹
CXL NULL Capability – Software shall ignore this structure and skip to the next CXL Capability	0	Undefined		P	D1, D2, LD, FMLD, DP1, UP1, USP, DSP, R, RC
CXL Capability (Section 8.2.4.1)	1	1	D1, D2, LD, FMLD, UP1, DP1, R, USP, DSP, RC	P	
CXL RAS Capability (Section 8.2.4.17)	2	3	D1, D2, LD, FMLD, UP1, DP1, R, USP, DSP	P, RC	
CXL Security Capability (Section 8.2.4.18)	3	1	DP1	All others	
CXL Link Capability (Section 8.2.4.19)	4	4	D1, D2, LD, FMLD, UP1, DP1, R, USP, DSP	P, RC	
CXL HDM Decoder Capability (Section 8.2.4.20)	5	3	Type 3 D2, LD, RC except RCH, USP	All others	Type 2 D2, D1
CXL Extended Security Capability (Section 8.2.4.21)	6	2	RC	All others	
CXL IDE Capability (Section 8.2.4.22)	7	2		P, D1, LD, UP1, DP1	D2, FMLD, R, USP, DSP
CXL Snoop Filter Capability (Section 8.2.4.23)	8	1	R	P, D1, D2, LD, FMLD, UP1, USP, DSP, RC	DP1
CXL Timeout and Isolation Capability (Section 8.2.4.24)	9	1		P, D1, D2, LD, FMLD, UP1, USP, DSP, RC	R
CXL.cachemem Extended Register Capability (Section 8.2.4.25)	0Ah	1		P	D1, D2, LD, FMLD, UP1, R, USP, DSP, RC
CXL BI Route Table Capability (Section 8.2.4.26)	0Bh	1	USP that requires explicit BI commit	All others	All other USPs
CXL BI Decoder Capability (Section 8.2.4.27)	0Ch	1	DSP or Type 2 D2 that advertises 256B Flit mode	P, D1, FMLD, UP1, DP1, R, all other USPs, RC	R ² , all other DSPs, all other D2s, LD
CXL Cache ID Route Table Capability (Section 8.2.4.28)	0Dh	1		All others	RC, USP
CXL Cache ID Decoder Capability (Section 8.2.4.29)	0Eh	1		P, D1, D2, LD, FMLD, UP1, DP1, R, DSP, RC	R, DSP
CXL Extended HDM Decoder Capability (Section 8.2.4.30)	0Fh	3		All others	RC, USP
CXL Extended Metadata Capability (Section 8.2.4.31)	10h	1		All others	CXL.mem capable LD or D2 that supports 256B Flit Mode

1. P - PCIe device, D1 - RCD, D2 - CXL device that is not RCD, LD - Logical Device, FMLD - Fabric Manager owned LD FFFFh, UP1 - RCD Upstream Port RCRB, DP1 - RCH Downstream Port, R - CXL root port, RC - CXL Host Bridge registers in CHBCR, USP - CXL Upstream Switch Port, DSP - CXL Downstream Switch Port. A physical component may be capable of operating in multiple modes (e.g., a CXL device may operate either in D1 mode or D2 mode based on the link training). In such cases, these definitions refer to the current mode of operation.
2. Strongly recommended for a host that supports 256B Flit mode.

Table 8-23. CXL.cache and CXL.mem Architectural Register Discovery

Offset	Register Name
00h	CXL_Capability_Header
04h (Length = n*4, where n is the number of capability headers)	An array of individual capability headers. See Table 8-22 for the enumeration.

Table 8-24. CXL.cache and CXL.mem Architectural Register Header Example (Primary Range)

Byte Offset	Register Name
00h	CXL_Capability_Header
04h	CXL_RAS_Capability_Header
08h	CXL_Security_Capability_Header
0Ch	CXL_Link_Capability_Header
10h	CXL.cachemem Extended Register Capability Header

Table 8-25. CXL.cache and CXL.mem Architectural Register Header Example (Any Extended Range)

Byte Offset	Register Name
00h	CXL_Capability_Header
04h	CXL BI Decoder Capability Header
08h	CXL NULL Capability Header
0Ch	CXL Cache ID Decoder Capability Header

8.2.4.1 CXL Capability Header Register (Offset 00h)

Bit Location	Attributes	Description
15:0	RO	CXL_Capability_ID: This defines the nature and format of the CXL_Capability register. For the CXL_Capability_Header register, this field must be 0001h.
19:16	RO	CXL_Capability_Version: This defines the version number of the CXL_Capability structure present. For this and the prior version of the specification, this field must be 1h.
23:20	RO	CXL_Cache_Mem_Version: This defines the version of the CXL.cachemem Protocol supported. For this and the prior versions of the specification, this field must be 1h.
31:24	RO	Array_Size: This defines the number of elements present in the CXL_Capability array, not including the CXL_Capability_Header element. Each element is 1 DWORD in size and is located contiguous with previous elements.

8.2.4.2 CXL RAS Capability Header (Offset: Varies)

Bit Location	Attributes	Description
15:0	RO	CXL_Capability_ID: This defines the nature and format of the CXL_Capability register. For the CXL_RAS_Capability_Pointer register, this field shall be 0002h.
19:16	RO	CXL_Capability_Version: This defines the version number of the CXL_Capability structure present. Version 3h represents the structure as defined in this specification.
31:20	RO	CXL_RAS_Capability_Pointer: This defines the offset of the CXL RAS Capability relative to the beginning of the CXL_Capability_Header register. Details in Section 8.2.4.17 .

8.2.4.3 CXL Security Capability Header (Offset: Varies)

Bit Location	Attributes	Description
15:0	RO	CXL_Capability_ID: This defines the nature and format of the CXL_Capability register. For the CXL_Security_Capability_Pointer register, this field shall be 0003h.
19:16	RO	CXL_Capability_Version: This defines the version number of the CXL_Capability structure present. For this version of the specification, this field must be 1h.
31:20	RO	CXL_Security_Capability_Pointer: This defines the offset of the CXL Security Capability relative to the beginning of the CXL_Capability_Header register. Details in Section 8.2.4.18 .

8.2.4.4 CXL Link Capability Header (Offset: Varies)

Bit Location	Attributes	Description
15:0	RO	CXL_Capability_ID: This defines the nature and format of the CXL_Capability register. For the CXL_Link_Capability_Pointer register, this field shall be 0004h.
19:16	RO	CXL_Capability_Version: This defines the version number of the CXL_Capability structure present. Version 4h represents the structure as defined in this specification.
31:20	RO	CXL_Link_Capability_Pointer: This defines the offset of the CXL Link Capability relative to the beginning of the CXL_Capability_Header register. Details in Section 8.2.4.19 .

8.2.4.5 CXL HDM Decoder Capability Header (Offset: Varies)

Bit Location	Attributes	Description
15:0	RO	CXL_Capability_ID: This defines the nature and format of the CXL_Capability register. For the CXL_HDM_Decoder_Capability_Pointer register, this field shall be 0005h.
19:16	RO	CXL_Capability_Version: This defines the version number of the CXL_Capability structure present. For this version of the specification, this field must be 3h.
31:20	RO	CXL_HDM_Decoder_Capability_Pointer: This defines the offset of the CXLHDM Decoder Capability relative to the beginning of the CXL_Capability_Header register. Details in Section 8.2.4.20 .

8.2.4.6 CXL Extended Security Capability Header (Offset: Varies)

Bit Location	Attributes	Description
15:0	RO	CXL_Capability_ID: This defines the nature and format of the CXL_Capability register. For the CXL_Extended_Security_Capability_Pointer register, this field shall be 0006h.
19:16	RO	CXL_Capability_Version: This defines the version number of the CXL_Capability structure present. For this version of the specification, this field must be 2h.
31:20	RO	CXL_Extended_Security_Capability_Pointer: This defines the offset of the CXL Extended Security Capability relative to the beginning of the CXL_Capability_Header register. Details in Section 8.2.4.21 .

8.2.4.7 CXL IDE Capability Header (Offset: Varies)

This capability header is present in all ports that implement CXL IDE.

Bit Location	Attributes	Description
15:0	RO	CXL_Capability_ID: This defines the nature and format of the CXL_Capability register. For the CXL_IDE_Capability_Header register, this field shall be 0007h.
19:16	RO	CXL_Capability_Version: This defines the version number of the CXL_Capability structure present. For this version of the specification, this field must be 2h.
31:20	RO	CXL IDE Capability Pointer: This defines the offset of the CXL IDE Capability relative to the beginning of the CXL_Capability_Header register. Details in Section 8.2.4.22 .

8.2.4.8 CXL Snoop Filter Capability Header (Offset: Varies)

This capability header is required for Root Ports and optional for RCH Downstream Ports.

Bit Location	Attributes	Description
15:0	RO	CXL_Capability_ID: This defines the nature and format of the CXL_Capability register. For the CXL_Snoop_Filter_Capability_Header register, this field shall be 0008h.
19:16	RO	CXL_Capability_Version: This defines the version number of the CXL_Capability structure present. For this version of the specification, this field shall be 1h.
31:20	RO	CXL Snoop Filter Capability Pointer: This defines the offset of the CXL Snoop Filter Capability relative to the beginning of the CXL_Capability_Header register. Details in Section 8.2.4.23 .

8.2.4.9**CXL Timeout and Isolation Capability Header (Offset: Varies)**

Bit Location	Attributes	Description
15:0	RO	CXL_Capability_ID : This defines the nature and format of the CXL Capability register. For the CXL Timeout and Isolation Capability Header register, this field shall be 0009h.
19:16	RO	CXL_Capability_Version : This defines the version number of the CXL Capability structure present. For this version of the specification, this field must be 1h.
31:20	RO	CXL_Timeout_and_Isolation_Capability_Pointer : This defines the offset of the CXL Timeout and Isolation Capability structure relative to the beginning of the CXL_Capability_Header register. Details in Section 8.2.4.24 .

8.2.4.10**CXL.cachemem Extended Register Capability (Offset: Varies)**

Bit Location	Attributes	Description
15:0	RO	CXL_Capability_ID : This defines the nature and format of the CXL Capability register. For the CXL.cachemem Extended Register Capability Header register, this field shall be 000Ah.
19:16	RO	CXL_Capability_Version : This defines the version number of CXL Capability structure present. For this version of the specification, this field must be 1h.
31:20	RO	CXL.cachemem Extended Register Capability Pointer : This defines the offset of the CXL.cachemem Extended Register Capability structure relative to the beginning of the CXL_Capability_Header register. Details in Section 8.2.4.25 .

8.2.4.11**CXL BI Route Table Capability Header (Offset: Varies)**

Bit Location	Attributes	Description
15:0	RO	CXL_Capability_ID : This defines the nature and format of the CXL Capability register. For the CXL BI Route Table Capability Header register, this field shall be 000Bh.
19:16	RO	CXL_Capability_Version : This defines the version number of CXL Capability structure present. For this version of the specification, this field must be 1h.
31:20	RO	CXL BI Route Table Capability Pointer : This defines the offset of the CXL BI Route Table Capability structure relative to the beginning of the CXL_Capability_Header register. Details in Section 8.2.4.26 .

8.2.4.12**CXL BI Decoder Capability Header (Offset: Varies)**

Bit Location	Attributes	Description
15:0	RO	CXL_Capability_ID : This defines the nature and format of the CXL Capability register. For the CXL BI Decoder Capability Header register, this field shall be 000Ch.
19:16	RO	CXL_Capability_Version : This defines the version number of CXL Capability structure present. For this version of the specification, this field must be 1h.
31:20	RO	CXL BI Decoder Capability Pointer : This defines the offset of the CXL BI Decoder Capability structure relative to the beginning of the CXL_Capability_Header register. Details in Section 8.2.4.27 .

8.2.4.13 CXL Cache ID Route Table Capability Header (Offset: Varies)

Bit Location	Attributes	Description
15:0	RO	CXL_Capability_ID: This defines the nature and format of the CXL Capability register. For the CXL Cache ID Route Table Capability Header register, this field shall be 000Dh.
19:16	RO	CXL_Capability_Version: This defines the version number of CXL Capability structure present. For this version of the specification, this field must be 1h.
31:20	RO	CXL Cache ID Route Table Capability Pointer: This defines the offset of the CXL Cache ID Route Table Capability structure relative to the beginning of the CXL_Capability_Header register. Details in Section 8.2.4.28 .

8.2.4.14 CXL Cache ID Decoder Capability Header (Offset: Varies)

Bit Location	Attributes	Description
15:0	RO	CXL_Capability_ID: This defines the nature and format of the CXL Capability register. For the CXL Cache ID Decoder Capability Header register, this field shall be 000Eh.
19:16	RO	CXL_Capability_Version: This defines the version number of CXL Capability structure present. For this version of the specification, this field must be 1h.
31:20	RO	CXL Cache ID Local Decoder Capability Pointer: This defines the offset of the CXL Cache ID Decoder Capability structure relative to the beginning of the CXL_Capability_Header register. Details in Section 8.2.4.29 .

8.2.4.15 CXL Extended HDM Decoder Capability Header (Offset: Varies)

Bit Location	Attributes	Description
15:0	RO	CXL_Capability_ID: This defines the nature and format of the CXL Capability register. For the CXL Extended HDM Decoder Capability Header register, this field shall be 000Fh.
19:16	RO	CXL_Capability_Version: This defines the version number of CXL Capability structure present. For this version of the specification, this field must be 3h and shall track the version of the CXL HDM Decoder Capability structure.
31:20	RO	CXL Extended HDM Decoder Capability Pointer: This defines the offset of the CXL Extended HDM Decoder Capability structure relative to the beginning of the CXL_Capability_Header register. Details in Section 8.2.4.30 .

8.2.4.16 CXL Extended Metadata Capability Header (Offset: Varies)

Bit Location	Attributes	Description
15:0	RO	CXL_Capability_ID: This defines the nature and format of the CXL Capability register. For the CXL Extended Metadata Capability Header register, this field shall be 0010h.
19:16	RO	CXL_Capability_Version: This defines the version number of CXL Capability structure present. For this version of the specification, this field must be 1h.
31:20	RO	CXL Extended Metadata Capability Pointer: This defines the offset of the CXL Extended Metadata Capability structure relative to the beginning of the CXL_Capability_Header register. Details in Section 8.2.4.31 .

8.2.4.17 CXL RAS Capability Structure

Offset	Register Name
00h	Uncorrectable Error Status Register
04h	Uncorrectable Error Mask Register
08h	Uncorrectable Error Severity Register
0Ch	Correctable Error Status Register
10h	Correctable Error Mask Register
14h	Error Capability and Control Register
18h	Header Log Registers

8.2.4.17.1 Uncorrectable Error Status Register (Offset 00h)

Bit Location	Attributes	Description
0	RW1CS	<p>Cache_Data_Parity: Internal Uncorrectable Data error such as Data Parity error or Uncorrectable Data ECC error on CXL.cache that are not signaled by using poison on the CXL interface. The Header Log register contains the H2D Data Header if detected by either a host or a DSP. The Header Log register contains the D2H Data Header if detected by either a device or a USP.</p> <p>For CXL RAS Capability Version ≥ 3, DWORD 0 of the Header Log register is reserved and the Data Header shall start at Byte Offset 4 of the Header Log register.</p> <p>For CXL RAS Capability Version < 3, the position of the Data Header in the Header Log register is not defined by this specification.</p>
1	RW1CS	<p>Cache_Address_Parity: Internal Uncorrectable Address Parity error or other uncorrectable errors associated with the Address field on CXL.cache. The Header Log register contains the H2D Request Header if detected by either a host or a DSP. The Header Log register contains D2H Request Header if detected by either a device or a USP.</p> <p>For CXL RAS Capability Version ≥ 3, DWORD 0 of the Header Log register is reserved and the Request Header shall start at Byte Offset 4 of the Header Log register.</p> <p>For CXL RAS Capability Version < 3, the position of the Request Header in the Header Log register is not defined by this specification.</p>
2	RW1CS	<p>Cache_BE_Parity: Internal Uncorrectable Byte Enable Parity error or other Byte Enable uncorrectable errors on CXL.cache. The Header Log register contains the D2H Data Header if detected by either a device or a USP.</p> <p>For CXL RAS Capability Version ≥ 3, DWORD 0 of the Header Log register is reserved and the Data Header shall start at Byte Offset 4 of the Header Log register.</p> <p>For CXL RAS Capability Version < 3, the position of the Data Header in the Header Log register is not defined by this specification.</p>
3	RW1CS	<p>Cache_Data_ECC: Internal Uncorrectable Data ECC error on CXL.cache that are not signaled using poison on the CXL interface. The Header Log register contains the H2D Data Header if detected by either a host or a DSP. The Header Log register contains the D2H Data Header if detected by either a device or a USP.</p> <p>Note: For CXL RAS Capability Version < 3, it is permissible to log any Uncorrectable Data error on CXL.cache in Bit 0 and not in this bit.</p> <p>For CXL RAS Capability Version ≥ 3, this bit is deprecated and all Uncorrectable Data errors on CXL.cache that are not signaled by using CXL poison are logged in bit 0.</p> <p>For CXL RAS Capability Version < 3, the position of the Data Header in the Header Log register is not defined by this specification.</p>
4	RW1CS	<p>Mem_Data_Parity: Internal Uncorrectable Data error such as Data Parity error or Uncorrectable Data ECC error on CXL.mem that are not signaled by using poison on the CXL interface. The Header Log register contains the M2S RWD Data Header if detected by either a host or a DSP. The Header Log register contains the S2M DRS Data header if detected by either a device or a USP.</p> <p>For CXL RAS Capability Version ≥ 3, DWORD 0 of the Header Log register is reserved and the Data Header shall start at Byte Offset 4 of the Header Log register.</p> <p>For CXL RAS Capability Version < 3, the position of the Data Header in the Header Log register is not defined by this specification.</p>

EVALUATION COPY

Bit Location	Attributes	Description
5	RW1CS	<p>Mem_Address_Parity: Internal Uncorrectable Address Parity error or other uncorrectable errors associated with the Address field on CXL.mem.</p> <p>For CXL RAS Capability Version <3, the position of the M2S Req message or M2S Rwd Data Header or a BISnp Req message in the Header Log register is not defined by this specification.</p> <ul style="list-style-type: none"> • Logging by a Host or a DSP: If bit 0 of the Header Log register is 0, the remainder of the Header Log contains the M2S Req message. If Bit 0 of the Header Log register is 1, the remainder of the Header Log contains the M2S Rwd Data Header. • Logging by a Device or a USP: The remainder of the Header Log contains the BISnp message. <p>For CXL RAS Capability Version >=3:</p> <ul style="list-style-type: none"> • Logging by a Host or a DSP: If DWORD 0 bit 0 of the Header Log register is 0, the Header Log register contains the M2S Req message, starting at Byte offset 4. If DWORD 0 bit 0 of the Header Log register is 1, the remainder of the Header Log contains the M2S Rwd Data Header. The Data Header shall start at Byte Offset 4 of the Header Log register. Bits 31:1 of DWORD 0 of the Header Log register are reserved. • Logging by a Device or a USP: Header Log register contains the BISnp Req message, starting at Byte offset 4.
6	RW1CS	<p>Mem_BE_Parity: Internal Uncorrectable Byte Enable Parity error or other Byte Enable uncorrectable errors on CXL.mem. The Header Log register contains the M2S Rwd Data Header if detected by either a host or a DSP. The Header Log register contains the S2M DRS Data header if detected by either a device or a USP.</p> <p>For CXL RAS Capability Version >=3, DWORD 0 of the Header Log register is reserved and the Data Header shall start at Byte Offset 4 of the Header Log register.</p> <p>For CXL RAS Capability Version <3, the position of the M2S Rwd or S2M DRS Data Header in the Header Log register is not defined by this specification.</p>
7	RW1CS	<p>Mem_Data_ECC: Internal Uncorrectable Data ECC error on CXL.mem. The Header Log register contains the M2S Rwd Data Header if detected by either a host or a DSP. The Header Log register contains the S2M DRS Data header if detected by either a device or a USP.</p> <p>Note: For CXL RAS Capability Version <3, it is permissible to log any Uncorrectable Data error on CXL.mem in Bit 4 and not in this bit. For CXL RAS Capability Version >=3, this bit is deprecated and all Uncorrectable Data errors on CXL.mem that are not signaled by using CXL poison are logged in bit 4.</p> <p>For CXL RAS Capability Version <3, the position of the Data Header in the Header Log register is not defined by this specification.</p>
8	RW1CS/RsvdZ	<p>REINIT_Threshold: REINIT Threshold Hit (i.e., $(\text{NUM_PHY_REINIT}) \geq (\text{MAX_NUM_PHY_REINIT})$). See Section 4.2.8.5.1 for the definitions of NUM_PHY_REINIT and MAX_NUM_PHY_REINIT. Header Log is not applicable. No data is logged in the Header Log.</p> <p>This bit is reserved for 256B Flit mode.</p>

EVALUATION COPY

Bit Location	Attributes	Description
9	RW1CS	<p>Rsvd_Encoding_Violation: Received unrecognized encoding. Header Log contains the entire flit received when operating in 68B Flit mode. This bit should be set upon a Link-Layer-related encoding violation.</p> <p>For CXL RAS Capability Version <3 and operating in 68B Flit mode, the scope of encoding checking should include the scope where it falls into the “Reserved” or “RSVD” definitions in Table 4-5, Table 4-6, and Table 4-9.</p> <p>For CXL RAS Capability Version >=3 and operating in 68B Flit mode, the scope of checking shall include the encodings that are marked as “Reserved” or “RSVD” in Table 4-5, Table 4-6, Table 4-9, and Table 4-10.</p> <p>For CXL RAS Capability Version <3 and operating in 256B Flit mode, the content of the Header Log register is not defined by this specification.</p> <p>For CXL RAS Capability Version >=3 and operating in 256B Flit mode, the scope of checking shall include the encodings that are marked as “Reserved” or “RSVD” in Table 4-14, Table 4-15, Table 4-16, Table 4-19, and Table 4-20. In these cases, DWORD 0 of the Header Log register must be either 0 or 1. The component is permitted to log other unsupported encodings beyond what is required by this specification. In that scenario, DWORD 0 must be set to 2. DWORD 0 of the Header Log register indicates what is captured in the remaining DWORDs.</p> <ul style="list-style-type: none"> • DWORD 0 = 0: DWORD 1 of the Header Log register shall contain the first DWORD in the offending slot • DWORD 0 = 1: The lower 16 bits of DWORD 1 of the Header Log register shall contain the Credit field • DWORD 0 = 2: The layout of the remaining DWORDs in the Header Log register is vendor specific
10	RW1CS	<p>Poison_Received: Received Poison from the peer. No data is logged in the Header Log.</p>
11	RW1CS	<p>Receiver_Overflow</p> <ul style="list-style-type: none"> • 0 = A buffer did not overflow • 1 = A buffer overflowed and the receiver of messages is unable to sink a message <p>The first four bits of DWORD 0 of the Header Log register indicate which buffer overflowed, and should be interpreted as follows:</p> <ul style="list-style-type: none"> • 0000b --> D2H Req (Applicable to the Downstream Port) • 0001b --> D2H Rsp (Applicable to the Downstream Port) • 0010b --> D2H Data (Applicable to the Downstream Port) • 0011b --> M2S Req (Applicable to the Upstream Port) • 0100b --> S2M NDR (Applicable to the Downstream Port) • 0101b --> S2M DRS (Applicable to the Downstream Port) • 0110b --> H2D Req (Applicable to the Upstream Port) • 0111b --> H2D Rsp (Applicable to the Upstream Port) • 1000b --> H2D Data (Applicable to the Upstream Port) • 1001b --> M2S Rwd (Applicable to the Upstream Port) • 1010b --> BISnsp (Applicable to the Downstream Port) • 1011b --> BIRsp (Applicable to the Upstream Port) • All other encodings are reserved <p>Bits [31:4] of DWORD 0 are reserved.</p>
13:12	RsvdZ	<p>Reserved (Do not use)</p>
14	RW1CS	<p>Internal_Error: Component-specific error. The format of the Header Log is component-specific.</p>
15	RW1CS	<p>CXL_IDE_Tx_Error: See Section 8.2.4.22.4 for the next level details. No data is logged in the Header Log.¹</p>

Bit Location	Attributes	Description
16	RW1CS	<p>CXL_IDE_Rx_Error: See Section 8.2.4.22.4 for the next level details.¹</p> <p>For CXL RAS Capability Version <3, no data is logged in the Header Log.</p> <p>For CXL RAS Capability Version >=3, DWORD 0 defines the content of subsequent DWORDs.</p> <p>If DWORD 0 is 0 (applies to Rx Error Status=6h)</p> <ul style="list-style-type: none"> • DWORD 1: Current Idle Flit count • DWORD 2: Expected Idle Flit count after early MAC termination • All other DWORDs are reserved <p>If DWORD 0 is 1 (applies to Rx Error Status=7h)</p> <ul style="list-style-type: none"> • DWORD 1: Current Idle Flit count • DWORD 2: Expected Idle Flit count after Key Refresh • All other DWORDs are reserved <p>If DWORD 0 is 2 (applies to Rx Error Status=7h)</p> <ul style="list-style-type: none"> • DWORD 1: Current Idle Flit count • DWORD 2: Expected Idle Flit count after IDE termination handshake • All other DWORDs are reserved <p>All other DWORD 0 values are reserved.</p>
17	RW1CS	<p>Extended_Metadata_Error: An error associated with Extended Metadata field.²</p> <p>DWORD 0 of the Header Log register captures the type of error:</p> <ul style="list-style-type: none"> • 0 = A Root Port in an Extended Metadata-aware host received unexpected Extended Metadata on S2M DRS. • 1 = An Extended Metadata-aware device received unexpected Extended Metadata on M2S Rwd. • 2 = A Root Port in an Extended Metadata-aware host expected but did not receive Extended Metadata on S2M DRS. • 3 = An Extended Metadata-aware device expected but did not receive Extended Metadata on M2S Rwd. <p>DWORD 1 of the Header Log register contains the following:</p> <ul style="list-style-type: none"> • Bits[15:0]: Tag field associated with the value of the transaction with the EMDErr. • Bits[17:16]: MetaField value of the transaction with the EMDErr. • Bits[19:18]: MetaValue value of the transaction with the EMDErr. • Bit[20]: Indicates that an EMD value was captured with the EMDErr and is stored in DWORD 1. Must be 0 if the Enable Extended Metadata Error Logging bit is 0. <p>DWORD 2 of the Header Log register captures the Extended Metadata field value if bit[20] of DWORD 2[1] is 1. This bit must be 0 if the Enable Extended Metadata Error Logging bit is 0.</p>
31:18	RsvdZ	Reserved

1. Introduced as part of Version=2.
2. Introduced as part of Version=3.

8.2.4.17.2 Uncorrectable Error Mask Register (Offset 04h)

The Uncorrectable Error Mask register controls reporting of individual errors. When a bit is set, the corresponding error status bit in Uncorrectable Error Status register upon the error event is not set, the error is not recorded or reported in the Header Log and is not signaled.

Bit Location	Attributes	Description
0	RWS	<p>Cache_Data_Parity_Mask Default value for this bit is 1.</p>
1	RWS	<p>Cache_Address_Parity_Mask Default value for this bit is 1.</p>
2	RWS	<p>Cache_BE_Parity_Mask Default value for this bit is 1.</p>

Bit Location	Attributes	Description
3	RWS	Cache_Data_ECC_Mask Default value for this bit is 1.
4	RWS	Mem_Data_Parity_Mask Default value for this bit is 1.
5	RWS	Mem_Address_Parity_Mask Default value for this bit is 1.
6	RWS	Mem_BE_Parity_Mask Default value for this bit is 1.
7	RWS	Mem_Data_ECC_Mask Default value for this bit is 1.
8	RWS/RsvdP	REINIT_Threshold_Mask Default value for this bit is 1. This bit is reserved for 256B Flit mode.
9	RWS	Rsvd-Encoding_Violation_Mask Default value for this bit is 1.
10	RWS	Poison_Received_Mask Default value for this bit is 1.
11	RWS	Receiver_Overflow_Mask Default value for this bit is 1.
13:12	RsvdP	Reserved (Do not use)
14	RWS	Internal_Error_Mask Default value for this bit is 1.
15	RWS	CXL_IDE_Tx_Mask ¹ Default value for this bit is 1.
16	RWS	CXL_IDE_Rx_Mask ¹ Default value for this bit is 1.
17	RWS	Extended_Meta_Data_Error_Mask ² Default value for this bit is 1.
31:18	RsvdP	Reserved

1. Introduced as part of Version=2.
2. Introduced as part of Version=3.

8.2.4.17.3 Uncorrectable Error Severity Register (Offset 08h)

The Uncorrectable Error Severity register controls whether an individual error is considered Non-fatal or Fatal error. An error is considered 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 considered non-fatal uncorrectable and shall not trigger a Viral indication. This register does not control whether an error is signaled as ERR_FATAL or ERR_NONFATAL over CXL.io.

Bit Location	Attributes	Description
0	RWS	Cache_Data_Parity_Severity Default value for this bit is 1.
1	RWS	Cache_Address_Parity_Severity Default value for this bit is 1.
2	RWS	Cache_BE_Parity_Severity Default value for this bit is 1.

EVALUATION COPY

Control and Status Registers

Bit Location	Attributes	Description
3	RWS	Cache_Data_ECC_Severity Default value for this bit is 1.
4	RWS	Mem_Data_Parity_Severity Default value for this bit is 1.
5	RWS	Mem_Address_Parity_Severity Default value for this bit is 1.
6	RWS	Mem_BE_Parity_Severity Default value for this bit is 1.
7	RWS	Mem_Data_ECC_Severity Default value for this bit is 1.
8	RWS/RsvdP	REINIT_Threshold_Severity Default value for this bit is 1. This bit is reserved for 256B Flit mode.
9	RWS	Rsvd-Encoding_Violation_Severity Default value for this bit is 1.
10	RWS	Poison_Received_Severity Default value for this bit is 1.
11	RWS	Receiver_Overflow_Severity Default value for this bit is 1.
13:12	RsvdP	Reserved (Do not use)
14	RWS	Internal_Error_Severity Default value for this bit is 1.
15	RWS	CXL_IDE_Tx_Severity ¹ Default value for this bit is 1.
16	RWS	CXL_IDE_Rx_Severity ¹ Default value for this bit is 1.
17	RWS	Extended_MetaData_Error_Severity ² Default value for this bit is 1.
31:18	RsvdP	Reserved

1. Introduced as part of Version=2.
2. Introduced as part of Version=3.

8.2.4.17.4 Correctable Error Status Register (Offset 0Ch)

Bit Location	Attributes	Description
0	RW1CS	Cache_Data_ECC: Internal correctable error such as correctable Data ECC error on CXL.cache.
1	RW1CS	Mem_Data_ECC: Internal correctable error such as correctable Data ECC error on CXL.mem.
2	RW1CS/RsvdZ	CRC_Threshold: CRC Threshold Hit. The CRC threshold is component specific. Applicable only to 68B Flit mode. Reserved for 256B Flit mode.
3	RW1CS/RsvdZ	Retry_Threshold: Retry Threshold Hit. ($\text{NUM_RETRY} \geq \text{MAX_NUM_RETRY}$). See Section 4.2.8.5.1 for the definitions of NUM_RETRY and MAX_NUM_RETRY. Applicable only to 68B Flit mode. Reserved for 256B Flit mode.
4	RW1CS	Cache_Poison_Received: Received Poison from the peer on CXL.cache.
5	RW1CS	Mem_Poison_Received: Received Poison from the peer on CXL.mem.
6	RW1CS	Physical_Layer_Error: Received error indication from Physical Layer. The error indication may or may not be associated with a CXL.cachemem flit.
31:7	RsvdZ	Reserved

8.2.4.17.5 Correctable Error Mask Register (Offset 10h)

The Correctable Error Mask register controls reporting of individual errors. When a bit is set in this register, the corresponding error status bit is not set upon the error event, and the error is not signaled.

Bit Location	Attributes	Description
0	RWS	Cache_Data_ECC_Mask Default value for this bit is 1.
1	RWS	Mem_Data_ECC_Mask Default value for this bit is 1.
2	RWS	CRC_Threshold_Mask Default value for this bit is 1.
3	RWS	Retry_Threshold_Mask Default value for this bit is 1.
4	RWS	Cache_Poison_Received_Mask Default value for this bit is 1.
5	RWS	Mem_Poison_Received_Mask Default value for this bit is 1.
6	RWS	Physical_Layer_Error_Mask Default value for this bit is 1.
31:7	RsvdP	Reserved

8.2.4.17.6 Error Capabilities and Control Register (Offset 14h)

Bit Location	Attributes	Description
5:0	ROS	First_Error_Pointer: This identifies the bit position of the first error reported in the Uncorrectable Error Status register.
8:6	RsvdP	Reserved
9	RO	Multiple_Header_Recording_Capability: If this bit is set, it indicates if recording of more than one error header is supported.
12:10	RsvdP	Reserved
13	RWS	Poison_Enabled: If this bit is 0, the CXL port shall treat poison received on CXL.cache or CXL.mem as an uncorrectable error and log the error in the Uncorrectable Error Status register. If this bit is 1, the CXL ports shall treat poison received on CXL.cache or CXL.mem as a correctable error and log the error in the Correctable Error Status register. This bit defaults to 1. This bit is hardwired to 1 in CXL Upstream Switch Port, CXL Downstream Switch Port, and CXL devices that are not eRCDs.
31:14	RsvdZ	Reserved

8.2.4.17.7 Header Log Registers (Offset 18h)

Header Log registers are accessed as a series of 32-bit wide individual registers even though it is represented as a single 512-bit long entity for convenience. In accordance with [Section 8.2.2](#), each individual register shall be accessed as an aligned 4-Byte quantity.

Bit Location	Attributes	Description
511:0	ROS	Header Log: The information logged here depends on the type of Uncorrectable Error Status bit recorded as described in Section 8.2.4.17.1 . If multiple errors are logged in the Uncorrectable Error Status register, the First_Error_Pointer field in the Error Capabilities and Control register identifies the error that this log corresponds to.

8.2.4.18 CXL Security Capability Structure

This capability structure applies only for RCH Downstream Ports.

Offset	Register Name
00h	CXL Security Policy Register

8.2.4.18.1 CXL Security Policy Register (Offset 00h)

Table 8-26. Device Trust Level

Bit Location	Attributes	Description
1:0	RW	<p>Device Trust Level</p> <ul style="list-style-type: none"> • 00b = Trusted CXL device. At this setting, a CXL device will be able to get access on CXL.cache for both host-attached and device-attached memory ranges. The Host can still protect security sensitive memory regions. • 01b = Trusted for device-attached Memory Range Only. At this setting, a CXL device will be able to get access on CXL.cache for device-attached memory ranges only. Requests on CXL.cache for host-attached memory ranges will be aborted by the Host. • 10b = Untrusted CXL device. At this setting, all requests on CXL.cache will be aborted by the Host. <p>Note: These settings only apply to requests on CXL.cache. The device can still source requests on CXL.io regardless of these settings. Protection on CXL.io will be implemented using IOMMU-based page tables.</p> <p>Default value of this field is 10b.</p>
31:2	RsvdP	Reserved

8.2.4.19 CXL Link Capability Structure

Offset	Register Name
00h	CXL Link Layer Capability Register
08h	CXL Link Control and Status Register
10h	CXL Link Rx Credit Control Register
18h	CXL Link Rx Credit Return Status Register
20h	CXL Link Tx Credit Status Register
28h	CXL Link Ack Timer Control Register
30h	CXL Link Defeature Register
38h	CXL Link Rx Credit Control2 Register
40h	CXL Link Rx Credit Return Status2 Register
48h	CXL Link Tx Credit Status2 Register

8.2.4.19.1 CXL Link Layer Capability Register (Offset 00h)

Bit Location	Attributes	Description
3:0	RWS	CXL Link Version Supported: The value in this field does not affect the link behavior. This field has been deprecated and software must not rely on its value.
7:4	RO	CXL Link Version Received: Version of CXL Specification received from INIT.Param flit. This field has been deprecated and software must not rely on its value.
15:8	RWS/RsvdP	LLR Wrap Value Supported: LLR Wrap value supported by this entity. Used for debug. The default value of this field will be implementation dependent. This field is reserved for 256B Flit mode.
23:16	RO/ RsvdP	LLR Wrap Value Received: LLR Wrap value received from INIT.Param flit. Used for debug. This field is reserved for 256B Flit mode.

Bit Location	Attributes	Description
28:24	RO/RsvdP	NUM_Retry_Received: Num_Retry value reflected in the last RETRY.Req message received. Used for debug. This field is reserved for 256B Flit mode.
33:29	RO/RsvdP	NUM_Phys_Reinit_Received: Num_Phys_Reinit value reflected in the last RETRY.Req message received. Used for debug. This field is reserved for 256B Flit mode.
41:34	RO/RsvdP	Wr_Ptr_Received: Wr_Ptr value reflected in the last RETRY.Ack message received. This field is reserved for 256B Flit mode.
49:42	RO/RsvdP	Echo_Eseq_Received: Echo_Eseq value reflected in the last RETRY.Ack message received. This field is reserved for 256B Flit mode.
57:50	RO/RsvdP	Num_Free_Buf_Received: Num_Free_Buf value reflected in the last RETRY.Ack message received. This field is reserved for 256B Flit mode.
58	RO/RsvdP	No_LL_Reset_Support: If set, indicates that the LL_Reset configuration bit is not supported. ¹
63:59	RsvdP	Reserved

1. Introduced as part of Version=2.

8.2.4.19.2 CXL Link Layer Control and Status Register (Offset 08h)

Bit Location	Attributes	Description
0	RW	LL_Reset: Re-initialize without resetting values in sticky registers. When this bit is set, the link layer reset is initiated. When link layer reset completes, hardware will clear the bit to 0. Entity triggering LL_Reset should ensure that link is quiesced. Support for this bit is optional. If LL_Reset is not supported, the NO_LL_Reset_Support bit in the CXL Link Layer Capability register shall be set (see Section 8.2.4.19.1). The use of this bit is expected to be for debug. Any production need for Link Layer re-initialization is to be satisfied using CXL Hot Reset.
1	RWS	LL_Init_Stall: If set, link layer stalls the transmission of the LLCTRL-INIT.Param flit until this bit is cleared. The default value of this bit is 0.
2	RWS	LL_Crd_Stall: If set, link layer stalls credit initialization until this bit is cleared. The reset default value of this bit is 0.
4:3	RO	INIT_State: This field reflects the current initialization status of the Link Layer, including any stall conditions controlled by bits 2:1: <ul style="list-style-type: none"> 00b = NOT_RDY_FOR_INIT (stalled or unstalled): LLCTRL-INIT.Param flit not sent 01b = PARAM_EX: LLCTRL-INIT.Param sent and waiting to receive it 10b = CRD_RETURN_STALL: Parameter exchanged successfully, and Credit return is stalled 11b = INIT_DONE: Link Initialization Done: LLCTRL-INIT.Param flit sent and received, and initial credit refund not stalled
12:5	RO/RsvdP	LL_Retry_Buffer_Consumed: Snapshot of link layer retry buffer consumed. This field is reserved for 256B Flit mode.
63:13	RsvdP	Reserved

8.2.4.19.3 CXL Link Layer Rx Credit Control Register (Offset 10h)

The default settings are component specific. The contents of this register represent the credits advertised by the component.

Software may program this register and issue a hot reset to operate the component with credit settings that are lower than the default. The values in these registers take effect on the next hot reset. If software configures any of these fields to a value that is higher than the default, the results will be undefined.

Bit Location	Attributes	Description
9:0	RWS	Cache Req Credits: Credits to advertise for CXL.cache Request channel at init. The default value represents the maximum number of CXL.cache Request channel credits that the component supports.
19:10	RWS	Cache Rsp Credits: Credits to advertise for CXL.cache Response channel at init. The default value represents the maximum number of CXL.cache Response channel credits that the component supports.
29:20	RWS	Cache Data Credits: Credits to advertise for CXL.cache Data channel at init. The default value represents the maximum number of CXL.cache Data channel credits that the component supports.
39:30	RWS	Mem Req_Rsp Credits: For an Upstream Port, this field represents the credits to advertise for CXL.mem Request channel at init. For a Downstream Port, this field represents the credits to advertise for CXL.mem NDR channel at init. The default value represents the maximum number of credits that the port supports.
49:40	RWS	Mem Data Credits: Credits to advertise for CXL.mem Data channel at init. For an Upstream Port, this field represents the number of advertised RwD channel credits at init. For a Downstream Port, this field represents the number of advertised DRS channel credits at init. The default value represents the maximum number of channel credits that the port supports.
59:50	RWS/RsvdP	BI Credits: For an Upstream Port, this field represents the number of advertised BIRsp channel credits at init. For a Downstream Port, this field represents the number of advertised BISnp channel credits at init. The default value represents the maximum number of the appropriate Back-Invalidate channel credits of which the port is capable. ¹ This field is reserved for 68B Flit mode and for components that do not support BI.
63:60	RsvdP	Reserved

1. Introduced as part of Version=3.

8.2.4.19.4 CXL Link Layer Rx Credit Return Status Register (Offset 18h)

Bit Location	Attributes	Description
9:0	RO	Cache Req Credits: Running snapshot of accumulated CXL.cache Request credits to be returned
19:10	RO	Cache Rsp Credits: Running snapshot of accumulated CXL.cache Response credits to be returned
29:20	RO	Cache Data Credits: Running snapshot of accumulated CXL.cache Data credits to be returned
39:30	RO	Mem Req_Rsp Credits: For an Upstream Port, this field represents the running snapshot of the accumulated CXL.mem Request channel credits to be returned. For a Downstream Port, this field represents the running snapshot of the accumulated CXL.mem NDR channel credits to be returned.
49:40	RO	Mem Data Credits: Running snapshot of accumulated CXL.mem Data credits to be returned. For an Upstream Port, this field represents the running snapshot of the accumulated RWD channel credits to be returned. For a Downstream Port, this field represents the running snapshot of the accumulated DRS channel credits to be returned.
59:50	RO/RsvdP	BI Credits: For an Upstream Port, this field represents the running snapshot of the accumulated BIRsp channel credits to be returned. For a Downstream Port, this field represents the running snapshot of accumulated BISnp channel credits to be returned. ¹ This field is reserved for 68B Flit mode and for components that do not support BI.
63:60	RsvdP	Reserved

1. Introduced as part of Version=3.

8.2.4.19.5 CXL Link Layer Tx Credit Status Register (Offset 20h)

Bit Location	Attributes	Description
9:0	RO	Cache Req Credits: Running snapshot of CXL.cache Request credits for Tx
19:10	RO	Cache Rsp Credits: Running snapshot of CXL.cache Response credits for Tx
29:20	RO	Cache Data Credits: Running snapshot of CXL.cache Data credits for Tx
39:30	RO	Mem Req_Rsp Credits: In the case of an Upstream Port, this field represents the running snapshot of the CXL.mem NDR channel credits for Tx. In the case of a Downstream Port, this field represents the running snapshot of the CXL.mem Request channel credits for Tx.
49:40	RO	Mem Data Credits: Running snapshot of CXL.mem Data credits for Tx. In the case of an Upstream Port, this field represents the number of DRS channel credits for Tx. In the case of a Downstream Port, this field represents the number of RWD channel credits for Tx.
59:50	RO/RsvdP	BI Credits: In the case of an Upstream Port, this field represents the running snapshot of the accumulated BIRsp channel credits for Tx. In the case of a Downstream Port, this field represents the running snapshot of accumulated BISnp channel credits for Tx. ¹ This field is reserved for 68B Flit mode and for components that do not support BI.
63:60	RsvdP	Reserved

1. Introduced as part of Version=3.

8.2.4.19.6 CXL Link Layer Ack Timer Control Register (Offset 28h)

The default settings are component specific.

Software may program this register and issue a hot reset to operate the component with settings that are different from the default. The values in these registers take effect on the next hot reset.

Bit Location	Attributes	Description
7:0	RWS	Ack Force Threshold: This specifies how many Flit Acknowledgments the Link Layer should accumulate before injecting an LLCRD. The recommended default value is 10h (16 decimal). If configured to a value greater than (LLR Wrap Value Received - 6), the behavior will be undefined. If configured to a value below 10h, the behavior will be undefined. See Section 4.2.8.2 for additional details.
17:8	RWS	Ack or CRD Flush Retimer: This specifies how many link layer clock cycles the entity should wait in case of idle, before flushing accumulated Acknowledgments or CRDs using an LLCRD. This applies for any case where accumulated Acknowledgments is greater than 1 or accumulated CRD for any channel is greater than 0. The recommended default value is 20h. If configured to a value below 20h, the behavior will be undefined. See Section 4.2.8.2 for additional details.
63:18	RsvdP	Reserved

8.2.4.19.7 CXL Link Layer Defeature Register (Offset 30h)

Bit Location	Attributes	Description
0	RWS/RsvdP	MDH Disable: Write 1 to disable MDH. Software needs to ensure it programs this value consistently on the Upstream Port and Downstream Port. After programming, a hot reset is required for the disable to take effect. The default value of this bit is 0.
63:1	RsvdP	Reserved

8.2.4.19.8 CXL Link Layer Rx Credit Control2 Register (Offset 38h)

Bit Location	Attributes	Description ¹
9:0	RWS/RsvdP	<p>Direct P2P Mem Req Rsp Credits² For an Upstream Port, this field represents the credits to advertise for the Direct P2P CXL.mem NDR channel at init. This field must be RWS if the Direct P2P Mem Capable bit in the DVSEC CXL Capability3 register (see Section 8.1.3.9) is set; otherwise, it is permitted to be hardwired to 0. For a Downstream Port, this field represents the credits to advertise for Direct P2P CXL.mem Request channel at init. The default value represents the maximum number of credits that the port supports. This field must be RWS for a PBR switch that supports Direct P2P CXL.mem; otherwise, it is permitted to be hardwired to 0.</p>
19:10	RWS/RsvdP	<p>Direct P2P Mem Data Credits¹ For an Upstream Port, this field represents the number of advertised Direct P2P CXL.mem DRS channel credits at init. This field must be RWS if the Direct P2P Mem Capable bit in the DVSEC CXL Capability3 register (see Section 8.1.3.9) is set; otherwise, it is permitted to be hardwired to 0. For a Downstream Port, this field represents the number of advertised Direct P2P CXL.mem Rwd channel credits at init. The default value represents the maximum number of channel credits of which the port is capable. This field must be RWS for a PBR switch that supports Direct P2P CXL.mem; otherwise, it is permitted to be hardwired to 0.</p>
63:20	RsvdP	Reserved

1. This register was introduced as part of Version=4.
2. This field was introduced as part of Version=4.

8.2.4.19.9 CXL Link Layer Rx Credit Return Status2 Register (Offset 40h)

Bit Location	Attributes	Description ¹
9:0	RO/RsvdP	<p>Direct P2P Mem Req Rsp Return Credits² For an Upstream Port, this field represents the running snapshot of the accumulated Direct P2P CXL.mem NDR channel credits to be returned. This field must be RO if the Direct P2P Mem Capable bit in the DVSEC CXL Capability3 register (see Section 8.1.3.9) is set; otherwise, it is RsvdP. For a Downstream Port, this field represents the running snapshot of the accumulated Direct P2P CXL.mem Request channel credits to be returned. This field must be RO for a PBR switch that supports Direct P2P CXL.mem; otherwise, it is RsvdP.</p>
19:10	RO/RsvdP	<p>Direct P2P Mem Data Return Credits¹ For an Upstream Port, this field represents the running snapshot of the accumulated Direct P2P CXL.mem DRS channel credits to be returned. This field must be RO if the Direct P2P Mem Capable bit in the DVSEC CXL Capability3 register (see Section 8.1.3.9) is set; otherwise, it is RsvdP. For a Downstream Port, this field represents the running snapshot of the accumulated Direct P2P CXL.mem Rwd channel credits to be returned. This field must be RO for a PBR switch that supports Direct P2P CXL.mem; otherwise, it is RsvdP.</p>
63:20	RsvdP	Reserved

1. This register was introduced as part of Version=4.
2. This field was introduced as part of Version=4.

8.2.4.19.10 CXL Link Layer Rx Credit Status2 Register (Offset 48h)

Bit Location	Attributes	Description ¹
9:0	RO/RsvdP	<p>Direct P2P Mem Req Rsp Current Credits² For an Upstream Port, this field represents the running snapshot of the Direct P2P CXL.mem Request channel credits for Tx. This field must be RO if the Direct P2P Mem Capable bit in the DVSEC CXL Capability3 register (see Section 8.1.3.9) is set; otherwise, it is RsvdP.</p> <p>For a Downstream Port, this field represents the running snapshot of the Direct P2P CXL.mem NDR channel credits for Tx. This field must be RO for a PBR switch that supports Direct P2P CXL.mem; otherwise, it is RsvdP.</p>
19:10	RO/RsvdP	<p>Direct P2P Mem Data Current Credits¹ For an Upstream Port, this field represents the number of Direct P2P CXL.mem RdW channel credits for Tx. This field must be RO if the Direct P2P Mem Capable bit in the DVSEC CXL Capability3 register (see Section 8.1.3.9) is set; otherwise, it is RsvdP.</p> <p>For a Downstream Port, this field represents the number of Direct P2P CXL.mem DRD channel credits for Tx. This field must be RO for a PBR switch that supports Direct P2P CXL.mem; otherwise, it is RsvdP.</p>
63:20	RsvdP	Reserved

1. This register was introduced as part of Version=4.
2. This field was introduced as part of Version=4.

8.2.4.20 CXL HDM Decoder Capability Structure

CXL HDM Decoder Capability structure facilitates routing of CXL.mem as well as UIO transactions that target HDM and optionally enables interleaving of HDM across CXL.mem-capable devices.

A CXL Host Bridge is identified as an ACPI device with a Hardware ID (HID) of "ACPI0016" and is associated with one or more CXL root ports. Any CXL Host Bridge that is associated with more than one CXL root port must contain one instance of this capability structure in the CHBCR. This capability structure resolves the target CXL root ports for a given memory address.

A CXL switch component may contain one Upstream Switch Port and one or more Downstream Switch Ports. A CXL Upstream Switch Port that is capable of routing CXL.mem traffic to more than one Downstream Switch Ports shall contain one instance of this capability structure. The capability structure, located in CXL Upstream Switch Port, resolves the target CXL Downstream Switch Ports for a given memory address.

A CXL Type 3 device that is not an eRCD shall contain one instance of this capability structure. A CXL Type 2 device that supports BI or supports UIO access to its HDM shall contain one instance of this capability structure. The capability structure, located in a device, translates the Host Physical Address (HPA) into a Device Physical Address (DPA) after taking any interleaving into account.

Offset	Register Name
00h	CXL HDM Decoder Capability Register
04h	CXL HDM Decoder Global Control Register
08h	Reserved
0Ch	Reserved

Offset	Register Name
Decoder 0:	
10h	CXL HDM Decoder 0 Base Low Register
14h	CXL HDM Decoder 0 Base High Register
18h	CXL HDM Decoder 0 Size Low Register
1Ch	CXL HDM Decoder 0 Size High Register
20h	CXL HDM Decoder 0 Control Register
24h	CXL HDM Decoder 0 Target List Low Register (not applicable to devices) CXL HDM Decoder 0 DPA Skip Low Register (devices only)
28h	CXL HDM Decoder 0 Target List High Register (not applicable to devices) CXL HDM Decoder 0 DPA Skip High Register (devices only)
2Ch	Reserved
Decoder 1:	
30h – 4Fh	CXL HDM Decoder 1 registers
	...
Decoder n:	
20h *n+ 10h: 20h*n + 2Fh	CXL HDM Decoder n registers (see Section 8.2.4.20.3 through Section 8.2.4.20.11). $0 \leq n < \text{Raw Decoder Count}$. The Raw Decoder count is derived from the Decoder Count field in the CXL HDM Decoder Capability register (see Section 8.2.4.20.1).

8.2.4.20.1 CXL HDM Decoder Capability Register (Offset 00h)

Bit Location	Attributes	Description
3:0	RO	<p>Decoder Count: Reports the number of memory address decoders implemented by the component. CXL devices shall not advertise more than 10 decoders. CXL switches and Host Bridges may advertise up to 32 decoders.</p> <ul style="list-style-type: none"> • 0h = 1 Decoder • 1h = 2 Decoders • 2h = 4 Decoders • 3h = 6 Decoders • 4h = 8 Decoders • 5h = 10 Decoders • 6h = 12 Decoders² • 7h = 14 Decoders² • 8h = 16 Decoders² • 9h = 20 Decoders² • Ah = 24 Decoders² • Bh = 28 Decoders² • Ch = 32 Decoders² • All other encodings are reserved
7:4	RO	<p>Target Count: The number of target ports each decoder supports (applicable only to Upstream Switch Port and CXL Host Bridge). Maximum of 8.</p> <ul style="list-style-type: none"> • 1h = 1 target port • 2h = 2 target ports • 4h = 4 target ports • 8h = 8 target ports • All other encodings are reserved
8	RO	<p>A11to8 Interleave Capable: If set, the component supports interleaving based on Address bits [11:8].</p> <p>CXL Host Bridges and Upstream Switch Ports shall always set this bit indicating support for interleaving based on Address bits [11:8].</p>
9	RO	<p>A14to12 Interleave Capable: If set, the component supports interleaving based on Address bits [14:12].</p> <p>CXL Host Bridges and switches shall always set this bit indicating support for interleaving based on Address bits [14:12].</p>
10	RO	<p>Poison On Decode Error Capability: If set, the component is capable of returning poison on read access to addresses that are not positively decoded by any HDM Decoders in this component. If cleared, the component is not capable of returning poison under such scenarios.</p>
11	RO	<p>3, 6, 12 Way Interleave Capable: If set, the CXL.mem devices supports 3-way, 6-way and 12-way interleaving, respectively. Not applicable to Upstream Switch Ports and CXL Host Bridges. Upstream Switch Ports and CXL Host Bridges shall hardwire this bit to 0.¹</p>
12	RO	<p>16 Way Interleave Capable: If set, the CXL.mem device supports 16-way interleaving.</p> <p>Not applicable to Upstream Switch Ports and CXL Host Bridges. Upstream Switch Ports and CXL Host Bridges shall hardwire this bit to 0.¹</p>
13	HwInit	<p>UIO Capable²</p> <ul style="list-style-type: none"> • For CXL.mem devices: If set, the device supports UIO accesses to its HDM • For USPs: If set, the switch is capable of routing UIO accesses that target HDM across its ports • For CXL Host Bridges: If set, all the root ports within this Host Bridge are capable of routing UIO requests that target HDM across root ports within this Host Bridge
15:14	RsvdP	Reserved

Bit Location	Attributes	Description
19:16	HwInit/ RsvdP	<p>UIO Capable Decoder Count: Reports the total number of memory address decoders that are implemented by components that support UIO. Software is permitted to set the UIO bit in non-consecutive HDM decoders as long as the number of UIO-enabled decoders does not exceed this count. If the software attempts to set the UIO bit (see Section 8.2.4.20.2) in an HDM decoder beyond this limit, the component shall fail the HDM decoder commit operation.</p> <p>See the Decoder Count field in this register for enumeration.</p> <p>This field is reserved for a component if the UIO Capable bit in this register is 0.</p> <p>This field is reserved for CXL.mem devices. A UIO-capable CXL.mem device is not permitted to limit the number of UIO-capable HDM decoders and must operate correctly even when the UIO bit is set in all of its HDM decoders.²</p>
20	HwInit	<p>MemData-NXM Capable: If set, the component supports MemData-NXM opcode (see Table 3-53). If cleared, the component does not support MemData-NXM opcode. All 256B Flit mode-capable components shall set this bit to 1.</p> <p>See Table 8-27 for the description of how this bit affects the handling of CXL.mem read requests in case of errors.²</p>
22:21	HwInit/ RsvdP	<p>Supported Coherency Models: Indicates the coherency models that are supported by a CXL.mem device. This field is reserved for all other components.²</p> <ul style="list-style-type: none"> • 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 1 or cleared to 0 by software before committing the HDM decoder.
31:23	RsvdP	Reserved

1. Introduced as part of Version=2.
2. Introduced as part of Version=3.

Table 8-27. CXL.mem Read Response - Error Cases

Error Case	Poison On Decode Error Enable	MemData-NXM Capable ¹	Component Behavior
HPA does not match any HDM decoder	0	1	Return MemData-NXM with no poison.
	1	1	Return MemData-NXM with poison.
	0	0	Return all 1s data response with no poison. Component to choose whether to send DRS only or NDR+DRS.
	1	0	Return poison response. Component to choose whether to send DRS only or NDR+DRS.
HPA matches an HDM decoder, but the address is not assigned to the requester (e.g., DPA is not assigned to the host by DCD)	0	X	Return all 1s data response with no poison. Target Range Type field in HDM Decoder n Control register chooses whether to send DRS only or NDR+DRS.
	1	X	Return poison response. Target Range Type field in HDM Decoder n Control register chooses whether to send DRS only or NDR+DRS.

1. X indicates don't care.

8.2.4.20.2 CXL HDM Decoder Global Control Register (Offset 04h)

Bit Location	Attributes	Description
0	RW/RO	<p>Poison On Decode Error Enable: This bit is RO and is hardwired to 0 if Poison On Decode Error Capability=0.</p> <p>See Table 8-27 for the description of how this bit affects the handling of CXL.mem read requests in case of errors.</p> <p>Note: Writes to addresses that are not positively decoded shall be dropped and a No Data Response (see Section 3.3.9) shall be sent regardless of the state of this bit.</p> <p>Default value of this bit is 0.</p>
1	RW	<p>HDM Decoder Enable: This bit is only applicable to CXL.mem devices and shall return 0 on CXL Host Bridges and Upstream Switch Ports. When this bit is set, device shall use HDM decoders to decode CXL.mem transactions and not use HDM Base registers in PCIe DVSEC for CXL devices (see Section 8.1.3.8.3, Section 8.1.3.8.4, Section 8.1.3.8.7, and Section 8.1.3.8.8). CXL Host Bridges and Upstream Switch Ports always use HDM Decoders to decode CXL.mem transactions.</p> <p>Default value of this bit is 0.</p>
31:2	RsvdP	Reserved

8.2.4.20.3 CXL HDM Decoder n Base Low Register (Offset 20h*n+10h)

Bit Location	Attributes	Description
27:0	RsvdP	Reserved
31:28	RWL	<p>Memory Base Low: Corresponds to bits 31:28 of the base of the address range managed by Decoder n. The locking behavior is described in Section 8.2.4.20.13.</p> <p>Default value of this field is 0h.</p>

8.2.4.20.4 CXL HDM Decoder n Base High Register (Offset 20h*n+14h)

Bit Location	Attributes	Description
31:0	RWL	<p>Memory Base High: Corresponds to bits 63:32 of the base of the address range managed by Decoder n. The locking behavior is described in Section 8.2.4.20.13.</p> <p>Default value of this register is 0000 0000h.</p>

8.2.4.20.5 CXL HDM Decoder n Size Low Register (Offset 20h*n+18h)

Bit Location	Attributes	Description
27:0	RsvdP	Reserved
31:28	RWL	<p>Memory Size Low: Corresponds to bits 31:28 of the size of the address range managed by Decoder n. The locking behavior is described in Section 8.2.4.20.13.</p> <p>Default value of this field is 0h.</p>

8.2.4.20.6 CXL HDM Decoder n Size High Register (Offset 20h*n+1Ch)

Bit Location	Attributes	Description
31:0	RWL	<p>Memory Size High: Corresponds to bits 63:32 of the size of address range managed by Decoder n. The locking behavior is described in Section 8.2.4.20.13.</p> <p>Default value of this register is 0000 0000h.</p>

8.2.4.20.7 CXL HDM Decoder n Control Register (Offset 20h*n+20h)

Bit Location	Attributes	Description
3:0	RWL	<p>Interleave Granularity (IG): The number of consecutive bytes that are assigned to each target in the Target List.</p> <ul style="list-style-type: none"> • 0h = 256 Bytes • 1h = 512 Bytes • 2h = 1024 Bytes (1 KB) • 3h = 2048 Bytes (2 KB) • 4h = 4096 Bytes (4 KB) • 5h = 8192 Bytes (8 KB) • 6h = 16384 Bytes (16 KB) • All other encodings are reserved <p>The device reports its desired interleave setting via the Desired_Interleave field in the DVSEC CXL Range 1/Range 2 Size Low register.</p> <p>The locking behavior is described in Section 8.2.4.20.13.</p> <p>Default value of this field is 0h.</p>
7:4	RWL	<p>Interleave Ways (IW): The number of targets across which Decoder n memory range is interleaved.</p> <ul style="list-style-type: none"> • 0h = 1 way (no interleaving) • 1h = 2-way interleaving • 2h = 4-way interleaving • 3h = 8-way interleaving • 4h = 16-way interleaving (valid only for CXL.mem devices)¹ • 8h = 3-way interleaving (valid only for CXL.mem devices)¹ • 9h = 6-way interleaving (valid only for CXL.mem devices)¹ • Ah = 12-way interleaving (valid only for CXL.mem devices)¹ • All other encodings are reserved <p>The locking behavior is described in Section 8.2.4.20.13.</p> <p>Default value of this field is 0h.</p>
8	RWL	<p>Lock On Commit: If set, all RWL fields in Decoder n shall become read only when Committed changes to 1.</p> <p>The locking behavior is described in Section 8.2.4.20.13.</p> <p>Default value of this bit is 0.</p>
9	RWL	<p>Commit: Software sets this to 1 to commit Decoder n.</p> <p>The locking behavior is described in Section 8.2.4.20.13.</p> <p>Default value of this bit is 0.</p> <p>A 1 to 0 transition of this bit shall cause the associated Committed bit to transition from 1 to 0.</p>
10	RO	<p>Committed: If 1, indicates Decoder n is active.</p>
11	RO	<p>Error Not Committed: If 1, indicates that the programming of Decoder n had an error and Decoder n is not active.</p>

Bit Location	Attributes	Description
12	RWL/RO	<p>Target Range Type: Formerly known as Target Device Type. This bit is RWL for CXL Host Bridges, and Upstream Switch Ports. This bit is permitted to be RO for devices that do not support this reconfigurability and it may return the value of 0 or 1 to represent the only coherency model that the devices support.</p> <ul style="list-style-type: none"> • 0 = Target is a Device Coherent Address range (HDM-D or HDM-DB) • 1 = Target is a Host-Only Coherent Address range (HDM-H) <p>The locking behavior is described in Section 8.2.4.20.13. Default value of this bit is 0.</p>
13	RWL/ RsvdP	<p>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 1.²</p> <ul style="list-style-type: none"> • 0 = Device is not permitted to issue BISnp requests to this range • 1 = Device is permitted to issue BISnp requests to this range
14	RWL	<p>UIO: This bit is RWL if the UIO Capable bit in the CXL HDM Decoder Capability register (see Section 8.2.4.20.1) is set; otherwise, it is permitted to be hardwired to 0. Software must not set this bit unless the UIO Capable bit is set to 1. Default value of this bit is 0.</p> <p>See Table 9-18 for how various components utilize the setting of this bit during the processing of UIO messages.²</p>
15	RsvdP	Reserved
19:16	RWL/ RsvdP	<p>Upstream Interleave Granularity (UIG): The aggregate interleave granularity applied to the HPA by the HDM decode stages that are upstream of this port. For enumeration of legal values, see the definition of Interleave Granularity in this register.²</p> <p>This bit is RWL for a switch and a Host Bridge if the UIO Capable bit in the CXL HDM Decoder Capability register (see Section 8.2.4.20.1) is set.</p> <p>This field is reserved for CXL.mem devices. This field is also reserved for switches and Host Bridges that are not UIO capable.</p> <p>Default value of this field is 0h.</p>
23:20	RWL/ RsvdP	<p>Upstream Interleave Ways (UIW): The aggregate Interleave granularity ways produced by HDM decode stages that are upstream of this port. For enumeration of legal values, see the definition of Interleave Ways in this register.²</p> <p>This bit is RWL for a switch and a Host Bridge if the UIO Capable bit in the CXL HDM Decoder Capability register (see Section 8.2.4.20.1) is set.</p> <p>This field is reserved for CXL.mem devices. This field is also reserved for switches and Host Bridges that are not UIO capable.</p> <p>Default value of this field is 0h.</p>
27:24	RWL/ RsvdP	<p>Interleave Set Position (ISP): The position of this component in the interleave set formed when all HDM decode stages that are upstream of this port are considered. Expressed as a 0-based quantity. For a switch and a Host Bridge, ISP must be configured to a value that is lower than the number of Upstream Interleave Ways (the raw value, not the encoded value); otherwise, the results are undefined.²</p> <p>This bit is RWL for a CXL.mem devices, a switch, and a Host Bridge if the UIO Capable bit in the CXL HDM Decoder Capability register (see Section 8.2.4.20.1) is set.</p> <p>This field is reserved for CXL.mem devices, switches, and Host Bridges that are not UIO capable.</p> <p>Default value of this field is 0h.</p>
31:28	RsvdP	Reserved

1. Introduced as part of Version=2.

2. Introduced as part of Version=3.

IMPLEMENTATION NOTE

UIW, UIG, and ISP Examples

The switch in [Figure 9-16](#) receives all the HPAs within the range 16-20 TB because interleaving is not performed upstream to the switch. If the switch is capable of routing UIO accesses to CXL.mem, then the HDM decoder that spans 16-20 TB in that switch should be configured as follows:

- UIW=0
- ISP=0
- UIG=Any legal value

The 4 CXL.mem devices, from left to right, are assigned ISP=0 through 3, respectively.

The switch in [Figure 9-17](#) receives every other 4K HPA chunk when the host accesses the range 16-20 TB because the Host Bridge is configured to 2-way interleave at 4K granularity. If the switch is capable of routing UIO accesses to CXL.mem, then the HDM decoder that spans 16-20 TB in that switch should be configured as follows:

- UIW=1 (every other chunk, so 2-way)
- ISP=0 because the switch shown in the figure receives the first chunk (ISP=1 for the switch is not shown in the figure)
- UIG= 4 (every chunk is 4K)

The 8 CXL.mem devices, from left to right, are assigned ISP=0 through 7, respectively.

The switch in [Figure 9-18](#) receives every 4th 2K HPA chunk when the host accesses the range 16-20 TB. If the switch is capable of routing UIO accesses to CXL.mem, then the HDM decoder that spans 16-20 TB in that switch should be configured as follows:

- UIW=2 (every fourth chunk, so 4-way)
- ISP=0 because the switch receives the first chunk
- UIG= 3 (every chunk is 2K)

The 8 CXL.mem devices, from left to right, are assigned ISP=0 through 7, respectively.

8.2.4.20.8 CXL HDM Decoder n Target List Low Register (Offset 20h*n+24h)

This register is not applicable to devices, which use this field as DPA Skip Low as described in [Section 8.2.4.20.9](#). The targets must be distinct and the identifier cannot repeat. For example, Target Port Identifiers for Interleave Way=0, 1, 2, 3 must be distinct if Control.IW=2.

The Target Port Identifier for a given Downstream Port is reported via the Port Number field in the Link Capabilities register (see PCIe Base Specification).

Bit Location	Attributes	Description
7:0	RWL	Target Port Identifier for Interleave Way=0: The locking behavior is described in Section 8.2.4.20.13 . Default value of this field is 00h.
15:8	RWL	Target Port Identifier for Interleave Way=1: Valid if Decoder n Control.IW>0. The locking behavior is described in Section 8.2.4.20.13 . Default value of this field is 00h.
23:16	RWL	Target Port Identifier for Interleave Way=2: Valid if Decoder n Control.IW>1. The locking behavior is described in Section 8.2.4.20.13 . Default value of this field is 00h.
31:24	RWL	Target Port Identifier for Interleave Way=3: Valid if Decoder n Control.IW>1. The locking behavior is described in Section 8.2.4.20.13 . Default value of this field is 00h.

8.2.4.20.9 CXL HDM Decoder n DPA Skip Low Register (Offset 20h*n + 24h)

This register is applicable only to devices. For non-devices, this field contains the Target List Low register as described in [Section 8.2.4.20.8](#).

Bit Location	Attributes	Description
27:0	RsvdP	Reserved
31:28	RWL	DPA Skip Low: Corresponds to bits 31:28 of the DPA Skip length which, when nonzero, specifies a length of DPA space that is skipped, unmapped by any decoder, prior to the HPA-to-DPA mapping provided by this decoder. Default value of this field is 0h.

8.2.4.20.10 CXL HDM Decoder n Target List High Register (Offset 20h*n+28h)

This register is not applicable to devices, which use this field as DPA Skip High as described in [Section 8.2.4.20.11](#). Returns the Target Port associated with Interleave Way 4 through 7.

The targets must be distinct. For example, all 8 Target Port Identifiers must be distinct if Control.IW=3.

Bit Location	Attributes	Description
7:0	RWL	Target Port Identifier for Interleave Way=4: Valid if Decoder n Control.IW>2. The locking behavior is described in Section 8.2.4.20.13 . Default value of this field is 00h.
15:8	RWL	Target Port Identifier for Interleave Way=5: Valid if Decoder n Control.IW>2. The locking behavior is described in Section 8.2.4.20.13 . Default value of this field is 00h.

Bit Location	Attributes	Description
23:16	RWL	Target Port Identifier for Interleave Way=6: Valid if Decoder n Control.IW>2. The locking behavior is described in Section 8.2.4.20.13 . Default value of this field is 00h.
31:24	RWL	Target Port Identifier for Interleave Way=7: Valid if Decoder n Control.IW>2. The locking behavior is described in Section 8.2.4.20.13 . Default value of this field is 00h.

8.2.4.20.11 CXL HDM Decoder n DPA Skip High Register (Offset 20h*n + 28h)

This register is applicable only to devices. For non-devices, this field contains the Target List High register as described in [Section 8.2.4.20.10](#).

Bit Location	Attributes	Description
31:0	RWL	DPA Skip High: Corresponds to bits 63:32 of the DPA Skip length which, when nonzero, specifies a length of DPA space that is skipped, unmapped by any decoder, prior to the HPA-to-DPA mapping provided by this decoder. Default value of this register is 0000 0000h.

8.2.4.20.12 Committing Decoder Programming

If Software intends to set Lock On Commit, Software must configure the decoders in order. In other words, decoder m must be configured and committed before decoder m+1 for all values of m. Decoder m must cover an HPA range that is below decoder m+1.

Each interleave decoder must be committed before it actively decodes CXL.mem transactions. Software configures all the registers associated with the individual decoder and optionally sets the Lock On Commit bit prior to setting the Commit bit. When the Commit bit in decoder m+1 transitions from 0 to 1 and Lock On Commit=1, the decoder logic shall perform the following consistency checks before setting Committed bit:

- Decoder[m+1].Base >= (Decoder[m].Base+Decoder[m].Size). This ensures that the Base of the decoder being committed is greater than or equal to the limit of the previous decoder. This check is not applicable when committing Decoder 0.
- Decoder[m+1].Base <= Decoder[m+1].Base+Decoder[m+1].Size (no wraparound)
- If Decoder[m+1].IW>=8, Decoder[m+1].Size is a multiple of 3.
- Target Port Identifiers for Interleave Way=0 through 2**IW -1 must be distinct. This ensures no two interleave ways are pointing to the same target.
- Decoder[m].Committed=1. This ensures that the previous decoder is committed and has passed the above checks.

Decoder logic does not allow Decoder[m+1] registers to be modified while these checks are in progress (Commit=1, (Committed OR ErrorNotCommitted)=0). If software attempts to modify Decoder[m+1] while the checks are in progress, it will lead to undefined behavior.

These checks ensure that all decoders within a given component are self-consistent and do not create aliasing.

It is legal for software to program Decoder Size to 0 and commit it. Such a decoder will not participate in HDM decode.

If these checks fail and the decoder is not committed, decoder logic shall set *Error Not Committed* flag. Software may remedy this situation by clearing the Commit bit, reprogramming the decoder with legal values and setting Commit bit once again.

If Lock On Commit=0, decoder logic does not implement the address aliasing checks. Software is fully responsible for avoiding aliasing and protecting the HDM Decoder registers via other mechanisms such as CPU page tables.

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 the BI bit in the HDM Decoder Control register is set before committing the HDM decoder; otherwise, the device operation is undefined. Software must ensure that the device and any applicable DSPs, USPs, and the RP are configured such that the device is able to issue a BISnp request before committing any HDM decoder with the 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.

8.2.4.20.13 Decoder Protection

Software may choose to set the Lock On Commit bit prior to setting Commit. If the Lock On Commit bit is 1, Decoder logic shall perform alias checks listed in the previous section prior to committing the decoder and further disallow modifications to all RWL fields in that decoder when the decoder is in Committed state.

If the Lock On Commit bit is 0, software may clear the Commit bit, reprogram the decoder fields, and then set the Commit bit again for the new values to take effect. Reprogramming the decoder while the Commit bit is set results in undefined behavior. To avoid misbehavior, software is responsible for quiescing memory traffic that is targeting the decoder while it is being reprogrammed. If decoder logic does not positively decode an address of a read, it may either return all 1s or return poison based on the CXL HDM Decoder Global Control register setting. During reprogramming, software must follow the same restrictions as the initial programming. Specifically, decoder m must be configured and committed before decoder m+1 for all values of m; Decoder m must cover an HPA range that is below decoder m+1 and all Targets must be distinct.

IMPLEMENTATION NOTE

Software may set Lock On Commit=1 in systems that do not support Hot-Plug. In such systems, the decoders are generally programmed at boot, can be arranged in increasing HPA order and never modified until the next reset.

If the system supports CXL Hot-Plug, software may need significant flexibility in terms of reprogramming the decoders during runtime. In such systems, software may choose to leave Lock On Commit=0.

IMPLEMENTATION NOTE

CXL Host Bridge and Upstream Switch Port Decode Flow

Step 1: Check if the incoming HPA satisfies $\text{Base} \leq \text{HPA} < \text{Base} + \text{Size}$ for any active decoder. If no decoder satisfies this equation for a write, drop the writes. If no decoder satisfies this equation for a read and Poison On Decode Error Enable=0, return all 1s. If no decoder satisfies this equation for a read and Poison On Decode Error Enable=1, return poison.

Step 2: If Decoder[n] satisfies this equation:

- Extract IW bits starting with bit position IG+8 in HPA¹. This returns the Interleave Way
- Send transactions to Downstream Port=Decoder[n].Target List[Interleave Way]

Example:

- HPA = 129 GB + 1028d
- Decoder[2].Base= 128 GB, Decoder[2].Size = 4 GB.
- Assume IW=2 (4 way), IG = 1 (512 bytes).

Step 1: Decoder[2] positively decodes this address, so n=2.

Step 2: Extracting bits 10:9 from HPA returns Interleave Way=2 (HPA=... xxxx 0000 0100 0000 0100b)

Forward access to Port number Decoder[2].Target List Low[23:16]

1. In the general case, the bits must be extracted from $(\text{HPA} - \text{Base}[n])$. However, Decoder Base is a multiple of 256M and the highest interleave granularity is 16K. Therefore, extracting IW bits from HPA still returns the correct Index value.

IMPLEMENTATION NOTE**Device Decode Logic**

As part of Commit processing flow, the device decoder logic may accumulate DPABase field for every decoder as follows:

- Decoder[0].DPABase = Decoder[0].DPASkip
- If IW <8, Decoder[m+1].DPABase = Decoder[m+1].DPASkip + Decoder[m].DPABase + (Decoder[m].Size / 2 ** Decoder[m].IW)
- If IW >=8, Decoder[m+1].DPABase = Decoder[m+1].DPASkip + Decoder[m].DPABase + (Decoder[m].Size / (3 * 2 ** Decoder[m].IW-8))

DPABase is not exposed to software, but may be tracked internally by the decoder logic to speed up decode process. Decoder[m].DPABase represents the lowest DPA that the lowest HPA decoded by Decoder[m] maps to. The DPA mappings for a device typically start at DPA 0 for Decoder[0] and are sequentially accumulated with each additional decoder used; however, the DPASkip field in the decoder may be used to leave ranges of DPA unmapped, as required by the needs of the platform.

During the decode:

Step 1: Check if the incoming HPA satisfies $\text{Base} \leq \text{HPA} < \text{Base} + \text{Size}$ for any active decoder. If no decoder satisfies this equation for a write, drop the writes. If no decoder satisfies this equation for a read and Poison On Decode Error Enable=0, return all 1s. If no decoder satisfies this equation for a read and Poison On Decode Error Enable=1, return poison.

Step 2: If Decoder[n] satisfies this equation.

- Calculate HPAOffset = HPA - Decoder[n].Base
- If IW <8, removes IW bits starting with bit position IG+8 in HPAOffset to get DPAOffset. This operation will right shift the bits above IG+IW+8 by IW positions.
 $\text{DPAOffset}[51:\text{IG}+8]=\text{HPAOffset}[51:\text{IG}+8+\text{IW}]$
 $\text{DPAOffset}[\text{IG}+7:0]=\text{HPAOffset}[\text{IG}+7:0]$.
- If IW >=8, an integer, divide by 3 operation is involved
 $\text{DPAOffset}[51:\text{IG}+8]=\text{HPAOffset}[51:\text{IG}+\text{IW}]/3$
 $\text{DPAOffset}[\text{IG}+7:0]=\text{HPAOffset}[\text{IG}+7:0]$
- DPA=DPAOffset + Decoder[n].DPABase.

The above calculation is applied by the device regardless of the Interleave Arithmetic field in the corresponding CFMWS entry.

Example:

- HPA = 129 GB + 1028d
- Software programmed Decoder[0].Base= 32 GB, Decoder[0].Size = 32 GB.
- Software programmed Decoder[1].Base= 128 GB, Decoder[1].Size = 4 GB.
- Assume IW=3 (8 way), IG = 1 (512 bytes) for both decoders.
- Decoder[1].DPABase= 32/8 GB = 4 GB

Step 1: Select Decoder[1].

Step 2:

- HPAOffset = 1 GB + 1028d (4000 0404h, 0404h=0000 0100 0000 0100b)
- Removing bits 11:9 from HPA returns DPAOffset=800 0004h.

Add DPABase 4 GB to get DPA= 1 0800 0004h.

Example 2:

- HPA = 128 GB + 24920d
- Software programmed Decoder[0].Base= 32 GB, Decoder[0].Size = 48 GB.
- Software programmed Decoder[1].Base= 128 GB, Decoder[1].Size = 24 GB.
- Assume IW=10 (12 way), IG = 1 (512 bytes) for both decoders. Notice the Size of both decoders is a multiple of 3.
- Decoder[1].DPABase= 48 GB/12 = 4 GB

Step 1: Select Decoder[1].

Step 2:

- HPAOffset = 24920d=0110 0001 0101 1000h
- DPAOffset[51:9]=HPAOffset[51:11]/3d= 0 1100b /3=04h
- DPAOffset[8:0]= HPAOffset[8:0]=1 0101 1000b
- DPAOffset=1001 0101 1000b=958h

Add DPABase 4 GB to get DPA= 1 0000 0958h.

8.2.4.21 CXL Extended Security Capability Structure

This capability structure applies only to the CXL Host Bridge and may be located in CHBCR.

Offset	Register Name
00h	CXL Extended Security Structure Entry Count.n (Max 256)
04h	Root Port 1 Security Policy
08h	Root Port 1 ID
0Ch	Root Port 2 Security Policy
10h	Root Port 2 ID
...	...
8*n-4	Root Port n Security Policy
8*n	Root Port n ID

Table 8-28. CXL Extended Security Structure Entry Count (Offset 00h)

Bit Location	Attributes	Description
7:0	HwInit	Root Port Count: The number of Extended Security Structures that are part of this capability structure. The number of entries must match the CXL.cache-capable Root Ports that are associated with this Host Bridge. Each entry consists of two DWORD registers - Security Policy and Root Port ID.
31:8	RsvdP	Reserved

Table 8-29. Root Port n Security Policy Register (Offset 8*n-4)

Bit Location	Attributes	Description
1:0	RW	Trust Level: If the host supports only 1 CXL.cache device per VCS, this field defines the Trust Level for the CXL.cache Device below Root Port n (see Table 8-26 for definition of this field). If the host supports more than 1 CXL.cache device per VCS, this field defines the Trust Level for the CXL.cache device below this root port that is operating in HDM-D mode. The CXL Cache ID Decoder Control register (see Section 8.2.4.29.2) describes if such a device is present and if present, the associated Cache ID. Default value of this field is 10b.
2	RW	Block CXL.cache HDM-DB: This bit controls how the root port handles CXL.cache requests from the set of devices that are using HDM-DB flow. The CXL Cache ID Decoder Control register (see Section 8.2.4.29.2) identifies if a device below this root port is using in HDM-D flow and the associated Cache ID, if applicable. ¹ <ul style="list-style-type: none"> • 0 = CXL.cache requests from any device using HDM-DB flow shall be permitted subject to other checks (see Section 9.15.2) • 1 = CXL.cache requests from any device using HDM-DB flow shall be unconditionally aborted by the root port Default value of this bit is 1.
31:3	RsvdP	Reserved

1. This bit was introduced as part of Version=2.

Table 8-30. Root Port n ID Register (Offset 8*n)

Bit Location	Attributes	Description
7:0	HwInit	Port Identifier of Root Port n (referenced using the Port Number field in the Link Capabilities register (see PCIe Base Specification)).
31:8	RsvdP	Reserved

8.2.4.22 CXL IDE Capability Structure

Offset	Register Name
00h	CXL IDE Capability Register
04h	CXL IDE Control
08h	CXL IDE Status
0Ch	CXL IDE Error Status
10h	Key Refresh Time Capability
14h	Truncation Transmit Delay Capability
18h	Key Refresh Time Control
1Ch	Truncation Transmit Delay Control
20h	Key Refresh Time Capability2

8.2.4.22.1 CXL IDE Capability (Offset 00h)

Bit Location	Attributes	Description
0	HwInit	CXL IDE Capable: When set, indicates that the Port supports CXL IDE
16:1	HwInit	Supported CXL IDE Modes <ul style="list-style-type: none"> • Bit[1]: If set, Skid mode is supported. • Bit[2]: If set, Containment mode is supported. If bit 0 of this register is set, this bit must be set as well. • Bits[16:3]: Reserved.
21:17	HwInit	Supported Algorithms: Indicates the supported algorithms for securing CXL IDE, encoded as: <ul style="list-style-type: none"> • 00h = AES-GCM 256-bit key size, 96-bit MAC • All other encodings are reserved
22	HwInit/RsvdP	IDE.Stop Capable: Indicates that the port Tx supports generation of IDE.Stop control flit and the port Rx supports processing of IDE.Stop control flit when operating in 256B Flit mode (see Section 11.3.10). This bit is reserved for ports that are not capable of operating in 256B Flit mode. ¹

Bit Location	Attributes	Description
23	HwInit/RsvdP	<p>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).²</p> <p>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 the 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. After IDE is terminated, the System Firmware or System Software may set the CXL_Latency_Optimized_256B_Flit_Enable bit in the Downstream Port and then retrain the link so that the link can transition to Latency-Optimized 256B Flit mode.</p>
31:24	RsvdP	Reserved

1. This bit was introduced as part of Version=2.
2. This bit was introduced as part of Version=3.

8.2.4.22.2 CXL IDE Control (Offset 04h)

Bit Location	Attributes	Description
0	RW	<p>PCRC Disable: When set, PCRC generation is disabled and MAC calculation does not include PCRC. Software must ensure that this bit is programmed consistently on both ends of the CXL link.</p> <p>Changes to this bit when CXL.cachemem IDE is active results in undefined behavior.</p> <p>The default value of this bit is 0.</p>
1	RW/RsvdP	<p>IDE.Stop Enable: Enables generation of IDE.Stop control flit by the port Tx and processing of IDE.Stop control flit by port Rx when operating in 256B Flit mode.¹</p> <p>This bit must be RW if the IDE Stop Capable bit is set; otherwise, it is permitted to be hardwired to 0. Software must not set this bit unless the IDE.Stop Capable bit is set to 1.</p> <p>The default value of this bit is 0.</p>
31:2	RsvdP	Reserved

1. This bit was introduced as part of Version=2.

8.2.4.22.3 CXL IDE Status (Offset 08h)

Bit Location	Attributes	Description
3:0	RO	<p>Rx IDE Status</p> <ul style="list-style-type: none"> • 0h = Reserved • 1h = Active Containment mode • 2h = Active Skid mode • 4h = Insecure State • All other encodings are reserved
7:4	RO	<p>Tx IDE Status</p> <ul style="list-style-type: none"> • 0h = Reserved • 1h = Active Containment mode • 2h = Active Skid mode • 4h = Insecure State • All other encodings are reserved
31:8	RsvdZ	Reserved

8.2.4.22.4 CXL IDE Error Status (Offset 0Ch)

Bit Location	Attributes	Description
3:0	RW1CS	<p>Rx Error Status: Describes the error condition that transitioned the link to Insecure State if IDE stream is active. The component behavior upon this transition is defined in Section 11.3.8.</p> <ul style="list-style-type: none"> • 0h = No Error • 1h = Integrity failure on received secure traffic • 2h = MAC or Truncated MAC received when the link is not in secure mode (when integrity is not enabled and the receiver detects MAC header) • 3h = MAC header received when not expected (No MAC epoch running, but the receiver detects a MAC header) • 4h = MAC header is not received when expected (MAC header not received within 6 flits after MAC epoch has terminated) • 5h = Truncated MAC flit is received when not expected (if the receiver gets truncated MAC flit corresponding to a completed MAC epoch) • 6h = After early MAC termination, the receiver detects a protocol flit before the truncation delay • 7h = This error code encompasses the following conditions: <ul style="list-style-type: none"> — Protocol flit received earlier than expected after key change (see Section 11.3.7 for the detailed timing requirements) — Rx IDE Stop.Enable=1 and a protocol flit received earlier than expected after an IDE Termination Handshake (see Section 11.3.10 for the detailed timing requirements) • 8h = CXL.cachemem IDE Establishment Security error. This error code encompasses the following conditions: <ul style="list-style-type: none"> — IDE.Start is received prior to a successful CXL_KEY_PROG since the last Conventional Reset — IDE.Start is received prior to a successful CXL_KEY_PROG since the last IDE.Start — IDE.Start is received prior to a successful CXL_K_SET_GO since the last Conventional Reset — IDE.Start is received prior to a successful CXL_K_SET_GO since the last IDE.Start — CXL_IDE_KM message received over a different SPDM session (see Section 11.4.2) — IDE.Start is received in the middle of a MAC epoch (see Section 11.3.7) • All other encodings are reserved
7:4	RW1CS	<p>Tx IDE Status</p> <ul style="list-style-type: none"> • 0h = No Error • All other encodings are reserved
8	RW1CS	<p>Unexpected IDE.Stop Received: This bit is set by the Rx port upon the following conditions:</p> <ul style="list-style-type: none"> • Received IDE.Stop Link Layer Control flit while CXL.cachemem IDE is active, but prior to a successful CXL_K_SET_STOP since the last IDE.Start (see Section 11.4.6) • Received IDE.Stop Link Layer Control flit while IDE Stop.Enable=0 and IDE Stop.Capable=1 (see Section 11.3.10) • Received IDE.Stop Link Layer Control flit while IDE session is not active (see Section 11.3.10) • Valid TMAC sequence not received before IDE.Stop (see Section 11.3.10) <p>In all of these cases, the Rx shall drop the IDE.Stop but shall not terminate the CXL.cachemem IDE session if one is active.</p>
31:9	RsvdZ	Reserved

8.2.4.22.5 Key Refresh Time Capability (Offset 10h)

Bit Location	Attributes	Description
31:0	HwInit	Rx Min Key Refresh Time: Number of IDE.Idle flits the receiver needs before it is ready to receive protocol flits after IDE.Start is received when operating in 68B Flit mode. Tx Key Refresh Time (see Section 8.2.4.22.7) field of the transmitter is configured by System Software to block transmission of protocol flits for at least this duration when switching keys (see Section 11.3.7) or terminating IDE (see Section 11.3.10) when the link is operating in 68B Flit mode.

8.2.4.22.6 Truncation Transmit Delay Capability (Offset 14h)

Bit Location	Attributes	Description
7:0	HwInit	Rx Min Truncation Transmit Delay: Number of IDE.Idle flits the receiver needs before it is ready to receive protocol flits after a Truncated MAC is received when operating in 68B Flit mode. The Tx Truncation Transmit Delay (see Section 8.2.4.22.8) field of the transmitter is configured, by software, to block transmission of protocol flits for at least this duration when the link is operating in 68B Flit mode.
15:8	HwInit	Rx Min Truncation Transmit Delay2: Number of IDE.Idle flits the receiver needs before it is ready to receive protocol flits after a Truncated MAC is received when operating in 256B Flit mode. The Tx Truncation Transmit Delay (see Section 8.2.4.22.8) field of the transmitter is configured, by software, to block transmission of protocol flits for at least this duration when the link is operating in 256B Flit mode. ¹
31:16	RsvdP	Reserved

1. This field was introduced as part of Version=2.

8.2.4.22.7 Key Refresh Time Control (Offset 18h)

Bit Location	Attributes	Description
31:0	RW	Tx Key Refresh Time: For 68B Flit mode, this register represents the minimum number of flits that the transmitter needs to block transmission of protocol flits after IDE.Start has been sent. For 256B Flit mode, this register represents the minimum number of flits that the transmitter needs to block transmission of protocol flits after IDE.Start has been sent or after IDE.Stop has been sent. Used when switching keys (see Section 11.3.7) or gracefully terminating IDE (256B Flit mode only, see Section 11.3.10). The default value of this field is 0.

8.2.4.22.8 Truncation Transmit Delay Control (Offset 1Ch)

Bit Location	Attributes	Description
7:0	RW	Tx Truncation Transmit Delay: Configuration parameter to account for the potential discarding of any precomputed values by the receiver. This parameter feeds into the computation of the minimum number of IDE.Idle flits that the Transmitter needs to send after sending a truncated MAC flit. See Equation 11-1 . The default value of this field is 00h.
31:8	RsvdP	Reserved

8.2.4.22.9 Key Refresh Time Capability2 (Offset 20h)

Bit Location	Attributes	Description ¹
31:0	HwInit	Rx Min Key Refresh Time2: Number of IDE.Idle flits the receiver needs to be ready to receive protocol flits after either IDE.Start or IDE.Stop is received when operating in 256B Flit mode. Tx Key Refresh Time (see Section 8.2.4.22.7) field of the transmitter is configured by System Software to block transmission of protocol flits for at least this duration when switching keys (see Section 11.3.7) or terminating IDE (see Section 11.3.10) when the link is operating in 256B Flit mode.

1. This register was introduced as part of Version=2.

8.2.4.23 CXL Snoop Filter Capability Structure

Offset	Register Name
00h	Snoop Filter Group ID
04h	Snoop Filter Capacity

8.2.4.23.1 Snoop Filter Group ID (Offset 00h)

Bit Location	Attributes	Description
15:0	HwInit	Group ID: Uniquely identifies a snoop filter instance that is used to track CXL.cache devices below this Port. All Ports that share a single Snoop Filter instance shall set this field to the same value.
31:16	RsvdP	Reserved

8.2.4.23.2 Snoop Filter Effective Size (Offset 04h)

Bit Location	Attributes	Description
31:0	HwInit	Capacity: Effective Snoop Filter Capacity representing the size of cache that can be effectively tracked by the Snoop Filter with this Group ID, in multiples of 64K.

8.2.4.24 CXL Timeout and Isolation Capability Structure

Offset	Register Name
00h	CXL Timeout and Isolation Capability Register
04h	Reserved
08h	CXL Timeout and Isolation Control Register
0Ch	CXL Timeout and Isolation Status Register

8.2.4.24.1 CXL Timeout and Isolation Capability Register (Offset 00h)

Bit Location	Attributes	Description
3:0	RO	<p>CXL.mem Transaction Timeout Ranges Supported: This field indicates support for transaction timeout ranges on CXL.mem.</p> <p>Four time value ranges are defined:</p> <ul style="list-style-type: none"> • Range A: Default range: 50us to 10ms. • Range B: 10ms to 250ms • Range C: 250ms to 4s • Range D: 4s to 64s <p>Bits are set according to the values listed below to show the supported timeout value ranges:</p> <ul style="list-style-type: none"> • 0h = Transaction Timeout programming is not supported – the function must implement a timeout value within the range of 50us to 10ms. • 1h = Range A • 2h = Range B • 3h = Ranges A and B • 6h = Ranges B and C • 7h = Ranges A, B, and C • Eh = Ranges B, C, and D • Fh = Ranges A, B, C, and D • All other encodings are reserved
4	RO	CXL.mem Transaction Timeout Supported: The value of 1 indicates support for CXL.mem Transaction Timeout mechanism.
7:5	RsvdP	Reserved
11:8	RO	<p>CXL.cache Transaction Timeout Ranges Supported: This field indicates support for transaction timeout ranges on CXL.cache.</p> <p>Four time value ranges are defined:</p> <ul style="list-style-type: none"> • Range A: Default range: 50us to 10ms. • Range B: 10ms to 250ms • Range C: 250ms to 4s • Range D: 4s to 64s <p>Bits are set according to the values listed below to show the supported timeout value ranges:</p> <ul style="list-style-type: none"> • 0h = Transaction Timeout programming is not supported – the function must implement a timeout value within the range of 50us to 10ms. • 1h = Range A • 2h = Range B • 3h = Ranges A and B • 6h = Ranges B and C • 7h = Ranges A, B, and C • Eh = Ranges B, C, and D • Fh = Ranges A, B, C, and D • All other encodings are reserved
12	RO	CXL.cache Transaction Timeout Supported: The value of 1 indicates support for CXL.cache Transaction Timeout mechanism.
15:13	RsvdP	Reserved
16	RO	CXL.mem Isolation Supported: This bit indicates support for Isolation on CXL.mem.
17	RO	CXL.mem Isolation Link Down Supported: This bit indicates support for triggering of Link Down on the CXL port if CXL.mem enters Isolation mode. This bit can only be set to 1 if the CXL.mem Isolation Supported bit is also set to 1.
18	RO	CXL.cache Isolation Supported: This bit indicates support for Isolation on CXL.cache.

EVALUATION COPY

Bit Location	Attributes	Description
19	RO	CXL.cache Isolation Link Down Supported: This bit indicates support for triggering of Link Down on the CXL Root Port if CXL.cache enters Isolation mode. This bit can only be set to 1 if the CXL.cache Isolation Supported bit is also set to 1.
24:20	RsvdP	Reserved
25	RO	Isolation ERR_COR Signaling Supported: If set, this bit indicates that the Root Port supports the ability to signal with ERR_COR when Isolation is triggered.
26	RO	Isolation Interrupt Supported: This bit indicates support for signaling an interrupt when Isolation is triggered.
31:27	RO	<p>Isolation Interrupt Message Number: This field indicates which MSI/MSI-X vector is used for the interrupt message generated in association with the CXL Timeout and Isolation Capability structure. This field is valid only if Isolation Interrupt Supported is 1.</p> <p>For MSI, the value in this field indicates the offset between the base Message Data and the interrupt message that is generated. Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the Multiple Message Enable field in the Message Control register for MSI.</p> <p>For MSI-X, the value in this field indicates which MSI-X Table entry is used to generate the interrupt message. The entry must be one of the first 32 entries even if the Function implements more than 32 entries. For a given MSI-X implementation, the entry must remain constant.</p> <p>If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software is permitted to enable only one mechanism at a time. If MSI-X is enabled, the value in this field must indicate the vector for MSI-X. If MSI is enabled or neither is enabled, the value in this field must indicate the vector for MSI. If software enables both MSI and MSI-X at the same time, the value in this field is undefined.</p>

8.2.4.24.2 CXL Timeout and Isolation Control Register (Offset 08h)

Bit Location	Attributes	Description
3:0	RW/RO	<p>CXL.mem Transaction Timeout Value: In CXL Root Port Functions that support Transaction Timeout programmability, this field allows system software to modify the Transaction Timeout Value for CXL.mem. Functions that support Transaction Timeout programmability must support the values given below corresponding to the programmability ranges indicated in the CXL.mem Transaction Timeout Ranges Supported field.</p> <p>Defined encodings:</p> <ul style="list-style-type: none"> • 0h = Default range: 50us to 10ms • Values available if Range A (50us to 10ms) is supported: <ul style="list-style-type: none"> — 1h = 50us to 100us — 2h = 1ms to 10ms • Values available if Range B (10ms to 250ms) is supported: <ul style="list-style-type: none"> — 5h = 16ms to 55ms — 6h = 65ms to 210ms • Values available if Range C (250ms to 4s) is supported: <ul style="list-style-type: none"> — 9h = 260ms to 900ms — Ah = 1s to 3.5s • Values available if Range D (4s to 64s) is supported: <ul style="list-style-type: none"> — Dh = 4s to 13s — Eh = 17s to 64s • All other encodings are reserved <p>Software is permitted to change the value in this field at any time. For Requests already pending when the Transaction Timeout Value is changed, hardware is permitted to use either the new or the old value for the outstanding Requests and is permitted to base the start time for each Request on either the time this value was changed or the time each request was issued.</p> <p>This field must be RW if the CXL.mem Transaction Timeout Supported bit is set; otherwise, it is permitted to be hardwired to 0h.</p> <p>The default value for this field is 0h.</p>
4	RW/RO	<p>CXL.mem Transaction Timeout Enable: When set, this bit enables CXL.mem Transaction Timeout detection mechanism.</p> <p>Software is permitted to set or clear this bit at any time. If there are outstanding Transaction when the bit is set, it is permitted but not required for hardware to apply the completion timeout mechanism to the outstanding Transactions. If this is done, it is permitted to base the start time for each Transaction on either the time this bit was set or the time each Request was issued.</p> <p>This bit must be RW if the CXL.mem Transaction Timeout Supported bit is set; otherwise, it is permitted to be hardwired to 0. Software must not set this bit unless the CXL.mem Transaction Timeout Supported bit is set.</p> <p>The default value for this bit is 0.</p>
7:5	RsvdP	Reserved

EVALUATION COPY

Bit Location	Attributes	Description
11:8	RW/RO	<p>CXL.cache Transaction Timeout Value: In CXL Root Port Functions that support Transaction Timeout programmability, this field allows system software to modify the Transaction Timeout Value for CXL.cache.</p> <p>Functions that support Transaction Timeout programmability must support the values given below corresponding to the programmability ranges indicated in the CXL.cache Transaction Timeout Ranges Supported field.</p> <p>Defined encodings:</p> <ul style="list-style-type: none"> • 0h = Default range: 50us to 10ms • Values available if Range A (50us to 10ms) is supported: <ul style="list-style-type: none"> — 1h = 50us to 100us — 2h = 1ms to 10ms • Values available if Range B (10ms to 250ms) is supported: <ul style="list-style-type: none"> — 5h = 16ms to 55ms — 6h = 65ms to 210ms • Values available if Range C (250ms to 4s) is supported: <ul style="list-style-type: none"> — 9h = 260ms to 900ms — Ah = 1s to 3.5s • Values available if Range D (4s to 64s) is supported: <ul style="list-style-type: none"> — Dh = 4s to 13s — Eh = 17s to 64s • All other encodings are reserved <p>Software is permitted to change the value in this field at any time. For Requests already pending when the Transaction Timeout Value is changed, hardware is permitted to use either the new or the old value for the outstanding Requests and is permitted to base the start time for each Request on either the time this value was changed or the time each request was issued.</p> <p>This bit must be RW if the CXL.cache Transaction Timeout Supported bit is set; otherwise, it is permitted to be hardwired to 0h.</p> <p>The default value for this field is 0h.</p>
12	RW/RO	<p>CXL.cache Transaction Timeout Enable: When set, this bit enables CXL.cache Transaction Timeout detection mechanism.</p> <p>Software is permitted to set or clear this bit at any time. If there are outstanding Transaction when the bit is set, it is permitted but not required for hardware to apply the completion timeout mechanism to the outstanding Transactions. If this is done, it is permitted to base the start time for each Transaction on either the time this bit was set or the time each Request was issued.</p> <p>This bit must be RW if the CXL.cache Transaction Timeout Supported bit is set; otherwise, it is permitted to be hardwired to 0. Software must not set this bit unless the CXL.cache Transaction Timeout Supported bit is set.</p> <p>The default value for this bit is 0.</p>
15:13	RW/RO	Reserved
16	RW/RO	<p>CXL.mem Isolation Enable: This field allows System Software to enable CXL.mem Isolation actions. If this field is set, Isolation actions will be triggered if either a CXL.mem Transaction Timeout is detected or if the CXL link went down.</p> <p>This bit must be RW if the CXL.mem Isolation Supported bit is set; otherwise, it is permitted to be hardwired to 0. Software must not set this bit unless the CXL.mem Isolation Supported bit is set. The software is required to quiesce the CXL.mem traffic passing through the Root Port when changing the state of this bit. If Software modifies this bit in the presence of CXL.mem traffic, the results are undefined.</p>
17	RW/RO	<p>CXL.mem Isolation Link Down Enable: When set, the CXL root port shall trigger a Link Down condition when CXL.mem enters Isolation.</p> <p>This bit must be RW if the CXL.mem Isolation Link Down Supported bit is set; otherwise, it is permitted to be hardwired to 0. Software must not set this bit unless the CXL.mem Isolation Link Down Supported bit is set.</p>

Bit Location	Attributes	Description
18	RW/RO	<p>CXL.cache Isolation Enable: This field allows System Software to enable CXL.cache Isolation actions. If this field is set, Isolation actions will be triggered if either a CXL.cache Transaction Timeout is detected or if the CXL link went down.</p> <p>This bit must be RW if the CXL.cache Isolation Supported bit is set; otherwise, it is permitted to be hardwired to 0. Software must not set this bit unless the CXL.cache Isolation Supported bit is set.</p> <p>The software is required to quiesce the CXL.cache traffic passing through the Root Port when changing the state of this bit. If Software modifies this bit in the presence of CXL.cache traffic, the results are undefined.</p>
19	RW/RO	<p>CXL.cache Isolation Link Down Enable: When set, the CXL root port shall trigger a Link Down condition when CXL.cache enters Isolation.</p> <p>This bit must be RW if the CXL.cache Isolation Link Down Supported bit is set; otherwise, it is permitted to be hardwired to 0. Software must not set this bit unless the CXL.cache Isolation Link Down Supported bit is set.</p>
24:20	RW/RO	Reserved
25	RW/RO	<p>Isolation ERR_COR Signaling Enable: When set, this bit enables the sending of an ERR_COR Message to indicate Isolation has been triggered. Default value of this bit is 0.</p> <p>This bit must be RW if the Isolation ERR_COR Signaling Supported bit is set; otherwise, it is permitted to be hardwired to 0. Software must not set this bit unless the Isolation ERR_COR Signaling Supported bit is set.</p>
26	RW/RO	<p>Isolation Interrupt Enable: When set, this bit enables the generation of an interrupt to indicate that Isolation has been triggered. Default value of this bit is 0.</p> <p>This bit must be RW if the Isolation Interrupt Supported bit is set; otherwise, it is permitted to be hardwired to 0. Software must not set this bit unless the Isolation Interrupt Supported bit is set.</p>
31:27	RW/RO	Reserved

8.2.4.24.3 CXL Timeout and Isolation Status Register (Offset 0Ch)

Bit Location	Attributes	Description
0	RW1CS/RsvdZ	CXL.mem Transaction Timeout: When set, this indicates that a CXL.mem transaction timed out.
3:1	RsvdZ	Reserved
4	RW1CS/RsvdZ	CXL.cache Transaction Timeout: When set, this indicates that a CXL.cache transaction timed out.
7:5	RsvdZ	Reserved
8	RW1CS/RsvdZ	<p>CXL.mem Isolation Status: This field indicates that Isolation mode for CXL.mem was triggered. When this bit is set, CXL.mem is in isolation and the link is forced to be down if the CXL.mem Isolation Link Down Enable bit is set.</p> <p>Software is permitted to clear this bit as part of recovery actions regardless of the state of other status bits, after which the CXL Root Port is no longer in Isolation mode for CXL.mem transactions. The link must transition through the Link Down state before software can attempt re-enumeration and device recovery.</p>
9	RW1CS/RsvdZ	CXL.mem Isolation Link Down Status: This field indicates that Isolation mode for CXL.mem was triggered because of Link Down.
11:10	RsvdZ	Reserved

Bit Location	Attributes	Description
12	RW1CS/RsvdZ	CXL.cache Isolation Status: This bit indicates that Isolation mode for CXL.cache was triggered. When this bit is set, CXL.cache is in isolation and the link is forced to be down if CXL.cache Isolation Link Down Enable is set. Software is permitted to clear this bit as part of recovery actions, after which the CXL Root Port is no longer in Isolation mode for CXL.cache transactions. The link must transition through the Link Down state before software can attempt re-enumeration and device recovery.
13	RW1CS/RsvdZ	CXL.cache Isolation Link Down Status: This bit indicates that Isolation mode for CXL.cache was triggered because of Link Down.
14	RO/RsvdZ	CXL RP Busy: When either the CXL.mem Isolation Status bit or the CXL.cache Isolation Status bit is set and this bit is set, the Root Port is busy with internal activity that must complete before software is permitted to clear the CXL.mem Isolation Status bit and the CXL.cache Isolation Status bit. If software violates this requirement, the behavior is undefined. This bit is valid only when either the CXL.mem Isolation Status bit or the CXL.cache Isolation Status bit is set; otherwise, the value of this bit is undefined. Default value of this bit is undefined.
31:15	RsvdZ	Reserved

8.2.4.25 CXL.cachemem Extended Register Capability

This capability identifies all the extended 4-KB ranges in the Component Register Space that host CXL.cachemem registers.

Offset	Register Name
00h	CXL.cachemem Extended Ranges Register

8.2.4.25.1 CXL.cachemem Extended Ranges Register (Offset 00h)

This register describes which 4-KB ranges in the Component Register Space that host CXL.cachemem Extended Range(s).

Bit Location	Attributes	Description
15:0	HwInit	<p>Extended Ranges Bitmap</p> <ul style="list-style-type: none"> Bits[0, 1, 14]: Reserved <p>More than one of the following bits may be set to 1.</p> <ul style="list-style-type: none"> Bit[2]: If set, the range 2000h-2FFFh within the Component Register space is a CXL.cachemem extended range Bit[3]: If set, the range 3000h-3FFFh within the Component Register space is a CXL.cachemem extended range Bit[4]: If set, the range 4000h-4FFFh within the Component Register space is a CXL.cachemem extended range Bit[5]: If set, the range 5000h-5FFFh within the Component Register space is a CXL.cachemem extended range Bit[6]: If set, the range 6000h-6FFFh within the Component Register space is a CXL.cachemem extended range Bit[7]: If set, the range 7000h-7FFFh within the Component Register space is a CXL.cachemem extended range Bit[8]: If set, the range 8000h-8FFFh within the Component Register space is a CXL.cachemem extended range Bit[9]: If set, the range 9000h-9FFFh within the Component Register space is a CXL.cachemem extended range Bit[10]: If set, the range A000h-AFFFh within the Component Register space is a CXL.cachemem extended range Bit[11]: If set, the range B000h-BFFFh within the Component Register space is a CXL.cachemem extended range Bit[12]: If set, the range C000h-CFFFh within the Component Register space is a CXL.cachemem extended range Bit[13]: If set, the range D000h-DFFFh within the Component Register space is a CXL.cachemem extended range Bit[15]: If set, the range F000h-FFFFh within the Component Register space is a CXL.cachemem extended range
31:16	RsvdP	Reserved

8.2.4.26 CXL BI Route Table Capability Structure

A switch uses this capability structure to manage updates to the routing of the BI messages in the upstream and downstream directions.

Revision 1 of this Capability Structure is optional for switches that do not require an explicit BI RT Commit operation. If this structure is present, it must be associated with the USP Function.

Revision 1 of this Capability Structure is not applicable to root ports, CXL devices, or DSPs.

See [Section 9.14.2](#) for details.

Offset	Register Name
00h	BI RT Capability
04h	BI RT Control
08h	BI RT Status

8.2.4.26.1 BI RT Capability (Offset 00h)

Bit Location	Attributes	Description
0	HwInit	Explicit BI RT Commit Required: If 1, indicates that the software must set the BI RT Commit bit anytime a new BI device is enabled anywhere below this port or any component below this port undergoes bus number re-assignment. If 1, the BI RT Commit bit, the BI RT Committed bit, the BI RT Commit Timeout Scale field, the BI RT Commit Timeout Base field, and BI RT Error Not Committed bit are implemented. BI RT Commit operation may be used by a component to update its internal structures or perform consistency checks.
31:1	RsvdP	Reserved

8.2.4.26.2 BI RT Control (Offset 04h)

Bit Location	Attributes	Description
0	RW/RsvdP	BI RT Commit: If Explicit BI RT Commit Required=1, software must cause this bit to transition from 0 to 1 to commit the BI-ID updates. The default value of this bit is 0. This bit must be RW if the Explicit BI RT Commit Required bit is set; otherwise, it is permitted to be hardwired to 0 and the BI Route Table update does not require an explicit commit. Software must not set this bit unless the Explicit BI RT Commit Required bit is set.
31:1	RsvdP	Reserved

8.2.4.26.3 BI RT Status (Offset 08h)

Bit Location	Attributes	Description
0	RO/RsvdP	BI RT Committed: When set to 1, it indicates that the last write that caused BI RT Commit bit to transition from 0 to 1 was successfully processed by the component. This bit is cleared when the software causes the BI RT Commit bit to transition from 1 to 0. This bit is reserved if Explicit BI RT Commit Required=0.
1	RO/RsvdP	BI RT Error Not Committed: When set to 1, it indicates that the last write that caused the BI RT Commit bit to transition from 0 to 1 was processed by the component, but resulted in an error. This bit is cleared when the software causes the BI RT Commit bit to transition from 1 to 0. This bit is reserved if Explicit BI RT Commit Required=0.
7:2	RsvdP	Reserved
11:8	HwInit/ RsvdP	BI RT Commit Timeout Scale: This field specifies the time scale associated with BI RT Commit Timeout. <ul style="list-style-type: none"> • 0000b = 1 us • 0001b = 10 us • 0010b = 100 us • 0011b = 1 ms • 0100b = 10 ms • 0101b = 100 ms • 0110b = 1 second • 0111b = 10 seconds • All other encodings are reserved This field is reserved if Explicit BI RT Commit Required=0.
15:12	HwInit / RsvdP	BI RT Commit Timeout Base: This field determines the BI RT Commit timeout. The timeout duration is calculated by multiplying the Timeout Base with the Timeout Scale. Failure to set either the BI RT Committed bit or the BI RT Error Not Committed bit within the timeout duration is treated as equivalent to commit error. In case of a timeout, the software must clear the BI RT Commit bit to 0 prior to setting it to 1 again. This field is reserved if Explicit BI RT Commit Required=0.
31:16	RsvdP	Reserved

8.2.4.27 CXL BI Decoder Capability Structure

This capability structure may be present in DSPs, root ports, or a device. The presence of this capability structure indicates that the component supports BI messages.

Offset	Register Name
00h	CXL BI Decoder Capability
04h	CXL BI Decoder Control
08h	CXL BI Decoder Status

See [Section 9.14.2](#) for details regarding the decoding of BI messages.

8.2.4.27.1 CXL BI Decoder Capability (Offset 00h)

Bit Location	Attributes	Description
0	HwInit/RsvdP	HDM-D Capable: If 1, it indicates that the Device supports HDM-D flows. If 0, it indicates that the Device does not support HDM-D flows. This bit is reserved for DSPs and Root Ports.
1	HwInit/RsvdP	Explicit BI Decoder Commit Required: If 1, indicates that the software must set BI Decoder Commit bit anytime a new BI device is enabled anywhere below this port or any component below this port undergoes bus number re-assignment. If 1, the BI Decoder Commit bit, the BI Decoder Committed bit, the BI Decoder Commit timeout Scale field, the BI Decoder Commit Timeout Base field, and BI Decoder Error Not Committed bit are implemented. BI Decoder Commit operation may be used by a component to update its internal structures or perform consistency checks. This bit is reserved for CXL devices and CXL root ports.
31:2	RsvdP	Reserved

8.2.4.27.2 CXL BI Decoder Control (Offset 04h)

See [Table 9-13](#) and [Table 9-14](#) for handling of BISnp and BIRsp messages by the DSP and RP.

Bit Location	Attributes	Description
0	RW/RsvdP	BI Forward DSP or RP: Controls whether BI messages are forwarded. The reset default is 0. This bit is reserved for CXL devices.
1	RW	BI Enable <ul style="list-style-type: none"> DSP or Root Port: If set to 1, indicates a BI-capable device is connected directly to this Downstream Port. Device: If set to 1, the device is allowed to generate BISnp requests to addresses covered by any of its local HDM decoders with BI=1 (see Section 8.2.4.20.7). The reset default is 0.
2	RW/RsvdP	BI Decoder Commit: If Explicit BI Decoder Commit Required=1, software must cause this bit to transition from 0 to 1 to commit the BI-ID assignment change to this BI Decoder instance. The default value of this field is 0. This bit must be RW if the Explicit BI Decoder Commit Required bit is set; otherwise, it is permitted to be hardwired to 0 and the BI Decoder update does not require an explicit commit. Software must not set this bit unless the Explicit BI Decoder Commit Required bit is set.
31:3	RsvdP	Reserved

8.2.4.27.3 CXL BI Decoder Status (Offset 08h)

Bit Location	Attributes	Description
0	RO	BI Decoder Committed: When set to 1, it indicates that the last write that caused the BI Decoder Commit bit to transition from 0 to 1 was successfully processed by the component. This bit is cleared when the software causes the BI Decoder Commit bit to transition from 1 to 0.
1	RO	BI Decoder Error Not Committed: When set to 1, it indicates that the last write that caused the BI Decoder Commit bit to transition from 0 to 1 was processed by the component, but resulted in an error. This bit is cleared when the software causes the BI Decoder Commit bit to transition from 1 to 0.
7:2	RsvdP	Reserved
11:8	HwInit	BI Decoder Commit Timeout Scale: This field specifies the time scale associated with BI Decoder Commit Timeout. <ul style="list-style-type: none"> • 0000b = 1 us • 0001b = 10 us • 0010b = 100 us • 0011b = 1 ms • 0100b = 10 ms • 0101b = 100 ms • 0110b = 1 second • 0111b = 10 seconds • All other encodings are reserved
15:12	HwInit	BI Decoder Commit Timeout Base: This field determines the BI Decoder Commit timeout. The timeout duration is calculated by multiplying the Timeout Base with the Timeout Scale. Failure to set either BI Decoder Committed bit or BI Decoder Error Not Committed bit within the timeout duration is treated as equivalent to commit error. In case of a timeout, the software must clear the BI Decoder Commit bit to 0 prior to setting it to 1 again.
31:16	RsvdP	Reserved

8.2.4.28 CXL Cache ID Route Table Capability Structure

The presence of this capability structure in the USP of a Switch indicates that the Switch supports CXL.cache protocol enhancements that enable multi-device scaling. Presence of this capability structure in the Host Bridge indicates that the Host supports CXL.cache protocol enhancements that enable multi-device scaling. This capability structure is mandatory if the Switch or the Host supports CXL.cache protocol enhancements that enable multi-device scaling.

The number of Cache ID Target entries is reported via the Cache ID Target Count field. For a CXL Switch, this field must be set to the maximum value permitted by the flit formats (10h for 256B flit format). The length of this capability structure is 10h + (2 * Cache ID Target Count) bytes.

See [Section 9.15.2](#) and [Section 9.15.4](#) for details.

Offset	Register Name
00h	CXL Cache ID Route Table Capability
04h	CXL Cache ID RT Control
08h	CXL Cache ID RT Status
0Ch	Reserved

Offset	Register Name
10h	CXL Cache ID Target 0
12h	CXL Cache ID Target 1
...	...

8.2.4.28.1 CXL Cache ID Route Table Capability (Offset 00h)

Bit Location	Attributes	Description
4:0	HwInit/RsvdP	Cache ID Target Count: The number of Cache ID Target entries in this capability structure. For a CXL switch, this field must be set to the maximum value amongst all the flit formats the switch supports. For example, a switch that supports the 68B flit format and the 256B flit format must set this to 10h even when the USP link is operating in 68B Flit mode. A Host Bridge may report a number that is smaller than the maximum value amongst all the flit formats the host supports.
7:5	RsvdP	Reserved
11:8	HwInit/RsvdP	HDM-D Type 2 Device Max Count: The number of Type 2 devices using HDM-D flows that this Host Bridge is capable of supporting. This field is reserved for switches.
15:12	RsvdP	Reserved
16	HwInit	Explicit Cache ID RT Commit Required: If 1, indicates that the software must set Cache ID RT Commit bit after any changes to this Cache ID Route Table for those changes to take effect. If 1, the Cache ID RT Commit bit, the Cache ID RT Committed bit, the Cache ID RT Commit timeout Scale field, the Cache ID RT Commit Timeout Base field, and Cache ID RT Error Not Committed bit are implemented. Cache ID RT Commit operation may be used by a component to update its internal structures or perform consistency checks.
31:17	RsvdP	Reserved

8.2.4.28.2 CXL Cache ID RT Control (Offset 04h)

Bit Location	Attributes	Description
0	RW/RsvdP	Cache ID RT Commit: If Explicit Cache ID RT Commit Required=1, software must cause this bit to transition from 0 to 1 to commit the contents of this Cache ID Route Table instance. The default value of this field is 0. This bit must be RW if the Cache ID RT Commit Required bit is set; otherwise, it is permitted to be hardwired to 0 and the Cache ID Route Table update does not require an explicit commit. Software must not set this bit unless the Explicit Cache ID RT Commit Required bit is set.
31:1	RsvdP	Reserved

8.2.4.28.3 CXL Cache ID RT Status (Offset 08h)

Bit Location	Attributes	Description
0	RO/RsvdP	Cache ID RT Committed: When set to 1, it indicates that the last write that caused the Cache ID RT Commit bit to transition from 0 to 1 was successfully processed by the component. This bit is cleared when the software causes the Cache ID RT Commit bit to transition from 1 to 0. This bit is reserved if Explicit Cache ID RT Commit Required=0.
1	RO/RsvdP	Cache ID RT Error Not Committed: When set to 1, it indicates that the last write that caused the Cache ID RT Commit bit to transition from 0 to 1 was processed by the component, but resulted in an error. This bit is cleared when the software causes the Cache ID RT Commit bit to transition from 1 to 0. This bit is reserved if Explicit Cache ID RT Commit Required=0.
7:2	RsvdP	Reserved
11:8	HwInit/ RsvdP	Cache ID RT Commit Timeout Scale: This field specifies the time scale associated with Cache ID RT Commit Timeout. <ul style="list-style-type: none"> • 0000b = 1 us • 0001b = 10 us • 0010b = 100 us • 0011b = 1 ms • 0100b = 10 ms • 0101b = 100 ms • 0110b = 1 second • 0111b = 10 seconds • All other encodings are reserved This field is reserved if Explicit Cache ID RT Commit Required=0.
15:12	HwInit/ RsvdP	Cache ID RT Commit Timeout Base: This field determines the Cache ID RT Commit timeout. The timeout duration is calculated by multiplying the Timeout Base with the Timeout Scale. Failure to set either the Cache ID RT Committed bit or the Cache ID RT Error Not Committed bit within the timeout value is treated as equivalent to commit error. In case of a timeout, the software must clear the Cache ID RT Commit bit to 0 prior to setting it to 1 again. This field is reserved if Explicit Cache ID RT Commit Required=0.
31:16	RsvdP	Reserved

8.2.4.28.4 CXL Cache ID Target N (Offset 10h+ 2*N)

Bit Location	Attributes	Description
0	RW	Valid <ul style="list-style-type: none"> • 0 = This Entry is invalid. • 1 = This Entry is valid. Further changes to any other fields in this register lead to undefined behavior. Software is permitted to update the Port Number field and set the Valid bit in a single register write operation. The reset default is 0.
7:1	RsvdP	Reserved
15:8	RW	Port Number: This is the Port Number to which an H2D transaction with CacheID=N maps. A switch and a Host Bridge route a CXL.cache H2D transaction with CacheID=N to the Downstream Port with this Port Number. The reset default is 00h.

8.2.4.29 CXL Cache ID Decoder Capability Structure

This capability structure may be present in DSPs and root ports. The presence of this capability structure indicates that the component supports CXL.cache protocol enhancements that enable multi-device scaling. This capability structure is mandatory if the switch or the host supports CXL.cache protocol enhancements that enable multi-device scaling. See [Section 9.15.2](#) for details.

Offset	Register Name
00h	CXL Cache ID Decoder Capability
04h	CXL Cache ID Decoder Control
08h	CXL Cache ID Decoder Status

8.2.4.29.1 CXL Cache ID Decoder Capability (Offset 00h)

Bit Location	Attributes	Description
0	HwInit	Explicit Cache ID Decoder Commit Required: If 1, indicates that the software must set the Cache ID Decoder Commit bit anytime a new CXL.cache device is enabled anywhere below this port. Also, the Cache ID Decoder Commit bit, the Cache ID Decoder Committed bit, the Cache ID Decoder Commit Timeout Scale field, the Cache ID Decoder Commit Timeout Base field, and Cache ID Decoder Error Not Committed bit are implemented. Cache ID Decoder Commit operation may be used by a component to update its internal structures or perform consistency checks.
31:1	RsvdP	Reserved

8.2.4.29.2 CXL Cache ID Decoder Control (Offset 04h)

Bit Location	Attributes	Description
0	RW	Forward Cache ID: 1 indicates that the Port forwards CXL.cache messages in both directions. The reset default is 0.
1	RW	Assign Cache ID: 1 indicates that this Downstream Port is connected directly to a CXL.cache Device and assigns a Cache ID=Local Cache ID to it. The reset default is 0.
2	RW	HDM-D Type 2 Device Present: 1 indicates that there is a Type 2 Device below this Downstream Port that is using HDM-D flows. The reset default is 0.
3	RW/RsvdP	Cache ID Decoder Commit: If Explicit Cache ID Decoder Commit Required=1, software must cause this bit to transition from 0 to 1 to commit the Cache ID assignment change to this Cache ID Decoder instance. The default value of this field is 0. This bit must be RW if the Explicit Cache ID Decoder Commit Required bit is set; otherwise, it is permitted to be hardwired to 0 and the Cache ID Decoder update does not require an explicit commit. Software must not set this bit unless the Explicit Cache ID Decoder Commit Required bit is set.
7:4	RsvdP	Reserved
11:8	RW	HDM-D Type 2 Device Cache ID: If HDM-D Type 2 Device Present=1, this field represents the Cache ID that has been assigned to the Type 2 device below this Downstream Port that is using HDM-D flows. This field may be used by the port to identify a Type 2 device that is using HDM-D flows and must not be used for assigning a Cache ID. The reset default is 0h.

Bit Location	Attributes	Description
15:12	RsvdP	Reserved
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.
31:20	RsvdP	Reserved

8.2.4.29.3 CXL Cache ID Decoder Status (Offset 08h)

Bit Location	Attributes	Description
0	RO/RsvdP	Cache ID Decoder Committed: When set to 1, it indicates that the last write that caused the Cache ID Decoder Commit bit to transition from 0 to 1 was successfully processed by the component. This bit is cleared when the software causes the Cache ID Decoder Commit bit to transition from 1 to 0. This bit is reserved if Explicit Cache ID Decoder Commit Required=0.
1	RO/RsvdP	Cache ID Decoder Error Not Committed: When set to 1, it indicates that the last write that caused the Cache ID Decoder Commit bit to transition from 0 to 1 was processed by the component, but resulted in an error. This bit is cleared when the software causes the Cache ID Decoder Commit bit to transition from 1 to 0. This bit is reserved if Explicit Cache ID Decoder Commit Required=0.
7:2	RsvdP	Reserved
11:8	HwInit/ RsvdP	Cache ID Decoder Commit Timeout Scale: This field specifies the time scale associated with Cache ID Decoder Commit Timeout. <ul style="list-style-type: none"> • 0000b = 1 us • 0001b = 10 us • 0010b = 100 us • 0011b = 1 ms • 0100b = 10 ms • 0101b = 100 ms • 0110b = 1 second • 0111b = 10 seconds • All other encodings are reserved This field is reserved if Explicit Cache ID Decoder Commit Required=0.
15:12	HwInit/ RsvdP	Cache ID Decoder Commit Timeout Base: This field determines the Cache ID Decoder Commit timeout. The timeout duration is calculated by multiplying the Timeout Base with the Timeout Scale. Failure to set either the Cache ID Decoder Committed bit or the Cache ID Decoder Error Not Committed bit within the timeout value is treated as equivalent to commit error. In case of a timeout, the software must clear the Cache ID Decoder Commit bit to 0 prior to setting it to 1 again. This field is reserved if Explicit Cache ID Decoder Commit Required=0.
31:16	RsvdP	Reserved

8.2.4.30 CXL Extended HDM Decoder Capability Structure

CXL Extended HDM Decoder Capability structure allows CXL Upstream Switch Ports to implement more HDM decoders than the limit defined in the CXL HDM Decoder Capability structure. A CXL Upstream Switch Port that is capable of routing CXL.mem traffic to more than one Downstream Switch Ports may contain one instance of this capability structure.

The layout of this capability structure is identical to the CXL HDM Decoder Capability structure and will track it (see [Section 8.2.4.20](#)).

8.2.4.31 CXL Extended Metadata Capability Register

This capability structure may be present in CXL.mem-capable devices that support 256B Flit mode. The presence of this capability structure indicates that the component is capable of storing and returning Extended Metadata.

This specification does not describe how a device with persistent memory capacity may implement Extended Metadata.

See [Section 4.3.3.2](#), [Table 3-43](#), and [Table 3-54](#) for details regarding Extended Metadata transfer over CXL.

Offset	Register Name
00h	CXL Extended Metadata Capability Register
04h	CXL Extended Metadata Control Register

8.2.4.31.1 CXL Extended Metadata Capability Register (Offset 00h)

Bit Location	Attributes	Description
6:0	RO	Max Size of Extended Metadata: Defines the maximum size of the Extended Metadata Field within the EMD trailer. Valid values are from 1 to 32. <ul style="list-style-type: none">• 1 = 1 bit of EMD• ...• 32 = 32 bits of EMD
7	RO	Reserved
8	RO	Support for Extended Metadata Error Logging: Indicates whether the component is capable of logging Extended Metadata content in the Header Log.
31:9	RO	Reserved

8.2.4.31.2 CXL Extended Metadata Control Register (Offset 04h)

The device behavior is undefined if the contents of this register are modified under the following conditions:

- CXL.mem accesses to the device are in progress
- Device is operating in 68B Flit mode

Modification to this register content shall have no impact on the Memory capacity reported via Memory_Size fields in the DVSEC CXL Range Size registers, CDAT content, Identify Memory Device output, and Get Partition Info output.

Bit Location	Attributes	Description
6:0	RWL	<p>Size of Extended Metadata: Defines the Extended Metadata Field size of a transfer. The device behavior is undefined if this register is set to a value that exceeds the Max Size of Extended Metadata reported via the CXL Extended Metadata Capability register.</p> <ul style="list-style-type: none"> • 1 = 1-bit EMD field. Corresponds to the LSB of the EMD Trailer. • ... • 31 = 31-bit EMD field, Corresponds to the 31 least significant bits of the EMD Trailer. • 32 = 32-bit EMD field <p>Locked by the CONFIG_LOCK bit (see Section 8.1.3.6).</p>
7	RO	Reserved
8	RWL/RO	<p>Enable Extended Metadata Error Logging: This bit must be RWL if the Support for Extended Metadata Error Logging bit in the CXL Extended Metadata Capability register is set; otherwise, this bit is permitted to be hardwired to 0. Software must not set this bit unless the Support for Extended Metadata Error Logging bit is set.</p> <p>If set, the device logs Extended Metadata content associated with the error, if possible, in the Header Log. See Section 8.2.4.17.1 for details.</p> <p>Locked by the CONFIG_LOCK bit (see Section 8.1.3.6).</p>
30:9	RO	Reserved
31	RWL	<p>Enable Extended Metadata Field Transfers: If set, the CXL device expects to receive and send Extended Metadata on data transfers via the trailer.</p> <p>Locked by the CONFIG_LOCK bit (see Section 8.1.3.6).</p>

8.2.5 CXL ARB/MUX Registers

The following registers are located within the 1-KB region of memory space assigned to CXL ARB/MUX. The register offsets below are listed from CXL ARB/MUX register space, starting at Offset E000h in the Component Register Range (see [Section 8.2.3](#)).

8.2.5.1 ARB/MUX PM Timeout Control Register (Offset 00h)

This register configures the ARB/MUX timeout mechanism for a PM Request ALMP that is awaiting a response, when operating in 256B Flit mode (see [Section 5.1.2.4.2.2](#)). This register is reserved in 68B Flit mode.

Bit	Attributes	Description
0	RW	<p>PMTTimeout Enable: When set, this enables the ARB/MUX timeout mechanism for PM Request ALMPs waiting for a response.</p> <p>Default value of this bit is 1.</p>
2:1	RW	<p>PMTTimeout Value: This field configures the timeout value that the ARB/MUX uses while waiting for PM Response ALMPs.</p> <ul style="list-style-type: none"> • 00b = 1 ms • All other encodings are reserved <p>Default value of this field is 00b.</p>
31:3	RsvdP	Reserved

8.2.5.2 ARB/MUX Uncorrectable Error Status Register (Offset 04h)

This register logs the timeouts that are encountered during ARB/MUX PM flows when operating in 256B Flit mode. This register is reserved in 68B Flit mode.

Bit	Attributes	Description
0	RW1CS	PM Timeout Error: For 256B Flit mode, this bit is set by the ARB/MUX to signal that a PM Request ALMP did not receive a response of ACTIVE.PMNAK or the corresponding PM Status ALMP by the time the PMTimeout counter expires. It must only be logged if PMTimeout Enable is set in the ARB/MUX PM Timeout Control register and the ARB/MUX is operating in 256B Flit mode.
1	RW1CS	L0p Timeout Error: For 256B Flit mode, this bit is set by the ARB/MUX to signal that an L0p Request ALMP did not receive a response from the remote Link partner by the time the PMTimeout counter expires. It must only be logged if PMTimeout Enable is set in the ARB/MUX PM Timeout Control register and the ARB/MUX is operating in 256B Flit mode.
31:2	RsvdZ	Reserved

8.2.5.3 ARB/MUX Uncorrectable Error Mask Register (Offset 08h)

This register controls the logging and signaling of the timeouts that are encountered during ARB/MUX PM flows when operating in 256B Flit mode. This register is reserved in 68B Flit mode.

Bit	Attributes	Description
0	RWS	PM Timeout Error Mask <ul style="list-style-type: none"> 0 = PM Timeout Error is logged as an Internal Uncorrected Error in the associated root port, similar to CXL.cachemem errors 1 = PM Timeout Error is not recorded or reported Default value of this bit is 1.
1	RWS	L0p Timeout Error Mask <ul style="list-style-type: none"> 0 = L0p Timeout Error is logged as an Internal Uncorrected Error in the associated root port, similar to CXL.cachemem errors 1 = L0p Timeout Error is not recorded or reported Default value of this bit is 1.
31:2	RsvdZ	Reserved

8.2.5.4 ARB/MUX Arbitration Control Register for CXL.io (Offset 180h)

Bit	Attributes	Description
3:0	RsvdP	Reserved
7:4	RW	CXL.io Weighted Round Robin Arbitration Weight: This is the weight assigned to CXL.io in the weighted round-robin arbitration between CXL protocols. Default value of this field is 0h.
31:8	RsvdP	Reserved

8.2.5.5 ARB/MUX Arbitration Control Register for CXL.cache and CXL.mem (Offset 1C0h)

Bit	Attributes	Description
3:0	RsvdP	Reserved
7:4	RW	CXL.cache and CXL.mem Weighted Round Robin Arbitration Weight: This is the weight assigned to CXL.cache and CXL.mem in the weighted round-robin arbitration between CXL protocols. Default value of this field is 0h.
31:8	RsvdP	Reserved

8.2.6 BAR Virtualization ACL Register Block

These registers are located at a 64-KB-aligned offset within one of the device's BARs (or BEI) as indicated by the Register Locator DVSEC (see [Section 8.1.9](#)) BAR Virtualization ACL Register Base register. They may be implemented by a CXL device that implements the DVSEC BAR Virtualization ACL Register Base register. The registers specify a standard way of communicating to the hypervisors which sections of the device BAR space are safe to assign to a Virtual Machine (VM) when the PF is directly assigned to that VM. Identifying which registers are unsafe for assigning to a VM will depend on the device micro architecture and the device security objectives, and is beyond the scope of this specification. However, examples could include registers that might affect correct operation of the device memory controller, perform device burn-in by altering its frequency or voltage, or bypass hypervisor protections for isolation of device memory assigned to one VM from the remainder of the system.

The registers consist of an array of 3 tuples of register blocks. Each tuple represents a set of contiguous registers that are safe to assign to a VM. The 3 tuples consist of the BAR number (or BAR Equivalent Index), Offset within the BAR to the start of the registers which can be safely assigned (64-KB aligned), and the size of the assigned register block (multiple of 64 KB).

Table 8-31. BAR Virtualization ACL Register Block Layout

Offset	Register Name
00h	BAR Virtualization ACL Size Register
Entry 0:	
08h	BAR Virtualization ACL Array Entry Offset Register[0]
10h	BAR Virtualization ACL Array Entry Size Register[0]
Entry 1:	
18h	BAR Virtualization ACL Array Entry Offset Register[1]
20h	BAR Virtualization ACL Array Entry Size Register[1]
Entry n:	
10h *n+ 8	...

8.2.6.1 BAR Virtualization ACL Size Register (Offset 00h)

Bit	Attributes	Description
9:0	HwInit	Number of Array Entries: Number of array elements starting at Offset 08h in this register block. Each array element consists of two 64-bit registers - Entry offset register, Entry Size register.
31:10	RsvdP	Reserved

8.2.6.1.1 BAR Virtualization ACL Array Entry Offset Register (Offset: Varies)

Bit	Attributes	Description
3:0	HwInit	Register BIR: Indicates which one of a Function's BARs, located beginning at Offset 10h in Configuration Space, or entry in the Enhanced Allocation capability with a matching BAR Equivalent Indicator (BEI), is being referenced. Defined encodings are: <ul style="list-style-type: none">• 0h = Base Address Register 10h• 1h = Base Address Register 14h• 2h = Base Address Register 18h• 3h = Base Address Register 1Ch• 4h = Base Address Register 20h• 5h = Base Address Register 24h• All other encodings are reserved
15:4	RsvdP	Reserved
63:16	HwInit	Start Offset: Offset[63:16] from the address contained by the function's BAR to the Register block within that BAR that can be safely assigned to a Virtual Machine. The starting offset is 64-KB aligned since Offset[15:0] are assumed to be 0.

8.2.6.1.2 BAR Virtualization ACL Array Entry Size Register (Offset: Varies)

Bit	Attributes	Description
15:0	RsvdP	Reserved
63:16	HwInit	Size: Indicates the Size[63:16] of the register space in bytes within the BAR that can be safely assigned to a VM. Size is a multiple of 64 KB since Size[15:0] are assumed to be 0.

8.2.7 CPMU Register Interface

Each CPMU implements a set of CPMU scoped registers and a set of Counter scoped registers. Unimplemented registers such as Counter Data and Counter Configuration registers for non-existent Counters follow the RsvdP behavior.

Table 8-32. CPMU Register Layout (Version=1) (Sheet 1 of 2)

Byte Offset	Length in Bytes	Register Name
00h	8	CPMU Capability (see Section 8.2.7.1.1)
08h	8	Reserved
10h	8	CPMU Overflow Status (see Section 8.2.7.1.2)
18h	8	CPMU Freeze (see Section 8.2.7.1.3)
20h	224	Reserved
100h	8	CPMU Event Capabilities [0] (see Section 8.2.7.1.4)

Table 8-32. CPMU Register Layout (Version=1) (Sheet 2 of 2)

Byte Offset	Length in Bytes	Register Name
108h	8	CPMU Event Capabilities [1]
...
1F8h	8	CPMU Event Capabilities [31]
200h	8	Counter Unit 0 - Counter Configuration (see Section 8.2.7.2.1)
208h	8	Counter Unit 1 - Counter Configuration
...
3F8h	8	Counter Unit 63 - Counter Configuration
400h	4	Counter Unit 0 Filter ID 0 - Filter Configuration (see Section 8.2.7.2.2)
404h	4	Counter Unit 0 Filter ID 1 - Filter Configuration
...
41Ch	4	Counter Unit 0 Filter ID 7 - Filter Configuration
420h	4	Counter Unit 1 Filter ID 0 - Filter Configuration
...
BFCh	4	Counter Unit 63 Filter ID 7 - Filter Configuration
C00h	8	Counter Unit 0 - Counter Data (see Section 8.2.7.2.3)
C08h	8	Counter Unit 1 - Counter Data
...
DF8h	8	Counter Unit 63 - Counter Data

8.2.7.1 Per CPMU Registers

Each CPMU instance is associated with a CPMU Capability register, a CPMU Overflow Status register, zero or one CPMU Freeze register, and one or more CPMU Event Capabilities registers.

8.2.7.1.1 CPMU Capability

The CPMU-wide capabilities shall be enumerated by the CPMU Capability register.

Bit	Attributes	Description
5:0	HwInit	Number of Counter Units: The number of Counter Units that are part of this CPMU, represented using 0-based encoding. <ul style="list-style-type: none"> • 00h = 1 Counter Unit • 01h = 2 Counter Units • ... • 3Fh = 64 Counter Units
7:6	RsvdP	Reserved
15:8	HwInit	Counter Width: The number of bits supported by every Counter Data register. If the value of this field is n, then each Counter Data register (see Section 8.2.7.2.3) implements n least significant bits and the maximum value it can count is $2^n - 1$.
19:16	RsvdP	Reserved
24:20	HwInit	Number of Event Capabilities Registers Supported: Indicates the number of CPMU Event Capabilities registers, represented using 0-based encoding. <ul style="list-style-type: none"> • 00h = 1 CPMU Event Capabilities register • 01h = 2 CPMU Event Capabilities registers • ... • 1Fh = 32 CPMU Event Capabilities registers

Bit	Attributes	Description
31:25	RsvdP	Reserved
39:32	HwInit	<p>Filters Supported: Bitmask that indicates the entire set of Filter IDs are supported by this CPMU. The Filter IDs available for a given Event may be restricted further. Table 13-5 describes which Filter IDs are permitted for each Event. Section 8.2.7.2.2 describes the details for each of the filters supported. The number of Filter Configuration registers per Counter Unit corresponds to the number of 1s in this field.</p>
43:40	RsvdP	Reserved
47:44	HwInit	<p>Interrupt Message Number: If Interrupt on Overflow Support=1, this field indicates which MSI/MSI-X vector is used for the interrupt message generated in association with this CPMU instance.</p> <p>For MSI, the value in this field indicates the offset between the base Message Data and the interrupt message that is generated. Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the Multiple Message Enable field in the Message Control register for MSI. For MSI-X, the value in this field indicates which MSI-X Table entry is used to generate the interrupt message. The entry shall be one of the first 16 entries even if the Function implements more than 16 entries. The value in this field shall be within the range configured by system software to the device. For a given MSI-X implementation, the entry shall remain constant.</p> <p>If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software is permitted to enable only one mechanism at a time. If MSI-X is enabled, the value in this field shall indicate the vector for MSI-X. If MSI is enabled or neither is enabled, the value in this field indicate the vector for MSI. If software enables both MSI and MSI-X at the same time, the value in this field is undefined. It is recommended that the component allocate a distinct Interrupt Message Number to each CPMU instance.</p>
48	HwInit	<p>Counters Writable while Frozen</p> <ul style="list-style-type: none"> • 0 = Indicates that the software must not write to any Counter Data register while that counter is enabled or frozen. If software writes to the Counter data register when counter is enabled or frozen, it leads to undefined behavior. Fixed Function Counter Data registers as well as Configurable Counter Data registers are always writable while disabled regardless of the state of this bit. Free-running Counter Data registers are never writable regardless of the state of this bit. • 1 = Indicates that the software is permitted to write and modify any Fixed-function Counter Data register or any Configurable Counter Data register while it is frozen.
49	HwInit	<p>Counter Freeze Support</p> <ul style="list-style-type: none"> • 0 = The CPMU does not support Counter Freeze capability. The CPMU Freeze register and the Global Freeze on Overflow bit in the Counter Configuration registers are reserved. • 1 = The CPMU supports Counter Freeze capability. The CPMU Freeze register and the Global Freeze on Overflow bit in the Counter Configuration registers are implemented.
50	HwInit	<p>Interrupt on Overflow Support</p> <ul style="list-style-type: none"> • 0 = The CPMU does not support generation of interrupts upon counter overflow. • 1 = The CPMU supports generation of interrupt upon counter overflow. Interrupt generation is controlled by the Interrupt on Overflow bit in the Counter Configuration register. The interrupt Message Number is reported in the Interrupt Message Number field.
59:51	RsvdP	Reserved
63:60	HwInit	<p>Version: Set to 1. The layout of CPMU registers for Version=1 is shown in Table 8-32.</p> <p>The version is incremented whenever the CPMU register structure is extended to add more functionality. Backward compatibility shall be maintained during this process. For all values of n, version n+1 may extend version n by replacing fields that are marked as reserved in version n or appending new registers but must not redefine the meaning of existing fields. Software that was written for a lower version may continue to operate on CPMU registers with a higher version but will not be able to take advantage of new functionality. Each field in the CPMU register structure is assumed to be introduced in version 1 of that structure unless specified otherwise in the field's definition in this specification.</p>

8.2.7.1.2 CPMU Overflow Status (Offset 10h)

The CPMU Overflow Status register indicates the overflow status associated with all the Counter Units.

When any bit in Overflow Status transitions from 0 to 1, the CPMU shall issue an MSI/MSI-X if the Interrupt on Overflow bit for the corresponding Counter Unit is 1.

Bit	Attributes	Description
C:0	RW1C	<p>Overflow Status: Bitmask with 1 bit per Counter Unit. The bit N indicates whether the Counter Unit N has encountered an overflow condition.</p> <ul style="list-style-type: none"> • 0 = The Counter Unit N has not encountered an overflow condition • 1 = The Counter Unit N has encountered an overflow condition where $0 \leq N \leq C$. <p>C equals the raw value reported by the Number of Counter Units field in the CPMU Capability register.</p>
63:C+1	RsvdP	Reserved

8.2.7.1.3 CPMU Freeze (Offset 18h)

The CPMU Freeze register indicates the freeze status associated with all the Counter Units and may be used to freeze or unfreeze individual Counter Units. This register is implemented only if the Counter Freeze Support bit in the CPMU Capability register is 1.

IMPLEMENTATION NOTE

If Counter Freeze Support as well as Counters Writable while Frozen are both 1, software may use the following flow to start counting multiple events simultaneously:

1. Freeze the counters that are involved in counting these events.
2. Initialize the Counter Data registers that correspond to these counters.
3. Unfreeze the counters.

Bit	Attributes	Description
C:0	RW/RsvdZ	<p>Freeze Control and Status: The attribute for the bits corresponding to Free-running Counter Units is RsvdZ.</p> <ul style="list-style-type: none"> • Writing 0 to bit N: The Counter Unit N is unfrozen and resumes counting unless Counter Enable=0, in which case the Counter Unit remains disabled. If the Counter Unit N is enabled but not currently frozen, it is unaffected and continues to count events. • Writing 1 to bit N: The Counter Unit N, if enabled, is frozen and stops counting further events, and retains its current value. If the Counter Unit N is already frozen when this bit is set, it remains frozen. <p>Reads return the current freeze status of each counter:</p> <ul style="list-style-type: none"> • If bit N reads as 0: The Counter Unit N is currently not frozen. The Counter Unit N may be disabled (Counter Enable=0), or may be enabled and counting events. • If bit N reads as 1: The Counter Unit N is currently frozen and not counting events. Counter Unit N remains frozen until explicitly unfrozen by software. <p>where $0 \leq N \leq C$.</p> <p>C equals the raw value reported by the Number of Counter Units field in the CPMU Capability register.</p>
63:C	RsvdZ	Reserved

8.2.7.1.4 CPMU Event Capabilities (Offset: Varies)

Each CPMU Event Capabilities register corresponds to an Event group and reports the set of Event IDs supported by the Counter Units in the CPMU for that Event group including the Fixed Counter Units. The number of CPMU Event Capabilities registers corresponds to the Number of Event Groups encoded in the CPMU Capability register.

Bit	Attributes	Description
31:0	HwInit	Supported Events: Bitmask that identifies the Event IDs within this Event Group that each Configurable Counter Unit in this CPMU is capable of counting. 0 is not a valid value.
47:32	HwInit	Event Group ID: The Group ID assigned to this Event Group by the vendor identified by the Event Vendor ID field.
63:48	HwInit	Event Vendor ID: The Vendor ID assigned by PCI-SIG to the vendor that defined this event. The values of 0000h and FFFFh are reserved per PCIe Base Specification.

8.2.7.2 Per Counter Unit Registers

8.2.7.2.1 Counter Configuration (Offset: Varies)

The Counter Configuration registers specify the set of events that are to be monitored by each Counter Unit and how they are counted. They also control interrupt generation behavior and the behavior upon overflow detection. The number of Counter Configuration registers is specified by the Number of Counter Units field of the CPMU Capability register. When a counter is enabled, changes to any field except for the Counter Enable results in undefined behavior.

Bit	Attributes	Description
1:0	HwInit	Counter Type <ul style="list-style-type: none"> 00b = This is a Free-running Counter Unit. Some of the fields in this register are RO. See individual field definitions. 01b = This is a Fixed-function Counter Unit. Some of the fields in this register are RO. See individual field definitions. 10b = This is a Configurable Counter Unit. 11b = Reserved.
7:2	RsvdP	Reserved
8	RW/RO	Counter Enable <ul style="list-style-type: none"> 0 = This Counter Unit is disabled 1 = This Counter Unit is enabled to count events If this is a free-running Counter Unit, this bit is RO and returns 1 to indicate this Counter Unit is always counting. If this bit is RW, the reset default of this bit is 0.
9	RW/RO	Interrupt on Overflow <ul style="list-style-type: none"> 0 = An Interrupt is not generated. 1 = Generate an Interrupt when this Counter Unit overflows. The interrupt Message Number is reported in the Interrupt Message Number field. This bit must be RW if the Interrupt on Overflow Support bit in the CPMU Capability register is set; otherwise, it is permitted to be hardwired to 0. Software must not set this bit unless the Interrupt on Overflow Support bit is set. If this bit is RW, the reset default of this bit is 0.
10	RW/RO	Global Freeze on Overflow <ul style="list-style-type: none"> 0 = No global freeze 1 = When this Counter Unit overflows, all Counter Units in the CPMU except the free-running Counter Units are frozen This bit must be RW if the Counter Freeze Support bit in the CPMU Capability register is set; otherwise, it is permitted to be hardwired to 0. Software must not set this bit unless the Counter Freeze Support bit is set. If this bit is RW, the reset default of this bit is 0.

Bit	Attributes	Description
11	RW/RO	Edge: When Edge is 1, the Counter Data is incremented when the Event State transitions from 0 to 1. The Event State is defined as the OR of the events enabled by the Events mask field. If this is a Free-running Counter Unit, this bit is RO. If this is a Fixed-function Counter Unit, this bit is RO. If this bit is RW, the reset default of this bit is 0.
12	RW/RO	Invert: See the definition of the Threshold field. If this is a Free-running Counter Unit, this bit is RO. If this is a Fixed-function Counter Unit, this bit is RO. If this bit is RW, the reset default of this bit is 0.
15:13	RsvdP	Reserved
23:16	RW/RO	Threshold: Some events may ordinarily increment the Counter Data by more than 1 per cycle. Queue entry count is one example of such an event. For such events, the Threshold field can be used to modify the counting behavior. If Threshold is 0, the Counter Data register is incremented by the raw event count. If Threshold is not 0 and Invert=0, Counter Data register is incremented by 1 every clock cycle where the raw event count is greater than or equal to the Threshold. If Threshold is not 0 and Invert=1, Counter Data register is incremented by 1 every clock cycle where the raw event count is less than or equal to the Threshold. For events that generate no more than one raw event per clock, Threshold shall be set to 1 by software. If this is a Free-running Counter Unit, this field is RO. If this is a Fixed-function Counter Unit, this field is RO. If this field is RW, the reset default of this field is 01h.
55:24	RW/RO	Events: Bitmask that specifies the set of events that are to be monitored by this counter, corresponding to the Event Group selected by the Event Group ID Index field. The set of supported events depends on the value of Event Group as well as the CPMU implementation. Setting unsupported bits results in undefined behavior. If this is a Free-running Counter Unit, this field is RO. More than one bit may be set. If this is a Fixed Function Counter Unit, this field is RO. More than one bit may be set. If this field is RW, the reset default of this field is 0000 0000h.
58:56	RsvdP	Reserved
63:59	RW/RO	Event Group ID Index: Identifies the CPMU Event Capabilities register that describes the Event Group ID. The value of 0 indicates the Event Vendor ID and Event Group ID that is identified by the first CPMU Event Capabilities register. If this is a Free-running Counter Unit, this field shall be RO and return the Event Group ID Index that this counter supports. The Event Group ID Index field for a Configurable Counter Unit must not be set to the Event Group ID Index reported by a Free-running Counter Unit. If this is a Fixed-function Counter Unit, this field shall be RO and return the Event Group ID Index that this counter supports. The Event Group ID Index field for a Configurable Counter Unit must not be set to the Event Group ID Index reported by a Fixed-function Counter Unit. If this field is RW, the reset default of this field is 00h.

8.2.7.2.2 Filter Configuration (Offset: Varies)

Each Counter Unit may support a set of Filter Configuration registers, one for each Filter ID. Filters constrain the counting of selected events based on one or more conditions specified in the Filter Configuration registers. For example, Counter Unit N Filter ID 0 - Filter Configuration register selects the HDM decoder(s) to monitor for events in counter N.

Each Counter Unit is associated with zero or more Filter Configuration registers, one for each supported Filter ID. The number of Filter Configuration registers per Counter Unit is derived by counting the 1s in the Filters Supported field in the CPMU Capability register.

If a filter is enabled for an event that it does not apply to, the Counter Unit behavior is undefined. When counting multiple events (multiple bits are set in Events field), Filter Configuration register must be set to all 1s. Otherwise, the Counter Unit behavior is undefined.

When multiple filters are configured for a counter, only the events that satisfy all the specified filters are counted (a logical AND of all the filter conditions).

When a counter is enabled, any changes to this register result in undefined behavior.

Bit	Attributes	Description
31:0	RW	<p>Filter Value: Specifies the filter value to be used for the Filter associated with this register.</p> <p>The reset default value is FFFF FFFFh.</p> <p>If this register is set to FFFF FFFFh, filtering is not performed for the associated Filter ID. When set to a value different from FFFF FFFFh, bits beyond the maximum value allowed for that filter are ignored.</p> <p>The encoding of this register varies based on the Filter ID. See Table 8-33 for the encoding.</p>

Table 8-33. Filter ID and Values

Filter ID	Description and Definition of the Filter Value Field
0	<p>Counts the events associated with the HDM decoder(s) specified in the Filter Value field in the Filter Configuration register. If bit n in the Filter Configuration register is set, events associated with HDM Decoder n are counted. For example, Filter Value=0Ah counts events associated with HDM Decoder 1 and HDM Decoder 3.</p>
1	<p>Counts the events associated with the combinations of the Channel, Rank and Bank Groups, and Banks that are specified in the Filter Value field. Refer to Table 8-46 for definitions of these terms.</p> <ul style="list-style-type: none"> Bits[7:0]: Bank Number, represented using 0-based encoding. The events associated with this DDR Bank are counted. If set to FFh, the CPMU shall count events associated with all Banks. Bits[15:8]: Bank Group, represented using 0-based encoding. The events associated with this DDR Bank Group are counted. If set to FFh, the CPMU shall count events associated with all Bank Groups. Bits[23:16]: Rank Number, represented using 0-based encoding. The events associated with this DDR Rank are counted. If set to FFh, the CPMU shall count events associated with all Ranks. Bits[31:24]: Channel Number, represented using 0-based encoding. The events associated with this DDR Channel are counted. If set to FFh, the CPMU shall count events associated with all Channels. <p>For example, Filter Value=0004 FF00h counts events associated with Bank 0 in all Bank Groups associated with Rank 4 in Channel 0.</p>
7:2	Reserved. Filter ID registers 2-7 for every counter are also reserved.

8.2.7.2.3 Counter Data (Offset: Varies)

The Counter Data register must be accessed as an 8-byte quantity.

Bit	Attributes	Description
N-1:0	RW	<p>Event Count: The current count value. If the Counters Writable while Frozen bit in the CPMU Capability register is 0, any changes to this register while the counter is Enabled or Frozen leads to undefined results. The value N should be chosen such that the counter takes more than one hour before the counter overflows, regardless of which Event it is counting. Once written, the counter continues to increment from the written value. A freeze operation causes the counter to stop accumulating additional events and to retain its value at the time of freeze. An unfreeze operation allows the counter to resume counting subsequent events. When the counter reaches its maximum value, it automatically wraps around upon the next event and starts counting from 0. This transition is defined as the overflow event. Other than the overflow scenario, the counter value is never decremented. N equals the raw value reported by the Counter Width field in the CPMU Capability register.</p>
63:N	RsvdP	Reserved

8.2.8 CXL Device Register Interface

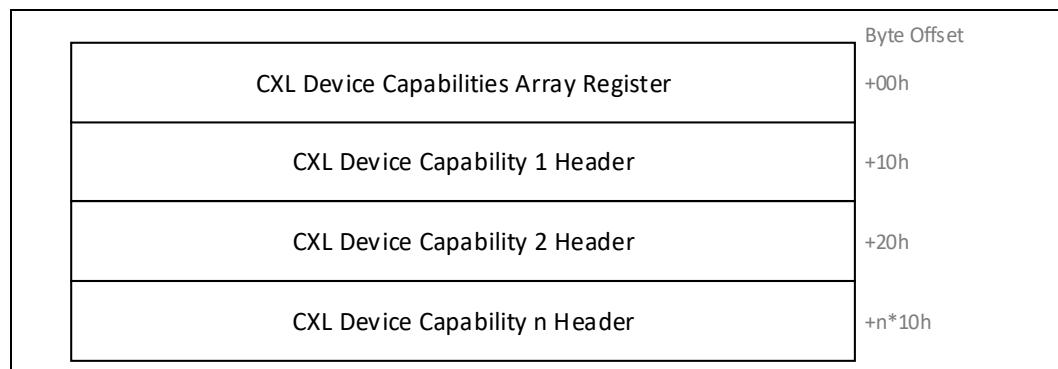
CXL device registers are mapped in memory space allocated via a standard PCIe BAR. The entry in the Register Locator DVSEC structure (see [Section 8.1.9](#)) with Register Identifier = 03h describes the BAR number and the offset within the BAR where these registers are mapped. The PCIe BAR shall be marked as prefetchable in the PCI header. At the beginning of the CXL device register block is a CXL Device Capabilities Array register that defines the size of the CXL Device Capabilities Array followed by a list of CXL Device Capability headers. Each header contains an offset to the capability-specific register structure from the start of the CXL device register block.

An MLD shall implement one instance of CXL Device registers in the MMIO space of each applicable LD.

No registers defined in [Section 8.2.8](#) are larger than 64-bit wide so that is the maximum access size allowed for these registers. If this rule is not followed, the behavior is undefined.

To illustrate how the fields fit together, the layouts in [Section 8.2.8.1](#), [Section 8.2.8.2](#), and [Figure 8-13](#) (Mailbox registers) are shown as wider than a 64-bit register. Implementations are expected to use any size accesses for this information up to 64 bits without loss of functionality – the information is designed to be accessed in chunks, each no greater than 64 bits.

Figure 8-12. CXL Device Registers



8.2.8.1 CXL Device Capabilities Array Register (Offset 00h)

Bits	Attributes	Description
15:0	RO	Capability ID: Defines the nature and format of the capability register structure. For the CXL Device Capabilities Array register, this field shall be cleared to 0000h.
23:16	RO	Version: Defines the version of the capability structure present. This field shall be set to 01h. Software shall check this version number during initialization to determine the layout of the device capabilities, treating an unknown version number as an error preventing any further access to the device by that software.
27:24	RO	Type: Identifies the type-specific capabilities in the CXL Device Capabilities Array. <ul style="list-style-type: none"> 0h = The type is inferred from the PCI Class code. If the PCI Class code is not associated with a type defined by this specification, no type-specific capabilities are present. 1h = Memory Device Capabilities (see Section 8.2.8.5). 2h = Switch Mailbox CCI Capabilities (see Section 8.2.8.6). All other encodings are reserved.
31:28	RO	Reserved
47:32	RO	Capabilities Count: The number of elements in the CXL device capabilities array, not including this header register. Each capability header element is 16 bytes in length and contiguous to previous elements.
127:48	RO	Reserved

8.2.8.2 CXL Device Capability Header Register (Offset: Varies)

Each capability in the CXL device capabilities array is described by a CXL Device Capability Header register that identifies the specific capability and points to the capability register structure in register space.

Bits	Attributes	Description
15:0	RO	Capability ID: Defines the supported capability register structure. See Section 8.2.8.2.1 for the list of capability identifiers.
23:16	RO	Version: Defines the version of the capability register structure. The version is incremented whenever the capability register structure is extended to add more functionality. Backward compatibility shall be maintained during this process. For all values of n, version n+1 may extend version n by replacing fields that are marked as reserved in version n or by appending new registers, but must not redefine the meaning of existing fields. Software that was written for a lower version may continue to operate on capability structures with a higher version but will not be able to take advantage of new functionality. If backward compatibility cannot be maintained, a new Capability ID shall be created. Each field in a capability register structure is assumed to be introduced in version 1 of that structure unless specified otherwise in the field's definition in this specification.
31:24	RO	Reserved
63:32	RO	Offset: Offset of the capability register structure from the start of the CXL device registers. The offset of performance-sensitive registers and security-sensitive registers shall be in separate 4-KB regions within the CXL device register space.
95:64	RO	Length: Size of the capability register structure in bytes.
127:96	RO	Reserved

8.2.8.2.1 CXL Device Capabilities

CXL device capability register structures are identified by a 2-byte identifier.

- Capability identifiers 0000h-3FFFh describe generic CXL device capabilities as specified in the table below.

- Capability identifiers 4000h-7FFFh describe type-specific capabilities associated with the type specified in the CXL Device Capabilities Array register (see [Section 8.2.8.1](#)).
- Capability identifiers 8000h-FFFFh describe vendor specific capabilities.

Capability identifiers 0000h-3FFFh that are not specified in this table are reserved.

Capability ID	Description	Required ¹	Version
0001h	Device Status Registers: Describes the generic CXL device status registers. Only one instance of this register structure shall exist per device.	M	02h
0002h	Primary Mailbox Registers: Describes the primary mailbox registers. Only one instance of this register structure shall exist per device.	M	01h
0003h	Secondary Mailbox Registers: Describes the secondary mailbox registers. At most one instance of this register structure shall exist per device.	O	01h

1. M = Mandatory for all devices that implement the CXL Device Register entry (Register Block Identifier=03h) in the Register Locator DVSEC (see [Section 8.1.9](#)). O = Optional.

8.2.8.3

Device Status Registers (Offset: Varies)

8.2.8.3.1

Event Status Register (Device Status Registers Capability Offset + 00h)

The Event Status register indicates which events are currently ready for host actions, such as fetching event log records. The host may choose to poll for these events by periodically reading this register, or it may choose to enable interrupts for some of these events. The only pollable/interruptible events that are not indicated in this register are mailbox command completions since each set of mailbox registers provides that information.

Unless specified otherwise in the field definitions below, each field is present in version 1 and later of this structure. The device shall report the version of this structure in the Version field of the CXL Device Capability Header register.

Bits	Attributes	Description
31:0	RO	Event Status: When set, one or more event records exist in the specified event log. Use the Get and Clear Event Records commands to retrieve and clear the event records. Once the event log has zero event records, the bit is cleared. <ul style="list-style-type: none"> • Bit[0]: Informational Event Log • Bit[1]: Warning Event Log • Bit[2]: Failure Event Log • Bit[3]: Fatal Event Log • Bit[4]: Dynamic Capacity Event Log¹ • Bits[31:5]: Reserved
63:32	RO	Reserved

1. This bit was introduced with Version=2.

8.2.8.4

Mailbox Registers (Offset: Varies)

The mailbox registers provide the ability to issue a command to the device. There are two types of mailboxes provided through the device's register interface: primary and secondary. Each mailbox represents a unique CCI instance in the device and the properties of each instance are defined in [Section 9.1.1](#). The secondary mailbox does

not support background operations. The status of a background operation issued to a device's primary mailbox can be retrieved from the Background Command Status register, as detailed in [Section 8.2.8.4.7](#).

The register interface for both types of mailboxes is the same and is described in this section. The difference between the two types of mailboxes is their intended use and commands allowed. Details on these differences are described in [Section 8.2.8.4.1](#) and [Section 8.2.8.4.2](#).

The primary and the secondary mailbox interfaces shall only be used in a single-threaded manner. It is software's responsibility to avoid simultaneous, uncoordinated access to the mailbox registers using techniques such as locking.

The mailbox command timeout is 2 seconds. Commands that require a longer execution time shall be completed asynchronously in the background. Only one command can be executed in the background at a time. The status of a background command can be retrieved from the Background Command Status register. Background commands do not continue to execute across Conventional Resets. For devices with multiple mailboxes, only the primary mailbox shall be used to issue background commands.

Devices may support sending MSI/MSI-X interrupts to indicate command status. Support for mailbox interrupts is enumerated in the Mailbox Capabilities register and enabled in the Mailbox Control register. Mailbox interrupts are only supported on the primary mailbox.

Unless specified otherwise in the field definitions for the mailbox registers below, each field is present in version 1 and later of these structures. The device shall report the version of these structures in the Version field of the CXL Device Capability Header register.

The flow for executing a command is described below. The term "Caller" represents the entity submitting the command:

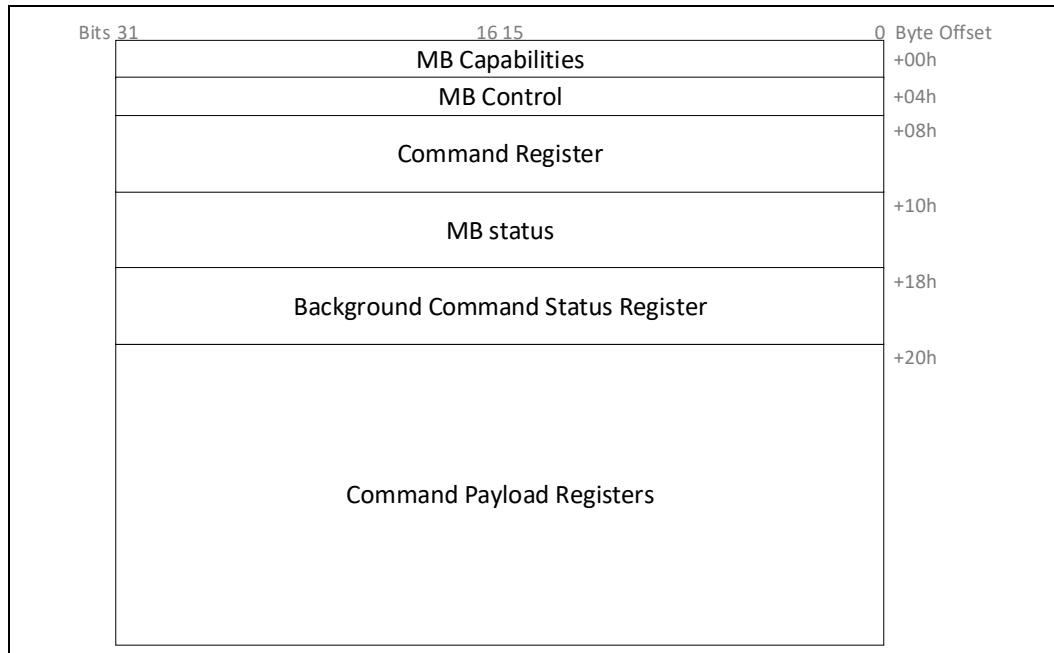
1. Caller reads MB Control register to verify doorbell is cleared.
2. Caller writes Command register.
3. Caller writes Command Payload registers if input payload is non-empty.
4. Caller writes MB Control register to set doorbell.
5. Caller either polls for doorbell to be cleared or waits for interrupt if configured.
6. Caller reads MB Status register to fetch Return code.
7. If command is successful, Caller reads Command register to get Payload Length.
8. If output payload is non-empty, Caller reads Command Payload registers.

In case of a timeout, the caller may attempt to recover the device by either issuing CXL or Conventional Reset to the device.

When a command is successfully started as a background operation, the device shall return the Background Command Started return code defined in [Section 8.2.8.4.5.1](#). While the command is executing in the background, the device should update the percentage complete in the Background Command Status register at least once per second. An ongoing background command may be aborted by issuing a Request Abort Background Operation command (see [Section 8.2.9.1.5](#)). It is strongly recommended that devices continue to accept new non-background commands while the background operation is running. The background operation shall not write to the Command Payload registers. Once the command completes in the background, the device shall update the Background Command Status register with the appropriate return code as defined in [Section 8.2.8.4.5.1](#). The caller may then retrieve the results of the background operation from the Background Command Status register.

The Mailbox registers are described in [Figure 8-13](#).

Figure 8-13. Mailbox Registers



8.2.8.4.1 Attributes of the Primary Mailbox

The primary mailbox supports all commands described in [Section 8.2.9](#). The primary mailbox also supports the optional feature to provide mailbox completion interrupts, if implemented by a device. Implementation of the primary mailbox is mandatory.

The exact details on how the primary mailbox is used may vary. The intended use is to provide the main method for submitting commands to the device, used by both pre-boot software and OS software. The platform shall coordinate the use of the primary mailbox so that only one software entity “owns” the mailbox at a given time and that the transfer of ownership happens in-between mailbox commands so that one entity cannot corrupt the mailbox state of the other. The intended practice is that the pre-boot software uses the primary mailbox until control is transferred to the OS being booted, and at that time the OS takes over sole ownership of the primary mailbox until the OS is shut down. Because the physical address of the primary mailbox can change as the result of a PCIe reconfiguration performed by the primary mailbox owner, each time the primary mailbox changes ownership, the new owner shall read the appropriate configuration registers to discover the current location of the mailbox registers, just as it does during device initialization.

8.2.8.4.2 Attributes of the Secondary Mailbox

The secondary mailbox, if implemented by a device, supports only a subset of the commands described in [Section 8.2.9](#). The Command Effects Log shall specify which commands are allowed on the secondary mailbox, and all other commands shall return the error Unsupported Mailbox or CCI. The secondary mailbox does not support mailbox completion interrupts. Implementation of the secondary mailbox is optional.

The exact details on how the secondary mailbox is used may vary. The intended use is to provide a method for submitting commands to the device by platform firmware that processes events while the OS owns the primary mailbox. By using the secondary mailbox, platform firmware does not corrupt the state of any in-progress mailbox operations on the primary mailbox.

The secondary mailbox shall return identical information as the primary mailbox for a Get Log command issued with Log Identifier=CEL. Devices shall indicate which commands are allowed on the secondary mailbox by setting the Secondary Mailbox Supported flag for the supported opcodes in the Command Effects Log. The set of commands that are supported on the secondary mailbox is implementation specific. It is recommended (but not required) that the secondary mailbox supports all commands in the Events, Logs, and Identify command sets defined in [Section 8.2.9](#).

Since the physical address of the secondary mailbox can change as the result of a PCIe reconfiguration performed by the primary mailbox owner, each time the secondary mailbox is used, the software using it shall read the appropriate configuration registers to discover the current location of the mailbox registers.

8.2.8.4.3 Mailbox Capabilities Register (Mailbox Registers Capability Offset + 00h)

Bits	Attributes	Description
4:0	RO	Payload Size: Size of the Command Payload registers in bytes, expressed as 2^n . The minimum size is 256 bytes ($n=8$) and the maximum size is 1 MB ($n=20$).
5	RO	MB Doorbell Interrupt Capable: When set, indicates the device supports signaling an MSI/MSI-X interrupt when the doorbell is cleared. Only valid for the primary mailbox. This bit shall be 0 for the secondary mailbox.
6	RO	Background Command Complete Interrupt Capable: When set, indicates the device supports signaling an MSI/MSI-X interrupt when a command completes in the background. Only valid for the primary mailbox. This bit shall be 0 for the secondary mailbox.
10:7	RO	Interrupt Message Number: This field indicates which MSI/MSI-X vector is used for the interrupt message generated in association with this mailbox instance. Only valid for the primary mailbox. This field shall be 0 for the secondary mailbox. For MSI, the value in this field indicates the offset between the base Message Data and the interrupt message that is generated. Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the Multiple Message Enable field in the Message Control register for MSI. For MSI-X, the value in this field indicates which MSI-X Table entry is used to generate the interrupt message. The entry shall be one of the first 16 entries even if the Function implements more than 16 entries. The value in this field shall be within the range configured by system software to the device. For a given MSI-X implementation, the entry shall remain constant. If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software is permitted to enable only one mechanism at a time. If MSI-X is enabled, the value in this field shall indicate the vector for MSI-X. If MSI is enabled or neither is enabled, the value in this field indicate the vector for MSI. If software enables both MSI and MSI-X at the same time, the value in this field is undefined.
18:11	RO	Mailbox Ready Time: Indicates the maximum amount of time in seconds after a Conventional or CXL reset for the Mailbox Interfaces Ready bit to become set in the Memory Device Status register. A value of 0 indicates the device does not report a mailbox ready time.
22:19	RO	Type: Identifies the type-specific commands supported by the mailbox. <ul style="list-style-type: none"> • 0h = The type is inferred from the PCI Class code. If the PCI Class code is not associated with a type defined by this specification, no type-specific commands are present. • 1h = Memory Device Commands (see Section 8.2.9.9). • 2h = FM API Commands (see Section 8.2.9.10). • All other encodings are reserved.
31:23	RsvdP	Reserved

8.2.8.4.4 Mailbox Control Register (Mailbox Registers Capability Offset + 04h)

Bits	Attributes	Description
0	RW/RO	Doorbell: When cleared, the device is ready to accept a new command. Set by the caller to notify the device that the command inputs are ready. RO when set. Cleared by the device when the command completes, or the command is started in the background.
1	RW/RO	MB Doorbell Interrupt: If doorbell interrupts are supported on this mailbox, this register is set by the caller to enable signaling an MSI/MSI-X interrupt when the doorbell is cleared. RO when the doorbell is set. Ignored if doorbell interrupts are not supported on this mailbox instance (MB Doorbell Interrupt Capable = 0 in the Mailbox Capabilities register). <ul style="list-style-type: none"> • 0 = Disabled • 1 = Enabled
2	RW/RO	Background Command Complete Interrupt: If background command complete interrupts are supported on this mailbox, this register is set by the caller to enable signaling an interrupt when the command completes in the background. Ignored if the command is not a background command. RO when the doorbell is set. Ignored if background command complete interrupts are not supported on this mailbox instance (Background Command Complete Interrupt Capable = 0 in the Mailbox Capabilities register). <ul style="list-style-type: none"> • 0 = Disabled • 1 = Enabled
31:3	RsvdP	Reserved

8.2.8.4.5 Command Register (Mailbox Registers Capability Offset + 08h)

This register shall only be used by the caller when the doorbell in the Mailbox Control register is cleared.

Bits	Attributes	Description
15:0	RW	Command Opcode: The command identifier. See Section 8.2.9 for the list of command opcodes.
36:16	RW	Payload Length: The size of the data in the command payload registers ($0 \leq \text{Payload Length} \leq \text{Payload Size}$ specified in the Mailbox Capabilities register). Expressed in bytes. Written by the caller to provide the command input payload size to the device prior to setting the doorbell. Written by the device to provide the command output payload size to the caller when the doorbell is cleared.
63:37	RsvdP	Reserved

8.2.8.4.5.1 Command Return Codes

In general, retries are not recommended for commands that return an error except when indicated in the return code definition.

Table 8-34. Command Return Codes (Sheet 1 of 2)

Value	Definition
0000h	Success: The command successfully completed.
0001h	Background Command Started: The background command successfully started. Refer to the Background Command Status register to retrieve the command result.
0002h	Invalid Input: A command input was invalid.
0003h	Unsupported: The command is not supported.
0004h	Internal Error: The command was not completed because of an internal device error.
0005h	Retry Required: The command was not completed because of a temporary error. An optional single retry may resolve the issue.

Table 8-34. Command Return Codes (Sheet 2 of 2)

Value	Definition
0006h	Busy: The device is currently busy processing a background operation. Wait until background command completes and then retry the command.
0007h	Media Disabled: The command could not be completed because it requires media access and media is disabled.
0008h	FW Transfer in Progress: Only one FW package can be transferred at a time. Complete the current FW package transfer before starting a new one.
0009h	FW Transfer Out of Order: The FW package transfer was aborted because the FW package content was transferred out of order.
000Ah	FW Verification Failed: The FW package was not saved to the device because the FW package verification failed.
000Bh	Invalid Slot: The FW slot specified is not supported or not valid for the requested operation.
000Ch	Activation Failed, FW Rolled Back: The new FW failed to activate and rolled back to the previous active FW.
000Dh	Activation Failed, Cold Reset Required: The new FW failed to activate. A cold reset is required.
000Eh	Invalid Handle: One or more Event Record Handles were invalid or specified out of order.
000Fh	Invalid Physical Address: The physical address specified is invalid.
0010h	Inject Poison Limit Reached: The device's limit on allowed poison injection has been reached. Clear injected poison requests before attempting to inject more.
0011h	Permanent Media Failure: The device could not clear poison because of a permanent issue with the media.
0012h	Aborted: The background command was aborted by the device, either on its own or as a result of a Request Abort Background Operation command.
0013h	Invalid Security State: The command is invalid in the current security state.
0014h	Incorrect Passphrase: The passphrase does not match the currently set passphrase.
0015h	Unsupported Mailbox or CCI: The command is not supported on the mailbox or CCI it was issued on.
0016h	Invalid Payload Length: The input payload length specified for the command is invalid or exceeds the component's Maximum Supported Message Size. The device is required to perform this check prior to processing any command defined in this specification.
0017h	Invalid Log: The log page is not supported or not valid.
0018h	Interrupted: The command could not be successfully completed because of an asynchronous event.
0019h	Unsupported Feature Version: The Feature version in the input payload is not supported.
001Ah	Unsupported Feature Selection Value: The selection value in the input payload is not supported.
001Bh	Feature Transfer in Progress: Only one Feature data can be transferred at a time for each Feature. Complete the current Feature data transfer before starting a new one.
001Ch	Feature Transfer Out of Order: The Feature data transfer was aborted because the Feature data content was transferred out of order.
001Dh	Resources Exhausted: The Device cannot perform the operation because resources are exhausted.
001Eh	Invalid Extent List: The Dynamic Capacity Extent List contains invalid starting DPA and length that are not contained within the Extent List the device is maintaining.
001Fh	Transfer Out of Order: The input parameters data transfer was aborted because it occurred out of order.
0020h	Request Abort Not Supported by Background Operation: The ongoing background operation does not support the Request Abort command.

8.2.8.4.6 Mailbox Status Register (Mailbox Registers Capability Offset + 10h)

Bits	Attributes	Description
0	RO	Background Operation: When set, the device is executing a command in the background. Only one command can be executing in the background; therefore, additional background commands shall be rejected with the busy return code. Refer to the Background Command Status register to retrieve the status of the background command. Only valid for the primary mailbox. This bit shall be 0 for the secondary mailbox.
31:1	RO	Reserved
47:32	RO	Return Code: The result of the command. Only valid after the doorbell is cleared. See Section 8.2.8.4.5.1 .
63:48	RO	Vendor Specific Extended Status: The vendor specific extended status information. Only valid after the doorbell is cleared.

8.2.8.4.7 Background Command Status Register (Mailbox Registers Capability Offset + 18h)

Reports information about the last command executed in the background since the last Conventional Reset. Zeroed if no background command status is available. Valid only for the primary mailbox. This register shall be zeroed on the secondary mailbox.

Bits	Attributes	Description
15:0	RO	Command Opcode: The command identifier of the ongoing command or the last command executed in the background. See Section 8.2.9 for the list of command opcodes.
22:16	RO	Percentage Complete: The percentage complete (0-100) of the background command.
31:23	RsvdP	Reserved
47:32	RO	Return Code: The result of the command run in the background. Only valid when Percentage Complete = 100 or when the Background Operation bit in the Mailbox Status register is cleared to 0. See Section 8.2.8.4.5.1 .
63:48	RO	Vendor Specific Extended Status: The vendor specific extended status of the last background command. Only valid when Percentage Complete = 100.

8.2.8.4.8 Command Payload Registers (Mailbox Registers Capability Offset + 20h)

These registers shall only be used by the caller when the doorbell in the Mailbox Control register is cleared.

Byte Offset	Length in Bytes	Attributes	Description
00h	Varies	RW	Payload: Written by the caller to provide the command input payload to the device prior to setting the doorbell. Written by the device to provide the command output payload back to the caller when the doorbell is cleared. The size of the payload data is specified in the Command register. Any data beyond the size specified in the Command register shall be ignored by the caller and the device. See Section 8.2.9 for the format of the payload data for each command.

8.2.8.5 Memory Device Capabilities

This section describes the capability registers specific to CXL memory devices that implement the PCI Header Class Code register defined in [Section 8.1.12.1](#) or advertise Memory Device Capabilities support in the CXL Device Capabilities Array register (see [Section 8.2.8.1](#)).

CXL memory device capability identifiers 4000h-7FFFh that are not specified in [Table 8-35](#) are reserved.

Table 8-35. CXL Memory Device Capabilities Identifiers

Capability ID	Description	Required ¹	Version
4000h	Memory Device Status Registers: Describes the memory device-specific status registers. Only one instance of this register structure shall exist per device.	M	01h

1. M = mandatory for all CXL memory devices.

8.2.8.5.1 Memory Device Status Registers (Offset: Varies)

The CXL memory device status registers provide information about the status of the memory device.

8.2.8.5.1.1 Memory Device Status Register (Memory Device Status Registers Capability Offset + 00h)

Unless specified otherwise in the field definitions below, each field is present in version 1 and later of this structure. The device shall report the version of this structure in the Version field of the CXL Device Capability Header register.

Bits	Attributes	Description
0	RO	Device Fatal: When set, the device has encountered a fatal error. Vendor specific device replacement or recovery is recommended.
1	RO	FW Halt: When set, the device has encountered an FW error and is not responding.
3:2	RO	Media Status: Describes the status of the device media. <ul style="list-style-type: none"> • 00b = Not Ready. Media training is incomplete. • 01b = Ready. The media trained successfully and is ready for use. • 10b = Error. The media failed to train or encountered an error. • 11b = Disabled. Access to the media is disabled. If the media is not in the Ready state, user data is not accessible.
4	RO	Mailbox Interfaces Ready: When set, the device is ready to accept commands through the mailbox register interfaces. Devices that report a nonzero mailbox ready time shall set this bit after a Conventional or CXL reset within the time reported in the Mailbox Capabilities register and it shall remain set until the next reset or the device encounters an error that prevents any mailbox communication.
7:5	RO	Reset Needed: When nonzero, indicates the least impactful reset type needed to return the device to the operational state. A cold reset is considered more impactful than a warm reset. A warm reset is considered more impactful than a hot reset, which is more impactful than a CXL reset. This field returns nonzero value if FW Halt is set, Media Status is in the Error or Disabled state, or the Mailbox Interfaces Ready does not become set. <ul style="list-style-type: none"> • 000b = Device is operational and a reset is not required • 001b = Cold Reset • 010b = Warm Reset • 011b = Hot Reset • 100b = CXL Reset (device must not report this value if it does not support CXL Reset) • All other encodings are reserved
63:8	RsvdP	Reserved

8.2.8.6 Switch Mailbox CCI Capability

This section describes the capability registers specific to Switch Mailbox CCIs that implement the PCI Header Class Code register defined in [Section 8.1.13.1](#) or advertise Switch Mailbox CCI Capabilities support in the CXL Device Capabilities Array register (see [Section 8.2.8.1](#)).

Switch Mailbox CCI capability identifiers 4000h-7FFFh that are not specified in [Table 8-36](#) are reserved.

Table 8-36. Switch Mailbox CCI Capabilities Identifiers

Capability ID	Description	Required ¹	Version
4000h	Switch Mailbox CCI Status Registers: Describes the switch mailbox CCI specific status registers. Only one instance of this register structure shall exist per device.	M	01h

1. M = Mandatory for all Switch Mailbox CCI devices.

8.2.8.6.1 Switch Mailbox CCI Status Registers (Offset: Varies)

The Switch Mailbox CCI status registers provide information about the status of the Switch's Mailbox interface.

8.2.8.6.1.1 Switch Mailbox CCI Status Register (Switch Mailbox CCI Status Registers Capability Offset + 00h)

Unless specified otherwise in the field definitions below, each field is present in version 1 and later of this structure. The device shall report the version of this structure in the Version field of the CXL Device Capability Header register.

Bits	Attributes	Description
0	RO	Mailbox Interfaces Ready: When set, the device is ready to accept commands through the mailbox register interfaces. Devices that report a nonzero mailbox ready time shall set this bit after a Conventional or CXL reset within the time reported in the Mailbox Capabilities register and it shall remain set until the next reset or the device encounters an error that prevents any mailbox communication.
63:1	RsvdP	Reserved

8.2.9 Component Command Interface

CXL component commands are identified by a 2-byte Opcode.

- Opcodes 0000h-3FFFh describe generic CXL component commands as specified in [Table 8-37](#).
- Opcodes 4000h-BFFFh describe type-specific commands associated with the type specified in the Mailbox Capabilities register (see [Section 8.2.8.4.3](#)). Two Type-Specific command sets are defined - Memory Device command (see [Table 8-126](#)) and FM API commands (see [Table 8-215](#)).
- Opcodes C000h-FFFFh describe vendor specific commands.

Opcodes also provide an implicit *major* version number, which means a command's definition will not change in an incompatible way in future revisions of this specification. Instead, if an incompatible change is required, the specification defining the change will define a new opcode for the changed command. Commands may evolve by defining new fields in the payload definitions that were originally defined as Reserved or by appending new fields, but only in a way where software written using the earlier definition will continue to work correctly, and software written to the new definition can use the 0 value or the payload size to detect components that do not support the new

field. This implicit *minor* versioning allows software to be written with the understanding that an opcode will only evolve by adding backward-compatible changes. If a component receives an input payload that is less than the size of the structure it has implemented, but greater than or equal to the Minimum Input Payload Length, then the component shall treat the unused portion of the structure as 0. For each command, any fields in the input payload that are not included in the calculation of the Minimum Input Payload Length are explicitly identified. For commands in which no fields are identified, all the fields in the input payload are to be included in the calculation of the Minimum Input Payload Length.

For several commands, the size of the payload that needs to be transferred for a unit of operation may exceed the minimum message size permitted by a CCI. Such commands are defined in such a manner that the payload may be split across multiple messages. For example, the device may implement a mailbox with Payload size of 256 Byte, but its Firmware image may be much larger. The Transfer FW (see [Section 8.2.9.3.2](#)) command is defined such that the host can split the Firmware image in 256 Byte or smaller chunks and transfer each chunk as a separate message over the Mailbox CCI.

Opcodes within the range of 0000h-3FFFh that are not specified in [Table 8-37](#) are reserved.

Generic CXL Device commands can be sent to devices and to switches as long as the devices and switches advertise support for those commands. FM API commands can be sent to switches and to certain types of devices that support them.

Some CCIs may receive commands from more than one command set. The command sets use an overlapping command opcode space, so the transport message contains a field to indicate which command set is being used. For MCTP messages, the DMTF-defined message type field indicates the command set. MCTP Base Specification (see DSP0246) describes a method by which the component advertises which command sets it supports.

Table 8-37. Generic Component Command Opcodes (Sheet 1 of 2)

Opcode				Required ¹				
				Type 1/2/3 Device		Switch	GFD	
Command Set Bits[15:8]		Command Bits[7:0]		Combined Opcode	Mailbox		Host Interface ²	FM Interface ³
00h	Information and Status	01h	Identify (Section 8.2.9.1.1)	0001h	PMB	MO	MS	MGFD
		02h	Background Operation Status (Section 8.2.9.1.2)	0002h	PMB	MO	MS	MGFD
		03h	Get Response Message Limit (Section 8.2.9.1.3)	0003h	PMB	MO	MS	P
		04h	Set Response Message Limit (Section 8.2.9.1.4)	0004h	PMB	MO	MS	P
		05h	Request Abort Background Operation (Section 8.2.9.1.5)	0005h	O	O	O	O
01h	Events	00h	Get Event Records (Section 8.2.9.2.2)	0100h	M	O	O	P
		01h	Clear Event Records (Section 8.2.9.2.3)	0101h	M	O	O	P
		02h	Get Event Interrupt Policy (Section 8.2.9.2.4)	0102h	M	P	P	P
		03h	Set Event Interrupt Policy (Section 8.2.9.2.5)	0103h	M	P	P	P
		04h	Get MCTP Event Interrupt Policy (Section 8.2.9.2.6)	0104h	PMB	O	O	P
		05h	Set MCTP Event Interrupt Policy (Section 8.2.9.2.7)	0105h	PMB	O	O	P
		06h	Event Notification (Section 8.2.9.2.8)	0106h	PMB	O	O	P
		07h	GFD Enhanced Event Notification (Section 8.2.9.2.9)	0107h	P	P	P	O
		08h	GFD to GAE Enhanced Event Notification (Section 8.2.9.2.10)	0108h	P	P	P	O
		09h	Get GAM Buffer (Section 8.2.9.2.11)	0109h	P	P	MGAE	P
02h	Firmware Update	0Ah	Set GAM Buffer (Section 8.2.9.2.12)	010Ah	P	P	MGAE	P
		00h	Get FW Info (Section 8.2.9.3.1)	0200h	O	O	O	P
		01h	Transfer FW (Section 8.2.9.3.2)	0201h	O	O	O	P
		02h	Activate FW (Section 8.2.9.3.3)	0202h	O	O	O	P

Table 8-37. Generic Component Command Opcodes (Sheet 2 of 2)

Opcode				Required ¹				
				Type 1/2/3 Device		Switch	GFD	
Command Set Bits[15:8]		Command Bits[7:0]	Combined Opcode	Mailbox	Type 3 MCTP		Host Interface ²	FM Interface ³
03h	Timestamp	00h	Get Timestamp (Section 8.2.9.4.1)	0300h	O	O	O	P O
		01h	Set Timestamp (Section 8.2.9.4.2)	0301h	O	O	O	P O
04h	Logs	00h	Get Supported Logs (Section 8.2.9.5.1)	0400h	M	O ⁴	O	MGFD MGFD
		01h	Get Log (Section 8.2.9.5.2)	0401h	M	O	O	MGFD MGFD
		02h	Get Log Capabilities (Section 8.2.9.5.3)	0402h	O	O	O	P O
		03h	Clear Log (Section 8.2.9.5.4)	0403h	O	O	O	P O
		04h	Populate Log (Section 8.2.9.5.5)	0404h	O	O	O	P O
		05h	Get Supported Logs Sub-List (Section 8.2.9.5.6)	0405h	O	O ⁴	O	P O
05h	Features	00h	Get Supported Features (Section 8.2.9.6.1)	0500h	O	O	O	P O
		01h	Get Feature (Section 8.2.9.6.2)	0501h	O	O	O	P O
		02h	Set Feature (Section 8.2.9.6.3)	0502h	O	O	O	P O
06h	Maintenance	00h	Perform Maintenance (Section 8.2.9.7.1)	0600h	O	O	O	P O
07h	PBR Components	00h	Identify PBR Component (Section 8.2.9.8.1)	0700h	O	MP	MP	P MGFD
		01h	Claim Ownership (Section 8.2.9.8.2)	0701h	O	MP	MP	P MGFD
		02h	Read CDAT (Section 8.2.9.8.3)	0702h	MP	MP	MP	MGFD MGFD

1. M = Mandatory for all devices that implement the CXL Device Register entry (Identifier=03h) in the Register Locator DVSEC (see [Section 8.1.9](#)); PMB = prohibited on the primary and secondary mailboxes; P = prohibited; MO = mandatory for components that implement an MCTP-based CCI; MP = Mandatory for PBR Components; MGFD = Mandatory for GFDs; MS = mandatory on MCTP-based CCIs for components that support the FM API MCTP message type; MGAE = Mandatory for GAEs; O = Optional.
2. "Host Interface" refers to commands issued/received via the GFD Proxying mechanism.
3. "FM Interface" refers to commands issued/received via the Fabric Crawl Out mechanism.
4. Type 3 devices implementing support for the Get Supported Logs opcode on an MCTP-based CCI shall also support the Get Supported Logs Sub-List opcode.

8.2.9.1 Information and Status Command Set

The Information and Status command set includes commands for querying component capabilities and status.

8.2.9.1.1 Identify (Opcode 0001h)

Retrieves status information about the component, including whether it is ready to process commands. A component that is not ready to process commands shall return 'Retry Required'.

Possible Command Return Codes:

- Success
- Internal Error
- Retry Required

Command Effects:

- None

Table 8-38. Identify Output Payload

Byte Offset	Length in Bytes	Description
00h	2	PCIe Vendor ID: Identifies the manufacturer of the component, as defined in PCIe Base Specification.
02h	2	PCIe Device ID: Identifier for this particular component assigned by the vendor, as defined in PCIe Base Specification.
04h	2	PCIe Subsystem Vendor ID: Identifies the manufacturer of the subsystem, as defined in PCIe Base Specification.
06h	2	PCIe Subsystem ID: Identifier for this particular subsystem assigned by the vendor, as defined in PCIe Base Specification.
08h	8	Device Serial Number: Unique identifier for this device, as defined in the Device Serial Number Extended Capability in PCIe Base Specification.
10h	1	Maximum Supported Message Size: The maximum supported size of the Message Payload (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 1024 bytes ($n=10$). 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 Message Payload 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 with Message Payload Length that exceeds the maximum size advertised in this field. The CXL specification guarantees that the size of the Identify Output Payload shall never exceed 256 bytes.
11h	1	Component Type: Indicates the type of component. <ul style="list-style-type: none"> • 00h = Switch • 03h = Type 3 Device • 04h = GFD • All other encodings are reserved

8.2.9.1.2 Background Operation Status (Opcode 0002h)

Retrieve information about outstanding Background Operations processing on the interface from which this command was received.

Possible Command Return Codes:

- Success

- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-39. Background Operation Status Output Payload

Byte Offset	Length in Bytes	Description
00h	1	Background Operation Status: Reports the status of outstanding Background Operations. <ul style="list-style-type: none"> • Bit[0]: Background Operation: Indicates whether a background operation is in progress, as defined in Section 8.2.8.4.6. • Bits[7:1]: Percentage Complete: The percentage complete (0-100) of the background command, as defined in Section 8.2.8.4.7.
01h	1	Reserved
02h	2	Command Opcode: The command identifier of the last command executed in the background. See Section 8.2.9 for the list of command opcodes.
04h	2	Return Code: The result of the command run in the background. Only valid when Percentage Complete = 100. See Section 8.2.8.4.5.1 .
06h	2	Vendor Specific Extended Status: The vendor specific extended status of the last background command. Valid only when Percentage Complete = 100.

8.2.9.1.3 Get Response Message Limit (Opcode 0003h)

Retrieves the current configured response message limit used by the component. The component shall limit the output payload size of commands with variably sized outputs such that the Message Payload does not exceed the value reported by this command.

Possible Command Return Codes:

- Success
- Internal Error
- Retry Required

Command Effects:

- None

Table 8-40. Get Response Message Limit Output Payload

Byte Offset	Length in Bytes	Description
00h	1	Response Message Limit: The configured maximum size of the Message Payload for the response message generated by the component (as defined in Figure 7-19) in bytes, expressed as $2^n \cdot 307.32$ pt. The minimum supported size is 256 bytes (n=8) and the maximum supported size is 1024 bytes (n=10).

8.2.9.1.4 Set Response Message Limit (Opcode 0004h)

Configures the response message limit used by the component. The component shall limit the output payload size of commands with variably sized outputs such that the Message Payload does not exceed the value programmed with this command.

Components shall return "Internal Error" if any errors within the component prevent it from limiting its response message size to a value lower than or equal to the requested size.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-41. Set Response Message Limit Input Payload

Byte Offset	Length in Bytes	Description
00h	1	Response Message Limit: The configured maximum size of the Message Payload for response messages generated by the component (as defined in Figure 7-19) in bytes, expressed as 2^n . The minimum supported size is 256 bytes ($n=8$) and the maximum supported size is 1024 bytes ($n=10$).

Table 8-42. Set Response Message Limit Output Payload

Byte Offset	Length in Bytes	Description
00h	1	Response Message Limit: The configured maximum size of the Message Payload for response messages generated by the component (as defined in Figure 7-19) in bytes, expressed as 2^n . The value returned is the Response Message Limit used by the component after processing this request, which may be less than the requested value. The minimum supported size is 256 bytes ($n=8$) and the maximum supported size is 1024 bytes ($n=10$). The FM shall discard any messages sent by the component with Message Payload Length that exceed the maximum size set with this field in a manner that prevents any internal receiver hardware errors.

8.2.9.1.5 Request Abort Background Operation (Opcode 0005h)

Request Abort Background Operation requests the device to abort the ongoing background operation on the same CCI. In response, the device may choose to interrupt the ongoing background operation or may choose to continue its execution until completion. If the background operation is interrupted, a Command Return code of Success shall be returned, the Percentage Complete field in the Background Command Status register shall indicate a value smaller than 100, and the Background Operation field in the Mailbox Status register shall be cleared. If the ongoing background operation does not support abort capability, the Request Abort Background Operation return code shall be set to Request Abort Not Supported by Background Operation. For this command, both the Input Payload and Output Payload are empty.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Request Abort Not Supported by Background Operation

Command Effects:

- None

8.2.9.2 Events

This section defines the standard event record format that all CXL devices shall use when reporting events to the host. Also defined are the Get Event Record and Clear Event Record commands that operate on those event records. The device shall support at least 1 event record within each event log. Devices shall return event records to the host in the temporal order the device detected the events in. The event occurring the earliest in time, in the specific event log, shall be returned first.

8.2.9.2.1 Event Records

This section describes the events reported by devices through the Get Event Records command. The device shall use the Common Event Record format when generating events for any event log.

A CXL memory device that implements the PCI Header Class Code defined in [Section 8.1.12.1](#) or advertises Memory Device Command support in the Mailbox Capabilities register (see [Section 8.2.8.4.3](#)) shall use the Memory Module Event Record format when reporting general device events and shall use either the General Media Event Record or DRAM Event Record when reporting media events.

Table 8-43. Common Event Record Format (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	10h	<p>Event Record Identifier: UUID representing the specific Event Record format. The following UUIDs are defined in this spec:</p> <ul style="list-style-type: none"> • fbc0a77-c260-417f-85a9-088b1621eba6 – General Media Event Record (see Table 8-45) • 601dcbb3-9c06-4eab-b8af-4e9bfb5c9624 – DRAM Event Record (see Table 8-46) • fe927475-dd59-4339-a586-79bab113b774 – Memory Module Event Record (see Table 8-47) • e71f3a40-2d29-4092-8a39-4d1c966c7c65 – Memory Sparing Event Record (see Table 8-48) • 77cf9271-9c02-470b-9fe4-bc7b75f2da97 – Physical Switch Event Record (see Table 7-77) • 40d26425-3396-4c4d-a5da-3d47263af425 – Virtual Switch Event Record (see Table 7-78) • 8dc44363-0c96-4710-b7bf-04bb99534c3f – MLD Port Event Record (see Table 7-79) • ca95afa7-f183-4018-8c2f-95268e101a2a – Dynamic Capacity Event Record (see Table 8-50)
10h	1	Event Record Length: Number of valid bytes that are in the event record, including all fields.

Table 8-43. Common Event Record Format (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
11h	3	<p>Event Record Flags: Multiple bits may be set.</p> <ul style="list-style-type: none"> • Bits[1:0]: Event Record Severity: The severity of the event. <ul style="list-style-type: none"> — 00b = Informational Event — 01b = Warning Event — 10b = Failure Event — 11b = Fatal Event • Bit[2]: Permanent Condition: The event reported represents a permanent condition for the device. This shall not be set when reporting Event Record Severity of Informational. • Bit[3]: Maintenance Needed: The device requires maintenance. This shall not be set when reporting Event Record Severity of Informational. • Bit[4]: Performance Degraded: The device is no longer operating at optimal performance. This shall not be set when reporting Event Record Severity of Informational. • Bit[5]: Hardware Replacement Needed: The device should immediately be replaced. This shall not be set when reporting Event Record Severity of Informational. If this bit is set and the Component Identifier field in the event record is valid, the hardware to be replaced is the hardware identified by the Component Identifier field. If this bit is set and the Component Identifier field in the event record is invalid or not part of the event record, the hardware to be replaced is the entire device. • Bit[6]: Maintenance Operation Subclass Valid Flag: If set, the Maintenance Operation Subclass is valid. This bit applies only to CXL devices. • Bits[23:7]: Reserved
14h	2	Event Record Handle: The event log unique handle for this event record. This is the value that the host shall use when requesting the device to clear events using the Clear Event Records command. This value shall be nonzero.
16h	2	Related Event Record Handle: Optional event record handle to another related event in the same event log. If there are no related events, this field shall be cleared to 0000h.
18h	8	Event Record Timestamp: The time the device recorded the event. The number of unsigned nanoseconds that have elapsed since midnight, 01-Jan-1970, UTC. If the device does not have a valid timestamp, return all 0s.
20h	1	Maintenance Operation Class: This field indicates the maintenance operation the device requests to initiate. If the device does not have any requests, this field shall be cleared to 00h. This field applies only to CXL devices. See Table 8-110 .
21h	1	Maintenance Operation Subclass: This field indicates the maintenance operation subclass that the device recommends to initiate. If the device does not have a specific recommendation for the subclass, the Maintenance Operation Subclass Valid flag shall be 0. This field applies only to CXL devices. See Table 8-110 .
22h	0Eh	Reserved
30h	50h	Event Record Data: Format depends on the Event Record Identifier

A number of Event Record formats include a Component Identifier Field. The format of this field is defined in [Table 8-44](#).

Table 8-44. Component Identifier Format

Byte Offset	Length in Bytes	Description
00h	1	ID Validity Flags: Indicators of which fields are valid. <ul style="list-style-type: none"> • Bit[0]: If set, PLDM Entity Identification Information is valid • Bit[1]: If set, PLDM Resource ID is valid • Bits[7:2]: Reserved
01h	15h	If ID Validity Flags[1:0]=00b: <ul style="list-style-type: none"> • Bytes[14:0] are reserved If ID Validity Flags[1:0]=01b: <ul style="list-style-type: none"> • Bytes[5:0] = PLDM Entity Identification Information for the lowest-level entity impacted • Bytes[14:6] are reserved If ID Validity Flags[1:0]=10b: <ul style="list-style-type: none"> • Bytes[5:0] are reserved • Bytes[9:6] = Resource ID for the lowest-level entity impacted • Bytes[14:10] are reserved. If ID Validity Flags[1:0]=11b: <ul style="list-style-type: none"> • Bytes[5:0] = PLDM Entity Identification Information for the lowest-level entity impacted • Bytes[9:6] = Resource ID for the lowest-level entity impacted • Bytes[14:10] are reserved The format of the Entity Identification Information (including the endianness) is defined in the Platform-Level Data Model (PLDM) for Platform Monitoring and Control Specification. The format of (including the endianness) is also defined in the PLDM for Platform Monitoring and Control Specification.

IMPLEMENTATION NOTE

The Component ID field points to the lowest isolated entity of a device's hierarchical model. When the Component ID format is governed by this specification, Entity Identification Information and/or Resource ID are handled as follows:

- If a CXL Type 3 Device has field-replaceable sub-components such as DIMMs and the error is associated with a specific DRAM device, the Entity Identification Information and/or Resource ID for that DRAM device is provided.
- If a specific DIMM on a CXL Type 3 Device fails training, but the device does not know which DRAM device on that DIMM is the source of the error, the Entity Identification Information and/or Resource ID for the failing DIMM is provided.
- If a CXL Type 3 Device reports an event, but it is unable to isolate to a specific sub-component or sub-FRU, then the Entity Identification Information and/or Resource ID for that CXL device is provided.

8.2.9.2.1.1 General Media Event Record

The General Media Event Record defines a general media related event. The device shall generate a General Media Event Record for each general media event occurrence. Unless specifically stated otherwise, event record content shall be generated with information on what triggered the event. For example, when a corrected error threshold is exceeded, the memory address and transaction type for the actual corrected error at the time of being exceeded shall be used.

If a General Media event occurs on multiple identifiable components related to the same cacheline, and if the Component Identifier field is valid, a General Media Event Record shall be generated for each component. Additionally, the Related Event Record Handle field of the Common Event Record (see [Table 8-43](#)) shall be used to describe such relationships.

The layout of a General Media event record is shown in [Table 8-45](#).

Table 8-45. General Media Event Record (Sheet 1 of 3)

Byte Offset	Length in Bytes	Description
00h	30h	Common Event Record: See corresponding common event record fields defined in Section 8.2.9.2.1 . The Event Record Identifier field shall be set to fbcd0a77-c260-417f-85a9-088b1621eba6 which identifies a General Media Event Record.
30h	8	Physical Address: The physical address where the memory event occurred. <ul style="list-style-type: none"> Bit[0]: Volatile: When set, indicates the DPA field is within the volatile memory range. When cleared, indicates the DPA is within the persistent memory range. Bit[1]: DPA not repairable.¹ Bits[5:2]: Reserved. Bits[7:6]: DPA[7:6]. Bits[15:8]: DPA[15:8]. ... Bits[63:56]: DPA[63:56].
38h	1	Memory Event Descriptor: Additional memory event information. Unless specified below, these shall be valid for every Memory Event Type reported. <ul style="list-style-type: none"> Bit[0]: Uncorrectable Event: When set, indicates the reported event is uncorrectable by the device. When cleared, indicates the reported event was corrected by the device. Bit[1]: Threshold Event: When set, the event is the result of a threshold on the device having been reached. When cleared, the event is not the result of a threshold limit. Bit[2]: Poison List Overflow Event: When set, the Poison List has overflowed, and this event is not in the Poison List. When cleared, the Poison List has not overflowed. Bits[7:3]: Reserved
39h	1	Memory Event Type: Identifies the type of event that occurred. The specific memory event types logged by the device will depend on the RAS mechanisms implemented in the device and is implementation dependent. <ul style="list-style-type: none"> 00h = Media ECC Error. ECC error discovered during normal operations or background operations other than scrub operations. 01h = Invalid Address. A host access was for an invalid address range. The DPA field shall contain the invalid DPA the host attempted to access. When returning this event type, the Poison List Overflow Event descriptor does not apply. 02h = Data Path Error. Internal device data path, media link, or internal device structure errors not directly related to the media. 03h = TE State Violation. The device detected a TE State access control violation during a memory request. Access control checks, device behavior, and logging are described in Section 11.5. 04h = Scrub Media ECC Error. ECC error discovered during scrub operations. 05h = Advanced Programmable Corrected Memory Error Counter Expiration. Indicates that corrected memory error counters used by an Advanced Programmable Threshold Feature have expired and reset. If this event type is set, the Threshold Event bit in the Memory Event Descriptor shall be set. 06h = CKID Violation. The device detected a CKID access control violation during a memory request. Access control checks, device behavior, and logging are described in Section 11.5. All other encodings are reserved. Note: The ECC errors event types listed above are detected by the memory media or memory controller.

Table 8-45. General Media Event Record (Sheet 2 of 3)

Byte Offset	Length in Bytes	Description
3Ah	1	<p>Transaction Type: The first device detected transaction that caused the event to occur.</p> <ul style="list-style-type: none"> • 00h = Unknown/Unreported. • 01h = Host Read. • 02h = Host Write. • 03h = Host Scan Media. • 04h = Host Inject Poison. • 05h = Media Patrol Scrub. • 06h = Media Management. • 07h = Internal Media Error Check Scrub. • 08h = Media Initialization. Media training and testing before Memory_Active or Memory_Active_Degraded is set. • All other encodings are reserved.
3Bh	2	<p>Validity Flags: Indicators of which fields are valid in the returned data.</p> <ul style="list-style-type: none"> • Bit[0]: When set, the Channel field is valid • Bit[1]: When set, the Rank field is valid • Bit[2]: When set, the Device field is valid • Bit[3]: When set, the Component Identifier field is valid • Bit[4]: When bit[3] and bit[4] are both set, the format of the Component Identifier field is governed by Table 8-44 • Bits[15:5]: Reserved
3Dh	1	<p>Channel: The channel of the memory event location. A channel is defined as an interface that can be independently accessed for a transaction. The CXL device may support one or more channels.</p>
3Eh	1	<p>Rank: The rank of the memory event location. A rank is defined as a set of memory devices on a channel that together execute a transaction. Multiple ranks may share a channel.</p>
3Fh	3	<p>Device: Bitmask that represents all devices in the rank associated with the memory event location.</p>
42h	10h	<p>Component Identifier: Device-specific component identifier for the event. This may describe a field-replaceable sub-component of the device. This field is valid when bit[3] of the Validity Flags field is set. When bit[4] of the Validity Flags field is set, the component identifier is defined by Table 8-44; otherwise, the component identifier format is device specific.</p>
52h	1	<p>Advanced Programmable Corrected Memory Error Threshold Event Flags: Field is valid if the Threshold Event bit is set in the Memory Event Descriptor field and an Advanced Programmable Threshold Feature is enabled.</p> <ul style="list-style-type: none"> • Bit[0]: The device has detected corrected memory errors in more than one memory media component before the counter has expired • Bit[1]: The programmable threshold has been exceeded • Bits[7:2]: Reserved

Table 8-45. General Media Event Record (Sheet 3 of 3)

Byte Offset	Length in Bytes	Description
53h	3	<p>Corrected Memory Error Count at Event: Accumulated corrected memory error count that is used by an Advanced Programmable Corrected Memory Error Threshold Feature at the time of the event. This field is valid if the Threshold Event bit is set in the Memory Event Descriptor field and an Advanced Programmable Corrected Memory Error Threshold Feature is enabled.</p> <p>For events in which the advanced programmable threshold is exceeded, this field value shall be equal to the threshold that is exceeded.</p> <p>For events in which the advanced programmable threshold counter has expired, this field value shall be a value greater than 0. Counter expiration events in which the corrected memory error count is 0 shall not generate a media event record.</p>
56h	1	<p>Memory Event Sub-Type: Identifies the sub-type of the Memory Event Type that occurred.</p> <p>If Memory Event Type = 02h (Data Path Error), this field is defined as follows:</p> <ul style="list-style-type: none"> • 00h = Memory Event Sub-Type Not Reported • 01h = Internal Datapath Error • 02h = Media Link Command Training Error • 03h = Media Link Control Training Error • 04h = Media Link Data Training Error • 05h = Media Link CRC Error • All other encodings are reserved <p>Other Memory Event Type values:</p> <ul style="list-style-type: none"> • 00h = Memory Event Sub-Type Not Reported • All other encodings are reserved
57h	29h	Reserved

1. If the DPA needs to be repaired but there is no resource available, then bit[1] of Physical Address field shall be set to 1 and Maintenance Needed bit in Event Record Flags shall be cleared to 0.

8.2.9.2.1.2 DRAM Event Record

The DRAM Event Record defines a DRAM-related event. The device shall generate a DRAM Event Record for each DRAM event occurrence. The layout of a DRAM Event Record is shown [Table 8-46](#). Unless specifically stated otherwise, event record content shall be generated with information on what triggered the event. For example, when a corrected error threshold is exceeded, the memory address and transaction type for the actual corrected error at the time of being exceeded shall be used.

If a DRAM event occurs on multiple components related to the same cacheline, and if the Component Identifier field is valid, a DRAM Event Record shall be generated for each component. Additionally, the Related Event Record Handle field of the Common Event Record (see [Table 8-43](#)) shall be used to describe such relationships.

Table 8-46. DRAM Event Record (Sheet 1 of 3)

Byte Offset	Length in Bytes	Description
00h	30h	Common Event Record: See corresponding common event record fields defined in Section 8.2.9.2.1 . The Event Record Identifier field shall be set to 601dcbb3-9c06-4eab-b8af-4e9bfb5c9624 which identifies a DRAM Event Record.
30h	8	<p>Physical Address: The physical address where the memory event occurred.</p> <ul style="list-style-type: none"> • Bit[0]: Volatile: When set, indicates the DPA field is within the volatile memory range. When cleared, indicates the DPA is within the persistent memory range. • Bit[1]: DPA not repairable.¹ • Bits[5:2]: Reserved. • Bits[7:6]: DPA[7:6]. • Bits[15:8]: DPA[15:8]. • ... • Bits[63:56]: DPA[63:56].
38h	1	<p>Memory Event Descriptor: Additional memory event information. Unless specified below, these shall be valid for every Memory Event Type reported.</p> <ul style="list-style-type: none"> • Bit[0]: Uncorrectable Event: When set, indicates the reported event is uncorrectable by the device. When cleared, indicates the reported event was corrected by the device. • Bit[1]: Threshold Event: When set, the event is the result of a threshold on the device having been reached. When cleared, the event is not the result of a threshold limit. • Bit[2]: Poison List Overflow Event: When set, the Poison List has overflowed, and this event is not in the Poison List. • Bits[7:3]: Reserved.
39h	1	<p>Memory Event Type: Identifies the type of event that occurred. The specific memory event types logged by the device will depend on the RAS mechanisms implemented in the device and is implementation dependent.</p> <ul style="list-style-type: none"> • 00h = Media ECC Error. ECC error discovered during normal operations or background operations other than scrub operations. • 01h = Scrub Media ECC Error. ECC error discovered during scrub operations. • 02h = Invalid Address. A host access was for an invalid address. The DPA field shall contain the invalid DPA the host attempted to access. When returning this event type, the Poison List Overflow Event descriptor does not apply. • 03h = Data Path Error. Internal device data path, media link, or internal device structure errors not directly related to the media. • 04h = TE State Violation. The device detected a Non-TEE access to TEE Exclusive memory or a TEE access to Non-TEE Exclusive memory and has returned all 1s if the Transaction Type is a Host Read or dropped the transaction if the Transaction Type is a Host Write of a partial cacheline. Memory events of this type shall utilize the Informational Event Log and report an Event Record Severity of Informational Event. This is described in Section 11.5. • 05h = Advanced Programmable Corrected Memory Error Counter Expiration. Indicates that corrected memory error counters used by an Advanced Programmable Threshold Feature have expired and reset. If this event type is set, the Threshold Event bit in the Memory Event Descriptor shall be set. • 06h = CKID Violation. The device detected a CKID value in the memory transaction that was outside the device's configured range, or the memory transaction CKID referenced an OSCKID utilizing a TEE opcode, or the memory transaction CKID referenced a TVMCKID utilizing a non-TEE opcode. Events of this type shall utilize the Informational Event Log and report an Event Record Severity of Informational Event. This is described in Section 11.5. • All other encodings are reserved. <p>Note: The ECC errors event types listed above are detected by the memory media or memory controller.</p>

Table 8-46. DRAM Event Record (Sheet 2 of 3)

Byte Offset	Length in Bytes	Description
3Ah	1	<p>Transaction Type: The first device detected transaction that caused the event to occur.</p> <ul style="list-style-type: none"> • 00h = Unknown/Unreported. • 01h = Host Read. • 02h = Host Write. • 03h = Host Scan Media. • 04h = Host Inject Poison. • 05h = Media Patrol Scrub. • 06h = Media Management. • 07h = Internal Media Error Check Scrub. • 08h = Media Initialization. Media training and testing before Memory_Active or Memory_Active_Degraded is set. • All other encodings are reserved.
3Bh	2	<p>Validity Flags: Indicators of which fields are valid in the returned data.</p> <ul style="list-style-type: none"> • Bit[0]: When set, the Channel field is valid • Bit[1]: When set, the Rank field is valid • Bit[2]: When set, the Nibble Mask field is valid • Bit[3]: When set, the Bank Group field is valid • Bit[4]: When set, the Bank field is valid • Bit[5]: When set, the Row field is valid • Bit[6]: When set, the Column field is valid • Bit[7]: When set, the Correction Mask field is valid • Bit[8]: When set, the Component Identifier field is valid • Bit[9]: When bit[8] and bit[9] are both set, the format of the Component Identifier field is governed by Table 8-44 • Bit[10]: When set, the Sub-channel field is valid • Bits[15:11]: Reserved
3Dh	1	<p>Channel: The channel of the memory event location. A channel is defined as an interface that can be independently accessed for a transaction. The CXL device may support one or more channels.</p>
3Eh	1	<p>Rank: The rank of the memory event location. A rank is defined as a set of memory devices on a channel that together execute a transaction. Multiple ranks may share a channel.</p>
3Fh	3	<p>Nibble Mask: Identifies one or more nibbles in error on the memory bus producing the event. Nibble Mask bit 0 shall be set if nibble 0 on the memory bus produced the event, etc. This field should be valid for corrected memory errors. See the implementation note following this table for an example of how this field is intended to be used.</p>
42h	1	<p>Bank Group: The bank group of the memory event location</p>
43h	1	<p>Bank: The bank number of the memory event location</p>
44h	3	<p>Row: The row number of the memory event location</p>
47h	2	<p>Column: The column number of the memory event location</p>
49h	20h	<p>Correction Mask: Identifies the bits in error within that nibble in error on the memory bus that is producing the event. The lowest nibble in error in the Nibble Mask uses Correction Mask 0, the next-lowest nibble uses Correction Mask 1, etc. Burst position 0 uses Correction Mask nibble 0, etc. Four correction masks allow for up to 4 nibbles in error. This field should be valid for corrected memory errors. See the implementation note following this table for an example of how this field is intended to be used.</p> <ul style="list-style-type: none"> • Offset 49h: Correction Mask 0 (8 bytes) • Offset 51h: Correction Mask 1 (8 bytes) • Offset 59h: Correction Mask 2 (8 bytes) • Offset 61h: Correction Mask 3 (8 bytes)

Table 8-46. DRAM Event Record (Sheet 3 of 3)

Byte Offset	Length in Bytes	Description
69h	10h	Component Identifier: Device-specific component identifier for the event. This may describe a field-replaceable sub-component of the device. This field is valid when bit[8] of the Validity Flags field is set. When bit[9] of the Validity Flags field is set, the component identifier is defined by Table 8-44 ; otherwise, the component identifier format is device specific.
79h	1	Sub-channel: The sub-channel of the memory event location.
7Ah	1	Advanced Programmable Corrected Memory Error Threshold Event Flags: This field is valid if the Threshold Event bit is set in the Memory Event Descriptor Field and the Advanced Programmable CVME Threshold Feature is enabled. <ul style="list-style-type: none"> • Bit[0]: The device has detected corrected memory errors in more than one memory media component before the counter has expired • Bit[1]: The programmable threshold has been exceeded • Bits[7:2]: Reserved
7Bh	3	CVME Count at Event: Accumulated CVME count used by the Advanced Programmable CVME Threshold Feature at the time of the event. This field is valid when the Threshold Event bit is set in the Memory Event Descriptor field and the Advanced Programmable CVME Threshold Feature is enabled. For events in which the advanced programmable threshold is exceeded, this field value shall be equal to the threshold that is exceeded. For events in which the advanced programmable threshold counter has expired, this field value shall be a value greater than 0. Counter expiration events in which the CVME count is 0 shall not generate a DRAM event record.
7Eh	1	Memory Event Sub-Type: Identifies the sub-type of Memory Event Type that occurred. If Memory Event Type = 03h (Data Path Error), this field is defined as follows: <ul style="list-style-type: none"> • 00h = Memory Event Sub-Type Not Reported • 01h = Internal Datapath Error • 02h = Media Link Command Training Error • 03h = Media Link Control Training Error • 04h = Media Link Data Training Error • 05h = Media Link CRC Error • All other encodings are reserved Other Memory Event Type values: <ul style="list-style-type: none"> • 00h = Memory Event Sub-Type Not Reported • All other encodings are reserved
7Fh	1	Reserved

1. If the DPA needs to be repaired but there is no resource available, then bit[1] of the Physical Address field shall be set to 1 and the Maintenance Needed bit in Event Record Flags shall be cleared to 0.

IMPLEMENTATION NOTE

The following example illustrates how the Nibble Mask and Correction Mask are used for a sample DDR4 and DDR5 DRAM implementation behind a CXL memory device where nibble #3 and #9 contain the location of the corrected error.

	Nibble #													
	0	1	2	3	4	...	8	9	10	...	17	18	19	
Burst Position	0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	
1	0000	0000	0000	0001	0000		0000	0010	0000		0000	0000	0000	
2	0000	0000	0000	0000	0000		0000	0000	0000		0000	0000	0000	
3	0000	0000	0000	0110	0000		0000	1111	0000		0000	0000	0000	
4	0000	0000	0000	1000	0000		0000	0101	0000		0000	0000	0000	
5	0000	0000	0000	0000	0000		0000	0000	0000		0000	0000	0000	
6	0000	0000	0000	1001	0000		0000	0011	0000		0000	0000	0000	
7	0000	0000	0000	0000	0000		0000	0000	0000		0000	0000	0000	
8	0000	0000	0000	0000	0000		0000	0000	0000		0000	0000	0000	
9	0000	0000	0000	0000	0000		0000	0000	0000		0000	0000	0000	
10	0000	0000	0000	0000	0000		0000	0000	0000		0000	0000	0000	
11	0000	0000	0000	0010	0000		0000	0000	0000		0000	0000	0000	
12	0000	0000	0000	0000	0000		0000	0000	0000		0000	0000	0000	
13	0000	0000	0000	0000	0000		0000	0000	0000		0000	0000	0000	
14	0000	0000	0000	0000	0000		0000	0000	0000		0000	0000	0000	
15	0000	0000	0000	0000	0000		0000	0000	0000		0000	0000	0000	
							DDR4		DDR5					
NibbleMask				0x000208				0x000208						
CorrMask[0]	0x0000000009086010				0x00000200009086010									<- Data[0] corresponds to 1st least-significant nibble
CorrMask[1]	0x000000000305F020					0x000000000305F020								<- Data[1] corresponds to 2nd least-significant nibble
CorrMask[2]	0x0000000000000000						0x0000000000000000							
CorrMask[3]	0x0000000000000000							0x0000000000000000						

8.2.9.2.1.3 Memory Module Event Record

The layout of a Memory Module Event Record is shown in [Table 8-47](#).

Table 8-47. Memory Module Event Record

Byte Offset	Length in Bytes	Description
00h	30h	Common Event Record: See corresponding common event record fields defined in Section 8.2.9.2.1 . The Event Record Identifier field shall be set to fe927475-dd59-4339-a586-79bab113b774 which identifies a Memory Module Event Record.
30h	1	Device Event Type: Identifies the type of event that occurred. The specific device event types logged by the device will depend on the RAS mechanisms implemented in the device and is implementation dependent. <ul style="list-style-type: none"> • 00h = Health Status Change. • 01h = Media Status Change. • 02h = Life Used Change. • 03h = Temperature Change. • 04h = Data Path Error. Internal device data path, media link or internal device structure errors not directly related to the media, training, or CRC. • 05h = LSA Error. An error occurred in the device Label Storage Area. • 06h = Unrecoverable Internal Sideband Bus Error. • 07h = Memory Media FRU Error. • 08h = Power Management Fault. • All other encodings are reserved.
31h	12h	Device Health Information: A complete copy of the device's health info at the time of the event. The format of this field is described in Table 8-133 .
43h	2	Validity Flags: Indicators of which fields within the record are valid. <ul style="list-style-type: none"> • Bit[0]: When set, the Component Identifier field is valid • Bit[1]: When bit[0] and bit[1] are both set, the format of the Component Identifier field is governed by Table 8-44 • Bits[15:2]: Reserved
45h	10h	Component Identifier: Device-specific component identifier for the event. This may describe a field-replaceable sub-component of the device. This field is valid when bit[0] of the Validity Flags field is set. When bit[1] of the Validity Flags field is set, the component identifier is defined by Table 8-44 ; otherwise, the component identifier format is device specific.
55h	1	Device Event Sub-Type: Identifies the sub-type of Device Event Type that occurred. If Device Event Type = 07h (Memory Media FRU Error), this field is defined as follows: <ul style="list-style-type: none"> • 00h = Device Event Sub-Type Not Reported. • 01h = Invalid Configuration Data. Configuration data (e.g., SPD) for memory media FRU is invalid or corrupt. • 02h = Unsupported Configuration Data. Configuration data (e.g., SPD) for memory media FRU is unsupported or unrecognized. • 03h = Unsupported Memory Media FRU. • All other encodings are reserved. Other Device Event Type values: <ul style="list-style-type: none"> • 00h = Device Event Sub-Type Not Reported. • All other encodings are reserved.
56h	2Ah	Reserved

8.2.9.2.1.4 Memory Sparing Event Record

The layout of a Memory Sparing Event Record is shown in [Table 8-48](#).

Table 8-48. Memory Sparing Event Record (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	30h	Common Event Record: See corresponding common event record fields defined in Section 8.2.9.2.1 . The Event Record Identifier field shall be set to e71f3a40-2d29-4092-8a39-4d1c966c7c65 which identifies a Memory Sparing Event Record.
30h	1	Maintenance Operation Class: This field identifies the Class of a maintenance operation. See Table 8-110 .
31h	1	Maintenance Operation Subclass: This field identifies the subclass of the maintenance operation together with the Maintenance Operation Class. See Table 8-110 .
32h	1	Flags <ul style="list-style-type: none"> • Bit[0]: Query Resources Flag: If set, the CXL device is responding to a check for resources • Bit[1]: Hard Sparing Flag: If set, the host performed a sparing of permanent type; therefore, the sparing will be irreversible and will survive any Conventional Reset • Bit[2]: Device Initiated: If set, the CXL device was the originator of the sparing action; otherwise, the host initiated the sparing action • Bits[7:3]: Reserved
33h	1	Result <ul style="list-style-type: none"> • 00h = Success • 01h = Resources Exhausted • FFh = General Failure • All other encodings are reserved
34h	2	Validity Flags: Indicators of which fields are valid within the returned data. <ul style="list-style-type: none"> • Bit[0]: When set, the Channel field is valid • Bit[1]: When set, the Rank field is valid • Bit[2]: When set, the Nibble Mask field is valid • Bit[3]: When set, the Bank Group field is valid • Bit[4]: When set, the Bank field is valid • Bit[5]: When set, the Row field is valid • Bit[6]: When set, the Column field is valid • Bit[7]: When set, the Component Identifier field is valid • Bit[8]: When bit[8] and bit[9] are both set, the format of the Component Identifier field is governed by Table 8-44 • Bit[9]: When set, the Sub-channel field is valid • Bits[15:10]: Reserved
36h	6	Reserved
3Ch	2	Spare Resource Available: Specifies the remaining quantity of sparing resources available of the same granularity and at the same location in which the sparing occurred after the sparing operation is performed. This field is valid if the Result field is cleared to 00h. <ul style="list-style-type: none"> • 00h = No more resources are available • 01h = 1 more resource is available • 02h = 2 more resources are available • 03h = 3 or more resources are available • All other encodings are reserved
3Eh	1	Channel: Spared address as reported in the DRAM event for the errors (see Table 8-46).
3Fh	1	Rank: Rank associated with the spared address as reported in the DRAM event for the errors (see Table 8-46).

Table 8-48. Memory Sparing Event Record (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
40h	3	Nibble Mask: Mask of the final nibbles that are associated with the spared bits by the command (this could be greater than the requested spare nibble). Format for spared address as reported in the DRAM event for the errors. See Table 8-46 .
43h	1	Bank Group: Bank group associated with the spared address as reported in the DRAM event for the errors (see Table 8-46).
44h	1	Bank: Bank associated with the spared address as reported in the DRAM event for the errors (see Table 8-46).
45h	3	Row: Row associated with the spared address as reported in the DRAM event for the errors (see Table 8-46).
48h	2	Column: Column associated with the spared address as reported in the DRAM event for the errors (see Table 8-46).
4Ah	10h	Component Identifier: Device-specific component identifier for the event. This may describe a field-replaceable sub-component of the device. This field is valid when bit[7] of the Validity Flags field is set. When bit[8] of the Validity Flags field is not set, the component identifier is device specific; otherwise, the component identifier format is defined by Table 8-44 . This field should be provided for all subclasses of Memory Sparing and PPR.
5Ah	1	Sub-Channel: The sub-Channel field associated with the spared address (see Table 8-46). This field is valid when bit[9] of the Validity Flags field is set. This field should be provided if the Maintenance Operation Class field is Memory Sparing.
5Bh	25h	Reserved

8.2.9.2.1.5 Vendor Specific Event Record

The layout of a vendor specific event record is shown in [Table 8-49](#).

Table 8-49. Vendor Specific Event Record

Byte Offset	Length in Bytes	Description
00h	30h	Common Event Record: See corresponding common event record fields defined in Section 8.2.9.2.1 . The Event Record Identifier field shall be set to Vendor specific UUID representing the format of this vendor specific event.
30h	50h	Vendor Specific Event Data

8.2.9.2.1.6 Dynamic Capacity Event Record

All Dynamic Capacity event records shall set the Event Record Severity field in the Common Event Record Format to Informational Event. All Dynamic Capacity related events shall be logged in the Dynamic Capacity Event Log. See [Table 8-50](#).

Table 8-50. **Dynamic Capacity Event Record (Sheet 1 of 2)**

Byte Offset	Length in Bytes	Description
00h	30h	Common Event Record: See corresponding common event record fields defined in Section 8.2.9.2.1 . The Event Record Identifier field shall be set to ca95afa7-f183-4018-8c2f-95268e101a2a which identifies a Dynamic Capacity Event Record.
30h	1	Dynamic Capacity Event Type: Identifies the type of Dynamic Capacity event that occurred. <ul style="list-style-type: none"> • 00h = Add Capacity. The device is requesting add of Dynamic Capacity for the host and the Dynamic Capacity Extent field in this structure specifies the added capacity. The host shall respond with a single Add Dynamic Capacity Response command. This event shall only be reported to the host. • 01h = Release Capacity. The device is requesting release of Dynamic Capacity from the host and the Dynamic Capacity Extent field in this structure specifies the released capacity. The host may respond with zero or more Release Dynamic Capacity requests. This event shall only be reported to the host. • 02h = Forced Capacity Release. The device has forcefully released Dynamic Capacity from the host and the Dynamic Capacity Extent field in this structure specifies the released capacity. The host may respond with zero or more Release Dynamic Capacity requests. This event shall only be reported to the host. • 03h = Region Configuration Updated. The configuration of a DC Region has been updated, as indicated by the Updated Region Index field. This event shall only be reported to the host. • 04h = Add Capacity Response. The host has responded to the Add Capacity event and the Dynamic Capacity Extent field in this structure specifies the capacity accepted by the host. This event shall only be reported to the FM. • 05h = Capacity Released. The host has released capacity and the Dynamic Capacity Extent field in this structure specifies the released capacity. This event shall only be reported to the FM. • All other encodings are reserved.
31h	1	Validity Flags <ul style="list-style-type: none"> • Bit[0]: When set, the Number of Available Extents field and the Number of Available Tags field are valid • Bits[7:1]: Reserved
32h	2	Host ID: For an LD-FAM device, the LD-ID of the host interface that triggered the generation of the event. For a GFD, the PBR ID (PID) of the Edge USP of the host that triggered the generation of the event. This field is valid only for events of type Add Capacity Response and Capacity Released.
34h	1	Updated Region Index: Index of region whose configuration was updated. This field is only valid for events of type Region Configuration Updated.
35h	1	Flags <ul style="list-style-type: none"> • Bit[0]: More: <ul style="list-style-type: none"> — 0 = This is the last (or the 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. • Bits[7:1]: Reserved
36h	2	Reserved

Table 8-50. Dynamic Capacity Event Record (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
38h	28h	Dynamic Capacity Extent: See Table 8-51.
60h	18h	Reserved
78h	4	Number of Available Extents: The remaining number of extents that the device supports, as defined in Section 9.13.3.3.
7Ch	4	Number of Available Tags: The remaining number of Tag values that the device supports, as defined in Section 9.13.3.3.

For simplicity, Dynamic Capacity Event Records with Dynamic Capacity Event Type field set to Add Capacity is referred to as Add Capacity Event Record. The term Release Capacity Event Record is used elsewhere in the document to refer to Dynamic Capacity Event Records with Dynamic Capacity Event Type set to Release Capacity. The term Forced Release Capacity Event Record is used to refer to Dynamic Capacity Event Records with Dynamic Capacity Event Type set to Forced Capacity Release.

Table 8-51. Dynamic Capacity Extent

Byte Offset	Length in Bytes	Description
00h	08h	Starting DPA: The start address of the first block of extent data. Always aligned to the Configured Region Block Size returned in Get Dynamic Capacity Configuration. The extent described by the Starting DPA and Length shall not cross a DC Region boundary. <ul style="list-style-type: none"> Bits[5:0]: Reserved Bits[63:6]: DPA[63:6]
08h	08h	Length: The number of contiguous bytes of extent data. Always a multiple of the configured Region Block Size returned in Get Dynamic Capacity Configuration. The Length shall be > 0. The extent described by the Starting DPA and Length shall not cross a DC Region boundary.
10h	10h	Tag: Optional opaque context field utilized by implementations making use of the Dynamic Capacity feature. Within sharable regions, all extents targeting the same range this field is required and all Tag values shall be identical for all hosts sharing the same extent. See Section 7.6.7.6.5 for how the Tag field is utilized while setting up sharing.
20h	02h	Shared Extent Sequence: The relative order that hosts place this extent within the virtual address space. Each instance of shared memory is presented to the sharing hosts with the same Tag value. The Shared Extent Sequence is used so that the same offsets reach the same data accessed by all hosts. For extents that describe non-shareable regions, this field shall be 0. For extents describing shareable regions this field shall be within the range of 0 to n-1 where n is the number of extents, with each value appearing only once. Extents may appear in any order and may contain gaps between them in DPA space.
22h	06h	Reserved

8.2.9.2.2 Get Event Records (Opcode 0100h)

Retrieve the next event records that may exist in the device's requested event log. This command shall retrieve as many event records from the event log that fit into the mailbox output payload. The device shall set the More Event Records indicator if there are more events to get beyond what fits in the output payload. Devices shall return event records to the host in the temporal order the device detected the events in. The event occurring the earliest in time, in the specific event log, shall be returned first.

Event records shall be cleared from the device for the device to recycle those entries for a future event. Each returned event record includes an event-log specific, nonzero, record handle that the host shall use when clearing events from the device's event log. The device shall maintain unique handles for every event that is placed in each event log.

In response to this command, the device shall return an overflow indicator when there are more events detected than could be stored in the specific event log. This indicator shall remain set until the host has consumed one or more events and called Clear Event Records to return the event handles to the device. When an event log overflows, the device shall retain all event records, in the specific event log, that occurred before the overflow event.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Media Disabled
- Busy

Command Effects:

- None

Table 8-52. Get Event Records Input Payload

Byte Offset	Length in Bytes	Description
00h	1	Event Log: The specific device event log for which to retrieve the next event records. <ul style="list-style-type: none">• 00h = Informational Event Log• 01h = Warning Event Log• 02h = Failure Event Log• 03h = Fatal Event Log• 04h = Dynamic Capacity Event Log• All other encodings are reserved

Table 8-53. Get Event Records Output Payload

Byte Offset	Length in Bytes	Description
00h	1	Flags <ul style="list-style-type: none"> Bit[0]: Overflow: This bit shall be set by the device when errors occur that the device cannot log without overwriting an existing log event. When set, the Overflow Error Count, First Overflow Event Timestamp and Last Overflow Event Timestamp fields shall be valid. This indicator shall remain set until the host has consumed one or more events and returned the event handles to the device or cleared all the events using the Clear Event Records command. Bit[1]: More Event Records: This bit shall be set by the device if there are more event records in this event log to retrieve than fit in the Get Event Records output payload. The host should continue to retrieve records using this command, until this indicator is no longer set by the device. Bits[7:2]: Reserved.
1	1	Reserved
2	2	Overflow Error Count : The number of errors detected by the device that were not logged because of an overflow of this event log. The counter does not wrap if more errors occur than can be counted. A value of 0 indicates the device does not have a count of errors that occurred after the overflow was established. This field is only valid if the overflow indicator is set.
4	8	First Overflow Event Timestamp : The time of the first event that caused the overflow of the event log to occur. This field is only valid if the overflow indicator is set. The number of unsigned nanoseconds that have elapsed since midnight, 01-Jan-1970, UTC. If the device does not have a valid timestamp, return 0.
0Ch	8	Last Overflow Event Timestamp : The time of the last event that the device detected since the overflow of the event log occurred. This field is only valid if the overflow indicator is set. The number of unsigned nanoseconds that have elapsed since midnight, 01-Jan-1970, UTC. If the device does not have a valid timestamp, return 0.
14h	2	Event Record Count : The number of event records in the Event Records list. A value of 0 indicates that there are no more event records to return.
16h	0Ah	Reserved
20h	Varies	Event Records : A list of returned Event Records.

8.2.9.2.3 Clear Event Records (Opcode 0101h)

Clear Event Records provides a mechanism for the host to clear events that it has consumed from the device's Event Log.

If the host has more events to clear than space in the input payload, it shall use multiple calls to Clear Event Records to clear them all.

If the Event Log has overflowed, the host may clear all the device's stored event logs for the requested Event Log instead of explicitly clearing each event with the unique handle. Events shall be cleared in temporal order. The device shall verify the event record handles specified in the input payload are in temporal order. If the device detects an older event record that will not be cleared when Clear Event Records is executed, the device shall return the Invalid Handle return code and shall not clear any of the specified event records.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error

- Retry Required
- Invalid Handle
- Invalid Payload Length
- Media Disabled
- Busy

Command Effects:

- Immediate Log Change

Table 8-54. Clear Event Records Input Payload

Byte Offset	Length in Bytes	Description
00h	1	Event Log: The specific device Event Log for which to clear the Event Records. <ul style="list-style-type: none"> • 00h = Informational Event Log • 01h = Warning Event Log • 02h = Failure Event Log • 03h = Fatal Event Log • 04h = Dynamic Capacity Event Log • All other encodings are reserved
01h	1	Clear Event Flags: If 0, the device shall clear the event records specified in the Event Record Handles list. <ul style="list-style-type: none"> • Bit[0]: Clear All Events: When set, the device shall clear all events that it currently has stored internally for the requested Event Log. Clear All Events is only allowed when the Event Log has overflowed; otherwise, the device shall return Invalid Input. • Bits[7:1]: Reserved.
02h	1	Number of Event Record Handles: The number of Event Record Handles in the Clear Event Records input payload. If Clear All Events is set, this shall be 0.
03h	3	Reserved
06h	Varies	Event Record Handles: A list of Event Record Handles the host has consumed and the device shall now remove from its internal Event Log store. These values are device specific and reported to the host in each event record using the Get Event Records command. All event record handles shall be nonzero value. A value of 0 shall be treated by the device as an invalid handle.

8.2.9.2.4 Get Event Interrupt Policy (Opcode 0102h)

Retrieve the settings for interrupts that are signaled for device events.

Possible Command Return Codes:

- Success
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-55. Get Event Interrupt Policy Output Payload

Byte Offset	Length in Bytes	Description
00h	1	<p>Informational Event Log Interrupt Settings: When enabled, the device shall signal an interrupt when the information event log transitions from having no entries to having one or more entries.</p> <ul style="list-style-type: none"> Bits[1:0]: Interrupt Mode: <ul style="list-style-type: none"> 00b = No interrupts 01b = MSI/MSI-X 10b = FW Interrupt (EFN VDM) 11b = Reserved Bits[3:2]: Reserved Bits[7:4]: Interrupt Message Number: See definition below
01h	1	<p>Warning Event Log Interrupt Settings: When enabled, the device shall signal an interrupt when the warning event log transitions from having no entries to having one or more entries.</p> <ul style="list-style-type: none"> Bits[1:0]: Interrupt Mode: <ul style="list-style-type: none"> 00b = No interrupts 01b = MSI/MSI-X 10b = FW Interrupt (EFN VDM) 11b = Reserved Bits[3:2]: Reserved Bits[7:4]: Interrupt Message Number: See definition below
02h	1	<p>Failure Event Log Interrupt Settings: When enabled, the device shall signal an interrupt when the failure event log transitions from having no entries to having one or more entries.</p> <ul style="list-style-type: none"> Bits[1:0]: Interrupt Mode: <ul style="list-style-type: none"> 00b = No interrupts 01b = MSI/MSI-X 10b = FW Interrupt (EFN VDM) 11b = Reserved Bits[3:2]: Reserved Bits[7:4]: Interrupt Message Number: See definition below
03h	1	<p>Fatal Event Log Interrupt Settings: When enabled, the device shall signal an interrupt when the fatal event log transitions from having no entries to having one or more entries.</p> <ul style="list-style-type: none"> Bits[1:0]: Interrupt Mode: <ul style="list-style-type: none"> 00b = No interrupts 01b = MSI/MSI-X 10b = FW Interrupt (EFN VDM) 11b = Reserved Bits[3:2]: Reserved Bits[7:4]: Interrupt Message Number: See definition below
04h	1	<p>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</p> <ul style="list-style-type: none"> Bits[1:0]: Interrupt Mode: <ul style="list-style-type: none"> 00b = No interrupts 01b = MSI/MSI-X 10b = Reserved 11b = Reserved Bits[3:2]: Reserved Bits[7:4]: Interrupt Message Number: See definition below

Interrupt Message Number is defined as follows:

If Interrupt Mode = MSI/MSI-X:

- For MSI, interrupt message number indicates the offset between the base Message Data and the interrupt message that is generated. Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the Multiple Message Enable field in the Message Control register for MSI.

- For MSI-X, interrupt message number indicates which MSI-X Table entry is used to generate the interrupt message. The entry shall be one of the first 16 entries even if the Function implements more than 16 entries. The value shall be within the range configured by system software to the device. For a given MSI-X implementation, the entry shall remain constant.
- If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software is permitted to enable only one mechanism at a time. If MSI-X is enabled, interrupt message number shall indicate the vector for MSI-X. If MSI is enabled or neither is enabled, interrupt message number shall indicate the vector for MSI. If software enables both MSI and MSI-X at the same time, interrupt message number is undefined.

If Interrupt Mode = FW Interrupt:

- Interrupt message number is the FW Interrupt vector the device uses to issue Firmware Notification via Event Firmware Notification (EFN) message.

8.2.9.2.5

Set Event Interrupt Policy (Opcode 0103h)

Change the settings for the interrupts that are signaled for device events. All event log interrupt settings shall be reset to 00b (No Interrupts) by the device on Conventional Reset.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- Immediate Policy Change

Table 8-56. Set Event Interrupt Policy Input Payload

Byte Offset	Length in Bytes	Description
00h	1	<p>Informational Event Log Interrupt Settings: Specifies the settings for the interrupt when the information event log transitions from having no entries to having one or more entries.</p> <ul style="list-style-type: none"> • Bits[1:0]: Interrupt Mode: <ul style="list-style-type: none"> — 00b = No interrupts — 01b = MSI/MSI-X — 10b = FW Interrupt (EFN VDM) — 11b = Reserved • Bits[3:2]: Reserved. • Bits[7:4]: FW Interrupt Message Number: Specifies the FW interrupt vector the device shall use to issue the firmware notification. Valid only if Interrupt Mode = FW Interrupt.
01h	1	<p>Warning Event Log Interrupt Settings: Specifies the settings for the interrupt when the warning event log transitions from having no entries to having one or more entries.</p> <ul style="list-style-type: none"> • Bits[1:0]: Interrupt Mode: <ul style="list-style-type: none"> — 00b = No interrupts — 01b = MSI/MSI-X — 10b = FW Interrupt (EFN VDM) — 11b = Reserved • Bits[3:2]: Reserved. • Bits[7:4]: FW Interrupt Message Number: Specifies the FW interrupt vector the device shall use to issue the firmware notification. Valid only if Interrupt Mode = FW Interrupt.
02h	1	<p>Failure Event Log Interrupt Settings: Specifies the settings for the interrupt when the failure event log transitions from having no entries to having one or more entries.</p> <ul style="list-style-type: none"> • Bits[1:0]: Interrupt Mode: <ul style="list-style-type: none"> — 00b = No interrupts — 01b = MSI/MSI-X — 10b = FW Interrupt (EFN VDM) — 11b = Reserved • Bits[3:2]: Reserved. • Bits[7:4]: FW Interrupt Message Number: Specifies the FW interrupt vector the device shall use to issue the firmware notification. Valid only if Interrupt Mode = FW Interrupt.
03h	1	<p>Fatal Event Log Interrupt Settings: Specifies the settings for the interrupt when the fatal event log transitions from having no entries to having one or more entries.</p> <ul style="list-style-type: none"> • Bits[1:0]: Interrupt Mode: <ul style="list-style-type: none"> — 00b = No interrupts — 01b = MSI/MSI-X — 10b = FW Interrupt (EFN VDM) — 11b = Reserved • Bits[3:2]: Reserved. • Bits[7:4]: FW Interrupt Message Number: Specifies the FW interrupt vector the device shall use to issue the firmware notification. Valid only if Interrupt Mode = FW Interrupt.
04h	1	<p>Dynamic Capacity Event Log Interrupt Settings: Specifies the settings for the interrupt when the Dynamic Capacity Event Log transitions from having no entries to having one or more entries.¹</p> <ul style="list-style-type: none"> • Bits[1:0]: Interrupt Mode: <ul style="list-style-type: none"> — 00b = No interrupts — 01b = MSI/MSI-X — 10b = Reserved — 11b = Reserved • Bits[7:2]: Reserved

1. This field is not included in the Minimum Input Payload Length.

8.2.9.2.6 Get MCTP Event Interrupt Policy (Opcode 0104h)

Retrieve the settings for interrupts that are signaled for component events over MCTP management media. When notifications are enabled for a particular log on a component, the component may issue no more than 1 Event Notification for that log type when the log contents transition from zero entries to one or more entries. The FM must be able to receive at least 1 Event Notification message from the component for each log for which interrupts have been enabled.

When notifications are enabled for Background Operation completion, the component shall issue an Event Notification upon the completion of every Background Operation. This setting shall be ignored if the component does not support Background Operations.

Possible Command Return Codes:

- Success
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-57. Payload for Get MCTP Event Interrupt Policy Output, Set MCTP Event Interrupt Policy Input, and Set MCTP Event Interrupt Policy Output

Byte Offset	Length in Bytes	Description
00h	2	Event Interrupt Settings: Bitmask indicating whether event notifications are enabled (1) or disabled (0) for a particular event. <ul style="list-style-type: none"> • Bit[0]: New uncleared Informational Event Log record(s) • Bit[1]: New uncleared Warning Event Log record(s) • Bit[2]: New uncleared Failure Event Log record(s) • Bit[3]: New uncleared Fatal Event Log record(s) • Bit[4]: New uncleared Dynamic Capacity Event Log record(s) • Bits[14:5]: Reserved • Bit[15]: Background Operation completed

8.2.9.2.7 Set MCTP Event Interrupt Policy (Opcode 0105h)

Change the settings for the interrupts that are signaled for component events. The receiver may capture the address of the requesting component in a transport-specific way. The subsequent enabled events shall be sent to that address.

Interrupts shall only be generated for events that occur after this command is received by the component with input parameters that enable logging for the corresponding log type. Components should immediately terminate the retransmission of Event Notification requests that have not been acknowledged if a request of this type has been received with input parameters that disable logging for the corresponding log type.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error
- Retry Required

- Invalid Payload Length

Command Effects:

- Immediate Policy Change

Input and output payloads for this command are documented in [Table 8-57](#).

8.2.9.2.8

Event Notification (Opcode 0106h)

This command is used by a CXL component to send notifications from its MCT-based CCIs to the agent that issued the Set MCTP Event Interrupt Policy request (see [Section 8.2.9.2.7](#)). It is only sent by CXL components. Any commands of this type received by a CXL component's MCTP-based CCI shall be silently discarded.

This command is a notification that there are new events to be read from the event logs specified in the "Event" payload field. A notification for log-based events is triggered when both of the following conditions are met: the log transitions from having no entries to having one or more entries and MCTP interrupts have been enabled with the Set MCTP Event Interrupt Policy command. Log entries are cleared with the Clear Event Logs command. After clearing log entries, the Get Event Records command should be used to confirm that all entries have been cleared from the log to ensure that the notification trigger has been re-armed. A notification for Background Operations is sent when a Background Operation completes.

The recipient of this message 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 (see [Figure 7-19](#)) until the recipient 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 recipient.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 8-58.

Event Notification Input Payload

Byte Offset	Length in Bytes	Description
00h	2	Event: Bitmask that indicates the event. <ul style="list-style-type: none"> • Bit[0]: Informational Event Log has uncleared record(s) • Bit[1]: Warning Event Log has uncleared record(s) • Bit[2]: Failure Event Log has uncleared record(s) • Bit[3]: Fatal Event Log has uncleared record(s) • Bit[4]: Dynamic Capacity Event Log has uncleared record(s) • Bits[14:5]: Reserved • Bit[15]: Background Operation completed

8.2.9.2.9

GFD Enhanced Event Notification (Opcode 0107h)

This VDM payload format is used by a GFD component to send notifications to the FM. It is only sent by GFD components.

This is a notification to the FM that there are new events to be read from the event logs specified in the “Event” payload field. A notification for log-based events is triggered when both of the following conditions are true:

- Log transitions from having no entries to having one or more entries
- MCTP interrupts have been enabled with the Set MCTP Event Interrupt Policy command

Log entries are cleared with the Clear Event Logs command. After clearing log entries, the FM should use the Get Event Records command to confirm that all entries have been cleared from the log to ensure that the notification trigger has been re-armed. A notification for Background Operations is sent when a Background Operation completes.

The FM acknowledges a notification by returning a response. The component shall resend the notification every 1 ms using the same Message Tag value in the 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.

GFDs may use the general Event Notification (see [Section 8.2.9.2.8](#)) mechanism to signal the FM, but are highly encouraged to use the Enhanced Event Notification format to deliver more-timely details about the specific event that triggered this notification.

Because GFDs do not have Event Logs per host, this enhanced event payload format shall be used by GFDs to signal individual host GAEs of specific GFD events that require host notification, such as Dynamic Capacity Change Events or asynchronous write failures.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 8-59. Enhanced Event Notification Input Payload (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	2	PBR_ID: PBR_ID of the GFD that is generating this Notification.
02h	2	Event: Bitmask that indicates the event logs that need attention. <ul style="list-style-type: none"> • Bit[0]: Informational Event Log has uncleared record(s) • Bit[1]: Warning Event Log has uncleared record(s) • Bit[2]: Failure Event Log has uncleared record(s) • Bit[3]: Fatal Event Log has uncleared record(s) • Bit[4]: Dynamic Capacity Event Log has uncleared record(s)¹ • Bits[14:5]: Reserved • Bit[15]: Background Operation completed
04h	1	GFD Notification Type: Encoded as per Table 8-60 .
05h	1	Notification Type Trigger: Details of root cause of error or event that triggered this notification. The encoding of this field is dependent on the GFD Notification Type. See Table 8-61 for these encodings.

Table 8-59. Enhanced Event Notification Input Payload (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
06h	2	RPID: Requester PID of request that was being processed for which an error was detected and/or this event notification triggered. If no specific request can be identified, the RPID shall not be marked valid. <ul style="list-style-type: none"> • Bits[11:0]: RPID of Request • Bits[15:12]: Reserved
08h	8	DPA: Device Physical Address decoded from requester's provided HPA. See Section 7.7.2.4 . DPA value is valid only if the Request Associated bit is set AND the DPA valid bit is set.
10h	8	HPA: Host Physical Address extracted from request. See Section 7.7.2.4 . HPA value is valid only if the Request Associated bit is set AND the HPA valid bit is set.
18h	2	Memory Group ID (GrpID): The GrpID value decoded from the DPA. GFDs allocate the device's DPA space to Memory Groups (see Section 7.7.2.5), which have a handle (GrpID). Every DPA that has memory allocated to back it must resolve to a defined GrpID. GrpID value is valid only if the Request Associated bit is set AND the Memory Group valid bit is set. Note: If this Enhanced Event Notification is sent to a non-FM PBR_ID, the Memory Group valid bit shall be false and this Memory Group ID field is reserved.
1Ah	1	Memory Req Opcode: The opcode extracted from the memory request. <ul style="list-style-type: none"> • Bits[3:0]: Memory Req Opcode (see Table 3-35) • Bits[7:4]: Reserved Memory Req Opcode value is valid only if the Request Associated bit is set AND the Memory Req Opcode bit is set.
1Bh	1	Reserved
1Ch	2	Flags: Various valid and condition flags. <ul style="list-style-type: none"> • Bit[0]: Reserved • Bit[1]: Request Associated: If 1, device has associated a single memory request with the triggering event, and the request's details may be valid, per remaining flags • Bit[2]: DPA Valid: If 1, DPA field is valid • Bit[3]: HPA Valid: If 1, HPA field is valid • Bit[4]: Memory Group ID Valid: If 1, GrpID value is valid • Bit[5]: Memory Req Opcode Valid: If 1, Memory Req Opcode value is valid • Bits[15:6]: Reserved
1Eh	2	Event Record Handle: The event log unique handle for the associated event record.
20h	8	Timestamp: Event Record Timestamp per Table 8-43 .

Table 8-60. GFD Notification Type Values (Sheet 1 of 2)

Notification Type	Description	Encoding
No-Op	Reserved for testing of Enhanced Event Notification delivery process. The FM shall ignore this notification.	00h
Standard Event Notification	Standard Event Notification. This command is a notification to the FM that there are new events to be read from the event logs specified in the Event payload field. No more specific Notification Type is applicable; however, some of the Enhanced Event Notification Payload fields may be valid.	01h
GFD Access Error	Memory request failed. Device could not validate Host SPID has access permissions to the specific HPA in the request, or could not decode a valid DPA from the supplied HPA.	02h
GFD Unrecognized Request	Device does not recognize the request Opcode.	03h

Table 8-60. GFD Notification Type Values (Sheet 2 of 2)

Notification Type	Description	Encoding
GFD Unexpected Packet	Device received a Response packet, a notification packet, a request type reserved for FM, etc., for which it has no associated context.	04h
GFD Invalid Input	Request fields contained invalid values or incompatible settings.	05h
GFD Persistent Flush Failure	Device's attempts to flush Persistent Memory Write Buffers have failed. This notification may be associated with the state of the Dirty Shutdown Count. See Table 8-133 .	06h
GFD Execution Failure: Non-Fatal	Device failed to successfully execute a Memory Request, device state is Non-Fatal and device is capable of processing subsequent Memory Requests.	07h
GFD Execution Failure: Fatal	Device failed to successfully execute a Memory Request, device state is Fatal and device is not capable of processing subsequent Memory Requests.	08h
GFD Health Alerts	Device Health Events have been logged and one or more notification thresholds have been reached.	09h
GFD Extent List Change	Dynamic Capacity associated with this host may have changed. Host must reread the Extent List of this GFD.	0Ah
Reserved	Reserved	0Bh - FFh

Table 8-61. GFD Notification Type Triggers (Sheet 1 of 2)

Notification Type	Notification Type Trigger Values
No-Op	<ul style="list-style-type: none"> Bits[7:0]: Reserved
Standard Event Notification	<ul style="list-style-type: none"> Bits[7:0]: Reserved
GFD Access Error	<ul style="list-style-type: none"> Bit[0]: HPA-to-DPA mapping is non-existent, HPA was not claimed by any of the host's decoders Bit[1]: DPA-to-HPA snoop mapping failed, Back-Invalidate HPA unresolved Bit[2]: DPA to Memory Group ID (GrpID) failed, GrpID is undefined Bit[3]: SPID access permission denied, GrpAccVec[GrpID] is not set in SAT[SPID] Bits[7:4]: Reserved
GFD Unrecognized Request	<ul style="list-style-type: none"> Bit[0]: Not a valid memory request opcode Bit[1]: Not a supported request type Bits[7:2]: Reserved
GFD Unexpected Packet	<ul style="list-style-type: none"> Bits[7:0]: Reserved
GFD Invalid Input	<ul style="list-style-type: none"> Bits[7:0]: Reserved
GFD Persistent Flush Failure	<ul style="list-style-type: none"> Bit[0]: Dirty Shutdown Counter has incremented (detected after reset) Bit[1]: Internal error detected during the flush (detected before reset) Bit[2]: Media error detected during flush (detected before reset) Bits[7:3]: Reserved

Table 8-61. GFD Notification Type Triggers (Sheet 2 of 2)

Notification Type	Notification Type Trigger Values
GFD Execution Failure: Non-Fatal	<ul style="list-style-type: none"> Bit[0]: DPA-to-HPA snoop mapping failed, Back-Invalidate HPA unresolved. Error occurred using a sharer's PBR_ID, not the requester's. Bit[1]: Back-Invalidate request received unsuccessful response Bit[2]: Back-Invalidate request received no response Bit[3]: Transient internal uncorrectable ECC error detected Bits[7:4]: Reserved
GFD Execution Failure: Fatal	<ul style="list-style-type: none"> Bit[0]: Persistent internal uncorrectable ECC error detected Bit[1]: Persistent media access error Bits[7:2]: Reserved
GFD Health Alerts	Health Alerts: Indicators of which alert thresholds have been exceeded. <ul style="list-style-type: none"> Bit[0]: When set, the Life Used Programmable Warning Threshold has tripped Bit[1]: When set, the Device Over-Temperature Programmable Warning Threshold has tripped Bit[2]: When set, the Device Under-Temperature Programmable Warning Threshold has tripped Bit[3]: When set, the Corrected Volatile Memory Error Programmable Warning Threshold has tripped Bit[4]: When set, the Corrected Persistent Memory Error Programmable Warning Threshold has tripped Bits[7:5]: Reserved

8.2.9.2.10 GFD to GAE Enhanced Event Notification (Opcode 0108h)

This command is used by a GFD component to send asynchronous event notifications to a host GAE. It is sent only by GFD components.

Because a GFD does not keep logs on a per-host basis, a standard Event Notification (see [Section 8.2.9.2.8](#)) will not suffice when signaling a host of an asynchronous error or event that has occurred as a result of processing a host's request. The host does not have access to the global event logs of the GFD.

This enhanced event payload format shall be used by GFDs to signal individual host GAEs of specific GFD events that require host attention, mainly to report asynchronous write failures. The enhanced event notification payload contains relevant fields from the host's failed memory request.

This Event Notification is transported to the target host GAE using the GAM VDM, as described in [Section 3.1.11.7](#). Its payload is logged in the GAM Buffer, a circular buffer in host memory.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error

Command Effects:

- None

Table 8-62. Enhanced Event Notification Input Payload

Byte Offset	Length in Bytes	Description
00h	2	PBR_ID: PBR_ID of the GFD that is generating this Notification.
02h	1	GFD Notification Type: Encoded as per Table 8-60 .
03h	1	Memory Req Opcode: The Opcode extracted from the memory request. <ul style="list-style-type: none"> • Bits[3:0]: Memory Req Opcode (see Table 3-35) • Bits[7:4]: Reserved Memory Req Opcode value is valid only if the Memory Req Opcode bit is set.
04h	2	Flags: Various valid and condition flags. <ul style="list-style-type: none"> • Bit[0]: Reserved • Bit[1]: Hardwire to 1 • Bit[2]: DPA Valid: If 1, DPA field is valid • Bit[3]: HPA Valid: If 1, HPA field is valid • Bit[4]: Reserved • Bit[5]: Memory Req Opcode Valid: If 1, Memory Req Opcode value is valid • Bits[15:6]: Reserved
06h	2	RPID: Requester PID of request that was being processed for which an error was detected and/or this event notification triggered. This PID must match one in use within this host's domain. <ul style="list-style-type: none"> • Bits[11:0]: RPID of Request • Bits[15:12]: Reserved
08h	8	DPA: Device Physical Address decoded from requester's provided HPA. See Section 7.7.2.4 . DPA value is valid only if the DPA valid bit is set.
10h	8	HPA: Host Physical Address extracted from request. See Section 7.7.2.4 . HPA value is valid only if the HPA valid bit is set.
18h	8	Timestamp: Event Record Timestamp per Table 8-43 , Common Event Record Format.

8.2.9.2.11 Get GAM Buffer (Opcode 0109h)

This command is used by a host to read the local pointer used for the GAM Buffer, as described in [Section 3.1.11.7](#) and [Section 8.2.9.2.10](#). It is supported only by GAEs.

The format of the GAM Buffer entries is defined in [Table 8-61](#).

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error

Command Effects:

- None

Table 8-63. Get GAM Buffer Response Payload

Byte Offset	Length in Bytes	Description
00h	8	<ul style="list-style-type: none"> Bit[0]: Valid: Indicates whether the GAM Buffer Address is valid (1) or invalid (0) Bit[1]: Buffer Overflow: When set, indicates that the GAE was unable to log a received GAM payload to the GAM Buffer because there were no available entries Bits[4:2]: Reserved Bits[51:5]: GAM Buffer Address[51:5]: Address to which the GAM payload will be written by the GAE Bits[55:52]: Reserved Bits[63:56]: Head Index: Indicates the index at which the GAE will log the next received GAM payload

8.2.9.2.12 Set GAM Buffer (Opcode 010Ah)

This command is used by a host to update the local pointer used for the GAM Buffer, as described in [Section 3.11.7](#) and [Section 8.2.9.2.10](#). It is supported only by GAEs.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error

Command Effects:

- None

Table 8-64. Set GAM Buffer Address Request Payload

Byte Offset	Length in Bytes	Description
00h	8	<ul style="list-style-type: none"> Bit[0]: Valid: Indicates whether the GAM Buffer Address is valid (1) or invalid (0) Bit[1]: Clear Overflow: When set, clears the "Buffer Overflow" bit reported in Get GAM Buffer Bits[4:2]: Reserved Bits[51:5]: GAM Buffer Address[51:5]: Address to which the GAM payload will be written by the GAE Bits[55:52]: Reserved Bits[63:56]: Tail Index: Indicates the index up to which the host has read GAM Buffer entries

8.2.9.3 Firmware Update

FW Update is an optional feature for devices to provide a mechanism to update the FW. If supported, the Get FW Info command and the Transfer FW command shall be supported; the Activate FW command is optional.

FW refers to an FW package that may contain multiple FW images. The management of multiple FW images or the FW on multiple controllers is the responsibility of the device and beyond the scope of this specification. Product vendors can implement any means of managing multiple FW images, provided those means do not conflict with or alter the specifications described herein.

The number of FW slots the device supports is vendor specific, up to four. The minimum FW slots supported shall be two, one slot for the active FW and one slot for an FW package that is staged for activation. Only one slot may be active at a time and only

one slot may be staged for activation at a time. The staged FW will transition to the active FW on next reset (as specified in the Transfer FW command effects) or upon successful execution of the Activate FW command, if supported. Other FW packages that are fully transferred and stored in a slot shall persist across power cycles.

8.2.9.3.1 Get FW Info (Opcode 0200h)

Retrieve information about the device FW.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-65. Get FW Info Output Payload

Byte Offset	Length in Bytes	Description
00h	1	FW Slots Supported: The number of FW slots the device supports, up to four. The minimum FW slots supported shall be two, one slot for the active FW and one slot for an FW that is staged for activation.
01h	1	FW Slot Info: Indicates the active and staged FW slots. <ul style="list-style-type: none"> • Bits[2:0]: The slot number of the active FW version. Only one slot may be active at a time. 0 is an illegal value. Values greater than "FW Slots Supported" are also illegal. • Bits[5:3]: The slot number of the FW that will be activated on the next reset or on successful execution of the Activate FW command, if supported. If 0, no FW is currently staged for activation. Refer to the FW Activation Capabilities to determine whether the FW can be activated at runtime. If the FW fails to activate, the device shall fall back to the previously active FW. Only one slot may be staged for activation at a time. Values greater than "FW Slots Supported" are illegal. • Bits[7:6]: Reserved
02h	1	FW Activation Capabilities: Defines the capabilities supported by the device for activating a new FW without a reset. <ul style="list-style-type: none"> • Bit[0]: When set, the device supports online FW activation with the Activate FW command • Bits[7:1]: Reserved
03h	13	Reserved
10h	16	Slot 1 FW Revision: Contains the revision of the FW package in Slot 1 formatted as an ASCII string. If there is no FW in Slot 1, this field shall be cleared to 0.
20h	16	Slot 2 FW Revision: Contains the revision of the FW package in Slot 2 formatted as an ASCII string. If there is no FW in Slot 2, this field shall be cleared to 0.
30h	16	Slot 3 FW Revision: Contains the revision of the FW package in Slot 3 formatted as an ASCII string. If there is no FW in Slot 3 or the device does not support 3 or more FW slots, this field shall be cleared to 0.
40h	16	Slot 4 FW Revision: Contains the revision of the FW package in Slot 4 formatted as an ASCII string. If there is no FW in Slot 4 or the device does not support 4 FW slots, this field shall be cleared to 0.

8.2.9.3.2 Transfer FW (Opcode 0201h)

Transfer all or part of an FW package from the caller to the device. FW packages and all parts of each FW package shall be 128-byte aligned.

If the FW package is transferred in its entirety, the caller makes one call to Update FW with Action = Full FW Transfer.

If an FW package is transferred in parts, the caller makes one call to Transfer FW with Action = Initiate FW Transfer, zero or more calls with Action = Continue FW Transfer, and one call with Action = End Transfer or Abort Transfer. The FW package parts shall be transferred in order; otherwise, the device shall return the FW Transfer Out of Order return code. Back-to-back retransmission of any FW package part is permitted during a transfer.

IMPLEMENTATION NOTE

Example of FW image transmissions:

The following example sequence transfers 380h bytes worth of an FW image. Note that the Offset field is expressed in multiples of 80h Bytes:

1. Transfer FW (Action=Initiate FW Transfer, Slot=x, Offset=0, 100h B package).
2. Transfer FW (Action=Continue FW Transfer, Slot=x, Offset=2, 100h B package).
3. Transfer FW (Action=Continue FW Transfer, Slot=x, Offset=4, 100h B package).
4. Transfer FW (Action=End Transfer, Slot=1, Offset=6, 80h B package).

The following example is also a valid in-order sequence. It also transfers 380h bytes worth of an FW image. Note that Bytes 100h-1FFh are transmitted twice, but are sent back-to-back and are therefore legal:

1. Transfer FW (Action=Initiate FW Transfer, Slot=x, Offset=0, 100h B package).
2. Transfer FW (Action=Continue FW Transfer, Slot=x, Offset=2, 100h B package).
3. Transfer FW (Action=Continue FW Transfer, Slot=x, Offset=2, 100h B package).
4. Transfer FW (Action=Continue FW Transfer, Slot=x, Offset=4, 100h B package).
5. Transfer FW (Action=End Transfer, Slot=1, Offset=6, 80h B package).

The following example is an invalid in-order sequence:

1. Transfer FW (Action=Initiate FW Transfer, Slot=x, Offset=0, 100h B package).
2. Transfer FW (Action=Continue FW Transfer, Slot=x, Offset=2, 100h B package).
3. Transfer FW (Action=Continue FW Transfer, Slot=x, Offset=3, 180h B package).
4. Transfer FW (Action=End Transfer, Slot=1, Offset=6, 80h B package).

The following example is also an invalid in-order sequence:

1. Transfer FW (Action=Initiate FW Transfer, Slot=x, Offset=0, 100h B package).
2. Transfer FW (Action=Continue FW Transfer, Slot=x, Offset=2, 100h B package).
3. Transfer FW (Action=Continue FW Transfer, Slot=x, Offset=5, 80h B package).
4. Transfer FW (Action=End Transfer, Slot=1, Offset=6, 80h B package).

If an FW package transfer that initiated on a Mailbox CCI is interrupted by a Conventional or CXL reset, the FW package transfer shall be aborted by the device. If a FW package transfer is aborted prior to the entire FW package being successfully stored on the device, the device shall require the FW package transfer to be started from the beginning of the FW package.

Only one FW package may be transferred to a single CCI at a time. The device shall return the FW Transfer in Progress return code if it receives a Transfer FW command with Action = Full FW Transfer or Action = Initiate FW Transfer until the current FW package transfer is completed or aborted.

In case of a device with multiple CCIs, the device shall return the FW Transfer in Progress return code if it receives a Transfer FW request on any other CCI than the CCI that initiated the current FW package transfer until that transfer is completed or aborted. The timeout between FW package parts is 30 seconds, after which point the device shall consider the current FW package transfer aborted.

Once the entire FW package is fully transferred to the device, the device shall verify the FW package and store it in the specified slot. Verification of the FW package is vendor specific.

In support of confidential computing, if the device has been locked while utilizing secure CXL TSP interfaces and the device was not enabled for FW update after being locked, the device shall reject the Transfer FW command by returning Invalid Security State status. See [Section 11.5](#) for details on locking a device and locked device behavior.

Possible Command Return Codes:

- Success
- Background Command Started
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Media Disabled
- Busy
- FW Transfer in Progress
- FW Transfer Out of Order
- FW Verification Failed
- Invalid Slot
- Aborted
- Invalid Security State

Command Effects:

- Configuration Change after Cold Reset
- CEL[11:10] Valid
- Configuration Change after Conventional Reset
- Configuration Change after CXL Reset
- Background Operation

- Request Abort Background Operation Command Supported

Table 8-66. Transfer FW Input Payload

Byte Offset	Length in Bytes	Description
00h	1	Action: Specifies the stage of the FW package transfer. <ul style="list-style-type: none"> • 00h = Full FW Transfer • 01h = Initiate FW Transfer • 02h = Continue FW Transfer • 03h = End Transfer • 04h = Abort Transfer • All other encodings are reserved
01h	1	Slot: Specifies the FW slot number to store the FW once the transfer is complete and the FW package is validated. Shall not be the active FW slot; otherwise, the Invalid Slot return code shall be returned. Valid only if Action = Full transfer or End Transfer; otherwise, ignored.
02h	2	Reserved
04h	4	Offset: The byte offset in the FW package data. Expressed in multiples of 128 bytes. Ignored if Action = Full FW Transfer.
08h	78h	Reserved
80h	Varies	Data: The FW package data.

8.2.9.3.3 Activate FW (Opcode 0202h)

Activate FW is an optional command to make a FW previously stored on the device with the Transfer FW command, the active FW. Devices may support activating a new FW while online or on reset, as indicated by the Get FW Info command.

If a device supports online firmware activation, this command may be executed as a background command as indicated by the command return code.

If the new FW fails to online activate, the device shall roll back to the previous FW, if possible. A cold reset may be required to restore the operating state of the FW on activation failure.

In support of confidential computing, if the device has been locked while utilizing secure CXL TSP interfaces and the device was not enabled for FW update after being locked, the device shall reject this command by returning Invalid Security State status. See [Section 11.5](#) for details on locking a device and locked device behavior.

Possible Command Return Codes:

- Success
- Background Command Started
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Busy
- Activation Failed, FW Rolled Back
- Activation Failed, Cold Reset Required
- Invalid Slot

- Aborted
- Invalid Security State

Command Effects:

- Configuration Change after Cold Reset
- CEL[11:10] Valid
- Configuration Change after Conventional Reset
- Configuration Change after CXL Reset
- Immediate Configuration Change
- Background Operation
- Request Abort Background Operation Command Supported

Table 8-67. Activate FW Input Payload

Byte Offset	Length in Bytes	Description
00h	1	Action: Specifies the activation method. <ul style="list-style-type: none"> • 00h = Online • 01h = On the next reset (as specified in the command effects) • All other encodings are reserved
01h	1	Slot: Specifies the FW slot number to activate. Shall not be the active FW slot; otherwise, the Invalid Slot return code shall be returned.

8.2.9.4 Timestamp

Timestamp is an optional setting that enables the host to set a time value in the device to correlate the device timer with the system time. The use of the timestamp is beyond the scope of this specification. The accuracy of the timestamp after it is set may be affected by vendor specific factors. Therefore, the timestamp shouldn't be used for time sensitive applications. Although the format of the timestamp is in nanoseconds, the resolution of time maintained by the device is implementation specific, so software shall not assume a device provides nanosecond resolution.

8.2.9.4.1 Get Timestamp (Opcode 0300h)

Get the timestamp from the device. Timestamp is initialized via the Set Timestamp command. If the timestamp has never been set, the output shall be 0.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-68. Get Timestamp Output Payload

Byte Offset	Length in Bytes	Description
00h	8	Timestamp: The current device timestamp which represents the value set with the Set Timestamp command plus the number of nanoseconds that have elapsed since the timestamp was set.

8.2.9.4.2 Set Timestamp (Opcode 0301h)

Set the timestamp on the device. It is recommended that the host set the timestamp after every Conventional or CXL Reset. Otherwise, the timestamp may be inaccurate.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- Immediate Policy Change

Table 8-69. Set Timestamp Input Payload

Byte Offset	Length in Bytes	Description
00h	8	Timestamp: The number of unsigned nanoseconds that have elapsed since midnight, 01-Jan-1970, UTC.

8.2.9.5 Logs

Commands to return device-specific logs.

8.2.9.5.1 Get Supported Logs (Opcode 0400h)

Retrieve the list of device-specific logs (identified by UUID) and the maximum size of each Log.

Possible Command Return Codes:

- Success
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-70. Get Supported Logs Output Payload

Byte Offset	Length in Bytes	Description
00h	2	Number of Supported Log Entries: The number of Supported Log Entries returned in the output payload.
02h	6	Reserved
08h	Varies	Supported Log Entries: Device-specific list of supported log identifier UUIDs and the current size of each log.

IMPLEMENTATION NOTE

It is strongly recommended that the Get Supported Logs Sub-List (see [Section 8.2.9.5.6](#)) is supported by Components and used by software instead of Get Supported Logs so that requesters may control the output payload size, as needed. Type 3 Devices that implement support for the Get Supported Logs opcode on an MCTP-based CCI shall also support the Get Supported Logs Sub-List opcode.

Table 8-71. Get Supported Logs Supported Log Entry

Byte Offset	Length in Bytes	Description
00h	10h	Log Identifier: UUID representing the log for which to retrieve data. The following Log Identifier UUIDs are defined in this specification: <ul style="list-style-type: none"> 0da9c0b5-bf41-4b78-8f79-96b1623b3f17 – Command Effects Log (CEL) 5e1819d9-11a9-400c-811f-d60719403d86 – Vendor Debug Log b3fab4cf-01b6-4332-943e-5e9962f23567 – Component State Dump Log f1720d60-a7a9-4306-a003-11948f9e077c – DDR5 Error Check Scrub (ECS) Log e6dfa32c-d13e-4a5c-8ca8-99bebbf731a4 – Media Test Capability Log 2c255522-8ce4-11ec-b909-0242ac120002 – Media Test Results Short Log c1fe0b3e-7a00-448e-a24e-a6aabbfe587a – Media Test Results Long Log
10h	4	Log Size: The maximum number of bytes of log data available to retrieve for the log identifier.

8.2.9.5.2 Get Log (Opcode 0401h)

Retrieve a log from the device, identified by a specific UUID. The host shall retrieve the size of the log first using the Get Supported Logs command, then issue enough of these commands to retrieve all the log information, incrementing the Log Offset each time. The device shall return Invalid Input if the Offset or Length fields attempt to access beyond the size of the log as reported by Get Supported Logs.

Possible Command Return Codes:

- Success
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Invalid Log
- Media Disabled
- Busy

Command Effects:

- None

Table 8-72. Get Log Input Payload

Byte Offset	Length in Bytes	Description
00h	10h	Log Identifier: UUID representing the log for which to retrieve data. See Table 8-71 for the list of Log Identifier UUIDs defined in this specification.
10h	4	Offset: The byte offset in the log data to return in the output payload.
14h	4	Length: Length in bytes of log data to return in the output payload.

Table 8-73. Get Log Output Payload

Byte Offset	Length in Bytes	Description
00h	Varies	Log Data

8.2.9.5.2.1 Command Effects Log (CEL)

The Command Effects Log (CEL) is a variable-length log page that reports each command supported by the CCI that was queried (with the exception of the secondary mailbox, as described in [Section 8.2.8.4.2](#)) and the effect each command may have on the device subsystem. The effects of a given command may be different based on various factors (e.g., the input payload specified when the command was issued). Therefore, all possible command effects for a given command shall be reported by the device in the CEL.

Devices shall implement the CEL for all commands supported by the device, including any vendor specific commands that extend beyond those specified in this specification.

Some host drivers may not allow unspecified commands to be passed through to the device if the commands are not advertised in the CEL.

The CEL shall use a Log Identifier of:

- 0da9c0b5-bf41-4b78-8f79-96b1623b3f17

Table 8-74. CEL Output Payload

Byte Offset	Length in Bytes	Description
00h	4	Command 1 CEL Entry: Contains the Command Effects Log Entry for 1st supported command.
04h	4	Command 2 CEL Entry: Contains the Command Effects Log Entry for 2nd supported command.
(4*(n-1))h	4	Command n CEL Entry: Contains the Command Effects Log Entry for nth supported command.

Each CEL entry shall have a specific set of bit definitions describing the effect of issuing the command as outlined below.

Table 8-75. CEL Entry Structure

Byte Offset	Length in Bytes	Description
00h	2	<p>Opcode: The command opcode.</p>
02h	2	<p>Command Effect: Bitmask that contains one or more effects for the command opcode.</p> <ul style="list-style-type: none"> • Bit[0]: Configuration Change after Cold Reset: When set, this opcode makes a driver visible change to the configuration of the device or data contained within persistent memory regions of the device. The change does not take effect until a device cold reset. • Bit[1]: Immediate Configuration Change: When set, this opcode makes an immediate driver visible change to the configuration of the device or data contained within persistent memory regions of the device. • Bit[2]: Immediate Data Change: When set, this opcode makes an immediate driver visible change to the data written to the device. • Bit[3]: Immediate Policy Change: When set, this opcode makes an immediate change to the policies used by the device. • Bit[4]: Immediate Log Change: When set, this opcode makes an immediate change to a device log. • Bit[5]: Security State Change: When set, this opcode results in an immediate driver visible change in the security state of the device. Security state changes that require a reboot to take effect do not use this effect. • Bit[6]: Background Operation: When set, this opcode is executed in the background. • Bit[7]: Secondary Mailbox Supported: When set, submitting this opcode via the secondary mailbox is supported; otherwise, this opcode will return Unsupported Mailbox or CCI if issued on the secondary mailbox. Only valid when returned on the primary or secondary mailbox. This bit is reserved if the CEL is being returned from any CCI other than the primary or secondary mailbox. • Bit[8]: Request Abort Background Operation Supported: When set, this opcode can be requested to be aborted by issuing a Request Abort Background Operation command. This bit is reserved if bit[6] is cleared to 0. • Bit[9]: CEL[11:10] Valid: When set, bit 11 and bit 10 of this field are valid. • Bit[10]: Configuration Change after Conventional Reset: When set, this opcode makes a driver visible change to the configuration of the device or data contained within persistent memory regions of the device. The change does not take effect until a Conventional Reset. • Bit[11]: Configuration Change after CXL Reset: When set, this opcode makes a driver visible change to the configuration of the device or data contained within persistent memory regions of the device. The change does not take effect until a CXL Reset. • Bits[15:12]: Reserved: Shall be cleared to 0h.

8.2.9.5.2.2 Vendor Debug Log

All devices that support a debug log shall support the Vendor Debug Log to allow the log to be accessed through a common host driver, for any device, with Log Identifier of:

- 5e1819d9-11a9-400c-811f-d60719403d86

The contents of the output payload are vendor specific.

8.2.9.5.2.3 Component State Dump Log

Log Identifier: b3fab4cf-01b6-4332-943e-5e9962f23567

The Component State Dump Log is an optional method for allowing vendor specific state information to be extracted using standard drivers.

The Component State Dump Log can be populated in two ways:

- Auto populate
- Manual populate using Populate Log

A component that supports the Component State Dump Log shall support at least one of the above methods.

The two methods and their associated trigger requirements are detailed in [Table 8-76](#). The Component State Dump Log shall be populated by a given method if the trigger occurs, and the logical AND of all the conditions for that trigger is true.

Table 8-76.

Component State Dump Log Population Methods and Triggers

Method	Trigger	Condition	Condition Reference
Auto Populate	Vendor-specific	Auto Populate Trigger Count Since Clear = 0	Table 8-77 . Component State Dump Log Format
		Auto Populate Supported = 1	Table 8-90 . Get Log Capabilities Output Payload
Manual Populate	Populate Log command received	Log Identifier = b3fab4cf-01b6-4332-943e-5e9962f23567	Table 8-92 . Populate Log Input Payload
		Populate Log Supported = 1	Table 8-90 . Get Log Capabilities Output Payload

The trigger for the auto populate method is vendor specific, but one example may be a severe internal error in the component.

When a population method triggers and all required conditions are met, any existing Component State Dump Data is cleared before populating the new log contents.

The log contents should persist across cold reset. The component shall indicate whether the log persists across cold reset using the Persistent Across Cold Reset bit in the Get Log Capabilities Output Payload.

If the component has Component State Dump Data available to be reported in the Component State Dump Log after a subsequent reset, the Component State Dump Log contents shall be available when the Mailbox Interfaces Ready bit in the Memory Device Status register is set to 1.

To handle corner cases related to an existing Component State Dump Log being overwritten by an Auto Populate trigger while host software is reading the existing contents of the log, host software must begin each Component State Dump Log fetch sequence by issuing a Get Log command with Offset = 0, followed by zero or more Get Log commands with nonzero offset. If the component is reset, host software must start a new fetch sequence.

If a Get Log command with nonzero Offset is received requesting the Component State Dump Log, the component shall apply the first applicable case from the following list:

- Return Invalid Input if the component has not previously returned Success for a Get Log command with Offset = 0 requesting the Component State Dump Log.
- Return Interrupted if the contents of the Component State Dump Log have changed since the last time the component returned Success for a Get Log command with Offset = 0 requesting the Component State Dump Log.
- Return Success and provide the log contents of the specified offset corresponding to the state of the Component State Dump Log when the current fetch sequence began (i.e., when the last Get Log command with Offset = 0 requesting the Component State Dump Log was completed with a return code of Success).

If a Get Log command with Offset = 0 is received requesting the Component State Dump Log, the component shall apply the first applicable case from the following list:

- Return Retry Required if the log contents are not fully populated.
- Return Success and provide the log contents starting at Offset = 0.

When starting a new fetch sequence for a Component State Dump Log that was manually populated using Populate Log, host software should issue a single Get Log command requesting the Component State Dump Log with Offset = 0 and Length >= 24h. Host software should verify that the Auto Populate Data flag is cleared to 0. If the Auto Populate Data flag is set to 1, this indicates the manually populated Component State Dump Log was overwritten by an Auto Populate operation, and the Component State Dump Data corresponds to the results of that Auto Populate operation.

If an Auto Populate trigger occurs while the component is processing a Manual Populate operation triggered by a Populate Log command, and the component is still capable of returning a completion, the component shall complete the Populate Log command with a return code of Interrupted.

The format of the Component State Dump Log is defined in [Table 8-77](#).

Table 8-77. Component State Dump Log Format

Byte Offset	Length in Bytes	Description
00h	4	Component State Dump Data Length: Length of the Component State Dump Data field in bytes.
04h	1	Auto Populate Trigger Count Since Clear: The number of times the component has encountered the Trigger for the Auto Populate method since the last time the Component State Dump Log was cleared. Tracking this value is optional. If this value is tracked, it should persist across cold reset. This value is cleared to 00h when the Component State Dump Log is cleared using Clear Log, or the Component State Dump Log is populated using Populate Log. If the component tracks this value, and the Auto Populate Data bit in the Flags field is set to 1, this field shall be at least 01h. If the component does not track this value, this field shall be cleared to 00h. This value shall saturate at FFh instead of rolling over to 00h.
05h	1	Event Log: The Event Log, as defined in Table 8-52 (Get Event Records Input Payload), containing the Associated Event Record Handle. If the Associated Event Record Handle is 0, the value of this field is undefined.
06h	2	Associated Event Record Handle: The Event Record Handle, as defined in Table 8-43 (Common Event Record Format), corresponding to an Event Record that is associated with the Auto Populate trigger that generated the Component State Dump Data. If there are no associated Event Records, this field shall be cleared to 00h. If there was an associated Event Record, but it is no longer in the Event Log, this field shall be cleared to 00h. If the Auto Populate Data bit in the Flags field is cleared to 0, this field is undefined.
08h	8	Timestamp: The Timestamp, as defined by the Timestamp field in Table 8-68 (Get Timestamp Output Payload), at the time the Component State Dump Data was generated.
10h	10h	Component State Dump Format UUID: Optional value to uniquely identify the format of the Component State Dump Data field. A value of all 0s indicates that the format of the Component State Dump Data is not indicated.
20h	4	Flags <ul style="list-style-type: none"> • Bit[0]: Auto Populate Data: Set to 1 if the Component State Dump Data was generated using the Auto Populate method. Cleared to 0 if the Component State Dump Data Length is 0, or if the Component State Dump Data was generated using Populate Log. • Bits[31:1]: Reserved
24h	1Ch	Reserved
40h	Varies	Component State Dump Data: Vendor specific.

8.2.9.5.2.4 DDR5 Error Check Scrub (ECS) Log

Log Identifier: f1720d60-a7a9-4306-a003-11948f9e077c

DDR5 ECS Log allows the host to observe the ECS operation results. The format of the DDR5 ECS Log is shown in [Table 8-78](#).

Table 8-78. DDR5 Error Check Scrub (ECS) Log (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	4	Common Header <ul style="list-style-type: none"> Bits[9:0]: Total Number of Entries: The total number of entries. If the log entry type is per DRAM, then the total number of entries depends on the number of memory media FRUs within the device and the number of monolithic memory dies for each memory media FRU. If the log entry type is per memory media FRU, then the total number of entries depends only on the number of memory media FRUs within the device (i.e., only one entry per memory media FRU). Bit[10]: If 0, indicates that the format of Component ID field is vendor specific. If 1, the format of the Component ID field is defined in Table 8-44. Bits[12:11]: Entry Type: The entry type of how the ECS log is reported. The entry type is defined commonly for all memory media FRUs within the device. <ul style="list-style-type: none"> 00b = Per DRAM 01b = Per Memory Media FRU All other encodings are reserved Bits[31:13]: Reserved
04h	10h	Entry 1 Component Identifier : This is used to identify a field replaceable sub-component associated with Entry[1]. When bit[10] of the Common Header is set, the component identifier is defined by Table 8-44 ; otherwise, the component identifier format is device specific.
14h	2	Entry 1 DDR5 ECS Configurations <ul style="list-style-type: none"> Bits[2:0]: ECS Threshold Count per Gb of Memory Cells: <ul style="list-style-type: none"> 011b = 256 (default) 100b = 1024 101b = 4096 All other encodings are reserved Bit[3]: Codeword/Row Count Mode: <ul style="list-style-type: none"> 0 = ECS counts rows with errors 1 = ECS counts codewords with errors Bits[15:4]: Reserved
16h	10h	Entry 1 Error Count and Address Information : If the entry type is DRAM, then each entry includes the error count and address information for each DRAM monolithic die. If the entry type is Memory Media FRU, then each entry includes the error count that is the sum of error counts accumulated from all DRAMs placed in the memory media FRU associated with the entry, and the address with the highest error count among all DRAMs placed in the memory media FRU associated with the entry. <ul style="list-style-type: none"> Bit[0]: Error Found: Error exists in the DRAM die and a valid log exists. If no error exists and the log is not required to be read, then this bit is cleared to 0. Bits[7:1]: Reserved. Bits[23:8]: Error Count or the Number of Rows or Codeword Errors: Error count is tracked based on two different modes selected by DDR5 ECS Control Feature — Codeword mode and Row Count mode. If the ECS is under Codeword mode, then the error count increments each time a codeword with check bit errors is detected. If the ECS is under Row Count mode, then the error counter increments each time a row with check bit errors is detected. The count is subject to the ECS Threshold count per Gbit, which masks error counts less than the Threshold. Bits[31:24]: Max Row Error Count: Number of errors within the row with the most errors. This count is also used to determine the DRAM row address with the maximum number of errors. Bits[95:32]: Address with Max Errors: Address with the highest error count, where the address is the DPA translated from the row, the bank, and the bank group addresses of the DRAM. Bits[127:96]: Reserved.
...
04h+ ((n-1)*22h)	10h	Entry n Component Identifier

Table 8-78. DDR5 Error Check Scrub (ECS) Log (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
14h+ ((n-1)*22h)	2	Entry n DDR5 ECS Configurations
16h+ ((n-1)*22h)	10h	Entry n Error Count and Address Information

8.2.9.5.2.5 Media Test Capability Log

Media Test Capability Log is a variable-length log structure that conveys the attributes of the different media tests that the CXL device supports. It is composed of a common header (see [Table 8-79](#)) and Media Test Capability Log Entries (see [Table 8-81](#)) for each supported test.

Table 8-79. Media Test Capability Log Output Payload

Byte Offset	Length in Bytes	Description
00h	16	Common Header: Reports attributes applicable to all the tests and general capabilities of the device.
10h	16	Test 1 Media Test Capability Log Entry: Contains the descriptor of the first supported media test.
20h	16	Test 2 Media Test Capability Log Entry: Contains the descriptor of the second supported media test.
...
(16+16*(n-1))h	16	Test n Media Test Capability Log Entry: Contains the descriptor of the nth supported media test.

[Table 8-80](#) describes the attributes that are common to all the tests that the device supports (e.g., Error Signature List Size, Media Test Result Long, and Short Log versions and Capabilities flags).

Table 8-80. Media Test Capability Log Common Header (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	1	Number of Supported Tests: Total number of test types that the device supports.
01h	4	Total Number of Error Signatures: Maximum number of Error Signatures tracked by the device that are related to all executed tests. Error Signatures of each test are reported in the Media Test Results Log. If the total number of errors occurred during the execution of the test overflows, it will be signaled in the Media Test Results Log.
05h	1	Media Test Result Long Log Version: Version of the Media Test Results Long Log that is used to report the test results (see Table 8-86).
06h	1	Media Test Result Short Log Version: Version of the Media Test Results Short Log that is used to report the test results (see Table 8-83).

Table 8-80. Media Test Capability Log Common Header (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
07h	1	<p>Capabilities: Common capabilities applicable to all the tests that the device supports.</p> <ul style="list-style-type: none"> Bit[0]: Data ECC Disablement Capability: Value 1 indicates that the ECC implemented for the user data on the CXL Type 3 device controller can be disabled. The area on the medium that is used to store ECC parity bits can be tested with the same algorithm and pattern as the area that contains user data. If cleared to 0, the ECC implemented on the CXL Type 3 device internal controller cannot be disabled and the area that is used for the ECC parity cannot be tested. Bit[1]: Metadata ECC Disablement Capability: Value 1 indicates that the ECC implemented for the metadata (e.g., Meta0-State Value) on the CXL Type 3 device controller can be disabled. The area on the medium that is used to store ECC parity bits can be tested with the same algorithm and pattern as the area that contains user data. If cleared to 0, the ECC implemented on the CXL Type 3 device internal controller cannot be disabled and the area that is used for the ECC parity cannot be tested. Bit[2]: Data and Metadata ECC Disablement Capability: Value 1 indicates that the ECC implemented for data and metadata (e.g., Meta0-State Value) on the CXL Type 3 device controller can be disabled. The area on the medium that is used to store ECC parity bits can be tested with the same algorithm and pattern as the area that contains user data. If cleared to 0, the ECC implemented on the CXL Type 3 device internal controller cannot be disabled and the area that is used for the ECC parity cannot be tested. If bit[0] and bit[1] are set to 1, bit[2] shall also be set to 1. Bit[3]: Metadata Area Testing Capability: Value 1 indicates that the area on the medium that is used to store metadata (e.g., Meta0-State Value and Extended Metadata) is tested with the same algorithm and pattern as the area that contains user data. If cleared to 0, the area that is used to store metadata is not tested. Bits[7:4]: Reserved.
08h	8	Reserved

For each media test that the device supports, an individual Media Test Capability Log Entry shall be defined. A single test shall be described by the fields defined in [Table 8-81](#), comprising the Test ID, the Algorithm of the Test, the estimated Execution time per GB, etc.

Table 8-81. Media Test Capability Log Entry Structure (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	2	Test ID: Unique ID to identify the Test. Test ID shall be used to reference the test to be executed in the Perform Maintenance command input payload.
02h	1	<p>Algorithm: Media test algorithm supported.</p> <ul style="list-style-type: none"> 00h = Reserved 01h = Write, Read and Compare pattern 02h = Checkerboard 03h = MARCH 04h = MATS 05h = MATS+ 06h = Walking 1s 07h = Walking 0s All other encodings are reserved

Table 8-81. Media Test Capability Log Entry Structure (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
03h	1	<p>Execution Time: Maximum test execution time per GB.</p> <ul style="list-style-type: none"> Bits[3:0]: These bits specify the time scale. <ul style="list-style-type: none"> 1h = 10 ms 2h = 100 ms 3h = 1 s 4h = 10 s 5h = 100 s 6h = 1000 s All other encodings are reserved Bits[7:4]: These bits specify the maximum operation latency with the time scale indicated in bits[3:0].
04h	2	<p>Capabilities:</p> <ul style="list-style-type: none"> Bit[0]: Address Configurable Flag: If set, the address range is configurable. Bit[1]: Inverse Pattern Support: If set, the test can be repeated, complementing the selected pattern. Bit[2]: Exit on Uncorrectable Error: The test can be stopped on the first uncorrectable error. If the ECC is disabled, any bit error on the media will result in an uncorrectable error. Bit[3]: Error Count Threshold Programmable. Bit[4]: Update Poison List on Uncorrectable Error: The device can record the DPA impacted by an uncorrectable error in the Poison List if the device supports the Poison List. Bits[8:5]: Addressing Mode: <ul style="list-style-type: none"> Bit[5]: Ascending Bit[6]: Descending Bit[7]: Algorithm Specific Bit[8]: Random Reserved. <p>For each supported algorithm, the permitted Address mode settings are summarized as follows:</p> <ul style="list-style-type: none"> Write pattern: Ascending, Descending, Random Write, Read and Compare pattern: Ascending, Descending, Random Read pattern: Ascending, Descending, Random Checkerboard: Algorithm specific MARCH: Algorithm specific MATS: Algorithm specific MATS+: Algorithm specific Walking 1s: Algorithm specific Walking 0s: Algorithm specific
06h	2	<p>Supported Patterns: Bitmap to advertise supported 64B patterns. This field is invalid if the Algorithm field value is 03h (Read pattern).</p> <ul style="list-style-type: none"> Bit[0]: User provided. Bit[1]: Vendor specific. Bit[2]: Generated by a PRBS whose length is reported in the PRBS length field. Bit[3]: DPA[63:0] value is repeated 8 times to cover 64B. If the device allows testing of the area that stores ECC parity or metadata, the DPA value shall be used to cover the corresponding bits. Bit[4]: 55h. Bit[5]: AAh. Bits[15:6]: Reserved.
08h	1	PRBS Length: Length of the PRBS sequence, calculated as $2^{\text{PRBS Length} - 1}$. Maximum value is 32. Value 0 is invalid. This field is valid if PRBS pattern is supported.
09h	7	Reserved

8.2.9.5.2.6 Media Test Results Logs

The Media Test Results Logs are variable-length logs that provide the results of one or more Media Tests. Two types of logs are available:

- **Media Test Results Short Log:** Status info and results of the execute tests (see [Table 8-82](#))
- **Media Test Results Long Log:** Detailed error information and error signatures of the executed tests (see [Table 8-85](#))

Media Test Result Logs are produced at the end of the execution of the tests and are cleared when a new test starts or when the Clear Log command is issued.

Media Test Results Short Log enumerates the results of the tests executed by the CXL device.

Table 8-82. Media Test Results Short Log

Byte Offset	Length in Bytes	Description
00h	10h	Common Header: Common output information from test execution.
10h	20h	Test 1 Media Test Results Short Log Entry: Contains the results of the first executed media test in short format.
30h	20h	Test 2 Media Test Results Short Log Entry: Contains the results of the second executed media test in short format.
...
(10h+20h*(n-1))h	20h	Test n Media Test Results Short Log Entry: Contains the results of the nth configured media test in short format.

Each Media Test Results Short Log Entry contains the results of the test executed (see [Table 8-84](#)). They are preceded by a common header described in [Table 8-83](#).

Table 8-83. Media Test Results Short Log Entry Common Header

Byte Offset	Length in Bytes	Description
00h	1	Number of Tests Executed: Total number of tests executed by the device. This number can be smaller than the total number of tests that were requested.
01h	1	Version: Version of the Media Test Result Short Log. This field shall be set to 01h. The version is incremented whenever the Media Test Results Short Log is extended to add more functionalities. Backward compatibility shall be maintained during this process. For all values of n, version n+1 may extend version n by replacing fields that are marked as reserved in version n or appending new fields but must not redefine the meaning of existing fields.
02h	1	Result: This field indicates the outcome of the execution of the requested list of tests. <ul style="list-style-type: none"> • 00h = All the tests completed successfully. • 01h = At least one test completed with a failure. • 02h = Test execution was interrupted by a Request Abort Background Operation command. All the tests that were complete, before the abort request was processed, ended successfully. • 03h = Test execution was interrupted by a Request Abort Background Operation command. At least one test completed with failure before the abort request ended. • All other encodings are reserved.
03h	Dh	Reserved

Table 8-84. Media Test Results Short Log Entry Structure

Byte Offset	Length in Bytes	Description
00h	2	Test ID: ID of the test.
02h	8	Start Time: Expressed as timestamp, it represents the number of unsigned nanoseconds that have elapsed since midnight, 01-Jan-1970, UTC.
0Ah	8	End Time: Expressed as timestamp, it represents the number of unsigned nanoseconds that have elapsed since midnight, 01-Jan-1970, UTC.
12h	1	Result: This field indicates the outcome of the test execution. <ul style="list-style-type: none"> • 00h = Completed with success • 01h = Completed with failure • 02h = Aborted by a Request Abort Background Operation command • All other encodings are reserved
13h	1	Flags: Bit values indicating conditions occurred during the test execution. <ul style="list-style-type: none"> • Bit[0]: Error Signature List Overflow • Bits[7:1]: Reserved
14h	4	Uncorrectable Error Count: Total number of uncorrectable memory errors that the device detected during the test. If the ECC is disabled and the Algorithm of the test performs a pattern compare, this field counts the pattern mismatches.
18h	4	Correctable Error Count: Total number of correctable memory errors that the device detected during the test. The output of the test will not change Corrected Error Counts tracked by the device and output by the Get Health Info command. This field is invalid and is equal to 0 if the ECC is disabled during the test.
1Ch	4	Reserved

Media Test Results Long Log reports the same fields defined for the Short version. It also includes the capacity tested by the device and the error signature, which consists of:

- Test iteration in which the error occurred
- Failed DPA with a flag that indicates the error type (i.e., uncorrectable or correctable)
- Memory component address following the format defined in the DRAM Event Record

When using the Media Test Results Long Log, two reporting options are available:

- **Complete:** Count and report all the error signatures (with and without threshold programmed)
- **Single error signature:**
 - If error count threshold is set, only the signature of the first error after the threshold is exceeded is reported
 - If error count threshold is not set, only the signature of the first error encountered is reported

The device tracks the error information in the Error Signature Lists. The total number of Error Signatures cannot exceed the value indicated by the Total Number of Error Signatures field in the Media Test Capability Log (see [Table 8-79](#)). If the Error Signature Configuration bit (see [Table 8-108](#)) is set and the total number of errors exceeds the Total Number of Error Signatures, bit[1] in the Flags field shall be set and the test execution shall be interrupted by the device. The test may be resumed from the point at which it was interrupted due to the lack of resources to log the error signatures.

Table 8-85. Media Test Results Long Log

Byte Offset	Length in Bytes	Description
00h	10h	Common Header: Common output information from test execution.
10h	variable ¹	Test 1 Media Test Results Long Log Entry: Contains the results of the first executed media test in long format.
variable	variable	Test 2 Media Test Results Long Log Entry: Contains the results of the second executed media test in long format.
...
variable	variable	Test n Media Test Results Long Log Entry: Contains the results of the nth configured media test in long format.

1. Each entry size depends on the number of error signatures logged by the device for each test executed.

Media Test Results Long Log Entries contain the results of the test executed (described in [Table 8-87](#)). They are preceded by a common header described in [Table 8-86](#). The error signatures are described in [Table 8-88](#); the number of error signatures logged for each test executed is reported by the field at byte offset 28h of the Media Test Results Long Log Entry Structure.

Table 8-86. Media Test Results Long Log Entry Common Header

Byte Offset	Length in Bytes	Description
00h	1	Number of Tests Executed: Total number of tests executed by the device. This number can be smaller than the total number of tests that were requested.
01h	1	Version: Version of the Media Test Result Long Log. This field shall be set to 1. The version is incremented whenever the Media Test Results Long Log is extended to add more functionalities. Backward compatibility shall be maintained during this process. For all values of n, version n+1 may extend version n by replacing fields that are marked as reserved in version n or appending new fields but must not redefine the meaning of existing fields.
02h	Eh	Reserved

Table 8-87. Media Test Results Long Log Entry Structure (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	2	Test ID: ID of the test.
02h	8	Start Time: Expressed as timestamp, it represents the number of unsigned nanoseconds that have elapsed since midnight, 01-Jan-1970, UTC.
0Ah	8	End Time: Expressed as timestamp, it represents the number of unsigned nanoseconds that have elapsed since midnight, 01-Jan-1970, UTC.
12h	1	Result: This field indicates the outcome of the test execution. <ul style="list-style-type: none"> • 00h = Completed with success • 01h = Completed with failure • 02h = Aborted by a Request Abort Background Operation command • All other encodings are reserved
13h	1	Flags: Bit values indicating conditions occurred during the test execution. <ul style="list-style-type: none"> • Bit[0]: Error Signature List Overflow • Bits[7:1]: Reserved
14h	4	Uncorrectable Error Count: Total number of uncorrectable memory errors that the device detected during the test configured by the host.

Table 8-87. Media Test Results Long Log Entry Structure (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
18h	4	Correctable Error Count: Total number of correctable memory errors that the device detected during the test configured by the host.
1Ch	4	Reserved
20h	8	Capacity Tested: This field indicates the capacity tested of the device. Expressed in multiples of 256 MB.
28h	4	Number of Error Signatures: Total number of error signatures reported by the device in the test.
2Ch	4	Reserved
30h	20h	Error Signature 1
50h	20h	Error Signature 2
...
30h+(N*32)h	20h	Error Signature N (N is the Number of error signatures)

Table 8-88. Error Signature (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	2	Iteration: This field indicates the test iteration in which the error occurred.
02h	8	<p>Physical Address: The physical address at which the error occurred during the test execution.</p> <ul style="list-style-type: none"> • Bit[0]: Volatile: <ul style="list-style-type: none"> — 0 = DPA is within the persistent memory range — 1 = DPA field is within the volatile memory range • Bits[2:1]: Error Type: <ul style="list-style-type: none"> — 00b = Uncorrectable — 01b = Correctable — All other encodings are reserved • Bit[3]: Inverse Pattern: This field is valid if the device supports the use of the inverse pattern (see Capability field in Media Test Capability Log Entry Structure, Table 8-81) and the test was configured to use the inverse pattern (through the Inverse field described in Table 8-109) <ul style="list-style-type: none"> — 1 = Error occurred on the test that used the inverse pattern • Bits[5:4]: Reserved • Bits[63:6]: DPA

Table 8-88. Error Signature (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
0Ah	2	<p>Validity Flags: Indicators of which fields are valid within the returned data.</p> <ul style="list-style-type: none"> • Bit[0]: — 1 = Channel field is valid • Bit[1]: — 1 = Rank field is valid • Bit[2]: — 1 = Nibble Mask field is valid • Bit[3]: — 1 = Bank Group field is valid • Bit[4]: — 1 = Bank field is valid • Bit[5]: — 1 = Row field is valid • Bit[6]: — 1 = Column field is valid • Bit[7]: — 1 = Correction Mask field is valid • Bit[8]: — 1 = Component Identifier field is valid • Bit[9]: When bit[8] and bit[9] are both set, the format of the Component Identifier field is governed by Table 8-44 • Bit[10]: — 1 = Sub-channel field is valid • Bits[15:11]: Reserved
0Ch	1	Channel: The channel of the memory error location. A channel is defined as an interface that can be independently accessed for a transaction. The CXL device may support one or more channels.
0Dh	1	Rank: The rank of the memory error location. A rank is defined as a set of memory devices on a channel that together execute a transaction. Multiple ranks may share a channel.
0Eh	3	Nibble Mask: Identifies one or more nibbles in error on the memory bus producing the event. Nibble Mask bit 0 shall be set if nibble 0 on the memory bus produced the event, etc. This field should be valid for corrected memory errors.
11h	1	Bank Group: The bank group of the memory error location.
12h	1	Bank: The bank number of the memory error location.
13h	3	Row: The row number of the memory error location.
16h	2	Column: The column number of the memory error location.
18h	20h	<p>Correction Mask: Identifies the bits in error within that nibble in error on the memory bus that produced the error. The lowest nibble in error in the Nibble Mask uses Correction Mask 0, the next-lowest nibble uses Correction Mask 1, etc. Burst position 0 uses Correction Mask nibble 0, etc. Four correction masks allow for up to 4 nibbles in error. This field should be valid for corrected memory errors.</p> <ul style="list-style-type: none"> • Offset 18h: Correction Mask 0 (8 bytes) • Offset 20h: Correction Mask 1 (8 bytes) • Offset 28h: Correction Mask 2 (8 bytes) • Offset 30h: Correction Mask 3 (8 bytes)
38h	10h	<p>Component Identifier: Device-specific component identifier. This may describe a field-replaceable sub-component of the device.</p> <p>This field is valid when bit[8] of the Validity Flags field is set. When bit[9] of the Validity Flags field is also set, the component identifier is defined by Table 8-44; otherwise, the component identifier format is device specific.</p>
48h	1	Sub-channel: The sub-channel of the memory event location
49h	7	Reserved

8.2.9.5.3 Get Log Capabilities (Opcode 0402h)

Gets capabilities related to the specified log. If the component supports this command, it shall be implemented for all Log Identifier UUIDs that the component supports. This command shall return Invalid Log if the specified Log Identifier is not supported by the component.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Invalid Log

Command Effects:

- None

Table 8-89. Get Log Capabilities Input Payload

Byte Offset	Length in Bytes	Description
00h	10h	Log Identifier: UUID representing the log for which to get capabilities.

Table 8-90. Get Log Capabilities Output Payload

Byte Offset	Length in Bytes	Description
00h	4	Parameter Flags <ul style="list-style-type: none"> • Bit[0]: Clear Log Supported: This bit is set to 1 if the log supports being cleared via the Clear Log command. • Bit[1]: Populate Log Supported: This bit is set to 1 if the log supports being populated via the Populate Log command. • Bit[2]: Auto Populate Supported: This bit is set to 1 if the log supports the ability of being auto populated. Details on the meaning of this bit are specific to each Log Identifier UUID. • Bit[3]: Persistent across Cold Reset: This bit is set to 1 if the log is persistent across Cold Reset. • Bits[31:4]: Reserved.

8.2.9.5.4 Clear Log (Opcode 0403h)

Clears the contents of the specified log.

This command shall return Invalid Log if the specified Log Identifier is not supported by the component.

This command shall return Invalid Input if the specified Log Identifier does not have the Clear Log Supported bit set to 1 in the Get Log Capabilities Output Payload.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Invalid Log

Command Effects:

- Immediate Log Change

Table 8-91. Clear Log Input Payload

Byte Offset	Length in Bytes	Description
00h	10h	Log Identifier: UUID representing the log to clear.

8.2.9.5.5 Populate Log (Opcode 0404h)

Populates the contents of the specified log.

This may be a background operation. If the component implements this command as a background operation for any supported Log Identifier, the Background Operation bit in the Command Effects Log entry for Populate Log shall be set to 1.

This command shall return Invalid Log if the specified Log Identifier is not supported by the component.

This command shall return Invalid Input if the specified Log Identifier does not have the Populate Log Supported bit set to 1 in the Get Log Capabilities Output Payload.

Possible Command Return Codes:

- Success
- Background Command Started
- Unsupported
- Invalid Input
- Internal Error
- Invalid Log
- Interrupted
- Busy
- Aborted

Command Effects:

- Immediate Log Change
- Background Operation (if the component implements this command as a background operation for any supported Log Identifier)
- (if the component implements this command as a background operation for any supported Log Identifier and also supports Request Abort Background Operation Command)

Table 8-92. Populate Log Input Payload

Byte Offset	Length in Bytes	Description
00h	10h	Log Identifier: UUID representing the log to populate.

8.2.9.5.6 Get Supported Logs Sub-List (Opcode 0405h)

Retrieve a sub-list of device-specific log identifiers (each identified by a UUID) and the maximum capacity of each log. This command can retrieve a maximum of 255 log entries. The output of this command shall be consistent with the output of the Get Supported Logs command.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-93. Get Supported Logs Sub-List Input Payload

Byte Offset	Length in Bytes	Description
00h	1	Maximum Number of Supported Log Entries: The maximum number of Supported Log Entries requested. This field shall have a minimum value of 01h.
01h	1	Start Log Entry Index: Index of the first requested Supported Log Entry.

Table 8-94. Get Supported Logs Sub-List Output Payload

Byte Offset	Length in Bytes	Description
00h	1	Number of Supported Log Entries: The number of Supported Log Entries returned in the output payload.
01h	1	Reserved
02h	2	Total Number of Supported Log Entries: The total number of Supported Log Entries supported by the component.
04h	1	Start Log Entry Index: Index of the first Supported Log Entry in the output payload.
05h	3	Reserved
08h	Varies	Supported Log Entries: Device-specific list of supported log identifier UUIDs and the maximum capacity of each log, as defined in Table 8-71 .

8.2.9.6 Features

A Feature is a configuration, control or capability whose setting(s) can be retrieved using Get Feature and optionally modified using Set Feature. Get Feature is used for reporting the values of the associated setting(s). The scope of a Feature is feature-specific and shall be described as part of each Feature's definition. The scope of the Feature may be at the CXL device, LD, Fabric Manager device, or a combination of all these levels.

If a Feature supports changeable attributes that are optional for an implementation, the Set Feature payload describes all changeable attributes and a field that specifies the attribute(s) to update. Any dependencies between different attributes shall be defined by the Feature specification.

If a Feature is supported on the secondary mailbox, the secondary mailbox shall return identical Set Feature Effects value as the primary mailbox for the Feature's Get Supported Features Supported Feature Entry.

Features may evolve by defining new fields in the payload definitions that were originally defined as reserved or by appending new fields.

8.2.9.6.1 Get Supported Features (Opcode 0500h)

Retrieve the list of supported device-specific features (identified by UUID) and general information about each Feature. The device shall return Invalid Input if the Starting Feature Index value is greater than the Device Supported Features value.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-95. Get Supported Features Input Payload

Byte Offset	Length in Bytes	Description
00h	4	Count: Count in bytes of the supported Feature data to return in the output payload. The device shall return no more bytes than requested, but it can return less bytes.
04h	2	Starting Feature Index: Index of the first requested Supported Feature Entry. Feature index is a zero-based value.
06h	2	Reserved

Table 8-96. Get Supported Features Output Payload

Byte Offset	Length in Bytes	Description
00h	2	Number of Supported Feature Entries: The number of Supported Feature Entries returned in the output payload.
02h	2	Device Supported Features: The number of supported Features.
04h	4	Reserved
08h	Varies	Supported Feature Entries: Device-specific list of supported feature identifier UUIDs and general information about each feature (see Table 8-97). For each Feature Entry, the device shall return all the bytes defined in Table 8-97 .

Table 8-97. Get Supported Features Supported Feature Entry (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	10h	Feature Identifier: UUID that represents the feature for which to retrieve data.
10h	2	Feature Index: A zero-based value that is used to uniquely identify the feature. The Feature Index shall be less than the Device Supported Features value.
12h	2	Get Feature Size: The maximum number of bytes that are required to retrieve this Feature data through the Get Feature command(s).
14h	2	Set Feature Size: The maximum number of bytes that are required to update this Feature data through the Set Feature command(s). This field shall have a value of 0 if this Feature cannot be changed.

Table 8-97. Get Supported Features Supported Feature Entry (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
16h	4	Attribute Flags <ul style="list-style-type: none"> Bit[0]: Changeable: If set to 1, the Feature attribute(s) can be changed. Bits[3:1]: Deepest Reset Persistence: Indicates the reset that maintains the current value of Feature attribute(s) (see Table 8-98). <ul style="list-style-type: none"> 000b = None. Any reset will restore the default value. The Saved Selection Supported bit shall be cleared to 0 for this value. 001b = CXL reset. 010b = Hot reset. 011b = Warm reset. 100b = Cold reset. All other encodings are reserved. Bit[4]: Persist across Firmware Update: If set to 1, the current value of Feature attribute(s) persist across a firmware update. Bit[5]: Default Selection Supported: If set to 1, the Feature supports the Default selection value in the Get Feature command. Bit[6]: Saved Selection Supported: If set to 1, the Feature supports the Saved selection value in the Get Feature command. Bits[31:7]: Reserved.
1Ah	1	Get Feature Version : Feature version. The Feature version is incremented whenever the format of the Get Feature output payload data is changed. All changes made to the format of the Get Feature output payload data shall be backward compatible with the previous version.
1Bh	1	Set Feature Version : Feature version. The Feature version is incremented whenever the format of the Feature data supported in the Set Feature command is changed. All changes made to the format of the Feature data in the Set Feature command shall be backward compatible with the previous version. This field shall have a value of 0 if the Feature cannot be changed.
1Ch	2	Set Feature Effects : Bitmask that contains one or more effects for the Set Feature. See the Command Effect field of the CEL Entry Structure in Table 8-75 for the definition of the bitmask. This field shall have a value of 0 if the Feature cannot be changed.
1Eh	18	Reserved

Table 8-98. Feature Attribute(s) Value after Reset

Reset Event	Reset Persistence ^{1 2 3}				
	0h: None	1h: CXL Reset	2h: Hot Reset	3h: Warm Reset	4h: Cold Reset
CXL Reset	Default Value	Saved Value	Current Value	Current Value	Current Value
Hot Reset	Default Value	Saved Value	Saved Value	Current Value	Current Value
Warm Reset	Default Value	Saved Value	Saved Value	Saved Value	Current Value
Cold Reset	Default Value	Saved Value	Saved Value	Saved Value	Saved Value

- Default Value**: The value set by the vendor when the device is shipped and cannot be changed by the host. If Saved Selection supported flag is 0, the Default Value is the Feature Current Value after reset.
- Current Value**: The current value of Feature attribute(s). If some of Feature attributes are writable, the value used by the device is the current attribute value which may be different than the Default Value or the Saved Value.
- Saved Value**: The value set after reset when Saved Selection Supported is 1. Saved Value shall be equal to Default Value when the device is shipped.

8.2.9.6.2 Get Feature (Opcode 0501h)

Retrieve the attributes of the Feature identified by a specific UUID. The caller discovers the size of the Feature first using the Get Supported Features command. The Get Feature command returns the bytes specified in the input payload by the Count payload

field and starting from the Offset payload field. The Device shall return Invalid Input if the Offset payload field is beyond the maximum size of the Feature as reported by Get Supported Features. If the Offset is less than the maximum size of the Feature and the sum of Offset and Count is greater than the maximum size of the Feature, the Device shall return the data from Offset to the maximum size of the Feature.

Possible Command Return Codes:

- Success
- Unsupported
- Unsupported Feature Selection Value
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-99. Get Feature Input Payload

Byte Offset	Length in Bytes	Description
00h	10h	Feature Identifier: UUID representing the Feature identifier for which data is being retrieved.
10h	2	Offset: The offset of the first byte in the Feature data to return in the output payload.
12h	2	Count: Count in bytes of the Feature data to return in the output payload.
14h	1	Selection: Specifies which value of the Feature to return in the output payload. <ul style="list-style-type: none"> • 0h = Current value. The current value of Feature attribute(s). If some of Feature attributes are writable, the current attribute value may be different than the Default or Saved value. • 1h = Default value. The default attribute value for the Feature at device manufacture time. Support for this selection is feature-specific. • 2h = Saved value. The last saved value of the Feature. This is the value of the last Set Feature with the Saved across Reset bit set. Support for this selection is feature-specific. • All other encodings are reserved.

Table 8-100. Get Feature Output Payload

Byte Offset	Length in Bytes	Description
00h	Varies	Feature Data

8.2.9.6.3 Set Feature (Opcode 0502h)

Update the attribute(s) of the Feature identified by a specific UUID. The caller may retrieve the Set Feature Size of the Feature by using the Get Supported Features command. One or more Set Feature commands may be required to transfer all the Feature data, incrementing the Feature Offset each time. The Device shall return Invalid Input if the Offset attempts to access beyond the Set Feature Size of the Feature as reported by Get Supported Features or the sum of Offset and Feature Data size exceeds the Set Feature Size of the Feature as reported by Get Supported Features.

If the Feature data is transferred in its entirety, the caller makes one call to Set Feature with Action = Full Data Transfer. The Offset field is not used and shall be ignored.

If a Feature data is transferred in parts, the caller makes one call to Set Feature with Action = Initiate Data Transfer, zero or more calls with Action = Continue Data Transfer, and one call with Action = Finish Data Transfer or Abort Data Transfer. The Feature data parts shall be transferred in ascending order based on the Offset value, and the Device shall return the Feature Transfer Out of Order return code if data parts are not transferred in ascending order. Back-to-back retransmission of any Set Feature data is permitted during a transfer. The Saved across Reset flag is valid for Set Feature command with Action = Initiate Data Transfer or Action = Full Data Transfer and shall be ignored for all other Action values. A Set Feature with Action = Abort Data Transfer shall be supported for Feature data that can be transferred using multiple Set Feature commands. An attempt to call Set Feature with Action = Abort Data Transfer for a Feature whose data has been fully transferred shall fail with Invalid Input.

Only one Feature may be updated at a time in the device. The device shall return the Feature Transfer in Progress return code if it receives a Set Feature command with Action = Full Data Transfer or Action = Initiate Data Transfer until the current Feature data transfer is completed or aborted.

If the Feature data transfer is interrupted by a Conventional or CXL reset, the Feature data transfer shall be aborted by the device. If a Feature data transfer is aborted prior to the entire Feature data being transferred, the device shall require the Feature data transfer to be started from the beginning of the Feature data.

Once the entire Feature data is fully transferred to the device (i.e., Action = Full Data Transfer or Action = Finish Data Transfer), the device shall update the attribute(s) of the Feature.

The Command Effects Log (CEL) entry for Set Feature shall describe all possible command effects (i.e., Bits 0 to 5) from supported Features that are changeable.

If a component receives an input payload that is less than the size of the structure it has implemented, but is greater than or equal to the Minimum Feature Data Size (as specified in the Feature definition), then it shall treat the unused portion of the structure as 0. For each feature, any fields in the feature data that are not included in the calculation of the Minimum Feature Data Size are explicitly identified. For features where no fields are identified, all the fields in the feature data are to be included in the calculation of the Minimum Feature Data Size.

In support of confidential computing, if the device has been locked while utilizing secure CXL TSP interfaces, the device shall reject any attempt to alter the features of the locked device by returning Invalid Security State status for this command. See [Section 11.5](#) for details on locking a device and locked device behavior.

Possible Command Return Codes:

- Success
- Unsupported
- Unsupported Feature Version
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Feature Transfer in Progress
- Feature Transfer Out of Order

- Invalid Security State

Command Effects:

- Configuration Change after Cold Reset
- Configuration Change after Conventional Reset
- Configuration Change after CXL Reset
- Immediate Configuration Change
- Immediate Data Change
- Immediate Policy Change
- Immediate Log Change
- Security State Change

Table 8-101. Set Feature Input Payload

Byte Offset	Length in Bytes	Description
00h	10h	Feature Identifier: UUID representing the Feature identifier for which data is being updated. The UUID value of all Fs is a special value that represents the current Feature whose data is in the process of being transferred.
10h	4	Set Feature Flags <ul style="list-style-type: none"> • Bits[2:0]: Action: Specifies the stage of Feature data transfer <ul style="list-style-type: none"> — 000b = Full Data Transfer — 001b = Initiate Data Transfer — 010b = Continue Data Transfer — 011b = Finish Data Transfer — 100b = Abort Data Transfer — All other encodings are reserved • Bit[3]: Saved across Reset: If set to 1, the modified value is saved across the Deepest Reset Persistence value for the Feature • Bits[31:4]: Reserved
14h	2	Offset: The byte offset of the Feature data to update.
16h	1	Version: Feature version of the data in Feature Data.
17h	9	Reserved
20h	Varies	Feature Data

8.2.9.7 Maintenance

8.2.9.7.1 Perform Maintenance (Opcode 0600h)

This command requests the device to execute the maintenance operation specified by the Maintenance Operation Class and the Maintenance Operation Subclass. If the operation is not supported, the command shall be terminated with Invalid Input Return Code.

The Perform Maintenance command may be performed in the foreground or in the background, based on the characteristics of the maintenance operation. When the device is executing a Perform Maintenance command in the background, it may indicate operation progress using the Background Command Status register.

No more than one maintenance operation may be initiated at a time.

The device shall terminate a foreground or background Perform Maintenance command with Busy Return Code if it is already processing a maintenance operation in the background.

Some restrictions may apply during the execution of a maintenance operation. For example, it might not be possible to read or write a CXL memory device. These restrictions are specified in the description of the maintenance operation and there can be Feature attributes that indicate device capabilities.

In support of confidential computing, if the device has been locked while utilizing secure CXL TSP interfaces, the device shall reject any attempt to perform maintenance PPR, sPPR, hPPR, built-in self-tests, and/or other maintenance operations that might alter the data and/or TE State on the device, affect the devices ability to maintain data coherency, and/or compromise the link's integrity by returning Invalid Security State status for this command. See [Section 11.5](#) for details on locking a device and locked device behavior.

Possible Command Return Codes:

- Success
- Unsupported
- Background Command Started
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Busy
- Transfer Out of Order
- Aborted
- Invalid Physical Address
- Resources Exhausted
- Invalid Security State

Command Effects:

- Immediate Configuration Change if a maintenance operation restricts the operations that a host can do
- Immediate Data Change if a maintenance operation impacts the data written to the device
- Immediate Log Change if a maintenance operation impacts a device log
- Background Operation if a maintenance operation is executed in background
- Request Abort Background Operation Command Supported

[Table 8-102](#) shows the input payload for this command. There is no output payload.

Table 8-102. Perform Maintenance Input Payload

Byte Offset	Length in Bytes	Description
00h	1	Maintenance Operation Class: This field identifies the Class of a maintenance operation. See Table 8-110 .
01h	1	Maintenance Operation Subclass: This field identifies the maintenance operation together with the Maintenance Operation Class. See Table 8-110 .
02h	Varies	Maintenance operation parameters.

8.2.9.7.1.1 PPR Maintenance Operations

Post Package Repair (PPR) maintenance operations may be supported by CXL devices that implement CXL.mem protocol. A PPR maintenance operation requests the CXL device to perform a repair operation on its media.

For example, a CXL device with DRAM components that support PPR features may implement PPR Maintenance operations. DRAM components may support two types of PPR: Hard PPR (hPPR), for a permanent row repair, and Soft PPR (sPPR), for a temporary row repair. sPPR is much faster than hPPR, but the repair is lost with a power cycle.

Based on DRAM PPR features, two maintenance operations are defined: sPPR and hPPR. Note that PPR maintenance operations may also apply to other types of media component.

During the execution of a PPR Maintenance operation, a CXL memory device:

- May or may not retain data
- May or may not be able to process CXL.mem requests correctly, including the ones that target the DPA involved in the repair

If the device is not capable of correctly processing a CXL.mem request during a PPR Maintenance operation, then:

- A read shall return poison
- A write shall be dropped, and an NDR shall be sent
- Any subsequent reads of DPA for which writes may have been incorrectly processed shall return poison

These CXL Memory Device capabilities are specified by Restriction Flags in the sPPR Feature and hPPR Feature (see [Section 8.2.9.7.2.1](#) and [Section 8.2.9.7.2.2](#), respectively).

sPPR maintenance operation may be executed at runtime, if data is retained and CXL.mem requests are correctly processed. For CXL devices with DRAM components, hPPR maintenance operation may be executed only at boot because data would not be retained.

When a CXL device identifies a failure on a memory component, the device may inform the host about the need for a PPR maintenance operation by using an Event Record, where the Maintenance Needed flag is set. The Event Record specifies the DPA that should be repaired. A CXL device may not keep track of the requests that have already been sent and the information on which DPA should be repaired may be lost upon power cycle.

The Host or the FM may or may not initiate a PPR Maintenance operation in response to a device request. It is possible to check whether resources are available by issuing a Perform Maintenance command for the PPR maintenance operation with the Query Resources flag set to 1. If the controller does not support reporting whether a resource is available, and a Perform Maintenance operation for PPR is issued with Query Resources set to 1, the controller shall return Invalid Input.

If resources are available, then the command shall be completed with the Success Return Code; otherwise, the command shall be completed with the Resources exhausted Return Code.

The host or the FM may initiate a repair operation by issuing a Perform Maintenance command, setting the Maintenance Operation Class to 01h (PPR), the Maintenance Operation Subclass to either 00h (sPPR) or 01h (hPPR), and indicating the DPA (if supported).

During the execution of a PPR maintenance operation, the device operation may be restricted as indicated by the Restriction Flags in the sPPR Feature and hPPR Feature (see [Section 8.2.9.7.2.1](#) and [Section 8.2.9.7.2.2](#), respectively).

Upon completion of a PPR maintenance operation, the device shall produce a Memory Sparing Event Record with updated resource availability, if the Memory Sparing Event Record Enable bit is set (see [Table 8-113](#) or [Table 8-116](#)).

8.2.9.7.1.2 sPPR Maintenance Operation

This maintenance operation requests the device to perform an sPPR operation. The sPPR Feature provides parameters and configurations related to this operation. See [Section 8.2.9.7.2.1](#). [Table 8-103](#) shows the input payload for this maintenance operation.

Table 8-103. sPPR Maintenance Input Payload

Byte Offset	Length in Bytes	Description
00h	1	Maintenance Operation Class: It shall be set to 01h.
01h	1	Maintenance Operation Subclass: It shall be cleared to 00h.
02h	1	Flags <ul style="list-style-type: none"> Bit[0]: Query Resources Flag: If set, the CXL device checks whether resources are available to perform the sPPR maintenance operation but does not attempt to perform the operation Bits[7:1]: Reserved
03h	8	DPA: Physical address to be repaired. This field is ignored if the DPA support flag in the sPPR Feature is cleared to 0 (see Table 8-113).
0Bh	3	Nibble Mask: Identifies one or more nibbles on the memory bus. Nibble mapping is the same of DRAM Event Record nibble mapping, see Table 8-46 . A Nibble Mask bit set to 1 indicates the request to perform sPPR operation in the specific device. All Nibble Mask bits set to 1 indicates the request to perform the operation in all devices. This field is ignored if the Nibble support flag in the sPPR Feature is cleared to 0 (see Table 8-113), and the sPPR is performed in all devices.

8.2.9.7.1.3 hPPR Maintenance Operation

This maintenance operation requests the device to perform an hPPR operation. The hPPR Feature provides parameters and configurations related to this operation, see [Section 8.2.9.7.2.2](#). [Table 8-104](#) shows the input payload for this maintenance operation.

Table 8-104. hPPR Maintenance Input Payload (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	1	Maintenance Operation Class: It shall be set to 01h.
01h	1	Maintenance Operation Subclass: It shall be set to 01h.

Table 8-104. hPPR Maintenance Input Payload (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
02h	1	Flags <ul style="list-style-type: none"> Bit[0]: Query Resources Flag: If set, the CXL device checks whether resources are available to perform the hPPR maintenance operation Bits[7:1]: Reserved
03h	8	DPA : Physical address to be repaired. This field is ignored if the DPA support flag bit in the hPPR Feature is cleared to 0 (see Table 8-116).
0Bh	3	Nibble Mask : Identifies one or more nibbles on the memory bus. Nibble mapping is the same of DRAM Event Record nibble mapping, see Table 8-46 . A Nibble Mask bit set to 1 indicates the request to perform hPPR operation in the specific device. All Nibble Mask bits set to 1 indicates the request to perform the operation in all devices. This field is ignored if the Nibble support flag in the hPPR Feature is cleared to 0 (see Table 8-116), and the hPPR is performed in all devices.

8.2.9.7.1.4 Memory Sparing Maintenance Operations

The associated Class and Subclasses are defined in [Table 8-110](#).

Memory sparing is defined as a repair function that replaces a portion of memory (the “spared memory”) with a portion of functional memory at that same DPA. The Subclasses for this operation vary in terms of the scope of the sparing being performed. The Cacheline sparing subclass refers to a sparing action that can replace a full cacheline. Row sparing is provided as an alternative to PPR sparing functions and its scope is that of a single DDR row. Bank sparing allows an entire bank to be replaced. Rank sparing is defined as an operation in which an entire DDR rank is replaced.

The Input Payload specifies the memory portion to be replaced. In particular, the Nibble Mask field in the Input Payload may be used to request sparing on specific components. The nibble mapping is the same as DRAM Event Record nibble mapping (see [Table 8-46](#)). Components are specified by setting the Nibble Mask Valid flag and the related Nibble Mask bits. The device may apply memory sparing to more components than requested. If the Nibble Mask Valid flag is 0, the memory sparing request is for all components.

If the host requests an operation Subclass for an address and the device is out of resources, the device shall respond with the Resources Exhausted return code.

The host may issue a query command by setting Query Resources flag in the Input Payload (see [Table 8-105](#)) to determine availability of sparing resources for a given address. In response to a query request, the device shall report the resource availability by producing the Memory Sparing Event Record (see [Table 8-48](#)) in which the Channel, Rank, Nibble Mask, Bank Group, Bank, Row, Column, Sub-Channel fields are a copy of the values specified in the request. If the controller does not support reporting whether a resource is available, and a Perform Maintenance operation for Memory Sparing is issued with Query Resources set to 1, the controller shall return Invalid Input.

All Memory Sparing operations shall be executed as background operations and are capable of being aborted by the Request Abort Background Operation command.

[Table 8-105](#) shows the input payload for this maintenance operation. The device shall communicate the operation’s results by producing a Memory Sparing Event Record (see [Table 8-48](#)) in response to the request.

Table 8-105. Memory Sparing Input Payload

Byte Offset	Length in Bytes	Description
00h	1	Maintenance Operation Class: It shall be set to 02h.
01h	1	Maintenance Operation Subclass: The legal values are defined in Table 8-110 .
02h	1	Flags <ul style="list-style-type: none"> Bit[0]: Query Resources Flag: If set, the CXL device checks whether resources are available to perform the maintenance operation. Bit[1]: Hard Sparing Flag: If set, the Host is requesting of the device that the sparing action be permanent. The device is permitted to perform a soft sparing action if hard sparing resources are unavailable for this subclass. Bit[2]: Sub-channel Valid Flag: If set, the Sub-channel field is valid; otherwise, that field shall be ignored. Bit[3]: Nibble Mask Valid Flag: If set, the Nibble Mask field is valid; otherwise, that field shall be ignored, and the sparing request applies to all components. Bits[7:4]: Reserved.
03h	1	Channel: The Channel field associated with the DPA that is to be spared. See Table 8-46 for definition. This field applies to all subclasses.
04h	1	Rank: The rank field associated with the DPA that is to be spared. See Table 8-46 for definition. This field applies to all subclasses.
05h	3	Nibble Mask: This field may be used to specify one or more components to be spared. See Table 8-46 for definition. It may be provided for all subclasses.
08h	1	Bank Group: The bank group associated with the DPA that is to be spared. See Table 8-46 for definition. This field applies to Cacheline Sparing subclass [00h], Row Sparing subclass [01h], and Bank Sparing subclass [02h]. It is reserved for all other subclasses.
09h	1	Bank: The bank associated with the DPA that is to be spared. See Table 8-46 for definition. This field applies to Cacheline Sparing subclass [00h], Row Sparing subclass [01h], and Bank Sparing subclass [02h]. It is reserved for all other subclasses.
0Ah	3	Row: The rank associated with the DPA that is to be spared. See Table 8-46 for definition. This field applies to Cacheline Sparing subclass [00h] and Row Sparing subclass [01h]. It is reserved for all other subclasses.
0Dh	2	Column: The column associated with the DPA that is to be spared. See Table 8-46 for definition. This field applies to Cacheline Sparing subclass [00h]. It is reserved for all other subclasses.
0Fh	1	Sub-channel: The sub-channel associated with the DPA that is to be spared. See Table 8-46 for definition. If the device implements sub-channels, this field applies to Cacheline Sparing subclass [00h]. It is reserved for all other subclasses.

8.2.9.7.1.5 Device Built-in Test Operations

This function is used to request a CXL.mem-capable device to execute one or more tests.

Media Test Subclass requires the device to execute one or more tests on the memory media. Media Tests that a device supports and their attributes can be discovered by getting the Media Test Capability Log (see [Section 8.2.9.5.2.5](#)). The host may discover the media tests that the device supports and then request the device to perform a single test or a list of tests from the supported tests. The test results can be retrieved accessing the Media Test Results Log (see [Section 8.2.9.5.2.6](#)). It is expected that CXL.mem traffic to the device is quiesced when a media test is started on the device. If CXL.mem requests are issued during tests execution, the device behavior is undefined. For configuring and launching the Media Test operation, Perform Maintenance Command shall have the Input Payload described in [Table 8-106](#).

One or more tests that belong to the same subclass may be requested via a single command (see [Table 8-107](#)). For each requested test, a single Test Parameters Entry shall be set up. Because multiple tests may be scheduled via a single command, the

Test Parameters length is variable. The Test Parameters may be fully transferred in a single chunk or transferred in multiple chunks by issuing multiple Perform Maintenance commands. If the Test Parameters are transferred in its entirety, the caller issues a single Perform Maintenance Command with Action = Full Transfer. If the Test Parameters are transferred in parts, the caller makes one call to Perform Maintenance with Action = Initiate Transfer, zero or more calls with Action = Continue Transfer, and one call with Action = End Transfer or Abort Transfer. The Test Parameters parts shall be transferred in order; otherwise, the device returns the Transfer Out of Order return code.

If the test is executing in background, the device may be asked to abort the test via the Request Abort Background Operation mailbox command. If a component supports Perform Maintenance Operation with this class, it must also support the Request Abort Background Operation command.

Table 8-106. Device Built-in Test Input Payload

Byte Offset	Length in Bytes	Description
00h	1	Maintenance Operation Class: It shall be set to 03h.
01h	1	Maintenance Operation Subclass <ul style="list-style-type: none"> • 00h = Media Test • 01h to BCh = Reserved • C0h to FFh = Vendor Specific Test
02h	1	Action: Specifies the stage of the Test Parameters transfer. <ul style="list-style-type: none"> • 00h = Full Transfer • 01h = Initiate Transfer • 02h = Continue Transfer • 03h = End Transfer • 04h = Abort Transfer • All other encodings are reserved
03h	4	Offset: The byte offset in the Test Parameters data. Expressed in multiples of 32 bytes. Ignored if Action = Full Transfer.
07h	1	Reserved
08h	20h+20h*n	Test parameters: See Table 8-107 .

Table 8-107. Test Parameters

Byte Offset	Length in Bytes	Description
00h	20h	Common Configuration Parameters: Input configuration parameters that apply to all the tests within a given subclass. The common configuration parameters for Media Test are defined in Table 8-108 . The common configuration parameters for vendor specific test are defined by the vendor.
20h	20h	Test 1 Parameters Entry: Input parameters of Test 1. The format of the Test Parameter Entry for Media Test is defined in Table 8-109 . The common configuration parameters for vendor specific test are defined by the vendor.
40h	20h	Test 2 Parameters Entry: Input parameters of Test 2.
...
(20h+20h*(n-1))	20h	Test n Parameters Entry: Input parameters of Test n.

Table 8-108. Common Configuration Parameters for Media Test Subclass

Byte Offset	Length in Bytes	Description
00h	1	Number of Tests: Total number of tests requested.
01h	8	Start Address: Start DPA of the test, applies to all the tests. If the requester needs to initiate media tests on different ranges, it shall issue one command per address range.
09h	8	Length: The range of physical addresses to test, applies to all the tests. This length shall be in multiples of 64 bytes.
11h	1	Media Test Results Configuration: Configuration flags applicable to Media Test Results Logs. <ul style="list-style-type: none"> • Bit[0]: Error Signature Configuration: <ul style="list-style-type: none"> — 0 = Complete. All the error signatures are reported. — 1 = Single error signature. If error count threshold is programmed, only the signature of the first error after the threshold is exceeded is reported. If error count threshold is not programmed, only the signature of the first error encountered is reported. • Bits[7:1]: Reserved
12h	1	Configuration Flags: Common configuration flags applicable to all the tests. <ul style="list-style-type: none"> • Bits[1:0]: ECC Disablement: <ul style="list-style-type: none"> — 00b = Data ECC is enabled and metadata ECC is enabled. — 01b = Data ECC is disabled and metadata ECC is enabled. This value is permitted if bit[0] in the Capabilities field of the Media Test Capability Log Common header (see Table 8-80) is set to 1. — 10b = Data ECC is enabled and metadata ECC is disabled. This value is permitted if bit[1] in the Capabilities field of the Media Test Capability Log Common header is set to 1. — 11b = Data ECC is disabled and metadata ECC is disabled. This value is permitted if bit[2] or both bit[0] and bit[1] in the Capabilities field of the Media Test Capability Log Common header are set to 1. • Bits[7:2]: Reserved
13h	Dh	Reserved

Table 8-109. Test Parameters Entry Media Test Subclass (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	2	Test ID: This field identifies the Test. The value discovered through the Media Test Capability Log Entry Structures (see Table 8-81).
02h	1	Number of iterations: Number of repetitions of the test.
03h	2	Flags <ul style="list-style-type: none"> • Bit[0]: Inverse Pattern Enable: If set, the test is repeated after complementing the selected pattern. • Bit[1]: Exit on Uncorrectable Error: The test is stopped on the first uncorrectable error. If bit[2] is set, the test will stop on the first uncorrectable error after exceeding the threshold. If the ECC is disabled, any bit error on the media results in an uncorrectable error. • Bit[2]: Error Count Threshold Programmed: If set, the error count threshold is provided via the Error Count Threshold field at offset 0Eh. • Bit[3]: Reserved. • Bits[7:4]: Addressing Mode: <ul style="list-style-type: none"> — 0h = Ascending — 1h = Descending — 2h = Algorithm Specific — 3h = Random — All other values are reserved • Bit[8]: Update Poison List on Uncorrectable Error: If set, the DPA impacted by uncorrectable error are added to the poison list if the Poison List is supported. • Bits[15:9]: Reserved.

Table 8-109. Test Parameters Entry Media Test Subclass (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
05h	2	Pattern Type: This field selects the 64B pattern applied during the test. <ul style="list-style-type: none"> • 00h = User provided • 01h = Vendor specific • 02h = PRBS • 03h = DPA[63:0] by eight • 04h = 55h • 05h = AAh • All other encodings are reserved
07h	1	Pattern Value: Pattern value provided by the user. This field is reserved if Pattern Type is not 00h.
08h	2	Vendor Specific: This field is set if Pattern Type field is 01h. The interpretation of this field is vendor specific.
0Ah	4	PRBS Seed: User provided PRBS Seed. This field is valid if Pattern Type is PRBS.
0Eh	2	Error Count Threshold: User-programmable error count threshold. If Flags[2] is set, error tracking starts after exceeding the Error Count Threshold.
10h	10h	Reserved

8.2.9.7.2 Features Associated with Maintenance Operations

Maintenance operations leverage the Features command set (see [Section 8.2.9.6](#)).

A Feature that provides capabilities and configurations may be defined for a maintenance operation. The list of maintenance operations that the device supports can be discovered by analyzing the device's supported Features. This can be accomplished by issuing the Get Supported Features command.

[Table 8-110](#) shows the Maintenance Operation Classes, Subclasses, and related Feature UUID.

Table 8-110. Maintenance Operation: Classes, Subclasses, and Feature UUIDs

Maintenance Operation Class		Maintenance Operation Subclass		Feature
Value	Description	Value	Description	UUID
00h	No operation	00h	No operation	-
01h	PPR	00h	Soft PPR	892ba475-fad8-474e-9d3e-692c917568bb
		01h	Hard PPR	80ea4521-786f-4127-afb1-ec7459fb0e24
		Others	Reserved	-
02h	Memory Sparing	00h	Cacheline - Memory Sparing	96C33386-91dd-44c7-9ecb-fdaf6503baC4
		01h	Row - Memory Sparing ¹	450ebf67-b135-4f97-a498-c2d57f279bed
		02h	Bank - Memory Sparing	78b79636-90ac-4b64-A4ef-faac5d18a863
		03h	Rank - Memory Sparing	34dbaff5-0552-4281-8f76-da0b5e7a76a7
		Others	Reserved	-
03h	Device Built-in Test	00h	Media Test	No associated feature
		01h to BFh	Reserved	-
		0C0h to 0FFh	Vendor Specific	Vendor Specific
04h to DFh	Reserved	All	Reserved	-
E0h to FFh	Vendor specific	All	Vendor specific	Vendor specific

- Row sparing in the Memory Sparing is equivalent to PPR; however, memory sparing is preferred when possible.

These Features represent maintenance operations capabilities and settings. Some fields of the Features are writable to configure the desired device behavior. [Table 8-111](#) shows the Maintenance Operation Feature format. The first 16 bytes are common to all Maintenance Operation Features.

Table 8-111. Common Maintenance Operation Feature Format

Byte Offset	Length in Bytes	Attribute	Description
00h	1	RO	Maximum Maintenance Operation Latency: This field indicates the maximum latency of the maintenance operation. <ul style="list-style-type: none"> Bits[3:0]: These bits specify the time scale. <ul style="list-style-type: none"> 0h = 1 us 1h = 10 us 2h = 100 us 3h = 1 ms 4h = 10 ms 5h = 100 ms 6h = 1 s 7h = 10 s All other encodings are reserved. Bits[7:4]: These bits specify the maximum operation latency with the time scale indicated in bits[3:0].
01h	2	RO	Operation Capabilities <ul style="list-style-type: none"> Bit[0]: Device Initiated Capability: A value of 1 indicates that the device has the capability to initiate a maintenance operation without host involvement Bits[15:1]: Reserved
03h	2	RW	Operation Mode <ul style="list-style-type: none"> Bit[0]: Device Initiated: A value of 1 indicates that the device may initiate the maintenance operation without host involvement. If cleared to 0, the device shall initiate the maintenance operation only when receiving an explicit maintenance command. If bit[0] of the Operation Capabilities field returns 0, this bit is considered reserved Bits[15:1]: Reserved
05h	1	RO	Maintenance Operation Class: This field specifies the Maintenance Operation Class. This value is set in Event Records to request this operation and in the Perform Maintenance input payload to initiate this maintenance operation.
06h	1	RO	Maintenance Operation Subclass: This field specifies the Maintenance Operation Subclass. This value is set in the Perform Maintenance input payload to initiate this maintenance operation.
07h	9	RsvdZ	Reserved
10h	Varies	-	Operation specific fields

8.2.9.7.2.1 sPPR Feature Discovery and Configuration

The UUID of this feature is defined in [Table 8-110](#).

[Table 8-112](#) shows the information returned in the Get Supported Features output payload for the sPPR Feature. Some Feature attributes are changeable. Feature attributes cannot be saved.

Table 8-112. Supported Feature Entry for the sPPR Feature

Byte Offset	Length in Bytes	Attribute	Value
00h	10h	Feature Identifier	892ba475-fad8-474e-9d3e-692c917568bb
10h	2	Feature Index	Device specific
12h	2	Get Feature Size	14h
14h	2	Set Feature Size	03h
16h	4	Attribute Flags	<ul style="list-style-type: none"> • Bit[0]: Vendor-specific value (Changeable) • Bits[3:1]: 000b (Deepest Reset Persistence=None. Any reset will restore the default value.) • Bit[4]: 0 (Persist across Firmware Update) • Bit[5]: 1 (Default Selection Supported) • Bit[6]: 0 (Saved Selection Supported) • Bits[31:7]: Reserved
1Ah	1	Get Feature Version	02h
1Bh	1	Set Feature Version	02h
1Ch	2	Set Feature Effects	<ul style="list-style-type: none"> • Bit[0]: 0 (Configuration Change after Cold Reset) • Bit[1]: 1 (Immediate Configuration Change) • Bit[2]: 0 (Immediate Data Change) • Bit[3]: 0 (Immediate Policy Change) • Bit[4]: Vendor-specific value (Immediate Log Change) • Bit[5]: 0 (Security State Change) • Bit[6]: 0 (Background Operation) • Bit[7]: Vendor-specific value (Secondary Mailbox Supported) • Bit[8]: 0 (Request Abort Background Operation Supported) • Bit[9]: 1 is recommended, 0 is permitted (CEL[11:10] Valid) • Bit[10]: 0 (Configuration Change after Conventional Reset) • Bit[11]: 0 (Configuration Change after CXL Reset) • Bits[15:12]: 0h
1Eh	18	Reserved	All 0s

Table 8-113 shows the output payload returned by a Get Feature command with Selection cleared to 0h (Current value) or set to 1h (Default value).

Table 8-113. sPPR Feature Readable Attributes (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	1	Maximum Maintenance Operation Latency: This field is defined in Table 8-111 .
01h	2	Operation Capabilities: This field is defined in Table 8-111 . Device Initiated capability bit shall be cleared to 0 if Restriction Flags bit[0] or bit[2] is set to 1.
03h	2	Operation Mode: This field is defined in Table 8-111 .
05h	1	Maintenance Operation Class: It shall be set to 01h (PPR).
06h	1	Maintenance Operation Subclass: It shall be cleared to 00h (Soft PPR).
07h	9	Reserved

Table 8-113. sPPR Feature Readable Attributes (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
10h	1	Soft PPR Flags <ul style="list-style-type: none"> Bit[0]: DPA Support Flag: If set, the device supports DPA argument in the Perform Maintenance command input payload. Bit[1]: Nibble Support Flag: If set, the device supports Nibble Mask argument in the Perform Maintenance command input payload. Bit[2]: Memory Sparing Event Record Capability Flag: If set, the device has the capability to produce a Memory Sparing Event Record upon completion of sPPR maintenance operation. This bit shall be set if Get Feature Version is 02h or greater. Bits[7:3]: Reserved.
11h	2	Restriction Flags <ul style="list-style-type: none"> Bit[0]: This bit indicates the ability of the device to process CXL.mem requests during sPPR maintenance operation execution. <ul style="list-style-type: none"> 0 = CXL.mem requests are correctly processed. 1 = Media is not accessible. The device shall return poison on read access and drop write access. Bit[1]: Reserved. Bit[2]: This bit indicates the ability of the device to retain data across an sPPR maintenance operation execution. <ul style="list-style-type: none"> 0 = Data is retained. 1 = Data may or may not be retained. Bits[15:3]: Reserved.
13h	1	sPPR Operation Mode <ul style="list-style-type: none"> Bit[0]: Memory Sparing Event Record Enable: If set, the Device shall produce a Memory Sparing Event Record upon completion of sPPR maintenance operation. This bit is reserved if Soft PPR Flags bit[2] is 0. Bits[7:1]: Reserved.

Table 8-114 shows the Feature Data for the Set Feature command.

Table 8-114. sPPR Feature Writable Attributes

Byte Offset	Length in Bytes	Description
00h	2	Operation Mode <ul style="list-style-type: none"> Bit[0]: Device Initiated: A value of 1 indicates that the device may initiate the sPPR maintenance operation without host involvement. If cleared to 0, the device shall initiate the maintenance operation only when receiving an explicit maintenance command. This bit is cleared upon a Cold Reset. Bits[15:1]: Reserved.
02h	1	sPPR Operation Mode¹ <ul style="list-style-type: none"> Bit[0]: Memory Sparing Event Record Enable: If set, the device shall produce a Memory Sparing Event Record upon completion of sPPR maintenance operation. This bit is reserved if Soft PPR Flags bit[2] is 0. Bits[7:1]: Reserved.

1. This field is not included the Minimum Feature Data Size.

8.2.9.7.2.2 hPPR Feature Discovery and Configuration

The UUID of this feature is defined in Table 8-110.

Table 8-115 shows the information returned in the Get Supported Features output payload for the hPPR Feature. Some Feature attributes are changeable. Feature attributes cannot be saved.

Table 8-115. Supported Feature Entry for the hPPR Feature

Byte Offset	Length in Bytes	Attribute	Value
00h	10h	Feature Identifier	80ea4521-786f-4127-afb1-ec7459fb0e24
10h	2	Feature Index	Device specific
12h	2	Get Feature Size	14h
14h	2	Set Feature Size	03h
16h	4	Attribute Flags	<ul style="list-style-type: none"> • Bit[0]: Vendor-specific value (changeable) • Bits[3:1]: 000b (Deepest Reset Persistence=None. Any reset will restore the default value.) • Bit[4]: 0 (Persist across Firmware Update) • Bit[5]: 1 (Default Selection Supported) • Bit[6]: 0 (Saved Selection Supported) • Bits[31:7]: Reserved
1Ah	1	Get Feature Version	02h
1Bh	1	Set Feature Version	02h
1Ch	2	Set Feature Effects	<ul style="list-style-type: none"> • Bit[0]: 0 (Configuration Change after Cold Reset) • Bit[1]: 1 (Immediate Configuration Change) • Bit[2]: 0 (Immediate Data Change) • Bit[3]: 0 (Immediate Policy Change) • Bit[4]: Vendor-specific value (Immediate Log change) • Bit[5]: 0 (Security State Change) • Bit[6]: 0 (Background Operation) • Bit[7]: Vendor-specific value (Secondary Mailbox Supported) • Bit[8]: 0 (Request Abort Background Operation Supported) • Bit[9]: 1 is recommended, 0 is permitted (CEL[11:10] Valid) • Bit[10]: 0 (Configuration Change after Conventional Reset) • Bit[11]: 0 (Configuration Change after CXL Reset) • Bits[15:12]: 0h
1Eh	2	Reserved	All 0s

Table 8-116 shows the output payload returned by a Get Feature command with Selection cleared to 0h (current value) or set to 1h (default value).

Table 8-116. hPPR Feature Readable Attributes (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	1	Maximum Maintenance Operation Latency: This field is defined in Table 8-111 .
01h	2	Operation Capabilities: This field is defined in Table 8-111 . Device Initiated capability bit shall be cleared to 0 if Restriction Flags Bit[0] or Bit[2] is set to 1.
03h	2	Operation Mode: This field is defined in Table 8-111 .
05h	1	Maintenance Operation Class: It shall be set to 01h (PPR).
06h	1	Maintenance Operation Subclass: It shall be set to 01h (Hard PPR).
07h	9	Reserved

Table 8-116. hPPR Feature Readable Attributes (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
10h	1	hPPR Flags <ul style="list-style-type: none"> Bit[0]: DPA Support Flag: If set, the device supports the DPA argument in the Perform Maintenance command input payload. Bit[1]: Nibble Support Flag: If set, the device supports the Nibble Mask argument in the Perform Maintenance command input payload. Bit[2]: Memory Sparing Event Record Capability Flag: If set, the device has the capability to produce a Memory Sparing Event Record upon completion of hPPR maintenance operation. This bit shall be set if Get Feature Version is 02h or greater. Bits[7:3]: Reserved.
11h	2	Restriction Flags <ul style="list-style-type: none"> Bit[0]: This bit indicates the ability of the device to process CXL.mem requests during hPPR maintenance operation execution. <ul style="list-style-type: none"> 0 = CXL.mem requests are correctly processed. 1 = Media is not accessible. The device shall return poison on read access and drop write access. Bit[1]: Reserved. Bit[2]: This bit indicates the ability of the device to retain data across an hPPR maintenance operation execution. <ul style="list-style-type: none"> 0 = Data is retained. 1 = Data may or may not be retained. Bits[15:3]: Reserved.
13	1	hPPR Operation Mode <ul style="list-style-type: none"> Bit[0]: Memory Sparing Event Record Enable: If set, the device shall produce a Memory Sparing Event Record upon completion of hPPR maintenance operation. This bit is reserved if hPPR Flags bit[2] is 0. Bits[7:1]: Reserved.

Table 8-117 shows the Feature Data for Set Feature command.

Table 8-117. hPPR Feature Writable Attributes

Byte Offset	Length in Bytes	Description
00h	2	Operation Mode <ul style="list-style-type: none"> Bit[0]: Device Initiated: A value of 1 indicates that the device may initiate the hPPR maintenance operation host involvement. If cleared to 0, the device shall initiate the maintenance operation only when receiving an explicit maintenance command. <ul style="list-style-type: none"> This bit is cleared upon a Cold Reset Bits[15:1]: Reserved.
02h	1	hPPR Operation Mode¹ <ul style="list-style-type: none"> Bit[0]: Memory Sparing Event Record Enable: If set, the device shall produce a Memory Sparing Event Record upon completion of hPPR maintenance operation. This bit is reserved if hPPR Flags bit[2] is 0. Bits[7:1]: Reserved.

1. This field is not included in the Minimum Feature Data Size.

8.2.9.7.2.3 Memory Sparing Features

The UIDs associated with these features are defined in Table 8-110.

Table 8-118 shows the information returned in the Get Supported Features output payload for the Enhanced Memory Sparing Feature. Some Feature attributes are changeable.

Table 8-118. Supported Feature Entry for the Memory Sparing Feature

Byte Offset	Length in Bytes	Attribute	Value
00h	10h	Feature Identifier	Depends on the Maintenance Operation subclass (see Table 8-110 for details)
10h	2	Feature Index	Device specific
12h	2	Get Feature Size	13h
14h	2	Set Feature Size	02h
16h	4	Attribute Flags	<ul style="list-style-type: none"> • Bit[0]: Vendor-specific value (Changeable) • Bits[3:1]: 000b (Deepest Reset Persistence=None. Any reset will restore the default value.) • Bit[4]: 0 (Persist across Firmware Update) • Bit[5]: 1 (Default Selection Supported) • Bit[6]: Vendor-specific value (Saved Selection Supported) • Bits[31:7]: Reserved
1Ah	1	Get Feature Version	01h
1Bh	1	Set Feature Version	01h
1Ch	2	Set Feature Effects	<ul style="list-style-type: none"> • Bit[0]: 0 (Configuration Change after Cold Reset) • Bit[1]: 1 (Immediate Configuration Change) • Bit[2]: 0 (Immediate Data Change) • Bit[3]: 0 (Immediate Policy Change) • Bit[4]: Vendor-specific value (Immediate Log Change) • Bit[5]: 0 (Security State Change) • Bit[6]: 0 (Background Operation) • Bit[7]: Vendor-specific value (Secondary Mailbox Supported) • Bit[8]: 0 (Request Abort Background Operation Supported) • Bit[9]: 1 (CEL[11:10] Valid) • Bit[10]: 0 (Configuration Change after Conventional Reset) • Bit[11]: 0 (Configuration Change after CXL Reset) • Bits[15:12]: 0h
1Eh	12h	Reserved	All 0s

[Table 8-119](#) shows the output payload returned by a Get Feature command with Selection cleared to 0h (Current value), or set to 1h (Default value) or 2h (Saved Value).

Table 8-119. Memory Sparing Feature Readable Attributes (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	1	Maximum Maintenance Operation Latency: This field is defined in Table 8-111 .
01h	2	Operation Capabilities: This field is defined in Table 8-111 . Device Initiated capability bit shall be cleared to 0 if Restriction flags bit[0] is set to 1.
03h	2	Operation Mode: This field is defined in Table 8-111 .
05h	1	Maintenance Operation Class: This field shall be set to 02h (Memory Sparing).
06h	1	Maintenance Operation Subclass: Depends on the scope of the sparing needed.

Table 8-119. Memory Sparing Feature Readable Attributes (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
07h	0Ah	Reserved
11h	2	<p>Restriction Flags</p> <ul style="list-style-type: none"> • Bit[0]: Sparing Side Effects: <ul style="list-style-type: none"> — 0 = The device preserves the memory content and remains responsive to CXL.mem requests during the sparing operation. — 1 = The device is permitted to drop CXL.mem write requests, return poison in response to CXL.mem read requests during the sparing operation. The device does not guarantee preservation of HDM contents across the sparing operation. • Bit[1]: Hard Sparing: If set, the device has the capability for performing the sparing that is irreversible and that can survive any Conventional Reset. • Bit[2]: Soft Sparing: If set, the device has the capability for performing the sparing in a non-permanent way; thus, the change will be reverted after any Conventional Reset. • Bits[15:3]: Reserved.

Table 8-120 shows the Feature Data for Set Feature command.

Table 8-120. Memory Sparing Feature Writable Attributes

Byte Offset	Length in Bytes	Description
00h	2	<p>Operation Mode</p> <ul style="list-style-type: none"> • Bit[0]: Device Initiated: A value of 1 indicates that the device may initiate the sparing maintenance operation without host involvement. If cleared to 0, the device shall initiate the maintenance operation only when receiving an explicit maintenance command. • Bits[15:1]: Reserved.

8.2.9.8 PBR Component Command Set

Support for this command set is required for all devices that are PBR link capable (i.e., PBR switches and GFDs).

8.2.9.8.1 Identify PBR Component (Opcode 0700h)

This command provides information about a component specific to its PBR fabric capabilities.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required

Command Effects:

- None

Table 8-121. Identify PBR Component Response Payload

Byte Offset	Length in Bytes	Description
00h	2	<ul style="list-style-type: none"> Bits[11:0]: PID: Assigned PID of this device or FFFh if no PID has been assigned Bits[15:12]: Reserved.
02h	2	<ul style="list-style-type: none"> Bits[11:0]: Primary FM PID: PID of the FM registered as primary FM. A value of FFFh indicates no primary FM is registered. Bits[15:12]: Reserved.
04h	10h	Primary FM UUID : UUID of the FM registered as primary FM.
14h	2	<ul style="list-style-type: none"> Bits[11:0]: Secondary FM PID: PID of the FM registered as secondary FM. A value of FFFh indicates no secondary FM is registered. Bits[15:12]: Reserved.
16h	10h	Secondary FM UUID : UUID of the FM registered as secondary FM.

8.2.9.8.2 Claim Ownership (Opcode 0701h)

This command is used by an FM to register itself as either the primary or secondary FM. The device's PID is also assigned as part of this operation. The device's PID assignment shall only be updated if the FW ownership registration operation is successful.

Registration of FMs and assignment of a PID apply to all CCIs on a PBR component.

Operation 0 (Register Primary FM and assign PID) shall fail with "Invalid Input" if a device already has a primary FM registered.

Operation 1 (Register Secondary FM) shall fail with "Invalid Input" if a device already has a secondary FM registered or if the request was initiated by an FM other than the registered primary FM.

Operation 2 (Update PID) is valid only when received from the primary FM and shall terminate with "Invalid Input" otherwise.

Operation 3 (Promote Secondary FM) is valid only when received from the secondary FM and shall terminate with "Invalid Input" otherwise.

Promoting a secondary FM to the primary FM position leaves the secondary FM position unregistered.

Attempting to register or assign a PID of FFFh shall result in an "Invalid Input" failure return code.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 8-122. Claim Ownership Request Payload

Byte Offset	Length in Bytes	Description
0h	1	Operation: Indicates which operation is being performed: <ul style="list-style-type: none"> • 00h = Register Primary FM and assign PID • 01h = Register Secondary FM • 02h = Update PID • 03h = Promote Secondary FM • All other encodings are reserved
1h	2	<ul style="list-style-type: none"> • Bits[11:0]: FM PID: PID of the FM requesting ownership. Valid only if Operation is cleared to 00h or set to 01h. • Bits[15:12]: Reserved.
3h	10h	UUID: UUID of the FM requesting ownership. Valid only if Operation is cleared to 00h or set to 01h.
13h	2	<ul style="list-style-type: none"> • Bits[11:0]: Assigned PID: PID value being assigned to the device. Valid only if Operation is 00h or 02h. • Bits[15:12]: Reserved.

Table 8-123. Claim Ownership Response Payload

Byte Offset	Length in Bytes	Description
0h	2	<ul style="list-style-type: none"> • Bits[11:0]: Primary FM PID: PID of the FM registered as primary FM. A value of FFFh indicates no primary FM is registered. • Bits[15:12]: Reserved.
2h	10h	Primary FM UUID: UUID of the FM registered as primary FM.
12h	2	<ul style="list-style-type: none"> • Bits[11:0]: Secondary FM PID: PID of the FM registered as secondary FM. A value of FFFh indicates no secondary FM is registered. • Bits[15:12]: Reserved
14h	10h	Secondary FM UUID: UUID of the FM registered as secondary FM.

8.2.9.8.3 Read CDAT (Opcode 0702h)

This command is used to read the CDAT from GAEs and GFDs.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 8-124. Read CDAT Request Payload

Byte Offset	Length in Bytes	Description
0h	2	<ul style="list-style-type: none"> Bits[11:0]: Target PID: PID of device routing path CDAT to query. Valid only for PBR switches. Bits[15:12]: Reserved.
2h	2	Reserved
4h	8	Start Byte : Offset in bytes into CDAT Data .
Ch	8	Number of Bytes : Size in bytes of CDAT Data requested.

Table 8-125. Read CDAT Response Payload

Byte Offset	Length in Bytes	Description
0h	8	Total CDAT Size : Size in bytes of the full CDAT.
8h	8	Number of Bytes : Size in bytes of returned CDAT Data .
10h	Varies	CDAT Data : CDAT for the specified target, as defined in the CDAT Specification.

8.2.9.9 Memory Device Command Sets

This section describes the commands specific to CXL memory devices that implement the PCI Header Class Code defined in [Section 8.1.12.1](#), advertise Memory Device Command support in the Mailbox Capabilities register (see [Section 8.2.8.4.3](#)), or report a Type value of 03h or 04h in the Identify response payload.

Opcodes also provide an implicit major version number, which means a command's definition shall not change in an incompatible way in future revisions of this specification. Instead, if an incompatible change is required, the specification defining the change shall define a new opcode for the changed command. Commands may evolve by defining new fields in the payload definitions that were originally defined as Reserved, but only in a way where software written using the earlier definition will continue to work correctly, and software written to the new definition can use the 0 value or the payload size to detect devices that do not support the new field. This implicit *minor* versioning allows software to be written with the understanding that an opcode shall only evolve by adding backward-compatible changes.

[Table 8-126](#) and the following sections use the terms "Persistent memory device" and "CXL Memory Device that supports Persistence" interchangeably. A persistent memory device behaves in the following ways:

- All writes targeting persistent memory ranges that have been completed on CXL, but are still held in volatile buffers on the device, shall be flushed to media under the following conditions:
 - Any reset event
 - Reception of GPF Phase 2
 - Surprise power loss
- If the device is unable, for any reason, to flush all the writes that have been completed on CXL to persistent memory successfully, the Device shall increment the Dirty Shutdown Count in the Health Info (see [Table 8-133](#)) on the next reset. Incrementing the Dirty Shutdown Count may be considered a failure event by the Host and may indicate user data loss.

Opcodes 4000-BFFFh that are not specified in [Table 8-126](#) are reserved.

Table 8-126. CXL Memory Device Command Opcodes (Sheet 1 of 3)

Opcode				Required				
				Type 1/2/3 Device ¹	GFD			
Command Set Bits[15:8]		Command Bits[7:0]		Combined Opcode	Mailbox	Type 3 MCTP	FM Interface ²	Host Interface ³
40h	Identify Memory Device	00h	Identify Memory Device (Section 8.2.9.9.1.1)	4000h	M	M	M	M
41h	Capacity Config and Label Storage	00h	Get Partition Info (Section 8.2.9.9.2.1)	4100h	O	O	P	P
		01h	Set Partition Info (Section 8.2.9.9.2.2)	4101h	O	O ⁴	P	P
		02h	Get LSA (Section 8.2.9.9.2.3)	4102h	PM	O	P	P
		03h	Set LSA (Section 8.2.9.9.2.4)	4103h	PM	O ⁴	P	P
42h	Health Info and Alerts	00h	Get Health Info (Section 8.2.9.9.3.1)	4200h	M	O	M	O
		01h	Get Alert Configuration (Section 8.2.9.9.3.2)	4201h	M	O	M	O
		02h	Set Alert Configuration (Section 8.2.9.9.3.3)	4202h	M	O	M	O
		03h	Get Shutdown State (Section 8.2.9.9.3.4)	4203h	PM	O	P	P
		04h	Set Shutdown State (Section 8.2.9.9.3.5)	4204h	PM	P	P	P
43h	Media and Poison Management	00h	Get Poison List (Section 8.2.9.9.4.1)	4300h	PM	O	O	O
		01h	Inject Poison (Section 8.2.9.9.4.2)	4301h	O	P	O	O
		02h	Clear Poison (Section 8.2.9.9.4.3)	4302h	O	O ⁴	O	O
		03h	Get Scan Media Capabilities (Section 8.2.9.9.4.4)	4303h	PM	O	O	O
		04h	Scan Media (Section 8.2.9.9.4.5)	4304h	PM	O ⁴	O	O
		05h	Get Scan Media Results (Section 8.2.9.9.4.6)	4305h	PM	O	O	O
44h	Sanitize and Media Operations	00h	Sanitize (Section 8.2.9.9.5.1)	4400h	O	O ⁴	O	O
		01h	Secure Erase (Section 8.2.9.9.5.2)	4401h	O	O ⁴	O	O
		02h	Media Operations (Section 8.2.9.9.5.3)	4402h	O	P	O	O

Table 8-126. CXL Memory Device Command Opcodes (Sheet 2 of 3)

Opcode				Required				
				Type 1/2/3 Device ¹		GFD		
Command Set Bits[15:8]		Command Bits[7:0]		Combined Opcode	Mailbox	Type 3 MCTP	FM Interface ²	Host Interface ³
45h	Persistent Memory Data-at-rest Security	00h	Get Security State (Section 8.2.9.9.6.1)	4500h	O	O	P	P
		01h	Set Passphrase (Section 8.2.9.9.6.2)	4501h	O	O ⁴	P	P
		02h	Disable Passphrase (Section 8.2.9.9.6.3)	4502h	O	O ⁴	P	P
		03h	Unlock (Section 8.2.9.9.6.4)	4503h	O	O ⁴	P	P
		04h	Freeze Security State (Section 8.2.9.9.6.5)	4504h	O	O ⁴	P	P
		05h	Passphrase Secure Erase (Section 8.2.9.9.6.6)	4505h	O	O ⁴	P	P
46h	Security Passthrough	00h	Security Send (Section 8.2.9.9.7.1)	4600h	O	O ⁴	P	P
		01h	Security Receive (Section 8.2.9.9.7.2)	4601h	O	O	P	P
47h	SLD QoS Telemetry	00h	Get SLD QoS Control (Section 8.2.9.9.8.1)	4700h	O	O	P	P
		01h	Set SLD QoS Control (Section 8.2.9.9.8.2)	4701h	O	O ⁴	P	P
		02h	Get SLD QoS Status (Section 8.2.9.9.8.3)	4702h	O	O	P	P
48h	Dynamic Capacity	00h	Get Dynamic Capacity Configuration (Section 8.2.9.9.9.1)	4800h	DC	P	P	P
		01h	Get Dynamic Capacity Extent List (Section 8.2.9.9.9.2)	4801h	DC	P	P	P
		02h	Add Dynamic Capacity Response (Section 8.2.9.9.9.3)	4802h	DC	P	P	P
		03h	Release Dynamic Capacity (Section 8.2.9.9.9.4)	4803h	DC	P	P	P
5Bh	GFD Component Management	00h	Identify GFD (Section 8.2.9.9.10.1)	4900h	P	P	M	M
		01h	Get GFD Status (Section 8.2.9.9.10.2)	4901h	P	P	M	M
		02h	Get GFD DC Region Configuration (Section 8.2.9.9.10.3)	4902h	P	P	M	M
		03h	Set GFD DC Region Configuration (Section 8.2.9.9.10.4)	4903h	P	P	O	O

Table 8-126. CXL Memory Device Command Opcodes (Sheet 3 of 3)

Opcode			Required			
			Type 1/2/3 Device ¹	GFD		
Command Set Bits[15:8]	Command Bits[7:0]	Combined Opcode	Mailbox	Type 3 MCTP	FM Interface ²	Host Interface ³
5Bh	GFD Component Management	04h	Get GFD DC Region Extent Lists (Section 8.2.9.9.10.5)	4904h	P	P
		05h	Get GFD DMP Configuration (Section 8.2.9.9.10.6)	4905h	P	P
		06h	Set GFD DMP Configuration (Section 8.2.9.9.10.7)	4906h	P	P
		07h	GFD Dynamic Capacity Add (Section 8.2.9.9.10.8)	4907h	P	P
		08h	GFD Dynamic Capacity Release (Section 8.2.9.9.10.9)	4908h	P	P
		09h	GFD Dynamic Capacity Add Reference (Section 8.2.9.9.10.10)	4909h	P	P
		0Ah	GFD Dynamic Capacity Remove Reference (Section 8.2.9.9.10.11)	490Ah	P	P
		0Bh	GFD Dynamic Capacity List Tags (Section 8.2.9.9.10.12)	490Bh	P	P
		0Ch	Get GFD SAT Entry (Section 8.2.9.9.10.13)	490Ch	P	P
		0Dh	Set GFD SAT Entry (Section 8.2.9.9.10.14)	490Dh	P	P
		0Eh	Get GFD QoS Control (Section 8.2.9.9.10.15)	490Eh	P	P
		0Fh	Set GFD QoS Control (Section 8.2.9.9.10.16)	490Fh	P	P
		10h	Get GFD QoS Status (Section 8.2.9.9.10.17)	4910h	P	P
		11h	Get GFD QoS BW Limit (Section 8.2.9.9.10.18)	4911h	P	P
		12h	Set GFD QoS BW Limit (Section 8.2.9.9.10.19)	4912h	P	P
		13h	Get GDT Configuration (Section 8.2.9.9.10.20)	4913h	P	P
		14h	Set GDT Configuration (Section 8.2.9.9.10.21)	4914h	P	P

1. M = Mandatory; PM = Mandatory for devices that support persistence; DC = mandatory for devices that support Dynamic Capacity; O = Optional. P = Prohibited.

It is prohibited for switches to support any commands from the Memory Device Command Set.

2. "FM Interface" refers to commands issued/received via the Fabric Crawl Out mechanism.

3. "Host Interface" refers to commands issued/received via the GFD Proxying mechanism.

4. Systems capable of management from Mailbox registers and an MCTP-based CCI shall ensure that these commands are not issued as MCTP messages while a device's mailboxes are operational.

8.2.9.9.1 Identify Memory Device

8.2.9.9.1.1 Identify Memory Device (Opcode 4000h)

Retrieve basic information about the memory device. If the HDM_Count field in DVSEC CXL Capability is 01b, the output payload is valid only if the Memory_Info_Valid flag in DVSEC CXL Range 1 Size Low (see [Section 8.1.3.8.2](#)) is 01b. If the HDM_Count field in DVSEC CXL Capability is 10b, the output payload is valid only if the Memory_Info_Valid flag in DVSEC CXL Range 1 Size Low as well as DVSEC CXL Range 2 Size Low (see [Section 8.1.3.8.6](#)) are both 1.

Possible Command Return Codes:

- Success
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-127. Identify Memory Device Output Payload (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	16	FW Revision: Contains the revision of the active FW formatted as an ASCII string. This is the same information that may be retrieved with the Get FW Info command.
10h	8	Total Capacity: This field indicates the total usable capacity of the device. Expressed in multiples of 256 MB. Total device usable capacity is divided between volatile only capacity, persistent only capacity, and capacity that can be either volatile or persistent. Total Capacity shall be greater than or equal to the sum of Volatile Only Capacity and Persistent Only Capacity.
18h	8	Volatile Only Capacity: This field indicates the total usable capacity of the device that may be used only as volatile memory. Expressed in multiples of 256 MB.
20h	8	Persistent Only Capacity: This field indicates the total usable capacity of the device that may be used only as persistent memory. Expressed in multiples of 256 MB.
28h	8	Partition Alignment: If the device has capacity that may be used as either volatile memory or persistent memory, this field indicates the partition alignment size. Expressed in multiples of 256 MB. Partitionable capacity is equal to Total Capacity - Volatile Only Capacity - Persistent Only Capacity. If 0, the device doesn't support partitioning the capacity into both volatile capacity and persistent capacity.
30h	2	Informational Event Log Size: The number of events that the device can store in the Informational Event Log before the log overflows. The device shall support 1 or more events in each Event Log.
32h	2	Warning Event Log Size: The number of events that the device can store in the Warning Event Log before the log overflows. The device shall support 1 or more events in each Event Log.
34h	2	Failure Event Log Size: The number of events that the device can store in the Failure Event Log before the log overflows. The device shall support 1 or more events in each Event Log.
36h	2	Fatal Event Log Size: The number of events that the device can store in the Fatal Event Log before the log overflows. The device shall support 1 or more events in each Event Log.
38h	4	LSA Size: The size of the Label Storage Area. Expressed in bytes. The minimum LSA size is defined in Section 9.13.2 .

Table 8-127. Identify Memory Device Output Payload (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
3Ch	3	Poison List Maximum Media Error Records: The maximum number of Media Error Records that the device can track in its Poison List. The device shall set the Poison List Overflow in the Get Poison List output if this limit is exceeded. The device shall size the poison list accordingly to limit the chances of the list overflowing.
3Fh	2	Inject Poison Limit: The device's supported maximum number of physical addresses that can be poisoned by the Inject Poison command. When 0, the device does not have a poison injection limit. When nonzero, the device has a maximum limit of poison that can be injected using the Inject Poison command.
41h	1	Poison Handling Capabilities: The device's poison handling capabilities. <ul style="list-style-type: none"> • Bit[0]: Injects Persistent Poison: When set and the device supports poison injection, any poison injected in non-volatile DPA shall remain persistent across all types of device resets. When cleared and the device supports poison injection, Conventional or CXL Reset shall automatically clear the injected poison. • Bit[1]: Scans for Poison: When set, the device shall periodically scan its media for errors and shall automatically alert the host of those errors. If cleared, the device does not periodically scan for memory errors and does not generate an alert. • Bits[7:2]: Reserved.
42h	1	QoS Telemetry Capabilities: Optional QoS Telemetry for memory SLD capabilities for management by system software (see Section 3.3.4). <ul style="list-style-type: none"> • Bit[0]: Egress Port Congestion Supported: When set, the associated feature is supported, and the Get SLD QoS Control, Set SLD QoS Control, and Get SLD QoS Status commands shall be implemented (see Section 3.3.4.3.4) • Bit[1]: Temporary Throughput Reduction Supported: When set, the associated feature is supported, and the Get SLD QoS Control and Set SLD QoS Control commands shall be implemented (see Section 3.3.4.3.5) • Bits[7:2]: Reserved
43h	2	Dynamic Capacity Event Log Size: The number of events that the device can store in the Dynamic Capacity Event Log before the log overflows. A DCD shall support one or more events in this Event Log. If the Dynamic Capacity Event Log overflows, the host shall retrieve the complete list of extents that utilize the Get Dynamic Capacity Extent List to retrieve the correct Dynamic Capacity.

8.2.9.9.2 Capacity Configuration and Label Storage

8.2.9.9.2.1 Get Partition Info (Opcode 4100h)

Get the Active and Next capacity settings for a memory device, describing the amount of volatile and persistent memory capacities available. The Active values describe the current capacities provided by the device in the currently active configuration. The Next values describe a new configuration that has not yet taken effect, to become active on the next reset (as specified in the Set Partition command effects). If the HDM_Count field in DVSEC CXL Capability is 01b, the output payload is valid only if the Memory_Info_Valid flag in DVSEC CXL Range 1 Size Low (see [Section 8.1.3.8.2](#)) is 01b. If the HDM_Count field in DVSEC CXL Capability is 10b, the output payload is valid only if the Memory_Info_Valid flag in DVSEC CXL Range 1 Size Low as well as DVSEC CXL Range 2 Size Low (see [Section 8.1.3.8.6](#)) are both 1.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error

- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-128. Get Partition Info Output Payload

Byte Offset	Length in Bytes	Description
00h	8	Active Volatile Capacity: Total device volatile memory capacity in multiples of 256 MB. This is the sum of the device's Volatile Only capacity and the capacity that is partitioned for volatile use. The device shall provide this volatile capacity starting at DPA 0.
08h	8	Active Persistent Capacity: Total device persistent memory capacity in multiples of 256 MB. This is the sum of the device's Persistent Only capacity and the capacity that is partitioned for persistent use. The device shall provide this persistent capacity starting at the DPA immediately following the volatile capacity.
10h	8	Next Volatile Capacity: If nonzero, this value shall become the Active Volatile Capacity on the next reset (as specified in the Set Partition command effects). If both this field and the Next Persistent Capacity field are 0, there is no pending change to the partitioning.
18h	8	Next Persistent Capacity: If nonzero, this value shall become the Active Persistent Capacity on the next reset (as specified in the Set Partition command effects). If both this field and the Next Volatile Capacity field are 0, there is no pending change to the partitioning.

8.2.9.9.2.2 Set Partition Info (Opcode 4101h)

Set the partitioning between volatile capacity and persistent capacity for the partitionable capacity. Partitionable capacity is equal to (Total Capacity - Volatile Only Capacity - Persistent Only Capacity). This command shall fail with an Unsupported error if there is no partitionable capacity (i.e., Identify Memory Device reports Partition Alignment as zero). The device shall return Invalid Input if the specified capacity is not aligned to the partition alignment requirement reported in the Identify Memory Device command. Using this command to change the size of the persistent capacity shall result in the loss of data stored.

In support of confidential computing, if the device has been locked while utilizing secure CXL TSP interfaces, the device shall reject any attempt to change the partitioning of the device with the Immediate flag set by returning Invalid Security State status for this command. See [Section 11.5](#) for details on locking a device and locked device behavior.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Media Disabled
- Busy
- Invalid Security State

Command Effects:

- Configuration Change after Cold Reset
- CEL[11:10] Valid
- Configuration Change after Conventional Reset
- Configuration Change after CXL Reset
- Immediate Configuration Change
- Immediate Data Change

Table 8-129. Set Partition Info Input Payload

Byte Offset	Length in Bytes	Description
00h	8	Volatile Capacity: The amount of partitionable capacity that shall be allocated to volatile capacity, in multiples of 256 MB aligned to the partition alignment requirement reported in the Identify Memory Device command. The remainder of the partitionable capacity shall be allocated to persistent capacity.
08h	1	Flags <ul style="list-style-type: none"> • Bit[0]: Immediate: When set, the change is immediately requested. If cleared, the change in partitioning shall become the “next” configuration, to become active on the next reset (as specified in the command effects). In this case, the new configuration shall be reported in the Next Volatile Capacity and Next Persistent Capacity fields returned by the Get Partition Info command. It is the caller’s responsibility to avoid immediate changes to the partitioning when the device is in use. • Bits[7:1]: Reserved.

8.2.9.9.2.3 Get LSA (Opcode 4102h)

The Label Storage Area (LSA) shall be supported by a memory device that provides persistent memory capacity and may be supported by a device that provides only volatile memory capacity. The format of the LSA is specified in [Section 9.13.2](#). The size of the Label Storage Area is retrieved from the Identify Memory Device command.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Media Disabled
- Busy

Command Effects:

- None

Table 8-130. Get LSA Input Payload

Byte Offset	Length in Bytes	Description
00h	4	Offset: The byte offset in the LSA to return in the output payload.
04h	4	Length: Length in bytes of LSA to return in the output payload.

Table 8-131. Get LSA Output Payload

Byte Offset	Length in Bytes	Description
00h	Varies	Data: Requested bytes from the LSA.

8.2.9.9.2.4 Set LSA (Opcode 4103h)

The format of the Label Storage Area is specified in [Section 9.13.2](#).

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Media Disabled
- Busy
- Invalid Security State

Command Effects:

- Immediate Configuration Change
- Immediate Data Change

Table 8-132. Set LSA Input Payload

Byte Offset	Length in Bytes	Description
00h	4	Offset: The byte offset in the LSA.
04h	4	Reserved
08h	Varies	Data: The data to be written to LSA at the specified offset.

8.2.9.9.3 Health Information and Alerts**8.2.9.9.3.1 Get Health Info (Opcode 4200h)**

Get the current instantaneous health of the device. It is not necessary to poll for health changes. Anytime the health of the device changes, the device shall add an appropriate event to its internal event log, update the Event Status register, and if configured, interrupt the host.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-133. Get Health Info Output Payload (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	1	<p>Health Status: Overall device health summary. Normal health status is all bits cleared.</p> <ul style="list-style-type: none"> • Bit[0]: Maintenance Needed: The device requires maintenance. When transitioning from 0 to 1, the device shall add an event to the Warning or Failure Event Log, update the Event Status register, and if configured, interrupt the host. • Bit[1]: Performance Degraded: The device is no longer operating at optimal performance. When transitioning from 0 to 1, the device shall add an event to the Warning Event Log, update the Event Status register, and if configured, interrupt the host. • Bit[2]: Hardware Replacement Needed: The device should immediately be replaced. When transitioning from 0 to 1, the device shall add an event to the Failure or Fatal Event Log, update the Event Status register, and if configured, interrupt the host. • Bit[3]: Memory Capacity Degraded: The device is no longer operating at the expected memory capacity. When transitioning from 0 to 1, the device shall add an event to the Warning Event Log, update the Event Status register, and if configured, interrupt the host. • Bits[7:4]: Reserved.
01h	1	<p>Media Status: Overall media health summary. When transitioning from any state to any other state, the device shall add an event to the Failure or Fatal Event Log, update the Event Status register, and if configured, interrupt the host. When transitioning from Not Ready to Normal state, no Event Record is required.</p> <ul style="list-style-type: none"> • 00h = Normal. The device's media is operating normally. • 01h = Not Ready. The device's media is not ready. • 02h = Write persistency Lost. The device cannot persist write requests but is able to read stored data. This is considered an abnormal status only for reporting if the device media can be written to and not an indicator of whether the device is in a security state that allows writing. • 03h = All data lost. All data has been lost from the device. • 04h = Write Persistence Loss in the Event of Power Loss. The device's ability to persist subsequent write requests may be lost in the event of a power loss. • 05h = Write Persistence Loss in Event of Shutdown. The device's ability to persist subsequent write requests may be lost when the device is shut down. • 06h = Write Persistence Loss Imminent. The device's ability to persist subsequent write requests may be lost. • 07h = All Data Loss in the Event of Power Loss. All data on the device may be lost in the event of a power loss. • 08h = All Data Loss in the Event of Shutdown. All data on the device may be lost when the device is shut down. • 09h = All Data Loss Imminent. All data on the device may be lost. • All other encodings are reserved.

Table 8-133. Get Health Info Output Payload (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
02h	1	<p>Additional Status</p> <ul style="list-style-type: none"> Bits[1:0]: Life Used: The device's current life used status. <ul style="list-style-type: none"> 00b = Normal. The device's life used is within normal operating range. 01b = Warning. The device's Life Used has risen to the user-programmable warning threshold. When transitioning from Normal to Warning status, the device shall add an event to the Warning Event Log, update the Event Status register, and if configured, interrupt the host. 10b = Critical. The device Life Used has risen to the point where the device's performance or reliability may be affected, and the device should be replaced. When transitioning from Normal or Warning to Critical status, the device shall add an event to the Failure or Fatal Event Log, update the Event Status register, and if configured, interrupt the host. 11b = Reserved. Bits[3:2]: Device Temperature: The device's current temperature status. <ul style="list-style-type: none"> 00b = Normal. The device's temperature is within normal operating range. When transitioning from Warning or Critical state to Normal, the device shall add an event to the Informational Event Log, update the Event Status register, and if configured, interrupt the host. 01b = Warning. The device's temperature has reached the user-programmable warning threshold. When transitioning from Normal to Warning status, the device shall add an event to the Warning Event Log, update the Event Status register, and if configured, interrupt the host. 10b = Critical. The device temperature has reached the point at which the device's performance or reliability may be affected, and immediate action should be taken to correct the device temperature. When transitioning from Normal or Warning to Critical status, the device shall add an event to the Failure or Fatal Event Log, update the Event Status register, and if configured, interrupt the host. 11b = Reserved. Bit[4]: Corrected Volatile Error Count: The device's current corrected volatile error count. <ul style="list-style-type: none"> 0 = Normal. The device's corrected error counts are below the warning threshold. 1 = Warning or Failure. The device's count of total corrected errors has risen to or above the user-programmable Warning or Failure threshold. When transitioning from Normal to Warning or Failure status, the device shall add an event to the Warning or Failure Event Log, update the Event Status register, and if configured, interrupt the host. Bit[5]: Corrected Persistent Error Count: The device's current corrected persistent error count. <ul style="list-style-type: none"> 0 = Normal. The device's corrected error counts are below the warning threshold. 1 = Warning. The device's count of total corrected errors has risen to or above the user-programmable warning threshold. When transitioning from Normal to Warning status, the device shall add an event to the Warning Event Log, update the Event Status register, and if configured, interrupt the host. Bits[7:6]: Reserved.
03h	1	Life Used : The device's used life as a percentage value (0-100) of factory-expected life span. A value of 100 means that the device's calculated useful life span has been reached and the device should be replaced. It does not imply that the device stops functioning when it reaches 100. Returns FFh if not implemented.
04h	2	Device Temperature : The device's current temperature in degrees Celsius, represented as a 2's complement value. Returns 7FFFh if not implemented.
06h	4	Dirty Shutdown Count : A monotonically increasing counter that is incremented whenever the device fails to save and/or flush data to the persistent media or is unable to determine whether data loss may have occurred. The count is persistent across power loss and wraps back to 0 at overflow.
0Ah	4	Corrected Volatile Error Count : The total number of correctable memory errors that the device has detected as having occurred in the volatile memory partition. The initial value of this counter shall be 0 and the counter shall saturate at FFFF FFFFh. The counter shall be maintained by the device and cannot be modified by the host. Return 0 for devices that do not track corrected errors. This count is reset on a Conventional Reset.
0Eh	4	Corrected Persistent Error Count : The total number of correctable memory errors that the device has detected as having occurred in the persistent memory partition. The initial value of this counter shall be 0 and the counter shall saturate at FFFF FFFFh. The counter shall be maintained by the device and cannot be modified by the host. Return 0 for devices that do not track corrected errors. This count is reset on a Conventional Reset.

8.2.9.9.3.2 Get Alert Configuration (Opcode 4201h)

Retrieve the device's critical alert and programmable warning configuration. Critical alerts shall automatically be configured by the device after a Conventional Reset. If supported, programmable warning thresholds shall be initialized to vendor-recommended defaults by the device on a Conventional Reset.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-134. Get Alert Configuration Output Payload (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	1	Valid Alerts: Indicators of which alert fields are valid in the returned data. <ul style="list-style-type: none"> • Bit[0]: When set, the Life Used Programmable Warning Threshold field is valid • Bit[1]: When set, the Device Over-Temperature Programmable Warning Threshold field is valid • Bit[2]: When set, the Device Under-Temperature Programmable Warning Threshold field is valid • Bit[3]: When set, the Corrected Volatile Memory Error Programmable Warning Threshold field is valid • Bit[4]: When set, the Corrected Persistent Memory Error Programmable Warning Threshold field is valid • Bits[7:5]: Reserved
01h	1	Programmable Alerts: Indicators of which device alerts are programmable by the host. <ul style="list-style-type: none"> • Bit[0]: When set, the Life Used Programmable Warning Threshold is programmable by the host • Bit[1]: When set, the Device Over-Temperature Programmable Warning Threshold field is programmable by the host • Bit[2]: When set, the Device Under-Temperature Programmable Warning Threshold field is programmable by the host • Bit[3]: When set, the Corrected Volatile Memory Error Programmable Warning is programmable by the host • Bit[4]: When set, the Corrected Persistent Memory Error Programmable Warning is programmable by the host • Bits[7:5]: Reserved
02h	1	Life Used Critical Alert Threshold: The device's default alert when the Life Used rises above this percentage-based value. Valid values are 0-100.
03h	1	Life Used Programmable Warning Threshold: The device's currently programmed warning threshold when the life used rises to or above this percentage-based value. Valid values are 0-100. The life used warning threshold shall be less than the life used critical alert value.
04h	2	Device Over-Temperature Critical Alert Threshold: The device's default critical over-temperature alert threshold when the device temperature rises to or above this threshold in degrees Celsius, represented as a 2's complement value.
06h	2	Device Under-Temperature Critical Alert Threshold: The device's default critical under-temperature alert threshold when the device temperature falls to or below this threshold in degrees Celsius, represented as a 2's complement value.

Table 8-134. Get Alert Configuration Output Payload (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
08h	2	Device Over-Temperature Programmable Warning Threshold: The device's currently programmed over-temperature warning threshold when the device temperature rises to or above this threshold in degrees Celsius, represented as a 2's complement value. Note that the device temperature set by this field (not the 2's complement value) shall be less than the device temperature set by the Device Over-Temperature Critical Alert Threshold field (not its 2's complement value).
0Ah	2	Device Under-Temperature Programmable Warning Threshold: The device's currently programmed under-temperature warning threshold when the device temperature falls to or below this threshold in degrees Celsius, represented as a 2's complement value. Note that the device temperature set by this field (not the 2's complement value) shall be higher than the device temperature set by the Device Under-Temperature Critical Alert Threshold field (not its 2's complement value).
0Ch	2	Corrected Volatile Memory Error Programmable Warning Threshold: The device's currently programmed warning threshold for corrected volatile memory errors before signaling a corrected error event to the host. A single event is generated whenever the total number of corrected errors on the device becomes equal to this threshold value and no corrected error events are generated before that has occurred.
0Eh	2	Corrected Persistent Memory Error Programmable Warning Threshold: The device's currently programmed warning threshold for corrected persistent memory errors before signaling a corrected error event to the host. A single event is generated whenever the total number of corrected errors on the device becomes equal to this threshold value and no corrected error events are generated before that has occurred.

8.2.9.9.3.3 Set Alert Configuration (Opcode 4202h)

Set Alert Configuration allows the host to configure programmable warning thresholds optionally. If supported, programmable warning thresholds shall be initialized to vendor-recommended defaults by the device on a Conventional Reset. After completion of this command, the requested programmable warning thresholds shall replace any previously programmed warning thresholds.

Any time a programmed warning threshold is reached, the device shall add an appropriate event record to its event log, update the Event Status register, and if configured, interrupt the host. If the conditions are already met for the newly programmed warning at the time this command is executed, the device shall immediately generate the event record and interrupt for the alert.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- Immediate Policy Change

Table 8-135. Set Alert Configuration Input Payload

Byte Offset	Length in Bytes	Description
00h	1	<p>Valid Alert Actions: Indicators of which alert fields are valid in the supplied input payload.</p> <ul style="list-style-type: none"> • Bit[0]: When set, the Life Used Programmable Warning Threshold Enable Alert Action and field shall be valid • Bit[1]: When set, the Device Over-Temperature Programmable Warning Threshold Enable Alert Action and field shall be valid • Bit[2]: When set, the Device Under-Temperature Programmable Warning Threshold Enable Alert Action and field shall be valid • Bit[3]: When set, the Corrected Volatile Memory Error Programmable Warning Threshold Enable Alert Action and field shall be valid • Bit[4]: When set, the Corrected Persistent Memory Error Programmable Warning Threshold Enable Alert Action and field shall be valid • Bits[7:5]: Reserved
01h	1	<p>Enable Alert Actions: The device shall enable the following programmable alerts.</p> <ul style="list-style-type: none"> • Bit[0]: When set, the device shall enable its Life Used Programmable Warning Threshold. When cleared, the device shall disable its life used programmable warning. • Bit[1]: When set, the device shall enable its Device Over-Temperature Programmable Warning Threshold. When cleared, the device shall disable its device Under-Temperature programmable warning. • Bit[2]: When set, the device shall enable its Device Under-Temperature Programmable Warning Threshold. When cleared, the device shall disable its device Under-Temperature programmable warning. • Bit[3]: When set, the device shall enable its Corrected Volatile Memory Error Programmable Warning Threshold. When cleared, the device shall disable its corrected volatile memory error programmable warning. • Bit[4]: When set, the device shall enable its Corrected Persistent Memory Error Programmable Warning Threshold. When cleared, the device shall disable its corrected persistent memory error programmable warning. • Bits[7:5]: Reserved.
02h	1	Life Used Programmable Warning Threshold: The device's updated life used programmable warning threshold.
03h	1	Reserved
04h	2	Device Over-Temperature Programmable Warning Threshold: The device's updated Over-Temperature programmable warning threshold.
06h	2	Device Under-Temperature Programmable Warning Threshold: The device's updated Under-Temperature programmable warning threshold.
08h	2	Corrected Volatile Memory Error Programmable Warning Threshold: The device's updated programmable warning threshold for corrected volatile memory errors before signaling a corrected error event to the host. A single event is generated whenever the total number of corrected errors on the device becomes equal to this threshold value and no corrected error events are generated before that has occurred.
0Ah	2	Corrected Persistent Memory Error Programmable Warning Threshold: The device's updated programmable warning threshold for corrected persistent memory errors before signaling a corrected error event to the host. A single event is generated whenever the total number of corrected errors on the device becomes equal to this threshold value and no corrected error events are generated before that has occurred.

8.2.9.9.3.4 Get Shutdown State (Opcode 4203h)

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error

- Retry Required
- Invalid Payload Length
- Media Disabled

Command Effects:

- None

Table 8-136. Get Shutdown State Output Payload

Byte Offset	Length in Bytes	Description
00h	1	<p>State: The current Shutdown State.</p> <ul style="list-style-type: none"> • Bit[0]: Dirty: A 1 value indicates the device's internal Shutdown State is "dirty". A 0 value indicates "clean". • Bits[7:1]: Reserved.

8.2.9.9.3.5 Set Shutdown State (Opcode 4204h)

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required
- Invalid Payload Length
- Media Disabled

Command Effects:

- Immediate Policy Change

Table 8-137. Set Shutdown State Input Payload

Byte Offset	Length in Bytes	Description
00h	1	<p>State: The current Shutdown State.</p> <ul style="list-style-type: none"> • Bit[0]: Dirty: A 1 value sets the device's internal Shutdown State to "dirty". A 0 value sets it to "clean". The device shall persistently store this state and use it after the next Conventional Reset to determine whether the Dirty Shutdown Count described in Section 8.2.9.9.3.1 gets updated. If the Shutdown State is "dirty", the device shall increment the Dirty Shutdown Count and then set the Shutdown State to "clean". This post-reset logic shall occur before the device accepts any commands or memory I/O. The value set by this mailbox command shall be overridden by the device in two cases: <ul style="list-style-type: none"> — On a successful GPF flow, the device shall set the Shutdown State to "clean" — When handling a shutdown/reset, if the device detects an internal failure that jeopardizes data integrity (e.g., a failed internal flush), the device shall set the Shutdown State to "dirty" • Bits[7:1]: Reserved

8.2.9.9.4 Media and Poison Management

8.2.9.9.4.1 Get Poison List (Opcode 4300h)

Get Poison List command shall return an unordered list of locations that are poisoned or result in poison if the addresses were accessed by the host. This command is not a background operation and the device shall return data without delay. The device may reject this command if the requested range spans the device's volatile and persistent partitions.

The device shall return the known list of locations with media errors for the requested address range when the device processes the command. Any time that the device detects a new poisoned location, the device shall add the DPA to the Poison List, add an appropriate event to its Warning, Informational, or Failure Event Log, update the Event Status register, and if configured, interrupt the host. In response, the host should reissue this command to retrieve the updated Poison List.

When poison is written:

- Using CXL.mem: The device shall add the new DPA to the device's Poison List and then shall set the error source to an external error.
- Using a CXL-defined poison injection interface (e.g., Inject Poison command): The device shall add the new DPA to the device's Poison List and then shall set the error source to an injected error.
- By the device because of a device-detected internal error (e.g., device media scrub discovers new media error): The device shall add the new DPA to the device's Poison List and then shall set the error source to an internal error.

When poison is cleared, the DPA shall no longer be reported in the device's Poison List.

If the device does not support poison list for volatile ranges and any location in the requested list maps to volatile, the device shall return Invalid Physical Address.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Media Disabled
- Invalid Physical Address
- Invalid Security State

Command Effects:

- None

Table 8-138. Get Poison List Input Payload

Byte Offset	Length in Bytes	Description
00h	8	<p>Poison List Flags: Flags that affect the returned list.</p> <ul style="list-style-type: none"> • Bit[0]: Restart Request: If set to 1, the device shall send the Poison List starting from the first entry, even if a previous transfer was incomplete. A device supporting this flag shall set the Restart Ack bit in the output payload in response to this flag being set. A device that does not support this flag must not set the Restart Ack bit. • Bits[5:1]: Reserved. <p>Get Poison List Physical Address: The starting DPA for which to retrieve the Poison List.</p> <ul style="list-style-type: none"> • Bits[7:6]: DPA[7:6] • Bits[15:8]: DPA[15:8] • ... • Bits[63:56]: DPA[63:56]
08h	8	Get Poison List Physical Address Length: The range of physical addresses for which to retrieve the Poison List. This length shall be in units of 64 bytes (e.g., if this field is 2h, that indicates the length is 128 bytes).

Table 8-139. Get Poison List Output Payload

Byte Offset	Length in Bytes	Description
00h	1	<p>Poison List Flags: Flags that describe the returned list.</p> <ul style="list-style-type: none"> • Bit[0]: More Media Error Records: When set, the device has more Media Error Records to return for the given Get Poison List address range. The host should keep issuing the Get List Poison command with the same Get Poison List Physical Address/Length, and retrieve records until this indicator is no longer set. Requests with a different Physical Address or Physical Address Length must return entries, that are within the address range specified by the new request, until More Media Error Records is not set; however, this is not guaranteed to be a complete list. • Bit[1]: Poison List Overflow: When set, the returned list has overflowed, and the returned list can no longer be considered a complete list. The device shall freeze the contents of the list and continue to report the overflow condition until the list is cleared and rebuilt by performing a Scan Media request using the full address range. There is no guarantee that rebuilding the list will remove the overflow condition. When set, the Overflow Timestamp field shall be valid. • Bit[2]: Scan Media in Progress: When set, a background operation to scan the media is executing and the returned list may or may not be a complete list while this flag is set. • Bit[3]: Restart Ack: Set by a device that supports the Restart Request flag in response to that flag being set in the Input Payload. • Bits[7:4]: Reserved.
01h	1	Reserved
02h	8	Overflow Timestamp: The time at which the device determined the poison list overflowed. This field is only valid if the overflow indicator is set. The number of unsigned nanoseconds that have elapsed since midnight, 01-Jan-1970, UTC. If the device does not have a valid timestamp, return 0.
0Ah	2	Media Error Record Count: Number of records in the Media Error Records list.
0Ch	14h	Reserved
20h	Varies	Media Error Records: The list of media error records.

The Media Error Records returned here are also used for the Get Scan Media Results command (see [Section 8.2.9.9.4.6](#)) and are defined in [Table 8-140](#).

Table 8-140. Media Error Record

Byte Offset	Length in Bytes	Description
00h	8	<p>Media Error Address: The DPA of the memory error and error source.</p> <ul style="list-style-type: none"> • Bits[2:0]: Error Source: The device shall report one of the following error sources with each DPA reported <ul style="list-style-type: none"> — 000b = Unknown. — 001b = External. Poison received from a source external to the device. — 010b = Internal. The device generated poison from an internal source. — 011b = Injected. The error was injected into the device for testing purposes. — 111b = Vendor Specific. — All other encodings are reserved. • Bits[5:3]: Reserved • Bits[7:6]: DPA[7:6] • Bits[15:8]: DPA[15:8] • ... • Bits[63:56]: DPA[63:56]
08h	4	Media Error Length: The number of adjacent DPAs in this media error record. This shall be nonzero. Devices may coalesce adjacent memory errors into a single entry. This length shall be in units of 64 bytes.
0Ch	4	Reserved

8.2.9.9.4.2 Inject Poison (Opcode 4301h)

An optional command to inject poison into a requested physical address. If the host injects poison using this command, the device shall return poison when the address is accessed through the CXL.mem bus.

Injecting poison shall add the new physical address to the device's poison list and the error source shall be set to an injected error. In addition, the device shall add an appropriate poison creation event to its internal Informational Event Log, update the Event Status register, and if configured, interrupt the host.

It is not an error to inject poison into a DPA that already has poison present and no error is returned.

In support of confidential computing, if the device has been locked while utilizing secure CXL TSP interfaces, the device shall reject any attempt to change the data on the device or inject poison by returning Invalid Security State status for this command. See [Section 11.5](#) for details on locking a device and locked device behavior.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required
- Invalid Payload Length
- Media Disabled
- Invalid Physical Address
- Inject Poison Limit Reached
- Invalid Security State

Command Effects:

- Immediate Data Change

Table 8-141. Inject Poison Input Payload

Byte Offset	Length in Bytes	Description
00h	8	Inject Poison Physical Address: The requested DPA at which poison shall be injected by the device. <ul style="list-style-type: none"> • Bits[5:0]: Reserved • Bits[7:6]: DPA[7:6] • Bits[15:8]: DPA[15:8] • ... • Bits[63:56]: DPA[63:56]

8.2.9.9.4.3 Clear Poison (Opcode 4302h)

An optional command to clear poison from the requested physical address and atomically write the included data in its place. This provides the same functionality as the host directly writing new data to the device.

Clearing poison shall remove the physical address from the device's Poison List. It is not an error to clear poison from an address that does not have poison set. If the device detects that it is not possible to clear poison from the physical address, the device shall return a permanent media failure code for this command.

In support of confidential computing, if the device has been locked while utilizing secure CXL TSP interfaces, the device shall reject any attempt to change the data on the device or clear poison by returning Invalid Security State status for this command. See [Section 11.5](#) for details on locking a device and locked device behavior.

This command must not modify the content of the Extended Metadata field associated with this address.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Media Disabled
- Permanent Media Failure
- Invalid Physical Address
- Invalid Security State

Command Effects:

- Immediate Data Change

Table 8-142. Clear Poison Input Payload

Byte Offset	Length in Bytes	Description
00h	8	Clear Poison Physical Address: The requested DPA from which poison shall be cleared by the device. <ul style="list-style-type: none"> • Bits[5:0]: Reserved • Bits[7:6]: DPA[7:6] • Bits[15:8]: DPA[15:8] • ... • Bits[63:56]: DPA[63:56]
08h	64	Clear Poison Write Data: The data the device shall always write into the requested physical address, atomically, while clearing poison if the location is marked as being poisoned.

8.2.9.9.4.4 Get Scan Media Capabilities (Opcode 4303h)

This command allows the device to report capabilities and options for the Scan Media feature based on the requested range. The device may reject this command if the range requested spans the device's volatile and persistent partitions.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Media Disabled
- Invalid Physical Address
- Invalid Security State

Command Effects:

- None

Table 8-143. Get Scan Media Capabilities Input Payload

Byte Offset	Length in Bytes	Description
00h	8	Get Scan Media Capabilities Start Physical Address: The starting DPA from which to retrieve Scan Media capabilities. <ul style="list-style-type: none"> • Bits[5:0]: Reserved • Bits[7:6]: DPA[7:6] • Bits[15:8]: DPA[15:8] • ... • Bits[63:56]: DPA[63:56]
08h	8	Get Scan Media Capabilities Physical Address Length: The range of physical addresses for which to retrieve Scan Media capabilities. This length shall be in units of 64 bytes.

Table 8-144. Get Scan Media Capabilities Output Payload

Byte Offset	Length in Bytes	Description
00h	4	Estimated Scan Media Time: The number of milliseconds that the device estimates are required to complete the Scan Media request over the range specified in the input. The device shall return all 0s if it cannot estimate a time for the specified range.

8.2.9.9.4.5 Scan Media (Opcode 4304h)

The Scan Media command causes the device to initiate a scan of a portion of its media for locations that are poisoned or result in poison if the addresses were accessed by the host. The device may update its Poison List as a result of executing the scan and shall complete any changes to the Poison List before signally completion of the Scan Media background operation. If the device updates its Poison List while the Scan Media background operation is executing, the device shall indicate that a media scan is in progress if Get Poison List is called during the scan. The host should use this command only if the poison list has overflowed and is no longer a complete list of the memory errors that exist on the media. The device may reject this command if the requested range spans the device's volatile and persistent partitions.

If interrupts are enabled for reporting internally or externally generated poison, and the poison list has not overflowed, the host should avoid using this command. It is expensive and may impact the performance of other operations on the device. This is intended only as a backup to retrieve the list of memory error locations in the event the poison list has overflowed.

Because the execution of a media scan may take significant time to complete, it is considered a background operation. The Scan Media command shall initiate the background operation and provide immediate status on the device's ability to start the scan operation. Any previous Scan Media results are discarded by the device upon receiving a new Scan Media command. Once the Scan Media command is successfully started, the Background Command Status register is used to retrieve the status. The Get Scan Media Results command shall return the list of poisoned memory locations.

Possible Command Return Codes:

- Success
- Unsupported
- Background Command Started
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Media Disabled
- Busy
- Aborted
- Invalid Physical Address
- Invalid Security State

Command Effects:

- Background Operation
- Request Abort Background Operation Command Supported

Table 8-145. Scan Media Input Payload

Byte Offset	Length in Bytes	Description
00h	8	Scan Media Physical Address: The starting DPA from which to start the scan. <ul style="list-style-type: none"> • Bits[5:0]: Reserved • Bits[7:6]: DPA[7:6] • Bits[15:8]: DPA[15:8] • ... • Bits[63:56]: DPA[63:56]
08h	8	Scan Media Physical Address Length: The range of physical addresses to scan. This length shall be in units of 64 bytes.
10h	1	Scan Media Flags <ul style="list-style-type: none"> • Bit[0]: No Event Log: When set, the device shall not generate event logs for media errors found during the Scan Media operation. • Bits[7:1]: Reserved.

8.2.9.9.4.6 Get Scan Media Results (Opcode 4305h)

Get Scan Media Results returns an unordered list of poisoned memory locations, in response to the Scan Media command. If the Scan Media command has not been called since the last Conventional Reset, the device shall return the Unsupported return code. The completion status for the Scan Media command is returned in the Background Command Status register and is not repeated here.

Because the returned list can be larger than the output payload size, it is possible to return the list in multiple calls to Get Scan Media Results. The More Media Error Records indicator shall be set by the device anytime there are more records to retrieve. The caller should continue to issue this command until this indicator is no longer set.

If the device cannot complete the scan and requires the host to retrieve scan media results before the device can continue the scan, the device shall set the Scan Media Stopped Prematurely indicator, return a valid Scan Media Restart Physical Address and Scan Media Restart Physical Address Length. This is the physical address range the device would require the Scan Media command to be called again with to continue the scan. It is the responsibility of the host to issue the Scan Media command, using this restart context, to guarantee that the Device's entire physical address range is eventually scanned.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required
- Invalid Payload Length
- Media Disabled
- Busy
- Invalid Security State

Command Effects:

- None

Table 8-146. Get Scan Media Results Output Payload

Byte Offset	Length in Bytes	Description
00h	8	Scan Media Restart Physical Address: The location from which the host should restart the Scan Media operation if the device could not complete the requested scan. The device shall report a valid restart address if it returns Scan Media Stopped Prematurely status. <ul style="list-style-type: none"> • Bits[5:0]: Reserved • Bits[7:6]: DPA[7:6] • Bits[15:8]: DPA[15:8] • ... • Bits[63:56]: DPA[63:56]
08h	8	Scan Media Restart Physical Address Length: The remaining range from which the host should restart the Scan Media operation if the device could not complete the requested scan. The device shall report a valid restart length if it returns Scan Media Stopped Prematurely status. This length shall be in units of 64 bytes.
10h	1	Scan Media Flags <ul style="list-style-type: none"> • 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 Physical Address & Scan Media Physical Address Length and retrieve records until this indicator is no longer set. • Bit[1]: Scan Stopped Prematurely: The device has run out of internal storage space for the error list. The device shall report a valid Scan Media Restart Physical Address and Scan Media Restart Physical Address Length to allow the host to restart the Scan Media command after retrieving the errors from the current scan. • Bits[7:2]: Reserved.
11h	1	Reserved
12h	2	Media Error Record Count: The number of records in the Media Error Records list.
14h	0Ch	Reserved
20h	Varies	Media Error Records: The list of media error records.

The Media Error Records returned here are also used for the Get Poison List command (see [Section 8.2.9.9.4.1](#)) and are defined in [Table 8-140](#).

8.2.9.9.5 Sanitize

8.2.9.9.5.1 Sanitize (Opcode 4400h)

Sanitize the device to securely repurpose or decommission it. This is done by ensuring that all user data and metadata, whether it resides in persistent capacity, volatile capacity, or the label storage area, is made permanently unavailable by whatever means is appropriate for the media type. The exact method used to sanitize is vendor specific. Sanitize also deletes all event logs on the device. Sanitize does not reset any internal usage statistics or counters and shall not artificially prolong the life of the device in any way. Unlike Secure Erase, which erases data by changing encryption keys, a successful Sanitize command ensures that no user data is available, encrypted or otherwise.

Once the Sanitize command has started successfully, the device shall be placed in the media disabled state. If the command fails or is interrupted by a reset or power failure, it shall remain in the media disabled state until a successful Sanitize command has been completed. In this state, the Media Status field in the Memory Device Status register will indicate 11b (Disabled), all memory writes to the device will have no effect, and all memory reads will return random values (no user data returned, even for

locations that the failed Sanitize operation didn't sanitize yet). Mailbox commands shall still be processed in the disabled state, except that commands that access Sanitized areas shall fail with the Media Disabled error code (Get/Set LSA, for example).

Prior to using the sanitize command, any security applied to the user data areas of the device shall be disabled.

This command does not have any input or output payloads.

In support of confidential computing, if the device has been locked while utilizing secure CXL TSP interfaces, the device shall reject any attempt to change the data or TE State on the device by returning Invalid Security State status for this command. See [Section 11.5](#) for details on locking a device and locked device behavior.

Possible Command Return Codes:

- Success
- Unsupported
- Background Command Started
- Internal Error
- Retry Required
- Invalid Payload Length
- Media Disabled
- Busy
- Aborted
- Invalid Security State

Command Effects:

- Immediate Data Change
- Security State Change
- Background Operation
- Request Abort Background Operation Command Supported

[8.2.9.9.5.2 Secure Erase \(Opcode 4401h\)](#)

Erase user data by changing the media encryption keys for all user data areas of the device.

Prior to using the secure erase command, any security applied to the user data areas of the device shall be disabled or unlocked.

This command does not have any input or output payloads.

In support of confidential computing, if the device has been locked while utilizing secure CXL TSP interfaces, the device shall reject any attempt to change the data or TE State on the device by returning Invalid Security State status for this command. See [Section 11.5](#) for details on locking a device and locked device behavior.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required

- Invalid Payload Length
- Media Disabled
- Invalid Security State

Command Effects:

- Immediate Data Change
- Security State Change

8.2.9.9.5.3 Media Operation (Opcode 4402h)

This command runs an operation on ranges of media. An output payload is provided only if defined for the specified operation.

The device shall respond with Invalid Input if the DPA range list refers to invalid media locations (e.g., blocks that are not present in the host's DC extent list), the Start DPA is not aligned to the DCD's DPA range granularity, the Length is not a multiple of the DPA range granularity, and/or an unsupported operation has been specified.

Changes to the contents of a cacheline whose coherency is managed under the HDM-DB model shall cause BISnp requests to be generated if required per [Section 2.2.1](#).

Possible Command Return Codes:

- Unsupported
- Background Command Started
- Invalid Input
- Internal Error
- Retry Required
- Busy

Command Effects:

- Immediate Data Change
- Background Operation

Table 8-147. Media Operation Input Payload

Byte Offset	Length in Bytes	Description
00h	1	Media Operation Class: This field identifies the Class of a media operation. See Table 8-149 .
01h	1	Media Operation Subclass: This field identifies the media operation together with the Media Operation Class. See Table 8-149 .
02h	2	Reserved
04h	4	DPA Range Count: The number of entries in the list of DPA ranges. A value of 0 for this field is valid and indicates that the DPA Range List parameter is not included in the request payload.
08h	Varies	DPA Range List: List of DPA ranges on which to run the operation as defined in Table 8-148 .
Varies	Varies	Operation-specific Arguments: Additional arguments, as specified by Operation definition.

Table 8-148. DPA Range Format

Byte Offset	Length in Bytes	Description
00h	8	Starting DPA <ul style="list-style-type: none"> Bits[5:0]: Reserved Bits[63:6]: DPA[63:6]
08h	8	Length: Expressed as number of bytes.

Table 8-149. Media Operations Classes and Subclasses

Class	Subclass	Description
00h – General	00h – Discovery	<p>The discovery operation returns a list of all Media Operations that the device supports, with the exception of the Discovery operation (Class=0, Subclass=0). Requests shall always specify a DPA Range Count of 0.</p> <p>The input payload is defined in Table 8-147. This operation defines operation-specific arguments as defined in Table 8-150.</p> <p>The output payload is defined in Table 8-151. The device shall return Invalid Input if DPA Range Count is not 0 or if Start Index exceeds the maximum list length of supported Media Operations.</p>
01h – Sanitize	00h – Sanitize	<p>Device shall sanitize all media in the specified DPA ranges as a result of this request using the method described in Section 8.2.9.9.5.1.</p> <p>The input payload is defined in Table 8-147. No operation-specific arguments are defined.</p> <p>No output payload is provided.</p>
	01h – Zero	<p>Device shall write 0s to all addresses spanning the DPA ranges such that a subsequent read from the host would return 0.</p> <p>The input payload is defined in Table 8-147. No operation-specific arguments are defined.</p> <p>No output payload is provided.</p>

Table 8-150. Discovery Operation-specific Arguments

Byte Offset	Length in Bytes	Description
0h	2	Start Index: Index of the first Media Operation to return.
2h	2	Number of Supported Operations: Maximum number of Media Operation to return.

Table 8-151. Media Operations Output Payload – Discovery Operation

Byte Offset	Length in Bytes	Description
00h	8	DPA Range Granularity: Minimum granularity in bytes at which the DPA ranges can be specified for Media Operations. Must be a power of 2 and a multiple of 40h.
08h	2	Total Number of Supported Operations: Total number of operations that the device supports.
0Ah	2	Number of Supported Operations: Number of entries in the Supported Operations List .
0Ch	Varies	Supported Operations List: List of supported Media Operations, as defined in Table 8-152 .

Table 8-152. Supported Operations List Entries

Byte Offset	Length in Bytes	Description
0h	1	Media Operation Class: This field identifies the Class of a media operation, as defined in Table 8-149 .
1h	1	Media Operation Subclass: This field identifies the media operation together with the Media Operation Class, as defined in Table 8-149 .

8.2.9.9.6**Persistent Memory Security**

Persistent Memory security is an optional feature that gates access to persistent memory with a user passphrase. When enabled, the persistent memory shall be locked on a Conventional Reset until the user passphrase is supplied with the Unlock command. When the persistent memory is locked, any commands that require access to the media shall return the Invalid Security State return code.

A master passphrase may optionally be supported to passphrase secure erase the persistent memory and disable security should the user passphrase be lost.

8.2.9.9.6.1**Get Security State (Opcode 4500h)**

This command retrieves the current persistent memory security state.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-153. Get Security State Output Payload

Byte Offset	Length in Bytes	Description
00h	4	Security State: Describes the current persistent memory security state. <ul style="list-style-type: none"> • Bit[0]: User Passphrase Set: The user passphrase is set. Persistent memory security is enabled. • Bit[1]: Master Passphrase Set: The master passphrase is set. • Bit[2]: Locked: The persistent memory is currently locked. • Bit[3]: Frozen: No changes can be made to the persistent memory security state of the device until a Cold Reset. • Bit[4]: User Passphrase Attempt Count Reached: An incorrect user passphrase was supplied three times in a row. A Cold Reset is required to reset the user passphrase attempt count. Until then, commands that require a user passphrase shall return Invalid Security State. • Bit[5]: Master Passphrase Attempt Count Reached: An incorrect master passphrase was supplied three times in a row. A Cold Reset is required to reset the master passphrase attempt count. Until then, commands that require a master passphrase shall return Invalid Security State. • Bits[31:6]: Reserved.

8.2.9.9.6.2 Set Passphrase (Opcode 4501h)

This command sets or changes the user or master passphrase. When the user passphrase is set, the device persistent memory shall be locked on Conventional Reset until the user passphrase is supplied. When the master passphrase is set, the master passphrase may be used to passphrase secure erase the device if the user passphrase is lost. The master passphrase shall only be set in the security disabled state when the user passphrase is not set.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Incorrect Passphrase
- Busy
- Invalid Security State

Command Effects:

- Security State Change

Table 8-154. Set Passphrase Input Payload

Byte Offset	Length in Bytes	Description
00h	1	Passphrase Type: Specifies the type of passphrase supplied in the input payload. <ul style="list-style-type: none"> • 00h = Master passphrase • 01h = User passphrase • All other encodings are reserved
01h	1Fh	Reserved
20h	20h	Current Passphrase: The current passphrase. Ignored if the passphrase is not currently set.
40h	20h	New Passphrase: The new passphrase.

8.2.9.9.6.3 Disable Passphrase (Opcode 4502h)

This command disables the user or master passphrase. When the user passphrase is disabled, the device persistent memory shall not be locked on Conventional Reset. When the master passphrase is disabled, the device shall return Invalid Input for the Passphrase Secure Erase command with the master passphrase.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Incorrect Passphrase

- Busy
- Invalid Security State

Command Effects:

- Security State Change

Table 8-155. Disable Passphrase Input Payload

Byte Offset	Length in Bytes	Description
00h	1	Passphrase Type: Specifies the type of passphrase supplied in the input payload. <ul style="list-style-type: none"> • 00h = Master passphrase • 01h = User passphrase • All other encodings are reserved
01h	1Fh	Reserved
20h	20h	Current Passphrase: The current passphrase.

8.2.9.9.6.4 Unlock (Opcode 4503h)

This command supplies the user passphrase to unlock the device persistent memory.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required
- Invalid Payload Length
- Incorrect Passphrase
- Busy
- Invalid Security State

Command Effects:

- Security State Change

Table 8-156. Unlock Input Payload

Byte Offset	Length in Bytes	Description
00h	20h	Current Passphrase: The current user passphrase.

8.2.9.9.6.5 Freeze Security State (Opcode 4504h)

This command prevents changes to persistent memory security state until a Cold Reset. In the frozen security state, the Set Passphrase, Disable Passphrase, Unlock, and Passphrase Secure Erase commands shall return Invalid Security State. This command does not have any input or output payloads.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required

- Invalid Payload Length
- Invalid Security State

Command Effects:

- Security State Change

8.2.9.9.6.6 Passphrase Secure Erase (Opcode 4505h)

This command erases the device persistent memory by changing the media encryption keys. The user passphrase shall be disabled after secure erase, but the master passphrase, if set, shall be unchanged.

In support of confidential computing, if the device has been locked while utilizing secure CXL TSP interfaces, the device shall reject any attempt to change the data or TE State on the device by returning Invalid Security State status for this command. See [Section 11.5](#) for details on locking a device and locked device behavior.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Media Disabled
- Incorrect Passphrase
- Invalid Security State

Command Effects:

- Immediate Data Change
- Security State Change

Table 8-157. Passphrase Secure Erase Input Payload

Byte Offset	Length in Bytes	Description
00h	1	Passphrase Type: Specifies the type of passphrase supplied in the input payload. <ul style="list-style-type: none"> • 00h = Master passphrase • 01h = User passphrase • All other encodings are reserved
01h	1Fh	Reserved
20h	20h	Current Passphrase: The current passphrase. If Passphrase Type is 01h (User Passphrase) and the user passphrase is not currently set or is not supported by the device, this value is ignored. If Passphrase Type is 00h (Master Passphrase), the master passphrase is required.

8.2.9.9.7 Security Passthrough

CXL devices may support security protocols defined in other industry specifications. The Security Send and Security Receive commands provide a transport interface to pass through security protocol data to the device.

8.2.9.9.7.1 Security Send (Opcode 4600h)

The Security Send command is used to transfer security protocol data to the device. The data structure transferred to the device as part of this command contains security protocol specific commands to be performed by the device. The data structure transferred may also contain data or parameters associated with the security protocol commands. Status and data that is to be returned to the host for the security protocol commands submitted by a Security Send command are retrieved with the Security Receive command.

The association between a Security Send command and subsequent Security Receive command is Security Protocol field dependent as defined in the Security Features for SCSI Commands (SFSC). Available from webstore.ansi.org.

In support of confidential computing, if the device has been locked while utilizing secure CXL TSP interfaces, the device shall reject any attempt to send security commands that could change the data or TE State on the device by returning Invalid Security State status for this command. See [Section 11.5](#) for details on locking a device and locked device behavior.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length
- Busy
- Invalid Security State

Command Effects:

- Security State Change

Table 8-158. Security Send Input Payload

Byte Offset	Length in Bytes	Description
00h	1	Security Protocol: Identifies the security command protocol interface.
01h	2	SP Specific: Contains bits 15:0 of the Security Protocol Specific Field defined in SFSC.
03h	5	Reserved
08h	Varies	Data

8.2.9.9.7.2 Security Receive (Opcode 4601h)

The Security Receive command transfers the status and data result of one or more Security Send commands that were previously submitted to the device.

The association between a Security Receive command and previous Security Send command is dependent on the Security Protocol. The format of the data to be transferred is dependent on the Security Protocol. Refer to SFSC for Security Protocol details.

Each Security Receive command returns the appropriate data corresponding to a Security Send command as defined by the rules of the Security Protocol. The Security Receive command data may or may not be retained if there is a loss of communication between the device and the host, or if a device Conventional Reset occurs.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-159. Security Receive Input Payload

Byte Offset	Length in Bytes	Description
00h	1	Security Protocol: Identifies the security command protocol interface.
01h	2	SP Specific: Contains bits 15:0 of the Security Protocol Specific Field defined in SFSC.
03h	5	Reserved

Table 8-160. Security Receive Output Payload

Byte Offset	Length in Bytes	Description
00h	Varies	Data

8.2.9.9.8 SLD QoS Telemetry

These commands enable system software to manage QoS Telemetry features in SLDs.

8.2.9.9.8.1 Get SLD QoS Control (Opcode 4700h)

This command retrieves the SLD's QoS control parameters, as defined in [Table 8-161](#). This command is mandatory if the Egress Port Congestion Supported bit or the Temporary Throughput Reduction Supported bit is set (see [Table 8-127](#)).

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-161. Get SLD QoS Control Output Payload and Set SLD QoS Control Input Payload

Byte Offset	Length in Bytes	Description
00h	1	QoS Telemetry Control: Default is 00h. <ul style="list-style-type: none">• Bit[0]: Egress Port Congestion Enable (see Section 3.3.4.3.4)• Bit[1]: Temporary Throughput Reduction Enable (see Section 3.3.4.3.5)• Bits[7:2]: Reserved
01h	1	Egress Moderate Percentage: Threshold in percent for the Egress Port Congestion mechanism to indicate moderate congestion. Valid range is 1-100. Default is 10.
02h	1	Egress Severe Percentage: Threshold in percent for the Egress Port Congestion mechanism to indicate severe congestion. Valid range is 1-100. Default is 25.
03h	1	Backpressure Sample Interval: Interval in ns for the Egress Port Congestion mechanism to take samples. Valid range is 0-15. Default is 8-ns sample interval, which corresponds to 800 ns of history. Value of 0 disables the mechanism. See Section 3.3.4.3.6 .

8.2.9.9.8.2 Set SLD QoS Control (Opcode 4701h)

This command sets the SLD's QoS control parameters. The input payload is defined in [Table 8-161](#). The device must complete the set operation before returning the response. The command response does not return a payload. This command will fail, returning Invalid Input, if any of the parameters are outside their valid range.

This command is mandatory if the Egress Port Congestion Supported bit or the Temporary Throughput Reduction Supported bit is set (see [Table 8-127](#)).

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- Immediate Policy Change

8.2.9.9.8.3 Get SLD QoS Status (Opcode 4702h)

This command retrieves the SLD's QoS status as defined in [Table 8-162](#).

This command is mandatory if the Egress Port Congestion Supported bit is set (see [Table 8-127](#)).

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-162. Get SLD QoS Status Output Payload

Byte Offset	Length in Bytes	Description
00h	1	Backpressure Average Percentage: Current snapshot of the measured Egress Port average congestion (see Section 3.3.4.3.9).

8.2.9.9.9 Dynamic Capacity

See [Section 9.13.3](#) for an overview of Dynamic Capacity Device (DCD).

8.2.9.9.9.1 Get Dynamic Capacity Configuration (Opcode 4800h)

Retrieve the DC Region configuration of the device. The returned Region Base and Length allow the device to report the DPA range that the host shall account for when programming the HDM decoders for Dynamic Capacity.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error

Command Effects:

- None

Table 8-163. Get Dynamic Capacity Configuration Input Payload

Byte Offset	Length in Bytes	Description
0h	1	Region Count: The maximum number of region configuration structures to return in the output payload.
1h	1	Starting Region Index: The zero-based index of the first requested region.

Table 8-164. Get Dynamic Capacity Configuration Output Payload (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	1	Number of Available Regions: The device shall report the total number of available Dynamic Capacity Regions. Each region may be unconfigured or configured with a different block size and capacity. A DCD shall report between 1 and 8 regions. All other values are reserved.
01h	1	Regions Returned: The number of region configurations returned in this payload.
02h	6	Reserved
08h	Varies	Region Configuration Structure: The current configuration of the regions (see Table 8-165) starting with Starting Region Index and ending with Min(Number of Available Regions-1, Starting Region Index+Region Count-1).
Varies	4	Total Number of Supported Extents: Total number of extents that the device supports on this LD.
Varies	4	Number of Available Extents: Remaining number of extents that the device supports, as defined in Section 9.13.3.3 .

Table 8-164. Get Dynamic Capacity Configuration Output Payload (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
Varies	4	Total Number of Supported Tags: Total number of Tag values that the device supports on this LD.
Varies	4	Number of Available Tags: Remaining number of Tag values that the device supports, as defined in Section 9.13.3.3 .

Table 8-165. DC Region Configuration

Byte Offset	Length in Bytes	Description
00h	8	Region Base: The device-assigned DPA base of this region aligned to 256 MB. All extents for this region shall describe a memory range \geq the Region Base and $<$ the Region Base + Region Decode Length.
08h	8	Region Decode Length: The device-assigned number of bytes of DPA that this region consumes, in multiples of 256 MB. This specifies the size of the DPA range that should be accounted for when programming HDM decoders for this region. All extents for this region shall describe a memory range \geq the Region Base and $<$ the Region Base + Region Decode Length. Report 0 if the region is unavailable for Dynamic Capacity.
10h	8	Region Length: The actual usable length of the region in bytes. This field shall be evenly divisible by the Region Block Size and shall be \leq Region Decode Length. Report 0 if the region is unavailable for Dynamic Capacity.
18h	8	Region Block Size: The Dynamic Capacity block size for this region in bytes. All Starting DPAs in all extents for this region shall be aligned to this block size. All Lengths in all extents for this region shall be multiples of this block size. Report 0 if the region is unavailable for Dynamic Capacity. Region Block Size must be a power of 2 and a multiple of 40h.
20h	4	DSMADHandle: The CDAT DSMAD instance with which this region can be utilized. The CDAT is utilized to describe the attributes associated with each region of memory and this provides the association to a specific CDAT DSMAS entry.
24h	1	Flags <ul style="list-style-type: none"> Bit[0]: Sanitize on Release: If 1, the device has been configured such that capacity released from this region will be sanitized using the method defined in Section 8.2.9.9.5.1 before it is made available to any host. Bits[7:1]: Reserved.
25h	3	Reserved

8.2.9.9.9.2 Get Dynamic Capacity Extent List (Opcode 4801h)

Retrieve the Dynamic Capacity Extent List. Since the Extent List is DPA based, a single Extent List can describe multiple regions. The host shall retrieve the complete Extent List using this command after enabling the Dynamic Capacity feature or anytime the Dynamic Capacity Event Log overflows. The device shall return Invalid Input if the Starting Extent Index value is greater than the Total Extent Count value. The Generation Number is used by the host to detect a changing Extent List during a multi-step list retrieval. The device shall report any changes to the Extent List utilizing the Dynamic Capacity Event Record mechanism.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error

Command Effects:

- None

Table 8-166. Get Dynamic Capacity Extent List Input Payload

Byte Offset	Length in Bytes	Description
0h	4	Extent Count: The maximum number of extents to return in the output payload. The device shall not return more extents than requested, but it may return fewer extents. If this field is cleared to 0, the device shall return the Total Extent Count and Extent List Generation Number without returning any extent data.
4h	4	Starting Extent Index: Index of the first requested extent. A value of 0 specifies the first extent in the list.

Table 8-167. Get Dynamic Capacity Extent List Output Payload

Byte Offset	Length in Bytes	Description
00h	4	Returned Extent Count: The number of extents returned in the output payload. Utilized when calculating the next Starting Extent Index when calling this command multiple times to retrieve the complete list.
04h	4	Total Extent Count: The total number of extents in the list.
08h	4	Extent List Generation Number: A device-generated value that is used to indicate whether the list has changed.
0Ch	4	Reserved
10h	Varies	Dynamic Capacity Extent[]: See Table 8-51 .

8.2.9.9.9.3 Add Dynamic Capacity Response (Opcode 4802h)

In response to an Add Capacity Event Record, or multiple Add Capacity Event records grouped via the More flag (see [Table 8-50](#)), the host is expected to respond with exactly one Add Dynamic Capacity Response acknowledgment, corresponding to the order of the Add Capacity Events received. If the order does not match, the device shall return Invalid Input. The Dynamic Capacity Response acknowledgment is used to communicate to the device the explicit portions of the added Dynamic Capacity that the host is now utilizing. The acknowledgment operation may consist of multiple back-to-back Add Dynamic Capacity Response messages that are linked using the More flag in the Input Payload (see [Table 8-168](#)). For non-sharable capacity, the 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 Extents present in the Add Capacity Event Record. 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 that describes the full capacity — it shall accept all or none of the sharable capacity or the device shall return Invalid Extent List.

The device shall report Invalid Physical Address if:

- One or more extents in the updated extent list specify a DPA range that is outside the range of the Extent List contained in the Add Capacity Event Record associated with this response
- One or more extents in the updated extent list specify a DPA range that has already been added with a previous call to Add Dynamic Capacity Response

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

- Overlapping DPA ranges for multiple extents
- Starting DPA not aligned to the Region Block Size

- Length not a multiple of the Region Block Size

The device shall report Resources Exhausted if the Updated Extent List would cause the device to exceed its extent tracking ability.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Invalid Physical Address
- Invalid Extent List
- Resources Exhausted

Command Effects:

- Immediate Configuration Change

Table 8-168. Add Dynamic Capacity Response Input Payload

Byte Offset	Length in Bytes	Description
0h	4	Updated Extent List Size: The number of added Extent List entries that follow. May be 0 if no capacity is added.
4h	1	Flags <ul style="list-style-type: none"> Bit[0]: More: <ul style="list-style-type: none"> 0 = This is the last (or only) payload associated with this operation. 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. Bits[7:1]: Reserved
5h	3	Reserved
8h	Varies	Updated Extent List: List of Updated Extents. See Table 8-169 for format of each entry. This list may be empty.

Table 8-169. Updated Extent

Byte Offset	Length in Bytes	Description
00h	8	Starting DPA: The address of the first block for this extent. <ul style="list-style-type: none"> Bits[5:0]: Reserved Bits[63:6]: DPA[63:6]
08h	8	Length: The number of contiguous bytes for this extent. Length shall be > 0.
10h	8	Reserved

8.2.9.9.9.4 Release Dynamic Capacity (Opcode 4803h)

Release Dynamic Capacity back to the device. This may be in response to a Release Capacity Event Record, a Forced Release Capacity Event Record, or an unsolicited release of capacity that is not associated with an event. This command is only called if the host has capacity to release to the device and the updated Extent List in the input payload describes the portions of the capacity that the host has released.

The device shall report Invalid Physical Address if:

- One or more extents in the updated extent list specify a DPA range that is out of range of a current Extent List maintained by the device
- One or more extents in the updated extent list specify a DPA range that has already been released with a previous call to Release Dynamic Capacity

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

- Overlapping DPA range 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

In support of confidential computing, if the device has been locked while utilizing secure CXL TSP interfaces, the device shall overwrite or cryptographically scramble the data and reset the TE State to 0 for all memory contained in the released capacity extent list, before the device allows the released memory capacity to be added to another host. This prevents a host from injecting data that could be silently consumed by another host when Dynamic Capacity is moved. See [Section 11.5](#) for details on locking a device and locked device behavior.

The device shall report Resources Exhausted if the Updated Extent List would cause the device to exceed its extent tracking ability.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Invalid Physical Address
- Invalid Extent List
- Resources Exhausted

Command Effects:

- Immediate Configuration Change

Table 8-170. Release Dynamic Capacity Input Payload

Byte Offset	Length in Bytes	Description
0h	4	Updated Extent List Size: The number of host-released Extent List entries that follow. Shall be > 0.
4h	1	Flags <ul style="list-style-type: none"> • Bit[0]: More: <ul style="list-style-type: none"> — 0 = This is the last (or only) payload associated with this operation. — 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. • Bits[7:1]: Reserved
5h	3	Reserved
8h	Varies	Updated Extent List[]: The extents that have been released by the host. For format, see Table 8-169 .

8.2.9.9.10 GFD Component Management Command Set

8.2.9.9.10.1 Identify GFD (Opcode 4900h)

This command retrieves the number of supported Memory Groups, total Dynamic Capacity of the device, the volatile and non-volatile dynamic capacities, the number of Device Media Partitions (DMPs; see [Section 8.2.9.9.10.6](#)), and supported DC Region configurations. In addition, a GFD will return the currently available capacity and media type details.

A GFD has only one DPA space and one set of DC Regions, thus every host and the FM see the same values. A GFD also has DMPs that are mapped to this one DPA space and associated with the one set of DC Regions. Thus, the GFD FM API enables the FM to establish the relationships of DMPs and DC Regions to the one DPA space.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required

Command Effects:

- None

Table 8-171. Identify GFD Response Payload (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	1	Number of Supported Device Media Partitions (DMPs): Device shall report the total number of available DMPs. GFDs shall report between 1 and 4 available DMPs. All other encodings are reserved.
01h	1	Number of Supported DC Regions: Device shall report the total number of Dynamic Capacity Regions available per host. GFDs shall report between 1 and 8 regions. All other encodings are reserved.
02h	2	Number of Supported Memory Group IDs (GrpIDs): Device shall report the total number of GrpIDs that the device supports.
04h	2	<ul style="list-style-type: none"> • Bits[3:0]: Supported Add Capacity Selection Policies: Bitmask that specifies the selection policies, as defined in Section 7.6.7.6.5, that the device supports when capacity is added. At least one policy shall be supported. A value of 1 indicates that a policy is supported, and a value of 0 indicates that a policy is not supported: <ul style="list-style-type: none"> — Bit[0]: Free — Bit[1]: Contiguous — Bit[2]: Prescriptive — Bit[3]: Must be 0 • Bits[15:4]: Reserved
06h	1	QoS Telemetry Capability: Optional QoS Telemetry for memory GFD capabilities for management by an FM (see Section 3.3.4): <ul style="list-style-type: none"> • Bit[0]: Egress Port Congestion Supported: When set, the associated feature is supported and the Get QoS Status command must be implemented (see Section 3.3.4.3.8) • Bit[1]: Temporary Throughput Reduction Supported: When set, the associated feature is supported (see Section 3.3.4.3.5) • Bits[7:2]: Reserved
07h	1	Reserved

Table 8-171. Identify GFD Response Payload (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
08h	2	<ul style="list-style-type: none"> Bits[1:0]: Supported Release Capacity Removal Policies: Bitmask that specifies the removal policies, as defined in Section 7.6.7.6.5, that the device supports when capacity is released. At least one policy shall be supported. A value of 1 indicates that a policy is supported, and a value of 0 indicates that a policy is not supported: <ul style="list-style-type: none"> — Bit[0]: Tag-based — Bit[1]: Prescriptive Bits[15:2]: Reserved
0Ah	1	Sanitize on Release Configuration Support Mask : Bitmask, where bit position corresponds to region number, indicating whether the Sanitize on Release capability is configurable (1) or not configurable (0) for that region.
0Bh	1	Reserved
0Ch	8	Total Dynamic Capacity : Total memory media capacity of the device available for dynamic assignment to a memory group in multiples of 256 MB.
14h	8	Region 0 Supported Block Size Mask : Indicates the block sizes that the region supports. Each bit indicates a power of 2 supported block size, where bit n being set indicates that block size 2^n is supported. Bits[5:0] and bits[63:52] shall be 0. At least one block size shall be supported.
1Ch	8	Region 1 Supported Block Size Mask : As defined in Region 0 Supported Block Size Mask. Valid only if Number of Supported Regions > 1.
24h	8	Region 2 Supported Block Size Mask : As defined in Region 0 Supported Block Size Mask. Valid only if Number of Supported Regions > 2.
2Ch	8	Region 3 Supported Block Size Mask : As defined in Region 0 Supported Block Size Mask. Valid only if Number of Supported Regions > 3.
34h	8	Region 4 Supported Block Size Mask : As defined in Region 0 Supported Block Size Mask. Valid only if Number of Supported Regions > 4.
3Ch	8	Region 5 Supported Block Size Mask : As defined in Region 0 Supported Block Size Mask. Valid only if Number of Supported Regions > 5.
44h	8	Region 6 Supported Block Size Mask : As defined in Region 0 Supported Block Size Mask. Valid only if Number of Supported Regions > 6.
4Ch	8	Region 7 Supported Block Size Mask : As defined in Region 0 Supported Block Size Mask. Valid only if Number of Supported Regions > 7.
54h	8	DMP 0 Supported Block Size Mask : Indicates the block sizes that DMP 0 supports. Each bit indicates a power of 2 supported block size, where bit n being set indicates that block size 2^n is supported. Bits[5:0] and bits[63:52] shall be 0. At least one block size shall be supported.
5Ch	8	DMP 1 Supported Block Size Mask : As defined in DMP 0 Supported Block Size Mask. Valid only if Number of Supported DMPs > 1.
64h	8	DMP 2 Supported Block Size Mask : As defined in DMP 0 Supported Block Size Mask. Valid only if Number of Supported DMPs > 2.
6Ch	8	DMP 3 Supported Block Size Mask : As defined in DMP 0 Supported Block Size Mask. Valid only if Number of Supported DMPs > 3.
74h	8	Volatile Only Capacity : This field indicates the total usable capacity of the device that may be used only as volatile memory. Expressed in multiples of 256 MB.
8Ch	8	Persistent Only Capacity : This field indicates the total usable capacity of the device that may be used only as persistent memory. Expressed in multiples of 256 MB.

8.2.9.9.10.2 Get GFD Status (Opcode 4901h)

This command retrieves the GFD's Status.

Possible Command Return Codes:

- Success
- Unsupported

- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 8-172. Get GFD Status Response Payload (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
0h	1	<ul style="list-style-type: none"> • Bit[0]: Device Fatal: When set, the device has encountered a fatal error. Vendor specific device replacement or recovery is recommended. • Bit[1]: FW Halt: When set, the device has encountered an FW error and is not responding. • Bits[3:2]: Media Status: Describes the status of the device media. <ul style="list-style-type: none"> — 00b = Not Ready. Media training is incomplete. — 01b = Ready. The media trained successfully and is ready for use. — 10b = Error. The media failed to train or encountered an error. — 11b = Disabled. Access to the media is disabled. If the media is not in the Ready state, user data is not accessible. • Bit[4]: Reserved. • Bits[7:5]: Reset Needed: When nonzero, indicates the least impactful reset type needed to return the device to the operational state. A cold reset is considered more impactful than a warm reset. A warm reset is considered more impactful than a hot reset, which is more impactful than a CXL reset. This field returns a nonzero value if FW Halt is set, Media Status is in the Error or Disabled state, or the Mailbox Interfaces Ready bit does not become set. <ul style="list-style-type: none"> — 000b = Device is operational and a reset is not required — 001b = Cold Reset — 010b = Warm Reset — 011b = Hot Reset — 100b = CXL Reset (device must not report this value if it does not support CXL Reset) — All other encodings are reserved
1h	1	<ul style="list-style-type: none"> • Bits[3:0]: Negotiated Link Mode: <ul style="list-style-type: none"> — 0h = Connection is not CXL or is disconnected — 1h = RCD mode — 2h = CXL 64B Flit and VH mode — 3h = Standard 256B Flit mode — 4h = CXL Latency-Optimized 256B Flit mode — 5h = PBR mode — All other encodings are reserved • Bits[7:4]: Reserved.
2h	1	<ul style="list-style-type: none"> • Bits[5:0]: Maximum Link Width: Value encoding matches the Maximum Link Width field in the PCIe Link Capabilities register in the PCIe Capability structure. • Bits[7:6]: Reserved.
3h	1	<ul style="list-style-type: none"> • Bits[5:0]: Negotiated Link Width: Value encoding matches the Negotiated Link Width field in PCIe Link Capabilities register in the PCIe Capability structure. • Bits[7:6]: Reserved.
4h	1	<ul style="list-style-type: none"> • Bits[5:0]: Supported Link Speeds Vector: Value encoding matches the Supported Link Speeds Vector field in the PCIe Link Capabilities 2 register in the PCIe Capability structure. • Bits[7:6]: Reserved.
5h	1	<ul style="list-style-type: none"> • Bits[5:0]: Max Link Speed: Value encoding matches the Max Link Speed field in the PCIe Link Capabilities register in the PCIe Capability structure. • Bits[7:6]: Reserved.
6h	1	<ul style="list-style-type: none"> • Bits[5:0]: Current Link Speed: Value encoding matches the Current Link Speed field in the PCIe Link Status register in the PCIe Capability structure. • Bits[7:6]: Reserved.

Table 8-172. Get GFD Status Response Payload (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
7h	1	<p>LTSSM State: Current link LTSSM Major state:</p> <ul style="list-style-type: none"> • 00h = Detect • 01h = Polling • 02h = Configuration • 03h = Recovery • 04h = L0 • 05h = L0s • 06h = L1 • 07h = L2 • 08h = Disabled • 09h = Loopback • 0Ah = Hot Reset • All other encodings are reserved <p>Link substates should be reported through vendor-defined diagnostics commands.</p>
8h	1	First Negotiated Lane Number
9h	2	<p>Link State Flags</p> <ul style="list-style-type: none"> • Bit[0]: Lane Reversal State: <ul style="list-style-type: none"> — 0 = Standard lane ordering — 1 = Reversed lane ordering • Bit[1]: Port PCIe Reset State (PERST#): <ul style="list-style-type: none"> — 0 = Not in reset — 1 = In reset • Bit[2]: Port Presence Pin State (PRSNT#): <ul style="list-style-type: none"> — 0 = Not present — 1 = Present • Bit[3]: Power Control State: <ul style="list-style-type: none"> — 0 = Power on — 1 = Power off • Bits[15:4]: Reserved

8.2.9.9.10.3 Get GFD DC Region Configuration (Opcode 4902h)

This command retrieves the GFD's Dynamic Capacity Region configurations.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 8-173. Get GFD DC Region Configuration Request Payload

Byte Offset	Length in Bytes	Description
0h	2	Reserved
2h	1	Region Count: Maximum number of region configurations to return in the response payload.
3h	1	Starting Region Index: Index of the first requested region.

Table 8-174. Get GFD DC Region Configuration Response Payload

Byte Offset	Length in Bytes	Description
0h	2	Reserved
2h	1	Number of Available Regions: As defined in Get Dynamic Capacity Configuration Output Payload.
3h	1	Number of Regions Returned: Number of entries in the Region Configuration List.
4h	Varies	Region Configuration List: GFD DC Region Info for region specified via Starting Region Index input field. The format of each entry is defined in Table 8-175 .
Varies	4	Total Number of Supported Extents: Total number of extents that the device supports.
Varies	4	Number of Available Extents: Remaining number of extents that the device supports, as defined in Section 9.13.3.3 .
Varies	4	Total Number of Supported Tags: Total number of Tag values that the device supports.
Varies	4	Number of Available Tags: Remaining number of Tag values that the device supports, as defined in Section 9.13.3.3 .

Table 8-175. GFD DC Region Configuration

Byte Offset	Length in Bytes	Description
00h	8	Region Base: As defined in Table 8-165 .
08h	8	Region Decode Length: As defined in Table 8-165 .
10h	8	Region Length: As defined in Table 8-165 .
18h	8	Region Block Size: As defined in Table 8-165 .
20h	1	<p>Note: More than one bit may be set at a time.</p> <ul style="list-style-type: none"> Bits[1:0]: Reserved Bit[2]: NonVolatile: As defined in the Flags field of Device Scoped Memory Affinity Structure defined in CDAT Specification Bit[3]: Sharable: As defined in the Flags field of Device Scoped Memory Affinity Structure defined in CDAT Specification Bit[4]: Hardware Managed Coherency: As defined in the Flags field of Device Scoped Memory Affinity Structure defined in CDAT Specification Bit[5]: Interconnect specific Dynamic Capacity Management: As defined in the Flags field of Device Scoped Memory Affinity Structure defined in CDAT Specification Bit[6]: Read-Only: As defined in the Flags field of Device Scoped Memory Affinity Structure defined in CDAT Specification Bit[7]: Reserved
21h	3	Reserved
24h	1	<ul style="list-style-type: none"> Bit[0]: Sanitize on Release: As defined in Table 8-165 Bits[7:1]: Reserved
25h	3	Reserved

8.2.9.9.10.4 Set GFD DC Region Configuration (Opcode 4903h)

This command sets the configuration of a DC Region. For a GFD, this command will set not only the block size, but the region base DPA, the region length, and the region decode length because these parameters are tied to the GFD's Device Memory Partitions.

This command shall fail with Unsupported when the **Sanitize on Release** field does not match the region's configuration, as reported from the **Get GFD DC Region Configuration**, and the device does not support reconfiguration of the **Sanitize on Release** setting, as advertised by the **Sanitize on Release Configuration Support Mask** in the **Identify GFD** response payload.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- Configuration Change after Cold Reset
- Immediate Configuration Change
- Immediate Data Change

Table 8-176. Set GFD DC Region Configuration Request Payload

Byte Offset	Length in Bytes	Description
00h	1	Region ID : Specifies which region to configure. Valid range is from 0 to 7.
01h	3	Reserved
04h	8	Region Block Size : As defined in Table 8-165 .
0Ch	1	<ul style="list-style-type: none"> • Bit[0]: Sanitize on Release: As defined in Table 8-165 • Bits[7:1]: Reserved
0Dh	3	Reserved
10h	8	Region Base : As defined in Table 8-165 .
18h	8	Region Decode Length : As defined in Table 8-165 .
20h	8	Region Length : As defined in Table 8-165 .

8.2.9.9.10.5 Get GFD DC Region Extent Lists (Opcode 4904h)

This command retrieves the Dynamic Capacity Extent List for a specified Memory Group ID (GrpID). This command shall fail with 'Invalid Input' if proxied from a host with no access rights to the specified Memory Group.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 8-177. Get GFD DC Region Extent Lists Request Payload

Byte Offset	Length in Bytes	Description
0h	2	GrpID: GFD's Memory Group ID for which the Extent List is requested.
2h	2	Reserved
4h	4	Extent Count: Maximum number of extents to return in the output payload. The device may not return more extents than requested; however, the device can return fewer extents. 0 is valid and allows the FM to retrieve the Total Extent Count and Extent List Generation Number without retrieving any extent data.
8h	4	Starting Extent Index: Index of the first requested extent. A value of 0 will retrieve the first extent in the list.

Table 8-178. Get GFD DC Region Extent Lists Response Payload

Byte Offset	Length in Bytes	Description
00h	2	GrpID: GFD's Memory Group ID for which the Extent List is requested.
02h	2	Reserved
04h	4	Starting Extent Index: Index of the first extent in the list.
08h	4	Returned Extent Count: Number of extents returned in Extent List[].
0Ch	4	Total Extent Count: Total number of extents in the list for the given GrpID.
10h	4	Extent List Generation Number: Device-generated value that is used to indicate that the list has changed.
14h	4	Reserved
18h	Varies	Extent List[]: Extent list for the specified Memory Group as defined in Table 8-51 .

8.2.9.9.10.6 Get GFD DMP Configuration (Opcode 4905h)

This command retrieves the GFD's Device Memory Partition configurations. A GFD's DMP configuration details are a mix of attributes that are set by the device based on media attributes and quantities, and those that are set by the FM to organize the available media capacities into associated DPA ranges.

On a GFD, DC Region block size and memory attributes shall align with those of the DMP that sources the media capacity. To which DMP a given DC Region is associated is determined by the respective DPA ranges claimed.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 8-179. Get GFD DMP Configuration Request Payload

Byte Offset	Length in Bytes	Description
0h	2	Reserved
2h	1	DMP Count: Maximum number of DMP configurations to return in the response payload. Value should be >0 and less than (Max DMPs Supported).
3h	1	Starting DMP Index: Index of the first requested DMP. An index of 0 shall return the configuration of the 1 st DMP.

Table 8-180. Get GFD DMP Configuration Response Payload

Byte Offset	Length in Bytes	Description
0h	2	Reserved
2h	1	Number of Available DMPs: As defined in Get GFD DCD Info response.
3h	1	Number of DMP Configurations Returned: The number of entries in the DMP Configuration List.
4h	Varies	DMP Configuration List[]: DMP Info for region specified via Starting DMP Index input field. The format of each entry is defined in Table 8-181 .

Table 8-181. GFD DMP Configuration

Byte Offset	Length in Bytes	Description
00h	8	DMP Base: As defined for DC Regions in Table 8-165 .
08h	8	DMP Decode Length: As defined for DC Regions in Table 8-165 .
10h	8	DMP Length: As defined for DC Regions in Table 8-165 (expected to be the same as DMP Decode Length).
18h	8	DMP Block Size: As defined for DC Regions in Table 8-165 .
20h	1	<p>Note: More than one bit may be set at a time.</p> <ul style="list-style-type: none"> • Bits[1:0]: Reserved • Bit[2]: NonVolatile: As defined in the Flags field of Device Scoped Memory Affinity Structure defined in Coherent Device Attribute Table (CDAT) Specification • Bit[3]: Sharable: As defined in the Flags field of Device Scoped Memory Affinity Structure defined in CDAT Specification • Bit[4]: Hardware Managed Coherency: As defined in the Flags field of Device Scoped Memory Affinity Structure defined in CDAT Specification • Bit[5]: Interconnect specific Dynamic Capacity Management: As defined in the Flags field of Device Scoped Memory Affinity Structure defined in CDAT Specification • Bit[6]: Read-Only: As defined in the Flags field of Device Scoped Memory Affinity Structure defined in CDAT Specification • Bit[7]: Reserved
21h	1	DMP Index: Index of this DMP.
22h	2	Reserved
24h	1	<ul style="list-style-type: none"> • Bit[0]: Sanitize on Release: As defined in Table 8-165 • Bits[7:1]: Reserved
25h	3	Reserved

8.2.9.9.10.7 Set GFD DMP Configuration (Opcode 4906h)

This command sets the configuration of a DMP. For a GFD, this command will set not only the block size, but the DMP base DPA, the region length, and the region decode length. This command does not set the media attributes as those are supplied by the device. This command is intended for use only at initial configuration of a GFD, or when the target DMP's current claimed DPA range is not in use (no valid bits set in the DMP's associated GDT entries).

This command shall fail with Invalid Input when the device has any defined Memory Groups active in the DMP's decoded DPA range.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- Immediate Configuration Change

Table 8-182. Set GFD DMP Configuration Request Payload

Byte Offset	Length in Bytes	Description
0h	2	Reserved
2h	1	DMP Count : Number of DMP configurations that are described in the DMP_List[]. Value should be >0 and not more than (Max DMPs Supported).
3h	1	Starting DMP Index : Index of the first requested DMP. An index of 0 shall return the configuration of the 1 st DMP.
4h	Varies	DMP Configuration List[] : Info for DMP specified via Starting DMP Index input field. The format of each entry is defined in Table 8-181 .

8.2.9.9.10.8 GFD Dynamic Capacity Add (Opcode 4907h)

This command adds Dynamic Capacity to the specified Memory Group ID (GrpID). It returns the list of all DC Extents that are mapped to the Memory Group ID.

A Selection Policy is specified to govern the device's selection of which memory resources to add:

- **Free**: Unassigned extents are selected by the device, with no requirement for contiguous blocks
- **Contiguous**: Unassigned extents are selected by the device and shall be contiguous
- **Prescriptive**: Extent list of capacity to assign is included in the request payload
- **Enable Shared Access**: Enable access to extent(s) previously added to another Memory Group in a region that reports the "Sharable" flag, as designated by the specified tag value

The FM might use the following example sequence to set up sharing of GFD Memory Groups between hosts:

- Issue GFD Dynamic Capacity Add Request with the GrpID for the associated Memory Group. The DC Region number must correspond to a region that is advertised as sharable and has the required memory attributes.
- If the above request is successful, all allowed hosts can be enabled for access to the new memory group, using the **Set GFD SAT** commands (see [Section 7.7.2.5](#)).
- The **Set GFD SAT** commands enable access by specific hosts to specific Memory Groups. The FM may enable the GFD to send a GFD Asynchronous Message VDM to the impacted hosts upon completion of a **Set GFD SAT** command. This notification causes each impacted host to reread the associated Extent Lists and thus learn of newly shared Extents.

If the FM is adding capacity to an already defined Memory Group, the current users (if any) of the Group are not automatically notified of a change in capacity. The FM can trigger such notifications from the GFD to the impacted hosts by reissuing the Set GFD SAT Entry command for each host with the Notify RPID bit set to 1 (see [Section 8.2.9.9.10.14](#)).

The FM might use the following example sequence to allocate a set of tagged capacity and allow it to be initialized by a host in one Memory Group and then shared with one or more hosts as read-only in another sharable region.

1. Issue a GFD Dynamic Capacity Add with the Selection Policy set to Free or Contiguous or Prescriptive, with the Memory Group ID associated with the first host. The region number must correspond to a region that is advertised as writable and sharable.
2. If the above request is successful, the tagged sharable capacity can be initialized by the first host.
3. Issue a GFD Dynamic Capacity Add Reference for the tag associated with the capacity. Holding this Reference prevents the tagged capacity from being freed and sanitized in step 4.
4. After the first host has initialized the tagged sharable capacity, issue GFD Dynamic Capacity Release for the tag associated with the capacity. The capacity associated with the Tag is preserved but is not accessible from any Memory Groups.
5. Issue GFD Dynamic Capacity Add with Selection Policy set to Enable Shared Access with the Memory Group ID associated with the second host, specifying a region that is Sharable and read-only. The Tag field must match the Tag value used in step 1.
6. Any other hosts that need to share the tagged capacity can be added to the Memory Group specified in step 5.
7. Issue a GFD Dynamic Capacity Remove Reference to remove the FM reference to the tagged capacity.
8. To withdraw the shared capacity, issue a GFD Dynamic Capacity Release command for the Memory Group.
9. When the tagged capacity has been released from all hosts, if the FM does not hold a reference, the tagged capacity will be sanitized (if appropriate) and freed, at which point the tag no longer exists and the capacity is available for future use.

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

- When the command is sent with an invalid Memory Group ID, or an invalid region number, or an unsupported Selection Policy
- When the Length field is not a multiple of the Block size

The command, with selection policy Enable Shared Access, shall also fail with Invalid Input under the following conditions:

- When the specified region is not Sharable
- When the tagged capacity is already mapped to any Host Group via a non-Sharable region
- When the tagged capacity cannot be added to the requested region due to device-imposed restrictions
- When the same tagged capacity is currently accessible by the same Memory Group

The command shall fail with Retry Required when the device determines that its FM CCI interface is already busy and cannot queue this command for execution at this time. (Forward progress of retried commands is beyond the scope of this specification.)

The command shall fail with Invalid Extent List under the following conditions:

- When the Selection Policy is set to Prescriptive and the Extent Count is invalid
- When the Selection Policy is set to Prescriptive and any of the DPAs are already accessible to the same Host Group

The command shall fail with Resources Exhausted under the following conditions:

- 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.
- If the total capacity requested cannot be added. The GFD shall not partially fulfill a GFD Dynamic Capacity Add request. These requests are 'All or Nothing' in nature, and the device shall atomically update impacted Extent Lists and Memory Groups upon success or leave them unchanged upon failure.
- There is insufficient contiguous space to satisfy a request with Selection Policy set to Contiguous.

The device shall report Resources Exhausted if the Updated Extent List would cause the device to exceed its extent tracking ability.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Extent List
- Resources Exhausted

Command Effects:

- Immediate Configuration Change

Table 8-183. GFD Dynamic Capacity Add Request Payload

Byte Offset	Length in Bytes	Description
00h	2	GrpID: Memory Group ID to which the requested capacity is to be added. If the GrpID is new and not in use, the device creates the Memory Group and attempts to assign the requested capacity to it. If the GrpID already exists, the device attempts to increase the existing capacity accordingly. The GrpID values start at 0, and continue consecutively up to (maximum number of supported Memory Group IDs – 1). All other GrpID values are reserved.
02h	1	<ul style="list-style-type: none"> Bits[3:0]: Selection Policy: Specifies the policy to use for selecting which extents comprise the added capacity: <ul style="list-style-type: none"> 0h = Free 1h = Contiguous 2h = Prescriptive 3h = Enable Shared Access All other encodings are reserved Bits[7:4]: Reserved
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.
04h	8	Length: Number of bytes of capacity to add. Always a multiple of the configured Region Block Size returned in Get DCD Info. Shall be > 0. This field is reserved when the Selection Policy is set to Prescriptive or Enable Shared Access.
0Ch	10h	Tag: Context field utilized by implementations that make use of the Dynamic Capacity feature. This field is reserved when the Selection Policy is set to Prescriptive.
1Ch	4	Extent Count: Number of extents in the Extent List. Present only when the Selection Policy is set to Prescriptive.
20h	4	Max Response Extent List Size: Maximum number of Extent List members to be returned in the response. The FM may use this field to limit the size of the response message.
24h	Varies	Extent List: Extent list of capacity to add as defined in Table 8-51 . Present only when the Selection Policy is set to Prescriptive.

Table 8-184. GFD Dynamic Capacity Add Response Payload

Byte Offset	Length in Bytes	Description
00h	2	GrpID: Memory Group ID to which the requested capacity was to be added.
02h	2	Reserved
04h	4	Starting Extent Index: Index of the first extent in the list.
08h	4	Returned Extent Count: Number of extents returned in Extent List[].
0Ch	4	Total Extent Count: Total number of extents in the list for that Memory Group. If the Total Extent Count is greater than the Returned Extent Count, the requester must use the Get GFD DC Region Extent Lists command to obtain the full list.
10h	4	Extent List Generation Number: Device-generated value that is used to indicate that the list has changed.
14h	4	Reserved
18h	Varies	Extent List[]: Extent list for the specified Memory Group, as defined in Table 8-51 . Unless limited by the requester to Max Response Extent List Size, this Extent List[] shall contain all capacity that is currently assigned to the Memory Group ID (GrpID).

8.2.9.9.10.9 GFD Dynamic Capacity Release (Opcode 4908h)

This command initiates the release of Dynamic Capacity from a GFD Memory Group. It returns the list of all DC Extents that are mapped to the Memory Group ID after this command was executed.

The FM is responsible for negotiating with all impacted hosts through out-of-band channels (beyond the scope of this specification) about the release of a DCD capacity in a GFD. After all impacted consumers have given up access attempts to the capacity to be reclaimed:

7. FM determines which Memory Group has capacity that needs to be reclaimed. The capacity of a Memory Group can be reduced, but only by reclaiming entire Extents. Further, if Extents are tagged (as they should be for all GFD allocations), tagged capacity should be reclaimed all at once. Thus, the FM must describe the capacity to be reclaimed as one or more sets of tagged extents.
8. Given the desired tagged extents, the FM must now negotiate with all impacted hosts through out-of-band channels (beyond the scope for this specification) about the release of any DCD capacity in a GFD.
9. After all impacted consumers have given up access attempts to the capacity to be reclaimed, the FM can issue the appropriate GFD Dynamic Capacity Release command for the targeted Memory Group with the list of tagged extents.
10. GFD will send the list of tagged extents that remain associated with the Memory Group in the response to the FM.
11. FM shall examine the remaining Extents of the Memory Group to validate that the correct capacity reduction has occurred. Because the FM had previously recalled all impacted capacity from the working images of the hosts, there should be no subsequent requests for these DPAs; thus, any capacity remaining within the Memory Group can still be in use.
12. If the Memory Group is devoid of any remaining capacity, the FM should remove all impacted hosts from the Memory Groups enabled list.
13. If the Memory Group has no remaining capacity, and no enabled RPIDs, the associated GrpID is automatically reclaimed to the 'unused' list by the GFD.

The FM uses the following sequence to remove a host's access to shared GFD Memory Groups:

1. FM disables hosts' accesses to given Memory Groups, utilizing the Set GFD SAT command (see [Section 8.2.9.9.10.14](#)).
2. Set GFD SAT command enables or disables access by specific hosts, but it does not automatically notify the hosts that their access rights or any DCD capacity changes have been made. The FM can enable the GFD to send such notifications by setting the Notify RPID bit in the command request.

A removal policy is specified to govern the device's selection of which memory resources to remove:

- **Tag-based:** Extents are selected by the device based on tag, with no requirement for contiguous extents
- **Prescriptive:** Extent list of capacity to release is included in request payload

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

- When the command is sent with an invalid GrpID, 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

The command shall fail with Invalid Extent List under the following conditions:

- When the Removal Policy is set to Prescriptive and the Extent Count is invalid
- When the Removal Policy is set to Prescriptive and any extents are part of sharable, tagged capacity
- When the Extent List includes blocks that are not currently assigned to the region

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 device shall report Resources Exhausted if the Updated Extent List would cause the device to exceed its extent tracking ability.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Extent List
- Resources Exhausted

Command Effects:

- Configuration Change after Cold Reset
- Immediate Configuration Change
- Immediate Data Change

Table 8-185. Initiate Dynamic Capacity Release Request Payload (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	2	GrpID: the Memory Group ID from which the capacity is being released.
02h	1	Flags <ul style="list-style-type: none"> • Bits[3:0]: Removal Policy: Specifies the policy to use for selecting which extents comprise the released capacity: <ul style="list-style-type: none"> — 0h = Tag-based — 1h = Prescriptive — All other encodings are reserved • Bit[4]: Reserved • Bit[5]: Sanitize on Release: <ul style="list-style-type: none"> — 1 = Device shall sanitize all released capacity as a result of this request using the method described in Section 8.2.9.9.5.1. If this is a shared capacity, the sanitize operation shall be performed after the last host has released the capacity. • Bits[7:6]: Reserved
03h	1	Reserved
04h	8	Length: Number of bytes of capacity to remove. Always a multiple of the configured Region Block Size returned in Get GFD DCD Info. Shall be > 0. This field is reserved when the Removal Policy is set to Prescriptive.
0Ch	10h	Tag: Tag associated with the capacity to be released, if the Removal Policy is set to Tag-Based. This field is reserved when the Removal Policy is set to Prescriptive.

Table 8-185. Initiate Dynamic Capacity Release Request Payload (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
1Ch	4	Extent Count: Number of extents in the Extent List. Present only when the Removal Policy is set to Prescriptive.
20h	4	Max Response Extent List Size: Maximum number of Extent List members to be returned in the response. The FM may use this field to limit the size of the response message.
24h	Varies	Extent List: Extent list of capacity to release as defined in Table 8-51 . Present only when the Removal Policy is set to Prescriptive.

Table 8-186. GFD Dynamic Capacity Release Response Payload

Byte Offset	Length in Bytes	Description
00h	2	GrpID: The Memory Group ID to which the requested capacity was to be released.
02h	2	Reserved
04h	4	Starting Extent Index: Index of the first extent in the list.
08h	4	Returned Extent Count: Number of extents returned in Extent List[].
0Ch	4	Total Extent Count: Total number of extents in the list for that Memory Group. If the Total Extent Count is greater than the Returned Extent Count, the requester must use the Get GFD DC Region Extent Lists command to obtain the full list.
10h	4	Extent List Generation Number: Device-generated value that is used to indicate that the list has changed.
14h	4	Reserved
18h	Varies	Extent List[]: Extent list for the specified Memory Group, as defined in Table 8-51 . Unless limited by the requester to Max Response Extent List Size, this Extent List[] shall contain all capacity that is currently assigned to the Memory Group ID (GrpID).

8.2.9.9.10.10GFD Dynamic Capacity Add Reference (Opcode 4909h)

This command prevents the tagged sharable capacity from being sanitized, freed, or reallocated, regardless of whether it is currently visible to any host groups via extent lists. The tagged capacity will remain allocated, and contents will be preserved even if all DCD Extents that reference it are removed.

This command has no effect and will return Success if the FM has already added a reference to the tagged capacity.

This command shall return Invalid Input if the Tag in the payload does not match an existing sharable tag.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- Configuration Change after Cold Reset
- Configuration Change after Conventional Reset

- Configuration Change after CXL Reset
- Immediate Configuration Change

Table 8-187. GFD Dynamic Capacity Add Reference Request Payload

Byte Offset	Length in Bytes	Description
0h	10h	Tag: Tag that is associated with the memory capacity to be preserved.

8.2.9.9.10.11GFD Dynamic Capacity Remove Reference (Opcode 490Ah)

This command removes a reference to tagged sharable capacity that was previously added via GFD Dynamic Capacity Add Reference (see [Section 8.2.9.9.10.10](#)). If there are no remaining extent lists that reference the tagged capacity, the memory will be freed and sanitized if appropriate.

This command shall return Invalid Input if the Tag in the payload does not match an existing sharable tag.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- Configuration Change after Cold Reset (if freed)
- Configuration Change after Conventional Reset (if freed)
- Configuration Change after CXL Reset (if freed)
- Immediate Configuration Change (if freed)

Table 8-188. GFD Dynamic Capacity Remove Reference Request Payload

Byte Offset	Length in Bytes	Description
0h	10h	Tag: Tag that is associated with the memory capacity.

8.2.9.9.10.12GFD Dynamic Capacity List Tags (Opcode 490Bh)

This command allows an FM to re-establish context by receiving a list of all existing tags and a flag indicating whether the FM holds a reference.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 8-189. GFD Dynamic Capacity List Tags Request Payload

Byte Offset	Length in Bytes	Description
0h	4	Starting Index: Index of the first tag to return.
4h	4	Max Tags: Maximum number of tags to return in the response payload. If max tags is 0, no tags list will be returned; however, the Generation Number shall be valid.

Table 8-190. GFD Dynamic Capacity List Tags Response Payload

Byte Offset	Length in Bytes	Description
0h	4	Generation Number: Generation number of the tags list. This number shall change every time the remainder of the command's payload would change.
4h	4	Total Number of Tags: Maximum number of tags to return in the response payload.
8h	4	Reserved
Ch	Varies	Tags List: List of GFD Dynamic Capacity Tag Information structures. The format of each entry is defined in Table 8-191 .

Table 8-191. GFD Dynamic Capacity Tag Information

Byte Offset	Length in Bytes	Description
00h	10h	Tag: Tag that is associated with the memory capacity.
10h	1	Flags <ul style="list-style-type: none"> Bit[0]: FM Holds Reference: When set, this bit indicates that the FM holds a reference on this Tag Bits[7:1]: Reserved
11h	3	Reserved

8.2.9.9.10.13 Get GFD SAT Entry (Opcode 490Ch)

This command retrieves the GFD's SPID Access Table (SAT) entry for a specific RPID. The RPID is the PID that defines a CXL.mem request initiator. For a host, the RPID is the host's Edge Switch's USP PID. Note that a host and its associated CXL devices will each have a unique RPID, and therefore each will use a different entry in the GDT.

The SAT entry is an N-bit vector (GrpAccVec). Each bit_k of the GrpAccVec is the enable access control for a specific Memory Group ID (GrpID). Each memory request initiator is given access to a specific memory group number GrpID by setting bit_k of the SAT vector, where $k=\text{GrpID}$. An SAT entry is therefore a list of Memory Groups to which a given memory requester (known by the given RPID) has access.

An FM enables multiple hosts and other requesters to share access to a given Memory Group by setting the corresponding Memory Group Mask bit in each requester's SAT entry.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error

- Retry Required

Command Effects:

- None

Table 8-192. Get GFD SAT Entry Request Payload

Byte Offset	Length in Bytes	Description
0h	2	Reserved
2h	2	SAT Entry Count: Number of SAT entries to return in the response payload. Value should be >0 and not greater than 4096.
4h	2	Starting RPID Index: Index of the first requested SAT entry. An index of 0 shall return the SAT for RPID = 0.

Table 8-193. Get GFD SAT Entry Response Payload

Byte Offset	Length in Bytes	Description
0h	2	Reserved
2h	2	Number of SAT Entries Returned: Number of entries in the SAT Entry List.
4h	2	Starting RPID Index: Index of the first SAT entry in the SAT_Entry_List[]. An index of 0 shall return the SAT for SPID = 0.
5h	Varies	SAT_Entry List[]: List of SAT entries requested via request. The format of each entry is defined in Table 8-194 .

Table 8-194. GFD SAT Entry Format

Byte Offset	Length in Bytes	Description
00h	4	<p>GrpAccVec: NG-bit permission mask for a given RPID. NG = Number of Memory Groups supported by this device.</p> <p>Note: More than one bit may be set at a time.</p> <ul style="list-style-type: none"> • Bits[NG-1:0]: GrpAccVec: <ul style="list-style-type: none"> — If GrpAccVec[k] = 0: Memory Group ID (where k = GrpID) is not accessible to requesters that use the RPID associated with this SAT entry — If GrpAccVec[k] = 1: Memory Group ID (where k = GrpID) is accessible with full RW access to requesters that use the RPID associated with this SAT entry • Bits[31:NG]: Reserved <p>Note: There is no mechanism to select read-only access on a per-RPID basis using the SAT entries. Writes to writable regions will be allowed, but writes to read-only regions will be dropped.</p>

8.2.9.9.10.14 Set GFD SAT Entry (Opcode 490Dh)

This command updates the GFD's SPID Access Table (SAT) entry for a specific RPID. The RPID is the PID that defines a CXL.mem request initiator. For a host, the RPID is the host's Edge Switch's USP PID. Note that a host and its associated CXL devices will each have a unique RPID, and therefore each will use a different entry in the GDT.

The SAT entry is an N-bit vector (GrpAccVec). Each bit_k of the vector is the enable access control for a specific Memory Group ID (GrpID). Each memory request initiator is given access to a specific memory group number GrpID by setting bit_k of the SAT vector, where k=GrpID. An SAT entry is therefore a list of Memory Groups to which a given memory requester (known by the given RPID) has access. The FM grants specific memory requesters access to different Memory Groups by setting the corresponding Memory Group Mask bit in that requester's SAT entry.

An FM enables multiple hosts and other requesters to share access to a given Memory Group by setting the corresponding GrpAccVec bit in each requester's SAT entry.

The FM may enable the GFD to send a GFD Asynchronous Message VDM to the impacted hosts upon completion of a Set GFD SAT command. This notification causes each impacted host to reread the associated Extent Lists and thus learn of changed permissions.

This command shall return Invalid Input when any bit GrpAccVec[k] is set when k is larger than (max Memory Groups -1) because this attempts to enable a Memory Group that cannot exist.

The device shall make single GrpAccVec updates atomically; the single GrpAccVec update is completely successful, or the failed attempt has no impact on the original GrpAccVec contents. If more than one GrpAccVec update is requested (i.e., more than one SPID's GrpAccVec is requested), each may atomically succeed, or fail. However, this command shall not return Success if multiple SAT GrpAccVec updates are requested and any one of them fails for any reason.

In the event of an unsuccessful request of a multiple GrpAccVec update request, the FM should perform an appropriate Get GFD SAT Entry request to determine which of the requested updates were successful, and which were unsuccessful.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- Immediate Configuration Change

Table 8-195. Set GFD SAT Entry Request Payload

Byte Offset	Length in Bytes	Description
0h	2	SAT Update Count: Number of SAT entries to be updated by this request. Value should be >0 and not greater than 4096, and shall further be limited by the maximum size of request payload that the device will accept.
2h	Varies	SAT Update List[]: List of SAT updates to be made. The format of each entry is defined in Table 8-196 . Note: The SAT update list format for a Set command is different than the SAT_Entry_List of the Get command.

Table 8-196. GFD SAT Update Format

Byte Offset	Length in Bytes	Description
0h	2	<ul style="list-style-type: none"> Bits[11:0]: RPID: Requester PID to which the following GrpAccVec is to be assigned Bits[15:12]: Reserved
2h	4	<p>GrpAccVec: NG-bit permission mask for a given RPID. NG = Number of Memory Groups supported by this device.</p> <p>Note: More than one bit may be set at a time.</p> <ul style="list-style-type: none"> Bits[NG-1:0]: GrpAccVec: <ul style="list-style-type: none"> If GrpAccVec[k] = 0: Memory Group ID (where k = GrpID) is not accessible to requesters that use the RPID associated with this SAT entry If GrpAccVec[k] = 1: Memory Group ID (where k = GrpID) is accessible with full RW access to requesters that use the RPID associated with this SAT entry Bits[30:NG]: Reserved Bit[31]: Notify RPID: If set to 1, the GFD shall send a GFD Asynchronous Message VDM to the RPID whose GrpAccVec is being updated <p>Note: There is no mechanism to select read-only access on a per-RPID basis using the SAT entries. Writes to writable regions will be allowed, but writes to read-only regions will be dropped.</p>

8.2.9.9.10.15Get GFD QoS Control (Opcode 490Eh)

This command retrieves the GFD's QoS control parameters.

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-197. QOS Payload for Get GFD QoS Control Response, Set GFD QoS Control Request, and Set GFD QoS Control Response

Byte Offset	Length in Bytes	Description
0h	1	<p>QoS Telemetry Control: Default is 00h.</p> <ul style="list-style-type: none"> Bit[0]: Egress Port Congestion Enable: See Section 3.3.4.3.8 Bit[1]: Temporary Throughput Reduction Enable: See Section 3.3.4.3.5 Bits[7:2]: Reserved
1h	1	Egress Moderate Percentage : Threshold in percent for Egress Port Congestion mechanism to indicate moderate congestion. Valid range is 1-100. Default is 10.
2h	1	Egress Severe Percentage : Threshold in percent for Egress Port Congestion mechanism to indicate severe congestion. Valid range is 1-100. Default is 25.
3h	1	Backpressure Sample Interval : Interval in ns for Egress Port Congestion mechanism to take samples. Valid range is 0-15. Default is 8 (800 ns of history for 100 samples). Value of 0 disables the mechanism. See Section 3.3.4.3.4 .
4h	2	ReqCmpBasis : Estimated maximum sustained sum of requests and recent responses across the entire device, serving as the basis for QoS Limit Fraction. Valid range is 0-65,535. Value of 0 disables the mechanism. Default is 0. See Section 3.3.4.3.7 .
6h	1	Completion Collection Interval : Interval in ns for Completion Counting mechanism to collect the number of transmitted responses in a single counter. Valid range is 0-255. Default is 64 (1.024 us of history, given 16 counters). See Section 3.3.4.3.9 .

8.2.9.9.10.16 Set GFD QoS Control (Opcode 490Fh)

This command sets the GFD's QoS control parameters, as defined in [Table 8-197](#). The device must complete the set operation before returning the response. The command response returns the resulting QoS control parameters, as defined in the same table. This command will fail, returning Invalid Input, if any of the parameters are outside their valid range.

Possible Command Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- Immediate Policy Change

Payload for Set GFD QoS Control Request and Response is documented in [Table 8-197](#).

8.2.9.9.10.17 Get GFD QoS Status (Opcode 4910h)

This command retrieves the GFD's QoS Status. This command is mandatory if the Egress Port Congestion Supported bit is set (see [Table 8-197](#)).

Possible Command Return Codes:

- Success
- Unsupported
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-198. Get GFD QoS Status Response Payload

Byte Offset	Length in Bytes	Description
0h	1	Backpressure Average Percentage: Current snapshot of the measured Egress Port average congestion (see Section 3.3.4.3.4).

8.2.9.9.10.18 Get GFD QoS BW Limit (Opcode 4911h)

This command retrieves the GFD's QoS bandwidth limit on a per-RPID basis (see [Section 3.3.4.3.7](#)).

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input

- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- None

Table 8-199. Payload for Get GFD QoS BW Limit Request

Byte Offset	Length in Bytes	Description
0h	2	Number of RPIDs: Number of RPIDs queried. This field shall have a minimum value of 1.
2h	2	<ul style="list-style-type: none"> • Bits[11:0]: Start RPID: ID of the first RPID in the QoS BW Limit List • Bits[15:12]: Reserved

Table 8-200. Payload for Get GFD QoS BW Limit Response

Byte Offset	Length in Bytes	Description
0h	2	Number of RPIDs: Number of RPIDs queried.
2h	2	<ul style="list-style-type: none"> • Bits[11:0]: Start RPID: ID of the first RPID in the QoS BW Limit List • Bits[15:12]: Reserved
4h	Number of RPIDs	QoS Limit Fraction: Byte array of allocated bandwidth limit fractions for RPIDs, starting at Start RPID. The valid range of each array element is 0-255. Default value is 0. Value in each byte is the fraction multiplied by 256.

8.2.9.9.10.19 Set GFD QoS BW Limit (Opcode 4912h)

This command sets the GFD's QoS bandwidth limit on a per-RPID basis, as defined in [Section 3.3.4.3.7](#). The device must complete the set operation before returning the response. The command response returns the resulting QoS bandwidth limit, as defined in [Table 8-201](#). This command will fail, returning Invalid Input, if any of the parameters are outside their valid range. This command will fail, returning Internal Error, if the device was able to set the QoS BW Limit for some of the RPIDs in the request, but not all the RPIDs.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Invalid Payload Length

Command Effects:

- Configuration Change after Cold Reset
- Immediate Configuration Change
- Immediate Data Change

Table 8-201. Payload for Set GFD BW Limit Request and Set GFD QoS BW Limit Response

Byte Offset	Length in Bytes	Description
0h	2	Number of RPIDs: Number of RPIDs queried.
2h	2	<ul style="list-style-type: none"> Bits[11:0]: Start RPID: ID of the first RPID in the QoS BW Limit List Bits[15:12]: Reserved
4h	Number of RPIDs	QoS Limit Fraction: Byte array of allocated bandwidth limit fractions for RPIDs, starting at Start RPID. The valid range of each array element is 0-255. Default value is 0. Value in each byte is the fraction multiplied by 256.

8.2.9.9.10.20Get GDT Configuration (Opcode 4913h)

This command retrieves the Host's GDT decoder configurations. The HPA base address, the DPA base address, and the decoded length establish the Host's chosen HPA-to-DPA mapping.

Note:

The Host ID included in the CCI request and response is the PBR_ID for the SPID that an actual memory request will contain.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required

Command Effects:

- None

Table 8-202. Get GDT Configuration Request Payload

Byte Offset	Length in Bytes	Description
0h	2	<ul style="list-style-type: none"> Bits[11:0]: RVID: PID of the host's Edge USP, or the PBR_ID of an accelerator functioning as a host. This is the PBR_ID that will be listed as the SPID in a memory request being validated against these GDT entries. Bits[15:12]: Reserved.
2h	2	Reserved
4h	1	GDT Count: Number of GDT configurations to return in the response payload. The device may not return more GDT configurations than requested; however, it can return fewer configurations. 0 is valid and allows the Host to retrieve the Total GDT Count without retrieving any GDT config data. Shall not be greater than 8.
5h	1	Starting GDT Index: Index of the first requested GDT configuration. A value of 0 will retrieve the first GDT entry in the table.

Table 8-203. Get GDT Configuration Response Payload

Byte Offset	Length in Bytes	Description
0h	2	<ul style="list-style-type: none"> Bits[11:0]: RVID: PID of the host's Edge USP, or the PBR_ID of an accelerator functioning as a host (see Section 7.7.2). This is the PBR_ID that will be listed as the SPID in a memory request being validated against these GDT entries. Bits[15:12]: Reserved.
2h	2	Reserved
4h	1	Starting GDT Index : Index of the first GDT configuration in the GDT.
5h	1	Returned GDT Count : Number of entries returned in GDT_Entry List[].
6h	1	Total GDT Count : Total number of available GDT entries.
7h	1	Reserved
8h	Varies	GDT Entry List[] : List of GDT Entries as defined in Table 8-204 .

Table 8-204. GDT Entry Format

Byte Offset	Length in Bytes	Description
00h	1	GDT Index : Entry number of this configuration description within the GDT.
01h	1	V : Valid entry setting: <ul style="list-style-type: none"> Bit[0]: <ul style="list-style-type: none"> 1 = Configuration Data in entry is valid 0 = Configuration Data in entry is invalid Bits[7:1]: Reserved
02h	1	Coh : Hardware Coherency Enabled setting: <ul style="list-style-type: none"> Bit[0]: <ul style="list-style-type: none"> 1 = Hardware Coherency is enabled 0 = Hardware Coherency is disabled Bits[7:1]: Reserved
03h	1	Reserved
04h	8	HPABase : The 64-bit host physical address that forms the base of the HPA range claimed by this decoder. The HPA base must be aligned with the block size of the DC Region being claimed by this decoder entry. <ul style="list-style-type: none"> Bits[5:0]: Reserved Bits[63:6]: HPA[63:6]
0Ch	8	DPABase : DPA of the first byte of the range claimed by this decoder. The DPA base must be aligned with the block size of the DC Region being claimed by this decoder entry. <ul style="list-style-type: none"> Bits[5:0]: Reserved Bits[63:6]: DPA[63:6]
14h	8	DPALen : Length of the DPA range claimed by this decoder: <ul style="list-style-type: none"> Bits[5:0]: Reserved Bits[63:6]: Size of decoded range
1Ch	1	Intlv : Number of interleave ways: <ul style="list-style-type: none"> Bits[3:0]: Intlv value, as defined in Table 7-82 Bits[7:4]: Reserved
1Dh	1	Gran : Granularity of the interleaving: <ul style="list-style-type: none"> Bits[3:0]: Gran value, as defined in Table 7-83 Bits[7:4]: Reserved
1Eh	1	Way : This GFD is 'way' or member number of the interleave set.
1Fh	1	Reserved

8.2.9.9.10.21 Set GDT Configuration (Opcode 4914h)

This command sets the Host's GDT decoder configurations. The HPA base address, the DPA base address, and the decoded length establish the Host's chosen HPA-to-DPA mapping.

The device shall report Unsupported if the Coh (hardware coherency enable) bit is set for a decoder range that is not mapped to a DC Region that supports hardware coherency.

The device shall report Invalid Input if it detects:

- Overlapping Starting DPA and Lengths for multiple GDT entries
- Starting DPA not aligned to the GDT Block Size
- Length not a multiple of the GDT Block Size

The device shall report Resources Exhausted if the number of GDT entries submitted exceed the number of GDT decoders that are available to this host.

Possible Command Return Codes:

- Success
- Unsupported
- Invalid Input
- Internal Error
- Retry Required
- Resources Exhausted

Command Effects:

- Immediate Configuration Change

Table 8-205. Set GDT Configuration Request Payload

Byte Offset	Length in Bytes	Description
0h	2	<ul style="list-style-type: none"> • Bits[11:0]: RVID: PID of the host's Edge USP, or the PBR_ID of an accelerator functioning as a host (see Section 7.7.6). This is the PBR_ID that will be listed as the SPID in a memory request being validated against these GDT entries. • Bits[15:12]: Reserved.
2h	2	Reserved
4h	1	GDT Count : Number of GDT configurations contained in this request payload. Count shall be greater than 0. Shall not be greater than 8.
5h	1	Starting GDT Index : Index of the first GDT to be configured. A value of 0 will configure the first GDT entry in the table.
6h	Varies	GDT_Entry List[] : List of GDT entry configurations for the specified host, formatted as defined in Table 8-204 .

8.2.9.9.11 Memory Device Features

8.2.9.9.11.1 Device Patrol Scrub Control Feature

The UUID of this feature is defined in [Table 8-206](#).

The Device Patrol Scrub proactively locates and makes corrections to errors during normal device operation. The patrol scrub control allows the requester to configure patrol scrub configurations during system boot or at runtime.

The patrol scrub control feature allows the requester to specify the number of hours during which the patrol scrub cycles must be completed, provided that the requested number is not less than the minimum number of hours for the patrol scrub cycle of which the device is capable. In addition, the patrol scrub controls allow the host to disable and enable Patrol Scrub. When the patrol scrub cycle duration is reported to the host via the Get Feature, the number of hours for the configured patrol scrub cycle returns the default value or the current value depending on the Selection field.

For memory errors that are detected during the patrol scrub, the CXL device shall generate either the General Media Event Record or the DRAM Event Record when the number of memory errors detected exceeds the configured threshold. Handling of both uncorrectable errors and corrected errors are expected by the patrol scrub. For the General Media Event Record, the Memory Event Type field is set to Scrub Media ECC error (04h). For the DRAM Event Record, the Memory Event Type field is set to Scrub Media ECC error (01h). In both cases, Transaction Type field is set to Media Patrol Scrub (05h).

[Table 8-206](#) shows the information returned in the Get Supported Features output payload for the Device Patrol Scrub Control Feature. Some feature attributes are changeable.

Table 8-206. Supported Feature Entry for the Device Patrol Scrub Control Feature

Byte Offset	Length in Bytes	Attribute	Description
00h	10h	Feature Identifier	96dad7d6-fde8-482b-a733-75774e06db8a
10h	2	Feature Index	Device Specific
12h	2	Get Feature Size	4
14h	2	Set Feature Size	2
16h	4	Attribute Flags	<ul style="list-style-type: none"> • Bit[0]: 1 (Changeable) • Bits[3:1]: 000b (Deepest Reset Persistence=None. Any reset will restore the default value.) • Bit[4]: 0 (Persist across Firmware Update) • Bit[5]: 1 (Default Selection Supported) • Bit[6]: 0 (Saved Selection Supported) • Bits[31:7]: Reserved
1Ah	1	Get Feature Version	01h
1Bh	1	Set Feature Version	01h
1Ch	2	Set Feature Effects	<ul style="list-style-type: none"> • Bit[0]: 0 (Configuration Change after Cold Reset) • Bit[1]: 1 (Immediate Configuration Change) • Bit[2]: 0 (Immediate Data Change) • Bit[3]: 0 (Immediate Policy Change) • Bit[4]: Vendor-specific value (Immediate Log Change) • Bit[5]: 0 (Security State Change) • Bit[6]: 0 (Background Operation) • Bit[7]: Vendor-specific value (Secondary Mailbox Supported) • Bit[8]: 0 (Request Abort Background Operation Supported) • Bit[9]: 1 (CEL[11:10] Valid) • Bit[10]: 0 (Configuration Change after Conventional Reset) • Bit[11]: 0 (Configuration Change after CXL Reset) • Bits[15:12]: 0h
1Eh	12h	Reserved	All 0s

[Table 8-207](#) shows the output payload returned by a Get Feature command with Selection cleared to 0h (Current value) or set to 1h (Default value).

Table 8-207. Device Patrol Scrub Control Feature Readable Attributes

Byte Offset	Length in Bytes	Description
0h	1	Device Patrol Scrub Capabilities <ul style="list-style-type: none"> Bit[0]: Patrol Scrub Cycle Change Capable: If set, scrub cycle (the interval of scrub results reported) can be changed to a value other than the default value. Bit[1]: Patrol Scrub Real-time Reporting Capable: If set, event records are generated as the device finds errors. Otherwise, event records are generated at the end of the scrub cycle.
1h	2	Device Patrol Scrub Cycle Support <ul style="list-style-type: none"> Bits[7:0]: Number of Hours for the Configured Patrol Scrub Cycle: A nonzero value that represents the currently configured scrub cycle in hours. If the Set Feature Effects field in the Supported Feature Entry is set to output default selection, then the default patrol scrub cycle will be shown. If the Set Feature Effects field is set to output saved selection, then the currently saved patrol scrub cycle updated by the host request will be shown. Bits[15:8]: Smallest Number of Hours for the Patrol Scrub Cycle Supported by the Device: It indicates the shortest scrub intervals that the device can support, expressed in hours. Legal value ranges from 01h to FFh.
3h	1	Device Patrol Scrub Flags <ul style="list-style-type: none"> Bit[0]: Patrol Scrub Enabled <ul style="list-style-type: none"> 0 = Scrub is disabled 1 = Scrub is enabled Bits[7:1]: Reserved

[Table 8-208](#) shows the Feature Data for Set Feature command.

Table 8-208. Device Patrol Scrub Control Feature Writable Attributes

Byte Offset	Length in Bytes	Description
0h	1	Device Patrol Scrub Cycle Change Request <ul style="list-style-type: none"> Bits[7:0]: Number of Hours for the Configured Scrub Cycle: A nonzero value that represents the currently configured scrub cycle in hours. Only the range of numbers greater than or equal to the smallest number of hours for the Patrol Scrub Cycle supported by the device is valid; any number outside the range is invalid and the Command Return Code of Invalid Input shall be returned. This field is reserved if Patrol Scrub Cycle Change Capable bit (see Table 8-207) is cleared to 0.
1h	1	Device Patrol Scrub Flags <ul style="list-style-type: none"> Bit[0]: Patrol Scrub Enable <ul style="list-style-type: none"> 0 = Disable Scrub 1 = Enable Scrub Bits[7:1]: Reserved

8.2.9.9.11.2 DDR5 Error Check Scrub Control Feature

The UUID of this feature is defined in [Table 8-209](#).

The Error Check Scrub (ECS) is a feature defined in JEDEC DDR5 SDRAM Specification (JESD79-5) and allows the DRAM to internally read, correct single-bit errors, and write back corrected data bits to the DRAM array while providing transparency to error counts. The ECS control feature may be used to configure ECS parameters.

The ECS control allows the requester to change the ECS threshold count provided that the request is within the definition specified in DDR5 mode registers, change the mode of operation between codeword count mode and row count mode, and reset the ECS counter.

The outputs from the ECS operation including the error count and the row address with the most errors are accessible from the DDR5 ECS Log, defined in [Table 8-78](#).

DDR5 ECS is able to handle corrected errors only. When a corrected error is fixed by ECS, the CXL device shall generate a DRAM Event Record or a Generic Media Event Record with the following attributes:

- Memory Event Type = Scrub Media ECC error (01h)
- Transaction Type = Internal Media Error Check Scrub (07h)

[Table 8-209](#) shows the information returned in the Get Supported Features output payload for the DDR5 Error Check Scrub (ECS) Control Feature. Some feature attributes are changeable.

Table 8-209. Supported Feature Entry for the DDR5 ECS Control Feature

Byte Offset	Length in Bytes	Attribute	Description
00h	10h	Feature Identifier	e5b13f22-2328-4a14-b8ba-b9691e893386
10h	2	Feature Index	Device Specific
12h	2	Get Feature Size	Varies ($n*4+1$, where n is the number of memory media FRUs in the device)
14h	2	Set Feature Size	Varies ($n*2+1$, where n is the number of memory media FRUs in the device)
16h	4	Attribute Flags	<ul style="list-style-type: none"> • Bit[0]: 1 (Changeable) • Bits[3:1]: 000b (Deepest Reset Persistence=None. Any reset will restore the default value.) • Bit[4]: 0 (Persist across Firmware Update) • Bit[5]: 1 (Default Selection Supported) • Bit[6]: 0 (Saved Selection Supported) • Bits[31:7]: Reserved
1Ah	1	Get Feature Version	01h
1Bh	1	Set Feature Version	01h
1Ch	2	Set Feature Effects	<ul style="list-style-type: none"> • Bit[0]: 0 (Configuration Change after Cold Reset) • Bit[1]: 1 (Immediate Configuration Change) • Bit[2]: 0 (Immediate Data Change) • Bit[3]: 0 (Immediate Policy Change) • Bit[4]: Vendor-specific value (Immediate Log Change) • Bit[5]: 0 (Security State Change) • Bit[6]: 0 (Background Operation) • Bit[7]: Vendor-specific value (Secondary Mailbox Supported) • Bit[8]: 0 (Request Abort Background Operation Supported) • Bit[9]: 1 (CEL[11:10] Valid) • Bit[10]: 0 (Configuration Change after Conventional Reset) • Bit[11]: 0 (Configuration Change after CXL Reset) • Bits[15:12]: 0h
1Eh	12h	Reserved	All 0s

Table 8-210 shows the output payload returned by a Get Feature command with Selection cleared to 0h (Current value) or set to 1h (Default value).

Table 8-210. DDR5 ECS Control Feature Readable Attributes

Byte Offset	Length in Bytes	Description
00h	1	Common DDR5 ECS Log Capabilities <ul style="list-style-type: none"> Bits[1:0]: Log Entry Type: The log entry type of how the ECS log is reported. The entry type is defined commonly for all memory media FRUs within the device. <ul style="list-style-type: none"> — 00b = Per DRAM — 01b = Per Memory Media FRU — All other encodings are reserved Bits[7:2]: Reserved.
01h	1	Memory Media FRU 1 DDR5 ECS Capabilities <ul style="list-style-type: none"> Bit[0]: ECS Real-time Reporting Capable: If capable, scrub events are generated as the device finds errors. If not capable, reporting is available only at the end of the scrub cycle. Per definition in JESD79-5, this bit shall be cleared to 0 because ECS is not capable of real-time reporting. Bits[7:1]: Reserved.
02h	2	Memory Media FRU 1 DDR5 ECS Configurations <ul style="list-style-type: none"> Bits[2:0]: ECS Threshold Count per Gb of Memory Cells: <ul style="list-style-type: none"> — 011b = 256 (default) — 100b = 1024 — 101b = 4096 — All other encodings are reserved Bit[3]: Codeword/Row Count Mode: <ul style="list-style-type: none"> — 0 = ECS counts rows with errors — 1 = ECS counts codewords with errors Bits[15:4]: Reserved
04h	1	Memory Media FRU 1 ECS Flags <ul style="list-style-type: none"> Bits[7:0]: Reserved
...
(1+4*(n-1))	1	Memory Media FRU n DDR5 ECS Capabilities : ECS capabilities of the nth memory media FRU of the device,
(2+4*(n-1))	2	Memory Media FRU n DDR5 ECS Configurations : ECS configurations of the nth memory media FRU of the device,
(4*n)	1	Memory Media FRU n ECS Flags : ECS flags of the nth memory media FRU of the device,

Table 8-211 shows the Feature Data for Set Feature command.

Table 8-211. DDR5 ECS Control Feature Writable Attributes

Byte Offset	Length in Bytes	Description
00h	1	Common DDR5 ECS Log Capabilities <ul style="list-style-type: none"> Bits[1:0]: Log Entry Type: The log entry type of how the ECS log is reported. The entry type is defined commonly for all memory media FRUs within the device. <ul style="list-style-type: none"> — 00b = Per DRAM — 01b = Per Memory Media FRU — All other encodings are reserved Bits[7:2]: Reserved.
01h	2	Memory Media FRU 1 DDR5 ECS Configurations <ul style="list-style-type: none"> Bits[2:0]: ECS Threshold Count per Gb of Memory Cells: <ul style="list-style-type: none"> — 011b = 256 (default) — 100b = 1024 — 101b = 4096 — All other encodings are reserved Bit[3]: Codeword/Row Count Mode: <ul style="list-style-type: none"> — 0 = ECS counts rows with errors — 1 = ECS counts codewords with errors Bit[4]: ECS Reset Counter: <ul style="list-style-type: none"> — 0 = Normal, counter running actively (default) — 1 = Reset ECC counter to default value (automatically cleared to 0 by the device after the reset is complete) Bits[15:5]: Reserved
...
(1+2*(n-1))	2	Memory Media FRU n DDR5 ECS Configurations

8.2.9.9.11.3 Advanced Programmable Corrected Volatile Memory Error Threshold Feature Discovery and Configuration

The UUID of this feature is defined in Table 8-212.

Table 8-212 shows the information returned in the Get Supported Features output payload for the Advanced Programmable Corrected Volatile Memory Error (CVME) Threshold Feature. Some feature attributes are changeable. Feature attributes cannot be saved.

Table 8-212. Supported Feature Entry for the Advanced Programmable Corrected Volatile Memory Error Threshold Feature (Sheet 1 of 2)

Byte Offset	Length in Bytes	Attribute	Description
00h	10h	Feature Identifier	1478ad9d-ce00-4733-9db8-f392a4c2d0cc
10h	2	Feature Index	Device Specific
12h	2	Get Feature Size	20h
14h	2	Set Feature Size	20h
16h	4	Attribute Flags	<ul style="list-style-type: none"> • Bit[0]: 1 (Changeable) • Bits[3:1]: 000b (Deepest Reset Persistence=None. Any reset will restore the default value.) • Bit[4]: 0 (Persist across Firmware Update) • Bit[5]: 1 (Default Selection Supported) • Bit[6]: 0 (Saved Selection Supported) • Bits[31:7]: Reserved
1Ah	1	Get Feature Version	01h
1Bh	1	Set Feature Version	01h

Table 8-212. Supported Feature Entry for the Advanced Programmable Corrected Volatile Memory Error Threshold Feature (Sheet 2 of 2)

Byte Offset	Length in Bytes	Attribute	Description
1Ch	2	Set Feature Effects	<ul style="list-style-type: none"> Bit[0]: 0 (Configuration Change after Cold Reset) Bit[1]: 1 (Immediate Configuration Change) Bit[2]: 0 (Immediate Data Change) Bit[3]: 0 (Immediate Policy Change) Bit[4]: Vendor-specific value (Immediate Log Change) Bit[5]: 0 (Security State Change) Bit[6]: 0 (Background Operation) Bit[7]: Vendor-specific value (Secondary Mailbox Supported) Bit[8]: 0 (Request Abort Background Operation Supported) Bit[9]: 1 (CEL[11:10] Valid) Bit[10]: 0 (Configuration Change after Conventional Reset) Bit[11]: 0 (Configuration Change after CXL Reset) Bits[15:12]: 0h
1Eh	12h	Reserved	All 0s

Table 8-213 shows the output payload returned by a Get Feature command with Selection cleared to 0h (Current value) or set to 1h (Default value).

Table 8-213. Advanced Programmable Corrected Volatile Memory Error Threshold Feature Readable Attributes (Sheet 1 of 4)

Byte Offset	Length in Bytes	Description
00h	1	<p>Supported Corrected Volatile Memory Error (CVME) Threshold Granularity Flags</p> <ul style="list-style-type: none"> Bit[0]: Per Memory Media FRU Granularity: If set, the device implements a dedicated CVME counter for each memory media FRU (e.g., counter per DIMM). The device shall generate a memory media event record when the CVME counter for a given memory media FRU exceeds a programmed threshold. Bit[1]: Per Rank Granularity: If set, the device implements a dedicated CVME counter for each rank of media on the device. The device shall generate a memory media event record when the CVME counter for a given rank exceeds a programmed threshold. Bits[7:2]: Reserved. <p>All memory devices support Full HDM Range Granularity as available by the Set Alert Configuration command.</p>
01h	1	<p>Configured Corrected Volatile Memory Error (CVME) Threshold Granularity</p> <ul style="list-style-type: none"> 00h = Full HDM Range Granularity. When used, the programmable thresholds shall cover the entire range of HDM on the device. The device shall maintain a global CVME counter for the entire device HDM. When the global CVME counter exceeds a programmed threshold, the device shall generate a memory media event record. 01h = Per Memory Media FRU Granularity. When used, the programmable thresholds are at a per-memory media FRU granularity (e.g., threshold per DIMM) across all memory media FRUs. The device shall maintain a separate CVME counter for each memory media FRU connected to the device. When the CVME counter for a given memory media FRU exceeds a programmed threshold, the device shall generate a memory media event record. 02h = Per Rank Granularity. When used, the programmable thresholds are at a per-rank of device granularity (e.g., threshold per rank) across all ranks. The device shall maintain a separate CVME counter for each rank of media on the device. When the CVME counter for a given rank exceeds a programmed threshold, the device shall generate a memory media event record. All other encodings are reserved.

Table 8-213. Advanced Programmable Corrected Volatile Memory Error Threshold Feature Readable Attributes (Sheet 2 of 4)

Byte Offset	Length in Bytes	Description
02h	1	<p>Supported Corrected Volatile Memory Error (CVME) Threshold Configuration Flags</p> <ul style="list-style-type: none"> Bit[0]: Corrected Error Type Filtering Supported: When set, the device is capable of filtering out corrected single bit errors and corrected multi-bit errors from its CVME counters used for advanced thresholding. Bit[1]: Patrol Scrub CVME Threshold Supported: When set, the device is capable of maintaining separate counters and thresholds for CVMEs detected during patrol scrub accesses. CVME counters used by Patrol Scrub CVME Thresholds are automatically reset at the end of every Patrol Scrub cycle. Bits[7:2]: Reserved. <p>Flags in this field apply to all severity thresholds (e.g., Informational, Warning, etc.). Feature does not support individual filtering per severity threshold.</p>
03h	1	<p>Configured Corrected Volatile Memory (CVME) Error Threshold Configuration Flags</p> <ul style="list-style-type: none"> Bit[0]: Single Bit Error Mask: When set, the device shall not include single bit errors in the corrected volatile memory errors count for the advanced programmable volatile memory error thresholds and subsequent memory media events associated with those thresholds. Bit[1]: Corrected Multi Bit Error Mask: When set, the device shall not include corrected multi-bit errors in the corrected volatile memory errors count for the advanced programmable volatile memory error thresholds and subsequent memory media events associated with those thresholds. Bit[2]: Enable Patrol Scrub CVME Threshold: When set, the device shall maintain separate counters and thresholds for CVMEs detected during patrol scrub. CVME counters used by Patrol Scrub CVME Thresholds are automatically reset at the end of every Patrol Scrub cycle. When cleared, the device shall combine CVME counts from patrol scrub and non-patrol scrub accesses. Additionally, when cleared, the device shall ignore all other parameters related to Patrol Scrub CVME Thresholds. Bit[3]: Enable Counter Expiration: When set, the device shall expire and restart all CVME counters used by the Advanced Programmable CVME Threshold Feature after a configured time duration. Counter expiration does not apply to Patrol Scrub CVME counters if Patrol Scrub Thresholds are enabled. If Patrol Scrub Thresholds are disabled, CVMEs counted by the patrol scrubber will reset alongside non-patrol scrub CVMEs when the counter expires. Bit[4]: Enable Counter Expiration Reporting: When set, the device shall generate a media event record each time the CVME counter has expired. Media event records are created with Informational severity and a flag to indicate Threshold Expiration. Counter expiration must be enabled and its timer input correctly configured for the device to report counter expiration. If the CVME count is 0 at the time of counter expiration, the device shall not generate a media event record. Use of this flag and a low expiration timer may yield a large number of events generated. Bits[7:5]: Reserved. <p>Flags in this field apply to all severity thresholds (e.g., Informational, Warning, etc.). Feature does not support individual filtering per severity threshold. Enabling all filters may still result in corrected error counting and thresholding for other error types not filterable by this feature.</p>
04h	3	<p>Configured Corrected Volatile Memory Error Counter Expiration Timer: Time in seconds after which the device shall expire and restart all corrected volatile memory error programmable counters. An input of 00 0000h shall render the expiration timer as disabled. Counter expiration does not apply to Patrol Scrub CVME counters if Patrol Scrub Thresholds are enabled.</p>

Table 8-213. Advanced Programmable Corrected Volatile Memory Error Threshold Feature Readable Attributes (Sheet 3 of 4)

Byte Offset	Length in Bytes	Description
07h	1	<p>Configured Corrected Volatile Memory Error Threshold Event Records Flags: Enabling an advanced CVME threshold shall cause the device to automatically disable any CVME thresholds set by other methods (such as the one set by the Set Alert Configuration command). Those CVME thresholds are not automatically re-enabled whenever advanced CVME thresholds are disabled. Enabling other CVME thresholds shall cause the device to automatically disable any advanced CVME thresholds. Advanced CVME thresholds also do not automatically re-enable when other CVME thresholds are disabled.</p> <p>If Patrol Scrub CVME Thresholds are enabled, the settings in this parameter shall only apply to non-patrol scrub CVMEs counters and thresholds.</p> <ul style="list-style-type: none"> • Bit[0]: Enable Programmable Informational Threshold: When set, the device shall enable the advanced programmable informational threshold and generate any informational memory media event records associated with that threshold. • Bit[1]: Enable Programmable Warning Threshold: When set, the device shall enable the advanced programmable warning threshold and generate any warning memory media event records associated with that threshold. • Bit[2]: Enable Programmable Failure Threshold: When set, the device shall enable the advanced programmable failure threshold and generate any failure memory media event records associated with that threshold. • Bit[3]: Enable Programmable Hardware Replacement Flags for Warning Events: When set, the device shall generate memory media events caused by the advanced programmable warning threshold using the 'Hardware Replacement Needed' flag. • Bit[4]: Enable Programmable Hardware Replacement Flags for Failure Events: When set, the device shall generate memory media events caused by the advanced programmable failure threshold using the 'Hardware Replacement Needed' flag. • Bits[7:5]: Reserved.
08h	3	<p>Programmable Corrected Volatile Memory Error Informational Event Threshold: The device's current programmable informational threshold for corrected volatile memory errors. A single memory media event record with informational severity is generated once the total number of CVMEs for the programmed threshold granularity becomes equal to or greater than this threshold value. The device may generate additional event records if the threshold expiration function is used. An input of 00 0000h shall render the Programmable CVME Informational Event Threshold as disabled. If Patrol Scrub CVME Thresholds are enabled, this parameter shall only apply to non-patrol scrub CVMEs regardless of whether the Patrol Scrub CVME Informational Event Threshold is disabled.</p>
0Bh	3	<p>Programmable Corrected Volatile Memory Error Warning Event Threshold: The device's current programmable warning threshold for corrected volatile memory errors. A single memory media event record with warning severity is generated once the total number of CVMEs for the programmed threshold granularity becomes equal to or greater than this threshold value. The device may generate additional event records if the threshold expiration function is used. An input of 00 0000h shall render the Programmable CVME Warning Event Threshold as disabled. If Patrol Scrub CVME Thresholds are enabled, this parameter shall only apply to non-patrol scrub CVMEs regardless of whether the Patrol Scrub CVME Warning Event Threshold is disabled.</p>
0Eh	3	<p>Programmable Corrected Volatile Memory Error Failure Event Threshold: The device's current programmable failure threshold for corrected volatile memory errors. A single memory media event record with failure severity is generated once the total number of CVMEs for the programmed threshold granularity becomes equal to or greater than this threshold value. The device may generate additional event records if the threshold expiration function is used. An input of 00 0000h shall render the Programmable CVME Failure Event Threshold as disabled. If Patrol Scrub CVME Thresholds are enabled, this parameter shall only apply to non-patrol scrub CVMEs regardless of whether the Patrol Scrub CVME Failure Event Threshold is disabled.</p>

Table 8-213. Advanced Programmable Corrected Volatile Memory Error Threshold Feature Readable Attributes (Sheet 4 of 4)

Byte Offset	Length in Bytes	Description
11h	1	<p>Configured Patrol Scrub CVME Threshold Event Records Flags: Enabling a Patrol Scrub CVME threshold shall cause the device to automatically disable any CVME thresholds set by other methods (such as the one set by the Set Alert Configuration command) regardless of whether Non-Patrol Scrub CVME thresholds are disabled. Those other CVME thresholds are not automatically re-enabled whenever Patrol Scrub CVME thresholds are disabled. Enabling other CVME thresholds shall cause the device to automatically disable any Patrol Scrub CVME thresholds. Patrol Scrub CVME thresholds also do not automatically re-enable when other CVME thresholds are disabled.</p> <p>Settings for this parameter are ignored if Patrol Scrub CVME Threshold is disabled.</p> <ul style="list-style-type: none"> • Bit[0]: Enable Programmable Informational Patrol Scrub Threshold: When set, the device shall enable the advanced programmable informational patrol scrub threshold and generate any informational memory media event records associated with that threshold. • Bit[1]: Enable Programmable Warning Patrol Scrub Threshold: When set, the device shall enable the advanced programmable warning patrol scrub threshold and generate any warning memory media event records associated with that threshold. • Bit[2]: Enable Programmable Failure Patrol Scrub Threshold: When set, the device shall enable the advanced programmable failure patrol scrub threshold and generate any failure memory media event records associated with that threshold. • Bit[3]: Enable Programmable Hardware Replacement Flags for Warning Patrol Scrub Events: When set, the device shall generate memory media events caused by the advanced programmable warning patrol scrub threshold using the 'Hardware Replacement Needed' flag. • Bit[4]: Enable Programmable Hardware Replacement Flags for Failure Patrol Scrub Events: When set, the device shall generate memory media events caused by the advanced programmable failure patrol scrub threshold using the 'Hardware Replacement Needed' flag. • Bits[7:5]: Reserved.
12h	3	Programmable Patrol Scrub CVME Informational Event Threshold: The device's current programmable informational threshold for corrected volatile memory errors found by the patrol scrubber. A single memory media event record with informational severity is generated whenever the total number of patrol scrub CVMEs for the programmed threshold granularity becomes equal to or greater than this threshold value. The device may generate additional event records after the patrol scrub cycle has completed and restarted its counters. An input of 00 0000h shall render the Programmable Patrol Scrub CVME Informational Event Threshold as disabled.
15h	3	Programmable Patrol Scrub CVME Warning Event Threshold: The device's current programmable warning threshold for corrected volatile memory errors found by the patrol scrubber. A single memory media event record with warning severity is generated whenever the total number of patrol scrub CVMEs for the programmed threshold granularity becomes equal to or greater than this threshold value. The device may generate additional event records after the patrol scrub cycle has completed and restarted its counters. An input of 00 0000h shall render the Programmable Patrol Scrub CVME Warning Event Threshold as disabled.
18h	3	Programmable Patrol Scrub CVME Failure Event Threshold: The device's current programmable failure threshold for corrected volatile memory errors found by the patrol scrubber. A single memory media event record with failure severity is generated whenever the total number of patrol scrub CVMEs for the programmed threshold granularity becomes equal to or greater than this threshold value. The device may generate additional event records after the patrol scrub cycle has completed and restarted its counters. An input of 00 0000h shall render the Programmable Patrol Scrub CVME Failure Event Threshold as disabled.
1Bh	5	Reserved

Table 8-214 shows the Feature Data for the Set Feature command.

Table 8-214. Advanced Programmable Corrected Volatile Memory Error Threshold Feature Writable Attributes (Sheet 1 of 3)

Byte Offset	Length in Bytes	Description
00h	1	<p>Configured Corrected Volatile Memory Error Threshold Granularity</p> <ul style="list-style-type: none"> • 00h = Full HDM Range Granularity. When used, the programmable thresholds shall cover the entire range of HDM on the device. The device shall maintain a global CVME counter for the entire device HDM. When the global CVME counter exceeds a programmed threshold, the device shall generate a memory media event record. • 01h = Per Memory Media FRU Granularity. When used, the programmable thresholds are at a per-memory media FRU granularity (e.g., threshold per DIMM) across all memory media FRUs. The device shall maintain a separate CVME counter for each memory media FRU connected to the device. When the CVME counter for a given memory media FRU exceeds a programmed threshold, the device shall generate a memory media event record. • 02h = Per Rank Granularity. When used, the programmable thresholds are at a per-rank of device granularity (e.g., threshold per rank) across all ranks. The device shall maintain a separate CVME counter for each rank of media on the device. When the CVME counter for a given rank exceeds a programmed threshold, the device shall generate a memory media event record. • All other encodings are reserved.
01h	1	<p>Configured Corrected Volatile Memory (CVME) Error Threshold Configuration Flags: Flags in this field apply to all severity thresholds (e.g., Informational, Warning, etc.). Feature does not support individual filtering per severity threshold. Enabling all filters may still result in corrected error counting and thresholding for other error types not filterable by this feature.</p> <ul style="list-style-type: none"> • Bit[0]: Enable Single Bit Error Mask: When set, the device shall not include single bit errors in the corrected volatile memory errors count for the advanced programmable volatile memory error thresholds and subsequent memory media events associated with those thresholds. • Bit[1]: Enable Corrected Multi Bit Error Mask: When set, the device shall not include corrected multi-bit errors in the corrected volatile memory errors count for the advanced programmable volatile memory error thresholds and subsequent memory media events associated with those thresholds. • Bit[2]: Enable Patrol Scrub CVME Threshold: When set, the device shall maintain separate counters and thresholds for CVMEs detected during patrol scrub and non-patrol scrub accesses. CVME counters used by Patrol Scrub CVME Thresholds are automatically reset at the end of every Patrol Scrub cycle. When disabled, the device shall combine CVME counts from patrol scrub and non-patrol scrub accesses. Additionally, when disabled, the device shall ignore all other parameters related to Patrol Scrub CVME Thresholds. • Bit[3]: Enable Counter Expiration: When set, the device shall expire and restart all CVME counters used by the Advanced Programmable CVME Threshold Feature after a configured time duration. Counter expiration does not apply to Patrol Scrub CVME counters if Patrol Scrub Thresholds are enabled. If Patrol Scrub Thresholds are disabled, CVMEs counted by the patrol scrubber will reset alongside non-patrol scrub CVMEs when the counter expires. • Bit[4]: Enable Counter Expiration Reporting: When set, the device shall generate a media event record each time the CVME counter has expired. Media event records are created with Informational severity and a flag to indicate Threshold Expiration. Counter Expiration must be enabled and its timer input correctly configured for the device to report counter expiration. If the CVME count is 0 at the time of counter expiration, the device shall not generate a media event record. • Bits[7:5]: Reserved.
02h	3	<p>Configured Corrected Volatile Memory Error Counter Expiration Timer: Time in seconds after which the device shall expire and restart all corrected volatile memory error programmable counters. An input of 00 0000h shall render the expiration timer as disabled. Counter expiration does not apply to Patrol Scrub CVME counters if Patrol Scrub Thresholds are enabled.</p>

Table 8-214. Advanced Programmable Corrected Volatile Memory Error Threshold Feature Writable Attributes (Sheet 2 of 3)

Byte Offset	Length in Bytes	Description
05h	1	<p>Configured Corrected Volatile Memory Error Threshold Event Records Flags: Enabling an advanced CVME threshold shall cause the device to automatically disable any CVME thresholds set by other methods (such as the one set by the Set Alert Configuration command). Those CVME thresholds are not automatically re-enabled whenever advanced CVME thresholds are disabled. Enabling other CVME thresholds shall cause the device to automatically disable any advanced CVME thresholds. Advanced CVME thresholds also do not automatically re-enable when other CVME thresholds are disabled.</p> <p>If Patrol Scrub CVME Thresholds are enabled, the settings in this parameter shall only apply to non-patrol scrub CVMEs counters and thresholds.</p> <ul style="list-style-type: none"> • Bit[0]: Enable Programmable Informational Threshold: When set, the device shall enable the advanced programmable informational threshold and generate any informational memory media event records associated with that threshold. • Bit[1]: Enable Programmable Warning Threshold: When set, the device shall enable the advanced programmable warning threshold and generate any warning memory media event records associated with that threshold. • Bit[2]: Enable Programmable Failure Threshold: When set, the device shall enable the advanced programmable failure threshold and generate any failure memory media event records associated with that threshold. • Bit[3]: Enable Programmable Hardware Replacement Flags for Warning Events: When set, the device shall generate memory media events caused by the advanced programmable warning threshold using the 'Hardware Replacement Needed' flag. • Bit[4]: Enable Programmable Hardware Replacement Flags for Failure Events: When set, the device shall generate memory media events caused by the advanced programmable failure threshold using the 'Hardware Replacement Needed' flag. • Bits[7:5]: Reserved.
06h	3	Programmable Corrected Volatile Memory Error Informational Event Threshold: The device's programmable informational threshold for corrected volatile memory errors. A single memory media event record with informational severity is generated whenever the total number of CVMEs for the programmed threshold granularity becomes equal to or greater than this threshold value. The device may generate additional event records if the threshold expiration function is used. An input of 00 0000h shall render the Programmable CVME Informational Event Threshold as disabled. If Patrol Scrub CVME Thresholds are enabled, this parameter shall only apply to non-patrol scrub CVMEs regardless of whether the Patrol Scrub CVME Informational Event Threshold is disabled.
09h	3	Programmable Corrected Volatile Memory Error Warning Event Threshold: The device's programmable warning threshold for corrected volatile memory errors. A single memory media event record with warning severity is generated whenever the total number of CVMEs for the programmed threshold granularity becomes equal to or greater than this threshold value. The device may generate additional event records if the threshold expiration function is used. An input of 00 0000h shall render the Programmable CVME Warning Event Threshold as disabled. If Patrol Scrub CVME Thresholds are enabled, this parameter shall only apply to non-patrol scrub CVMEs regardless of whether the Patrol Scrub CVME Warning Event Threshold is disabled.
0Ch	3	Programmable Corrected Volatile Memory Error Failure Event Threshold: The device's programmable failure threshold for corrected volatile memory errors. A single memory media event record with failure severity is generated whenever the total number of CVMEs for the programmed threshold granularity becomes equal to or greater than this threshold value. The device may generate additional event records if the threshold expiration function is used. An input of 00 0000h shall render the Programmable CVME Failure Event Threshold as disabled. If Patrol Scrub CVME Thresholds are enabled, this parameter shall only apply to non-patrol scrub CVMEs regardless of whether the Patrol Scrub CVME Failure Event Threshold is disabled.

Table 8-214. Advanced Programmable Corrected Volatile Memory Error Threshold Feature Writable Attributes (Sheet 3 of 3)

Byte Offset	Length in Bytes	Description
0Fh	1	<p>Configured Patrol Scrub CVME Threshold Event Records Flags: Enabling a Patrol Scrub CVME threshold shall cause the device to automatically disable any CVME thresholds set by other methods (such as the one set by the Set Alert Configuration command) regardless of whether Non-Patrol Scrub CVME thresholds are disabled. Those other CVME thresholds are not automatically re-enabled whenever Patrol Scrub CVME thresholds are disabled. Enabling other CVME thresholds shall cause the device to automatically disable any Patrol Scrub CVME thresholds. Patrol Scrub CVME thresholds also do not automatically re-enable when other CVME thresholds are disabled.</p> <p>Settings for this parameter are ignored if Patrol Scrub CVME Threshold is disabled.</p> <ul style="list-style-type: none"> • Bit[0]: Enable Programmable Informational Patrol Scrub Threshold: When set, the device shall enable the advanced programmable informational patrol scrub threshold and generate any informational memory media event records associated with that threshold. • Bit[1]: Enable Programmable Warning Patrol Scrub Threshold: When set, the device shall enable the advanced programmable warning patrol scrub threshold and generate any warning memory media event records associated with that threshold. • Bit[2]: Enable Programmable Failure Patrol Scrub Threshold: When set, the device shall enable the advanced programmable failure patrol scrub threshold and generate any failure memory media event records associated with that threshold. • Bit[3]: Enable Programmable Hardware Replacement Flags for Warning Patrol Scrub Events: When set, the device shall generate memory media events caused by the advanced programmable warning patrol scrub threshold using the 'Hardware Replacement Needed' flag. • Bit[4]: Enable Programmable Hardware Replacement Flags for Failure Patrol Scrub Events: When set, the device shall generate memory media events caused by the advanced programmable failure patrol scrub threshold using the 'Hardware Replacement Needed' flag. • Bits[7:5]: Reserved.
10h	3	Programmable Patrol Scrub CVME Informational Event Threshold: The device's programmable informational threshold for corrected volatile memory errors found by the patrol scrubber. A single memory media event record with informational severity is generated whenever the total number of patrol scrub CVMEs for the programmed threshold granularity becomes equal to or greater than this threshold value. The device may generate additional event records after the patrol scrub cycle has completed and restarted its counters. An input of 00 0000h shall render the Programmable Patrol Scrub CVME Informational Event Threshold as disabled.
13h	3	Programmable Patrol Scrub CVME Warning Event Threshold: The device's programmable warning threshold for corrected volatile memory errors found by the patrol scrubber. A single memory media event record with warning severity is generated whenever the total number of patrol scrub CVMEs for the programmed threshold granularity becomes equal to or greater than this threshold value. The device may generate additional event records after the patrol scrub cycle has completed and restarted its counters. An input of 00 0000h shall render the Programmable Patrol Scrub CVME Warning Event Threshold as disabled.
16h	3	Programmable Patrol Scrub CVME Failure Event Threshold: The device's programmable failure threshold for corrected volatile memory errors found by the patrol scrubber. A single memory media event record with failure severity is generated whenever the total number of patrol scrub CVMEs for the programmed threshold granularity becomes equal to or greater than this threshold value. The device may generate additional event records after the patrol scrub cycle has completed and restarted its counters. An input of 00 0000h shall render the Programmable Patrol Scrub CVME Failure Event Threshold as disabled.
19h	7	Reserved

8.2.9.10 FM API Commands

CXL FM API commands are identified by a 2-byte Opcode as specified in [Table 8-215](#). Opcodes also provide an implicit *major* version number, which means a command's definition shall not change in an incompatible way in future revisions of this specification. Instead, if an incompatible change is required, the specification defining the change shall define a new opcode for the changed command. Commands may evolve by defining new fields in the payload definitions that were originally defined as Reserved, but only in a way where software written using the earlier definition will continue to work correctly, and software written to the new definition can use the 0 value or the payload size to detect devices that do not support the new field. This implicit *minor* versioning allows software to be written with the understanding that an opcode shall only evolve by adding backward-compatible changes.

Table 8-215. CXL FM API Command Opcodes (Sheet 1 of 5)

Opcode				Required ¹				
				Type 1/2/3 Device		Switch		
Command Set Bits[15:8]		Command Bits[7:0]		Combined Opcode	Mailbox ²	Type 3 MCTP	Host Mailbox	FM Interface ³
51h	Physical Switch	00h	Identify Switch Device (Section 7.6.7.1.1)	5100h	P	P	MSW	MSW
		01h	Get Physical Port State (Section 7.6.7.1.2)	5101h	P	P	MSW	MSW
		02h	Physical Port Control (Section 7.6.7.1.3)	5102h	P	P	P	O
		03h	Send PPB CXL.io Configuration Request (Section 7.6.7.1.4)	5103h	P	P	P	O
		04h	Get Domain Validation SV State (Section 7.6.7.1.5)	5104h	P	P	O	O
		05h	Set Domain Validation SV (Section 7.6.7.1.6)	5105h	P	P	O	P
		06h	Get VCS Domain Validation SV State (Section 7.6.7.1.7)	5106h	P	P	P	O
		07h	Get Domain Validation SV (Section 7.6.7.1.8)	5107h	P	P	P	O
52h	Virtual Switch	00h	Get Virtual CXL Switch Info (Section 7.6.7.2.1)	5200h	P	P	P	O
		01h	Bind vPPB (Section 7.6.7.2.2)	5201h	P	P	P	O
		02h	Unbind vPPB (Section 7.6.7.2.3)	5202h	P	P	P	O
		03h	Generate AER Event (Section 7.6.7.2.4)	5203h	P	P	P	O
53h	MLD Port	00h	Tunnel Management Command (Section 7.6.7.3.1)	5300h	P	MO	P	O
		01h	Send LD CXL.io Configuration Request (Section 7.6.7.3.2)	5301h	P	P	P	O
		02h	Send LD CXL.io Memory Request (Section 7.6.7.3.3)	5302h	P	P	P	O

Table 8-215. CXL FM API Command Opcodes (Sheet 2 of 5)

Opcode				Required ¹				
				Type 1/2/3 Device		Switch		
Command Set Bits[15:8]		Command Bits[7:0]		Combined Opcode	Mailbox ²	Type 3 MCTP	Host Mailbox	FM Interface ³
54h	MLD Components	00h	Get LD Info (Section 7.6.7.4.1)	5400h	MM	MMS	P	P
		01h	Get LD Allocations (Section 7.6.7.4.2)	5401h	MM	MO	P	P
		02h	Set LD Allocations (Section 7.6.7.4.3)	5402h	MO	MO	P	P
		03h	Get QoS Control (Section 7.6.7.4.4)	5403h	MM	MO	P	P
		04h	Set QoS Control (Section 7.6.7.4.5)	5404h	MM	MO	P	P
		05h	Get QoS Status (Section 7.6.7.4.6)	5405h	MO	MO	P	P
		06h	Get QoS Allocated BW (Section 7.6.7.4.7)	5406h	MM	MO	P	P
		07h	Set QoS Allocated BW (Section 7.6.7.4.8)	5407h	MM	MO	P	P
		08h	Get QoS BW Limit (Section 7.6.7.4.9)	5408h	MM	MO	P	P
		09h	Set QoS BW Limit (Section 7.6.7.4.10)	5409h	MM	MO	P	P
55h	Multi-Headed Devices	00h	Get Multi-Headed Info (Section 7.6.7.5.1)	5500h	MHO	MHS	P	P
		01h	Get Head Info (Section 7.6.7.5.2)	5501h	MHO	MHO	P	P
56h	DCD Management	00h	Get DCD Info (Section 7.6.7.6.1)	5600h	DCO	DCS	P	P
		01h	Get Host DC Region Configuration (Section 7.6.7.6.2)	5601h	DCO	DCS	P	P
		02h	Set DC Region Configuration (Section 7.6.7.6.3)	5602h	DCO	DCS	P	P
		03h	Get DC Region Extent Lists (Section 7.6.7.6.4)	5603h	DCO	DCS	P	P
		04h	Initiate Dynamic Capacity Add (Section 7.6.7.6.5)	5604h	DCO	DCS	P	P
		05h	Initiate Dynamic Capacity Release (Section 7.6.7.6.6)	5605h	DCO	DCS	P	P
		06h	Dynamic Capacity Add Reference (Section 7.6.7.6.7)	5606h	DCO	DCS	P	P
		07h	Dynamic Capacity Remove Reference (Section 7.6.7.6.8)	5607h	DCO	DCS	P	P
		08h	Dynamic Capacity List Tags (Section 7.6.7.6.9)	5608h	DCO	DCS	P	P

Table 8-215. CXL FM API Command Opcodes (Sheet 3 of 5)

Opcode				Required ¹			
				Type 1/2/3 Device		Switch	
Command Set Bits[15:8]		Command Bits[7:0]	Combined Opcode	Mailbox ²	Type 3 MCTP	Host Mailbox	FM Interface ³
57h	PBR Switch	00h	Identify PBR Switch (Section 7.7.13.1)	5700h	P	P	P MPSW
		01h	Fabric Crawl Out (Section 7.7.13.2)	5701h	P	P	P MPSW
		02h	Get PBR Link Partner Info (Section 7.7.13.3)	5702h	P	P	P MPSW
		03h	Get PID Target List (Section 7.7.13.4)	5703h	P	P	P MPSW
		04h	Configure PID Assignment (Section 7.7.13.5)	5704h	P	P	P MPSW
		05h	Get PID Binding (Section 7.7.13.6)	5705h	P	P	P MPSW
		06h	Configure PID Binding (Section 7.7.13.7)	5706h	P	P	P MPSW
		07h	Get Table Descriptors (Section 7.7.13.8)	5707h	P	P	P MPSW
		08h	Get DRT (Section 7.7.13.9)	5708h	P	P	P MPSW
		09h	Set DRT (Section 7.7.13.10)	5709h	P	P	P MPSW
		0Ah	Get RGT (Section 7.7.13.11)	570Ah	P	P	O
		0Bh	Set RGT (Section 7.7.13.12)	570Bh	P	P	O
		0Ch	Get LDST/IDT Capabilities (Section 7.7.13.13)	570Ch	P	P	O ⁴ P
		0Dh	Set LDST/IDT Configuration (Section 7.7.13.14)	570Dh	P	P	O ⁴ P
		0Eh	Get LDST Segment Entries (Section 7.7.13.15)	570Eh	P	P	O ⁴ P
		0Fh	Set LDST Segment Entries (Section 7.7.13.16)	570Fh	P	P	O ⁴ P
		10h	Get LDST IDT DPID Entries (Section 7.7.13.17)	5710h	P	P	O ⁴ P
		11h	Set LDST IDT DPID Entries (Section 7.7.13.18)	5711h	P	P	O ⁴ P
		12h	Get Completer ID-Based Re-Router Entries (Section 7.7.13.19)	5712h	P	P	O ⁴ P
		13h	Set Completer ID-Based Re-Router Entries (Section 7.7.13.20)	5713h	P	P	O ⁴ P
		14h	Get LDST Access Vector (Section 7.7.13.21)	5714h	P	P	O ⁴ P

Table 8-215. CXL FM API Command Opcodes (Sheet 4 of 5)

Opcode				Required ¹				
				Type 1/2/3 Device		Switch		
Command Set Bits[15:8]		Command Bits[7:0]		Combined Opcode	Mailbox ²	Type 3 MCTP	Host Mailbox	FM Interface ³
57h	PBR Switch	15h	Get VCS LDST Access Vector (Section 7.7.13.22)	5715h	P	P	P	O
		16h	Configure VCS LDST Access (Section 7.7.13.23)	5716h	P	P	P	O
58h	Global Memory Access Endpoint	00h	Identify GAE (Section 7.7.14.1)	5800h	P	P	MG	P ⁴
		01h	Get PID Interrupt Vector (Section 7.7.14.2)	5801h	P	P	MG	P ⁴
		02h	Get PID Access Vectors (Section 7.7.14.3)	5802h	P	P	MG	P ⁴
		03h	Get FAST/IDT Capabilities (Section 7.7.14.4)	5803h	P	P	MG	P ⁴
		04h	Set FAST/IDT Configuration (Section 7.7.14.5)	5804h	P	P	MG	P ⁴
		05h	Get FAST Segment Entries (Section 7.7.14.6)	5805h	P	P	MG	P ⁴
		06h	Set FAST Segment Entries (Section 7.7.14.7)	5806h	P	P	MG	P ⁴
		07h	Get IDT DPID Entries (Section 7.7.14.8)	5807h	P	P	MG	P ⁴
		08h	Set IDT DPID Entries (Section 7.7.13.18)	5808h	P	P	MG	P ⁴
		09h	Proxy GFD Management Command (Section 7.7.14.10)	5809h	P	P	MG	P ⁴
		0Ah	Get Proxy Thread Status (Section 7.7.14.11)	580Ah	P	P	MG	P ⁴
		0Bh	Cancel Proxy Thread (Section 7.7.14.12)	580Bh	P	P	MG	P ⁴

Table 8-215. CXL FM API Command Opcodes (Sheet 5 of 5)

Opcode				Required ¹				
				Type 1/2/3 Device		Switch		
Command Set Bits[15:8]		Command Bits[7:0]		Combined Opcode	Mailbox ²	Type 3 MCTP	Host Mailbox	FM Interface ³
59h	Global Memory Access Endpoint Management	00h	Identify VCS GAE (Section 7.7.15.1)	5900h	P	P	P	MG
		01h	Get VCS PID Access Vectors (Section 7.7.15.2)	5901h	P	P	P	MG
		02h	Configure VCS PID Access (Section 7.7.15.3)	5902h	P	P	P	MG
		03h	Get VendPrefixL0 State (Section 7.7.15.4)	5903h	P	P	P	O
		04h	Set VendPrefixL0 State (Section 7.7.15.5)	5904h	P	P	P	O

1. O = Optional, P = Prohibited, MM = Mandatory for all MLD components (prohibited for SLD components), MO = Optional for MLD components (prohibited for SLD components), MMS = Mandatory for all MLD components that implement an MCTP-based CCI (prohibited for SLD components), MHO = Optional for Multi-Headed devices, MHS = Mandatory for all Multi-Headed devices that implement an MCTP-based CCI, DCO = Optional for DCDs, DCS = Mandatory for all DCDs that implement an MCTP-based CCI, MSW = mandatory on MCTP-based CCIs for all switches that support the FM API MCTP message type, MPSW = Mandatory for PBR Switches, MG = Mandatory for GAE.
2. The FM-owned LD in MLD components is accessible using the transport defined in [Section 7.6.3.1](#).
3. "FM Interface" is defined as any CCI that is configured for FM use, including a Mailbox CCI.
4. A Mailbox CCI configured for FM use is permitted to also operate as a GAE. In such a case, these commands may optionally be supported.

§ §

9.0

Reset, Initialization, Configuration, and Manageability

9.1

CXL Boot and Reset Overview

9.1.1

General

Boot and Power-up sequencing of CXL devices follows the applicable form-factor specifications and as such, will not be discussed in detail in this section.

CXL devices can encounter three types of resets.

- Hot Reset – Triggered via link (via LTSSM or Link Down)
- Warm Reset – Triggered via external signal, PERST# (or equivalent, form-factor-specific mechanism)
- Cold Reset – Involves main Power removal and PERST# (or equivalent, form-factor-specific mechanism)

These three reset types are labeled as Conventional Reset. Function Level Reset (see [Section 9.5](#)) and CXL Reset (see [Section 9.7](#)) are not considered to be Conventional Resets. These definitions are consistent with PCIe* Base Specification.

Flex Bus Physical Layer link states across cold reset, warm reset, surprise reset, and Sx entry match PCIe Physical Layer link states.

This chapter highlights the differences that exist between CXL and native PCIe for these reset operations.

A PCIe device generally cannot determine which system-level flow triggered a Conventional Reset. System-level reset and Sx-entry flows require coordinated coherency domain shutdown before the sequence can progress. Therefore, the CXL flow will adhere to the following rules:

- Warnings shall be issued to all CXL devices before the system initiates system-level reset and Sx-entry transitions.
- CXL PM messages shall be used to communicate between the host and the device. Devices must respond to these messages with the correct acknowledge, even if no actions are actually performed on the device. To prevent deadlock in cases where one or more downstream components do not respond, the host must implement a timeout, after which the host proceeds as if the response has been received.
- A device shall correctly process the reset trigger regardless of whether they are preceded by these warning messages. Not all device resets are preceded by a warning message. For example, setting Secondary Bus Reset bit in a Downstream Port above the device results in a device hot-reset, but it is not preceded by any warning message. It is also possible that the PM VDM warning message may be lost due to an error condition.

Sx states are system Sleep States and are enumerated in ACPI Specification.

9.1.2

Comparing CXL and PCIe Behavior

Table 9-1 summarizes the difference in event sequencing and signaling methods across System Reset and Sx flows, for CXL.io, CXL.cache, CXL.mem, and PCIe.

The terms used in the table are as follows:

- **Warning:** An early notification of the upcoming event. Devices with coherent cache or memory are required to complete outstanding transactions, flush internal caches as needed, and then place memory in a safe state such as Self-refresh as required. Devices are required to complete all internal actions and then respond with a correct Ack to the processor
- **Signaling:** Actual initiation of the state transition, using either wires and/or link-layer messaging

Table 9-1. Event Sequencing for Reset and Sx Flows

Case	PCIe	CXL
System Reset Entry	Warning: None. Signaling: LTSSM Hot Reset.	Warning: PM2IP (ResetWarn, System Reset) ¹ . Signaling: LTSSM Hot Reset.
Surprise System Reset Entry	Warning: None. Signaling: LTSSM detect-entry or PERST#.	Warning: None. Signaling: LTSSM detect-entry or PERST#.
System Sx Entry	Warning: PME_Turn_Off/Ack. Signaling: PERST# (Main power will go down).	Warning: PM2IP (ResetWarn, Sx) ¹ . PME_Turn_Off/Ack. Signaling: PERST# (Main power will go down).
System Power Failure	Warning: None.	Warning: PM2IP (GPF Phase 1 and Phase 2) ¹ ; see Section 9.8 .

1. CXL PM VDM with different encodings for different events. If CXL.io devices do not respond to the CXL PM VDM, the host may still end up in the correct state due to timeouts.

9.1.2.1

Switch Behavior

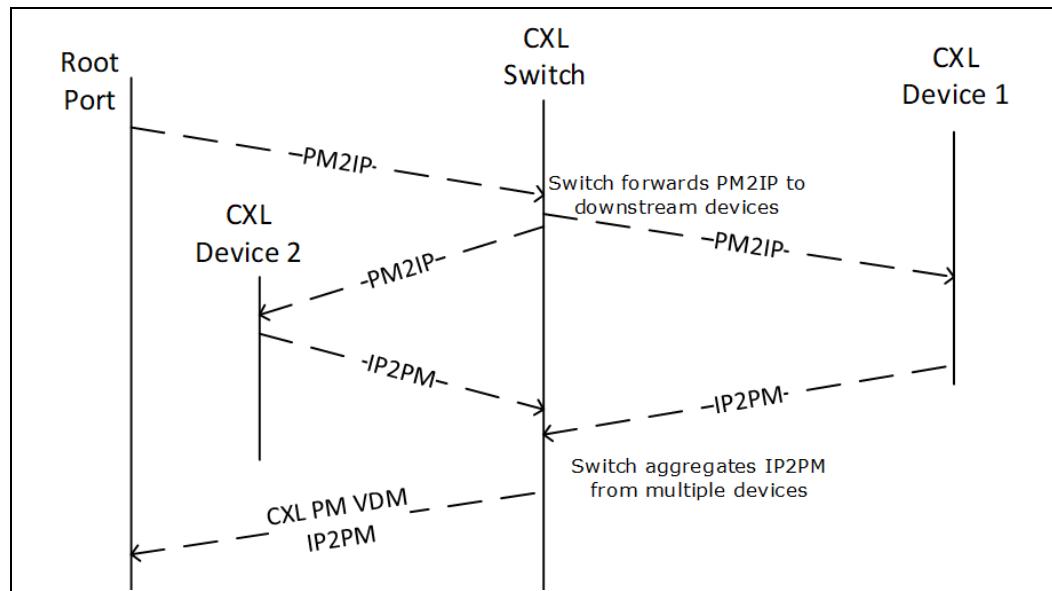
When a CXL Switch (physical or virtual) is present, the Switch shall forward PM2IP messages received on its primary interface to CXL components on the secondary interface subject to rules specified below. The Switch shall aggregate IP2PM messages from the secondary interface prior to responding on its primary interface subject to rules specified below. (See [Table 3-1](#) for PM Commands.) When communicating with a pooled device, these messages shall carry LD-ID TLP Prefix in both directions.

Table 9-2. CXL Switch Behavior Message Aggregation Rules (Sheet 1 of 2)

PM Logical Opcode Value	PM Command	Action
0	AGENT_INFO	<ul style="list-style-type: none"> • Do not forward PM2IP messages to downstream Devices. • Execute Credits and PM Initialization flow against the downstream entity whenever a link trains up in CXL mode. • Save CAPABILITY_VECTOR from the response.
2	RESETPREP	<ul style="list-style-type: none"> • Never forward PM2IP messages to PCIe links. • Forward PM2IP messages to all active downstream CXL links. • Gather the IP2PM messages from all active downstream CXL links.

Table 9-2. CXL Switch Behavior Message Aggregation Rules (Sheet 2 of 2)

PM Logical Opcode Value	PM Command	Action
4	PMREQ	<ul style="list-style-type: none"> Never forward PM2IP messages to PCIe links. Forward PM2IP messages to all active downstream CXL links. Gather the IP2PM messages from all active downstream CXL links. “Conglomerate” Latency Tolerance Reporting (LTR) requests from all Devices by following the rules defined in LTR Mechanism section in PCIe Base Specification.
6	GPF	<ul style="list-style-type: none"> Never forward PM2IP messages to PCIe links. Never forward PM2IP messages to all downstream CXL links that returned CAPABILITY_VECTOR[1]=0. Forward PM2IP messages to all downstream CXL links that returned CAPABILITY_VECTOR[1]=1 and gather the IP2PM responses from all such links.
FEh	CREDIT_RTN	<ul style="list-style-type: none"> Do not forward PM2IP message to downstream Devices. PM Credit management on the primary interface is independent of PM credit management on the secondary interface.

Figure 9-1. PMREQ/RESETPREP Propagation by CXL Switch

9.2 CXL Device Boot Flow

CXL devices shall follow the appropriate form factor specification regarding the boot flows.

This specification uses the terms “Warm Reset” and “Cold Reset” in a manner that is consistent with PCIe Base Specification.

9.3 CXL System Reset Entry Flow

In an OS-orchestrated reset flow, it is expected that the CXL devices are already in an Inactive State with their contexts flushed to the system memory or CXL-attached memory before the platform reset flow is triggered.

In a platform-triggered reset flow (e.g., due to a fatal error), a CXL device may not be in an Inactive State when the device receives the ResetPrep message.

During system reset flow, the host shall issue a CXL PM VDM (see [Table 3-1](#)) to the downstream CXL components with the following values:

- PM Logical Opcode[7:0]=RESETPREP
- Parameter[15:0]=REQUEST
- ResetType = System Reset
- PrepType = General Prep

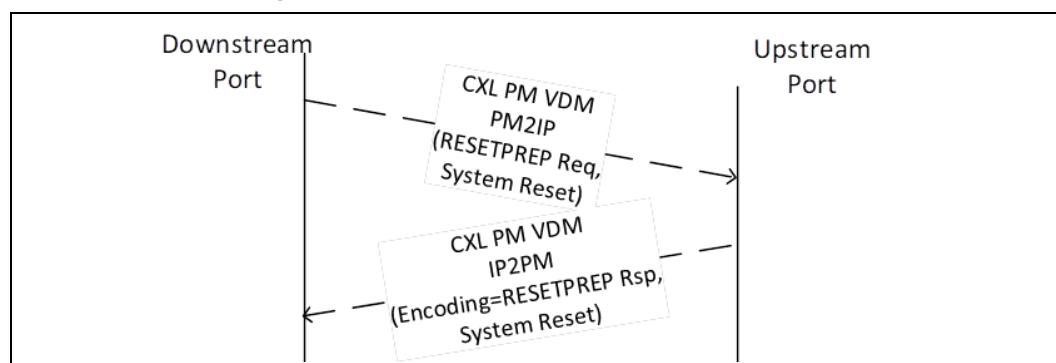
The CXL device shall flush any relevant context to the host, clean up the data serving the host, and then place any CXL device connected memory into a safe state such as self-refresh. The CXL device shall take any additional steps that are necessary for the CXL host to enter LTSSM Hot Reset. After all the Reset preparation is complete, the CXL device shall issue a CXL PM VDM with the following value:

- PM Logical Opcode[7:0]=RESETPREP
- Parameter[15:0]=RESPONSE
- ResetType = System Reset
- PrepType = General Prep

The CXL device may have PERST# asserted after the reset handshake is complete. On PERST# assertion, the CXL device should clear any sticky content internal to the device unless they are on auxiliary power. The CXL device's handling of sticky register state is consistent with PCIe Base Specification.

To prevent a deadlock in the case where one or more downstream components do not respond with an Ack, the host must implement a timeout, after which the host proceeds as if the response has been received.

Figure 9-2. CXL Device Reset Entry Flow



CXL Device Sleep State Entry Flow

Since OS is always the orchestrator of Sx entry flows, it is expected that the CXL devices are already in an Inactive State with their contexts flushed to the CPU-attached memory or CXL-attached memory before the Sx entry flow is triggered.

During Sx entry flow, the host shall issue a CXL PM VDM (see [Table 3-1](#)) to the downstream components with the following values:

- PM Logical Opcode[7:0]=RESETPREP
- Parameter[15:0]=REQUEST

- ResetType = System transition from S0 to Sx (S1, S3, S4, or S5)
- PrepType = General Prep

The CXL device shall flush any relevant context to the host, clean up the data serving the host, and then place any CXL device connected memory into a safe state such as self-refresh. The CXL device shall take any additional steps that are necessary for the CXL host to initiate an L2 entry flow. After all the Sx preparation is complete, the CXL device shall issue a CXL PM VDM with the following values:

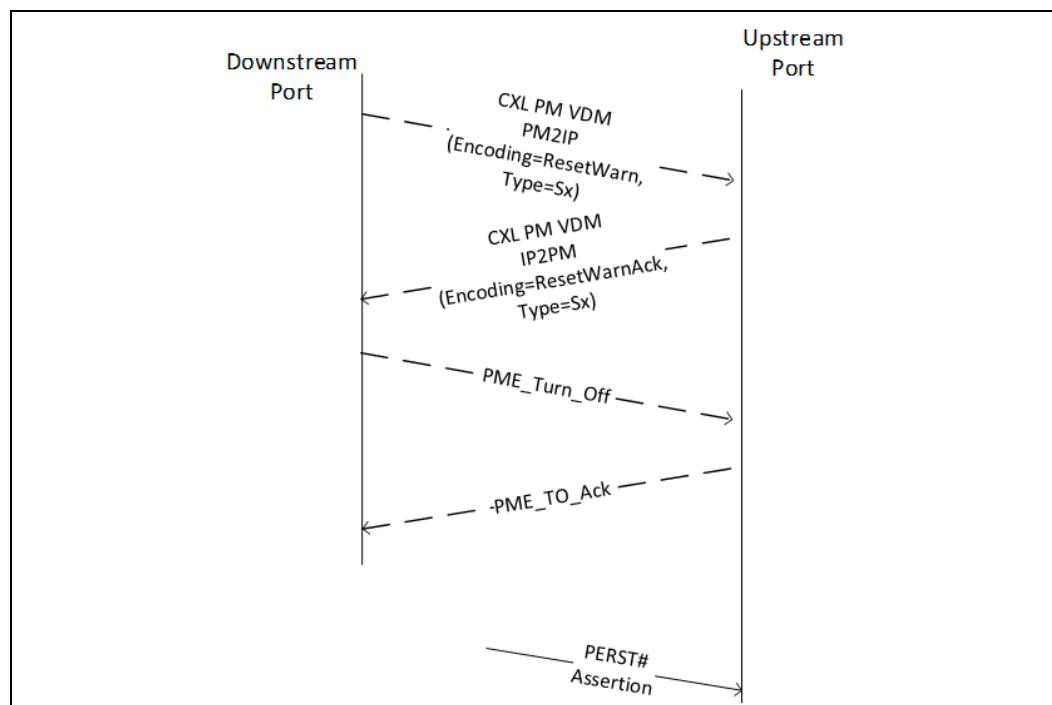
- PM Logical Opcode[7:0]=RESETPREP
- Parameter[15:0]=RESPONSE
- ResetType = System transition from S0 to Sx (based on the target sleep state)
- PrepType = General Prep

PERST# to the CXL device may be asserted any time after this handshake is complete. On PERST# assertion, the CXL device should clear any sticky content internal to the device unless they are on auxiliary power. The CXL device's handling of sticky register state is consistent with PCIe Base Specification.

CXL.mem-capable adapters may need auxiliary power to retain memory context across S3.

Note:

Figure 9-3. CXL Device Sleep State Entry Flow



Function Level Reset (FLR)

The PCIe FLR mechanism enables software to quiesce and reset Endpoint hardware with Function-level granularity. CXL devices expose one or more PCIe functions to host software. These functions can expose FLR capability and existing PCIe-compatible software can issue an FLR to these functions. PCIe Base Specification provides specific

guidelines regarding the impact of an FLR on PCIe function level state and control registers. For compatibility with existing PCIe software, CXL PCIe functions shall follow those guidelines if the Functions support FLR. For example, any software-readable state that potentially includes secret information associated with any preceding use of the Function must be cleared by an FLR.

FLRs do not affect the CXL.cache and CXL.mem protocols. Any CXL.cache-related and CXL.mem-related control registers, including CXL DVSEC structures and state held by the CXL device, are not affected by FLRs. The memory controller that hosts the HDM is not reset by an FLR. After an FLR, all address translations associated with the corresponding Function are invalidated in accordance with PCIe Base Specification. Since the CXL Function accesses cache using the system physical address held in the address translation cache, the Function is unable to access any cachelines after the FLR until software explicitly re-enables ATS. The device is not required to write back its cache during an FLR flow. To avoid an adverse effect on the performance of other Functions, it is strongly recommended that the device not write back its cache content during an FLR if the cache is shared by multiple functions. Cache coherency must be maintained.

In some cases, system software may use an FLR to attempt error recovery. In the context of CXL devices, errors in CXL.cache logic and in CXL.mem logic cannot be recovered by an FLR. An FLR may succeed in recovering from CXL.io domain errors.

In a CXL device other than an eRCD, all Functions that participate in CXL.cache or CXL.mem are required to support either FLR or CXL Reset (see [Section 9.7](#)).

9.6

Cache Management

A CXL-unaware OS or PCIe bus driver is unaware of CXL.cache capability. The device driver is expected to be aware of this CXL.cache capability and may manage the CXL.cache. Software shall not assume that lines in device cache that map to HDM will be flushed by CPU cache flush instructions. The behavior may vary from one host to another.

System software may wish to ensure that a CXL.cache-capable device does not contain any valid cachelines without resetting the system or the entire device. Since a device is not required to clear cache contents upon FLR, separate control and status bits are defined for this purpose. This capability is highly recommended for CXL.cache-capable eRCDs and mandatory for all other CXL.cache-capable devices. The capability is advertised via the Cache Writeback and Invalidate Capable flag in the DVSEC CXL Capability register (see [Section 8.1.3.1](#)).

Software shall take the following steps to ensure that the Device does not contain any valid cachelines:

1. Set Disable Caching=1. This bit is located in the DVSEC CXL Control2 register (see [Section 8.1.3.4](#)).
2. Set Initiate Cache Write Back and Invalidation=1. This step may be combined with the previous step as a single configuration space register write to the DVSEC CXL Control2 register (see [Section 8.1.3.4](#)).
3. Wait until Cache Invalid=1. This bit is located in the DVSEC CXL Status2 register (see [Section 8.1.3.5](#)). Software may leverage the cache size reported in the DVSEC CXL Capability2 register (see [Section 8.1.3.7](#)) to compute a suitable timeout value.

Software is required to Set Disable Caching=0 to re-enable caching. When the Disable Caching bit transitions from 1 to 0, the device shall transition the Cache Invalid bit to 0 if it was previously set to 1.

9.7

CXL Reset

CXL.cache resources and CXL.mem resources such as controllers, buffers, and caches are likely to be shared at the device level. CXL Reset is a mechanism that is used to reset all CXL.cache states and CXL.mem states in addition to CXL.io in all non-Virtual Functions that support CXL.cache protocols and/or CXL.mem protocols. Reset of CXL.io has the same scope as FLR. [Section 9.5](#) describes FLR in the context of CXL devices. CXL Reset will not affect non-CXL Functions or the physical link. Non-CXL Function Map DVSEC capability is used to advertise to the System Software which non-Virtual Functions are considered non-CXL (i.e., they neither participate in CXL.cache nor in CXL.mem).

All Functions in an SLD that participate in CXL.cache or CXL.mem are required to support either FLR or CXL Reset. MLDs, on the other hand, are required to support CXL Reset.

Capability, Control, and Status fields for CXL Reset are exposed in configuration space of Device 0, Function 0 of a CXL device but these affect all physical and virtual functions within the device that participate in CXL.cache or CXL.mem.

The system software is responsible for quiescing all the Functions that are impacted due to reset of the CXL.cache state and CXL.mem state in the device and offlining any associated HDM ranges. Once the CXL Reset is complete, all CXL Functions on the device must be re-initialized prior to use.

CXL Reset may be issued by the System Software or the Fabric Manager. To quiesce the impacted non-virtual Functions prior to issuing CXL Reset, the System Software shall complete the following actions for each of the CXL non-virtual Functions:

1. Offline any volatile or persistent HDM Ranges. When offlining is complete, there shall be no outstanding or new CXL.mem transactions to the affected CXL Functions.
2. Configure these Functions to stop initiating new CXL.io requests. This procedure is identical to that for FLR.

The FM may issue CXL Reset for various cases described in [Chapter 7.0](#). In the case of the FM use of CXL Reset, there may be outstanding commands in the device which shall be silently discarded.

CXL.io reset of the device shall follow the definition of FLR in PCIe Base Specification. Note that only PCIe-mapped memory shall be cleared or randomized by the non-virtual Functions during FLR.

Reset of the CXL.cache state and CXL.mem state as part of the CXL Reset flow at the device level has the following behavior:

- All outstanding or new CXL.mem reads shall be silently discarded. Previously accepted writes to persistent HDM ranges shall be persisted. Writes to volatile HDM ranges may be discarded.
- The device caches (Type 1 Devices and Type 2 Devices) shall be written back and invalidated by the device. Software is not required to write back and invalidate the device cache (see [Section 9.6](#)) prior to issuing the CXL Reset.
- No new CXL.cache requests shall be issued except for the above cache-flushing operation. Snoops shall continue to be serviced.
- Contents of volatile HDM ranges may or may not be retained and the device may optionally clear or randomize these ranges if this capability is supported and is requested during CXL Reset (see the CXL Reset Mem Clr Capable bit in the DVSEC CXL Capability register and the CXL Reset Mem Clr Enable bit in the DVSEC Control2 register in [Section 8.1.3.1](#) and [Section 8.1.3.4](#), respectively). Contents of the persistent HDM ranges will be retained by the device.

- Any errors during a CXL Reset shall be logged in the error status registers in the usual manner. Failure to complete a CXL Reset shall result in the CXL Reset Error bit in the DVSEC CXL Status2 Register being set. The system software may choose to retry CXL Reset, assert other types of device resets, or restart the system in response to a CXL Reset failure.
- Unless specified otherwise, all non-sticky registers defined in this specification shall be initialized to their default values upon CXL Reset. The CONFIG_LOCK bit in the DVSEC Config Lock register (see [Section 8.1.3.6](#)) and any register fields that are locked by CONFIG_LOCK shall not be affected by CXL Reset. Any sticky registers, such as the error status registers, shall be preserved across CXL Reset. If the device is in the viral state, it shall remain in that state after a CXL Reset.

If the device is unable to complete CXL Reset within the specified timeout period, the System Software shall consider this a failure and may choose to take action similar to when the CXL Reset Error bit is set.

A pooled Type 3 device (MLD) must ensure that only the LD assigned to the host that is issuing CXL Reset is impacted. This includes the clearing or randomizing of the volatile HDM ranges on the device. Other LDs must continue to operate normally.

9.7.1

Effect on the Contents of the Volatile HDM

Because ownership of the volatile HDM ranges may change following a CXL Reset, it is important to ensure that there is no leak of volatile memory content that was present prior to the CXL Reset. (This condition does not apply to persistent memory content whose security is ensured by other means not discussed here.)

There are two cases to consider:

- The device remains bound to the same host and the System Software reallocates the volatile HDM ranges to a different software entity. The System Software is often responsible for ensuring that the memory range is re-initialized prior to any allocation. The device may implement an optional capability to perform clearing or randomizing of all impacted volatile HDM ranges. This may be invoked using the optional Secure Erase function (see [Section 8.2.9.9.5.2](#)). Optionally, the device may be capable of clearing or randomizing volatile HDM content as part of CXL Reset. If this capability is available, the System Software may take advantage of it. However, since this is an optional capability, the System Software should not depend on it.
- The device is migrated to a different host with FM involvement as described in [Chapter 7.0](#). The FM must use either Secure Erase operation (see [Section 8.2.9.9.5.2](#)) or utilize CXL Reset if the CXL Reset Mem Clr capability exists to clear or randomize any volatile HDM ranges prior to re-assigning device to a different host.

Capability for clearing and randomizing volatile HDM ranges in the device is reported by the CXL Reset Mem Clr Capable bit in the DVSEC CXL Capability register. If present, this capability may optionally be used by setting the CXL Reset Mem Clr Enable bit in the DVSEC CXL Control2 register.

9.7.2

Software Actions

System Software or Fabric Manager shall follow these steps while performing CXL Reset:

1. Verify that the device supports CXL Reset by consulting the CXL Reset Capable bit in the DVSEC CXL Capability register (see [Section 8.1.3.1](#)).
2. Prepare the system for CXL Reset as described in [Section 9.7](#).

3. Determine whether the device supports the CXL Reset Mem Clr capability bit by consulting the DVSEC CXL Capability register (see [Section 8.1.3.1](#)).
4. If the device supports the CXL Reset Mem Clr capability, program the CXL Reset Mem Clr Enable bit in the DVSEC Control2 register (see [Section 8.1.3.4](#)) as required.
5. Determine the timeout for completion by consulting the CXL Reset Timeout field in the DVSEC CXL Capability register.
6. Set the Initiate CXL Reset=1 in the DVSEC CXL Control2 register.
7. Wait for CXL Reset Complete=1 or CXL Reset Error=1 in the DVSEC CXL Status2 register (see [Section 8.1.3.5](#)) for up to the timeout period.

System Software should follow these steps while re-initializing and onlining a device:

1. Set up the device as required to enable functions impacted by CXL Reset.
2. Optionally check whether the device performed clearing or randomizing of memory during the CXL Reset. If yes, skip software-based initialization prior to re-allocation. If not, perform software-based initialization.

9.7.3

CXL Reset and Request Retry Status (RRS)

The device must successfully complete the configuration write that triggered the CXL Reset. The device behavior in response to Configuration Space access to the device within 100 ms of initiating a CXL Reset is undefined. After 100 ms from the issuance of CXL Reset, the CXL Function is permitted to return RRS for all Configuration Space accesses except to the CXL Status2 register. After 100 ms from the issuance of CXL Reset, software should not access any device register other than the CXL Status2 register until CXL Reset completion, timeout, or error.

9.8

Global Persistent Flush (GPF)

Global Persistent Flush (GPF) is a hardware-based mechanism associated with persistent memory that is used to flush cache and memory buffers to a persistence domain. A persistence domain is defined as a location that is guaranteed to preserve the data contents across a restart of the device containing the data. GPF operation is global in nature because all CXL agents that are part of a cache coherency domain participate in the GPF flow. A CXL.cache coherency domain consists of one or more hosts, all CXL Root Ports that belong to these hosts, and the virtual hierarchies associated with these Root Ports.

GPF may be triggered in response to an impending non-graceful shutdown such as a sudden power loss. The host may initiate GPF to ensure that any in-flight data is written back to persistent media prior to a power loss. GPF may also be triggered upon other asynchronous or synchronous events that may or may not involve power loss. The complete list of such events, the mechanisms by which the host is notified, and coordination across CXL Root Ports are beyond the scope of this specification.

9.8.1

Host and Switch Responsibilities

With the exception of eRCHs, all hosts and all CXL switches shall support GPF as outlined in this section.

GPF flow consists of two phases, GPF Phase 1 and GPF Phase 2. During Phase 1, the devices are expected to stop injecting new traffic and write back their caches. During Phase 2, the persistent devices are expected to flush their local write buffers to a persistence domain. This two-phase approach ensures that a device does not receive any new traffic while it is flushing its local memory buffers. The host shall enforce a

barrier between the two phases. The host shall ensure that it stops injecting new CXL.cache transactions and that its local caches are written back prior to entering GPF Phase 2.

In certain configurations, the cache write back step may be skipped during GPF Phase 1. There are various possible reasons for implementing this mode of operation that are beyond the scope of this specification. One possible reason could be that the host does not have the required energy to write back all the caches before the power loss. When operating in this mode, the system designer may use other means, beyond the scope of this specification, to ensure that the data that is meant to be persistent is not lost. The host shall set the Payload[1] flag in the GPF Phase 1 request to indicate that the devices shall write back their caches during Phase 1. The host uses a host-specific mechanism to determine the correct setting of Payload[1].

During each phase, the host shall transmit a CXL GPF PM VDM request to each GPF-capable device or Switch that is connected directly to each of its Root Ports and then wait for a response. [Table 3-1](#) describes the format of these messages. The Switch's handling of a GPF PM VDM is described in [Section 9.1.2.1](#). The CXL Root Ports and CXL downstream Switch Ports shall implement timeouts to prevent a single device from blocking GPF forward progress. These timeouts are configured by system software (see [Section 8.1.6](#)). A host or a Switch may assume that the GPF timeouts configured across Downstream Ports at the same level in the hierarchy are identical. If a Switch detects a timeout, it shall set the Payload[8] in the response to indicate an error condition. This enables a CXL Root Port to detect GPF Phase 1 errors anywhere in the virtual hierarchy it spawns. If an error is detected by any Root Port in the coherency domain, the host shall set the Payload[8] flag during the Phase 2 flow, thereby informing every CXL device of an error during GPF Phase 1. Persistent devices may log this indication in a device-specific manner and make this information available to system software. If the host is positively aware that the GPF event will be followed by a power failure, it should set Payload[0] in the GPF Phase 1 request message. If the host cannot guarantee that the GPF event will be followed by a power failure, it shall not set Payload[0] in the GPF Phase 1 request message.

The CXL devices and switches must be able to receive and process GPF messages without dependency on any other PM messages. GPF messages do not use a credit, and CREDIT_RTN messages are not expected in response to a GPF request.

The host may reset the device any time after GPF Phase 2 completes.

If the host detection or processing of a GPF event and a reset event overlap, the host may process either event and ignore the other event. If the host detection or processing of a GPF event and an Sx event overlap, the host may process either event and ignore the other event. If host detects a GPF event while it is entering a lower power state, the host is required to process the GPF event in a timely manner.

9.8.2

Device Responsibilities

If a device supports GPF, it shall set bit 1 of the CAPABILITY_VECTOR field in its AGENT_INFO response (see [Table 3-1](#)). All CXL devices with the exception of eRCDs shall support GPF. An eRCD may support GPF functionality. If a device supports GPF, the Device shall respond to all GPF request messages regardless of whether the Device is required to take any action. The host may interpret a lack of response within a software-configured timeout window as an error. For example, a Type 3 device may or may not take any specific action during GPF Phase 1 other than generating a GPF Phase 1 response message.

Upon receiving a GPF Phase 1 request message, a CXL device shall execute the following steps in the specified order:

1. Stop injecting new CXL.cache transactions except for cache write backs described in step 3.

2. If CXL.cache capable and Payload[1]=1, disable caching. This will ensure that the device no longer caches any coherent memory and thereby not cache any writes that are received over the CXL interface in its CXL.cache.
3. If CXL.cache capable and Payload[1]=1, write back all modified lines in the device cache. The memory destination may be local or remote.
 - To minimize GPF latency, the device should ignore lines that are not dirty.
 - To minimize GPF latency, the device should not write back lines that it knows are mapped to volatile memory. The mechanism by which the device obtains this knowledge is beyond the scope of this specification.
 - The device must use device internal mechanisms to write back all dirty lines that are mapped to its local persistent HDM.
 - The device must write back all dirty lines that are not mapped to its local HDM and may be of persistent type. Each such dirty line must be written back to the destination HDM in two steps:
 - i. Issue DirtyEvict request to the host (see [Section 3.2.4.2.15](#)).
 - ii. Issue CLFlush request to the host (see [Section 3.2.4.2.13](#)).
4. Indicate that the device is ready to move to GPF Phase 2 by sending a GPF Phase 1 response message. Set the Payload[8] flag in the response if the Phase 1 processing was unsuccessful.

A device may take additional steps to reduce power draw from the system if the Payload[0] flag is set in the request message indicating that power failure is imminent. For example, a device may choose to not wait for responses to the previously issued reads before initiating the write back operation [step 3] above as long as the read responses do not impact persistent memory content.

Until the GPF Phase 2 request message is received, the device must respond to and complete any accesses that it receives over the CXL interface. This is to ensure that the other requesters can continue to make forward progress through the GPF flow.

Upon receiving a GPF Phase 2 request, a CXL device shall execute the following steps in the specified order:

1. If it is a persistent memory device and the Payload[8] flag is set, increment the Dirty Shutdown Count (see [Section 8.2.9.9.3.1](#)).
2. Flush internal memory buffers to local memory if applicable.
3. Acknowledge the request by sending a GPF Phase 2 response message.
4. Enter the lowest possible power state.

As this exchange may be performed in the event of an impending power loss, it is important that any flushing activity in either phase is performed in an expedient manner, and that the acknowledgment of each phase is sent as quickly as possible.

A device may have access to an alternate power source (e.g., a device with a large memory buffer may include a charged capacitor or battery) and may acknowledge GPF Phase 2 requests as soon as it has switched over to the alternate power source. Such a device shall ensure that PERST# assertion does not interfere with the local flush flow and shall correctly handle a subsequent power-up sequence even if the local flush is in progress.

A device is not considered to be fully operational after it receives a GPF Phase 1 Request. In this state, a device shall correctly process a Conventional Reset request, and return to operational state upon successful completion of these resets.

If the device detection or processing of a GPF event and a reset event overlap, the device may process either event and ignore the other event. If the device detection or processing of a GPF event and an Sx event overlap, the device may process either event and ignore the other event. If a device receives a GPF request while it is entering a lower power state, it shall process the GPF request in a timely manner.

A pooled device is composed of multiple LDs that are assigned to different Virtual Hierarchies. Because a GPF event may or may not be coordinated across these hierarchies, each LD shall be capable of independently processing GPF messages targeting that individual LD, without affecting any other LD within the MLD. An MLD cannot enter a lower power state until all LDs associated with the device have indicated that they are ready to enter the lower power state. In addition, the MLD must be able to process multiple GPF events (from different VCS targeting unique LDs).

If a device receives a GPF Phase 2 request message without a prior GPF Phase 1 request message, it shall respond to that GPF Phase 2 request message.

9.8.3

Energy Budgeting

It is often necessary to assess whether a system has sufficient energy to handle GPF during a power failure scenario. System software may use the information available in various CXL DVSEC registers along with its knowledge of the remainder of the system to make this determination.

This information may also be used to calculate appropriate GPF timeout values at various points in the CXL hierarchy. See the implementation note below. The timeout values are configured through GPF DVSEC for CXL Ports (see [Section 8.1.6](#)).

IMPLEMENTATION NOTE

System software may determine the total energy needs during power failure GPF. There may always be a nonzero possibility that power failure GPF may not successfully complete (e.g., under unusual thermal conditions or fatal errors). The goal of the system designer is to ensure that the probability of failure is sufficiently low and meets the system design objectives.

The following high-level algorithm may be followed for calculating timeouts and energy requirements

1. Iterate through every CXL device and calculate T1 and T2 as defined in Column "Time needed" in [Table 9-3](#).
2. Calculate T1MAX and T2MAX.
 - a. T1MAX = MAX of T1 values calculated for all devices plus propagation delay, host-side processing delays, and any other host/system-specific delays.
 - b. T2MAX = MAX of T2 values calculated for all devices in the hierarchy plus propagation delay, host-side processing delays, and any other host/system-specific delays. This could be same as GPF Phase 2 timeout at RC.
3. Calculate E1 and E2 for each device. See Column "Energy needed" in [Table 9-3](#).
4. Do summation over all CXL devices (E1+E2). Add energy needs for host and non-CXL devices during this window.

IMPLEMENTATION NOTE**Continued**

The GPF timeout registers in the root port and the Downstream Switch Port CXL Port GPF Capability structure may be programmed to T1MAX and T2MAX, respectively. Device active power is the amount of power that the device consumes in D0 state and may be reported by the device via Power Budgeting Extended Capability as defined in PCIe Base Specification. Cache size is reported via PCIe DVSEC for CXL devices (Revision 1). This computation may have to be redone periodically as some of these factors may change. When a CXL device is hot-added/removed, it may warrant recomputation. Refer to [Table 9-3](#).

Cache size, T2, and GPF Phase 2 Power parameters are reported by the device via GPF DVSEC for CXL devices (see [Section 8.1.7](#)). The other parameters are system dependent. System software may use ACPI HMAT to determine average persistent memory bandwidth, but the software could apply additional optimizations if it is aware of the specific persistent device the accelerator is operating on. In some cases, System Firmware may be the one performing this computation. Since System Firmware may or may not be aware of workloads, it may make conservative assumptions.

If the system determines that it does not have sufficient energy to handle all CXL devices, it may be able to take certain steps, such as to reconfigure certain devices to stay within the system budget by reducing the size of cache allocated to persistent memory or limit persistent memory usages. Several system level and device-level optimizations are possible:

- Certain accelerators may always operate on volatile memory and could skip the flush. For these accelerators, T1 would be 0.
- Device could partition cache among volatile vs. non-volatile memory and thus lower T1. Such partitioning may be accomplished with assistance from system software.
- A device could force certain blocks (e.g., execution engines) into a lower power state upon receiving a GPF Phase 1 request.
- Device may include a local power source and therefore could lower its T1 and T2.
- System software may configure all devices so that all T1s and T2s are roughly equal. This may require performance and/or usage model trade-offs.

Table 9-3. GPF Energy Calculation Example

Device Step	Time Needed	Energy Needed
Stop traffic generation	Negligible	Negligible
Disable caching	Negligible	Negligible
Write back cache content to persistent memory	T1= Cache size * % of lines in cache mapped to persistent memory / worst-case persistent memory bandwidth.	E1= T1MAX * device active Power
Flush local Memory buffers to local memory	T2	E2= T2 * GPF Phase 2 Power

9.9

Hot-Plug

By definition, RCDs and RCHs do not support Hot-Plug.

CXL Root Ports and CXL Downstream Switch Ports may support Hot-Add and managed Hot-Remove. All CXL Ports shall be designed to avoid electrical damage upon surprise Hot-Remove. All CXL switches and CXL devices, with the exception of eRCDs, shall be capable of being Hot-Plugged, subject to the Form Factor limitations. In a managed Hot-Remove flow, software is notified of a hot removal request. This provides CXL-aware system software the opportunity to write back device cachelines and to offline device memory prior to removing power. During a Hot-Add flow, CXL-aware system software discovers the CXL.cache and CXL.mem capabilities of the adapter and initializes them so they are ready to be used.

CXL leverages PCIe Hot-Plug model and Hot-Plug elements as defined in PCIe Base Specification and the applicable form-factor specifications.

CXL isolation is the mechanism that is used for graceful handling of Surprise Hot-Remove of CXL adapters. If a CXL adapter that holds modified lines in its cache is removed without any prior notification and CXL.cache isolation is not enabled, subsequent accesses to those addresses may result in timeouts that may be fatal to host operation. If a CXL adapter with HDM is removed without any prior notification and CXL.mem isolation is not enabled, subsequent accesses to HDM locations may result in timeouts that may be fatal to host operation.

All CXL Downstream Ports, including RCH Downstream Ports, shall hardwire the Hot-Plug Surprise bit in the Slot Capabilities register to 0. Software may leverage Downstream Port Containment capability of the Downstream Port to gracefully handle surprise hot removal of PCIe adapters or contain errors that result from surprise hot removal or Link Down of CXL adapters.

Support for Coherent Device Attribute Table (CDAT) by way of ReadTable DOE (see [Section 8.1.11](#)) is optional for eRCDs, but mandatory for all other CXL devices and is also mandatory for CXL switches. Software may use this interface to learn about performance and other attributes of the device or the Switch.

The Host Bridge and Upstream Switch Ports implement the HDM Decoder Capability structure. Software may program these to account for the HDM capacity with an appropriate interleaving scheme (see [Section 9.13.1](#)). Software may choose to leave the decoders unlocked for maximum flexibility and use other protections (e.g., page tables) to limit access to the registers. All unused decoders are unlocked by definition and software may claim these to decode additional HDM capacity during a Hot-Add flow.

All CXL.cache-capable devices, with the exception of eRCDs, shall implement the Cache Writeback and Invalidation capability (see [Section 9.6](#)). Software may use this capability to ensure that a CXL.cache-capable device does not have any modified cachelines prior to removing power.

Software shall ensure that the device has completed Power Management Initialization (see [Section 8.1.3.5](#)) prior to enabling its CXL.cache capabilities or CXL.mem capabilities if the device reports PM Init Completion Reporting Capable=1.

Software shall ensure that it does not enable a CXL.cache device below a given Root Port if the Root Port does not support CXL.cache. The Root Port's capabilities are exposed via the DVSEC Flex Bus Port Capability register. All CXL.cache-capable devices should expose the size of their cache via the DVSEC CXL Capability2 register. Software may cross-check this against the host's effective snoop filter capabilities (see [Section 8.2.4.23.2](#)) during Hot-Add of CXL.cache-capable device. Software may configure the Cache_SF_Coverage field in the DVSEC CXL Control register to indicate to

the device how much snoop filter capacity it should use (0 being a legal value). In extreme scenarios, software may disable CXL.cache devices to avoid snoop filter over-subscription.

During Hot-Add, System Software may reassess the GPF energy budget and take corrective action if necessary.

Hot-Add of an eRCD may result in unpredictable behavior if the device is exposed to software. The following mechanisms are defined to ensure that an eRCD that is hot-added in runtime is not discoverable by standard PCIe software:

- For Root Ports connected to Hot-Plug capable slots, it is recommended that System Firmware set the Disable_RCD_Training bit (see [Section 8.2.1.3.2](#)) after System Firmware PCIe enumeration completion, but before OS hand-off. This will ensure that a CXL root port will fail link training if an eRCD is hot-added. A Hot-Plug event may be generated in these cases, and the Hot-Plug handler may be invoked. The Hot-Plug handler may treat this condition as a failed Hot-Plug, notify the user, and then power down the slot.
- A Downstream Switch Port may itself be hot-added and cannot rely on System Firmware setting the Disable_RCD_Training bit. A Switch shall not report a Link Up condition and shall not report presence of an adapter when it is connected to an eRCD. System Firmware or CXL-aware software may still consult DVSEC Flex Bus Port Status (see [Section 8.2.1.3.3](#)) and discover that the Port is connected to an eRCD.

IMPLEMENTATION NOTE

CXL Type 3 device Hot-Add flow

1. System Firmware may prepare the system for a future Hot-Add (e.g., pad resources to accommodate the needs of an adapter to be hot-added).
2. User hot-adds a CXL memory expander in an empty slot. Downstream Ports bring up the link in CXL VH mode.
3. PCIe Hot-Plug interrupt is generated.
4. Bus driver performs the standard PCIe Hot-Add operations, thus enabling CXL.io. This process assigns BARs to the device.
5. CXL-aware software (e.g., CXL bus driver in OS, the device driver, or other software entity) probes CXL DVSEC capabilities on the device and ensures that the HDM is active. Memory may be initialized either by hardware, by the FW on the adapter or the device driver.
6. CXL-aware software configures the CXL DVSEC structures on the device, switches, and Host Bridge (e.g., GPF DVSEC, HDM decoders).
7. CXL-aware software notifies the OS memory manager about the new memory and its attributes such as latency and bandwidth. Memory manager processes a request and adds the new memory to its allocation pool.
8. The user may be notified via attention indicator or some other user interface of successful completion.

IMPLEMENTATION NOTE

CXL Type 3 device-managed Hot-Remove flow

1. User initiates a Hot-Remove request via attention button or some other user interface.
2. The standard PCIe Hot-Remove flow is triggered (e.g., via Hot-Plug interrupt if attention button was used).
3. CXL-aware software (e.g., CXL bus driver in OS, the device driver, or other software entity) probes CXL DVSEC capabilities on the device and determines active memory ranges.
4. CXL-aware software requests the OS memory manager to vacate these ranges.
5. If the Memory Manager is unable to fulfill this request (e.g., because of presence of pinned pages), CXL-aware software will return an error to the Hot-Remove handler, which will notify the user that the operation has failed.
6. If the Memory Manager is able to fulfill this request, CXL-aware system software reconfigures HDM Decoders in CXL switches and Root Ports. This is followed by the standard PCIe Hot-Remove flow that will process CXL.io resource deallocation.
7. If the PCIe Hot-Remove flow fails, the user is notified that the Hot-Remove operation has failed; otherwise, the user is notified that the Hot-Remove flow has successfully completed.

IMPLEMENTATION NOTE

CXL Type 1 device Hot-Add flow

1. System Firmware may prepare the system for a future Hot-Add (e.g., pad MMIO resources to accommodate the needs of an adapter to be hot-added).
2. The user Hot-Adds a CXL Type 1 device in an empty slot. The Downstream Port brings up the link in CXL VH operation with 68B Flit mode.
3. A PCIe Hot-Plug interrupt is generated.
4. The bus driver performs the standard PCIe Hot-Add operations, thus enabling CXL.io. This process assigns BARs to the device.
5. CXL-aware software (e.g., CXL bus driver in OS, the device driver, or other software entity) probes CXL DVSEC capabilities on the device. If the device is hot-added below a Root Port that cannot accommodate a CXL.cache-enabled device, Hot-Add is rejected. If the device has a cache that is larger than what the host snoop filter can handle, Hot-Add is rejected. The user may be notified via attention indicator or some other user interface of this.
6. If the above checks pass, CXL-aware software configures the CXL DVSEC structures on the device and switches (e.g., GPF DVSEC).
7. The Hot-Add flow is complete. The user may be notified via attention indicator or some other user interface of successful completion.

9.10 Software Enumeration

This section describes two types of CXL device enumeration flows. Although discovery of CXL devices follows the PCIe model, there are some important differences:

- RCD Enumeration: As the name suggests, RCD mode (see [Section 9.11.1](#)) imposes some restrictions and leads to a much-simpler enumeration flow. Each RCD is exposed to host software as one or more PCIe Root Complex Integrated Endpoints as indicated by setting PCI Express* Capabilities Register.Device/Port Type=RCIEP. Each RCD creates a new PCIe enumeration hierarchy that is compatible with an ACPI-defined PCIe Host Bridge (PNP ID PNP0A08). The RCD enumeration flow is described in [Section 9.11](#).
- CXL VH enumeration: A CXL root port is the root of a CXL VH. A CXL VH may include zero or more CXL switches, zero or more PCIe switches, zero or more PCIe devices, and one or more CXL devices that are not in RCD mode. A CXL VH represents a software view and may differ from the physical topology. The CXL VH enumeration flow is described in [Section 9.12](#).

A CXL device cannot claim I/O resources because it is not a Legacy Endpoint. For the definition of Legacy Endpoint, see PCIe Base Specification.

9.11 RCD Enumeration

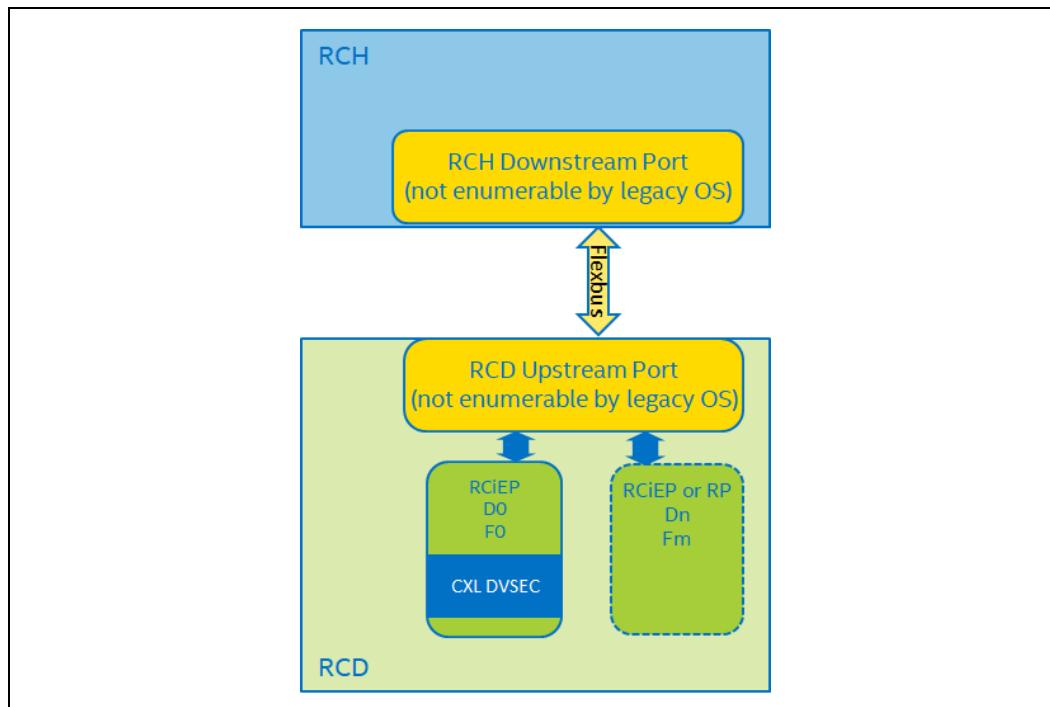
9.11.1 RCD Mode

Restricted CXL device (RCD) mode is a CXL operating mode with the following restrictions:

- Hot-Plug is not supported
- CXL devices operating in this mode always set the Device/Port Type field in the PCI Express Capabilities register to RCiEP
- Flit modes other than 68B Flit mode are not supported
- Routing types other than HBR are not supported
- Link is not visible to non-CXL-aware software

9.11.2 PCIe Software View of an RCH and RCD

Figure 9-4. PCIe Software View of an RCH and RCD



Because the CXL link is not exposed to CXL-unaware OSs, the System Firmware view of the hierarchy is different than that of the CXL-unaware OS.

9.11.3 System Firmware View of an RCH and RCD

The functionality of the RCH Downstream Port and the RCD Upstream Port can be accessed via memory mapped registers. These will not show up in a standard PCI* bus scan by CXL-unaware OSs. The base addresses of these registers are set up by System Firmware and System Firmware can use that knowledge to configure CXL.

System Firmware configures the RCH Downstream Port to decode the memory resource needs of the CXL device as expressed by PCIe BARs and Upstream Port BAR(s). PCIe BARs are not to be configured to decode any HDM that are associated with the CXL device.

9.11.4 OS View of an RCH and RCD

Each RCH-RCD pair is presented as one ACPI Host bridge. The _BBN method for this Host Bridge matches the bus number that hosts the RCD.

This ACPI Host Bridge spawns a legal PCIe hierarchy. All PCIe Endpoints located in the RCD are children of this ACPI Host Bridge. These Endpoints may appear directly on the Root bus number or may appear behind a Root Port located on the Root bus.

The _CRS method for PCIe root bridge returns bus and memory resources claimed by the CXL Endpoints. _CRS response does not include HDM on CXL.mem-capable devices, and does not comprehend any Upstream Port BARs (hidden from OS).

A CXL-aware OS may use CXL Early Discovery Table (CEDT) or _CBR object in ACPI namespace to locate the Downstream Port registers and Upstream Port registers. CEDT enumerates all CXL Host Bridges that are present at the time of OS hand-off and _CBR is limited to CXL Host Bridges that are hot-added.

9.11.5

System Firmware-based RCD Enumeration Flow

Because RCDs do not support Hot-Add, RCDs can be fully enumerated by System Firmware prior to OS hand-off.

In the presence of RCD mode, the hardware autonomous mode selection flow cannot automatically detect the number of retimers. If the system includes retimers, the System Firmware shall follow these steps to ensure that the number of retimers is correctly configured:

1. Prior to the link training, the System Firmware should set the DVSEC Flex Bus Port control register, based on the available information, to indicate whether there are 0, 1, or 2 retimers present. (It is possible that retimers on a CXL add-in card or a backplane may not be detected by the System Firmware prior to link training and the initial programming may not account for all retimers in the path.)
2. After the link training completes successfully or fails, the System Firmware should read the Retimer Presence Detected and Two Retimers Presence Detected values logged in the PCIe standard Link Status 2 register and determine whether they are consistent with what was set in the Flex Bus Port DVSEC in the previous step. If they are different, the System Firmware should bring the Link Down by setting the Link Disable bit in the Downstream Port, update the Retimer1_Present and Retimer2_Present bits in the Flex Bus Port DVSEC, and then re-initiate link training.

9.11.6

RCD Discovery

1. Parse configuration space of Device 0, Function 0 on the Secondary bus # and discover CXL-specific attributes. These are exposed via PCIe DVSEC for CXL Devices Capability structures. See [Section 8.1.3](#).
2. If the device supports CXL.cache, configure the CPU coherent bridge and then set the Cache_Enable bit in the DVSEC CXL Control register.
3. If the device supports CXL.mem, check Mem_HwInit_Mode by reading the DVSEC CXL Capability register and determine the number of supported HDM ranges by reading the HDM_Count field in the same register.
4. If Mem_HwInit_Mode=1:
 - The device must set the Memory_Info_Valid bit in each applicable DVSEC CXL Range X Size Low register (X=1, 2) within 1 second of reset deassertion.
 - The device must set the Memory_Active_Valid bit in each applicable DVSEC CXL Range X Size Low register (X=1, 2) within the Memory_Active_Timeout duration of reset deassertion.
 - When Memory_Info_Valid is 1, System Firmware reads the Memory_Size_High and Memory_Size_Low fields for each supported HDM range. If System Firmware cannot delay boot until the Memory_Active bit is set, the System Firmware may continue with HDM base assignment and may delay OS hand-off until the Memory_Active bit is set.
 - System Firmware computes the size of each HDM range and maps those in system address space.
 - System Firmware programs the Memory_Base_Low and the Memory_Base_High fields for each HDM range.
 - System Firmware programs the ARB/MUX arbitration control registers if necessary.

- System Firmware sets CXL.mem Enable. Once Memory_Active=1, Any subsequent accesses to HDM are decoded and routed to the local memory by the device.
 - Each HDM range is later exposed to the OS as a separate, memory-only NUMA node via ACPI SRAT.
 - System Firmware obtains CDAT from the UEFI device driver or directly from the device via Table Access DOE (see [Section 8.1.11](#)) and then uses this information during construction of the memory map, ACPI SRAT, and ACPI HMAT. See ACPI Specification, CDAT Specification, and UEFI Specification for further details.
5. If Mem_HwInit_Mode =0
- The device must set the Memory_Info_Valid bit in each applicable DVSEC CXL Range X Size Low register (X=1, 2) within 1 second of reset deassertion.
 - When Memory_Info_Valid is 1, System Firmware reads the Memory_Size_High and Memory_Size_Low fields for supported HDM ranges.
 - System Firmware computes the size of each HDM range and maps those in system address space.
 - System Firmware programs the Memory_Base_Low and the Memory_Base_High fields for each HDM range.
 - System Firmware programs the ARB/MUX arbitration control registers if necessary.
 - System Firmware sets CXL.mem Enable. Any subsequent accesses to the HDM ranges are decoded and completed by the device. The reads shall return all 1s and the writes will be dropped.
 - Each HDM range is later exposed to the OS as a separate, memory-only NUMA node via ACPI SRAT.
 - If the memory is initialized prior to OS boot by UEFI device driver:
 - The UEFI driver is responsible for causing Memory_Active to be set. The driver can accomplish that by device-specific methods, such as by setting a device-specific register bit.
 - After Memory_Active is set, any subsequent accesses to the HDM range are decoded and routed to the local memory by the device.
 - System Firmware uses the information supplied by UEFI driver or Table Access DOE (see [Section 8.1.11](#)) during construction of the memory map and ACPI HMAT. See UEFI Specification for further details.
 - If the memory is initialized by an OS device driver post OS boot:
 - System Firmware may use the information supplied by UEFI driver or Table Access DOE (see [Section 8.1.11](#)) during construction of the memory map and ACPI HMAT. See UEFI Specification for further details. In the future, a CXL-aware OS may extract this information directly from the device via Table Access DOE.
 - At OS hand-off, System Firmware reports that the memory size associated with HDM NUMA node is 0.
 - The OS device driver is responsible for causing the Memory_Active bit to be set to 1 by using device-specific methods after memory initialization is complete. Any subsequent accesses to the HDM are decoded and routed to the local memory by the device.
 - Memory availability is signaled to the OS via an OS-specific mechanism.

CXL.io resource needs are discovered as part of PCIe enumeration. PCIe Root Complex registers, including Downstream Port registers, are appropriately configured to decode these resources. CXL Downstream Ports and Upstream Ports require MMIO resources. These are also accounted for during this process.

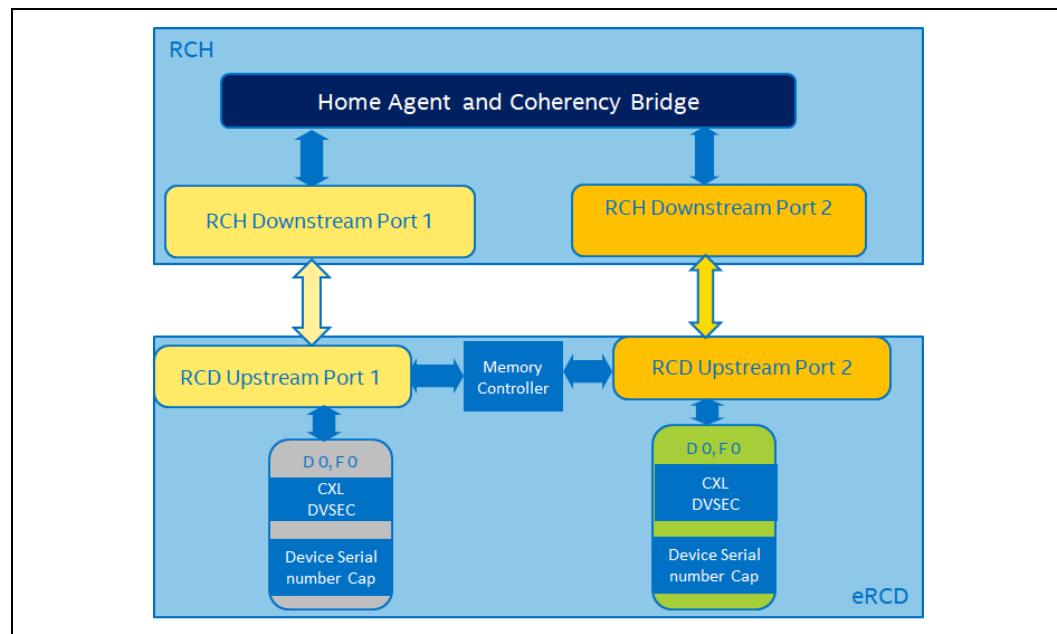
System Firmware programs the memory base and limit registers in the Downstream Port to decode CXL Endpoint MMIO BARs, CXL Downstream Port MMIO BARs, and CXL Upstream Port MMIO BARs.

9.11.7 eRCDs with Multiple Flex Bus Links

This section is applicable only to eRCDs that are directly connected to an eRCH. It does not apply to CXL VH. Also, it does not apply to eRCDs that are connected to CXL switches.

9.11.7.1 Single CPU Topology

Figure 9-5. One CPU Connected to a Dual-Headed RCD by Two Flex Bus Links



In this configuration, the System Firmware shall report two PCI Host Bridges to the OS, one that hosts Device 0, Function 0 on the left, and a second one that hosts Device 0, Function 0 on the right. Both Device 0, Function 0 instances implement PCIe DVSEC for CXL Devices and a Device Serial Number PCIe Extended Capability. A Vendor ID and serial number match indicates that the two links are connected to a single CXL device, which enables System Firmware to perform certain optimizations.

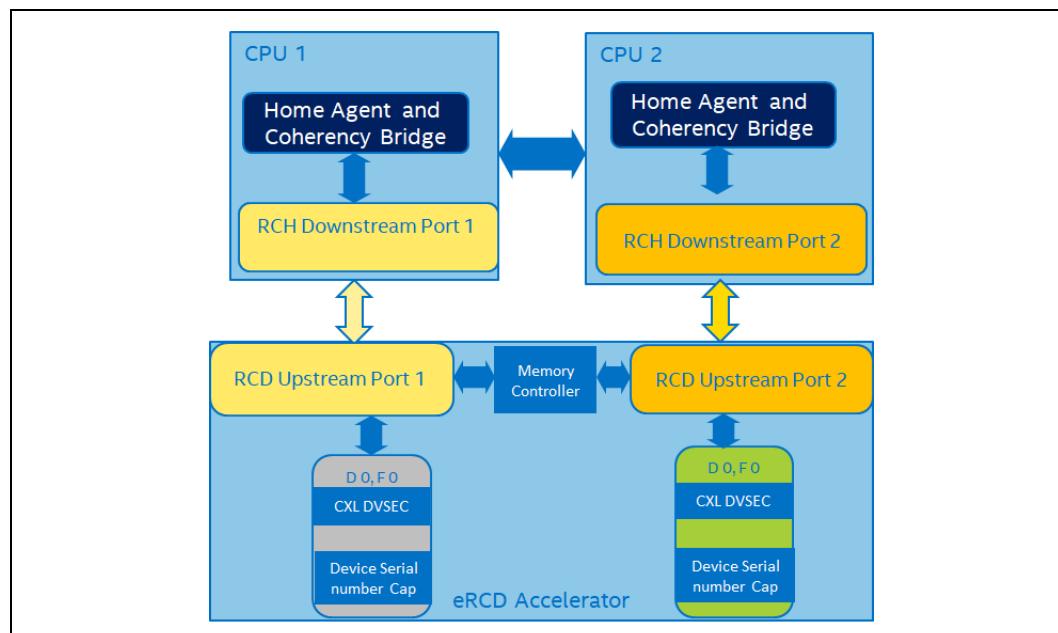
In some cases, the CXL device may expose a single CXL device function that is managed by the CXL device's driver, whereas the other Device 0, Function 0 represents a dummy device. In this configuration, application software may submit work to the single CXL device instance. However, the CXL device hardware is free to use both links for traffic and snoops as long as the programming model is not violated.

The System Firmware maps the HDM into system address space using the rules listed in [Table 9-4](#).

Table 9-4. Memory Decode Rules in Presence of One CPU/Two Flex Bus Links

Left D0, F0 Mem_Capable	Left D0, F0 Mem_Size	Right D0, F0 Mem_Capable	Right D0, F0 Mem_Size	System Firmware Requirements
0	N/A	0	N/A	No HDM.
1	M	0	N/A	Range of size M decoded by Left Flex Bus link. Right Flex Bus link does not receive CXL.mem traffic.
0	N/A	1	N	Range of size N decoded by Right Flex Bus link. Left Flex Bus link does not receive CXL.mem traffic.
1	M	1	N	Two ranges set up, Range of size M decoded by Left Flex Bus link, Range of size N decoded by Right Flex Bus link.
1	M	1	0	Single range of size M. CXL.mem traffic is interleaved across two links.
1	0	1	N	Single range of size N. CXL.mem traffic is interleaved across two links.

9.11.7.2 Multiple CPU Topology

Figure 9-6. Two CPUs Connected to One CXL Device by Two Flex Bus Links

In this configuration, System Firmware shall report two PCI Host Bridges to the OS, one that hosts Device 0, Function 0 on the left, and a second one that hosts Device 0, Function 0 on the right. Both Device 0, Function 0 instances implement PCIe DVSEC for CXL Devices and a Device Serial Number PCIe Extended Capability. A Vendor ID and serial number match indicates that the two links are connected to a single accelerator, which enables System Firmware to perform certain optimizations.

In some cases, the accelerator may choose to expose a single accelerator function that is managed by the accelerator device driver and handles all work requests. This may be necessary if the accelerator framework or applications do not support distributing work across multiple accelerator instances. Even in this case, both links should spawn a legal

PCIe Host Bridge hierarchy with at least one PCIe function. However, the accelerator hardware is free to use both links for traffic and snoops as long as the programming model is not violated. To minimize the snoop penalty, the accelerator needs to be able to distinguish between the system memory range decoded by CPU 1 vs. CPU 2. The device driver can obtain this information via ACPI SRAT and communicate it to the accelerator using device-specific mechanisms.

The System Firmware maps the HDM into system address space using the following rules. Unlike the single CPU case, the System Firmware shall never interleave the memory range across the two Flex Bus links.

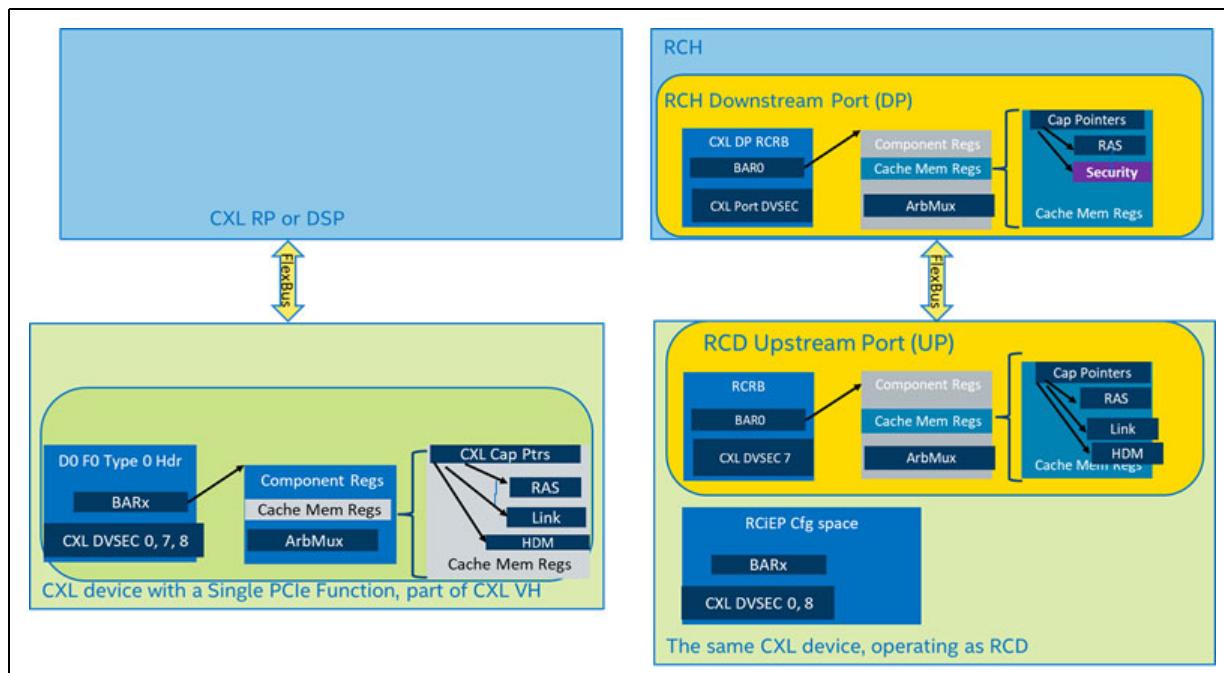
Table 9-5. Memory Decode Rules in Presence of Two CPU/Two Flex Bus Links

Left D0, F0 Mem_Capable	Left D0, F0 Mem_Size	Right D0, F0 Mem_Capable	Right D0, F0 Mem_Size	System Firmware Requirements
0	N/A	0	N/A	No HDM
1	M	0	N/A	Range of size M decoded by Left Flex Bus link. Right Flex Bus link does not receive CXL.mem traffic.
1	M	1	0	
0	N/A	1	N	Range of size N decoded by Right Flex Bus link. Left Flex Bus link does not receive CXL.mem traffic.
1	0	1	N	
1	M	1	N	Two ranges set up, Range of size M decoded by Left Flex Bus link, Range of size N decoded by Right Flex Bus link.

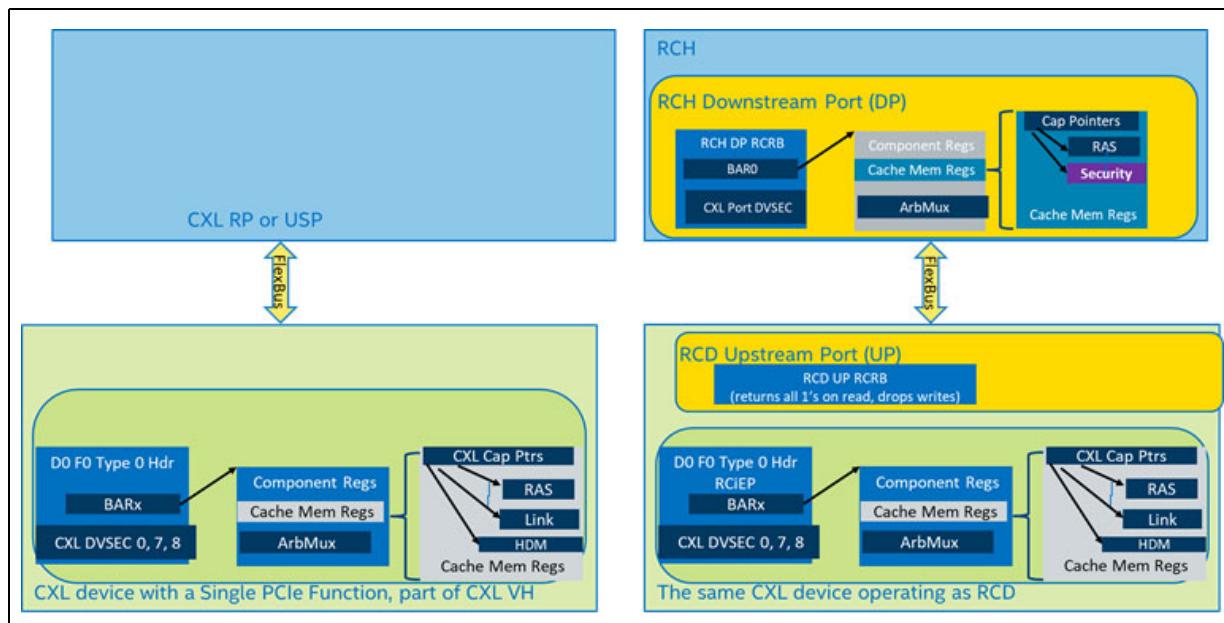
9.11.8 CXL Devices Attached to an RCH

When an eRCD is attached to an RCH, the register layout matches [Figure 9-4](#).

When a CXL device other than an eRCD is attached to a CXL RP or a CXL DSP, the device's Upstream Port registers are accessed via the CXL Device's PCIe Configuration space and BAR. A CXL device may be designed so that the layout of the device's Upstream Port and Component Registers follow [Figure 9-4](#) when connected to an RCH. For such a device, some of these registers must be remapped so that they are accessible via an RCD Upstream Port RCRB (see [Section 8.2.1.2](#), [Section 8.2.1.3](#), and [Section 8.2.2](#)). This register remapping is illustrated in [Figure 9-7](#). The left half shows the register layout when a CXL device with a single PCIe Function is connected to a CXL Root Port or CXL DSP. The right half shows the register layout when the same device is connected to an RCH. Such a device shall capture the upper address bits [63:12] of the first memory read received after link initialization as the base address of the Upstream Port RCRB (see [Section 8.2.1.2](#)).

Figure 9-7. CXL Device Remaps Upstream Port and Component Registers

A CXL device may be designed so that the layout of the device's Upstream Port and Component Registers still follows the CXL device layout for a CXL VH when connected to an RCH. In that case, the register remapping is unnecessary. This is illustrated in [Figure 9-8](#). The left half shows the register layout when a CXL device with a single PCIe Function is connected to a CXL Root Port or a CXL DSP. The right half shows the register layout when the same device is connected to an RCH. Such a device shall capture the upper address bits [63:12] of the first memory read received after link initialization as the base address of the Upstream Port RCRB, but all reads to the Upstream Port RCRB range shall return all 1s. Additionally, all writes shall be completed, but silently dropped by such a device. The software may rely upon this behavior to detect a device. Note that the DWORD read to RCRB Base + 4 KB is guaranteed to return a value other than FFFF FFFFh when directed at an eRCD or a CXL device that follows the [Figure 9-4](#) register layout when connected to an RCH (see [Figure 8-10](#)). An RCD is also permitted to implement the register mapping scheme shown in the right half of [Figure 9-8](#). In both cases, the RCD appears as an RCiEP.

Figure 9-8. CXL Device that Does Not Remap Upstream Port and Component Registers

IMPLEMENTATION NOTE

Host Firmware/Software Flow for detecting the RCD Registers Mapping Scheme

1. System Firmware reads DVSEC Flex Bus Port Status register (see [Section 8.2.1.3.3](#)) in the Downstream Port to determine whether the link trained up in RCD mode. If the Cache_Enabled bit or Mem_Enabled bit is set to 1, but the CXL 68B Flit and VH Enabled bit is cleared to 0, it indicates RCD mode.
2. If an RCD is detected, System Firmware programs the Downstream Port registers to decode the 8-KB RCRB range among other memory ranges.
3. System Firmware issues a DWORD read to the address RCRB Base + 4 KB. As explained in [Section 8.2.1.2](#), the device captures the address and assigns it as the base of RCRB. The device implementation may rely on a read to RCRB Base + 4 KB since the CXL 1.1 specification requires such a read.
4. If the returned data is not FFFF FFFFh, the System Firmware assumes that the register layout follows the right side of [Figure 9-7](#) and enumerates the device accordingly.
5. If the returned data is FFFF FFFFh and the Register Locator DVSEC includes a pointer to the Component Registers, the System Firmware assumes that the register layout follows the right side of [Figure 9-8](#) and enumerates the device accordingly.

9.12

CXL VH Enumeration

At the top level, a CXL system may be represented to the system software as zero or more CXL Host bridges, and zero or more PCIe Host Bridges. A CXL Host Bridge is a software concept that represents one of the following:

- A collection of CXL Root Ports that share some logic, such as CHBCR

- An RCH-RCD pair
- One or more CXL Root Complex Integrated Endpoints, all of which are part of the Root Complex and appear at the same bus number

Enumeration of PCIe Host Bridges and PCIe hierarchy underneath them is governed by PCIe Base Specification. Enumeration of CXL Host Bridges is described below.

In an ACPI-compliant system, CXL Host Bridges are identified with an ACPI Hardware ID (HID) of "ACPI0016". CXL Early Discovery Table (CEDT) may be used to differentiate between the three software concepts listed above. RCD enumeration is described in [Section 9.11](#).

9.12.1 CXL Root Ports

Each CXL Host Bridge is associated with a Base Bus Number. If the Host Bridge is not associated with RCDs or CXL RCiEPs, that bus number shall contain one or more CXL Root Ports. These Root Ports appear in PCIe configuration space with a Type 1 header, and the Device/Port Type field in the PCIe Capabilities Register shall identify these as standard PCIe Root Ports. Unless specified otherwise, CXL Root Ports may implement all Capabilities that are defined in PCIe Base Specification as legal for PCIe Root Ports. These Root Ports can be in one of four states:

- Disconnected
- Connected to an eRCD
- Connected to CXL Device that is not an eRCD, or connected to a CXL Switch
- Connected to a PCIe Device/Switch

[Section 9.12.3](#) describes how software can determine the current state of a CXL Root Port and the corresponding enumeration algorithm.

9.12.2 CXL Virtual Hierarchy

CXL Root Ports may be directly connected to a CXL device that is not an eRCD, or a CXL Switch. These Root Ports spawn a CXL Virtual Hierarchy (VH). Enumeration within a CXL VH is described below.

These CXL devices appear as a standard PCIe Endpoints with a Type 0 Header. The CXL device's primary function (Device 0, Function 0) shall carry one instance of CXL DVSEC ID 0 with Revision 1 or greater. Software may use this DVSEC instance to distinguish a CXL device from an ordinary PCIe device. Unless specified otherwise, CXL devices may implement all Capabilities that are defined in PCIe Base Specification as legal for PCIe devices.

A CXL VH may include zero or more CXL switches. Specific configuration constraints are documented in [Chapter 7.0](#). From an enumeration software perspective, each CXL Switch consists of one Upstream Switch Port and one or more Downstream Switch Ports.

The configuration space of the Upstream Switch Port of a CXL Switch has a Type 1 header and the Device/Port Type field in the PCIe Capabilities Register shall identify it as an Upstream Port of a PCIe Switch. The configuration space carries one instance of the CXL DVSEC ID 3 and one instance of DVSEC ID 7. The DVSEC Flex Bus Port Status register in CXL DVSEC ID 7 structure of the peer Port shall indicate that CXL VH operation with 68B Flit mode was negotiated with the Upstream Switch Port during link training. Unless specified otherwise, CXL Upstream Switch Ports may implement all Capabilities that are defined in PCIe Base Specification as legal for PCIe Upstream Switch Ports.

The configuration space of a Downstream Switch Port of CXL Switch also has a Type 1 header, but the Device/Port Type field in the PCIe Capabilities Register shall identify these as a Downstream Port of a PCIe Switch. Unless specified otherwise, CXL Downstream Switch Ports may implement all Capabilities that are defined in PCIe Base Specification as legal for PCIe Downstream Switch Ports. All these Ports are CXL capable and can be in one of four states, just like the CXL Root Ports:

- Disconnected
- Connected to an eRCD
- Connected to CXL Device that is not an eRCD, or connected to a CXL Switch
- Connected to a PCIe Device/Switch

[Section 9.12.3](#) describes how software can determine the current state of a CXL Downstream Switch Port and the corresponding enumeration algorithm.

A CXL Downstream Switch Port may be connected to another CXL Switch or a CXL device. The rules for enumerating CXL switches and CXL devices are already covered earlier in this section.

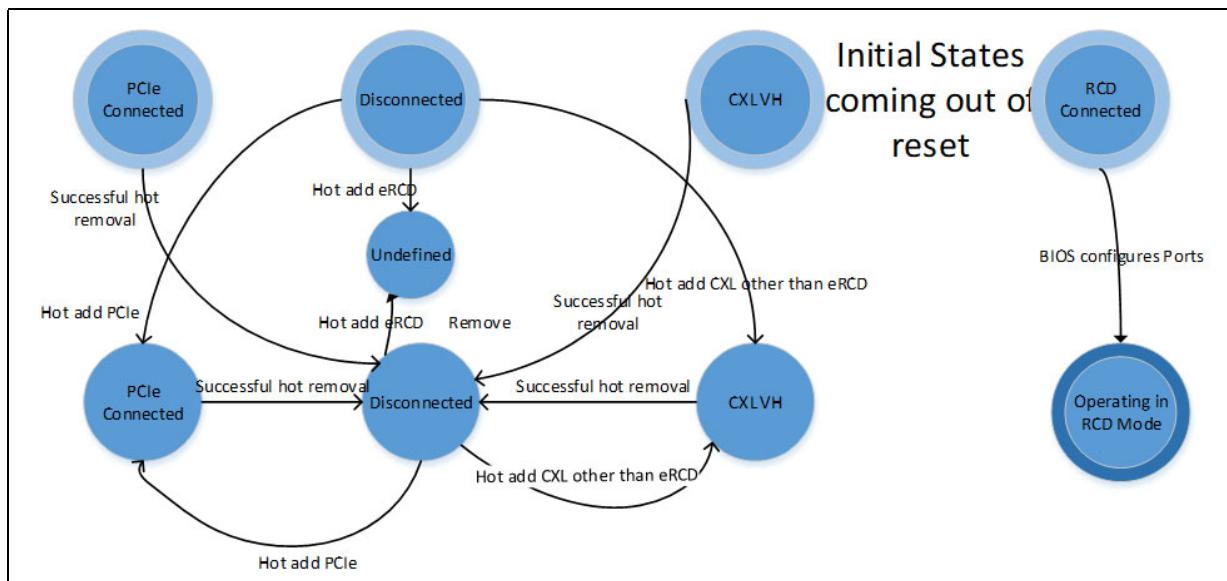
9.12.3

Enumerating CXL RPs and DSPs

Software may use the combination of the Link Status registers and the CXL DVSEC ID 7 capability in root port or DSP configuration space to determine which state a CXL Downstream Port is in, as follows:

1. CXL root port or DSP is in the Disconnected state when they do not have an active link. The status of the link can be detected by following PCIe Base Specification. If the link is not up, software shall ignore the CXL DVSEC ID 3 and the CXL DVSEC ID 7 capability structures. A Hot-Add event may transition a Disconnected Port to a CXL Connected state or a PCIe Connected state. Hot-adding an eRCD adapter will transition the Port to an Undefined state.
2. CXL root port or DSP connected to a CXL device that is not an RCD or connected to a CXL switch shall expose one instance of the CXL DVSEC ID 3 and one instance of the CXL DVSEC ID 7 capability structures. The DVSEC Flex Bus Port Status register in the CXL DVSEC ID 7 structure shall indicate that CXL VH operation with 68B Flit mode was successfully negotiated during link training. System Firmware may leave the Unmask SBR and the Unmask Link Disable bits in the Port Control register of the Downstream Port at the default (0) values to prevent CXL-unaware PCIe software from resetting the device and the link, respectively.
3. CXL root port or DSP connected to an eRCD shall expose one instance of the CXL DVSEC ID 3 and one instance of the CXL DVSEC ID 7 capability structures. The DVSEC Flex Bus Port Status register in the CXL DVSEC ID 7 structure shall indicate that CXL VH operation with 68B Flit mode was not negotiated, but that either the CXL.cache protocol or the CXL.mem protocol was negotiated during link training. There are two possible substates:
 - a. Not Operating with RCH Downstream Port addressing - Immediately after the link negotiation, the Port registers appear in the PCIe configuration space with a Type 1 header.
 - b. Operating with RCH Downstream Port addressing - System Firmware may program the RCRB Base register in the Port's CXL DVSEC ID 3 capability structure to transition the Port to this mode. Once the Port is in this mode, it can only transition out of the mode after a reset. A Downstream Port operating in this mode shall ignore hot reset requests received from the Upstream Port.
4. CXL root port or DSP connected to a PCIe device/switch may or may not expose the CXL DVSEC ID 3 and the CXL DVSEC ID 7 capability structures.

- If the PCIe root port configuration space contains an instance of the CXL DVSEC ID 3 structure, it shall also contain an instance of the CXL DVSEC ID 7 structure.
- If the PCIe root port configuration space contains an instance of the CXL DVSEC ID 7 structure, the DVSEC Flex Bus Port Status register shall indicate that this Port did not train up in CXL mode. Software shall ignore the contents of the CXL DVSEC ID 3 structure for such a Port.

Figure 9-9. CXL Root Port/DSP State Diagram

If the Port is in the disconnected state, the branch does not need further enumeration.

If the Port is connected to a CXL device other than an eRCD or connected to a CXL switch, the software follows [Section 9.12.2](#) for further enumeration until it reaches the leaf endpoint.

If the Port is connected to an RCD, the software follows [Section 9.12.4](#) to enumerate the device.

If the Port is connected to a PCIe device/switch, the enumeration flow is governed by PCIe Base Specification.

9.12.4 eRCD Connected to a CXL RP or DSP

An eRCD may be connected to a CXL Root Port or a CXL Downstream Switch Port. Each RCD Function must report itself as an RCIEP and therefore cannot appear, to software, to be below a PCIe-enumerable Downstream Port. System Firmware is responsible for detecting such a case and reconfiguring the CXL Ports in the path so that the RCD appears to software to be directly connected to an RCH Downstream Port and not in a CXL VH.

9.12.4.1 Boot time Reconfiguration of CXL RP or DSP to Enable an eRCD

- At reset, the Downstream Port registers are visible in the PCIe configuration space with a Type 1 header. During enumeration, System Firmware shall identify all the Downstream Ports that are connected to the eRCD by reading the DVSEC ID 7 register instead of the Link status registers.

- If the link training was successful, the DVSEC Flex Bus Port Status register in the CXL DVSEC ID 7 structure shall indicate that CXL VH operation with 68B Flit mode was not negotiated, but shall indicate that either the CXL.cache protocol or the CXL.mem protocol was negotiated during link training.
 - If the link training was unsuccessful, the DVSEC Flex Bus Port Received Modified TS Data Phase1 Register in the CXL DVSEC ID 7 structure shall indicate that the device is CXL capable but not CXL VH capable. A DSP shall not report link-up status in the PCIe Link Status register when the DSP detects an eRCD on the other end to prevent the CXL-unaware software from discovering the eRCD.
2. System Firmware identifies MMIO and bus resource needs for all RCDs below a CXL root port. System Firmware adds MMIO resources needed for the RCH Downstream Port RCRB and RCD Upstream Port RCRB (8-KB MMIO per link) and CXL Component registers (128-KB MMIO per link).
 3. System Firmware assigns MMIO and bus resources and programs the Alternate MMIO Base/Limit and Alternate Bus Base/Limit registers in all the Root Ports and the Switch Ports in the path and the eRCD BARs except the Downstream Ports that are directly connected to eRCDs. These Alternate decoders are described in [Section 8.1.5](#).
 4. System Firmware sets the Alt BME and Alt Memory and ID Space Enable bits in all the Root Ports and the Switch Ports in the path of every eRCD.
 5. For each Downstream Port that is connected to an eRCD, System Firmware programs the CXL RCRB Base Address. System Firmware then writes 1 to the CXL RCRB Enable bit, which transitions the port addressing to RCH addressing. The Downstream Port registers now appear in MMIO space at CXL RCRB Base and not in configuration space. System Firmware issues a read to the address CXL RCRB Base + 4 KB. The RCD Upstream Port captures its RCRB Base as described in [Section 8.1.5](#). System Firmware configures Upstream Port and Downstream Port registers, as necessary. If this is a DSP, the Downstream Port shall ignore any hot reset requests received from the Upstream Port.
 6. System Firmware configures the eRCD, using the algorithm described in [Section 9.11.6](#).

The System Firmware shall report each RCD under a separate Host Bridge and not as part of the CXL VH.

The Switch shall ensure that there is always a DSP visible at Device 0, Function 0.

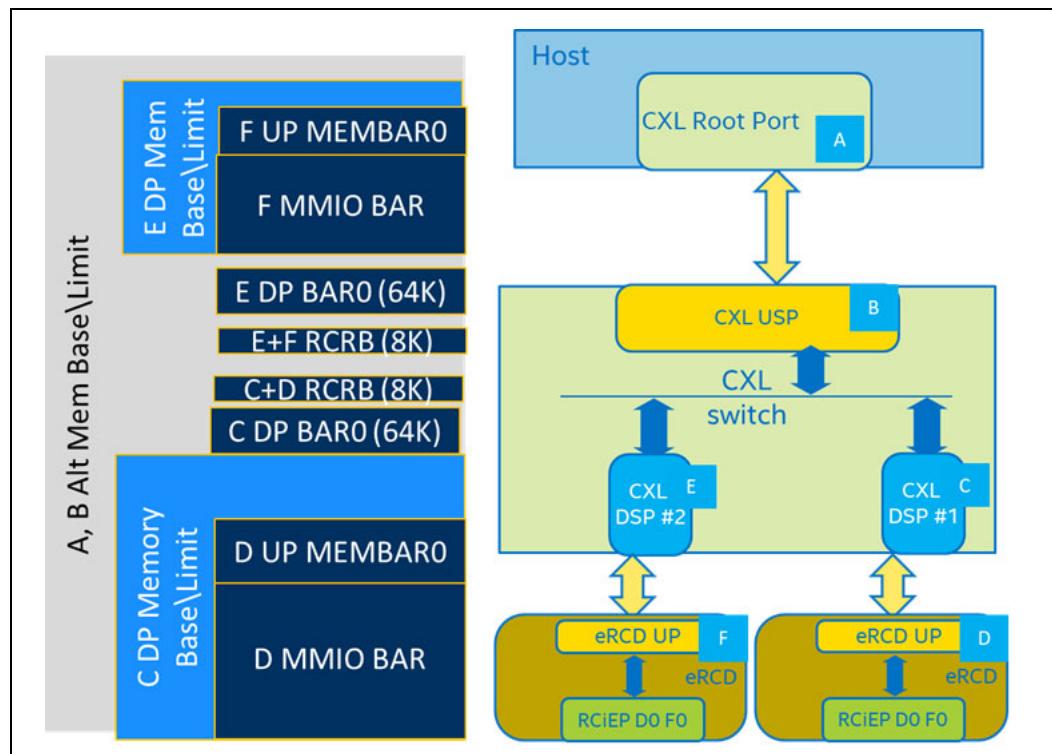
These concepts are illustrated by the configuration shown in [Figure 9-10](#). In this configuration, eRCD F and D are attached to a CXL Switch. The Switch DSPs are labeled E and C. The Switch USP and the CXL Root Port are labeled B and A, respectively. The left half of [Figure 9-10](#) represents the address map and how the normal decoders and the Alt Mem decoders of A, B, C, and E are configured.

If the host accesses an MMIO address belonging to D, the access flows through A, B, and C as follows:

1. Host issues a read.
2. A Alt Decoder positively decodes the access and sends to B because A's Alt MSE=1.
3. B Alt Decoder positively decodes the access because B's Alt MSE=1.
4. C normal decoder positively decodes the access and forwards it to D because C MSE=1.
5. D positively decodes and responds because D MSE=1.

EVALUATION COPY

Figure 9-10. eRCD MMIO Address Decode - Example



The left half of Figure 9-11 represents the configuration space map for the same configuration as in Figure 9-10 and how the bus decoders and the Alt Mem decoders of A, B, C, and E are configured.

If the host accesses configuration space of F, the access flows through A, B, and E as follows:

1. Host issues configuration read to F's configuration space
2. A's Alt Decoder positively decodes, forwards to B as Type 1
3. B's Alt Decoder positively decodes, forwards down as Type 1
4. E's RCRB regular decoder positively decodes, forwards to F as Type 0 because the bus number matches E's RCRB Secondary Bus number
5. F positively decodes and responds

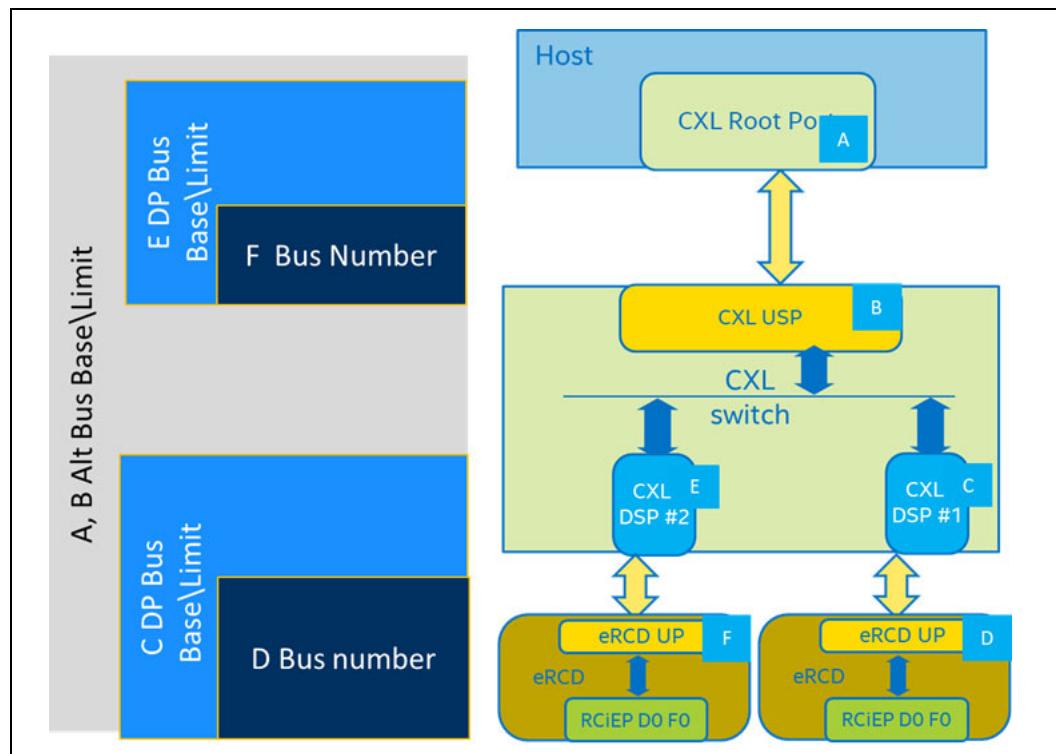
If D detects a protocol or link error, the error signal will flow to the system via the following path:

1. D issues ERR_ message with the Requester ID of D.
2. C shall not expose DPC capability.
3. C forwards ERR_ message to B.
4. B forwards the message to A.
5. A forwards the message to RCEC in the Root Complex because the requester's bus number hits Alt Bus Decoder.
6. RCEC generates MSI if enabled.

EVALUATION COPY

7. Root Complex Event Collector Endpoint Association Extended Capability of RCEC describes that it can handle errors from bus range = Alt Bus Decoder in RP.
8. A shall not trigger DPC upon ERR_ message. Because the requester's bus number hits Alt Bus Decoder, it is treated differently than a normal ERR_ message.

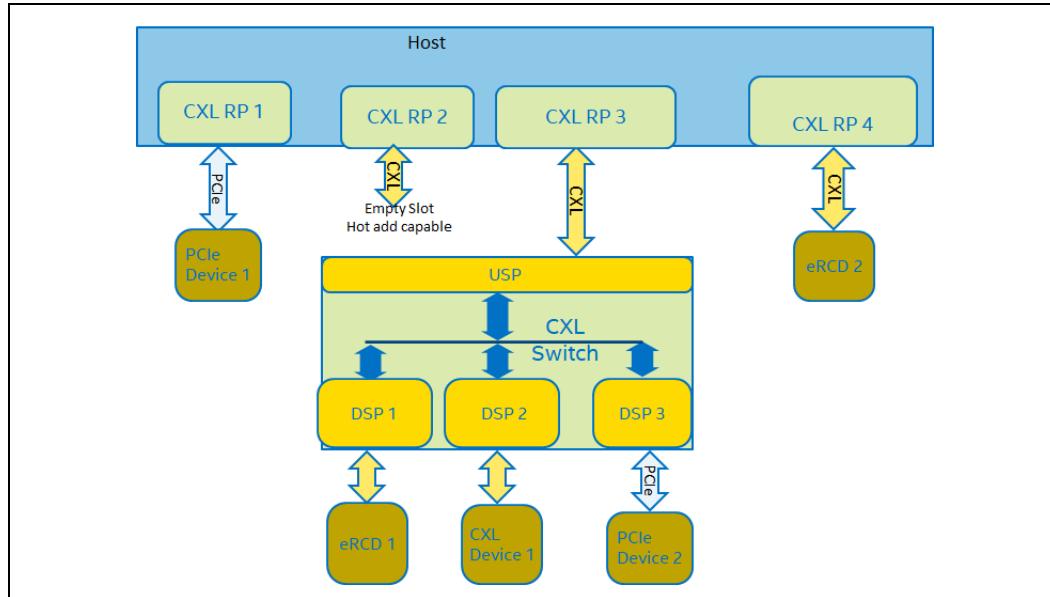
Figure 9-11. eRCD Configuration Space Decode - Example



9.12.5 CXL eRCD below a CXL RP and DSP - Example

Figure 9-12 represents the physical connectivity of a host with four Root Ports, one Switch, and 5 devices. The corresponding software view is shown in Figure 9-13. Note that the numbers (e.g., the “1” in PCIe Device 1) in this diagram do not represent the device number or the function number.

Figure 9-12. Physical Topology - Example



As shown in Figure 9-12, the Switch makes eRCD 1, below its DSP (DSP 1), appear as an RCIEP under an RCH. eRCD 1 is exposed as a separate Host Bridge to the Operating System. The device hosts a CXL DVSEC ID 0 instance in Device 0, Function 0 Configuration Space. The RCH Downstream Port registers and the RCD Upstream Port registers appear in MMIO space as expected.

When a CXL Root Port detects a PCIe device (PCIe Device 1), the Root Port trains up in PCIe mode. The Root Port configuration space (Type 1) may include the CXL DVSEC ID 3 and the CXL DVSEC ID 7. If present, the DVSEC ID 7 instance will indicate that the link trained up in PCIe mode. Other CXL DVSEC ID structures may be present as well.

If a CXL Root Port (RP 2) is connected to an empty slot, its configuration space (Type 1) hosts the CXL DVSEC ID 3 and the CXL DVSEC ID 7, but the DVSEC ID 7 shall indicate no CXL connectivity and the PCIe Link status register shall indicate that there is no PCIe connectivity. Other CXL DVSEC ID structures may be present as well. The user can hot-add a CXL device other than eRCD, a CXL Switch, or a PCIe device in this slot.

A CXL Root Port (RP 3) connected to a CXL Switch spawns a CXL VH. The Root Port as well as the Upstream Switch Port configuration space (Type 1) each host an instance of CXL DVSEC ID 3 and an instance of CXL DVSEC ID 7, but the DVSEC ID 7 instance will indicate that these Ports are operating in CXL VH operation with 68B Flit mode. Other CXL DVSEC ID structures may be present as well.

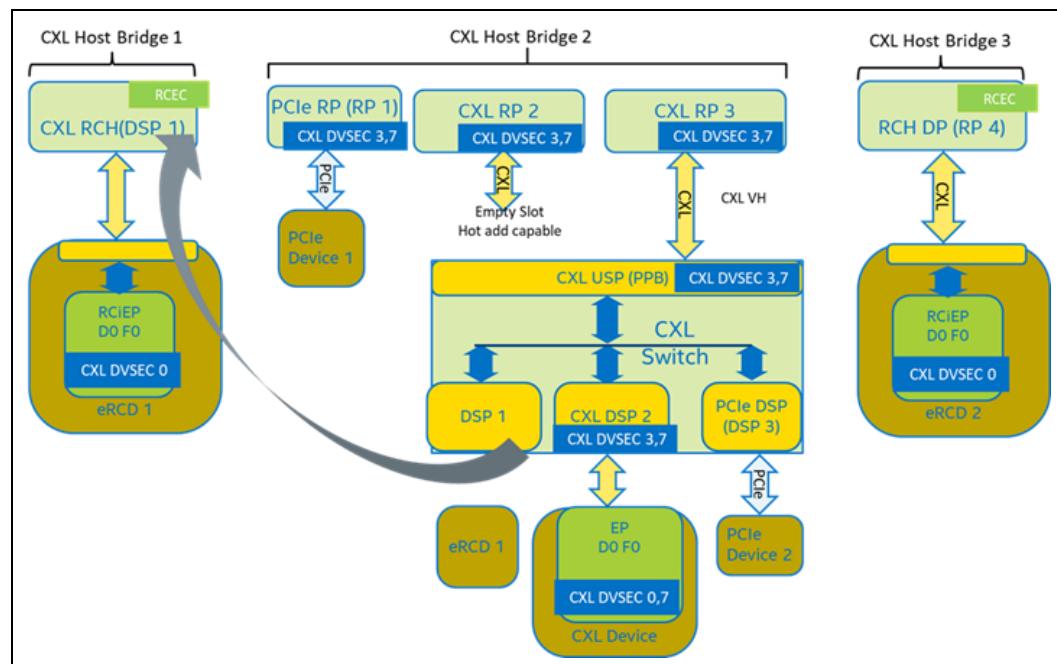
If a CXL Downstream Switch Port (DSP 2) is connected to a CXL device that is not an eRCD, DSP 2's configuration space (Type 1) hosts an instance of CXL DVSEC ID 3 and an instance of CXL DVSEC ID 7, but the DVSEC ID 7 instance will indicate that this Port is connected to a CXL device and is part of a CXL VH. Other CXL DVSEC ID structures may be present as well.

In this example, CXL Downstream Switch Port (DSP 3) is connected to a PCIe device and its configuration space (Type 1) does not host an instance of CXL DVSEC ID 7. Absence of a CXL DVSEC ID 7 indicates that this Port is not operating in the CXL mode. Note that it is legal for DSP 3 to host a DVSEC ID 7 instance as long as the DVSEC Flex Bus Port Status Register in the DVSEC ID 7 structure reports that the link is not operating in CXL mode.

If a CXL Root Port (RP 4) is connected to an eRCD, the Root Port operates as an RCH Downstream Port. eRCD 2 appears as an RCiEP under its own Host Bridge. This device hosts an instance of the CXL DVSEC ID 0 in Device 0, Function 0 Configuration Space. The RCH Downstream Port registers and the RCD Upstream Port registers appear in MMIO space as expected.

If the Switch is Hot-Pluggable, System Firmware may instantiate a _DEP object in the ACPI namespace to indicate that Device 1 is dependent on the CXL USP. A legacy PCIe bus driver interprets that to mean that the Switch hot removal has a dependency on eRCD 1, even though the ACPI/PCIe hierarchy does not show such a dependency.

Figure 9-13. Software View



9.12.6

Mapping of Link and Protocol Registers in CXL VH

In the presence of an eRCD, the link and protocol registers appear in MMIO space (RCRB and Component registers in the Downstream Port and the Upstream Port). See [Figure 9-7](#) and [Figure 9-8](#).

Because a CXL Virtual Hierarchy appears as a true PCIe hierarchy, the Component Register block is mapped using a standard BAR of CXL components.

Each CXL Host Bridge that is not an RCH includes CHBCR, which includes the registers that are common to all Root Ports under that Host Bridge. In an ACPI-compliant system, the base address of this register block is discovered via ACPI CEDT or the _CBR method. The CHBCR includes the HDM Decoder registers.

Each CXL Root Port carries a single BAR that maps the associated Component Register block. The offset within that BAR is discovered via the CXL DVSEC ID 8. See [Section 8.1.9](#). The layout of the Component Register Block is shown in [Section 8.2.3](#).

Each CXL device that is not an RCD can map its Component Register Block to any of its 6 BARs and a 64-KB-aligned offset within that BAR. The BAR number and the offset are discovered via CXL DVSEC ID 8. A Type 3 device Component Register Block includes HDM Decoder registers.

Each CXL USP carries a single BAR that maps the associated Component Register block. The offset within that BAR is discovered via CXL DVSEC ID 8. The Upstream Switch Port Component Register Block contains the registers that are not associated with a particular Downstream Port, such as HDM Decoder registers.

Each CXL DSP carries a single BAR that points to the associated CHBCR, the format of which closely mirrors that of a Root Port. The offset within that BAR is discovered via CXL DVSEC ID 8.

Figure 9-14. CXL Link/Protocol Register Mapping in a CXL VH

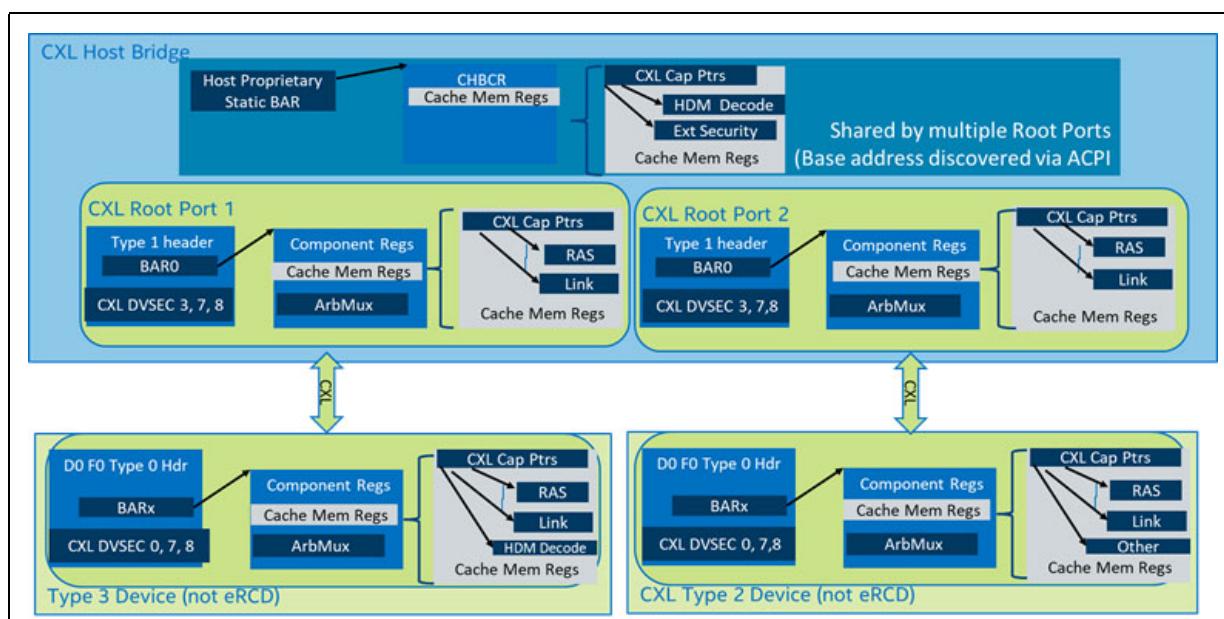
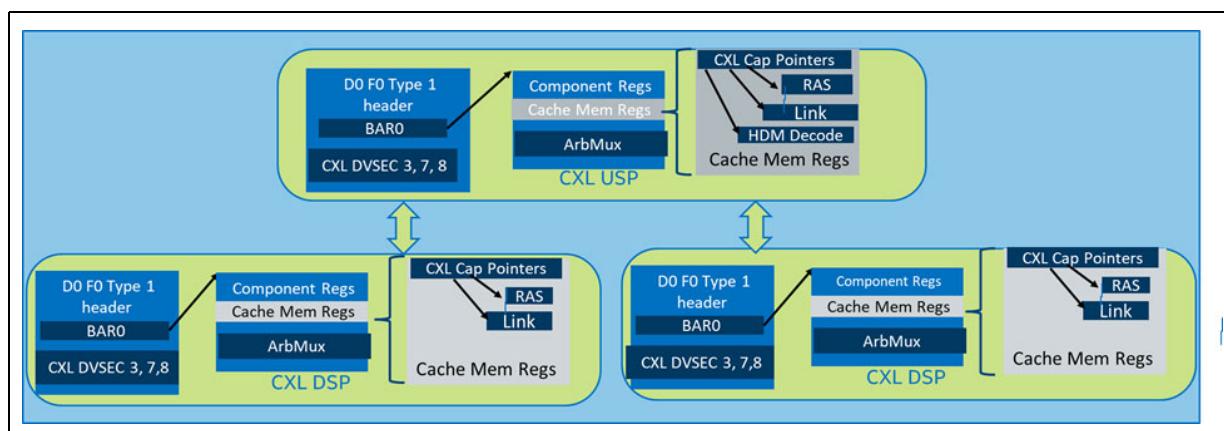


Figure 9-15. CXL Link/Protocol Registers in a CXL Switch



9.13

Software View of HDM

HDM is exposed to the OS/VMM as normal memory. However, HDM likely has different performance/latency attributes compared to host-attached memory. Therefore, a system with CXL.mem devices can be considered as a heterogeneous memory system.

ACPI HMAT was introduced for such systems and can report memory latency and bandwidth characteristics associated with different memory ranges. ACPI Specification version 6.2 and later carry the definition of revision 1 of HMAT. As of August 2018, ACPI WG has decided to deprecate revision 1 of HMAT because it had a number of shortcomings. As a result, the subsequent discussion refers to revision 2 of HMAT. In addition, ACPI has introduced a new type of Affinity structure called Generic Affinity (GI) Structure. GI structure is useful for describing execution engines such as accelerators that are not processors. CXL.mem-capable accelerators will result in two SRAT entries - One GI entry to represent the accelerator cores and one memory entry to represent the attached HDM. GI entry is especially useful when describing the CXL.cache accelerator. Previous to the introduction of GI, the CXL.cache accelerator could not be described as a separate entity in SRAT/HMAT and had to be combined with the attached CPU. With this specification change, the CXL.cache accelerator can be described as a separate proximity domain. _PXM method can be used to identify the proximity domain associated with the PCI device. Since Legacy OSs do not understand GI, System Firmware is required to return the processor domain that is most closely associated with the I/O device when running such an OS. ASL code can use bit 17 of Platform-Wide _OSC Capabilities DWORD 2 to detect whether the OS supports GI.

System Firmware must construct and report SRAT and HMAT to the OS in systems with CXL.cache devices and CXL.mem devices. Since System Firmware is not aware of HDM properties, that information must come from the CXL device in the form of CDAT. A device may export CDAT via Table Access DOE or via a UEFI driver.

System Firmware combines the information that it has about the host and CXL connectivity with CDAT content obtained from various CXL components during construction of SRAT and HMAT.

9.13.1

Memory Interleaving

Memory interleaving allows consecutive memory addresses to be mapped to different CXL devices at a uniform interval. eRCDs may support a limited form of interleaving as described in [Section 9.11.7.1](#), whereby memory is interleaved across the two links between a CPU and a dual-headed device.

The CXL 2.0 specification introduced a mechanism for interleaving across different devices. The set of devices that are interleaved together is known as the Interleave Set.

An Interleave Set is identified by the following:

- Base HPA - Multiple of 256 MB
- Size - Also a Multiple of 256 MB
- Interleave Way
- Interleave Granularity
- Targets (applicable only to Root Ports and Upstream Switch Ports)

These terms are described below.

Interleave Way: A CXL Interleave Set may contain either 1, 2, 3, 4, 6, 8, 12, or 16 CXL devices. 1-way Interleave is equivalent to no interleaving. The number of devices in an Interleave set is known as Interleave Ways (IW).

Interleave Granularity: Each device in an Interleave Set decodes a specific number of consecutive bytes, called Chunk, in HPA Space. The size of Chunk is known as Interleave Granularity (IG). The starting address of each Chunk is a multiple of IG.

- CXL Host Bridges (except RCH) and CXL switches must support the following IG values:
 - 256 Bytes (Interleaving on HPA[8])
 - 512 Bytes (Interleaving on HPA[9])
 - 1024 Bytes (Interleaving on HPA[10])
 - 2048 Bytes (Interleaving on HPA[11])
 - 4096 Bytes (Interleaving on HPA[12])
 - 8192 Bytes (Interleaving on HPA[13])
 - 16384 Bytes (Interleaving on HPA[14])
- CXL memory devices must support at least one of the two IG groups as reported via the CXL HDM Decoder Capability register (see [Section 8.2.4.20.1](#)):
 - Group 1: Interleaving on HPA[8], HPA[9], HPA[10], and HPA[11]
 - Group 2: Interleaving on HPA[12], HPA[13], and HPA[14]

Target: The HDM Decoders in the CXL Host Bridge are responsible for looking up the incoming HPA in a CXL.mem transaction and forwarding the HPA to the appropriate Root Port Target. The HDM Decoders in the CXL Upstream Switch Port are responsible for looking up the incoming HPA in a CXL.mem transaction and forwarding the HPA to the appropriate Downstream Switch Port Target.

An HDM Decoder in a device is responsible for converting HPA into DPA by stripping off specific address bits. These flows are described in [Section 8.2.4.20.13](#).

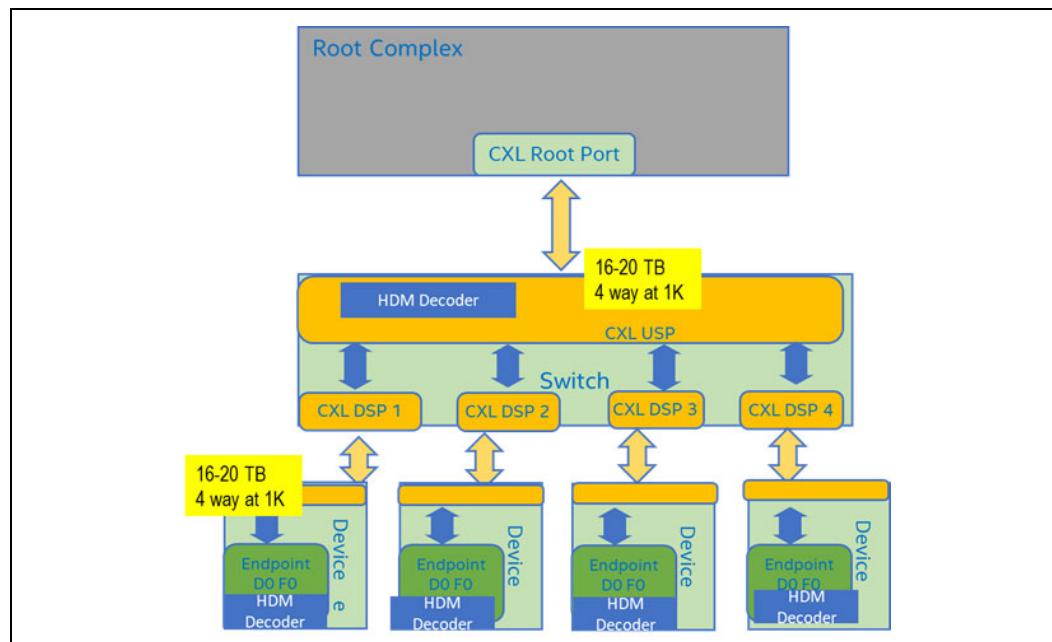
An Interleave Set is established by programming an HDM Decoder and committing it (see [Section 8.2.4.20.12](#)). The number of decoders implemented by a component are enumerated via the CXL HDM Decoder Capability register (see [Section 8.2.4.20.1](#)). HDM Decoders within a component must be configured in a congruent manner and the Decoder Commit flow performs certain self-consistency checks to assist with correct programming.

Software is responsible for ensuring that HDM Decoders located inside the components along the path of a transaction must be configured in a consistent manner.

[Figure 9-16](#) illustrates a simple memory fan-out topology with 4 memory devices behind a CXL Switch. A single HDM Decoder in each Device as well as the Upstream Switch Port is configured to decode the HPA range 16 to 20 TB, at 1-KB granularity. The leftmost Device receives 1-KB ranges starting with HPAs 16 TB, 16 TB + 4 KB, 16 TB+8KB, ..., 20 TB - 4 KB (every 4th Chunk). The Root Complex does not participate in the interleaving process.

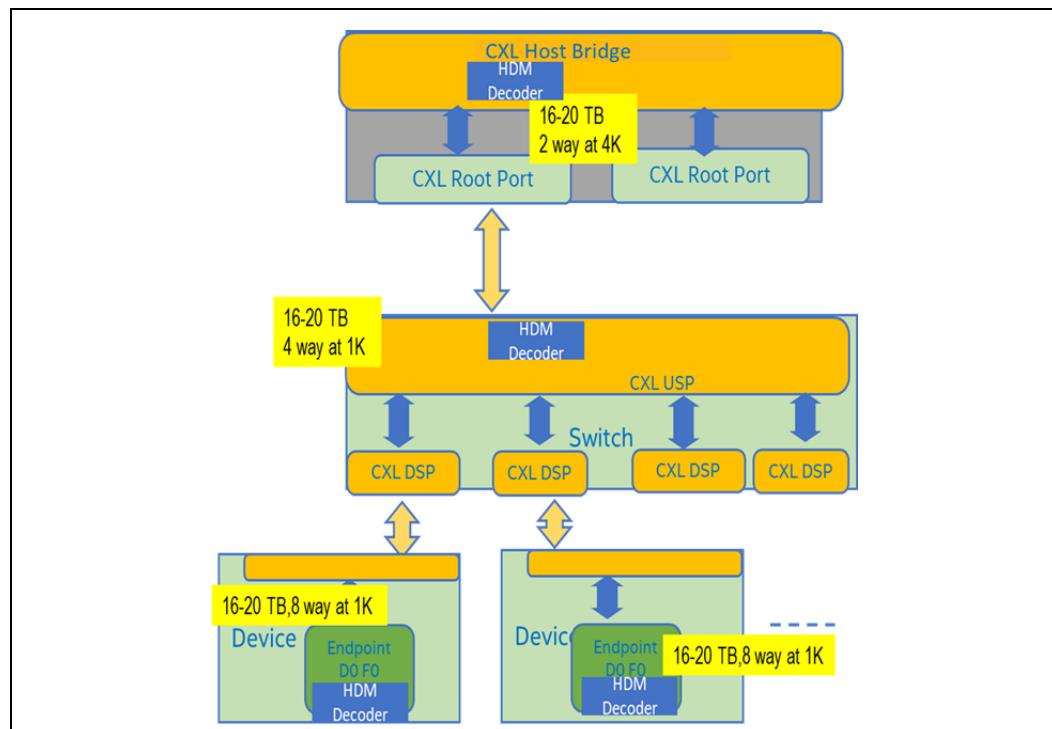
EVALUATION COPY

Figure 9-16. One-level Interleaving at Switch - Example



Multiple-level interleaving is supported as long as all the levels use different, but consecutive, HPA bits to select the target and no Interleave Set has more than 8 devices. This is illustrated in [Figure 9-17](#) and [Figure 9-18](#).

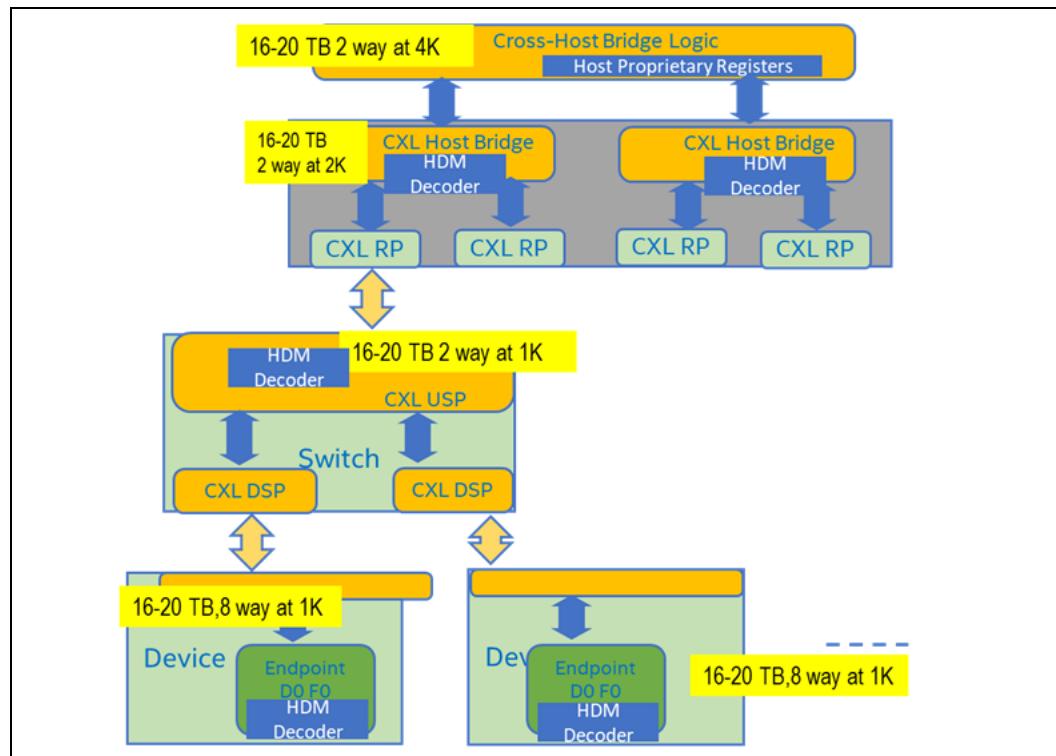
Figure 9-17. Two-level Interleaving



[Figure 9-17](#) illustrates a two-level Interleave scheme where the Host Bridge as well as the switch participates in the interleaving process. This topology has 4 memory devices behind each CXL Switch. One HDM Decoder in each of the 8 devices, both Upstream Switch Ports and the Host Bridge are configured to decode the HPA range 16 to 20 TB. The Host Bridge partitions the address range in two halves at 4-KB granularity (based on HPA[12]), with each half directed to a Root Port. Each Upstream Switch Port further splits each half into 4 subranges at 1-KB granularity (based on HPA[11:10]). To each device, it appears as though the HPA range 16-20 TB is 8-way interleaved at 1-KB granularity based on HPA[12:10]. The leftmost Device receives 1-KB ranges starting with HPAs 16 TB, 16 TB+8KB, 16 TB+16KB, ..., 20 TB-8KB.

[Figure 9-18](#) illustrates a three-level Interleave scheme where the cross-host Bridge logic, the Host Bridge, and the switch participate in the interleaving process. The cross-host Bridge logic is configured to interleave the address range in two halves, using host proprietary registers at 4-KB granularity. One HDM Decoder in 8 devices, 4 Upstream Switch Ports, and 2 Host Bridges are configured to decode the HPA range 16 to 20 TB. The Host Bridge further subdivides the address range in two at 2-KB granularity (using HPA[11]). The Upstream Switch Port in every switch further splits HPA space into 2 subranges at 1-KB granularity (using HPA[10]). To each device, it appears as though the HPA range 16-20 TB is 8-way interleaved at 1-KB granularity based on HPA[12:10]. Similar to [Figure 9-17](#), the leftmost Device receives 1-KB ranges starting with HPAs 16 TB, 16 TB+8KB, 16 TB+16KB, ..., 20 TB-8KB.

Figure 9-18. Three-level Interleaving Example



9.13.1.1 Legal Interleaving Configurations: 12-way, 6-way, and 3-way

This section describes the legal 12-way, 6-way, and 3-way interleaving configurations. The term IGB represents the interleave granularity in number of bytes. The cross-host Bridge Interleaving logic selects the target Host Bridge according to the configurations specified in [Table 9-6](#), [Table 9-7](#), and [Table 9-8](#), respectively. The Root Complex and the switch select the target port as described in [Section 9.18.1](#).

Table 9-6. 12-Way Device-level Interleave at IGB

Cross-host Bridge Logic Interleaving	CXL Root Complex-level Interleaving	CXL Switch-level Interleaving
12 way at IGB	No interleaving	No interleaving/Absent
6 way at 2*IGB	2 way at IGB	No interleaving/Absent
6 way at 2*IGB	No interleaving	2 way at IGB
3 way at 4*IGB	4 way at IGB	No interleaving
3 way at 4*IGB	No interleaving	4 way at IGB
3 way at 4*IGB	2 way at IGB	2 way at 2*IGB
3 way at 4*IGB	2 way at 2*IGB	2 way at IGB

Table 9-7. 6-Way Device-level Interleave at IGB

Cross-host Bridge Logic Interleaving	CXL Host Bridge-level Interleaving	CXL Switch-level Interleaving
6 way at IGB	No interleaving	No interleaving/Absent
3 way at 2*IGB	2 way at IGB	No interleaving
3 way at 2*IGB	No interleaving	2 way at IGB

Table 9-8. 3-Way Device-level Interleave at IGB

Cross-host Bridge Logic Interleaving	CXL Host Bridge-level Interleaving	CXL Switch-level Interleaving
3 way at IGB	No interleaving	No interleaving/Absent

9.13.2 CXL Memory Device Label Storage Area

CXL memory devices that provide volatile memory, such as DRAM, may be exposed with different interleave geometries each time the system is booted. This can happen due to the addition or removal of other devices or changes to the platform's default interleave policies. For volatile memory, these changes to the interleave usually do not impact host software since there's generally no expectation that volatile memory contents are preserved across reboots. However, with persistent memory, the exact preservation of the interleave geometry is critical so that the persistent memory contents are presented to host software the same way each time the system is booted.

Similar to the interleaving configuration, persistent memory devices may be partitioned into *namespaces*, which define volumes of persistent memory. These namespaces must also be reassembled the same way each time the system is booted to prevent data loss.

[Section 8.2.9](#) defines mailbox operations for reading and writing the *Label Storage Area* (LSA) on CXL memory devices: Get LSA and Set LSA. In addition, the Identify Memory Device mailbox command exposes the size of the LSA for a given CXL memory device. The LSA allows both interleave and namespace configuration details to be stored persistently on all the devices involved, so that the configuration data "follows the

device” if the device is moved to a different socket or machine. The use of an LSA is analogous to how disk RAID arrays write configuration information to a reserved area of each disk in the array so that the geometry is preserved across configuration changes.

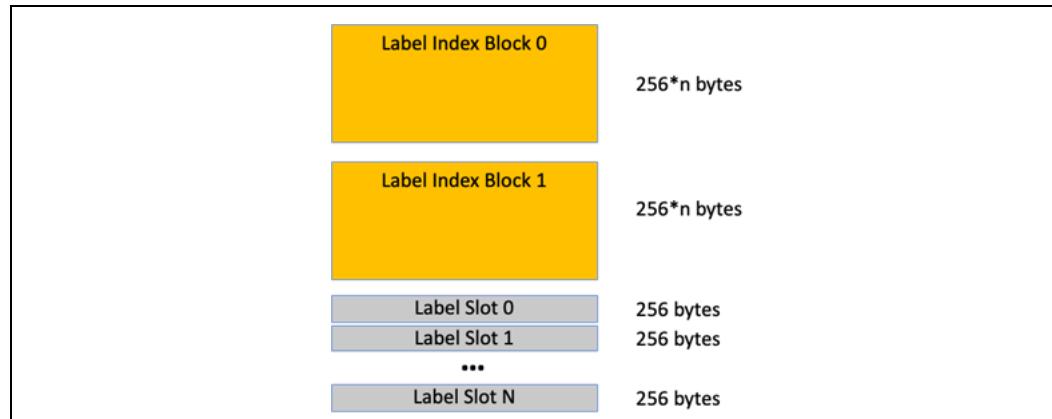
A CXL memory device may contribute to multiple persistent memory interleave sets, limited by interleave resources such as HDM decoders or other platform resources. Each persistent memory Interleave Set may be partitioned into multiple namespaces, limited by resources such as label storage space and supported platform configurations.

The LSA format and the rules for updating and interpreting the LSA are specified in this section. CXL memory devices do not directly interpret the LSA, they just provide the storage area and mailbox commands for accessing it. Software configuring Interleave Sets and namespaces, such as pre-boot firmware or host operating systems shall follow the LSA rules specified here to correctly inter-operate with CXL-compliant memory devices.

9.13.2.1 Overall LSA Layout

The LSA consists of two Label Index Blocks followed by an array of label slots. As shown in [Figure 9-19](#), the Label Index Blocks are always a multiple of 256 bytes in size, and each label slot is exactly 256 bytes in size.

Figure 9-19. Overall LSA Layout



The LSA size is implementation dependent and software must discover the size using the Identify Memory Device mailbox command. The minimum allowed size is two index blocks, 256-bytes each in length, two label slots (providing space for a minimal one region label and one namespace label), and one free slot to allow for updates. This makes the total minimum LSA size 1280 bytes. It is recommended (but not required) that a device provides for configuration flexibility by implementing an LSA large enough for two region labels per device and one namespace label per 8 GB of persistent memory capacity available on the device.

All updates to the LSA shall follow the update rules laid out in this section, which guarantee that the LSA remains consistent in the face of interruptions such as power loss or software crashes. There are no atomicity requirements on the Set LSA mailbox operation – it simply updates the range of bytes provided by the caller. Atomicity and consistency of the LSA is achieved using checksums and the principle that only free slots (currently unused) are written to – in-use data structures are never written, avoiding the situation where an interrupted update to an in-use data structure makes it

inconsistent. Instead, all updates are made by writing to a free slot and then following the rules laid out in this section to atomically swap the in-use data structure with the newly written copy.

The LSA layout uses *Fletcher64* checksums. [Figure 9-20](#) shows a Fletcher64 checksum implementation that produces the correct result for the data structures in this specification when run on a 64-bit system. When performing a checksum on a structure, any multi-byte integer fields shall be in little-endian byte order. If the structure contains its own checksum, as is commonly the case, that field shall contain 0 when this checksum routine is called.

Figure 9-20. Fletcher64 Checksum Algorithm in C

```
/*
 * checksum -- compute a Fletcher64 checksum
 */
uint64_t
checksum(void *addr, size_t len)
{
    uint32_t *p32 = addr;
    uint32_t *p32end = addr + len;
    uint32_t lo32 = 0;
    uint32_t hi32 = 0;

    while (p32 < p32end) {
        lo32 += *p32++;
        hi32 += lo32;
    }

    return (uint64_t)hi32 << 32 | lo32;
}
```

The algorithm for updating the LSA is single-threaded. Software is responsible for protecting a device's LSA so that only a single thread is updating the LSA at any time. This is typically done with a common mutex lock.

9.13.2.2

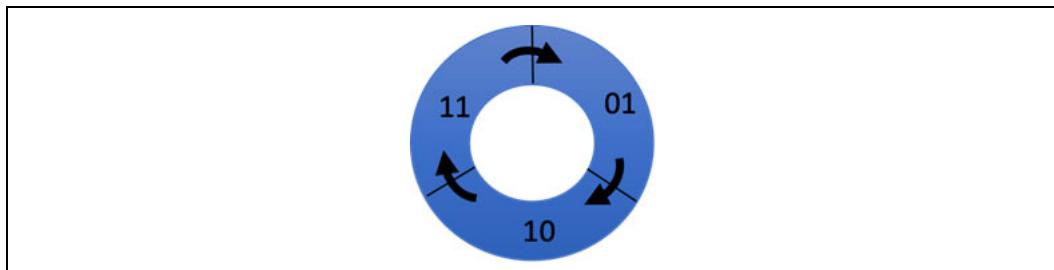
Label Index Blocks

[Table 9-9](#) shows the layout of a Label Index Block.

Table 9-9. Label Index Block Layout

Field	Byte Offset	Length in Bytes	Description
Sig	00h	10h	Signature indicating a Label Index Block. Shall be set to "NAMESPACE_INDEX\0".
Flags	10h	3	No flags defined yet, shall be 0.
LabelSize	13h	1	Shall be 1. This indicates the size of labels in this LSA in multiples of 256 bytes (e.g., 1 for 256, 2 for 512, etc.).
Seq	14h	4	Sequence number. Only the two least significant bits of this field are used and shown in Figure 9-21 . All other bits shall be 0.
MyOff	18h	8	Offset of this index block in the LSA. Label Index Block 0 shall have 0 in this field, Label Index Block 1 shall have the size of the index block as its offset.
MySize	20h	8	Size of an index block in bytes. Shall be a multiple of 256.
OtherOff	28h	8	Offset of the other index block paired with this one.
LabelOff	30h	8	Offset of the first slot where labels are stored.
NSlot	38h	4	Total number of label slots.
Major	3Ch	2	The major version number of this layout. Shall be 2.
Minor	3Eh	2	The minor version number of this layout. Shall be 1.
Checksum	40h	8	Fletcher64 checksum of all fields in this Label Index Block. This field is assumed to be 0 when the checksum is calculated.
Free	48h	Varies	NSlot bits, padded with 0s to align index block to 256 bytes.

When reading Label Index Blocks, software shall consider index blocks to be valid only when their Sig, MyOff, OtherOff, and Checksum fields are correct. In addition, any blocks with Seq cleared to 0 are discarded as invalid. Finally, if more than 1 Label Index Block is found to be valid, the one with the older sequence number (immediately counterclockwise to the other, according to [Figure 9-21](#)) is discarded. If all checks pass and the sequence numbers match, the index block at the higher offset shall be considered the valid block. If no valid Label Index Blocks are found, the entire LSA is considered uninitialized.

Figure 9-21. Sequence Numbers in Label Index Blocks

When updating the Label Index Block, the current valid block, according to the rules above, is never directly written to. Instead, the alternate block is updated with the appropriate fields and a sequence number that is immediately clockwise as shown in [Figure 9-21](#). It is the appearance of a new block that passes all the checks and has a higher sequence number that makes this update atomic in the face of interruption.

Using this method of atomic update, software can allocate and deallocate label slots, even multiple slots, in a single, atomic operation. This is done by setting the Free bits to indicate which slots are free and which are in-use, and then updating the Label Index Block atomically as described above. To ensure that it is always possible to update a label atomically, there must always be at least one free label slot. That way, any used

label slots can be updated by writing the new contents to the free slot and using the Label Index Block update algorithm to mark the new version and in-use and the old version and free in one atomic operation. For this reason, software must report a “label storage area full” error when a caller tries to use the last label slot.

The Free field contains an array of NSlot bits, indicating which label slots are currently free. The Label Index Block is then padded with 0 bits until the total size is a multiple of 256 bytes. This means that up to 1472 label slots are supported by Label Index Blocks that are 256 bytes in length. For 1473 to 3520 label slots, the Label Index Block size must be 512 bytes in length, and so on.

9.13.2.3

Common Label Properties

Three types of labels may occupy the label slots in the LSA: Region Labels, Namespace Labels, and Vendor Specific Labels. The first two are identified by type fields containing UUIDs as specified in the following sections. Vendor Specific Labels contain a type UUID determined by the vendor per IETF RFC 4122. Software shall ignore any labels with unknown types. In this way, the Type field in the labels provides a *major version number*, where software can assume that a UUID that it expects to find indicates a label that it understands, since only backward-compatible changes are allowed to the label layout from the point where that UUID first appears in a published CXL specification.

Region Labels and Namespace Labels contain a 4-byte Flags field, used to indicate the existence of new features. Since those features must be backward compatible, software may ignore unexpected flags encountered in this field (no error generated). Software should always write 0s for Flags bits that were not defined at the time of implementation. In this way, the Flags field provide a *minor version number* for the label.

It is sometimes necessary to update labels atomically across multiple CXL devices. For example, when a Region or Namespace is being defined, the labels are written to every device that contributes to it. Region Labels and Namespace Labels define a flag, UPDATING, that indicates a multi-device update is in-progress. Software shall follow this flow when creating or updates a set of labels across devices:

1. Write each label across all devices with the UPDATING flag set.
2. Update each label, using the update algorithm described in the previous section, clearing the UPDATING flag.

Any time software encounters a set of labels with any UPDATING flags, it shall execute these rules:

- If there are missing labels (some components with the expected UUID are missing), then the entire set of labels is rolled-back due to the update operation being interrupted before all labels are written. The roll-back means marking each label in the set as free, following the update algorithm described in the previous section.
- If there are no missing labels, then the entire set of labels is rolled-forward, completing the interrupted update operation by removing the UPDATING flag from all labels in the set, following the update algorithm described in the previous section.

When sets of Region Labels or Namespace Labels are found to have missing components, software shall consider them invalid and not attempt to configure the regions or surface the namespaces with these errors. Exactly how these errors are reported and how users recover from them is implementation-specific, but it is recommended that software first report the missing components, providing the opportunity to correct the misconfiguration, before deleting the erroneous regions or namespaces.

9.13.2.4 Region Labels

Region labels describe the geometry of a persistent memory Interleave Set (the term “region” is synonymous with “Interleave Set” in this section). Once software has configured a functional Interleave Set for a set of CXL memory devices, region labels are added to the LSA for each device that contributes capacity to it. [Table 9-10](#) shows the layout of a Region Label.

Table 9-10. Region Label Layout

Field	Byte Offset	Length in Bytes	Description
Type	00h	16	Shall contain this UUID: 529d7c61-da07-47c4-a93f-ecdf2c06f444. In the future, if a new, incompatible Region Label is defined, it shall be assigned a new UUID in the CXL specification defining it.
UUID	10h	16	UUID of this region per RFC 4122. This field is used to match up labels on separate devices that together describe a region.
Flags	20h	4	<p>Boolean attributes of the region:</p> <ul style="list-style-type: none"> • 0000 0008h = UPDATING <p>The UPDATING flag is used to coordinate Region Label updates across multiple CXL devices, as described in Section 9.13.2.3.</p> <p>All bits below 0000 0008h are reserved and shall be written as 0 and ignored when read.</p> <p>All bits above 0000 0008h are currently unused and shall be written as 0. The intention is to indicate the existence of backward-compatible features added in the future, so any unexpected 1s in this area shall be ignored (i.e., not treated as an error).</p>
NLabel	24h	2	Total number of devices in this Interleave Set (interleave ways).
Position	26h	2	Position of this device in the Interleave Set, starting with the first device in position 0 and counting up from there.
DPA	28h	8	The DPA where the region begins on this device.
RawSize	30h	8	The capacity this device contributes to the Interleave Set (bytes).
HPA	38h	8	If nonzero, this region needs to be mapped at this HPA. This field is for platforms that need to restore an Interleave Set to the same location in the system memory map each time. A platform that does not support this shall report an error when a nonzero HPA field is encountered.
Slot	40h	4	Slot index of this label in the LSA.
Interleave Granularity	44h	4	<p>The number of consecutive bytes that are assigned to this device:</p> <ul style="list-style-type: none"> • 0 = 256 Bytes • 1 = 512 Bytes • 2 = 1024 Bytes (1 KB) • 3 = 2048 Bytes (2 KB) • 4 = 4096 Bytes (4 KB) • 5 = 8192 Bytes (8 KB) • 6 = 16384 Bytes (16 KB) • All other encodings are reserved
Alignment	48h	4	<p>The desired region alignment in multiples of 256 MB:</p> <ul style="list-style-type: none"> • 0 = No desired alignment • 1 = 256-MB desired alignment • 2 = 512-MB desired alignment • etc.
Reserved	4Ch	ACh	Shall be 0.
Checksum	F8h	8	Fletcher64 checksum of all fields in this Region Label. This field is assumed to be 0 when the checksum is calculated.

9.13.2.5 Namespace Labels

Namespace labels describe partitions of persistent memory that are exposed as volumes to software, analogous to NVMe* namespaces or SCSI logical unit numbers (LUNs). Exactly how an operating system uses these volumes is beyond the scope of this specification – namespaces may be exposed to applications directly, exposed via file systems, or used internally by the operating system. [Table 9-11](#) shows the layout of a Namespace Label.

Table 9-11. Namespace Label Layout

Field	Byte Offset	Length in Bytes	Description
Type	00h	10h	Shall contain this UUID: 68bb2c0a-5a77-4937-9f85-3caf41a0f93c. In the future, if a new, incompatible Namespace Label is defined, it shall be assigned a new UUID in the CXL specification defining it.
UUID	10h	10h	UUID of this namespace per IETF RFC 4122. All labels for this namespace shall contain matching UUIDs.
Name	20h	40h	“Friendly name” for the namespace, null-terminated UTF-8 characters. This field may be cleared to all 0s if no name is desired.
Flags	60h	4	<p>Boolean attributes of the region:</p> <ul style="list-style-type: none"> • 0000 0008h = UPDATING <p>The UPDATING flag is used to coordinate Namespace Label updates across multiple CXL devices, as described in Section 9.13.2.3.</p> <p>All bits below 0000 0008h are reserved and shall be written as 0 and ignored when read.</p> <p>All bits above 0000 0008h are currently unused and shall be written as 0. The intention is to indicate the existence of backward-compatible features added in the future, so any unexpected 1s in this area shall be ignored (i.e., not treated as an error).</p>
NRange	64h	2	Number of discontiguous ranges that this device contributes to namespace, used when the capacity contributed by this device is not contiguous. Each contiguous range will be described by a label and NRange described how many labels were required.
Position	66h	2	Position of this device in the range set, starting with zero for the first label and counting up from there.
DPA	68h	8	The DPA where the namespace begins on this device.
RawSize	70h	8	The capacity this range contributes to the namespace (bytes).
Slot	78h	4	Slot index of this label in the LSA.
Alignment	7Ch	4	<p>The desired region alignment in multiples of 256 MB:</p> <ul style="list-style-type: none"> • 0 = No desired alignment • 1 = 256-MB desired alignment • 2 = 512-MB desired alignment • etc.
RegionUUID	80h	10h	UUID of the region that contains this namespace. If a valid region does not exist with this UUID, then this namespace is also considered unusable.
AddressAbstractionUUID	90h	10h	If nonzero, the address abstraction used by this namespace. Software defines the UUIDs used in this field and their meaning in software-specific and beyond the scope of this specification.
LBASize	A0h	2	If nonzero, logical block size of this namespace.
Reserved	A2h	56h	Shall be 0.
Checksum	F8h	8	Fletcher64 checksum of all fields in this Namespace Label. This field is assumed to be 0 when the checksum is calculated.

9.13.2.6 Vendor-specific Labels

Table 9-12 shows the layout of a Vendor-specific Label. Other than the Type field and the Checksum field, the vendor is free to store anything in the remaining 232 (E8h) bytes of the label.

Table 9-12. Vendor Specific Label Layout

Field	Byte Offset	Length in Bytes	Description
Type	00h	10h	Vendor-specific UUID.
	10h	E8h	Vendor-specific content.
Checksum	F8h	8	Fletcher64 checksum of all fields in this Vendor-specific Label. This field is assumed to be 0 when the checksum is calculated.

9.13.3 Dynamic Capacity Device (DCD)

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 8.2.9.9.9.3](#)). Similarly, the Extent List does contain extents that are still pending release acceptance from the host via the Release Dynamic Capacity command (see [Section 8.2.9.9.9.4](#)). [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.

EVALUATION COPY

Figure 9-22. Extent List Example (No Sharing)

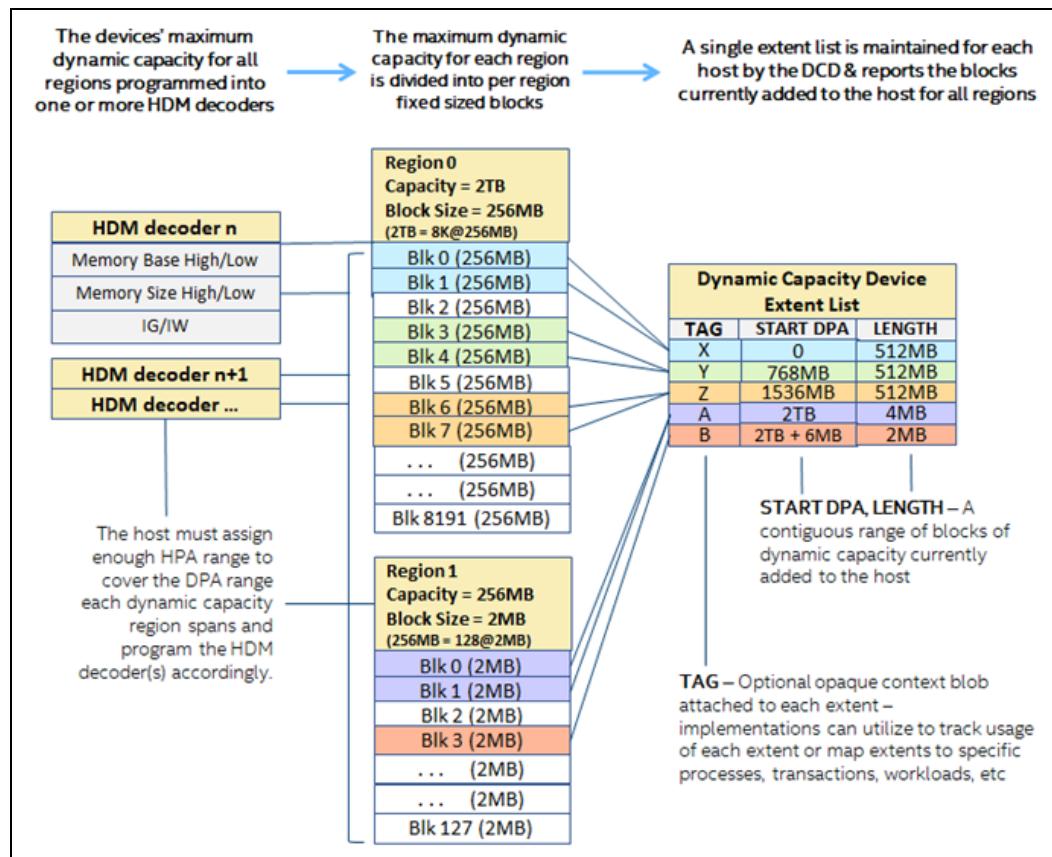
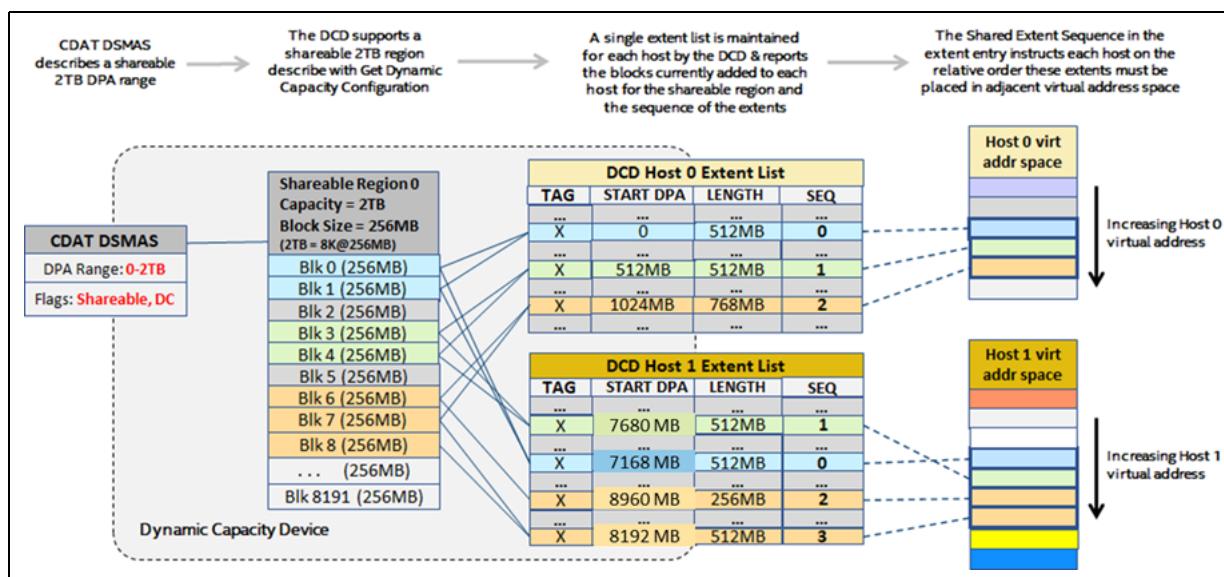


Figure 9-23. Shared Extent List Example

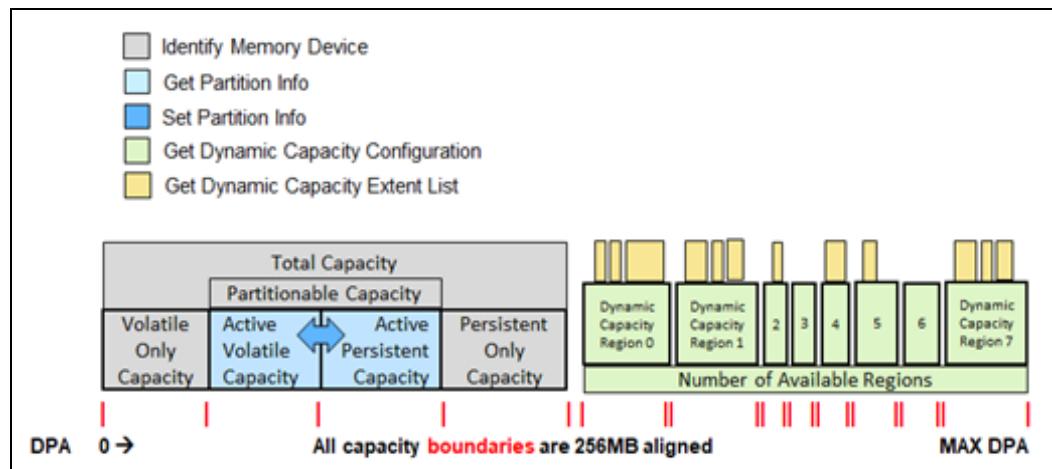


Dynamic Capacity is organized into 1 to 8 DC Regions as defined by the device. [Figure 9-24](#) illustrates this. Each DC Region has a unique maximum potential capacity, supported block size, and memory attributes. While the beginning and end of each region is 256 MB aligned, the start of the first block of data within each region is controlled by the DCD and aligned to the Dynamic Capacity block size configured for that region. Because the Extent List is DPA based, a single list can describe the extents in all regions. When the host fetches the current Extent List using the Get Dynamic Capacity Extent List mailbox command, the returned Extent List contains the device-assigned starting DPA and length for each extent that is assigned to the host. Regions are used in increasing-DPA order, with Region 0 being used for the lowest DPA of Dynamic Capacity and Region 7 for the highest DPA.

The DCD controls which DPA range it assigns to each region for each host. The DPA ranges exposed by the device to each host are independent of one another.

If the host issues a read to a DPA that is not allocated to the host, the device behavior is specified in [Table 8-27](#). If the host issues a write to a DPA that is not allocated to the host, the device shall drop the write and send an NDR (see [Section 3.3.9](#)) as a response. If the host issues a write to any DPA in a read-only DC Region, the device shall drop the write and send an NDR (see [Section 3.3.9](#)) as a response.

Figure 9-24. DCD DPA Space Example



The attributes associated with each region are described in the device's CDAT. The device associates each supported region with a specific DSMAS instance so the host can determine the memory attributes associated with each given region. A device that supports Dynamic Capacity shall report its configured regions in one or more CDAT DSMAS structures and shall set the Dynamic Capacity DSMAS Flag in each structure to indicate a Dynamic Capacity supported range. When reporting the region configuration, the DCD shall supply the DSMAD Handle with which each region is associated.

Devices that instantiate multiple LDs, including MLDs and Multi-Headed devices, share certain region configuration parameters, as defined in [Table 7-67](#), across all LDs in that device.

The basic sequence to utilize Dynamic Capacity include:

- Utilize Get Supported Logs sub-list (see [Section 8.2.9.5.6](#)) or Get Supported Logs (see [Section 8.2.9.5.1](#)) and Get Log (see [Section 8.2.9.5.2](#)) to retrieve the Command Effects Log (CEL). Verify that the necessary Dynamic Capacity commands are returned in the CEL, indicating Dynamic Capacity is supported by the device.

- Issue Get Dynamic Capacity Configuration command: The device reports its number of available regions and each region's base address, length, block size, and DSMAD Handle (see [Section 8.2.9.9.9.1](#)).
- Program the HDM decoders appropriately for each region's base and length from Get Dynamic Capacity Configuration data. The host may utilize one or more HDM decoders to span the current configuration of Dynamic Capacity reported by the device. It is strongly recommended that the host provide adequate decoder size to cover all of the regions that are enabled. If not, the host may not be able to accept some of the Add Dynamic Capacity offers from the DCD.
- Retrieve the initial Extent List with one or more calls to Get Dynamic Capacity Extent List (see [Section 8.2.9.9.9.2](#)). If the list contains extents, then that memory can be utilized immediately.

The basic sequence to add Dynamic Capacity to a host:

- The DCD adds an Add Capacity Event Record (see [Section 8.2.9.2.1.6](#)) 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 8-50](#)), 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.
- When the host software retrieves the Add Capacity event record containing the extent of the capacity to be added, it responds back to the device with the updated extent for the exact capacity it added with a single call to Add Dynamic Capacity Response (see [Section 8.2.9.9.3](#)). This allows the host to control exactly how much of the added capacity it wishes to utilize, which may be less than the amount of capacity sent in the add capacity event, or even 0.
- If supported by the device, the host may utilize Get Poison List or Scan Media with the Starting DPA and Length of the added capacity extent to check for poisoned addresses.

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 8.2.9.2.1.6](#)) 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 8-50](#)), 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.
- When the host software retrieves the Release Capacity event record containing the extent of the capacity to be released, the host software releases some or all of the capacity from use and responds back to the device with the updated Extent List for the exact capacity it released using the Release Dynamic Capacity command (see [Section 8.2.9.9.4](#)). If desired, the host may choose to make unavailable the contents of the capacity being released by whatever means it chooses, including but not limited to issuing the Sanitize or Secure Erase commands, if supported by the device, before the Release Dynamic Capacity command. The host may call Release Dynamic Capacity multiple times, returning different portions of the total capacity over time, in response to the Release Capacity event record. This allows the host to control exactly how much of the released capacity it wishes to release and when it is released.

Prior to issuing Release Dynamic Capacity command, the host software is required to off-line the capacity and complete the necessary coherence management actions.

The basic sequence to release Dynamic Capacity asynchronously from a host (not associated with an event from the device):

- The host may release Dynamic Capacity back to the device, at any time, without receiving a Release Capacity Event Record by calling Release Dynamic Capacity (see [Section 8.2.9.9.9.4](#)) with an Extent List containing specific released capacity.

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 occurs and the host shall utilize Get Dynamic Capacity Extent List to retrieve a new list of host accessible DC blocks.

LD-FAM based DCD shall forcefully release any shared Dynamic Capacity associated with an LD upon a Conventional Reset or a CXL Reset of that LD. MH-SLD or MH-MLD based DCD shall forcefully release shared Dynamic Capacity associated with all associated hosts upon a Conventional Reset of a head. LD-FAM based DCD shall forcefully release shared Dynamic Capacity associated with all associated hosts upon a Conventional Reset of the entire DCD. No Forced Capacity Release Event Record is created when capacity is released as a result of a reset.

The host retrieves the Release Capacity event record containing the extent of the capacity that has been released. The host may respond back to the device with the updated Extent List for the released capacity using the Release Dynamic Capacity command. The host may call Release Dynamic Capacity multiple times, returning different portions of the total capacity over time. Host responses to this event are optional and shall not influence the device's release of the capacity.

9.13.3.1 DCD Management By FM

LD-FAM DCDs implement multiple LDs to support multiple host interfaces and can dynamically assign and reassign memory capacity among those LDs. All G-FAM Devices (GFDs) are DCDs since GFDs exclusively use Dynamic Capacity mechanisms for their capacity management.

The FM is responsible for discovering a DCD's capabilities and for configuring memory assignment.

1. The FM issues Get DCD Info (see [Section 7.6.7.6.1](#)) to discover the number of supported hosts, supported features, and dynamic memory capacity. The current assignment of capacity to a specific host is queried with Get Host DC Region Configuration and Get DC Region Extent Lists (see [Section 7.6.7.6.2](#) and [Section 7.6.7.6.4](#), respectively). See [Section 8.2.9.9.10](#) for the equivalent GFD commands.
2. Resources are assigned to each host using Initiate Dynamic Capacity Add and Initiate Dynamic Capacity Release (see [Section 7.6.7.6.5](#) and [Section 7.6.7.6.6](#), respectively). The device generates a Dynamic Capacity Event Record (see [Section 8.2.9.9.9.4](#)) to notify the FM of any host responses. See [Section 7.7.2](#) and [Section 7.7.14](#) for the equivalent GFD commands and policies.

9.13.3.2 Setting up Memory Sharing

The FM may use the following sequence to set up sharing between hosts, where all hosts are able to read and write to the shared capacity:

1. Issue Initiate Dynamic Capacity Add Request with the Selection Policy set to Free or Contiguous or Prescriptive with the Host ID associated with the first host. The region number must correspond to a region that is advertised as sharable.
2. If the above request is successful as indicated by a new Add Capacity Response event in the Dynamic Capacity Event record, issue Initiate Dynamic Capacity Add Request with Selection Policy=Enable Shared access with the Host ID associated with the second host. The Tag field must match the Tag value used in step 1.
3. Repeat step 2 for any other hosts that need to share this memory range.

The FM may use the following example sequence to allocate a set of tagged capacity and allow it to be initialized by a host and then shared with one or more hosts as read-only.

1. Issue Initiate Dynamic Capacity Add Request with the Selection Policy set to Free or Contiguous or Prescriptive with the Host ID associated with the first host. The region number must correspond to a region that is advertised as writable and sharable.
2. If the above request is successful, the tagged shared capacity can be initialized by the first host.
3. Issue a Dynamic Capacity Add Reference Request for the tag associated with the capacity. Holding this Reference prevents the tagged capacity from being freed and sanitized in step 4.
4. After the first host has initialized the tagged shared capacity, issue an Initiate Dynamic Capacity Release Request for the tag associated with the capacity, and then await completion.
5. If the request in step 4 is successful as indicated by a new Release Capacity Response event in the Dynamic Capacity Event record, the capacity associated with the Tag is preserved but not mapped to any hosts.
6. Issue an Initiate Dynamic Capacity Add Request with Selection Policy=Enable Shared Access with the Host ID associated with the second host, specifying a Region that is Sharable and read-only. The Tag field must match the Tag value used in step 1.
7. Repeat step 5 for any other hosts that need to share the tagged capacity.
8. Issue a Dynamic Capacity Remove Reference Request to remove the FM reference to the tagged capacity.
9. To withdraw the shared capacity, issue a Initiate Dynamic Capacity Release command for each host.
10. When the tagged capacity has been released from all hosts, if the FM does not hold a reference, the tagged capacity will be sanitized (if appropriate) and freed, at which point the tag no longer exists and the capacity is available for future use.

9.13.3.3 Extent List Tracking

The storage of extent list information, including individual extents and their associated tags, consumes resources in a DCD. As such, DCDs are permitted to limit the number of extents and number of tags of which they are capable of tracking. This capability is reported in a DCD's **Get Host DC Region Configuration** and **Get Dynamic Capacity Configuration** responses.

A DCD is responsible for tracking all extents and tags that comprise extent lists in the following states:

- **Pending:** Defining capacity specified in an **Initiate Dynamic Capacity Add** request that has not been responded to by a host
- **Added:** Defining capacity that has been accepted by a host as part of an **Add Dynamic Capacity** request and is present in the extent list returned to the host in the response to a **Get Dynamic Capacity Extent List** request
- **FM-referenced:** Defining capacity to which an FM reference has been added, as reported by the **FM Holds Reference** bit in the response to **Dynamic Capacity List Tags**

A DCD reports its **Number of Available Extents** and **Number of Available Tags** as its total capacity minus all extents and tags tracked for capacity in the Pending, Added, and FM-referenced states, respectively.

9.13.4

Capacity or Performance Degradation

A CXL device may detect an unrecoverable error during its initialization and may be able to operate with a reduced capacity or reduced performance. If this failure results in capacity degradation and it is detected prior to `Memory_Info_Valid=1`, the device shall update the `Memory_Size` fields in the corresponding DVSEC CXL Range Size registers (see [Section 8.1.3.8.1](#), [Section 8.1.3.8.2](#), [Section 8.1.3.8.5](#), and [Section 8.1.3.8.6](#)), CDAT DSMAS structures, response to Identify Memory Device command, and response to Get Partition Info command to report the reduced size. It is recommended that the device also set the `Memory Capacity Degraded` flag in the Health Status field (see [Table 8-133](#)).

If the failure results in performance degradation and it is detected prior to `Memory_Info_Valid=1`, the CDAT DSLBIS structure shall be updated and the `Performance Degraded` flag in the Health Status field (see [Table 8-133](#)) should be set. If `Mem_HwInit_Mode=1`, `Memory_Active` bit(s) shall be set when the memory range is fully initialized and available for software use.

If this failure is detected after the `Memory_Info_Valid` bit is set, but before the `Memory_Active` bit is set, the device shall not set the `Memory_Active` bit. The device updates the CDAT in the following manner:

- CDAT sequence number shall be incremented to indicate to SW that CDAT content has changed.
- If the failure results in capacity degradation, the CDAT DSEMITS entries shall mark the bad memory as “`EFIUnusableMemory`” indicating to the SW that it shall not use the associated DPA range on this device. The `Memory Capacity Degraded` flag in the Health Status field (see [Table 8-133](#)) shall be set.
- If the failure results in performance degradation, the CDAT DSLBIS structure shall be updated and the `Performance Degraded` flag in the Health Status field (see [Table 8-133](#)) shall be set.

If `Mem_HwInit_Mode=1`, `Memory_Active_Degraded` shall be set when the reduced capacity is fully initialized and available for software use.

The device capacity reported by Identify Memory Device (see [Section 8.2.9.9.1.1](#)) and Get Partition Info (see [Section 8.2.9.9.2.1](#)) commands shall be consistent with capacity advertised by CDAT that is not marked as `EFIUnusableMemory`.

9.14

Back-Invalidate Configuration

This section describes how System Software may discover whether a component supports Back-Invalidate and how BI-IDs are assigned.

9.14.1 Discovery

Back-Invalidate (BI) messages require the link to operate in 256B Flit mode. Alternate Protocol Negotiation flow establishes the optimal Flit mode and PCIe DVSEC for Flex Bus Port registers (see [Section 8.2.1.3](#)) identifies the negotiated Flit mode. The presence of the CXL BI Decoder Capability Structure indicates that the component is capable of supporting BI.

9.14.2 Configuration

Before enabling a device to issue BI requests, System Software must ensure that the device, the host, and any switch(es) in the path are capable of BI and that the link(s) between the device and the host are operating in 256B Flit mode.

BI-capable Downstream Ports and devices advertise the CXL BI Decoder Capability Structure (see [Section 8.2.4.27](#)). System Software configures them to enable BI functionality. The BI-ID of a device must be unique within a VH. This is ensured by using the device's Bus Number as the BI-ID. The Downstream Port decode functionality is described in [Table 9-13](#) and [Table 9-14](#).

Table 9-13. Downstream Port Handling of BISnp

BI Enable Value	BI Forward Value	Behavior
0	0	Discard
0	1	Forward upstream as is
1	0	Perform the following checks: <ul style="list-style-type: none"> Locate the HDM decoder in the USP or RC that decodes the BISnp address. Verify that the BI bit in that HDM decoder is set. Optionally, verify that the Target Port that corresponds to the BISnp address matches the port that generated the BISnp request. If this is a DSP: <ul style="list-style-type: none"> If above checks pass, Set BI-ID= Secondary Bus Number and forward upstream; otherwise, discard. If this is a root port: <ul style="list-style-type: none"> If above checks pass, forward upstream; otherwise, discard. Root port may use host proprietary mechanisms to initialize BI-ID and route the associated BIRsp messages.
1	1	Discard (Invalid setting)

Table 9-14. Downstream Port Handling of BIRsp

BI Enable Value	BI Forward Value	Behavior
0	0	Discard
0	1	Forward downstream as is
1	0	If this is a DSP: <ul style="list-style-type: none"> If BI-ID=Secondary Bus Number, forward downstream; otherwise, discard. If this is a root port: <ul style="list-style-type: none"> Use host-specific checks to ensure correct routing of the BISnp response. Forward downstream if these checks pass; otherwise, discard.
1	1	Discard (Invalid setting)

The USP in a BI-capable Switch may advertise the CXL BI Route Table capability Structure (see [Section 8.2.4.26](#)). If a USP receives an M2S BIRsp message, the USP shall look up the Port Number associated with the Bus Number that is carried in the message's BI-ID field, and then forward the message to that Port. The BI-ID is guaranteed to correspond to a valid BI-capable device, specifically the one that generated the BISnp request. If the Port Number does not match any DSP due to incorrect programming, the BIRsp message shall be dropped.

If a USP receives an S2M BISnp message, the USP may look up the Port Number associated with the Bus Number that is carried in the message's BI-ID field, and then verify that the Port Number matches the Port Number of the originating DSP before forwarding the BISnp message upstream. If the Port Number derived from this structure does not match the DSP's Port Number, the BISnp message may be dropped.

IMPLEMENTATION NOTE

System software may use the following sequence to configure a BI-capable Device D below a Switch S as follows:

1. Verify that all the CXL link(s) between Device D and the host are operating in 256B Flit mode.
2. Ensure the device has been assigned a valid Bus number.
3. Enable BI on the DSP of Switch S that is directly connected to Device D:
 - a. BI Forward=0.
 - b. BI Enable=1.
4. If the DSP's BI Decoder Capability register indicates Explicit BI Decoder Commit Required=1, commit the BI-ID changes via the following sequence:
 - a. BI Decoder Commit=0 to rearm.
 - b. BI Decoder Commit=1.
 - c. Poll bits 0 and 1 of the BI Decoder Status register until timeout or one of them is set. The timeout value is reported in the BI Decoder Status register.
 - d. If BI Decoder Committed=1, the changes were committed. Proceed to step 5.
 - e. If BI Decoder Error Not Committed=1, the changes were not committed. Software should treat this as an error condition.
 - f. If neither bit is set and the timeout is reached, Software should treat this as an error condition.
5. If the USP implements CXL BI Route Table Capability Structure and Explicit BI RT Commit Required=1, commit the BI-ID changes as follows:
 - a. BI RT Decoder Commit=0 to rearm.
 - b. BI RT Decoder Commit=1.
 - c. Poll bits 0 and 1 of the BI RT Status register until timeout or one of them is set. The timeout value is reported in the BI RT Status register.
 - d. If BI RT Error Not Committed=1, the changes were not committed. Software should treat this as an error condition.
 - e. If BI RT Committed=1, the changes were committed. Proceed to step 6.
 - f. If neither bit is set and the timeout is reached, Software should treat this as an error condition.
6. If the previous steps were successful, configure the Root Port that is directly connected to Switch S to forward BI messages if it isn't already set up that way:
 - a. If BI Forward=0, set BI Forward=1.
 - b. Ensure BI Enable=0.
7. If the previous steps were successful, configure Device D to enable BI:
 - a. BI Enable=1.
8. If the previous steps were successful, inform the device driver that Device D may now issue BI requests.

IMPLEMENTATION NOTE

System software may use the following sequence to deallocate the BI-ID B that was previously assigned to Device D below Switch S as follows:

1. Notify Device D's device driver that Device D is no longer allowed to issue BI requests and then wait for acknowledgment.
2. Configure Device D to disable BI:
 - a. BI Enable=0.
3. Configure the DSP of Switch S that is directly connected to Device D to unassign BI-ID B as follows:
 - a. BI Forward=0.
 - b. BI Enable=0.
4. If the DSP's CXL BI Decoder Capability register indicates Explicit BI Decoder Commit Required=1, commit the BI-ID changes as follows:
 - a. BI Decoder Commit=0 to rearm.
 - b. BI Decoder Commit=1.
 - c. Poll bits 0 and 1 of the BI Decoder Status register until timeout or one of them is set. The timeout value is reported in the BI Decoder Status register.
 - d. If BI Decoder Error Not Committed=1, the changes were not committed. Software should treat this as an error condition.
 - e. If BI Decoder Committed=1, the changes were committed. Proceed to step 5.
 - f. If neither bit is set and the timeout is reached, Software should treat this as an error condition.
5. If the USP implements CXL BI Route Table Capability Structure and Explicit BI RT Commit Required=1, commit the BI-ID changes as follows:
 - a. BI RT Commit=0 to rearm.
 - b. BI RT Commit=1.
 - c. Poll bits 0 and 1 of the BI RT Status register until timeout or one of them is set. The timeout value is reported in the BI RT Status register
 - d. If BI RT Error Not Committed=1, the changes were not committed. Software should treat this as an error condition.
 - e. If BI RT Committed=1, the changes were committed. Proceed to step 6.
 - f. If neither bit is set and the timeout is reached, Software should treat this as an error condition.
6. If the previous steps were successful, and no other devices in this VCS have been assigned a BI-ID, configure the Root Port that is directly connected to Switch S to stop forwarding BI messages as follows:
 - a. BI Forward=0.

Ensure BI Enable=0.

9.14.3

Mixed Configurations

This section describes scenarios where a BI-capable device is plugged into a system that does not support BI.

9.14.3.1 BI-capable Type 2 Device

If a BI-capable Type 2 device is connected to a Downstream Port that does not support 256B Flit mode, the device is able to detect this condition during the Hardware Autonomous Mode Negotiation (see [Section 6.4.1.1](#)) and fall back to another mode (e.g., Type 2 HDM-D mode or PCIe mode) based on the device vendor's policy.

If a BI-capable Type 2 device is connected to a switch that supports BI, but the host does not support BI, the device cannot be operated in BI mode. In this case, the System Software or the System Firmware may choose to reconfigure the Type 2 device to operate in a fallback mode.

It is legal for BI-capable Type 2 devices to not support HDM-D flow; however, such a device must support fallback to either operate as a PCIe device, Type 1 device, or a Type 3 device. These flows are described in [Section 9.14.3.2](#).

If a Type 2 device advertises support for HDM-D flow via the BI Decoder Capability register (see [Section 8.2.4.27.1](#)), the device is operated in that mode as long as the number of Type 2 devices using HDM-D flow does not exceed the host's capabilities and the CXL specification restrictions. A CXL Type 2 device that supports HDM-D flow may be unable to operate in that mode due to system configuration restrictions. In many scenarios, the device may be unable to make that determination on its own and may require assistance from System Software or System Firmware. See [Section 9.14.3.2](#).

9.14.3.2 Type 2 Device Fallback Modes

[Table 9-15](#) describes the actions that System Software or System Firmware may take when a Type 2 device cannot be operated in either HDM-DB mode or in HDM-D mode, based on the Fallback Capability field value in the DVSEC CXL Capability2 register (see [Section 8.1.3.7](#)).

Table 9-15. CXL Type 2 Device Behavior in Fallback Operation Mode

Register Value ¹	Behavior
00b	The device can be operated as an RCD. If the device does not support HDM-DB flow, it supports HDM-D flow. If the device supports HDM-DB flow, it also supports HDM-D flow and must return HDM-D Capable=1 (see Section 8.2.4.27.1). If the device cannot be operated as a Type 2 device, it must be disabled.
01b	The device supports either HDM-DB flow or HDM-D flow or both. In addition, it can operate as a PCIe device. If the device cannot be operated in either HDM-DB mode or in HDM-D mode, System Firmware or System Software may disable Alternate Protocol Negotiation by programming the DSP registers and retraining the link so that the link comes up in PCIe mode.
10b	The device supports either HDM-DB flow or HDM-D flow or both. In addition, it can operate as a CXL Type 1 device. If the device cannot be operated in either HDM-DB mode or in HDM-D mode, System Firmware or System Software may reconfigure the DVSEC Flex Bus Port Control register (see Section 8.2.1.3.2) in the Downstream Port above the device to not advertise CXL.mem and then retrain the link, thereby bringing up the device as a CXL Type 1 device.
11b	The device supports either HDM-DB flow or HDM-D flow or both. In addition, it can operate as a CXL Type 3 device. If the device cannot be operated in either HDM-DB mode or in HDM-D mode, System Firmware or System Software may reconfigure the Flex Bus Port Control register (see Section 8.2.1.3.2) in the Downstream Port above the device to not advertise CXL.cache and then retrain the link, thereby bringing up the device as a CXL Type 3 device.

1. Fallback Capability field values in the DVSEC CXL Capability2 register (see [Section 8.1.3.7](#)).

More-complex policies, such as configuring the Device to operate in CXL.io only mode or another mode based on peer devices, are possible; however, those policies are beyond the scope of this specification.

9.14.3.3

BI-capable Type 3 Device

A BI-capable Type 3 device is required to operate correctly when System Software has not enabled BI. In this case, the device functionality that is dependent on BI will not be available.

If a BI-capable Type 3 device is connected to a Downstream Port that does not support 256B Flit mode, the device may continue to advertise BI capability via the CXL BI Decoder Capability Structure (see [Section 8.2.4.27](#)). The System Software shall ensure that the BI bit in none of the HDM decoders in the device, the switch, or the host that spans the device's HDM is set. If a BI-capable Type 3 device is present in a system where the host does not support BI, the System Software shall ensure that the BI bit in none of the HDM decoders in the device, the switch, or the host that spans the device's HDM is set. In both cases, the System Software is responsible for ensuring that the BI bit in the CXL BI Decoder Control register (see [Section 8.2.4.27.2](#)) in the device, as well as the Downstream Port it is connected to, is programmed to 0.

9.15

Cache ID Configuration and Routing

The CXL 3.0 specification introduces protocol enhancements that allow for more than one active CXL.cache agent per VCS. The identity of the CXL.cache agent is carried via the CacheID field in the CXL.cache messages. If the CXL link is operating in 256B Flit mode, the CXL.cache messages can carry 4 CacheID bits. Before enabling more than one CXL.cache device per VCS, Software must ensure that the host and any switch(es) in the path advertise the CXL Cache ID Decoder Capability Structure, and that all the link(s) between the lowest-level switch and the host are operating in 256B Flit mode.

Downstream Ports advertise the CXL Cache ID Decoder Capability structure to indicate that the Downstream Ports can assign and decode the CacheID field in CXL.cache messages (see [Section 8.2.4.29](#)). Software configures the Downstream Ports to enable CacheID forwarding functionality and assign a CacheID to the device. The CacheID must be unique within a VH and must account for the constraints placed by the Flit mode and the host capabilities.

Any CXL.cache device can operate correctly in a system that is capable of supporting more than one active CXL.cache agent per VCS; however, System Firmware or System Software that is aware of this new capability and capable of correctly configuring the switch and/or host is required to take advantage of this capability.

9.15.1

Host Capabilities

The host requires dedicated resources to track each CacheID source. As such, it is necessary to account for host constraints when assigning CacheID. The host constraints are expressed in terms of the total number of CacheIDs that the host can track per CXL Host Bridge. This information is conveyed via the Cache ID Target Count field in the CXL Cache ID Route Table Capability register (see [Section 8.2.4.28.1](#)) associated with the Host Bridge.

9.15.2

Downstream Port Decode Functionality

Downstream Port decode functionality is described in [Table 9-16](#) and [Table 9-17](#). The associated registers are defined in [Section 8.2.4.14](#).

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	Set CacheID=Local Cache ID and forward upstream. 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 addition to the checks documented in [Table 9-16](#), the root port shall implement the following steps before forwarding the message upstream:

- If HDM-D Type 2 Device Present=1, compare CacheID with the HDM-D Type 2 Device Cache ID field. If there is a match, identify this device as a Type 2 device that is using HDM-D flows. The host shall follow the HDM-D flows when responding to this device, which includes enforcing the setting in the CXL.cache Trust Level field of the Root Port Security Policy register (see [Table 8-29](#)).
- If the Requester is using HDM-DB flows, abort the request if Block CXL.cache HDM-DB=1.

Table 9-17. Downstream Port Handling of H2D Response Message and H2D Request Message

Assign Cache ID Value	Forward Cache ID Value	Behavior
0	0	Discard
0	1	Forward downstream as is
1	0	If CacheID=Local CacheID, forward downstream; otherwise, discard. The link between the device and the Downstream Port may be operating in 68B Flit mode, in which case the H2D message received by the device does not contain the CacheID field. The device shall ignore the CacheID field in H2D messages, if present.
1	1	Discard (Invalid setting)

D2H response messages and D2H data messages do not carry CacheID and are always routed back to the host.

9.15.3 Upstream Switch Port Routing Functionality

When a USP receives a D2H request message from a DSP, the USP shall forward the message upstream. A USP may look up the Port Number associated with the CacheID field in the message from the CXL Cache ID Route Table and may compare that to the Port Number of the DSP that the message came from before forwarding the message.

When a USP receives an H2D request message, H2D data message or an H2D response message, the USP shall use the message's CacheID field to look up the corresponding CXL Cache ID Target N register (see [Section 8.2.4.28.4](#)). If the Valid bit in the Cache ID Target register is 0, the H2D message shall be discarded without a response. If the Valid bit is 1, the message shall be forwarded to the local DSP based on the Port Number field that is programmed in the CXL Cache ID Target N register.

D2H response messages and D2H data messages do not carry CacheID and are always routed back to the host.

If a USP receives CXL.cache message over a link operating in 68B Flit mode, it shall process the request as if the CacheID field is 0. A switch that is not capable of decoding CacheID field must be configured such that no more than one DSP is enabled for CXL.cache traffic (indicated by Cache_Enable=1 in the DVSEC Flex Bus Port Status register; see [Section 8.2.1.3.3](#)). The USP shall direct all H2D traffic to that DSP.

9.15.4 Host Bridge Routing Functionality

When the Host Bridge receives the equivalent of an H2D request or an H2D response message from the host, the Host Bridge logic shall use the CacheID field to look up the corresponding CXL Cache ID Target N register (see [Section 8.2.4.28.4](#)). If the Valid bit is 0, the H2D message is discarded. If the Valid bit is 1, the message is forwarded to the local root port based on the Port Number field that is programmed in the CXL Cache ID Target N register.

When the Host Bridge receives a D2H request message from the root port, the Host Bridge shall forward the message to the host, using host-specific mechanisms. The Host Bridge may optionally look up the root port that is associated with the CacheID and discard the message if the message was received from a different root port.

IMPLEMENTATION NOTE

System Software may use the following sequence to allocate a Cache ID to a BI-capable CXL.cache Device D below a Switch S and enable the Device to generate CXL.cache transactions that target any memory:

1. Verify that the CXL link between Switch S and the host is operating in 256B Flit mode.
2. Identify an unused and legal CacheID value, c , and allocate it to Device D. Software must take into account the current Flit mode, as well as the Cache ID Target Count fields, while assigning Cache IDs to devices.
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. Assign Cache ID=1.
4. If the above DSP of Switch S reports Explicit Cache ID Decoder Commit Required=1, commit the Cache ID changes as follows:
 - a. Cache ID Decoder Commit=0 to rearm.
 - b. Cache ID Decoder Commit=1.
 - c. Poll bits 0 and 1 of the Cache ID Decoder Status register until timeout or one of them is set. The timeout value is reported in the Cache ID Decoder Status register.
 - d. If Cache ID Decoder Error Not Committed=1, the changes were not committed. Software should treat this as an error condition.
 - e. If Cache ID Decoder Committed=1, the changes were committed. Proceed to Step 5.
 - f. If neither bit is set and the timeout is reached, software should treat this as an error condition.
5. Configure the USP of Switch S to route Cache ID c :
 - a. Route Table[c]= Port Number register of the DSP that is connected directly to Device D.
6. If the USP reports Explicit Cache ID RT Commit Required=1, commit the Cache ID changes as follows:
 - a. Cache ID RT Commit=0 to rearm.
 - b. Cache ID RT Commit=1.
 - c. Poll bits 0 and 1 of the Cache ID RT Status register until timeout or one of them is set. The timeout value is reported in the Cache ID RT Status register.
 - d. If Cache ID RT Error Not Committed=1, the changes were not committed. Software should treat this as an error condition.
 - e. If Cache ID RT Committed=1, the changes were committed. Proceed to Step 7.
 - f. If neither bit is set and the timeout is reached, software should treat this as an error condition.
7. Configure the Root Port, R, that is directly connected to Switch S to decode the CXL.cache messages from Device D:
 - a. If Forward Cache ID=0, set Forward Cache ID=1.
 - b. Ensure Assign Cache ID=0.
8. If the previous steps were successful, configure the CXL Cache ID Route Table (see [Section 8.2.4.28.1](#)) in the Host Bridge:
 - a. Route Table[c].Port Number=Port Number register of Root Port R.
9. If the previous steps were successful, inform the device driver that Device D may now issue CXL.cache requests.

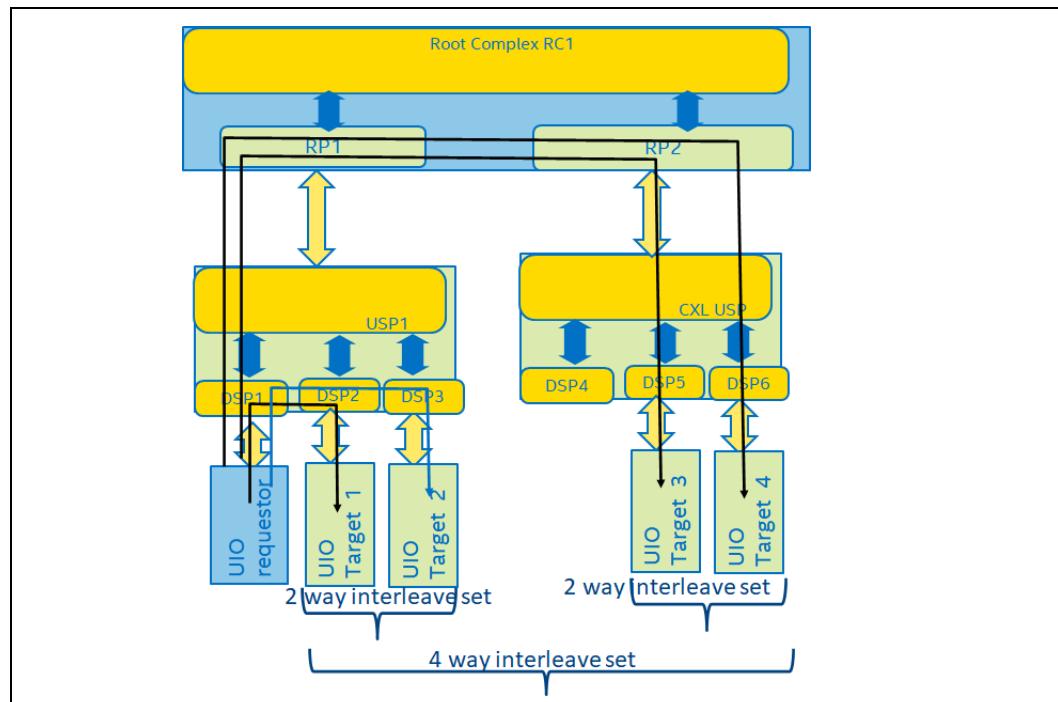
9.16

UIO Direct P2P to HDM

CXL.mem devices that can complete UIO requests that target its HDM, advertise the capability via the UIO Capable bit in the CXL HDM Decoder Capability register (see [Section 8.2.4.20.1](#)). CXL switches may allow routing of UIO accesses to HDM in the same VH as the UIO requester and advertise this capability via the same bit. CXL Host Bridges may allow routing of UIO accesses to host memory or HDM below another root ports in the same Host Bridge and advertise this capability via this bit. Prior to setting up a UIO path from a UIO requester to an HDM or to host memory, the Software must consult the capabilities of the target device and any switches or Host Bridges in the path.

[Figure 9-25](#) shows a configuration with four CXL.mem devices that form three separate interleave sets and how a UIO requester is able to access the HDM range. UIO accesses to UIO Target 1 and UIO Target 2 are directly routed by the switch, whereas UIO accesses to UIO Target 3 and UIO Target 4 are routed through the host. As shown, UIO Target 1 and UIO Target 2 participate in a 2-way interleave set. The UIO requester can efficiently access this interleave set without going through the host.

Figure 9-25. UIO Direct P2P to Interleaved HDM



The HDM that is a target of P2P UIO accesses must be part of either a 1-way, 2-way, 4-way, 8-way, or 16-way interleave set. Any HDM that is part of a 3-way, 6-way, or 12-way interleave arrangement cannot be a P2P UIO target. The HDM address must be carved out of a CFMWS entry with Interleave Arithmetic=Standard Modulo arithmetic (see [Table 9-22](#)). In addition, P2P UIO traffic is not protected by Selective IDE Streams.

In addition, Software must configure the switch and Host Bridge HDM decoders with additional information regarding any HDM interleaving calculations that are performed upstream to it before setting the UIO bit in that HDM decoder. The UIG, UIW, and ISP fields allow the switch and the Host Bridge to determine whether the UIO target address belongs to itself or to a peer component. The rules regarding the processing of UIO Direct P2P to HDM requests are described in [Table 9-18](#). The ISP field in the target CXL.mem device allow the device to determine how it should respond.

These requirements are in addition to the UIO related requirements that are defined in PCIe Base Specification.

9.16.1 Processing of UIO Direct P2P to HDM Messages

This section describes how CXL components handle UIO Direct P2P accesses to HDM.

UIO To HDM Enable bit is defined in [Section 8.1.5.2](#) and allows System Software to control whether a requester below a switch can use UIO to access HDM.

Table 9-18. Handling of UIO Accesses

Received by	UIO Address	Behavior
CXL.mem device that reports UIO Capable=1 (see Section 8.2.4.20.1)	Complete match with an HDM decoder with UIO=1	Respond to the UIO request per PCIe Base Specification
	Complete match with an HDM decoder with UIO=0	Return Completer Abort, do not commit data if it is a UIO write
	Partial match with an HDM decoder, irrespective of the UIO bit	Return Completer Abort, do not commit data if it is a UIO write
	Mismatch	Handle per PCIe Base Specification
USP ingress of a CXL Switch that reports UIO Capable=1 (see Section 8.2.4.20.1)	Either Partial or Complete match with an HDM decoder, irrespective of the UIO bit	Identify the port number of the target DSP and forward
	Mismatch	Handle per PCIe Base Specification
DSP ingress of a CXL Switch that reports UIO Capable=1 (see Section 8.2.4.20.1)	Complete match with an HDM decoder with UIO=1 ¹ and UIO To HDM Enable=1	Identify the port number of the target DSP and forward to that peer port regardless of ACS configuration including egress control vector
	Complete match with an HDM decoder with UIO=0 and UIO To HDM Enable=1	Forward toward the host regardless of ACS configuration including egress control vector
	Partial match with an HDM decoder and UIO To HDM Enable=1	Forward toward the host regardless of ACS configuration including egress control vector
	Complete or Partial match, and UIO To HDM Enable=0	Return Completer Abort
	Mismatch	Handle per PCIe Base Specification
RP ingress of a Host Bridge that reports UIO Capable=1 (see Section 8.2.4.20.1)	Complete match with an HDM decoder with UIO=1	Identify the port number of the target RP and forward to that peer port, subject to host-specific access controls
	Complete match with an HDM decoder with UIO=0	Handle via host-specific mechanisms
	Partial match with an HDM decoder	Handle via host-specific mechanisms
	Mismatch	Handle via host-specific mechanisms

- Because the DSP does not take length into account during this check, transactions that cross an interleave boundary get forwarded to the device that owns the starting address. They are aborted by the device because the device checks the length field. If the UIO traffic is encrypted using Stream IDE, some of the address bits may be encrypted and the switch may unknowingly forward these to the wrong device, which will issue a Completer Abort.

9.16.1.1 UIO Address Match (DSP and Root Port)

For a DSP or a root port, UIO address is considered a complete match if there exists an HDM Decoder[n] (see [Section 8.2.4.20](#) and [Section 8.2.4.30](#)) for which the following conditions are true:

- AT field in the UIO request indicates that it is carrying a translated address.
- UIO.Address[63:2] ≥ Decoder[n].Base[63:2].

3. $\text{UIO.Address}[63:2] + \text{UIO.Length}[63:2] \leq \text{Decoder}[n].\text{Base}[63:2] + \text{Decoder}[n].\text{Size}[63:2]$.
4. Either of these sub-conditions are true:
 - a. $\text{Decoder}[n].\text{UIW} = 0$
 - b. $\text{UIO.Address}[\text{Decoder}[n].\text{UIW} + \text{Decoder}[n].\text{IG} + 7 : \text{Decoder}[n].\text{IG} + 8] = \text{ISP}$

where $\text{UIO.Address}[63:2]$ is derived from the Address field in the UIO TLP request, and $\text{UIO.Length}[63:2]$ is derived from the Length field in the UIO TLP request.

DSP calculations use the HDM decoders in the corresponding USP. The root port calculations make use of the HDM decoders in the associated Host Bridge.

The first condition is in place because HDM decoder operates on translated address. The second and the third condition ensures that all addresses fall within one of the HDM decoders. The fourth condition ensures that the interleave set positions match (i.e., a CXL.mem request from the host to the start address would ordinarily be decoded by this component). 4a is the trivial case where the memory is not interleaved.

If the first three conditions are met but the fourth condition is not met, it is considered a partial match. If the first three conditions are not met, it is considered a mismatch.

9.16.1.2 UIO Address Match (CXL.mem Device)

For a CXL.mem device, UIO address is considered a complete match if there exists an HDM Decoder[n] (see [Section 8.2.4.20](#) and [Section 8.2.4.30](#)) for which the following conditions are true:

1. AT field in the UIO request indicates it is carrying a translated address.
2. $\text{UIO.Address}[63:2] \geq \text{Decoder}[n].\text{Base}[63:2]$.
3. $\text{UIO.Address}[63:2] + \text{UIO.Length}[63:2] \leq \text{Decoder}[n].\text{Base}[63:2] + \text{Decoder}[n].\text{Size}[63:2]$.
4. Either of these sub-conditions are true:
 - a. $\text{Decoder}[n].\text{UIW} = 0$
 - b. $\text{UIO.Address}[\text{Decoder}[n].\text{IG} + 7 : \text{Decoder}[n].\text{IG} + 8] = \text{ISP}$
5. $\text{UIO.Address}[\text{Decoder}[n].\text{IG} + 7 : 2] + \text{UIO.Length}[\text{Decoder}[n].\text{IG} + 7 : 2] \leq (2^{**} \text{IG} + 8)$.

The first three conditions are identical to the DSP case. The terms involved in the fourth check are different, but it serves the same purpose (i.e., ensures that a CXL.mem request from the host to the start address would ordinarily be decoded by this component). The fifth condition ensures that the access does not cross an interleave boundary, thus ensuring that all the addresses that are referenced by the request are owned by the device.

If the first three conditions are met but either of the other two conditions are not met, it is considered a partial match. If the first three conditions are not met, it is considered a mismatch.

9.17

Direct P2P CXL.mem for Accelerators

The Direct P2P CXL.mem feature enables accelerators to use .mem semantics to access peer Type 3 devices. This feature is supported only by PBR Fabrics, and each accelerator and peer Type 3 device must be attached directly to an Edge Port. Configuration of the Fabric and Edge Ports is performed by the host and FM.

Through mechanisms beyond the scope of this specification, the FM is preconfigured or informed of which Type 3 device(s) (i.e., SLD, MLD, or GFD) are to be configured for Direct P2P CXL.mem access by a given accelerator.

9.17.1

Peer SLD Configuration

Host software and the FM may use the following high-level flow to configure Direct P2P CXL.mem communication between an accelerator and a peer Type 3 SLD:

1. The FM binds the SLD's Edge Port to the host VH of the accelerator, setting the vPPB.root.PID field to the PBR ID (PID) of the accelerator's Edge Port. This enables the host to configure the SLD, but the accelerator to carry out CXL.mem transactions with the SLD.
2. Using the **Set LDST Segment Entries** command (see [Section 7.7.13.16](#)), the host configures the LDST in the accelerator's Edge Port with one or more LDST Segments for the HPA range of the SLD, specifying the vPPB of the SLD's Edge Port.
3. Host software configures the SLD, notably its HDM Decoders, on behalf of the accelerator. HDM addresses in the SLD are HPAs.

9.17.2

Peer MLD Configuration

Host software and the FM may use the following high-level flow to configure Direct P2P CXL.mem communication between one or more accelerators that belong to the host and a peer Type 3 MLD:

1. The FM binds a vPPB in the MLD's Edge Port to the host VH of its accelerator(s) and an additional vPPB for each accelerator under that host that will be accessing the MLD. Each of these will have a distinct LD-ID. For each vPPB assigned to an accelerator, the vPPB.root.PID field is set to the PID of the accelerator's Edge Port.
2. Using the **Set LDST Segment Entries** command (see [Section 7.7.13.16](#)), the host configures the LDST in each accelerator's Edge Port with one or more LDST Segments for the HPA range of the accelerator's LD, specifying the accelerator's vPPB in the MLD's Edge Port.
3. Host software configures its LDs in the MLD, notably their HDM Decoders, on behalf of itself and its accelerator(s). HDM addresses in the LD of the host and the LD(s) of the accelerator(s) are HPAs.

9.17.3

Peer GFD Configuration

Host software and the FM may use the following high-level flow to configure Direct P2P CXL.mem communication between one or more accelerators that belong to a host and a peer Type 3 GFD:

1. The FM configures the GFD for host access normally, while configuring each of the host's accelerators as an additional RPID within the GFD.
2. Using the **Set FAST Segment Entries** command (see [Section 7.7.14.7](#)), the host configures the FAST decoder in its Edge Port as well as each accelerator's Edge Port with one or more FAST Segments for the HPA range, specifying the GFD's PID.

9.18

CXL OS Firmware Interface Extensions

9.18.1

CXL Early Discovery Table (CEDT)

CXL Early Discovery Table enables OSs to locate CXL Host Bridges and the location of Host Bridge registers early during the boot (i.e., prior to parsing of ACPI namespace). The information in this table may be used by early boot code to perform pre-initialization of CXL hosts, such as configuration of CXL.cache and CXL.mem.

9.18.1.1 CEDT Header

The pointer to CEDT is found in RSDT or XSDT, as described in ACPI Specification. An ACPI specification-compliant CXL system shall support CEDT and shall include a CHBS entry for every CXL host bridge that is present at boot.

CEDT begins with the following header.

Table 9-19. CEDT Header

Field	Length in Bytes	Byte Offset	Description
Header:			
Signature	4	00h	'CEDT'. Signature for the CXL Early Discovery Table.
Length	4	04h	Length, in bytes, of the entire CEDT.
Revision	1	08h	Value is 2.
Checksum	1	09h	Entire table must sum to 0.
OEM ID	6	0Ah	OEM ID
OEM Table ID	8	10h	Manufacturer Model ID
OEM Revision	4	18h	OEM Revision
Creator ID	4	1Ch	Vendor ID of the utility that created the table.
Creator Revision	4	20h	Revision of the utility that created the table.
CEDT Structure[n]	Varies	24h	A list of CEDT structures for this implementation.

Table 9-20. CEDT Structure Types

Value	Description
0	CXL Host Bridge Structure (CHBS)
1	CXL Fixed Memory Window Structure (CFMWS)
2	CXL XOR Interleave Math Structure (CXIMS)
3	RCEC Downstream Port Association Structure (RDPAS)
4	CXL System Description Structure (CSDS) ¹
5-255	Reserved

1. Introduced in Revision 2 of CEDT.

9.18.1.2 CXL Host Bridge Structure (CHBS)

The CHBS structure describes a CXL Host Bridge.

In an ACPI-compliant system, there shall be one instance of the CXL Host Bridge Device object in ACPI namespace (HID="ACPI0016") for every CHBS entry. The _UID object under a CXL Host Bridge object, when evaluated, shall match the UID field in the associated CHBS entry.

Table 9-21. CHBS Structure

Field	Length in Bytes	Byte Offset	Description
Type	1	00h	=0 to indicate that this is a CHBS entry
Reserved	1	01h	Reserved
Record Length	2	02h	Length of this record (20h).
UID	4	04h	CXL Host Bridge Unique ID. Used to associate a CHBS instance with a CXL Host Bridge instance. The value of this field shall match the output of _UID under the associated CXL Host Bridge in ACPI namespace.
CXL Version	4	08h	<ul style="list-style-type: none"> • 0000 0000h: RCH • 0000 0001h: Host Bridge that is associated with one or more CXL root ports
Reserved	4	0Ch	Reserved
Base	8	10h	<ul style="list-style-type: none"> • If CXL Version = 0000 0000h, this represents the base address of the RCH Downstream Port RCRB • If CXL Version = 0000 0001h, this represents the base address of the CHBCR <p>See Table 8-17 for more details.</p>
Length	8	18h	<ul style="list-style-type: none"> • If CXL Version = 0000 0000h, this field must be set to 8 KB (2000h) • If CXL Version = 0000 0001h, this field must be set to 64 KB (1 0000h)

9.18.1.3 CXL Fixed Memory Window Structure (CFMWS)

The CFMWS structure describes zero or more Host Physical Address (HPA) windows that are associated with each CXL Host Bridge. Each window represents a contiguous HPA range that may be interleaved across one or more targets, some of which are CXL Host Bridges. Associated with each window are a set of restrictions that govern its usage. It is the OSPM's responsibility to utilize each window for the specified use.

The HPA ranges described by CFMWS may include addresses that are currently assigned to CXL.mem devices. Before assigning HPAs from a fixed-memory window, the OSPM must check the current assignments and avoid any conflicts.

For any given HPA, it shall not be described by more than one CFMWS entry.

Table 9-22. CFMWS Structure (Sheet 1 of 4)

Field	Length in Bytes	Byte Offset	Description
Type	1	00h	1 = indicates this is a CFMWS entry
Reserved	1	01h	Reserved
Record Length	2	02h	Length of this record = 024h + 4 * NIW. NIW is the raw count of Interleave ways whereas ENIW is the encoded value: <ul style="list-style-type: none"> • If ENIW<8, NIW=2**ENIW • If ENIW≥8, NIW=3* 2** (ENIW-8)
Reserved	4	04h	Reserved
Base HPA	8	08h	Base of this HPA range. This value shall be a 256-MB-aligned address.
Window Size	8	10h	The total number of consecutive bytes of HPA this window represents. This value shall be a multiple of NIW*256 MB.

Table 9-22. CFMWS Structure (Sheet 2 of 4)

Field	Length in Bytes	Byte Offset	Description
Encoded Number of Interleave Ways (ENIW)	1	18h	The encoded number of targets with which this window is interleaved. The valid encoded values are specified in the Interleave Ways field of the CXL HDM Decoder n Control register (see Section 8.2.4.20.7). This field determines the number of entries in the Interleave Target List, starting at Offset 24h.
Interleave Arithmetic	1	19h	This field defines the arithmetic used for mapping HPA to an interleave target in the Interleave Target List: <ul style="list-style-type: none"> • 00h = Standard Modulo arithmetic • 01h = Modulo arithmetic combined with XOR • All other encodings are reserved
Reserved	2	1Ah	Reserved
Host Bridge Interleave Granularity (HBIG)	4	1Ch	The number of consecutive bytes within the interleave that are decoded by each target in the Interleave Target List represented in an encoded format. The valid values are specified in the Interleave Granularity field of the CXL HDM Decoder n Control register (see Section 8.2.4.20.7).
Window Restrictions	2	20h	A bitmap describing the restrictions being placed on the OSPM's use of the window. It is the OSPM's responsibility to adhere to these restrictions. Failure to adhere to these restrictions results in undefined behavior. More than one bit within this field may be set: <ul style="list-style-type: none"> • Bit[0]: Device Coherent: Formerly known as CXL Type 2 Memory: <ul style="list-style-type: none"> — 1 = Window is configured to expose device-coherent memory (HDM-D if Bit[5]=0 ; HDM-DB if Bit[5]=1). • Bit[1]: Host-only Coherent: Formerly known as CXL Type 3 Memory: <ul style="list-style-type: none"> — 1 = Window is configured to expose host-only coherent memory (HDM-H). If an HDM decoder that is mapped to this windows has the BI bit set, it will result in undefined behavior. • Bit[2]: Volatile: <ul style="list-style-type: none"> — 1 = Window is configured for use with volatile memory. • Bit[3]: Persistent: <ul style="list-style-type: none"> — 1 = Window is configured for use with persistent memory. • Bit[4]: Fixed Device Configuration: <ul style="list-style-type: none"> — 1 = Any device ranges that have been assigned an HPA from this window must not be reassigned. • Bit[5]: BI: <ul style="list-style-type: none"> — 1 = Window is configured for use with Back-Invalidate flows. • Bits[15:6]: Reserved
QTG ID	2	22h	The ID of the QoS Throttling Group associated with this window. The _DSM for retrieving QTG ID is utilized by the OSPM to determine to which QTG a device HDM range should be assigned. This field must not exceed the Max Supported QTG ID returned by the _DSM for retrieving QTG.

Table 9-22. CFMWS Structure (Sheet 3 of 4)

Field	Length in Bytes	Byte Offset	Description
Interleave Target List	4*NIW	24h	<p>A list of all the Interleave Targets. The number of entries in this list shall match the Number of Interleave Ways (NIW). The order of the targets reported in this List indicates the order in the Interleave Set.</p> <p>For Interleave Sets that only span CXL Host Bridges, this is a list of CXL Host Bridge _UIDs that are part of the Interleave Set. In this case, for each _UID value in this list, there must exist a corresponding CHBS structure.</p> <p>If the Interleave Set spans non-CXL domains, this list may contain values that do not match _UID field in any CHBS structures. These entries represent Interleave Targets that are not CXL Host Bridges.</p> <p>The set of HPAs decoded by Entry N in the Interleave Target List shall satisfy the following equations:</p> <ol style="list-style-type: none"> 1. Base HPA <= HPA < Base HPA + Windows Size: where the Base HPA and Window size shall be multiple of NIW. If (Interleave Arithmetic==0): <ol style="list-style-type: none"> a. If ENIW=0 N=0 b. If ENIW=1 N= HPA[8+HBIG] c. If ENIW<8 AND ENIW>1 N = HPA[7+HBIG+ENIW:8+HBIG] d. If NIW = 8 // 3 way N = HPA[51:8+HBIG] MOD 3 e. If NIW=9 // 6 way N = HPA[8+HBIG] + 2* HPA[51:9+HBIG] MOD 3 f. If NIW=10 //12 way N = HPA[9+HBIG:8+HBIG] + 4* HPA[51:10+HBIG] MOD 3 2. If (Interleave Arithmetic==1): <ol style="list-style-type: none"> a. If NIW=0 //1 way N=0 b. If NIW =1 // 2 way N = XORALLBITS (HPA AND XORMAP[0]) If NIW=2 // 4 way N = XORALLBITS (HPA AND XORMAP[0]) + 2* XORALLBITS (HPA AND XORMAP[1])

Table 9-22. CFMWS Structure (Sheet 4 of 4)

Field	Length in Bytes	Byte Offset	Description
Interleave Target List	4*NIW	24h	<p>c. If NIW=3 // 8 way $N = \text{XORALLBITS}(\text{HPA AND XORMAP}[0]) +$ $* \text{XORALLBITS}(\text{HPA AND XORMAP}[1]) +$ $* \text{XORALLBITS}(\text{HPA AND XORMAP}[2])$</p> <p>d. If NIW=4 //16 way $N = \text{XORALLBITS}(\text{HPA AND XORMAP}[0]) +$ $2 * \text{XORALLBITS}(\text{HPA AND XORMAP}[1]) +$ $4 * \text{XORALLBITS}(\text{HPA AND XORMAP}[2]) +$ $8 * \text{XORALLBITS}(\text{HPA AND XORMAP}[3])$</p> <p>e. If NIW =8 // 3 way, same as Interleave Arithmetic=0 $N = \text{HPA}[51:8+\text{HBIG}] \text{ MOD } 3$</p> <p>f. If NIW =9 // 6 way $N = \text{XORALLBITS}(\text{HPA AND XORMAP}[0])$ $+ 2 * \text{HPA}[51:9+\text{HBIG}] \text{ MOD } 3$</p> <p>g. If NIW=10 // 12 way $N = \text{XORALLBITS}(\text{HPA AND XORMAP}[0])$ $+ 2 * \text{XORALLBITS}(\text{HPA AND XORMAP}[1])$ $+ 4 * \text{HPA}[51:10+\text{HBIG}] \text{ MOD } 3$</p> <p>N is 0 based ($0 \leq N < \text{NIW}$). Where XORALLBITS is an operation that outputs a single bit by XORing all the bits in the input. AND is a standard bitwise AND operation and XORMAP[m] is the mth element (m is 0 based) in the XORMAP array that is part of the CXIMS entry with the matching HBIG value.</p>

9.18.1.4 CXL XOR Interleave Math Structure (CXIMS)

If a CFMWS entry reports Interleave Arithmetic=1, there must be one CXIMS entry associated with the HBIG value in the CFMWS. CXIMS carries an array of bitmaps. Each bitmap represents the bits that are XORed together to calculate the individual bits of the Interleave Way as described in the definition of the Interleave Target List field in CFMWS. The host implementation is responsible for selecting an XOR function that generates even distribution of addresses and does not lead to address aliasing.

Table 9-23. CXIMS Structure

Field	Length in Bytes	Byte Offset	Description
Type	1	00h	2 = Indicates that this is a CXIMS entry
Reserved	1	01h	Reserved
Record Length	2	02h	Length of this record = 8 + 8 * NIB.
Reserved	2	04h	Reserved
HBIG	1	06h	Host Bridge Interleave Granularity to which this CXIMS instance corresponds. See Table 9-22 for the definition of the term HBIG.
Number of Bitmap Entries (NIB)	1	07h	The number of entries in the XORMAP list.
XORMAP List	8 * NIB	08h	A list of Bitmaps. XORMAP[0] is the first entry.

9.18.1.5 RCEC Downstream Port Association Structure (RDPAS)

RDPAS structure enables error handler to locate the Downstream Port(s) that report errors to a given RCEC. For every RCEC, zero or more entries of this type are permitted.

Table 9-24. RDPAS Structure

Field	Length in Bytes	Byte Offset	Description
Type	1	00h	3 = Indicates that this is an RDPAS entry
Reserved	1	01h	Reserved
Record Length	2	02h	Length of this record = 10h
RCEC Segment Number	2	04h	The PCIe segment number associated with this RCEC
RCEC BDF	2	06h	<ul style="list-style-type: none"> • Bits[2:0]: RCEC Function Number • Bits[7:3]: RCEC Device Number • Bits[15:8]: RCEC Bus Number
Protocol Type	1	07h	<ul style="list-style-type: none"> • 00h = The error source is CXL.io • 01h = The error source is CXL.cachemem
Base Address	8	08h	If Protocol Type = CXL.io, this field shall be the RCRB base associated with the Downstream Port. If Protocol Type = CXL.cachemem, this will be the Component Base Register Base associated with the Downstream Port.

IMPLEMENTATION NOTE

CXL-aware software may take the following steps upon observing an Uncorrected Internal Error or an Corrected Internal Error being logged in an RCEC at Segment Number S and BDF=B.

If the CEDT contains RDPAS structures:

- For all RDPAS structures where RCEC Segment Number=S and RCEC BDF= B:
 - If Protocol Type=CXL.io, read the Base Address field and use that information to access the RCRB AER registers and determine whether any errors are logged there
 - If Protocol Type=CXL.cachemem, read the Base Address field and use that information to access the Component Register RAS Capability registers (see [Section 8.2.4.17](#)) and determine whether any errors are logged there

Else:

- Probe all CXL Downstream Ports and determine whether they have logged an error in the CXL.io or CXL.cachemem status registers

9.18.1.6 CXL System Description Structure (CSDS)

The CSDS describes CXL System Wide Description and Configuration.

In a system, there shall be only one instance of the CSDS in the CEDT.

Table 9-25. CSDS Structure

Field	Length in Bytes	Byte Offset	Description
Type	1	00h	4 = Indicates that this is a CSDS entry
Reserved	1	01h	Reserved
Record Length	2	02h	Length of this record = 08h
System Capabilities	2	04h	A bitmap that describes system-wide capabilities. More than one bit within this field is permitted to be set. <ul style="list-style-type: none"> • Bit[0]: Cmp-M: <ul style="list-style-type: none"> — 1 = System is configured for use with devices that return modified data using the Cmp-M response. • Bit[1]: No Clean Writeback: Specifies the clean writeback behavior of the host. <ul style="list-style-type: none"> — 0 = The host may or may not generate clean writebacks — 1 = The host guarantees to never generate clean writeback transactions at the host's cacheline granularity • Bits[15:2]: Reserved
Reserved	2	06h	Reserved

9.18.2 CXL _OSC

According to ACPI Specification, _OSC (Operating System Capabilities) is a control method that is used by OSs to communicate to the System Firmware the capabilities supported by the OS and to negotiate ownership of specific capabilities.

The _OSC interface defined in this section applies only to “Host Bridge” ACPI devices that originate CXL hierarchies. As specified in [Section 9.12](#), these ACPI devices must have an _HID of (or a _CID that includes) EISAID(“ACPI0016”). CXL _OSC is required for a CXL VH. CXL _OSC is optional for an RCD. A CXL Host Bridge also originates a PCIe hierarchy and will have a _CID of EISAID(“PNP0A08”). As such, a CXL Host Bridge device may expose both CXL _OSC and PCIe _OSC.

The _OSC interface for a CXL Host Bridge is identified by the Universal Unique Identifier (UUID) 68f2d50b-c469-4d8a-bd3d-941a103fd3fc.

A revision ID of 1 encompasses fields defined within this section, composed of 5 DWORDs, as listed in [Table 9-26](#).

Table 9-26. _OSC Capabilities Buffer DWORDs (Sheet 1 of 2)

_OSC Capabilities Buffer DWORD #	Description
1	Contains bits that are generic to _OSC and defined by ACPI. These include status and error information.
2	PCIe Support Field as defined by PCI Firmware Specification.

Table 9-26. _OSC Capabilities Buffer DWORDs (Sheet 2 of 2)

_OSC Capabilities Buffer DWORD #	Description
3	PCIe Control Field as defined by PCI Firmware Specification.
4	CXL Support Field: Bits defined in the CXL Support Field provide information regarding CXL features supported by the OS. Just like the PCIe Support field, contents in the CXL Support Field are passed in a single direction; the OS will disregard any changes to this field when returned.
5	CXL Control Field: Just like the PCIe Control Field, bits defined in the CXL Control Field are used to submit OS requests for control/handling of the associated feature, typically including but not limited to features that utilize native interrupts or events that are handled by an OS-level driver. If any bits in the CXL Control Field are returned cleared (i.e., masked to 0) by the _OSC control method, the respective feature is designated as unsupported by the platform and must not be enabled by the OS. Some of these features may be controlled by System Firmware prior to OS boot or during runtime for an OS that is unaware of these features, while others may be disabled/inoperative until native OS support for such features is available. If the CXL _OSC control method is absent from the scope of a Host Bridge device, then the OS must not enable or attempt to use any features defined in this section for the hierarchy originated by the Host Bridge. Doing so could conflict with System Firmware operations, or produce undesired results. It is recommended that a machine with multiple Host Bridge devices should report the same capabilities for all Host Bridges, and also negotiate control of the features described in the CXL Control Field in the same way for all Host Bridges.

Table 9-27. Interpretation of CXL _OSC Support Field

Support Field Bit Offset	Interpretation
0	RCD and RCH Port Register Access Supported The OS sets this bit to 1 if it supports access to RCD and RCH Port registers as defined in Section 9.11 . Otherwise, the OS clears this bit to 0.
1	CXL VH Register Access Supported The OS sets this bit to 1 if it supports access to CXL VH component registers as defined in Section 9.12 . If this bit is 1, bit 0 must also be 1. Otherwise, the OS clears this bit to 0.
2	CXL Protocol Error Reporting Supported The OS sets this bit to 1 if it supports handling of CXL Protocol Errors. Otherwise, the OS clears this bit to 0. If the OS sets this bit, it must also set either bit 0 or bit 1 above. Note: Firmware may retain control of AER if the OS does not support CXL Protocol Error reporting because the owner of AER owns CXL Protocol error management.
3	CXL Native Hot-Plug Supported The OS sets this bit to 1 if it supports CXL hot-add and managed CXL Hot-Remove without firmware assistance. Otherwise, the OS clears this bit to 0. If the OS sets this bit, it must request PCIe Native Hot-Plug control. If PCIe Native Hot-Plug control is granted to the OS, such an OS must natively handle CXL Hot-Plug as well. If the OS sets this bit, it must also set bit 1 above.
4-31	Reserved

Table 9-28. Interpretation of CXL _OSC Control Field, Passed in via Arg3

Control Field Bit Offset	Interpretation
0	CXL Memory Error Reporting Control The OS sets this bit to 1 to request control over CXL Memory Error Reporting. If the OS successfully receives control of this feature, it must handle memory errors from all CXL.mem-capable devices that support this capability, as described in Section 12.2.3.2 . If the OS sets this bit, the OS must also set either bit 0 or bit 1 in the CXL _OSC Support Field (see Table 9-26).
1-31	Reserved

Table 9-29. Interpretation of CXL _OSC Control Field, Returned Value

Control Field Bit Offset	Interpretation
0	CXL Memory Error Reporting Control The firmware sets this bit to 1 to grant control over CXL Memory Expander Error Reporting. If firmware grants control of this feature, firmware must ensure that memory expanders are not configured in Firmware First error reporting mode. If control of this feature was requested and denied or was not requested, firmware returns this bit cleared to 0.
1-31	Reserved

9.18.2.1 Rules for Evaluating _OSC

This section defines when and how the OS must evaluate _OSC, as well as restrictions on firmware implementations.

9.18.2.1.1 Query Support Flag

If the Query Support Flag (_OSC Capabilities Buffer DWORD 1, bit 0) is set by the OS while evaluating _OSC, hardware settings are not permitted to be changed by firmware in the context of the _OSC call. It is strongly recommended that the OS evaluate _OSC with the Query Support Flag set until _OSC returns the Capabilities Masked bit cleared to negotiate the set of features to be granted to the OS for native support. A platform may require a specific combination of features to be natively supported by an OS before granting native control of a given feature.

9.18.2.1.2 Evaluation Conditions

The OS must evaluate _OSC under the following conditions:

- During initialization of any driver that provides native support for features described in the section above. These features may be supported by one or many drivers, but should be evaluated only by the main bus driver for that hierarchy. Secondary drivers must coordinate with the bus driver to install support for these features. Drivers shall not relinquish control of previously obtained features. That is, bits set in _OSC Capabilities Buffer DWORD 3 and DWORD 5 after the negotiation process must be set on all subsequent negotiation attempts.
- When a Notify(<device>, 8) is delivered to the CXL Host Bridge device.
- Upon resume from S4, System Firmware will handle context restoration when resuming from S1 through S3.

If a CXL Host Bridge device exposes CXL _OSC, CXL-aware OSPM shall evaluate CXL _OSC and not evaluate PCIe _OSC.

9.18.2.1.3 Sequence of _OSC Calls

The following rules govern sequences of calls to _OSC that are issued to the same Host Bridge and occur within the same boot:

- The OS is permitted to evaluate _OSC an arbitrary number of times.
- If the OS declares support of a feature in the Status Field in one call to _OSC, then it must preserve the set state of that bit (and thereby declare support for that feature) in all subsequent calls.
- If the OS is granted control of a feature in the Control Field in one call to _OSC, then it must preserve the set state of that bit (requesting that feature) in all subsequent calls.

- Firmware shall not reject control of any feature it has previously granted control to.
- There is no mechanism for the OS to relinquish control of a previously requested and granted feature.

9.18.2.1.4 ASL Example

```

Device (CXL0)
{
    Name( _HID,EISAID("ACPI0016")) // CXL Host Bridge
    Name( _CID, Package(2) {
        EISAID("PNP0A03"), // PCI Compatible Host Bridge
        EISAID("PNP0A08") // PCI Express Compatible Host Bridge
    })
}

Name (SUPP,0) // PCI _OSC Support Field value
Name (CTRL,0) // PCI _OSC Control Field value
Name (SUPC,0) // CXL _OSC Support Field value
Name (CTRC,0) // CXL _OSC Control Field value

Method( _OSC,4)
{
    // Check for proper UUID
    If(LEqual(Arg0,ToUUID("68f2d50b-c469-4d8a-bd3d-941a103fd3fc")))
    {
        // Create DWord-addressable fields from the Capabilities Buffer
        CreateDWordField(Arg3,0,CDW1)
        CreateDWordField(Arg3,4,CDW2)
        CreateDWordField(Arg3,8,CDW3)
        CreateDWordField(Arg3,12,CDW4)
        CreateDWordField(Arg3,16,CDW5)
        // Save Capabilities DWord2, 3. 4. 5
        Store(CDW2,SUPP)
        Store(CDW3,CTRL)
        Store(CDW4,SUPC)
        Store(CDW4,CTRC)
        ..
        ..
    } Else {
        Or(CDW1,4,CDW1) // Unrecognized UUID
        Return(Arg3)
    }
} // End _OSC
// Other methods such as _BBN, _CRS, PCIe _OSC
} //End CXL0

```

9.18.3 CXL Root Device Specific Methods (_DSM)

_DSM is a control method that enables devices to provide device-specific functions for the benefit of the device driver. See ACPI Specification for details. [Table 9-30](#) lists the _DSM Functions that are associated with the CXL Root Device (HID="ACPI0017").

Table 9-30. _DSM Definitions for CXL Root Device

UUID	Revision	Function	Description
F365F9A6-A7DE-4071-A66A-B40C0B4F8E52	1	1	Retrieve QTG ID (see Section 9.18.3.1)
	-	All other	Reserved

All other Function values are reserved. The Revision field represents the version of the individual _DSM Function. The Revision associated with a _DSM Function is incremented whenever that _DSM Function is extended to add more functionality. Backward compatibility shall be maintained during this process. Specifically, for all values of n, a _DSM Function with Revision n+1 may extend Revision ID n by assigning meaning to the fields that are marked as reserved in Revision n but must not redefine the meaning of existing fields and must not change the size or type of I/O parameters.

Software that was written for a lower Revision may continue to operate on _DSM Functions with a higher Revision but will not be able to take advantage of new functionality. It is legal for software to invoke a _DSM Function and pass in any nonzero Revision ID value that does not exceed the Revision ID defined in this specification for that _DSM Function.

For example, if the most-current version of this specification defines Revision ID=4 for _DSM Function Index f, software is permitted to invoke the _DSM Function with Function Index f with a Revision ID value that belongs to the set {1, 2, 3, 4}.

9.18.3.1 [_DSM Function for Retrieving QTG ID](#)

This section describes how the OSPM can request the firmware to determine the optimum QoS Throttling Group (QTG) to which a device HDM range should be assigned, based on its performance characteristics. It is strongly recommended that OSPM evaluate this _DSM Function to retrieve QTG recommendations and map the device HDM range to an HPA range that is described by a CFMWS entry that follows the platform recommendations.

For each Device Scoped Memory Affinity Structure (DSMAS) in the Device CDAT, the OSPM should calculate the Read Latency, Write Latency, Read Bandwidth, and Write Bandwidth from the CXL Root Port within the same VCS. The term DSMAS is defined in Coherent Device Attribute Table Specification. This calculation must consider the latency and bandwidth contribution of any intermediate switches. The OSPM should call this _DSM with the performance characteristics for the Device HDM range thus calculated, utilize the return ID value(s) to pick an appropriate CFMWS, and then map the DSMAS DPA range to HPAs that are covered by that CFMWS. This process may be repeated for each DSMAS memory range that the OSPM wishes to utilize from the device.

Location:

This object shall be a child of the CXL Root Device (HID="ACPI0017").

Arguments:

Arg0: UUID: f365f9a6-a7de-4071-a66a-b40c0b4f8e52

Arg1: Revision ID: 1

Arg2: Function Index: 01h

Arg3: A package of memory device performance characteristic. The package consists of 4 DWORDs.

Package {

Read Latency

Write Latency

Read Bandwidth

Write Bandwidth

}

Return:

A package containing two elements - a WORD that returns the maximum throttling group that the platform supports, and a package containing the QTG ID(s) that the platform recommends.

```
Package {
    Max Supported QTG ID
    Package {QTG Recommendations}
}
```

Table 9-31. _DSM for Retrieving QTG, Inputs, and Outputs

Field	Size	Description
Input Package:		
Read Latency	DWORD	The best-case read latency as measured from the CXL root port within the same VCS, expressed in picoseconds.
Write Latency	DWORD	The best-case write latency as measured from the CXL root port within the same VCS, expressed in picoseconds.
Read Bandwidth	DWORD	The best-case read bandwidth as measured from the CXL root port within the same VCS, expressed in MB/s.
Write Bandwidth	DWORD	The best-case write bandwidth as measured from the CXL root port within the same VCS, expressed in MB/s.
Return Package:		
Max Supported QTG ID	WORD	<p>The highest QTG ID supported by the platform. The platform must be capable of supporting all QTGs whose ID, Q, satisfies the following equation:</p> $0 > Q \geq \text{Max Supported QTG ID}$ <p>For every value of Q, there may be zero or more CFMWS entries.</p>
QTG Recommendations	Package	<p>A package that consists of 0 or more WORD elements. It is a prioritized list of QTG IDs that are considered acceptable by the platform for the specified performance characteristics. If the package contains more than one element, element[n] is preferred by the platform over element[n+1]. If the package is empty, the platform is unable to find any suitable QTGs for this set of input values. If the OSPM does not follow platform QTG recommendations, it may result in severe performance degradation.</p> <p>Every element in this package must be no greater than the Max Supported QTG ID above.</p> <p>For example, if QTG Recommendations = Package () {2, 1}, the OSPM should first attempt to assign from QTG ID 2, and then attempt to assign QTG ID 1 if an assignment cannot be found in QTG ID 2.</p>

9.19**Manageability Model for CXL Devices**

Manageability is the set of capabilities that a managed entity exposes to a management entity. In the context of CXL, a CXL device is the managed entity. These capabilities are generally classified in sensors and effectors. An Event Log is an example of a sensor, whereas the ability to update the device firmware is an example of an effector. Sensors and effectors can either be accessed in-band (i.e., by OS/VMM resident software), or out-of-band (i.e., by firmware running on a management controller that is OS independent).

In-band software can access a CXL device's manageability capabilities by issuing PCIe configuration read/write or MMIO read/write transactions to its Mailbox registers. These accesses are generally mediated by the CXL device driver. This is consistent with how PCIe adapters are managed.

Out-of-band manageability in S0 state can leverage transports for which an MCTP binding specification has been defined. This assumes that the CXL.io path will decode and forward MCTP over PCIe VDMs in both directions. Form factors, such as PCIe CEM Specification, provision two SMBUS pins (clock and data). The SMBUS path can be used for out-of-band manageability in Sx state or in the Link Down case. This is consistent with PCIe adapters. CXL components may also support additional management capabilities defined in other specifications, such as Platform-Level Data Model (PLDM).

9.20

Component Command Interface

Runtime management of CXL components is facilitated by a Component Command Interface (CCI). A CCI represents a command target that is used to process management and configuration commands that are issued to the component.

Table 8-37, Table 8-126, and Table 8-215 define the commands that a CCI can support.

A component can implement multiple CCIs of varying types that operate independently of one another and that have a uniquely defined list of supported commands. There are 2 types of CCIs:

- Mailbox Registers: A component can expose up to 2 mailboxes through its Mailbox registers, as defined in Section 8.2.8.4. Each mailbox represents a unique CCI instance.
- MCTP-based CCIs: Components with MCTP-capable interconnects can expose up to 1 CCI per interconnect. There is a 1:1 relationship between the component's MCTP-based CCIs and MCTP-capable interconnects. Transfer of commands via MCTP uses the transport protocol defined in Section 7.6.3.

All CCIs shall comply with the properties described in Section 9.20.1.

9.20.1

CCI Properties

Components that implement more than one CCI shall process commands from those CCIs in a manner that avoids starvation so that commands submitted to one CCI do not prevent commands from other CCIs from being handled. The exact algorithm for accepting commands from multiple CCIs is implementation specific. Each CCI within a component reports its supported command list through the Command Effects Log (CEL), as described in Section 8.2.9.5.2.1.

Interface-specific properties of commands, background operation, and timeouts are defined in Section 8.2.8.4 for mailbox CCIs and in Section 9.20.2 for MCTP-based CCIs. Each CCI can support the execution of only one background command at a time.

When a command is successfully started as a background operation, the component shall return the Background Command Started return code defined in Section 8.2.8.4.5.1. While the command is executing in the background, the component should update the percentage complete at least once per second.

A component may return the Busy return code if a command is sent to initiate a Background Operation while a Background Operation is already running on any other CCI.

An ongoing background command may be aborted by issuing a Request Abort Background Operation command on the same CCI (see Section 8.2.9.1.5).

Each CCI within a component shall maintain a unique context with respect to the following capabilities:

- Events, including reading contents, clearing entries, and configuring interrupt settings, and
- CEL content

IMPLEMENTATION NOTE

It is recommended that components with multiple CCIs that support commands that run as Background Operations only advertise support for those commands on one CCI.

Coordination between management entities attempting concurrent commands over separate CCIs that have component-level impact (e.g., FW update, etc.) is beyond the scope of this specification.

9.20.2**MCTP-based CCI Properties**

The CCI command timeout is 2 seconds, measured from when the command has been received by the component to when the component has started to transmit its response. Components should respond within this time limit; otherwise, requesters may timeout. Requesters must account for round-trip transmission time in addition to the command timeout.

IMPLEMENTATION NOTE

MCTP-based CCIs are intended to provide a dedicated management interface that operates independently from the state of any of the component's CXL interfaces; it is strongly recommended, but not required, that commands initiated on MCTP-based CCIs are not interrupted by Conventional Resets or any other changes of state of a component's CXL interface(s).

MCTP-based CCIs report background operation status using the Background Operation Status command as defined in [Section 8.2.9.1.2](#).

In the event of a command timeout, the requester may retransmit the request. New Message Tags shall be used every time that a request is retransmitted. Requesters may discard responses that arrive after the command timeout period has lapsed.

Commands sent to MCTP-based CCIs on MLD components are processed by the FM-owned LD.

§ §

10.0 Power Management

10.1 Statement of Requirements

All CXL implementations are required to support Physical Layer Power management as defined in this chapter. CXL Power management is divided into protocol-specific Link Power management and CXL Physical Layer power management. The ARB/MUX Layer is also responsible for managing protocol-specific Link Power Management between the Protocols on both sides of the links. The ARB/MUX coordinates the Power Management states between Multiple Protocols on both sides of the links, consolidates the Power states, and drives the Physical Layer Power Management.

10.2 Policy-based Runtime Control - Idle Power - Protocol Flow

10.2.1 General

For CXL-connected devices, there is a need to optimize power management of the entire system, with the device included.

As such, a hierarchical power-management architecture scheme is defined, where the discrete device is viewed as a single autonomous entity, with thermal and power management executed locally, but in coordination with the processor. Vendor-defined Messages (VDMs) over CXL are used to coordinate state transitions with the processor. The coordination between the primary power management controller on the host and the device is best accomplished via PM2IP and IP2PM messages that are encoded as VDMs.

Since native support of PCIe* is also required, support of more-simplified protocols is also possible. [Table 10-1](#) highlights the required and recommended handling method for Idle transitions.

Table 10-1. Runtime-Control - CXL vs. PCIe Control Methodologies

Case	PCIe	CXL ¹
Pkg-C Entry/Exit	Devices that do not share coherency with CPU can work with the PCIe profile: <ul style="list-style-type: none"> • LTR-notifications from Device • Allow-L1 signaling from CPU on Pkg_C entry 	Optimized handshake protocol, for all non-PCIe CXL profiles: <ul style="list-style-type: none"> • LTR-notifications from Device • PMreq/Rsp (VDM) signaling between CPU and device on Pkg_C entry and exit²

1. All CXL components support PM VDMs and use PM Controller - PM Controller sequences where possible.

2. PM2IP: VDMs are associated with different Reset/PM flows.

10.2.2 Package-level Idle (C-state) Entry and Exit Coordination

At a high level, a discrete CXL device that is coherent with the processor is treated like another processor package. The expectation is that there is coordination and agreement between the processor and the discrete device before the platform can enter idle power state. Neither the device nor the processor can individually enter a low-power state as long as its memory resources are needed by the other components.

For example, in a case where the device may contain shared High-Bandwidth memory (HBM), while the processor controls the system's DDR, if the device wants to be able to enter a low-power state, the Device must take into account the processor's need for accessing the HBM. Likewise, if the processor wants to enter a low-power state, the processor must take into account, among other things, the need for the device to access DDR. These requirements are encapsulated in the LTR requirements that are provided by entities that need QoS for memory access. In this case, we would have a notion of LTR for DDR access and LTR for HBM access. We would expect the device to inform the processor about its LTR with regard to DDR, and the processor to inform the device about its LTR with regard to HBM.

Latency requirements can be managed by using either of the following two methods:

- CXL devices that do not share coherency with the CPU (i.e., either a shared coherent memory or a coherent cache), can notify the processor of changes in its latency tolerance via the PMReq() and PMRsp() messages. When appropriate latency is supported and the processor execution has stopped, the processor will enter an Idle state and proceed to transition the Link to L1 (see [Link-Layer Section 10.3](#)).
- CXL devices that include a coherent cache or memory device are required to coordinate their state transitions using the CXL-optimized, VDM-based protocol, which includes the ResetPrep(), PMReq(), PMRsp(), and PMGo() messages, to prevent memory coherency loss.

10.2.2.1 PMReq Message Generation and Processing Rules

The rules associated with generation and processing of PMReq.Req, PMReq.Rsp, and PMReq.Go messages are as follows:

- A CXL device communicates its latency tolerance via a PMReq.Req message. A host communicates its latency tolerance either via a PMReq.Rsp message or a PMReq.Go message.
- A CXL device is permitted to unilaterally generate a PMReq.Req message as long as the Device has the necessary credits. A host shall not generate a PMReq.Req message.
- A CXL device shall not generate a PMReq.Rsp message. A host is permitted to unilaterally generate a PMReq.Rsp message as long as the Host has the necessary credits, even if the Host has never received a PMReq.Req message. A CXL device must process a PMReq.Rsp message normally, even if that CXL device has never previously issued a PMReq.Req message.
- A CXL device is not permitted to generate a PMReq.Go message. A host is permitted to unilaterally generate a PMReq.Go message as long as the Host has the necessary credits, even if the Host has never received a PMReq.Req message. A CXL device must process a PMReq.Go message normally, even if that CXL device has never:
 - Previously issued a PMReq.Req message.
 - Received a PMReq.Rsp message.
- A CXL device must continue to operate correctly, even if the device never receives a PMReq.Rsp in response to the device generating a PMReq.Req.
- A CXL device must continue to operate correctly, even if the device never receives a PMReq.Go in response to the device generating PMReq.Req.
- The Requirement bit associated with the non-snoop Latency Tolerance field in the PMReq messages must be cleared to 0 by all non-eRCD components.

[Section 10.2.3](#) and [Section 10.2.4](#) include example flows that illustrate these rules.

Table 10-2. PMReq(), PMRsp(), and PMGo() Encoding

Message	PM Logical Opcode[7:0]	Parameter[15:0]
PMReq.Req, abbreviated as PMReq	04h	0001h
PMReq.Rsp, abbreviated as PMRsp	04h	0000h
PMReq.Go, abbreviated as PMGo	04h	0004h or 0005h

10.2.3 PkgC Entry Flows

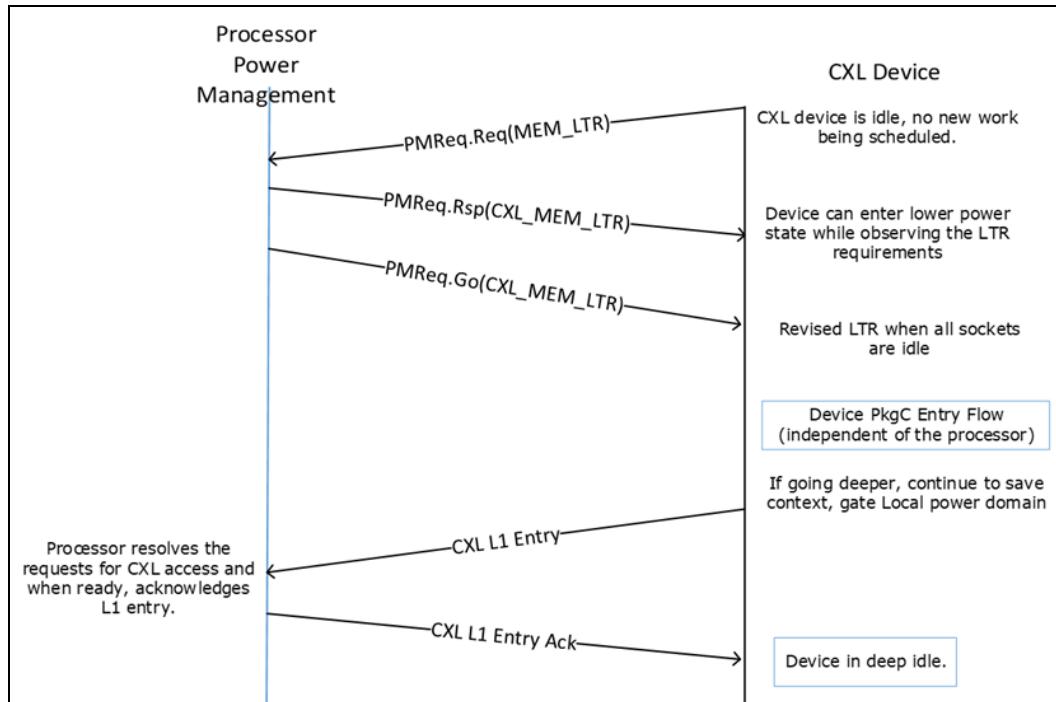
Figure 10-1. PkgC Entry Flow Initiated by Device - Example

Figure 10-1 illustrates the PkgC entry flow. When a Device needs to enter a higher-latency Idle state, in which the CPU is not active, the Device will issue a PMReq.Req with the LTR field marking the memory-access tolerance of the entity. As specified in Section 10.2.2.1, a device may unilaterally generate PMReq.Req to communicate any changes to its latency, without any dependency on receipt of a prior PMReq.Rsp or PMReq.Go. Specifically, a device may transmit two PMReq.Req messages without an intervening PMReq.Rsp from the host. The LTR value communicated by the device is labeled MEM_LTR, and represents the Device's latency tolerance regarding CXL.cache accesses and it could be different from what is communicated via LTR messages over CXL.io.

If Idle state is allowed, the processor will respond with a matching PMReq.Rsp message, with the negotiated allowable latency-tolerance LTR (labeled CXL_MEM_LTR). Both entities can independently enter an Idle state without coordination as long as the shared resources remain accessible.

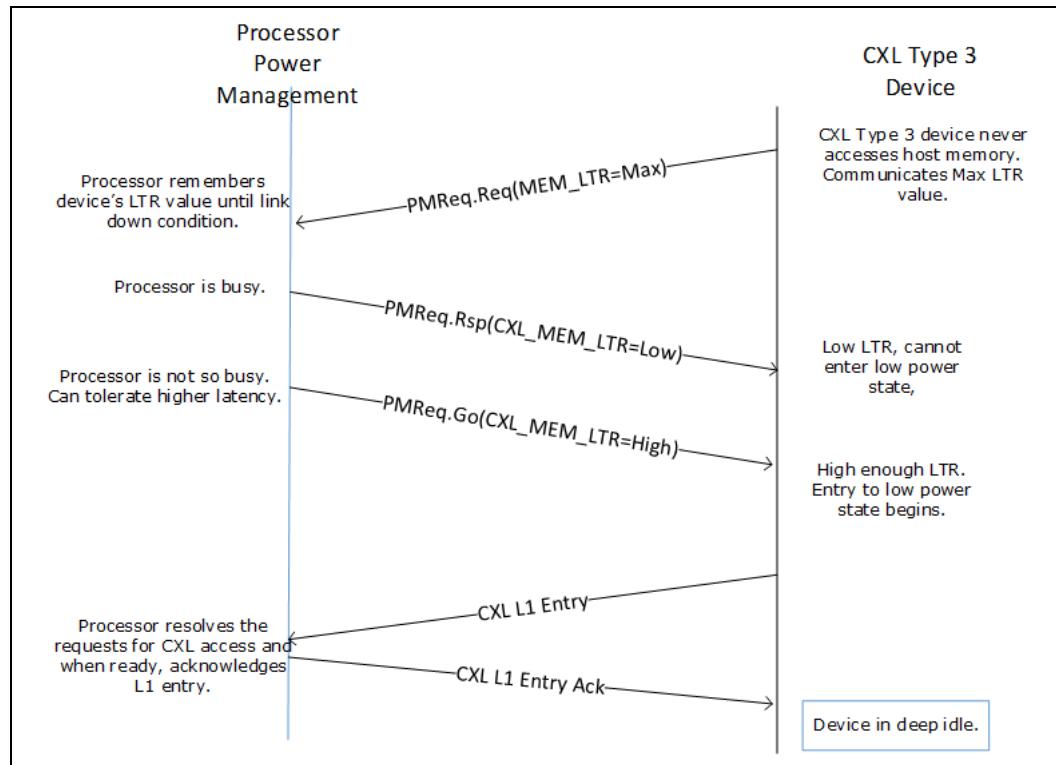
For a full PkgC entry, both entities need to negotiate as to the depth/latency tolerance by responding with a PMReq.Rsp message that includes the agreeable latency tolerance. After the master power management agent has coordinated LTR across all

the agents within the system, the agent will send a PMReq.Go() with the correct Latency field set (labeled CXL_MEM_LTR), indicating that local idle power actions can be taken subject to the communicated latency-tolerance value.

In case of a transition into deep-idle states, mostly typical of client systems, the device will initiate a CXL transition into L1.

These diagrams represent sequences, but do not imply any timing requirements. A host may respond much later with a PMReq.Rsp to a PMReq.Req from a device when the Host is ready to enter a low-power state, or the Host may not respond at all. A device, having sent a PMReq.Req, shall not implement a timeout to wait for PMReq.Rsp or PMReq.Go. Similarly, a device is not required to reissue PMReq.Req if the Device's latency-tolerance requirements have not changed since previous communication and the link has remained up. As shown in [Figure 10-2](#), a CXL Type 3 device may issue PMReq.Req after the link is up to indicate to the host that the Device either has no latency requirements or has a high latency tolerance. The host may communicate any changes to its latency expectations to such a device. Such a device may initiate low-power entry based only on the latency-tolerance value that the Device receives from the host, as shown in [Figure 10-2](#). When the host communicates a sufficiently high latency-tolerance value to the device, the device may enter a low-power state. A CXL Type 3 device may enter and exit a low-power state based only on the PMReq.Go message that the Device received from the host, without dependency on a prior PMReq.Rsp.

Figure 10-2. PkgC Entry Flows for CXL Type 3 Device - Example



10.2.4 PkgC Exit Flows

Figure 10-3. PkgC Exit Flows - Triggered by Device Access to System Memory

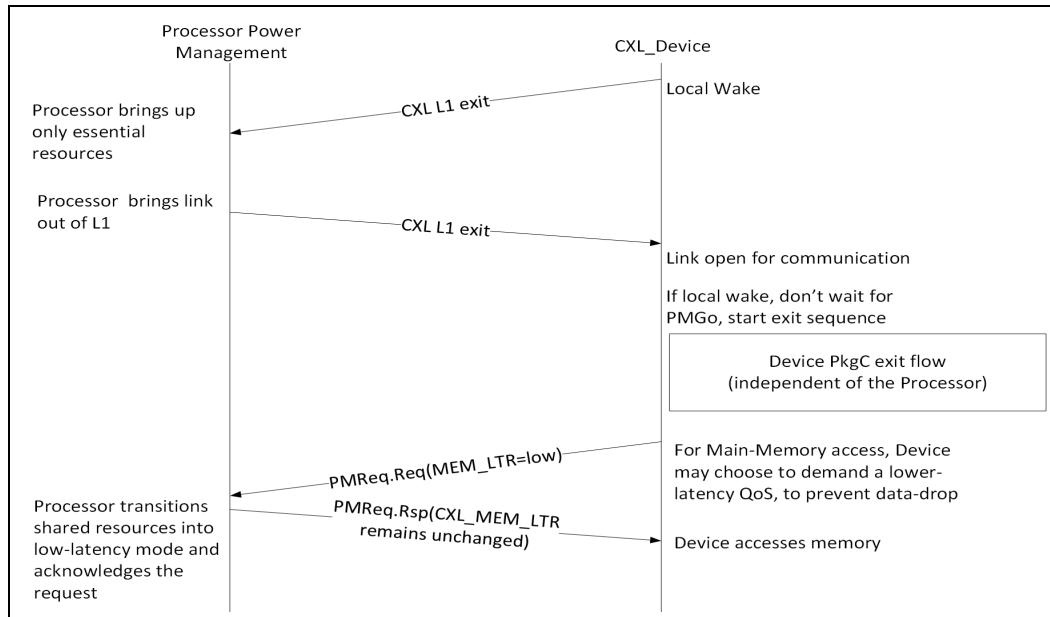


Figure 10-3 illustrates the PkgC exit flow initiated by the device. Link state during Idle may be in one of the select L1.x states, during Deep-Idle (as depicted here). In-band wake signaling will be used to transition the link back to L0. For more details, see Section 10.3.

After the CXL link exits L1, signaling can be used to transfer the device into a PkgC state, in which shared resources are available across CXL. The device requests a low-latency tolerance value to the processor. Based on that value, the processor will bring the shared resources out of Idle and communicate its latest latency requirements with a PMReq.Rsp().

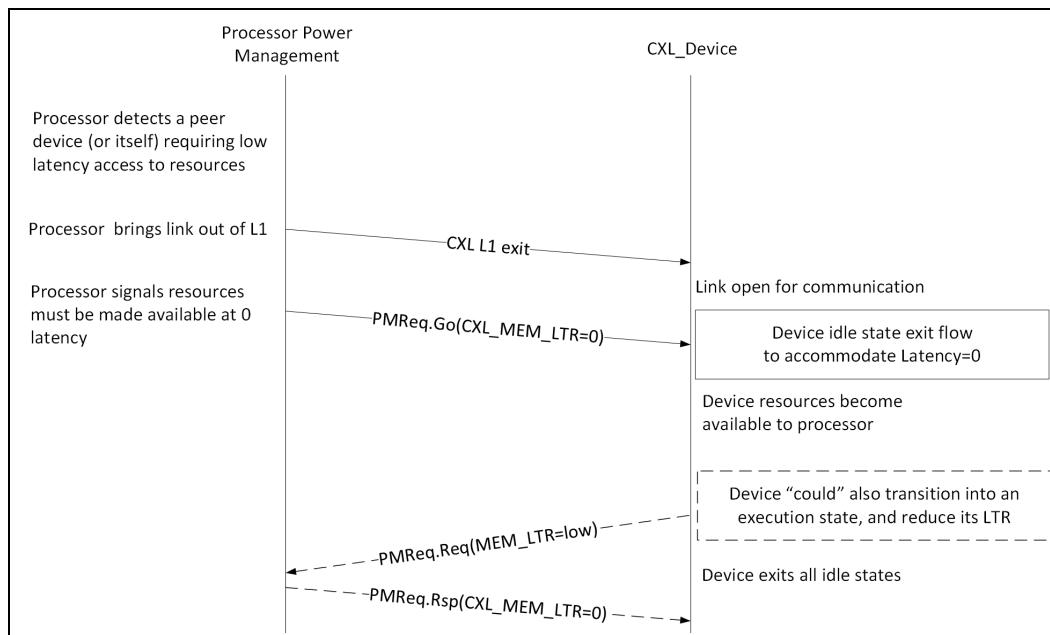
Figure 10-4. PkgC Exit Flows - Execution Required by Processor

Figure 10-4 illustrates the PkgC exit flow initiated by the processor. In the case where the processor, or one of the peer devices connected to the processor, must have coherent low-latency access to system memory, the processor will initiate a Link L1 exit toward the device.

After the link is running, the processor will follow with a PMGo(Latency=0), indicating some device in the platform requires low-latency access to coherent memory and resources. A device that receives PMReq.Go with Latency=0 must ensure that further low-power actions that might impede memory access are not taken.

10.2.5

CXL Physical Layer Power Management States

CXL Physical layer supports L1 and L2 states as defined in PCIe Base Specification. CXL Physical Layer does not support L0s. The entry and exit conditions from these states are also as defined in PCIe Base Specification. The notable difference is that for CXL Physical Layer, entry and exit from Physical Layer Power Management states is directed by the CXL ARB/MUX.

10.3

CXL Power Management

CXL Link Power Management supports Active Link State Power Management (ASPM), and L1 and L2 are the only 2 Power states supported. For 256B Flit mode, L0p negotiation is also supported. The PM Entry/Exit process is further divided into 3 phases as described below.

For 68B Flit mode, if the LTSSM goes through Recovery before the ARB/MUX vLSM moves to PM state, then the PM Entry process must restart from Phase 1, if the conditions for PM entry are still met after exit from Recovery and ARB/MUX Status Synchronization Protocol. For 256B Flit mode, the PM entry handshakes are not impacted by Link Recovery transitions because Link Recovery is not forwarded to the ARB/MUX vLSMs.

10.3.1 CXL PM Entry Phase 1

CXL PM Entry Phase 1 involves protocol-specific mechanisms to negotiate entry into a supported PM state. As shown in Figure 10-5, in 256B Flit mode, this transition does not require any synchronization between the ARB/MUX instances on the two ends. 68B Flit mode, however, does require such synchronization (see Figure 10-6). After the conditions to enter the PM state as defined in Section 10.2 are satisfied, the Transaction Layer is ready for Phase 2 entry and directs the ARB/MUX to enter the PM State.

Figure 10-5. CXL Link PM Phase 1 for 256B Flit Mode

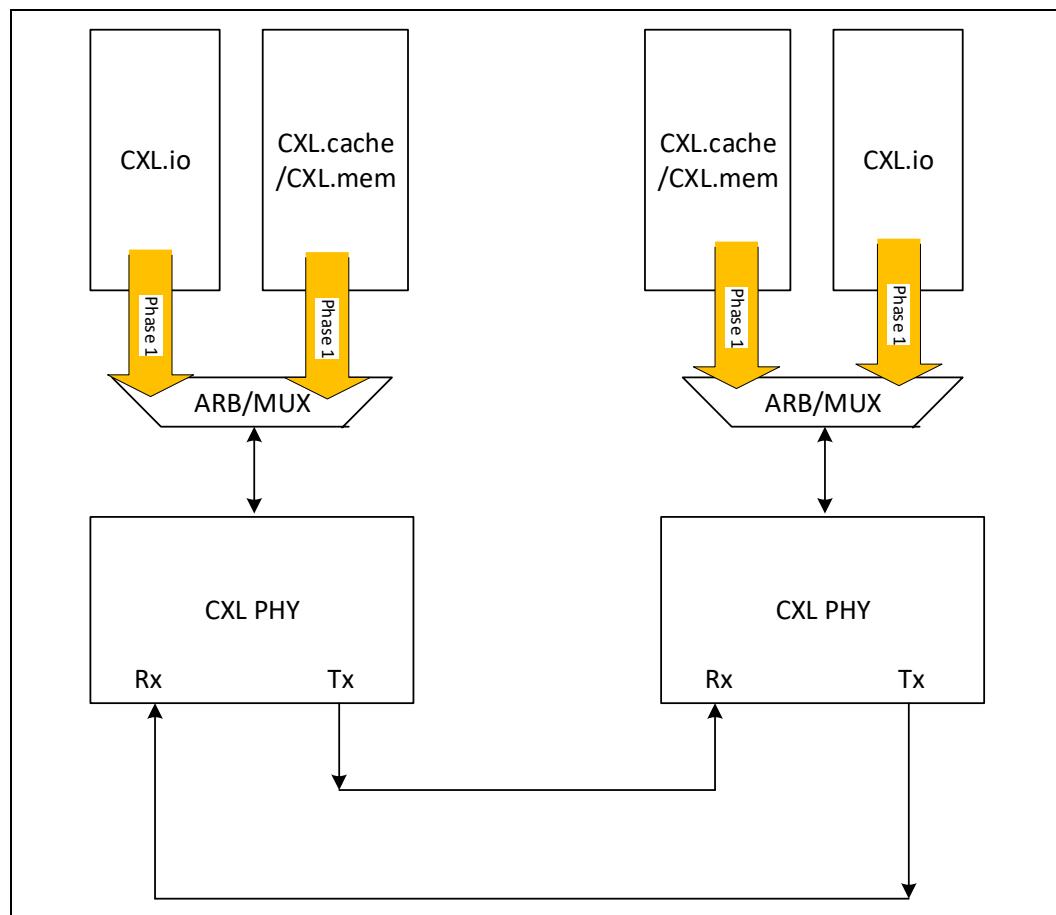
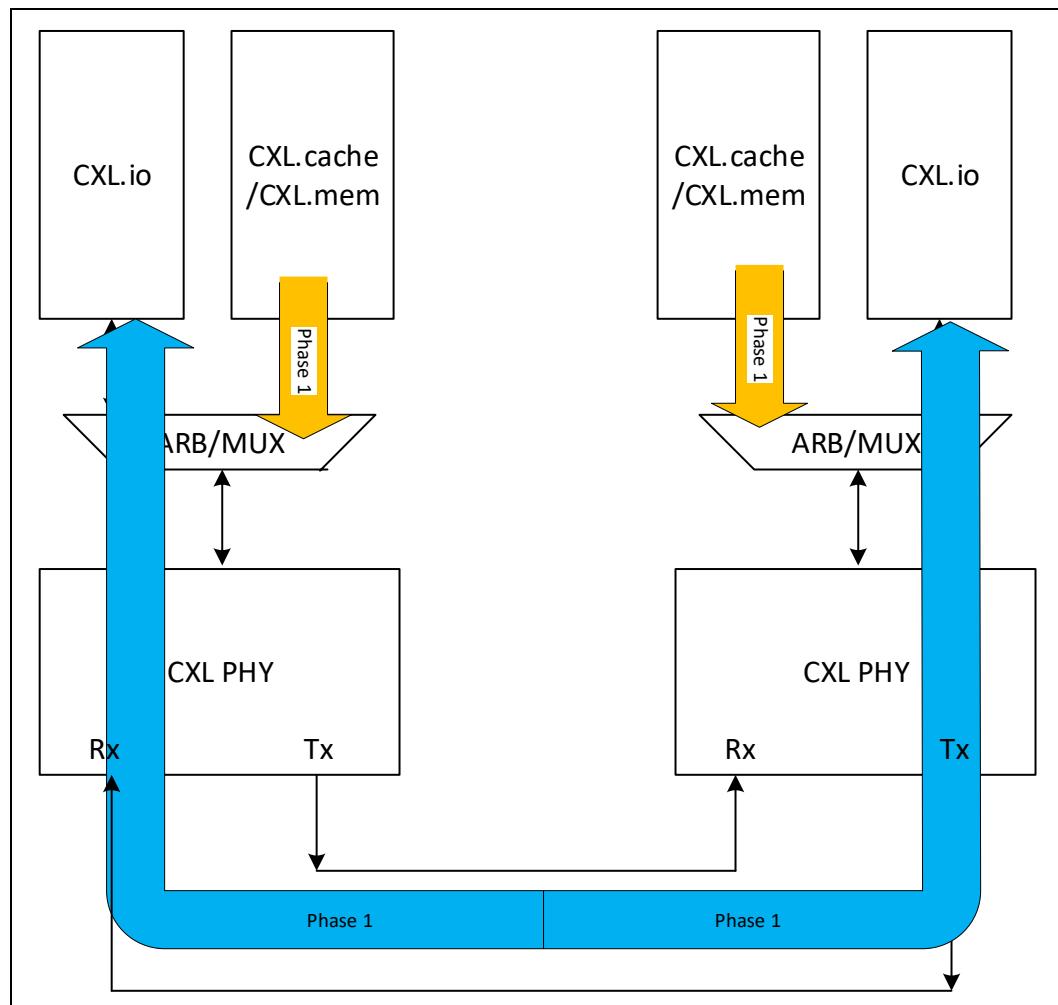


Figure 10-6. CXL Link PM Phase 1 for 68B Flit Mode

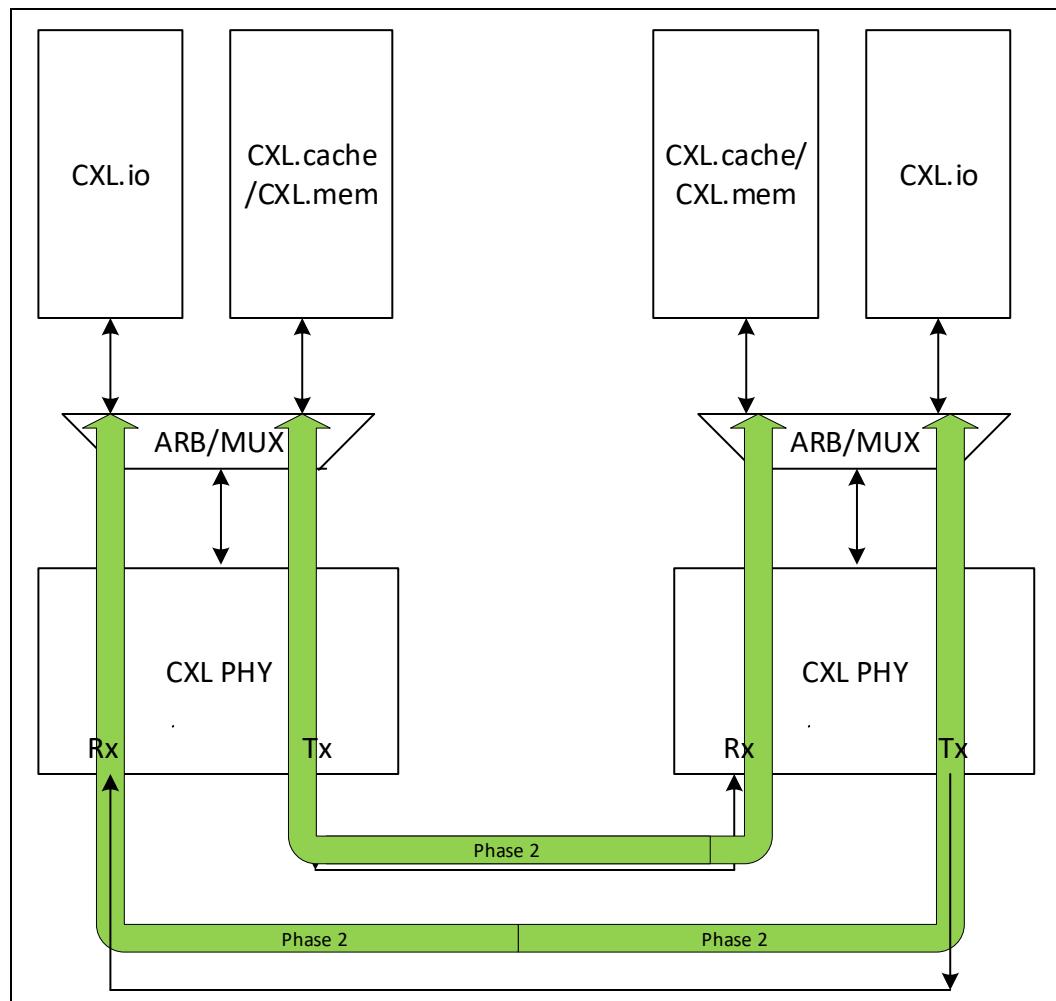
10.3.2

CXL PM Entry Phase 2

When directed by the Transaction Layer to enter PM, the ARB/MUX initiates the CXL PM Entry Phase 2 process. Phase 2 consists of bringing the ARB/MUX interface of both sides of the Link into a supported PM state. ALMPs are used to coordinate PM state entry as described below. Phase 2 entry is independently managed for each protocol. The Physical Layer continues to be in L0 until all the Transaction Layers enter Phase 2 state.

EVALUATION COPY

Figure 10-7. CXL Link PM Phase 2



Rules for the Phase 2 entry into ASPM are as follows (summarized in Figure 10-7):

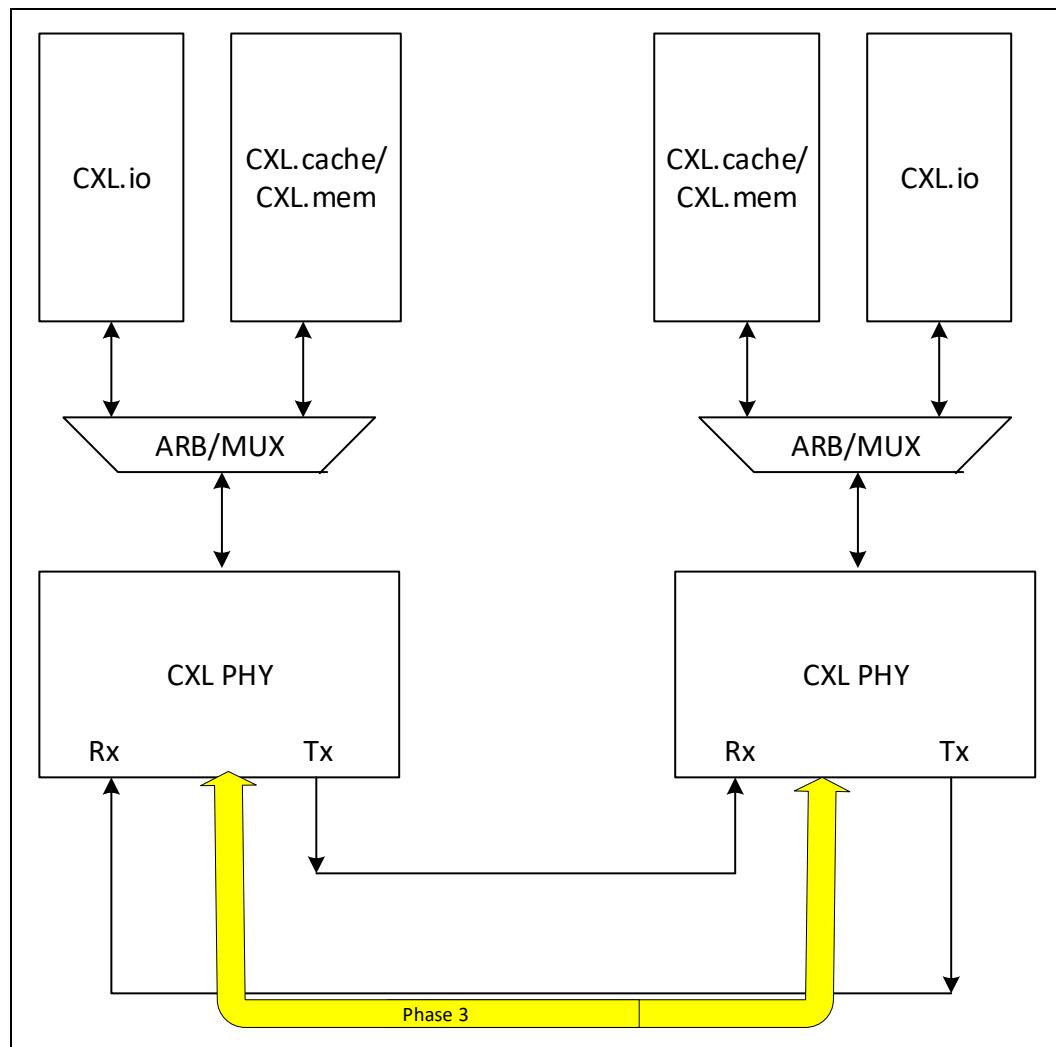
1. Phase 2 Entry into the supported PM State is always initiated by the ARB/MUX on the Downstream Component.
2. When directed by the Transaction Layer, the ARB/MUX on the Downstream Component must transmit an ALMP request to enter vLSM state PM.
3. When the ARB/MUX on the Upstream Component is directed to enter L1 and receives an ALMP request from the Downstream Component, the Upstream Component responds with an ALMP response indicating acceptance of entry into L1 state. The Transaction Layer on the Upstream Component must also be notified that the ARB/MUX port has accepted entry into the supported PM state.
4. The Upstream Component ARB/MUX port does not respond with an ALMP response if not directed by the upper layers to enter PM state.
5. When the ARB/MUX on the Downstream Component is directed to enter L1 and receives an ALMP response from the Upstream Component, the ARB/MUX notifies acceptance of entry into the PM state to the Transaction Layer on the Downstream Component.

6. The Downstream Component ARB/MUX port must wait for at least 1 ms (not including time spent in recovery states) for a response from the Upstream Component before retrying PM entry. The Downstream Component ARB/MUX is permitted to abort the PM entry before the 1-ms timeout by sending an Active Request ALMP for the corresponding vLSM.
7. L2 entry is an exception to Rule 6. Protocol must ensure that the Upstream Component is directed to enter L2 before setting up the conditions for the Downstream Component to request L2 state entry. This ensures that L2 abort or L2 Retry conditions do not exist. The Downstream Component may use indications such as the PME_Turn_Off message or a RESETPREP VDM to trigger L2 state entry.
8. The Transaction Layer on either side of the Link is permitted to directly exit from L1 state after the ARB/MUX interface enters L1 state.

10.3.3

CXL PM Entry Phase 3

CXL PM Entry Phase 3 is a conditional phase of PM entry and is executed only when all the Protocol interfaces of ARB/MUX have entered the same virtual PM state. The phase consists of bringing the Tx lanes to electrical idle and is always initiated by the Downstream Component. As shown in [Figure 10-8](#), the PHY Layers on the two ends of the link communicate. If the link transitions to recovery during or after entry into electrical idle, the Downstream Component must wait for at least 1 us after entering L0 before re-initiating entry into electrical idle. This is to allow sufficient time for an Active State Request ALMP transfer to occur in case either side wants to initiate a PM exit (and to provide sufficient time for the remote ARB/MUX to stop requesting PM entry to LogPHY). The electrical idle entry flow is defined in the “Power Management” chapter of PCIe Base Specification.

Figure 10-8. CXL PM Phase 3**10.3.4****CXL Exit from ASPM L1**

Components on either end of the Link may initiate exit from the L1 Link State. The ASPM L1 exit depends on whether the exit is from Phase 3 or Phase 2 of L1. The exit is hierarchical and Phase 3 must exit before Phase 2.

Phase 3 exit is initiated when directed by the ARB/MUX from either end of the link. The ARB/MUX Layer initiates exit from Phase 3 when there is an exit requested on any one of its primary protocol interfaces. The Phase 3 ASPM L1 exit is the same as exit from L1 state as defined in PCIe Base Specification. The steps are followed until the LTSSM enters L0 state. Protocol-level information is not permitted to be exchanged until the vLSM on the ARB/MUX interface has exited L1 state.

Phase 2 exit involves bringing the protocol interface independently out of L1 state at the ARB/MUX. The Transaction Layer directs the ARB/MUX state to exit vLSM state. If the PHY is in Phase 3 L1, then the ARB/MUX waits for the PHY LTSSM to enter L0 state. After the PHY is in L0 state, the following rules apply:

1. The ARB/MUX on the protocol side that is triggering an exit transmits an ALMP requesting entry into Active state.
2. Any ARB/MUX interface that receives the ALMP request to enter Active State must transmit an ALMP acknowledge response on behalf of that interface. The ALMP acknowledge response is an indication that the corresponding protocol side is ready to process received packets.
3. Any ARB/MUX interface that receives the ALMP request to enter Active State must also transmit an ALMP Active State request on behalf of that interface if not already sent.
4. Protocol-level transmission must be permitted by the ARB/MUX after an Active State Status ALMP is transmitted and received. This guarantees that the receiving protocol is ready to process packets.

10.3.5

L0p Negotiation for 256B Flit Mode

See [Chapter 5.0](#) for the L0p negotiation rules.

10.4

CXL.io Link Power Management

CXL.io Link Power Management is as defined in PCIe Base Specification with the following notable differences:

- RCD links support ASPM-directed L1 entry but do not support PCI-PM-directed L1 entry. An eRCD is not required to initiate entry into L1 state when software transitions the device into D3Hot or D1 device state. When a component is not operating in RCD mode, the component shall support ASPM L1 as well as PCI-PM. As such, a component not operating in RCD mode shall initiate CXL.io L1 entry when the device is placed in D3Hot or D1 device state.
- L0s state is not supported.

All CXL functions shall implement PCI* Power Management Capability Structure as defined in PCIe Base Specification and shall support D0 and D3 device states.

10.4.1

CXL.io ASPM Entry Phase 1 for 256B Flit Mode

There must not be any DLLP exchanges initiated for PM entry for 256B Flit mode. The Link Layer on each side independently requests its local ARB/MUX to enter a PM state. The ARB/MUX Layers on both sides of the Link coordinate entry into a PM state using ALMPs as part of Phase 2.

10.4.2

CXL.io ASPM L1 Entry Phase 1 for 68B Flit Mode

The first phase consists of completing the ASPM L1 negotiation rules as defined in PCIe Base Specification with the following notable exception for the rules in case of acceptance of ASPM L1 Entry:

- All rules up to the completion of the ASPM L1 handshake are maintained; however, the process of bringing the Transmit Lanes into Electrical Idle state are divided into 2 additional phases described in [Section 10.3](#).

See PCIe Base Specification for the PCIe ASPM L1 Entry flow.

10.4.3 CXL.io ASPM L1 Entry Phase 2

Phase 2 of L1 entry consists of bringing the CXL.io ARB/MUX interface of both sides of the Link into L1 state. ALMPs are used to coordinate L1 state entry. For 256B Flit mode, the ALMP exchange rules are the same for CXL.io and CXL.cachemem, and are defined in [Chapter 5.0](#).

The rules for Phase 2 entry into ASPM L1 for 68B Flit mode are as follows:

1. CXL.io on the Upstream Component must direct the ARB/MUX to be ready to enter L1 before returning the PM_Request_Ack DLLPs as shown above in Phase 1.
2. When the PM_Request_Ack DLLPs are successfully received by the CXL.io on the Downstream Component, the CXL.io must direct the ARB/MUX on the Downstream Component to transmit the ALMP request to enter vLSM state L1.
3. When the ARB/MUX on the Upstream Component is directed to enter L1 and receives an ALMP request from the Downstream Component, the ARB/MUX notifies the CXL.io that the interface has received an ALMP request to enter L1 state and has entered L1 state.
4. When the Upstream Component is notified of the vLSM state L1 entry, the Upstream Component ceases sending PM_Request_Ack DLLPs.
5. When the ARB/MUX on the Downstream Component is directed to enter L1 and receives ALMP Status from the Upstream Component, the ARB/MUX notifies the CXL.io that the interface has entered L1 state.

10.4.4 CXL.io ASPM Entry Phase 3

Phase 3 entry is dependent on the vLSM state of multiple protocols and is managed by the ARB/MUX as described in [Section 10.3.3](#).

10.5 CXL.cache + CXL.mem Link Power Management

CXL.cache and CXL.mem both support only ASPM. Unlike CXL.io, there is no PM Entry handshake defined between the Link Layers. Each side independently requests the ARB/MUX to enter L1. The ARB/MUX Layers on both sides of the Link coordinate the entry into a PM state using ALMPs. CXL.cache + CXL.mem Link Power Management follows the process for PM entry and exit as defined in [Section 10.3](#).

§ §

11.0 CXL Security

11.1 CXL IDE Overview

CXL Integrity and Data Encryption (CXL IDE) defines mechanisms for providing Confidentiality, Integrity, and Replay protection for data that traverses the CXL link. The cryptographic schemes are aligned with current industry best practices. CXL IDE supports a variety of usage models while providing for broad interoperability. CXL IDE can be used to secure traffic within a Trusted Execution Environment (TEE) that is composed of multiple components (see [Section 11.5](#)).

This chapter focuses on the changes for CXL.cache and CXL.mem traffic that traverses the link, and updates and constraints to PCIe* Base Specification that govern CXL.io traffic.

- CXL.io IDE definition including CXL.io IDE key establishment is based on PCIe IDE. Differences and constraints for CXL.io usage are identified in [Section 11.2](#).
- CXL.cachemem IDE may use the CXL.io-based mechanisms for discovery, negotiation, device attestation, and key negotiation using a standard mechanism as described in [Section 11.4](#).

In this specification, the term CXL IDE is used to refer to the scheme that protects CXL.io, CXL.cache, and CXL.mem traffic. The term CXL.cachemem IDE is used to refer to the protections associated with CXL.cache and CXL.mem traffic.

IMPLEMENTATION NOTE: SECURITY MODEL

Assets

Assets that are in scope are as follows:

- Transactions (data + metadata communicated) between the two sides of the physical link. Only the definition for providing integrity, replay protection and encryption/decryption of traffic between the ports is included in this specification.

Notes:

- This threat model does not cover the security exposure due to inadequate Device implementation.
- Agents that are on each side of the physical link are within the trust boundary of the respective devices/hardware blocks in which they reside. These agents will need to provide implementation-specific mechanisms to protect data internal to the device and any external connections over which such data can be sent by the device. Mechanisms for such protection are beyond the scope of this definition.
- Symmetric cryptographic keys are used to provide confidentiality, integrity, and replay protection of data in transit between physically connected CXL ports. This specification does not define mechanisms for protecting these keys inside the host and the device.
- Certificates and asymmetric keys that are used for device authentication and key exchange are beyond the scope of this specification. The device attestation and key exchange mechanism determine the security model for those assets.

TCB

The TCB consists of the following:

- Functional blocks on each side of the link that implement the link encryption and integrity.
- Agents that are used to configure the cryptographic engines in the functional blocks on each side of link. For example, trusted firmware/software agent and/or security agent hardware and firmware that implement key exchange protocol or facilitate programming of the keys.
- Other hardware blocks in the device that may have direct or indirect access to the assets, including those that perform operations such as reset, debug, and link power management.

Adversaries and Threats

- Threats from physical attacks on links, including cases where an adversary can examine data intended to be confidential, modify data or protocol metadata, record and replay recorded transactions, reorder and/or delete data flits, inject transactions including requests/data or non-data responses, using lab equipment, purpose-built interposers, and/or malicious Extension Devices.
- Threats arising from physical replacement of a trusted device with an untrusted device, and/or removal of a trusted device and accessing the trusted device with a system that is under an adversary's control.
- CXL.cachemem IDE provides point-to-point protection. Any switches present in the path between the Host and the Endpoint, or between two Endpoints, must support this specification. In these cases, such switches will be in the TCB.

Denial of service attacks are beyond the scope of this specification.

11.2 CXL.io IDE

CXL.io IDE follows the PCIe IDE definition. This section covers the notable constraints and differences between the CXL.io IDE definition and the PCIe IDE definition.

Table 11-1. Mapping of PCIe IDE to CXL.io

PCIe IDE Definition	CXL.io Support	Notes
Link IDE stream	Supported	Required for CXL.cachemem IDE. CXL.cachemem will only use keys associated with Link IDE stream.
Selective IDE stream	Supported	Selective IDE stream applies only to CXL.io.
Aggregation	Supported	PCIe-defined aggregation levels apply only to CXL.io traffic.
Switches with flow-through selective IDE streams	Supported	CXL Switches need to support Link IDE streams. CXL Switches may either operate as a boundary for selective IDE streams or forward the IDE streams toward Endpoints.
PCRC mechanism	Supported	PCRC mechanism may be optionally enabled for the CXL.io ports.

One of the PCIe IDE reserved sub-stream encodings (1000b) is assigned for CXL.cachemem usage.

11.3 CXL.cachemem IDE

All protocol-level retryable flits are encrypted and integrity protected.

When operating in 68B Flit mode:

- Link Layer control flits and flit CRC are not encrypted or integrity protected. There is no confidentiality or integrity on these flits.
- Link CRC shall be calculated on encrypted flits. Link retries occur first and only flits that pass Link CRC will be decrypted and then integrity checked.

When operating in 256B Flit mode:

- Link Layer control information, flit header, and flit CRC/FEC is not encrypted or integrity protected. There is no confidentiality protection, integrity protection, or replay protection for this content.
- Link CRC shall be calculated on encrypted flits. Link retries occur first and only flits that pass Link CRC will be decrypted and then integrity checked.

Any integrity check failures shall result in all future secure traffic being dropped.

Multi-Data Header capability must be supported. This allows packing of multiple (up to 4) data headers into a single slot, followed immediately by 16 slots of all-data.

IDE will operate on a flit granularity for CXL.cache and CXL.mem protocols. IDE makes use of the Advanced Encryption Standard-Galois Counter Mode Advanced Encryption and Advanced Decryption Functions (referred to herein as AES-GCM), as defined in NIST* Special Publication 800-38D. AES-GCM with a 256-bit key size shall be used for confidentiality protection, integrity protection, and replay protection. The AES-GCM Functions take three inputs:

- additional authentication data (AAD; denoted as A)
- plaintext (denoted as P)
- initialization vector (denoted as IV)

Key refresh without any data loss must be supported. There are a number of scenarios where the keys need to be refreshed. Some examples include:

- An accelerator device that is migrated from one VM (or process) to a different VM (or process).
- Crypto considerations (concerns about key wear-out) for long-running devices or devices that are part of the platform.

Key refresh is not expected to occur frequently. It is acceptable to take a latency/bandwidth hit; however, there must not be any data loss.

Encrypted PCRC mechanism is supported to provide robustness against hard and soft faults that are internal to the encryption and decryption engines. Encrypted PCRC integrates into the standard MAC check mechanism, does not consume incremental link bandwidth, and can be implemented without adding significant incremental latency. PCRC is mandatory for CXL.cachemem IDE and is enabled by default.

11.3.1 CXL.cachemem IDE Architecture in 68B Flit Mode

IDE shall operate on a flit granularity for CXL.cachemem protocols. IDE makes use of the AES-GCM algorithm, and AES-GCM takes three inputs – A , P , and IV – as described earlier in [Section 11.3](#).

In the case of CXL.cachemem protocol header flits, the 32 bits of the flit header that are part of Slot 0 map to A – it is not encrypted, but it is integrity protected. The remainder of the Slot 0/1/2/3 contents maps to P , which is encrypted and integrity protected (see handling of MAC slot below). CXL.cachemem protocol also supports ADF. In the case of an ADF, all 4 slots in the flit map to P .

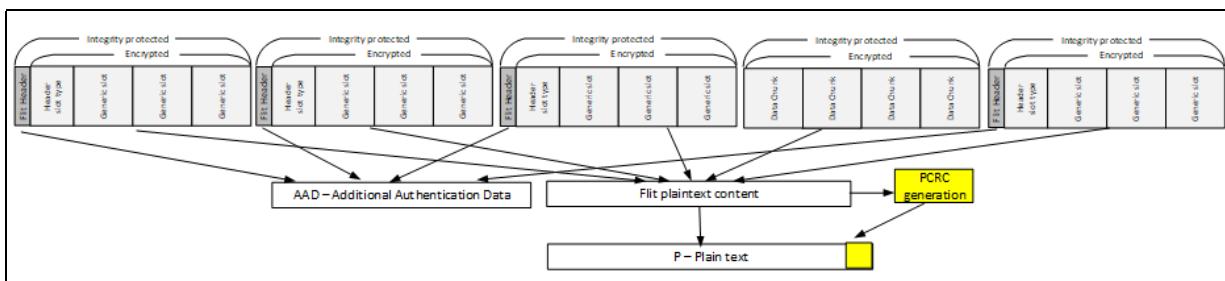
The link CRC is not encrypted or integrity protected. The CRC is calculated on the flit content after the flit has been encrypted.

As with other protocol flits, IDE flits shall be covered by link layer mechanisms for detecting and correcting errors. This process shall operate on flits after the flits are cryptographically processed by the transmitter and before the flits are submitted for cryptographic processing by the receiver.

AES-GCM is applied to an aggregation of multiple flits referred to as a MAC epoch. The number of flits in the aggregation is determined by the Aggregation Flit Count (see [Section 11.3.5](#) for details). If PCRC (see [Section 11.3.3](#)) is enabled in the CXL IDE Control register (see [Section 8.2.4.22.2](#)), the 32 bits of PCRC shall be appended to the end of the aggregated flit content to contribute to the final P value that is integrity protected. However, the 32 bits of PCRC are not transmitted across the link.

[Figure 11-1](#) shows the mapping of the flit contents into A and P for the case of aggregation of MAC across 5 flits.

Figure 11-1. 68B Flit: CXL.cachemem IDE Showing Aggregation of 5 Flits



EVALUATION COPY

The Message Authentication Code (MAC), also referred to as the authentication tag in NIST Special Publication 800-38D, shall be 96 bits. The MAC must be transmitted in a Slot 0 header of type H6 (see [Figure 4-12](#)). Unlike other Slot 0 headers, the MAC itself is neither encrypted nor integrity protected. [Figure 11-2](#) shows the mapping of flit contents to A and P for the case of aggregation of MAC across 5 flits with one of the flits carrying a MAC.

Figure 11-2. 68B Flit: CXL.cachemem IDE Showing Aggregation across 5 Flits where One Flit Contains MAC Header in Slot 0

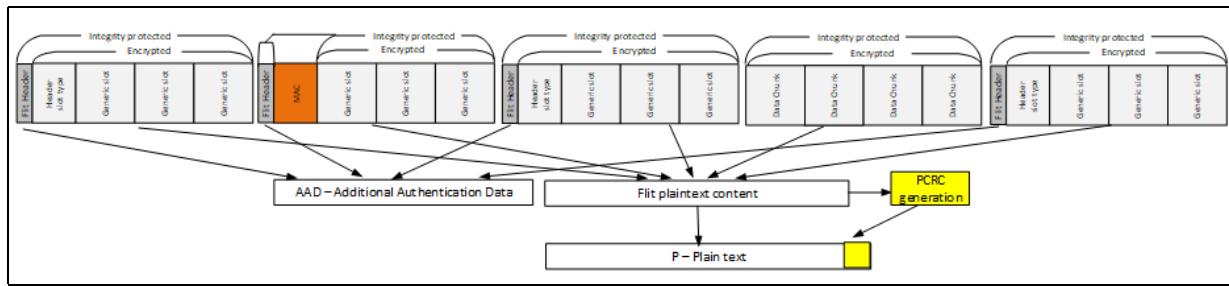
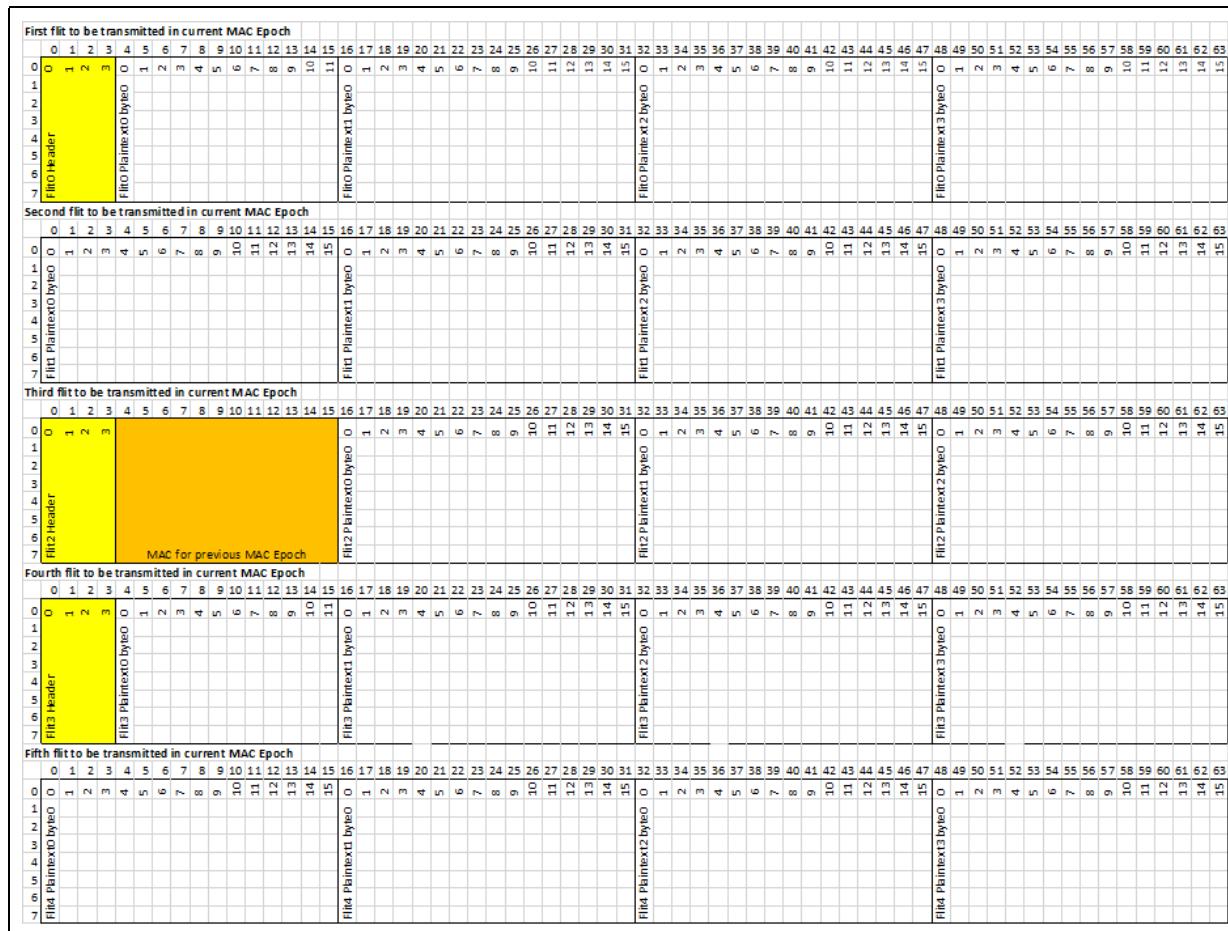


Figure 11-3 provides a more-detailed view of the 5-flit MAC epoch example. Flit0 shown on the top is the first flit to be transmitted in this MAC epoch. The figure shows the header fields that are only integrity protected, and plaintext content that is encrypted and integrity protected. Flit0 plaintext0 byte0 is the first byte of the plaintext. Flit1 plaintext0 byte0 shall immediately follow flit0 plaintext3 byte15.

Figure 11-3. 68B Flit: More-detailed View of a 5-Flit MAC Epoch Example



[Figure 11-4](#) shows the mapping of the header bytes to AES-GCM AAD (A) for the example in [Figure 11-3](#).

Figure 11-4. 68B Flit: Mapping of AAD Bytes for the Example Shown in Figure 11-3

AAD for current MAC Epoch																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
1																
2																
3																
4	Flit0 Header				Flit2 Header				Flit3 Header				Pad with zeros			
5																
6																
7																

11.3.2 CXL.cachemem IDE Architecture in 256B Flit Mode

If the header slot is used for sending control messages other than IDE.MAC, the entire flit shall not carry any protocol traffic. This applies for other usages of IDE type (IDE.TMAC, IDE.Start, and IDE.Idle), In-band Error, and INIT.

The receiver uses 4 bits of the header slot that encode the slot type to determine whether the slot contains control or protocol information. If the header slot is carrying protocol information, then 4 bits of the header slot that encode the slot type will map to AES-GCM input A. Although the slot type will not be encrypted, it is integrity protected. If the header slot is carrying control information, then the entire slot is neither encrypted nor integrity protected.

In case the header slot is carrying protocol information, then the plaintext (P) starts at bit 20. To simplify implementation and align to the AES block size of 128 bits, 20 bits of 0s shall be padded in front of the header slot content and the padded 128 bits of information shall be mapped to AES-GCM input P .

- The padded header slot will be used when calculating PCRC (see [Section 11.3.3](#)).
- Encrypted pad will not be transmitted on the link. Receiver must reconstruct the ciphertext for the padded region when calculating the AES-GCM MAC.

Credit return (CRD) field does not carry any confidential data. The CRD field needs to be integrity protected, so the CRD field shall map to AES-GCM input A.

The rules for handling latency-optimized flits are as follows:

- Slot 7 bytes shall be packed together before mapping to AES-GCM input P .
- Slot 8 is only 12 bytes long. It shall be padded with 32 bits of 0 at the end of slot content. This will enable subsequent slots to be aligned on the 128-bit AES block boundary.
- Packed Slot 7 and padded Slot 8 should be used when calculating PCRC (see [Section 11.3.3](#)).
- Receiver must reconstruct the ciphertext for the padded region in Slot 8 when calculating AES-GCM MAC.
- AES-GCM input A for each flit shall be padded with 0s to align it to 32 bits.
- Header slot contains protocol header: slot_type | CRD | 012.
- Header slot contains MAC: CRD | 016.

In the case of 256B flits, only Slot 0 and Slot 15 contribute to the AAD. [Figure 11-5](#), [Figure 11-6](#), [Figure 11-7](#), [Figure 11-8](#), [Figure 11-9](#), and [Figure 11-10](#) depict handling of the AAD field.

[Figure 11-5](#) depicts the case when Slot 0 contains LLCTRL (H8) slot format encoding with an IDE.MAC message. In this case, Slot 0 does not contain any bits that require integrity protection; therefore, Slot 0 is not IDE protected.

Figure 11-5. 256B Flit: Handling of Slot 0 when it Carries H8

bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	bytes
0			B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	
1																	
2																	
3	2B HDR																
4																	
5																	
6																	
7																	
Slot 0 contains LLCTRL (H8) slot format encoding with IDE.MAC message																	
No IDE protection																	
Integrity protection only (AES GCM AAD)																	
Encrypt and Integrity protect (AES-GCM P)																	

[Figure 11-6](#) shows the case where Slot 0 contains protocol header slot format encoding (H0 - H7, H9 - H15). The first 2 bytes that contain the flit header are not IDE protected. Bits 0 - 3 of Slot 0 that carry the slot format encoding are not encrypted, but are integrity protected and therefore map to the AAD field. The remaining bits of the slot are encrypted and integrity protected (additional details regarding mapping to the P field are provided in [Figure 11-11](#)).

Figure 11-6. 256B Flit: Handling of Slot 0 when it Does Not Carry H8

bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	bytes
0																	
1																	
2																	
3	2B HDR		b0-3	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	
4																	
5																	
6			b4-7														
7																	
Slot 0 contains protocol header slot format encodings																	
No IDE protection																	
Integrity protection only (AES GCM AAD)																	
Encrypt and Integrity protect (AES-GCM P) - detailed handling shown in subsequent figure																	

Handling of Slot 15 is shown in [Figure 11-7](#). When the flit carries protocol information, the CRD field carried in Slot 15 needs to be integrity protected. In this case, the first two bytes of CRD information (Credit return byte 0 and Credit return byte 1) map to the AES-GCM AAD field.

Figure 11-7. 256B Flit: Handling of Slot 15

bytes																							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15								
C0B0	C0B1																						
CRC (8B)								FEC (6B)															
No IDE protection																							
Integrity protection only (AES GCM AAD)																							
Encrypt and Integrity protect (AES-GCM P)																							

Figure 11-8 shows how the bits that only need to be integrity protected are mapped to the AAD field when the first flit carries a protocol header in Slot 0 and the second flit carries IDE.MAC in Slot 0.

Figure 11-8. Mapping of Integrity-only Protected Bits to AAD - Case 1

bit	Flit0 H8 MAC								Flit1 protocol header							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
0	C0b0	C1b0		0	0				C0b4	C1b4						0
1	C0b1	C1b1		0	0				C0b5	C1b5						0
2	C0b2	C1b2		0	0	b0-3			C0b6	C1b6						0
3	C0b3	C1b3		0	0				C0b7	C1b7						0
4	C0b4	C1b4		0	0	C0b0	C1b0			0	0					0
5	C0b5	C1b5		0	0	C0b1	C1b1			0	0					0
6	C0b6	C1b6		0	0	C0b2	C1b2			0	0					0
7	C0b7	C1b7		0	0	C0b3	C1b3			0	0					0
(b) First flit carries MAC in Slot 0																
No IDE protection																
Header bits for Integrity protection only (AES GCM AAD)																
Zero padding required by CXL.cachemem IDE. Include these bits in the computation of AAD length.																

Figure 11-9 shows how the bits that only need to be integrity protected are mapped to the AAD field when the first flit carries IDE.MAC in Slot 0 and the second flit carries a protocol header.

Figure 11-9. Mapping of Integrity-only Protected Bits to AAD - Case 2

bit	Flit1 H8 MAC								Flit1 protocol header							
	4	5	6	7	4	5	6	7	4	5	6	7	4	5	6	7
0	C0b0	C1b0		0	0				C0b4	C1b4						0
1	C0b1	C1b1		0	0	b0-3			C0b5	C1b5						0
2	C0b2	C1b2		0	0				C0b6	C1b6						0
3	C0b3	C1b3		0	0				C0b7	C1b7						0
4	C0b4	C1b4		0	0	C0b0	C1b0			0	0					0
5	C0b5	C1b5		0	0	C0b1	C1b1			0	0					0
6	C0b6	C1b6		0	0	C0b2	C1b2			0	0					0
7	C0b7	C1b7		0	0	C0b3	C1b3			0	0					0
First flit carries MAC in Slot 0																
No IDE protection																
Header bits for Integrity protection only (AES GCM AAD)																
Zero padding required by CXL.cachemem IDE. Include these bits in the computation of AAD length.																

Figure 11-10 shows the third case of how the bits that only need to be integrity protected are mapped to the AAD field. In this case, both flits carry protocol headers in Slot 0. When the flit carries a protocol header, there are 20 bits that require integrity protection. These 20 bits are made up of 4 bits of Slot encoding (Slot 0) and 16 bits of CRD (Slot 15). These are padded with trailing 0s to create a 32-bit AAD input.

Figure 11-10. Mapping of Integrity-only Protected Bits to AAD - Case 3

bit	Flit0 protocol header				Flit1 protocol header				bytes
	0	1	2	3	4	5	6	7	
0	C0b4	C1b4		0	C0b4	C1b4		0	
1	C0b5	C1b5		0	C0b5	C1b5		0	
2	C0b6	C1b6		0	C0b6	C1b6		0	
3	C0b7	C1b7		0	C0b7	C1b7		0	
4	C0b0	C1b0	0	0	C0b0	C1b0	0	0	
5	C0b1	C1b1	0	0	C0b1	C1b1	0	0	
6	C0b2	C1b2	0	0	C0b2	C1b2	0	0	
7	C0b3	C1b3	0	0	C0b3	C1b3	0	0	

Both flits carry protocol header in Slot 0

No IDE protection

Header bits for Integrity protection only (AES GCM AAD)

Zero padding required by CXL.cachemem IDE. Include these bits in the computation of AAD length.

Because there can be only one IDE.MAC within any given MAC epoch, it is impossible for both flits to carry IDE.MAC. Such a case does not exist and hence not shown here.

Figure 11-11 shows the transmitter's handling of bits that require both encryption and integrity protection for the standard 256B flit when Slot 0 contains LLCTRL (H8) Slot format encoding with an IDE.MAC message. Slot 0 content is not IDE protected. Slots 1 - 14 are mapped to P.

Figure 11-11. Standard 256B Flit - Mapping to AAD and P bits when Slot 0 carries H8

bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15												
slot0	2B HDR																											
slot1	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot2	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot3	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot4	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot5	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot6	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot8	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot9	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot10	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot11	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot12	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot13	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot14	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot15	C0B0	C0B1	CRC (8B)										FEC (6B)															
Slot 0 contains LLCTRL (H8) slot format encoding with IDE.MAC message																												
No IDE protection																												
Integrity protection only (AES GCMAAD)																												
Encrypt and Integrity protect (AES-GCM P)																												
Zero padding required by CXL.cachemem IDE. Encrypt and Integrity protect (AES-GCM P). Include these bits in the computation of P length.																												

Figure 11-12 shows the transmitter's handling of bits that require both encryption and integrity protection for the standard 256B flit when Slot 0 contains protocol header slot format encoding (H0 - H7, H9 - H15). Slot 0 contains 108 bits, starting from bit 4 of the slot header. These bits are padded with leading 0s to align the content to a 128-bit boundary. The padded Slot 0 content, and Slots 1 - 14, are mapped to P .

Figure 11-12. Standard 256B Flit - Mapping to AAD and P bits when Slot 0 Does Not Carry H8

		bytes															
	bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
slot0	0	0	0	0													
	1	0	0	0													
	2	0	0	0													
	3	0	0	0													
	4	0	0	0						B1	B2	B3	B4	B5	B6	B7	
	5	0	0	0						B8	B9	B10	B11	B12	B13	B14	B15
	6	0	0	0						B1	B2	B3	B4	B5	B6	B7	
	7	0	0	0						b4-7							
slot1	0-7	BO	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
slot2	0-7	BO	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
slot3	0-7	BO	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
slot4	0-7	BO	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
slot5	0-7	BO	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
slot6	0-7	BO	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
slot7	0-7	BO	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
slot8	0-7	BO	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
slot9	0-7	BO	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
slot10	0-7	BO	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
slot11	0-7	BO	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
slot12	0-7	BO	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
slot13	0-7	BO	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
slot14	0-7	BO	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
slot15		COBO	COB1				CRC (8B)									FEC (6B)	
Slot 0 contains protocol header slot format encoding (H0 – H7, H9 – H 15)																	
No IDE protection																	
Integrity protection only (AES GCMAAD)																	
Encrypt and Integrity protect (AES-GCM P)																	
Zero padding required by CXL.cachemem IDE. Encrypt and Integrity protect (AES-GCM P). Include these bits in the computation of P length.																	

EVALUATION COPY

Figure 11-13 shows the transmitter's handling of bits that require both encryption and integrity protection for the latency-optimized 256B flit when Slot 0 contains LLCTRL (H8) Slot format encoding with an IDE.MAC message. Slot 0 content is not IDE protected. Slots 1 - 14 are mapped to P .

Figure 11-13. Latency-Optimized 256B Flit - Mapping to AAD and P Bits when Slot 0 Carries H8

	bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15												
slot0		2B HDR	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13													
slot1	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot2	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot3	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot4	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot5	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot6	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot7	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot8	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	0	0	0	0												
slot9	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot10	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot11	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot12	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot13	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot14	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15												
slot15		C0B0	C0B1	Slot8 B10, B11		CRC (8B)				FEC (6B)																			
Slot 0 contains LLCTRL (H8) slot format encoding with IDE.MAC message																													
No IDE protection																													
Integrity protection only (AES GCM AAD)																													
Encrypt and Integrity protect (AES-GCMP)																													
Zero padding required by CXL.cache mem IDE. Encrypt and Integrity protect (AES-GCM P). Include these bits in the computation of P length.																													

Figure 11-14 shows the transmitter's handling of bits that require both encryption and integrity protection for the latency-optimized 256B flit when Slot 0 contains protocol header slot format encoding (H0 - H7, H9 - H15). Slot 0 contains 108 bits, starting from bit 4 of the slot header. These bits are padded with leading 0s to align the content to a 128-bit boundary. The padded Slot 0 content, and Slots 1 - 14, are mapped to P .

Figure 11-14. Latency-Optimized 256B Flit - Mapping to AAD and P Bits when Slot 0 Does Not Carry H8

		bytes																															
	bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15																
slot0	0	0	0	0																													
	1	0	0	0																													
	2	0	0	0																													
	3	0	0	0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13																
	4	0	0																														
	5	0	0		b4-7																												
	6	0	0																														
	7	0	0																														
slot1	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15																
slot2	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15																
slot3	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15																
slot4	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15																
slot5	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15																
slot6	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15																
slot7	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15																
slot8	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	0	0	0	0																
slot9	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15																
slot10	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15																
slot11	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15																
slot12	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15																
slot13	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15																
slot14	0-7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15																
slot15	C0B0 C0B1	Slot8 B10, B11		CRC (8B)				FEC (6B)																									
Slot 0 contains protocol header slot format encoding (H0 - H7, H9 - H 15)																																	
No IDE protection																																	
Integrity protection only (AES GCM AAD)																																	
Encrypt and Integrity protect (AES-GCMP)																																	
Zero padding required by CXL.cache mem IDE. Encrypt and Integrity protect (AES-GCM P). Include these bits in the computation of P length.																																	

When operating in Skid mode, implementations can choose to maximize the benefits of latency optimization by decrypting and processing Slot 8 bytes 0 - 9, and Slots 9 - 14 as soon as they are received. Only MAC computation and decryption of Slot 8 bytes 10 - 11 needs to wait until Slot 14 is received. In such cases, implementation-specific mechanisms should exist to unwind IDE processing if CRC/FEC checks fail.

11.3.3 Encrypted PCRC

A polynomial with the coefficients 1EDC 6F41h shall be used for PCRC calculation. PCRC calculation shall begin with an initial value of FFFF FFFFh. The PCRC shall be calculated across all the bytes of plaintext in the aggregated flits that are part of the given MAC epoch. PCRC calculation shall begin with bit0 byte0 of flit plaintext content and sequentially include bits 0 - 7 for each byte of the flit contents that are mapped to the plaintext. After accumulating the 32-bit value across the flit contents, the PCRC value shall be finalized by taking 1's complement of the bits of accumulated value to obtain PCRC[31:0].

On the transmitter side (see [Figure 11-15](#)), the PCRC value shall be appended to the end of the aggregated flit plaintext content, encrypted, and then included in the MAC calculation. The encrypted PCRC value is not transmitted across the link.

EVALUATION COPY

On the receiver side (see [Figure 11-16](#)), the PCRC value shall be recalculated based on the received, decrypted ciphertext. When the last flit of the current MAC epoch has been processed, the accumulated PCRC value shall be XORed (encrypted) with the AES keystream bits that immediately follow the values that are used for decrypting the received cipher flit. This encrypted PCRC value shall be appended to the end of the received ciphertext for the purposes of MAC computation.

Figure 11-15. Inclusion of the PCRC Mechanism in the AES-GCM Advanced Encryption Function

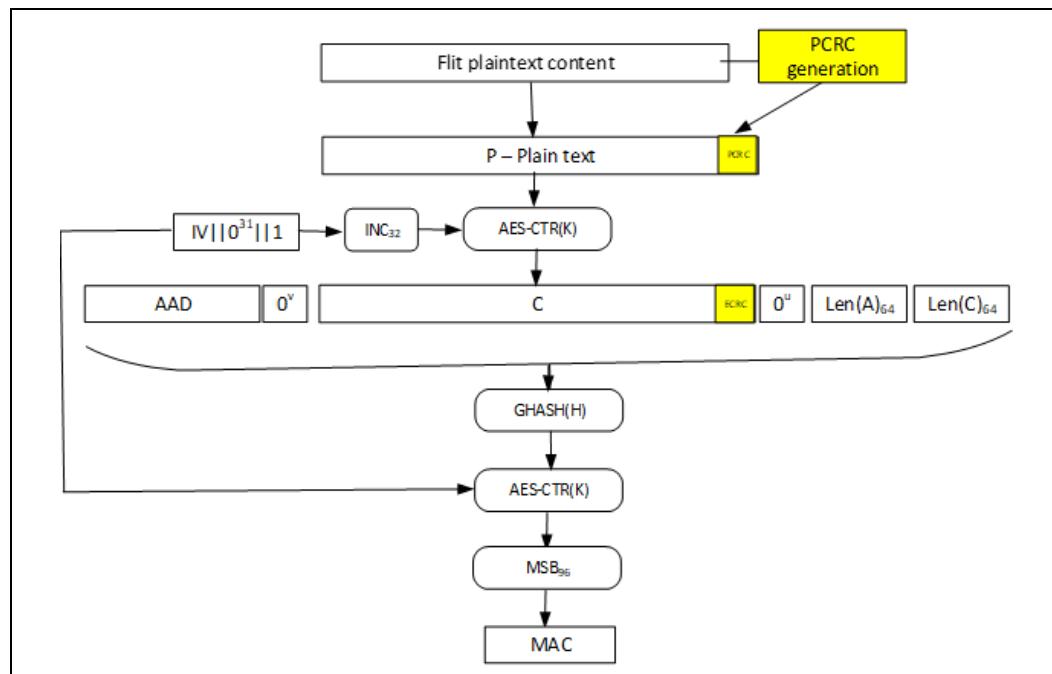
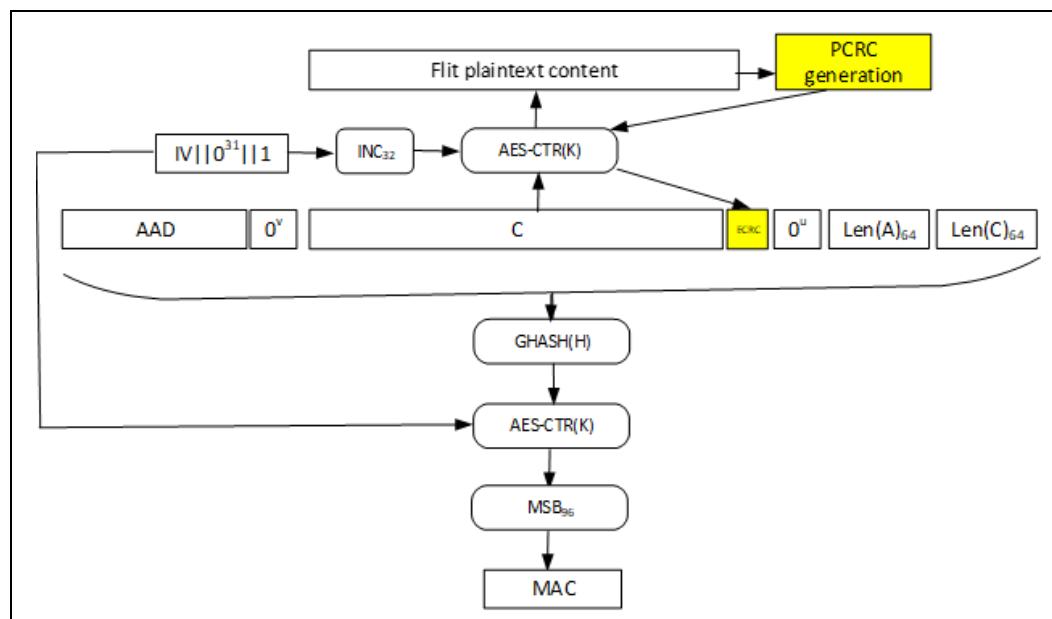


Figure 11-16. Inclusion of the PCRC Mechanism in the AES-GCM Advanced Decryption Function



11.3.4 Cryptographic Keys and IV

Initialization of a CXL.cachemem IDE Stream involves multiple steps. It is possible that some of these steps can be merged or performed in a different order. The first step is to establish the authenticity and identity of the components that contain the two ports that operate as endpoints for a CXL.cachemem IDE Stream. The second step is to establish the IDE Stream keys. In some cases, these two steps may be combined. Third, the IDE is configured. Finally, the establishment of the IDE Stream is triggered.

CXL.cachemem IDE may make use of CXL.io IDE mechanisms for device attestation and key exchange using a standard mechanism, as described in [Section 11.4](#).

IV construction of CXL.cachemem IDE is described below. A 96-bit *IV* of deterministic construction is used as per NIST Special Publication 800-38D for AES-GCM.

All ports shall support the Default *IV* Construction. The default *IV* construction is as follows:

- A fixed field is located at bits 95:64 of the *IV*, where bits 95:92 contain the sub-stream identifier, 1000b, and bits 91:64 are all 0s. The same sub-stream encoding is used for both transmitted and received flits; however, the keys that the port uses during transmit and receive flows must be distinct.
- Bits 63:0 of the *IV* are referred to as the invocation field. The invocation field contains a monotonically incrementing counter with rollover properties. The invocation field is initially set to the value 0000 0001h for each sub-stream upon establishment of the IDE Stream including a rekeying flow. If the CXL.cachemem IV Generation Capable bit in CXL_QUERY_RESP returns the value of 1, the port is capable of initially setting *IV* to a value other than what is generated via the Default *IV* Construction. See the CXL_KEY_PROG message definition (see [Section 11.4.5](#)) for details.

In either case, the invocation field is incremented every time an *IV* is consumed. Neither the transmitter nor the receiver are required to detect *IV* rollover¹ and are not required to take any special action when the *IV* rolls over.

11.3.5 CXL.cachemem IDE Modes

CXL.cachemem IDE supports two modes of operation:

- Containment mode: In Containment mode, the data is released for further processing only after the integrity check passes. This mode impacts both latency and bandwidth. The latency impact is due to the need to buffer several flits until the integrity value has been received and checked. The bandwidth impact comes from the fact that integrity value is sent quite frequently. If Containment mode is supported and enabled, the devices (and hosts) shall use an Aggregation Flit Count of 5 in 68B Flit mode and 2 in 256B Flit mode.
- Skid mode: Skid mode allows the data to be released for further processing without waiting for the integrity value to be received and checked. This allows for less-frequent transmission of the integrity value. Skid mode allows for near-zero latency overhead and low bandwidth overhead. In this mode, data modified by an adversary is potentially consumed by software; however, such an attack will subsequently be detected when the integrity value is received and checked. If Skid mode is supported and enabled, all devices (and hosts) shall use an Aggregation Flit Count of 128 in 68B Flit mode and of 32 in 256B Flit mode. When using this mode, the software and application stack must be capable of tolerating attacks within a narrow time window, or the result is undefined.

1. For a x16 link operating at 32 GT/s, a 64-bit *IV* will take longer than 1000 years to roll over.

11.3.5.1 Discovery of Integrity Modes and Settings

Each port shall enumerate the modes that the port supports and other capabilities via registers in the CXL IDE Capability Structure (see [Section 8.2.4.22](#)). All devices adherent to this specification shall support Containment mode.

11.3.5.2 Negotiation of Operating Mode and Settings

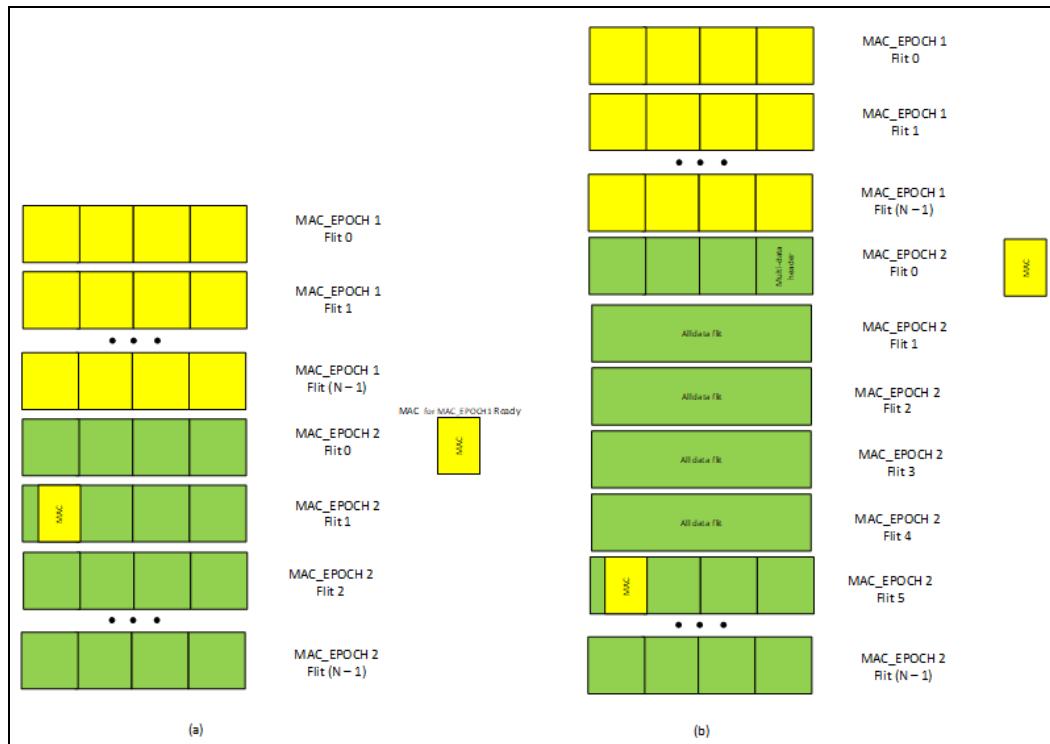
The operating mode and timing parameters are configured in the CXL IDE Capability Structure (see [Section 8.2.4.22](#)) prior to enabling of CXL.cachemem IDE.

11.3.5.3 Rules for MAC Aggregation

The rules for generation and transfer of MAC are as follows:

- MAC epoch: A MAC epoch is defined as the set of consecutive flits that are part of a given aggregation unit. The IDE mode (see [Section 11.3.5](#)) determines the number of flits in a standard MAC epoch. This number is known as Aggregation Flit Count (referred to as N below). Every MAC epoch with the exception of early MAC termination (see [Section 11.3.6](#)) carries N flits. A given MAC header shall contain the tag for exactly one MAC epoch. The transmitter shall accumulate the integrity value over flits in exactly one MAC epoch (that is at most N flits) prior to transmitting the MAC epoch.
- In all cases, the transmitter must send MACs in the same order as MAC epochs.
- [Figure 11-17](#) shows an example of MAC generation and transmission for one MAC epoch in the presence of back-to-back protocol traffic for the 68B flit format. [Figure 11-17](#) (a) shows that the earliest MAC may be transmitted, assuming that the transmitter completes MAC computation (and gets MAC ready) one cycle after the MAC epoch completes. The earliest flit to be transmitted or received is shown on the top of the figure. Thus, Flits 0 to N-1 (shown in yellow) belonging to MAC Epoch 1 are transmitted in that order. The MAC is calculated over Flits 0 to N-1.

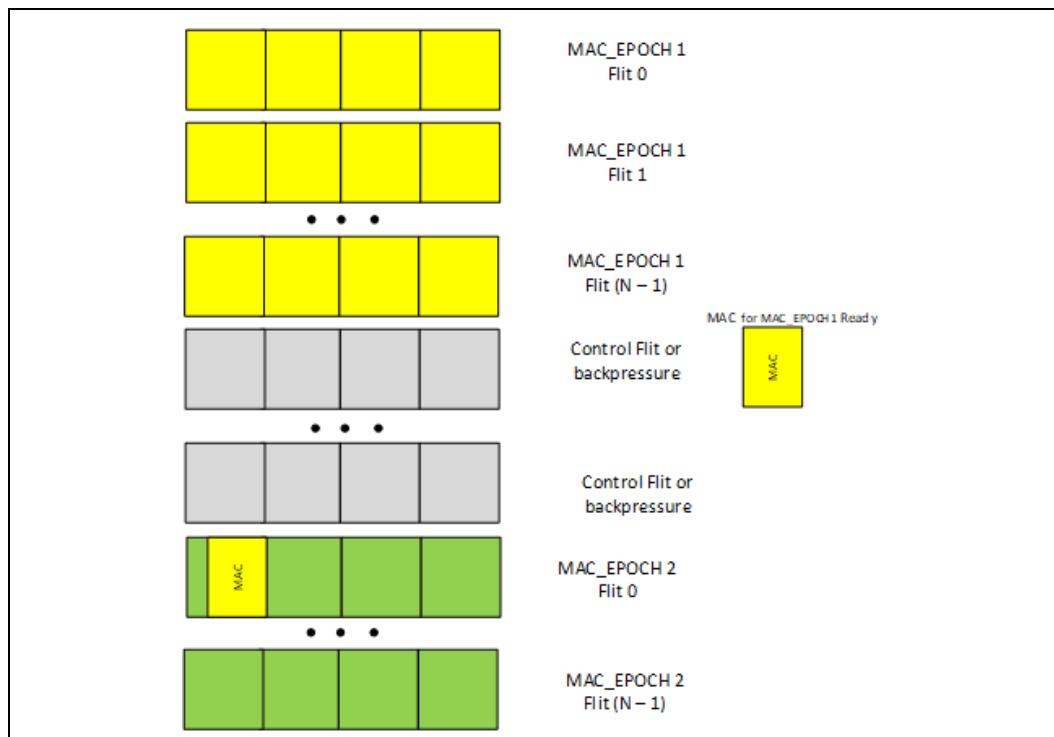
Figure 11-17. MAC Epochs and MAC Transmission in Case of Back-to-Back Traffic
(a) Earliest MAC Header Transmit
(b) Latest MAC Header Transmit in the Presence of Multi-Data Header



- The transmitter shall send the MAC header that contains this integrity value at the earliest possible time. Protocol flits belonging to the next MAC epoch are permitted to be sent between the last flit of the current epoch and the transmission of the MAC header for the current epoch. This is needed to handle the transmission of all-data flits and is also useful for avoiding bandwidth bubbles due to MAC calculation latency. It is recommended that the transmitter send the MAC header on the first available Slot 0 header immediately after the MAC calculations are complete.
- On the receiver side, the receiver may expect the MAC header to come in on any protocol flit, from first to sixth protocol flits, after the last flit of the previous MAC epoch (see [Figure 11-17 \(b\)](#)).

EVALUATION COPY

Figure 11-18. Example of MAC Header Being Received in the First Flit of the Current MAC Epoch



- In Containment mode, the receiver must not release flits of a given MAC epoch for consumption until the MAC header that contains the integrity value for those flits has been received and the integrity check has passed. In 68B Flit mode, because the receiver can receive up to 5 protocol flits that belong to the current MAC epoch before receiving the MAC header for the previous MAC epoch, the receiver shall buffer the current MAC epoch's flits to ensure that there is no data loss. For example, referring to [Figure 11-17 \(b\)](#), both the yellow and green flits are buffered until MAC Epoch 1's MAC header is received and the integrity check passes. If the check passes, the yellow flits can be released for consumption. The green flits cannot, however, be released until the green MAC flit has been received and the integrity verified. [Section 11.3.8](#) defines the receiver behavior upon integrity check failure.
- In Skid mode, the receiver may decrypt and release the flits for consumption as soon as they are received. The MAC value shall be accumulated as needed and then checked when the MAC header for the flits in the MAC epoch arrives. Again, referring to [Figure 11-17 \(b\)](#), both the yellow and green flits may be decrypted and released for consumption without waiting for the MAC header for MAC Epoch 1 to be received and verified. When MAC Epoch 1's MAC header is received, the header is verified. If the check passes, there is no action to be taken. If the MAC header is not received within 6 protocol flits after the end of the previous MAC epoch, the receiver shall treat the absence of MAC as an error. [Section 11.3.8](#) defines the receiver behavior upon integrity check failure, a missing MAC header, or a delayed MAC header.
- In 68B Flit mode, in all cases (including the cases with multi-data headers), at most 5 protocol flits belonging to the current MAC epoch are allowed to be transmitted prior to the transmission of the MAC for the previous MAC epoch. If the MAC header is not received within 6 protocol flits after the end of the previous MAC epoch, the receiver shall treat the absence of MAC as an error.

- In 256B Flit mode, in all cases, at most 1 protocol flit that belongs to the current MAC epoch is allowed to be transmitted prior to the transmission of the MAC for the previous MAC epoch. If the MAC header is not received within 2 protocol flits after the end of the previous MAC epoch, the receiver shall treat the absence of MAC as an error.

IMPLEMENTATION NOTE

In Containment mode, the receiver must not release any decrypted flits for consumption unless their associated MAC check has been performed and passed. This complies with the algorithm for the AES-GCM Authenticated Decryption Function as defined in NIST Special Publication 800-38D.

In Skid mode, the receiver is permitted to release any decrypted flits for consumption without waiting for their associated MAC check to be performed. Unless there are additional device-specific mechanisms to prevent this consumption, the use of Skid mode will not meet the requirements of the above-mentioned algorithm.

Solution stack designers must carefully weigh the benefits vs. the constraints when choosing between Containment mode and Skid mode. Containment mode guarantees that potentially corrupted data will not be consumed. Skid mode provides data privacy and eventual detection of data integrity loss, with significantly less latency impact and link-bandwidth loss compared to Containment mode. However, the use of Skid mode may be more vulnerable to security attacks and will require additional device-specific mechanisms if it is necessary to prevent corrupt data from being consumed.

11.3.6 Early MAC Termination

A transmitter is permitted to terminate the MAC epoch early and transmit the MAC for the flits in a truncated MAC epoch when fewer than the Aggregation Flit Count of flits have been transmitted in the current MAC epoch. This can occur as part of link idle handling. The link may be ready to go idle after the transmission of a number of protocol flits, less than the Aggregation Flit Count, in the current MAC epoch.

The following rules shall apply to the early MAC epoch termination and the MAC transmission.

- The transmitter is permitted to terminate the MAC epoch early if and only if the number of protocol flits in the current MAC epoch is less than Aggregation Flit Count. The MAC for this truncated MAC epoch shall be transmitted by itself in the IDE.TMAC Link Layer Control flit (see [Table 4-10](#)). This subtype is referred to as a Truncated MAC flit within this specification.
- Any subsequent protocol flits would become part of a new MAC epoch and would be transmitted after the Truncated MAC flit.
- The MAC for the truncated MAC epoch is calculated identically to the MAC calculation for normal cases, except that it is accumulated over fewer flits.

[Figure 11-20](#) shows an example of truncating the current MAC epoch after 3 protocol flits. Flits in current MAC epoch can contain any valid protocol flit including a header flit that contains the MAC for the previous MAC epoch. The MAC for the current MAC epoch shall be sent using a Truncated MAC flit. The Truncated MAC flit will be transmitted following the three protocol flits of the current MAC epoch with no other intervening protocol flits from the next MAC epoch.

EVALUATION COPY

Figure 11-19. Early Termination and Transmission of Truncated MAC Flit

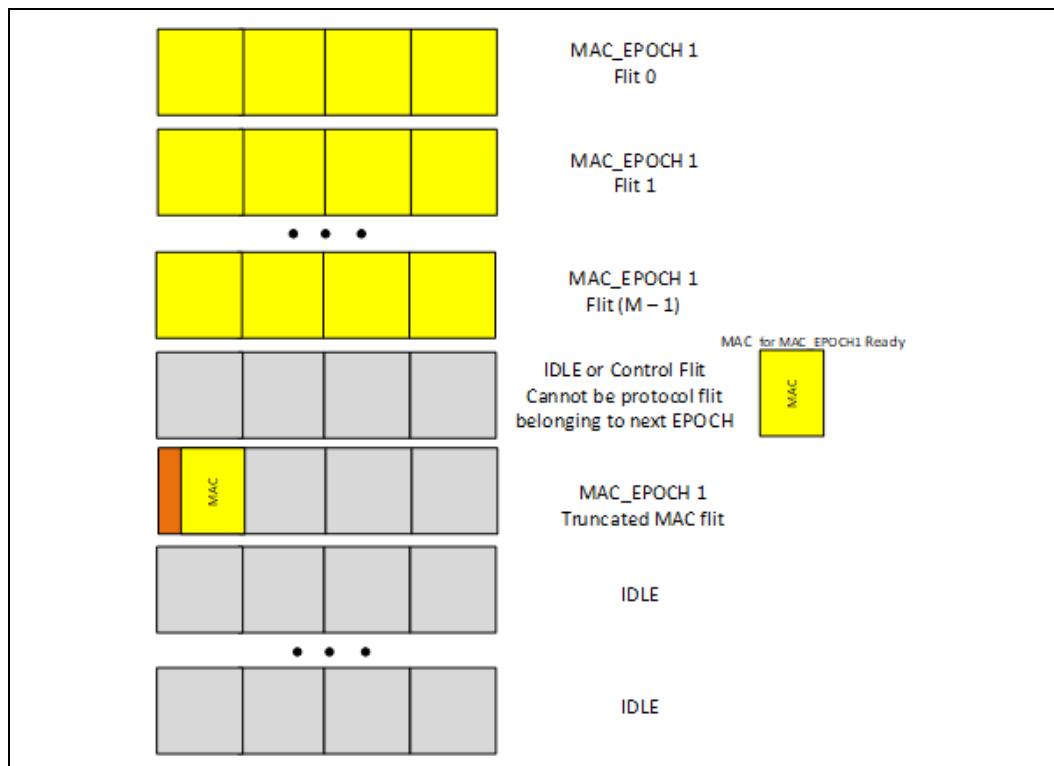
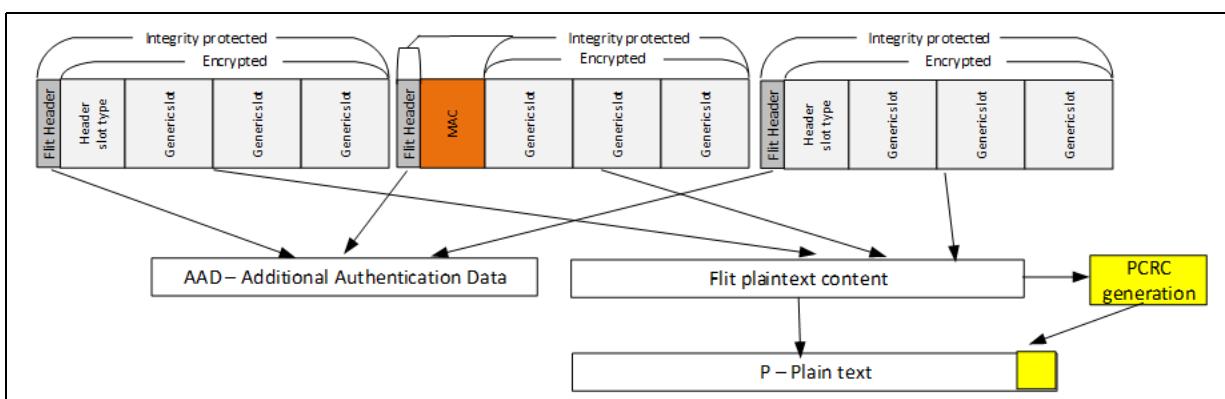


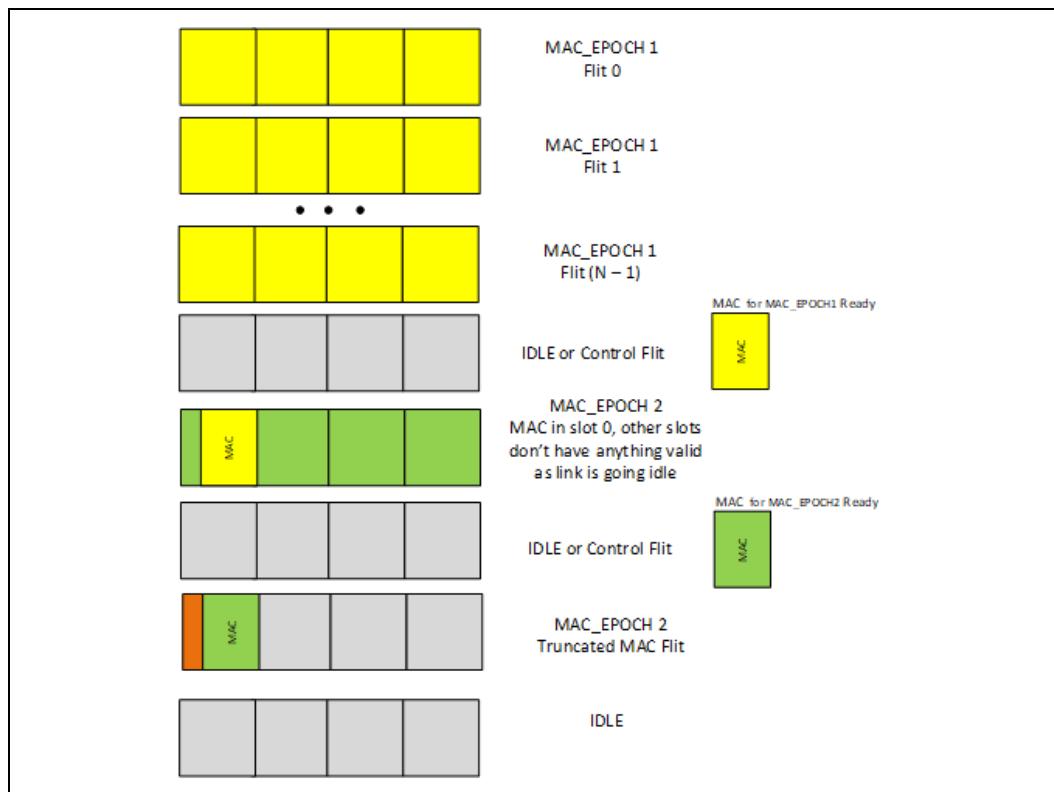
Figure 11-20. CXL.cachemem IDE Transmission with Truncated MAC Flit



In the case where the link goes idle after sending exactly the Aggregation Flit Count number of flits in the MAC epoch, then the Truncated MAC flit as defined above must not be used. The MAC header must be part of the next MAC epoch. This new MAC epoch is permitted to be terminated early using the Truncated MAC flit (see Figure 11-21).

EVALUATION COPY

Figure 11-21. Link Idle Case after Transmission of Aggregation Flit Count Number of Flits



After the transmitter sends out the MAC flit for all the previous flits that were in flight, the transmitter may go idle. The receiver is permitted to go idle after the MAC flit that corresponds to previously received flits has been received and verified. IDE.Idle control flits are retryable and may be resent as part of replay.

After early MAC termination and transmittal of the Truncated MAC, the transmitter must send at least TruncationDelay number of IDE.Idle flits before it can transmit any protocol flits. TruncationDelay is defined via the following equation:

Equation 11-1.

$$\text{TruncationDelay} = \text{Min}(\text{Remaining Flits}, \text{Tx Truncation Transmit Delay})$$

Tx Truncation Transmit Delay (see [Section 8.2.4.22.8](#)) is a configuration parameter to account for the potential discarding of any precalculated AES keystream values for the current MAC epoch that need to be discarded. Remaining Flits represent the number of flits remaining to complete the current MAC epoch and is calculated as follows:

Equation 11-2.

$$\text{Remaining Flits} = \text{Aggregation Flit Count} - \text{Number of protocol flits transmitted in current MAC epoch}$$

11.3.7

Handshake to Trigger the Use of Keys

Each port exposes a register interface that software can use to program the transmit and receive keys and their associated parameters. These programmed keys remain pending in registers until activation. While the keys are in the process of being exchanged and configured in the Upstream and Downstream Ports, the link may actively be using a previously configured key. The new keys shall not take effect until the actions described below are taken.

The mechanism described below is used to switch the backup keys to the active state. This is needed to ensure that the Transmitter and Receiver change to using the programmed keys in a coordinated manner.

After the keys are programmed into pending registers on both sides of the link, receipt of the CXL_K_SET_GO request shall cause each transmitter on each port to trigger the transmission of an IDE.Start Link Layer Control flit (see [Table 4-3](#)).

After the IDE.Start flit has been sent, all future protocol flits shall be protected by the new keys. To allow the receiver to prepare to receive the flits protected by the new key, the Transmitter is required to send IDE.Idle flits, as defined in [Table 4-10](#) for the number of flits specified by the Tx Key Refresh Time field in the Key Refresh Time Control register (see [Section 8.2.4.22.7](#)) prior to sending any protocol flits with the new key. These IDE.Idle flits are not encrypted or integrity protected. To prepare to use the new keys, the Tx Key Refresh Time in the transmitter must be configured to a value that is higher than the worst-case latency in the receiver, which is advertised by the receiver via Rx Min Key Refresh Time field in the Key Refresh Time Capability register (see [Section 8.2.4.22.5](#)) or Rx Min Key Refresh Time2 field in the Key Refresh Time Capability2 register (see [Section 8.2.4.22.9](#)), depending on the Flit mode.

After receiving the IDE.Start flit, the receiver must change to using the new keys if the transmitter has met the AES-GCM requirements. During key refresh, it is recommended that the transmitter send an IDE.TMAC before sending an IDE.Start.

It is also permissible for the transmitter to send an IDE.Start after the MAC epoch ends but before the corresponding MAC header is transmitted. In this scenario, the receiver must use the old keys to decrypt the message and to check the MAC.

The transmitter must not send an IDE.Start in the middle of a MAC epoch because doing so violates the fundamental AES-GCM requirement that a single key be used as the input. If the IDE.Start is received in the middle of a MAC epoch, then the receiver shall drop the IDE.Start. The receiver may also set the Rx Error Status field in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#)) to CXL.cachemem IDE Establishment Security error and may transition to Insecure State upon detecting this condition.

The IDE.Start flit shall be ordered with respect to the protocol flits. In case of link-level retries, the receiver shall complete retries of previously sent protocol flits before handling the IDE.Start flit and changing to the new key. Other events such as link retraining can occur in the middle of this flow as long as the ordering specified above is maintained.

11.3.8

Error Handling

CXL IDE does not impact or require any changes to the link CRC error handling and the link retry flow.

CXL.cachemem IDE error conditions are enumerated and logged in the Rx Error Status field, Tx Error Status or Unexpected IDE.Stop received fields in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#)). When a CXL.cachemem IDE error is detected,

the appropriate bits in the Uncorrectable Error Status register (see [Section 8.2.4.17.1](#)) are also set and the error is signaled using the standard CXL.cachemem protocol error signaling mechanism.

Unless stated otherwise, errors logged in Rx Error Status field or Tx Error Status field cause the CXL.cachemem IDE stream to transition from Active State to Insecure State if it is Active at the time of the error. Note that some of the error conditions that are logged under CXL.cachemem IDE Establishment may not always result in termination of CXL.cachemem IDE stream.

Upon transition to Insecure state:

- Any buffered protocol flits are dropped and all subsequent protocol traffic is dropped until the link is reset.
- Components shall prevent any leakage of keys or user data. The component may need to implement mechanisms to clear the data/state or have access control to prevent leakage of secrets. Such mechanisms and actions are component specific and beyond the scope of this specification.

11.3.9 Switch Support

A CXL switch that supports CXL.cachemem IDE must support Link IDE for CXL.io traffic. A CXL switch may also optionally support Selective Stream IDE for CXL.io traffic, including flow-through Selective IDE Streams. A CXL switch may only support Selective Stream IDE in flow-through mode for CXL.io traffic. In this case, CXL.cachemem IDE cannot be enabled on the host side. In the case of multi-VCS capable switches, CXL IDE may be enabled on a per-root port basis. However, after any root port has enabled CXL IDE, the downstream link from the switch to the MLDs that support CXL IDE, must also have Link IDE enabled. Thus, the traffic from a root port which has not enabled CXL IDE that is targeting an MLD that has enabled CXL IDE would be encrypted and integrity protected between the switch and the device.

IMPLEMENTATION NOTE: IDE CONFIGURATION OF CXL SWITCHES

The following examples describe three different models for configuring the CXL.cachemem IDE and performing key exchanges with the CXL switches and the devices attached to them.

Model A

Host performs key exchange with the CXL switch and enables CXL IDE. The host will then enumerate the Downstream Ports in the CXL switch and perform key exchange with those downstream devices that support CXL IDE. The Host then programs the keys into the respective Downstream Ports of the switch and enables CXL IDE.

Model B

Host performs key exchange with the CXL switch and enables CXL IDE. In parallel, CXL switch will enumerate downstream devices and then perform key exchange with those downstream devices that support CXL IDE. The Switch then programs the keys into the respective Downstream Ports of the switch and enables CXL IDE. Host may obtain a report from the CXL switch regarding the enabling of CXL IDE for downstream devices, which includes information about the public key that was used to attest to the device EP. Host may directly obtain an attestation from the device Endpoint and confirm that the Endpoint in question has the same public key that the Switch used as part of the key exchange.

Model C

An out-of-band agent may configure keys into the host, switch, and devices via out-of-band means and then directly enable CXL IDE.

11.3.10 IDE Termination Handshake

This section describes a mechanism that disables IDE on both the transmitter and receiver. This is accomplished via IDE.Stop control flit (see [Table 4-20](#)). This optional capability for 256B Flit mode simplifies the software synchronization and quiescing requirements. This ensures that the transmitter and receiver disable CXL.cachemem IDE in a coordinated manner.

After IDE is enabled and functional, receipt of a CXL_K_SET_STOP request shall cause each transmitter on each IDE.Stop capable port to trigger the transmission of an IDE.Stop Link Layer Control flit (see [Table 4-20](#)) if enabled by programming CXL IDE Control register (see [Section 8.2.4.22.2](#)). The transmitter shall ensure that the currently active MAC epoch is terminated using an IDE.TMAC prior to sending an IDE.Stop message with no intervening protocol flits. IDE.TMAC sent before IDE.Stop shall follow the standard rules for early MAC termination defined in [Section 11.3.6](#). If a valid TMAC sequence is not received before IDE.Stop, the IDE.Stop shall be dropped and Unexpected IDE.Stop received bit in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#)) shall be set.

After the IDE.Stop is sent, all future protocol flits shall not be IDE protected. To allow the receiver to cleanly clear any pending IDE states, including precomputed information, the transmitter is required to send IDE.Idle flits, as defined in [Table 4-10](#), for the number of flits specified by the Tx Key Refresh Time field in the Key Refresh Time Control register (see [Section 8.2.4.22.7](#)) prior to sending any protocol flits without IDE protection.

After receiving an IDE.Stop flit, the receiver must complete all pending actions for the currently active MAC epoch prior to disabling IDE.

Any IDE.Stop message that is received prior to a successful CXL_K_SET_STOP shall be dropped and the Unexpected IDE.Stop received bit in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#)) shall be set.

If the IDE.Stop is received by a receiver that is IDE.Stop Capable but is not configured to process IDE.Stop, it shall drop the IDE.Stop flit and the Unexpected IDE.Stop received bit in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#)) shall be set. If the Rx port receives an IDE.Stop while the IDE stream is inactive, the Rx port shall drop the IDE.Stop flit and set the Unexpected IDE.Stop received bit in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#)).

11.4

CXL.cachemem IDE Key Management (CXL_IDE_KM)

System software or system firmware may follow this specification to configure the ports at both ends of a CXL link that have matching CXL.cachemem IDE keys, Initial IV, and other settings, in an interoperable way. The software or firmware entity that performs this activity is referred to as CXL.cachemem IDE Key Management Agent (CIKMA).

The port pairs, also called the partner ports, may consist of the following:

- A CXL RP and a CXL USP
- A CXL RP and a CXL EP
- A CXL DSP and a CXL EP

CXL root port CXL.cachemem IDE key programming may be performed via host-specific method and may not use the programming steps described in this section.

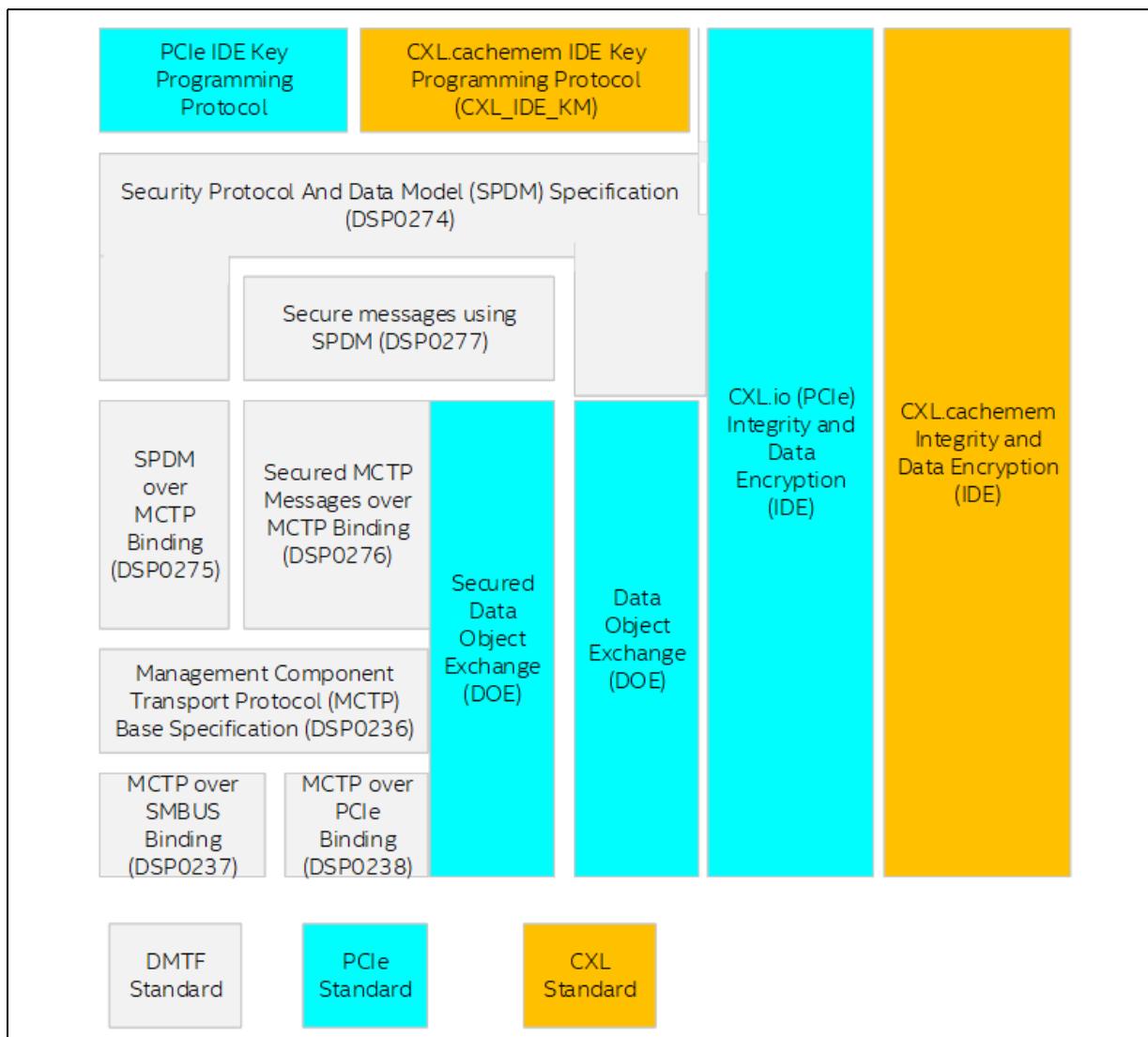
The CXL.cachemem IDE Establishment flow consists of three major steps:

1. CIKMA reads CXL IDE capability registers on both ends of the CXL link and configures various CXL.cachemem IDE control registers on both ends of the CXL

link. See [Section 8.2.4.21](#) for definition of these registers and the programming guidelines.

2. CIKMA sets up an SPDM secure session with each of the partner ports that are being set up. This is accomplished by issuing SPDM key exchange messages over transports such as PCIe DOE or MCTP. If one of the partner ports is an RP and the RP supports a proprietary IDE programming flow, an SPDM secure session with RP may not be needed.
3. CIKMA queries port capabilities, optionally obtains locally generated key and *IV* from each port if they are capable, configures CXL.cachemem IDE Rx/Tx keys/*IV*, and enables CXL.cachemem IDE using CXL_IDE_KM messages that are defined in [Section 11.4.1](#). These messages are secured using the SPDM session key that was established by CIKMA via the previous step.

Figure 11-22. Various Interface Standards that are Referenced by this Specification and their Lineage



11.4.1 CXL_IDE_KM Protocol Overview

CXL_IDE_KM Messages are constructed as SPDM vendor-defined requests and SPDM vendor-defined responses. All request messages begin with a standard Request Header (see [Table 11-2](#)) and all response messages carry a standard Response Header (see [Table 11-3](#)). For the definition of individual fields in the Request and Response Header, please refer to DSP0274. Unless specified otherwise, the behaviors specified in DSP0236, DSP0237, DSP0238, DSP0274, DSP0275, DSP0276, DSP0277, and PCIe Base Specification apply.

CXL_IDE_KM Messages shall be confidentiality and integrity protected in accordance with DSP0277. These secured messages may be sent over a variety of transports, including Secured CMA/SPDM Messages over DOE (see PCIe Base Specification) or Secured Messages over MCTP (see DSP0276).

All CXL.cachemem IDE-capable CXL Switches and Endpoints shall support CMA/SPDM and Secured CMA/SPDM Data Object types over PCIe DOE mailbox. The specific rules regarding the placement of the DOE mailbox are governed by PCIe Base Specification. These data object types are defined in PCIe Base Specification. All CXL.cachemem IDE-capable switches and devices shall support CXL_IDE_KM protocol and CXL_IDE_KM being sent as Secured CMA/SPDM Data Objects.

CXL.cachemem IDE-capable switches and devices may optionally support CXL_IDE_KM messages over MCTP.

The maximum amount of time that the Responder has to provide a response to a CXL_IDE_KM request is 1 second. The requester shall wait for 1 second plus the transport-specific, round-trip transport delay prior to concluding that the request resulted in an error.

11.4.2 Secure Messaging Layer Rules

CXL_IDE_KM messages shall not be issued before an SPDM secure session has been established between the two ports. Any CXL_IDE_KM messages that are not secured shall be silently dropped by the receiver. The first CXL_IDE_KM request message after the SPDM secure session setup shall be CXL_QUERY.

After a successful response to CXL_QUERY, this SPDM session may be used to establish a CXL.cachemem IDE Stream. While this SPDM Session is in progress, any CXL_IDE_KM messages received using a different session ID shall be silently dropped and shall not generate a CXL_IDE_KM response. Any CXL_IDE_KM messages that fail integrity check shall be silently dropped and shall not generate a CXL_IDE_KM response. The act of terminating this SPDM Session or establishment of a different SPDM Secure session by themselves shall not affect the state of the CXL.cachemem IDE stream.

If SPDM Session S1 is used to establish a CXL.cachemem IDE Stream I1, termination of SPDM Session S1 followed by receipt of any valid CXL_IDE_KM message with a new Session S2 shall transition CXL.cachemem IDE Stream I1 to Insecure State. The transition shall occur prior to processing the newly received CXL_IDE_KM message unless the receiver can ensure, via mechanisms not defined here, that S1 and S2 were set up by the same entity; otherwise, the receiver drops the CXL_IDE_KM message with a new Session S2. If the CXL.cachemem IDE stream enters Insecure State due to this condition, the receiver shall set the Rx Error Status field in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#)) to CXL.cachemem IDE Establishment Security error.

It is permitted for a single DOE mailbox instance be used to service CXL_IDE_KM messages as well as CXL.io IDE_KM messages. It is permitted for a single SPDM session be used to set up CXL.io IDE stream as well as CXL.cachemem IDE stream with

a component. The operation and the establishment of CXL.cachemem IDE stream is independent of the operation and establishment of CXL.io IDE stream. It is permitted for a component to support CXL.io IDE but not CXL.cachemem IDE, and vice versa. If a component supports both CXL.io IDE and CXL.cachemem IDE, it may be operated in a mode where only one of the two is active. It is permitted for CXL_IDE_KM messages to be interleaved with IDE_KM messages. CIKMA shall ensure there is at most one outstanding SPDM request of any kind at any time in accordance with DSP0274.

11.4.3 CXL_IDE_KM Common Data Structures

For consistency and reuse reasons, the names of the individual messages follow PCIe Base Specification except for the addition of the prefix CXL, and the message contents closely match PCIe Base Specification.

Unless specified otherwise, all fields are defined as little-endian.

Please refer to DSP0274 for definitions of the fields in the CXL_IDE_KM Request header and Response header.

Table 11-2. CXL_IDE_KM Request Header

Byte Offset	Length in Bytes	Description
0h	1	SPDMVersion
1h	1	RequestResponseCode : Value is 0FEh (VENDOR_DEFINED_REQUEST).
2h	1	Reserved
3h	1	Reserved
4h	2	StandardsID : Value is 03h (PCI-SIG*), indicating that the Vendor ID is assigned by the PCI-SIG.
6h	1	Length of Vendor ID : Value is 02h.
7h	2	Vendor ID : Value is 1E98h (CXL).
9h	2	Request Length : The number of bytes in the message that follow this field. Varies based on the operation that is being requested.

Table 11-3. CXL_IDE_KM Successful Response Header

Byte Offset	Length in Bytes	Description
0h	1	SPDMVersion
1h	1	RequestResponseCode : Value is 07Eh (VENDOR_DEFINED_RESPONSE).
2h	1	Reserved
3h	1	Reserved
4h	2	StandardsID : Value is 03h (PCI-SIG), indicating that the Vendor ID is assigned by the PCI-SIG.
6h	1	Length of Vendor ID : Value is 02h.
7h	2	Vendor ID : Value is 1E98h (CXL).
9h	2	Response Length : The number of bytes in the message that follow this field. Varies based on the operation that was requested.

Table 11-4. CXL_IDE_KM Generic Error Conditions

Error Condition	Response	Effect on an Active CXL.cachemem IDE Stream
CXL_IDE_KM message carries an Object ID that is not defined in this specification	No response is generated. The request is silently dropped.	No change
Unrecognized SPDM major version		

11.4.4 Discovery Messages

The CXL_QUERY request is used to discover the CXL.cachemem IDE capabilities and the current configuration of a port. The port supplies this information in the form of CXL_QUERY RESP response. CIKMA shall not issue another type of CXL_IDE_KM request after CXL_QUERY until CIKMA has received a successful CXL_QUERY RESP response. If CXL_QUERY request is not successful, CIKMA is permitted to retry it.

CIKMA may cross-check the CXL IDE Capability Structure contents that are returned by CXL_QUERY RESP against the component's CXL IDE Capability Structure register values. CIKMA shall abort the CXL.cachemem IDE Establishment flow if CIKMA detects a mismatch.

Table 11-5. CXL_QUERY Request

Byte Offset	Length in Bytes	Description
0h	Bh	Standard Request Header: See Table 11-2.
Bh	1	Protocol ID: Value is 0.
Ch	1	Object ID: Value is 0, indicating CXL_QUERY request.
Dh	1	Reserved
Eh	1	PortIndex: See PCIe Base Specification.

Table 11-6. CXL_QUERY Processing Errors

Error Condition	Response	Effect on an Active CXL.cachemem IDE Stream
Protocol ID is nonzero		
Invalid Request Length	No response is generated. The request is silently dropped.	No change
PortIndex does not correspond to a valid port		

Table 11-7. Successful CXL_QUERY_RESP Response (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	Bh	Standard Response Header: See Table 11-3.
0Bh	1	Protocol ID: Value is 0.
0Ch	1	Object ID: Value is 1, indicating CXL_QUERY response.
0Dh	1	Reserved

Table 11-7. Successful CXL_QUERY_RESP Response (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
0Eh	1	PortIndex: See PCIe Base Specification.
0Fh	1	Dev/Fun Number: See PCIe Base Specification.
10h	1	Bus Number: See PCIe Base Specification.
11h	1	Segment: See PCIe Base Specification.
12h	1	MaxPortIndex: See PCIe Base Specification.
13h	1	<ul style="list-style-type: none"> • Bits[3:0]: CXL IDE Capability Version: Must be set to 1. See Section 8.2.4.7. • Bit[4]: CXL.cachemem IV Generation Capable: <ul style="list-style-type: none"> — 0 = Port is not capable of locally generating the 96-bit IV and shall always use the default IV construction. — 1 = Port is capable of locally generating the 96-bit IV. If a CXL_KEY_PROG message indicates Use Default IV=0, the port shall use the Locally generated CXL.cachemem IV in the Tx path and shall use the IV value supplied as part of the CXL_KEY_PROG message in the Rx path. The port shall return the Locally generated CXL.cachemem IV as part of CXL_GETKEY_ACK response. • Bit[5]: CXL.cachemem IDE Key Generation Capable: <ul style="list-style-type: none"> — 0 = Port is not capable of locally generating an IDE key. — 1 = Port is capable of locally generating the IDE key, shall use that key in the Tx path, and then return that key as part of the CXL_GETKEY_ACK response. • Bit[6]: CXL_K_SET_STOP Capable: <ul style="list-style-type: none"> — 0 = Port does not support CXL_K_SET_STOP. — 1 = Port supports CXL_K_SET_STOP. • Bit[7]: Reserved
14h	Varies	CXL IDE Capability Structure: For CXL IDE Capability Version=1, the length shall be 20h. Carries the contents of the CXL IDE Capability Structure of this port (see Section 8.2.4.21).

11.4.5 Key Programming Messages

Each CXL.cachemem IDE-capable port shall be capable of storing four keys - Rx active, Tx active, Rx pending, and Tx pending. If CXL.cachemem IDE is active, the Tx active key is used to encrypt the flits and generate the MAC. If CXL.cachemem IDE is active, the Rx active key is used to decrypt the flits and verify the MAC in the Rx direction. This specification does not define a mechanism for directly updating the active keys. A Conventional Reset shall reset the active CXL.cachemem IDE Stream and transition the stream to Insecure State. A CXL reset shall reset the active CXL.cachemem IDE Stream and transition the stream to Insecure State. Transition of the CXL.cachemem IDE session to Insecure State shall clear all the keys, make the keys unreadable, and then mark the keys as invalid. An FLR shall not affect an active CXL.cachemem IDE Stream or the CXL.cachemem IDE keys.

The CXL_KEY_PROG request is used to supply the pending keys. Offset 11h, Bit 1, is used to select between the Rx and the Tx. If CXL.cachemem IV Generation Capable=1, the CXL_KEY_PROG request may also be used to establish the Initial CXL.cachemem IDE IV value to be used with the new IDE session including the rekeying flow.

If both ports (Port1 and Port2) return CXL.cachemem IV Generation Capable=1 in QUERY_RSP, it is recommended that CIKMA issue a CXL_GETKEY request to both ports and obtain Locally generated CXL.cachemem IV values. When issuing a CXL_KEY_PROG message to Port1 Rx and Port2 Tx, CIKMA should initialize the Initial CXL.cachemem IDE IV field (Offset 13h+KSIZE) to match the Port2 Locally generated CXL.cachemem IV and set Default IV=0. When issuing a CXL_KEY_PROG message to Port1 Tx and Port2 Rx, CIKMA should initialize the Initial CXL.cachemem IDE IV field (Offset 13h+KSIZE) to match the Port1 Locally generated CXL.cachemem IV and set Default IV=0.

If either port returns CXL.cachemem IV Generation Capable=0 in QUERY_RSP, CIKMA should set Use Default IV=1 in the CXL_KEY_PROG messages to both ports to indicate that the ports should use the default IV construction in Rx directions and Tx directions.

If Port1 and Port2 are partner ports and if Port1 returns CXL.cachemem IDE Key Generation Capable=1 in QUERY_RSP, it is recommended that CIKMA issue a CXL_GETKEY request to Port1 and obtain its Locally generated CXL.cachemem IDE Key. When issuing the CXL_KEY_PROG message to Port1 Tx and Port2 Rx, CIKMA should initialize the CXL.cachemem IDE Key field to match the Port1 Locally generated CXL.cachemem IDE Key. If Port2 returns CXL.cachemem IDE Key Generation Capable=1 in QUERY_RSP, it is recommended that CIKMA issue a CXL_GETKEY request to Port2 and obtain its Locally generated CXL.cachemem IDE Key. When issuing a CXL_KEY_PROG message to Port2 Tx and Port1 Rx, CIKMA should initialize the CXL.cachemem IDE Key field to match the Port2 Locally generated CXL.cachemem IDE Key. The port is expected to return a different IDE key during every CXL_GETKEY_ACK response. Therefore, CIKMA should ensure that the CXL.cachemem IDE Key supplied during a CXL_KEY_PROG request matches the locally generated CXL.cachemem IDE Key from the previous CXL_GETKEY_ACK responses from that port.

Table 11-8. CXL_KEY_PROG Request

Byte Offset	Length in Bytes	Description
00h	Bh	Standard Request Header: See Table 11-2 .
0Bh	1	Protocol ID: Value is 0.
0Ch	1	Object ID: Value is 2, indicating CXL_KEY_PROG request.
0Dh	2	Reserved
0Fh	1	Stream ID: Value is 0.
10h	1	Reserved
11h	1	<ul style="list-style-type: none"> • Bit[0]: Reserved • Bit[1]: RxTxB: <ul style="list-style-type: none"> — 0 = Rx — 1 = Tx • Bit[2]: Reserved • Bit[3]: Use Default IV: <ul style="list-style-type: none"> — 0 = Port shall use the Initial IV specified at Offset 13h+KSIZE — 1 = Port shall use the Default IV construction • Bits[7:4]: Key Sub-stream: Value is 1000b.
12h	1	PortIndex: See PCIe Base Specification.
13h	KSIZE	CXL.cachemem IDE Key: Program the Pending Key with this value. KSIZE must be 32 for this version of the specification. For layout, see PCIe Base Specification.
13h+KSIZE	12h	Initial CXL.cachemem IDE IV: Overwrites the Pending Initial IV. This field must be ignored if Use Default IV=1. Byte Offsets 16h+KSIZE:13h+KSIZE carry the IV DWORD, IV[95:64]. Byte Offsets 20h+KSIZE:17h+KSIZE carry the IV DWORD, IV[63:32]. Byte Offsets 24h+KSIZE:21+KSIZE carry the IV DWORD, IV[31:0].

Table 11-9 lists the various error conditions that a responder may encounter that are unique to CXL_KEY_PROG and how the conditions are handled. When these conditions are detected, the responder shall respond with a CXL_KP_ACK and set the Status field to a nonzero value.

Table 11-9. CXL_KEY_PROG Processing Errors

Error Condition	Response	Effect on an Active CXL.cachemem IDE Stream
Invalid Request Length		
PortIndex does not correspond to a valid port		
Protocol ID is nonzero		
Stream ID is nonzero		
Key Sub-stream is not 1000b		
CXL_KEY_PROG received prior to CXL_QUERY		
Request to set the Tx Key, but the input Tx key is identical to the current Rx Pending Key. This check is optional.		
Request to set the Rx Key, but the input Rx key is identical to the current Tx Pending Key. This check is optional.		
Request to program the key failed because the pending key slot has a valid key	Do not update the key and IV. Return CXL_KP_ACK with Status=01h.	No change
Request to update Tx key, but the supplied key does not match the locally generated CXL.cachemem IDE key returned during the last CXL_GETKEY_ACK response.		
Request to update Tx IV, but the supplied IV does not match the Locally generated CXL.cachemem IV returned during the last CXL_GETKEY_ACK response.		
The port returned IV Generation Capable=0 in QUERY_RSP, but Use Default IV in CXL_KEY_PROG was not set.		

Upon successful processing of CXL_KEY_PROG, the responder shall acknowledge by sending the CXL_KP_ACK response with Status=0. The nonzero Status values not listed here are reserved by this specification but should be interpreted as an error condition by the requester.

Table 11-10. CXL_KP_ACK Response

Byte Offset	Length in Bytes	Description
00h	Bh	Standard Response Header: See Table 11-3 .
0Bh	1	Protocol ID: Value is 0.
0Ch	1	Object ID: Value is 3, indicating CXL_KP_ACK response.
0Dh	2	Reserved
0Fh	1	Stream ID: Value is 0.
10h	1	Status: See Table 11-9 .
11h	1	<ul style="list-style-type: none"> • Bit[0]: Reserved • Bit[1]: RxTxB: See PCIe Base Specification • Bits[3:2]: Reserved • Bits[7:4]: Key Sub-stream: Value is 1000b
12h	1	PortIndex: See PCIe Base Specification.

11.4.6 Activation/Key Refresh Messages

The CXL_K_SET_GO request is used to prepare an Rx port for a CXL.cachemem IDE Stream. The port shall respond with a CXL_K_GOSTOP_ACK message to indicate that the port is ready.

The CXL_K_SET_GO request is also used to instruct a Tx port to generate an IDE.Start Link Layer Control flit and to start a CXL.cachemem IDE Stream that is protected with the pending Tx key as outlined in [Section 11.3.7](#). As part of successful CXL_K_SET_GO processing, the Tx port shall copy the pending key to be the active key and mark the pending key slot as invalid. If CXL.cachemem IV Generation Capable=1 and the last CXL_KEY_PROG request indicated Use Default IV=0, the Initial CXL.cachemem IDE IV shall also be re-initialized to the value supplied as part of the CXL_KEY_PROG request. If CXL.cachemem IV Generation Capable=0 or the last CXL_KEY_PROG request indicated Use Default IV=1, Default IV construction shall be used. All subsequent protocol flits shall be protected by the new active key until the port enters Insecure State.

Upon receipt of an IDE.Start Link Layer Control flit, the Rx port shall copy the pending key to the active key slot and then mark the pending key slot as invalid. If CXL.cachemem IV Generation Capable=1 and the last CXL_KEY_PROG request indicated Use Default IV=0, the Initial CXL.cachemem IDE IV shall also be re-initialized to the value supplied as part of the CXL_KEY_PROG request. If CXL.cachemem IV Generation Capable=0 or the last CXL_KEY_PROG request indicated Use Default IV=1, Default IV construction shall be used. All subsequent protocol flits shall be protected by the new active key until the port enters Insecure State.

If the Rx port receives an IDE.Start Link Layer Control flit prior to a successful CXL_KEY_PROG since the last Conventional Reset, the Rx port shall drop the IDE.Start flit and then optionally set the Rx Error Status field in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#)) to CXL.cachemem IDE Establishment Security error. If the Rx port receives an IDE.Start Link Layer Control flit while CXL.cachemem IDE is active, but prior to a successful CXL_KEY_PROG since the last IDE.Start, the Rx port shall either (1) drop the IDE.Start flit and then optionally program Rx Error Status=8h to CXL.cachemem IDE Establishment Security error or (2) set the Rx Error Status field in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#)) to CXL.cachemem IDE Establishment Security error and then transition to Insecure State.

If the Rx port receives an IDE.Start Link Layer Control flit prior to a successful CXL_K_SET_GO since the last Conventional Reset, the Rx port shall drop the IDE.Start flit and then optionally set the Rx Error Status field in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#)) to CXL.cachemem IDE Establishment Security error. If the Rx port receives an IDE.Start Link Layer Control flit while CXL.cachemem IDE is active, but prior to a successful CXL_K_SET_GO since the last IDE.Start, the Rx port shall either (1) drop the IDE.Start flit and then optionally set the Rx Error Status field in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#)) to CXL.cachemem IDE Establishment Security error or (2) program Rx Error Status=8h to CXL.cachemem IDE Establishment Security error and then transition to Insecure State.

Offset 11h, Bit 1, is used to select between the Rx and Tx. Offset 11h, Bit 3, controls whether the CXL.cachemem IDE operates in Skid mode or Containment mode.

CIKMA should issue a CXL_K_SET_GO request message to an Rx port and wait for success before issuing a CXL_K_SET_GO request message to the partner Tx port.

Table 11-11. CXL_K_SET_GO Request

Byte Offset	Length in Bytes	Description
00h	Bh	Standard Request Header: See Table 11-2 .
0Bh	1	Protocol ID: Value is 0.
0Ch	1	Object ID: Value is 4, indicating CXL_K_SET_GO structure.
0Dh	2	Reserved
0Fh	1	Stream ID: Value is 0.
10h	1	Reserved
11h	1	<ul style="list-style-type: none"> • Bit[0]: Reserved • Bit[1]: RxTxB: See PCIe Base Specification • Bit[2]: Reserved • Bit[3]: CXL IDE Mode: <ul style="list-style-type: none"> — 0 = Skid mode — 1 = Containment mode • Bits[7:4]: Key Sub-stream: Value is 1000b
12h	1	PortIndex: See PCIe Base Specification.

[Table 11-12](#) lists the various error conditions that a responder may encounter that are unique to CXL_K_SET_GO and how the conditions are handled.

Table 11-12. CXL_K_SET_GO Error Conditions

Error Condition	Response	Effect on an Active CXL.cachemem IDE Stream
Port receives CXL_K_SET_GO request, and pending key is invalid		
If a port receives CXL_K_SET_GO request with an IDE mode that is not supported		
If a port receives CXL_K_SET_GO request while IDE is active and the current IDE mode does not match Byte Offset 11, bit 3, in the new CXL_K_SET_GO request	No response is generated. The request is silently dropped.	No change
Protocol ID is nonzero		
Stream ID is nonzero		
Key Sub-stream is not 1000b		
PortIndex does not correspond to a valid port		
Invalid Request Length		

When a port receives a valid CXL_K_SET_STOP request, the port shall clear the active and pending CXL.cachemem IDE keys and then transition to IDE Insecure State. No errors shall be logged in the IDE Status register when an IDE stream is terminated in response to CXL_K_SET_STOP because this is not an error condition. If both ports support the IDE.Stop message as advertised by the CXL IDE Capability register (see [Section 8.2.4.22.1](#)), CIKMA may enable IDE.Stop on both ends of the link by programming the CXL IDE Control register (see [Section 8.2.4.22.2](#)). If IDE.Stop is enabled on both ends, it is unnecessary to quiesce the CXL.cache and CXL.mem traffic prior to issuing the CXL_K_SET_STOP request. If IDE.Stop is enabled, CIKMA is required to issue a CXL_K_SET_STOP to the Rx and then wait for an acknowledgment of CXL_K_SET_STOP before issuing a CXL_K_SET_STOP to the Tx. If IDE.Stop is not enabled, the Software is expected to quiesce the CXL.cache and CXL.mem traffic prior

to issuing a CXL_K_SET_STOP request to a port that is actively participating in CXL.cachemem IDE to prevent spurious CXL.cachemem IDE errors. The port shall respond with a CXL_K_GOSTOP_ACK message after the port has successfully processed a CXL_K_SET_STOP request.

If the Rx port receives an IDE.Stop Link Layer Control flit while the CXL.cachemem IDE is active, but prior to a successful CXL_K_SET_STOP since the last IDE.Start or any other CXL IDE Key Programming message, the Rx port shall drop the IDE.Stop flit, set the Unexpected IDE.Stop received bit in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#)) but not transition to Insecure State.

Table 11-13. CXL_K_SET_STOP Request

Byte Offset	Length in Bytes	Description
00h	Bh	Standard Request Header: See Table 11-2 .
0Bh	1	Protocol ID: Value is 0.
0Ch	1	Object ID: Value is 5, indicating CXL_K_SET_STOP structure.
0Dh	2	Reserved
0Fh	1	Stream ID: Value is 0.
10h	1	Reserved
11h	1	<ul style="list-style-type: none"> • Bit[0]: Reserved • Bit[1]: RxTxB: See PCIe Base Specification • Bits[3:2]: Reserved • Bits[7:4]: Key Sub-stream: Value is 1000b
12h	1	PortIndex: See PCIe Base Specification.

[Table 11-14](#) lists the various error conditions that a responder may encounter that are unique to CXL_K_SET_STOP and how the conditions are handled.

Table 11-14. CXL_K_SET_STOP Error Conditions

Error Condition	Response	Effect on an Active CXL.cachemem IDE Stream
Port does not support CXL_K_SET_STOP (CXL_K_SET_STOP Capable=0)		
Protocol ID is nonzero		
Stream ID is nonzero	No response is generated. The request is silently dropped.	No change
Key Sub-stream is not 1000b		
PortIndex does not correspond to a valid port		
Invalid Request Length		

Table 11-15. CXL_K_GOSTOP_ACK Response (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	Bh	Standard Response Header: See Table 11-3 .
0Bh	1	Protocol ID: Value is 0.
0Ch	1	Object ID: Value is 6, indicating CXL_K_GOSTOP_ACK structure.
0Dh	2	Reserved
0Fh	1	Stream ID: Value is 0.

Table 11-15. CXL_K_GOSTOP_ACK Response (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
10h	1	Reserved
11h	1	<ul style="list-style-type: none"> Bit[0]: Reserved Bit[1]: RxTxB: See PCIe Base Specification Bits[3:2]: Reserved Bits[7:4]: Key Sub-stream: Value is 1000b
12h	1	PortIndex : See PCIe Base Specification.

11.4.7 Get Key Messages

If the QUERY_RSP response message from the port indicates CXL.cachemem IDE Key Generation Capable=1 or CXL.cachemem IV Generation Capable=1, the port shall support the CXL_GETKEY message.

The CXL_GETKEY message is used to get the Locally generated CXL.cachemem IDE Key from the port and Locally generated CXL.cachemem IV.

Table 11-16. CXL_GETKEY Request

Byte Offset	Length in Bytes	Description
00h	Bh	Standard Request Header : See Table 11-2 .
0Bh	1	Protocol ID : Value is 0.
0Ch	1	Object ID : Value is 7, indicating CXL_GETKEY request.
0Dh	2	Reserved
0Fh	1	Stream ID : Value is 0.
10h	1	Reserved
11h	1	<ul style="list-style-type: none"> Bits[3:0]: Reserved Bits[7:4]: Key Sub-stream: Value is 1000b
12h	1	PortIndex : See PCIe Base Specification.

[Table 11-17](#) lists the various error conditions that a responder may encounter that are unique to CXL_GETKEY and how the conditions are handled.

Table 11-17. CXL_GETKEY Processing Error

Error Description	Error Code	Effect on an Active CXL.cachemem IDE Stream
Invalid Request Length		
PortIndex does not correspond to a valid port		
Protocol ID is nonzero		
Stream ID is nonzero		
Key Sub-stream is not 1000b		
CXL_GETKEY received prior to CXL_QUERY		
Port does not support CXL_GETKEY		

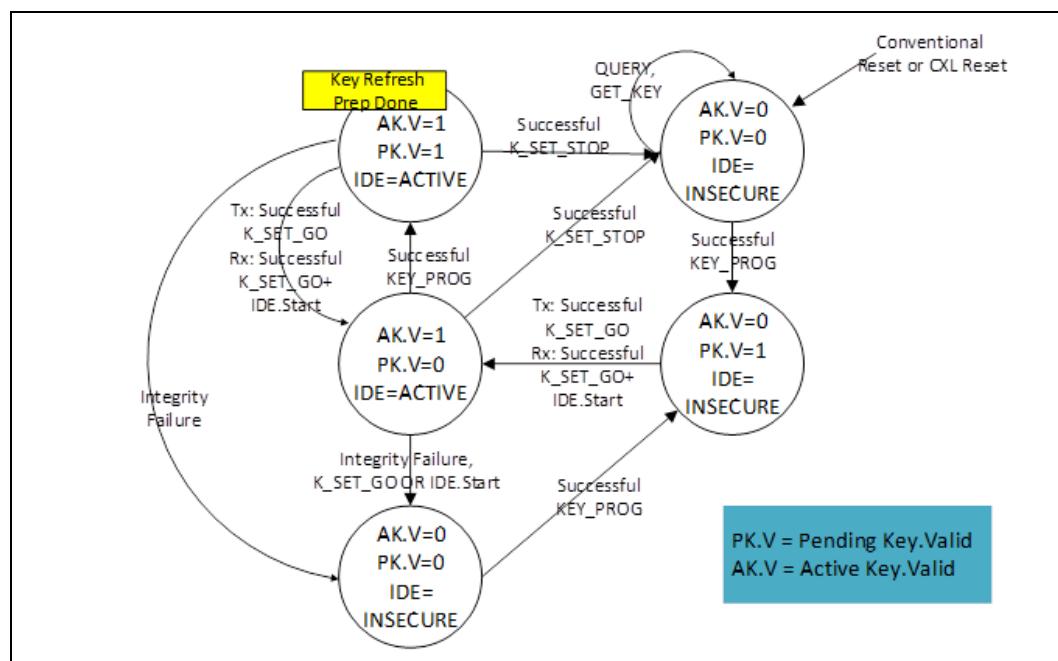
Upon successful processing of CXL_GETKEY, the responder shall acknowledge by sending the CXL_GETKEY_ACK response.

Table 11-18. CXL_GETKEY_ACK Response

Byte Offset	Length in Bytes	Description
00h	Bh	Standard Response Header: See Table 11-3.
0Bh	1	Protocol ID: Value is 0.
0Ch	1	Object ID: Value is 8, indicating CXL_GETKEY_ACK response.
0Dh	2	Reserved
0Fh	1	Stream ID: Value is 0.
10h	1	Reserved
11h	1	<ul style="list-style-type: none"> Bits[3:0]: Reserved Bits[7:4]: Key Sub-stream: Value is 1000b
12h	1	PortIndex: See PCIe Base Specification.
13h	KSIZE	Locally Generated CXL.cachemem IDE Key: The KSIZE must be 32 for this version of the specification. For layout, see PCIe Base Specification. This field must be ignored if the QUERY_RSP response message from the port indicates CXL.cachemem IDE Key Generation Capable=0.
13h+KSIZE	12	Locally Generated CXL.cachemem IV: This field must be ignored if the QUERY_RSP response message from the port indicates CXL.cachemem IV Generation Capable=0. Byte Offsets 16h+KSIZE:13h+KSIZE carry the IV DWORD, IV[95:64]. Byte Offsets 20h+KSIZE:17h+KSIZE carry the IV DWORD, IV[63:32]. Byte Offsets 24h+KSIZE:21h+KSIZE carry the IV DWORD, IV[31:0].

Figure 11-23 illustrates various key states and their transitions. Note that this figure is not meant to be exhaustive and does not include several legal transition arrows for simplicity.

Figure 11-23. Active and Pending Key State Transitions



IMPLEMENTATION NOTE

Establishing CXL.cachemem IDE between a DSP and EP - Example

In this example, host software plays the role of the CIKMA. The switch implementation is such that the USP implements the DOE capability on behalf of all the DSPs and the specific DSP that is involved here is referenced as Port 4. Further, it is also assumed that the desired mode of operation is Skid mode. Host Software reads and configures the CXL IDE capability registers on the DSP and on the EP. See [Section 8.2.4.21](#) for the definition of these registers and programming guidelines.

1. Host Software sets up independent SPDM secure sessions with the USP and the EP. This is accomplished by issuing SPDM key exchange messages over PCIe DOE.
2. All subsequent messages are secured as per DSP0277. The messages to/from the USP are secured using the SPDM session key established with the USP. The messages to/from the EP are secured using the session key established with the EP.
 - a. Host Software sends a CXL_QUERY (PortIndex=4) message to the USP DOE mailbox. The USP returns a CXL_QUERY_RESP. Host Software compares the CXL_QUERY_RESP contents against the CXL IDE Capability structure associated with the DSP. Host Software exits with an error if there is a mismatch or a timeout. In this example, the USP reports that it supports Locally generated CXL.cachemem IV and Locally generated CXL.cachemem IDE Key.
 - b. Host Software sends a CXL_QUERY (PortIndex=0) message to the EP DOE mailbox. The EP returns a CXL_QUERY_RESP. Host Software compares the CXL_QUERY_RESP contents against the CXL IDE Capability structure associated with the EP. Host Software exits with an error if there is a mismatch or a timeout. In this example, the EP reports that it supports Locally generated CXL.cachemem IV and Locally generated CXL.cachemem IDE Key.
 - c. Host Software issues a CXL_GETKEY request to the USP and saves the Locally generated CXL.cachemem IDE Key from the response as KEY2 and the Locally generated CXL.cachemem IV from the response as IV2. Host Software issues CXL_GETKEY request to the EP and saves the Locally generated a CXL.cachemem IDE Key from the response as KEY1 and the Locally generated Tx IV from the response as IV1.
 - d. Host Software programs the Rx pending key in the EP by sending a CXL_KEY_PROG(RxTxB=0, Use Default IV=0, KEY2, IV2) message to the EP DOE mailbox. Host Software programs the Tx pending keys in the DSP by sending a CXL_KEY_PROG(PortIndex=4, RxTxB=1, Use Default IV=0, KEY2, IV2) message to the USP DOE mailbox. Host Software programs the Rx pending keys in the DSP by sending a CXL_KEY_PROG(PortIndex=4, RxTxB=0, Use Default IV=0, KEY1, IV1) message to the USP DOE mailbox. Host Software programs the Tx pending key in the EP by sending a CXL_KEY_PROG(RxTxB=1, Use Default IV=0, KEY1, IV1) message to the EP DOE mailbox. These 4 steps can be performed in any order. Host Software exits with an error if the CXL_KP_ACK indicates an error or if there is a timeout.

IMPLEMENTATION NOTE

Continued

- e. Host Software instructs DSP Rx to be ready by sending a CXL_K_SET_GO(PortIndex=4, Skid mode, RxTxB=0) to the USP DOE mailbox. Host Software instructs the EP Rx to be ready by sending a CXL_K_SET_GO(Skid mode, RxTxB=0) to the EP DOE mailbox. Host Software exits with an error if either CXL_K_SET_GO request times out.
- f. Host Software instructs DSP Tx to enable CXL.cachemem IDE by sending a CXL_K_SET_GO(PortIndex=4, Skid mode, RxTxB=1) to the USP DOE mailbox. DSP sends an IDE.Start Link Layer Control flit to EP and thus initiating IDE in one direction using KEY1 and Starting IV=IV1. Host Software exits with an error if the CXL_K_SET_GO request times out.
- g. Host Software instructs EP Tx to enable CXL.cachemem IDE by sending a CXL_K_SET_GO(Skid mode, RxTxB=1) to the EP DOE mailbox. EP sends an IDE.Start Link Layer Control flit to the DSP and thus initiates IDE in the other direction, using KEY2 and Starting IV=IV2. Host Software exits with an error if CXL_K_SET_GO request times out.
- h. At the end of these steps, all the CXL.cachemem protocol flits traveling between the DSP and EP are protected by IDE.

If both ports support Locally generated CXL.cachemem IDE Key and Locally generated CXL.cachemem IV, the following sequence will result in a programming error and should be avoided:

1. CIKMA issues CXL_GETKEY request to Port1 and saves the key, KEY1
2. CIKMA issues another CXL_GETKEY request to Port1 and saves the IV, IV1
3. CIKMA issues CXL_KEY_PROG request to Port1 Tx and passes in KEY1 and IV1.
4. Port1 returns a CXL_KP_ACK with Status=08h because the 2nd CXL_GETKEY request changed its locally CXL.cachemem IDE generated key to KEY2, which is not equal to KEY1.

11.5

CXL Trusted Execution Environments Security Protocol (TSP)

11.5.1

Overview

Virtualization-based Trusted Execution Environments (TEE) are used to host confidential computing workloads that are isolated from hosting environments. This specification refers to such TEE as Trusted Execution Environment VMs (TVMs) to distinguish them from traditional virtual machines.

The PCI-SIG TEE Device Interface Security Protocol (TDISP) ECR specifies the architecture of a framework for trusted I/O virtualization to include PCIe devices within the TVM trust boundary. The CXL TEE Security Protocol (CXL-TSP), complements the PCI-SIG TDISP specification by specifying mechanisms to include direct attached CXL memory devices within the TVM trust boundary specifically for confidential computing scenarios.

11.5.2 Scope

This CXL security content scope focuses on features that are needed for confidential computing utilizing CXL Type 3 memory expander devices, referred to as targets in the TSP, directly connected to CXL Root Ports owned by the host which is an initiator in TSP. TSP defines the security objectives, capabilities, and interfaces, and the host, initiator, and target behaviors that are required to create a secure CXL memory hierarchy that meets the needs of confidential computing. The scope does not include details on initiator or target security implementation.

- This scope includes support for the following:
 - SPDM 1.2 or newer for authentication and attestation
 - Directly connected LDs, SLDs, and MH-SLDs
 - Dynamic Capacity devices
 - HDM-H memory
 - 256B flit format
 - Memory pooling — Multiple initiators accessing the same physical memory on a device but not sharing access to it
 - Comprehensive Trust security model
 - Selective Trust security model
 - Implicit 64B Cacheline TE State Access Control
 - Explicit TE State Access Control
- This scope does not include the following:
 - CXL switches
 - Devices connected via a CXL switch, including MLDs
 - UIO Direct Peer-to-peer (P2P) to HDM
 - Direct P2P using CXL.mem
 - Direct P2P using CXL.io
 - Type 1 and Type 2 accesses to Type 3 HDM memory
 - HDM-D or HDM-DB memory
 - PBR and 68B flit formats
 - Memory sharing — Multiple initiators accessing the same physical memory on a device and simultaneously sharing access to it

11.5.3 Threat Model

This version of TSP shall focus on providing confidential computing support for direct attached CXL memory. Direct attached memory shall be defined as using the CXL protocol to communicate with a memory device, or target and the CXL Root Ports of the host, without intermediaries in the middle of the two. Within the context of extending CXL for confidential computing, one of TSP's objectives is to minimize the Trusted Computing Base (TCB). The TSP supports both a selective trust and comprehensive trust security model.

11.5.3.1 Definitions

The following additional terms are utilized in this threat model section:

- **Attacker:** Entity that wants to extract information from a communication or influence a computation by modifying information that flows between two participants.
- **Confidential Computing:** Confidential Computing protects Data in Use, Data in Transit and Data at Rest. Data at Rest applies to the CXL memory device and Data in Transit is applies to TSP Transport Security such as CXL IDE. Trusted Execution Environments (TEEs) prevent unauthorized access or modification of applications and data while in use, thereby increasing the security assurances for organizations that manage sensitive and regulated data.
- **Covert Channel:** Method for an accomplice inside a trusted entity to signal to an attacker outside a trusted entity.
- **Target:** Participant in the protocol that does not forward packets to other participants. The memory device.
- **Host:** Location in which multiple participants concurrently reside. The host is an initiator that contains CXL Root Ports.
- **Information:** Data or properties of the data exchanged between two participants that would allow the attacker to take or cause an adverse action. Examples include cryptographic keys, questions being considered, decisions of the TEE, results of a database query, etc.
- **Intermediary/switch:** Participant in the protocol that routes or forwards packets to targets. TSP initially focuses on direct attached confidential computing scenarios; thus, switch support in the threat model is beyond the scope of this specification.
- **Participant:** Initiator or target in a communication that utilizes a correct and error free implementation of the protocol.
- **Protocol Secrets:** Secrets that shall be protected, from users of the protocol and/or attackers, to maintain the TEE.
- **Side Channel:** Ability of an attacker to extract information without the knowledge of the participating parties.
- **Trusted Execution Environment (TEE):** Execution environment designed to provide secure separation between itself and any other computation. The environment may include or be extended to multiple devices.

11.5.3.2 Assumptions

The threat model described below is based on the following assumptions:

- CXL does not guarantee that messages arrive in order. It requires initiator ordering. If the initiator has two messages that must be ordered, Message A and Message B, the initiator shall wait until Message A is acknowledged before submitting Message B.
- CXL relies on industry-standard secure protocols: SPDM and PCIe.
- CXL relies on industry-standard capabilities: Secure boot, trusted boot, and attestation.
- There are no errors in the implementation of the protocol, regardless of whether implemented in hardware, software, or firmware.
- An implementation of the protocol shall not disclose protocol secrets to an attacker. The participants shall have a secure location in which to store and/or retain this information.
- For confidential computing, everything inside the TEE shall not be observable by an attacker outside the TEE.

EVALUATION COPY

- When data is securely delivered to an attached target, the target shall protect that data from attacks. The TSP protocol facilitates multiple means of protecting the data. How the device implements such protections is beyond the scope of the threat model and TSP.
- There are non-overlapping resources for distinct hosts.
- Hardware in the host is trusted to maintain protocol separation between TEEs and keep TEEs isolated from one another.
- Hardware in the host is trusted to maintain protocol separation and isolation between Root Ports. Root Ports shall not accept incorrectly formatted transactions. If a host has a single port through which multiple sessions flow, the host hardware shall keep the sessions isolated and reliably deliver the transactions to the session owners, as defined by the hardware configuration.
- A correct implementation of the protocol. This means that the attacker cannot be inside the protocol.
- The protocol cannot defend against an attacker from within the TEE. Host hardware shall be responsible for defending the TEE from an attacker inside a host. An attacker in an initiator or target can use the protocol but is constrained by the requirements of this threat model.
- The target is directly connected to the host; thus, there are no attackers in the intermediaries in the initial TSP threat model. Targets connected via CXL switches have not been evaluated and the presence of switches are considered to be outside the threat model. Fabric-attached memory may require initiator-based memory encryption to keep the intermediaries out of the TCB and shall be addressed in a future version of the TSP.
- TEEs that require confidentiality of the information flowing between the initiator and the target, shall enable a CXL-approved Transport Security such as CXL IDE.
- A target can concurrently hold data for a computation for multiple initiators, resulting in data from multiple participants residing concurrently on the target. The target shall be responsible for keeping each initiator's data or computations separate and isolated. If the target has multiple ports, the target hardware shall keep the ports isolated and independent.
- If a target has a single port through which multiple sessions flow, the target hardware shall keep the sessions isolated and reliably deliver the transactions to their respective session owners, as defined by the hardware configuration. The target determines the threats from which initiator data is protected.
- Initiators and targets shall utilize an SPDM 1.2 or newer connection to authenticate and attest the target. For authenticated and trusted direct attached targets, multiple initiators can communicate with the same target without leaking information to other initiators.
- The protocol shall carry sufficient information to allow the target to maintain separation between initiators. Additionally, the protocol shall contain sufficient information to enable the target to enforce ciphertext hiding if needed.
- The protocol supports both initiator-based and target-based memory encryption so it shall carry sufficient information for the memory device to prevent access by non-TEEs to TEE memory.
- The protocol shall minimize the number of bits transmitted in the clear. These bits can be utilized as a covert channel if an application inside an initiator is compromised. PBR removes this exposure because all address bits can be encrypted; however, this is beyond the scope of the initial TSP.

11.5.3.3 Threats and Mitigations

Table 11-19 outlines the security threats considered as part of the threat model and how the threat is mitigated.

Table 11-19. Security Threats and Mitigations

Primary Threat of Attacker	Threat Mitigation
Extract protocol secrets	<ul style="list-style-type: none"> Transport Security such as CXL IDE TSP initiator-based or target-based memory encryption
Masquerade as a legitimate initiator or target	<ul style="list-style-type: none"> SPDM attestation and authentication SPDM mutual authentication
Insert itself between an initiator and target as a manipulator-in-the-middle	<ul style="list-style-type: none"> Prevent physical attack SPDM attestation and authentication Transport Security such as CXL IDE
Derive actionable data or information from observed packets (side channel)	<ul style="list-style-type: none"> Minimize the number of address bits transmitted in the clear (possibly 0) Transport security such as CXL IDE
Insert data and/or requests/responses into an initiator target communication	Transport Security such as CXL IDE
Modify data and/or requests/responses exchanged in an initiator target communication	Transport Security such as CXL IDE
Remove data and/or requests/responses from an initiator target communication	Transport Security such as CXL IDE
Replay legitimate packets	Transport Security such as CXL IDE
A non-TEE reading or writing of TEE data	TSP TE State checking for access control
A TEE reading or writing unauthorized non-TEE data	TEE is correctly configured to allow access only to memory for which the TEE is authorized to access
One TEE reading or writing another TEE's data	<ul style="list-style-type: none"> TEE is correctly configured to allow access only to memory for which the TEE is authorized to access TSP initiator-based or target-based memory encryption to protect each TEE's data from other TEEs
A TEE on one host reading and/or writing resources that belong to a TEE of another host	<ul style="list-style-type: none"> Hosts shall maintain separation and isolation between TEEs TSP initiator-based or target-based memory encryption
A target on one port being able to send requests and/or responses on another port	Hardware within the host is trusted to maintain protocol separation and isolation between RPs. RPs shall not accept incorrectly formatted transactions. If a host has a single port through which multiple sessions flow, the host hardware shall keep the sessions isolated and reliably deliver the transactions to the session owners, as defined by the hardware's configuration.

11.5.4 Reference Architecture

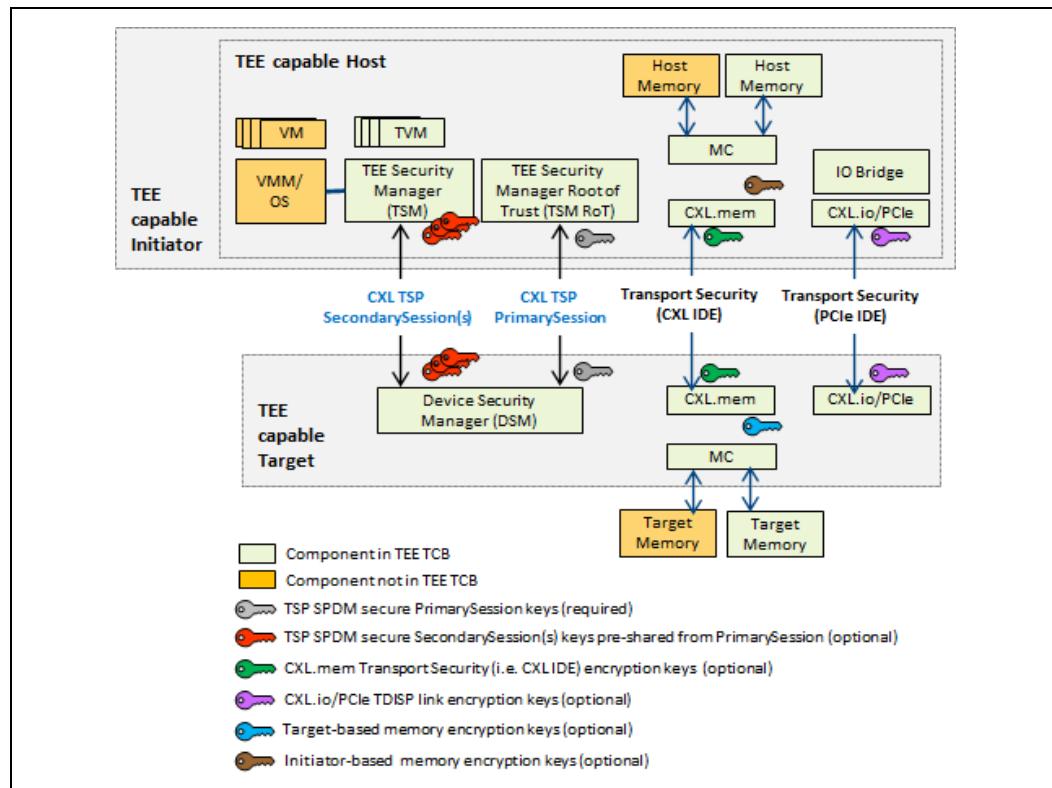
The reference architecture covers the security requirements and behaviors that are needed to support confidential computing use cases and covers the architectural scope, detecting TSP support, CMA/SPDM, attestation and authentication, memory encryption, transport security, access control, configuration, and Dynamic Capacity.

11.5.4.1 Architectural Scope

Figure 11-24 outlines the major components that the TSP considers to be inside the TCB or outside the TCB, the different connections between the TEE-capable initiator and TEE-capable target memory device, and those connections that are specified by the TSP. Hosts are the only initiators defined for this version of the TSP architecture for

support of direct attached confidential computing in the TSP architecture. Not all initiators are hosts because other CXL accelerators and PCIe P2P devices can function as initiators, the focus of future TSP additions.

Figure 11-24. Reference Architecture



For implementations that utilize initiator-based memory encryption or target-based memory encryption, it is recommended to enable Transport Security (such as CXL IDE) as discussed in [Section 11.5.4.7](#).

Securing CXL.io is optional from a TSP perspective.

11.5.4.2 Determining TSP Support

For targets that support the TSP, the DVSEC CXL Capability register TSP Capable bit (see [Section 8.1.3.1](#)) shall be set by the target to indicate support for the TSP requests and responses detailed in the following sections. This bit also indicates to the initiator that the target supports the MemRdFill memory request which is required for deadlock prevention with partial writes and initiator-based encryption.

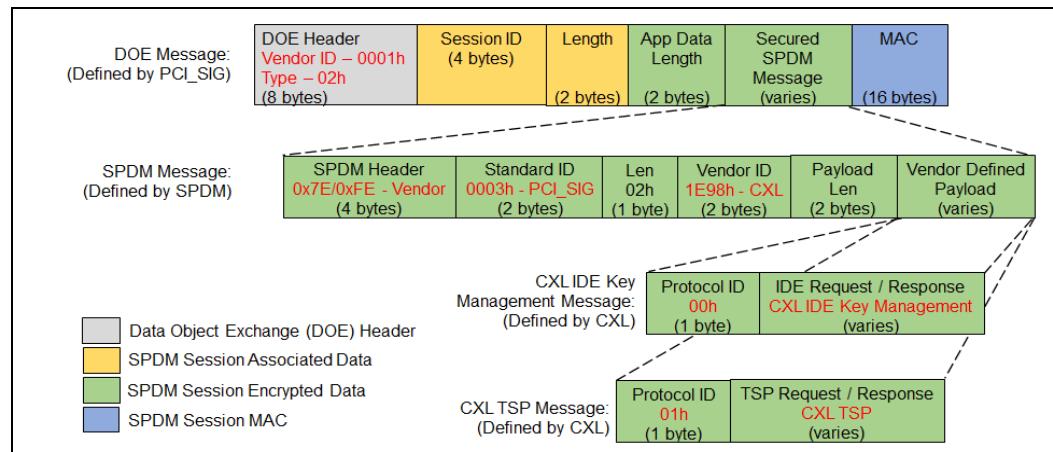
11.5.4.3 CMA/SPDM

CMA/SPDM 1.2 or later secure sessions are utilized with CXL Vendor defined payloads for all TSP request and response payloads defined herein. The Protocol ID in the first byte of the Vendor Defined Payload identifies TSP requests independently from IDE or other requests that may also be defined by CXL. [Figure 11-25](#) outlines the encapsulation of the TSP-defined payloads in a CMA/SPDM message, which is similar to those defined in the PCI-SIG TDISP, with the following changes to establish CXL control of the message interpretation:

- The DOE Data Object type shall report Vendor ID 0001h and Data Object Type 02h to point to Secured CMA/SPDM.
- The CMA/SPDM vendor defined message Standards ID shall utilize 0003h to indicate that PCI-SIG is the body that assigned the CMA/SPDM Message Vendor ID.
- The CMA/SPDM vendor defined message Vendor ID shall utilize 1E98h, indicating that the CXL Consortium assigned the interpretation of the CMA/SPDM vendor defined payload.
- The first byte in the CXL vendor defined payload is the Protocol ID. All CXL.cachemem IDE Key Management requests shall utilize Protocol ID = 00h. All TSP request and response messages shall utilize Protocol ID = 01h.

The encapsulation of TSP requests and responses inside an encrypted SPDM session is shown in [Figure 11-25](#).

Figure 11-25. CMA/SPDM, CXL IDE, and CXL TSP Message Relationship



The Session ID that precedes the CMA/SPDM payload contains the TSP session utilized for each request or response payload. The TSP specification utilizes two types of sessions:

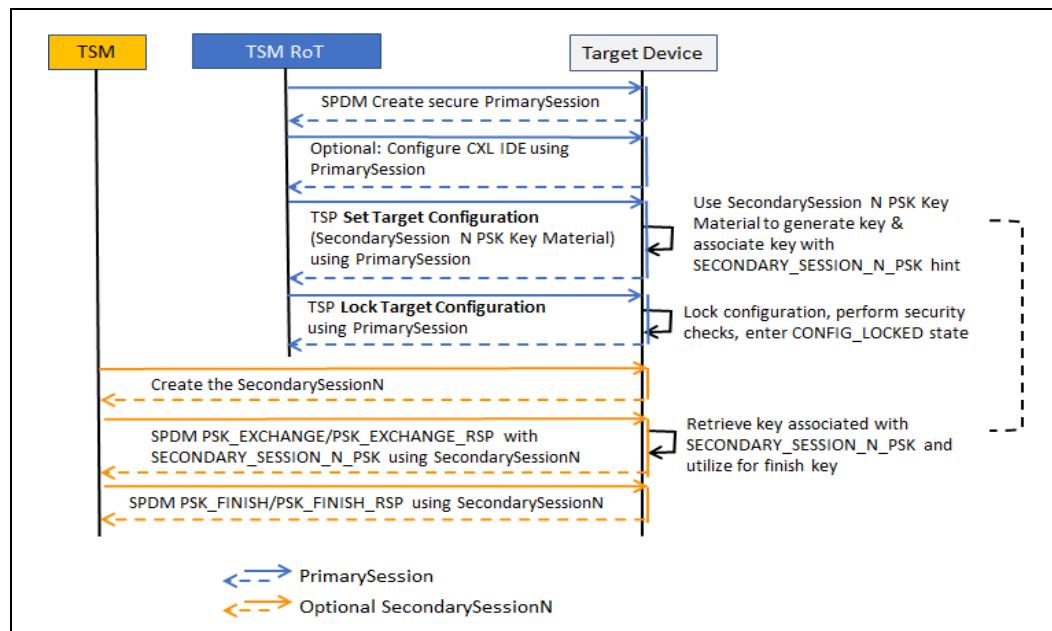
- PrimarySession:** Required CMA/SPDM session that is established between the host and the target.
 - Utilized to configure and lock the target as defined by TSP.
 - For target-based memory encryption, this session may be utilized to set or clear memory encryption keys. The session utilized to set a key shall be the same session that is utilized to clear the same key.
 - PrimarySession is the CMA/SPDM session that is utilized to send the Set Target Configuration request. The target shall terminate any existing SecondarySession(s) anytime a new PrimarySession is established.
 - PrimarySession shall be utilized for all TSP and Transport Security (such as CXL IDE) requests and responses (see [Figure 11-25](#)).
 - Primary SPDM Session shall be utilized to provision PSK Key Material for establishing each Secondary SPDM Session(s)
- SecondarySession(s):** Optional CMA/SPDM sessions that are generated from the PrimarySession by utilizing CMA/SPDM PSK_EXCHANGE between the host and the target.
 - For target-based memory encryption, this session may be utilized to set or clear memory encryption keys. Some host implementations may need to utilize a separate but related SPDM SecondarySession for setting and clearing

memory keys independently from the PrimarySession that is utilized to configure and lock the configuration. The session utilized to set a key shall be the same session that is utilized to clear the same key.

- Target advertises the number of SecondarySession(s) that it supports in Get Target Capabilities.
- Initiator can configure the number of SecondarySession(s) to utilize and the type of TEE opcode checking each will use through Set Target Configuration.
- PrimarySession is independent of these sessions. The termination or closing of any SecondarySession(s) shall have no effect on the PrimarySession.
- These sessions are independent of the PrimarySession. Termination or closing of the PrimarySession shall have no effect on these sessions. However, if a new PrimarySession is started, the target shall terminate these sessions since new secondary sessions will need to be generated based on the new primary session key material.
- CMA/SPDM session that successfully completes the CMA/SPDM PSK that utilizes the correct PSK hint shall be considered a valid SecondarySession.

Figure 11-26 outlines the high-level sequence for creating the PrimarySession and how TMVSession PSK Key Material is utilized by the target to generate keys for a secure CMA/SPDM SecondarySession(s).

Figure 11-26. CMA/SPDM Sessions Creation Sequence



11.5.4.4 Authentication and Attestation

Because the TSP interface requires requests and responses to utilize a CMA/SPDM 1.2 (or later) secure session, target attestation and authentication is accomplished using the CMA/SPDM-defined secure session setup sequence.

11.5.4.5 TE State Changes and Access Control

The TEE Exclusive State (TE State) of memory indicates whether the content of the memory is for TEE or non-TEE data.

Initiators that generate memory accesses shall determine the TEE status of each memory transaction, referred to as the TEE Intent. TEEs are permitted to access both exclusive and non-exclusive memory, while non-TEE entities are permitted to access only memory that is not intended for the exclusive use of a TEE.

Access control is outlined in the following sections and is defined as the verification of the TEE Intent against the TE State of the memory being accessed and the resulting target behavior when the verification fails. Access control is split into Write Access Control and Read Access Control that can be supported by the target independently and enabled independently by the initiator.

Initiators shall not generate memory accesses with TEE Intent if those accesses do not arise within the execution context of a TEE. Initiators that generate memory accesses that originate within the execution context of a TEE shall understand the request's TEE Intent, based on the specific design of the TEE architecture, and shall express the correct TEE Intent. Because each request carries the correct TEE Intent, it is unnecessary for a request to indicate whether it originates from a TEE.

Initiators shall convey TEE Intent in a request by utilizing the TEE-specific M2S Req and M2S Rwd opcodes specified in the CXL Transaction Layer. Targets shall convey the TE state by utilizing the S2M NDR and S2M DRS response opcodes specified in the CXL Transaction Layer. The specific opcodes utilized are specified in the tables provided later in this section.

Hosts that support implicit and explicit TE State changes:

- May enable either mechanism individually or both mechanisms at the same time on the target. When enabling implicit and explicit in-band TE State changes simultaneously, the TE State granularity utilized for explicit in-band TE State changes shall be 64B.
- Shall account for interleaving and send a single TE State change request to each target for a given interleave set.

Targets that have TE State changes enabled:

- Shall change the TE State of memory at a 64B cacheline granularity for implicit changes and at a 64B or greater granularity for explicit changes.
- Shall support explicit in-band TE State changes with a granularity of 64B when supporting implicit TE State changes
- Shall utilize implicit and/or explicit TE State changes as enabled by the host.
- Shall support the TEUpdate memory transaction when implicit or explicit in-band TE State changes are enabled.
- Shall return the current TE State saved for the memory location being accessed

Targets that have TE State changes disabled and CKID-based memory encryption disabled:

- Shall return the TEE Intent from the memory request in the response opcode

Targets that have read and/or write access control enabled:

- Shall implement TE State changes
- Shall follow the rules defined below for implicit or explicit target behavior for updating TE State, verifying TE State, and responding to access control state verification

Targets optionally provide an event log entry of all dropped writes or failed reads that occur in response to failed TE State checks to aid in root-cause analysis of unexpected behavior by reporting a General Media or DRAM Event Record with a Memory Event Type of TE State Violation.

Targets that implement CKID-based target encryption shall perform CKID-type checks as described in [Section 11.5.4.6.2.1](#).

The granularity utilized for TE State changes shall be consistent with the interleave granularity being configured. For example, if the host utilizes a 4K TE State change granularity on each target that is part of a 16-way interleave set with a 256B interleave granularity, each target will utilize 256B of DPA space to change 4K of TE State.

11.5.4.5.1 TEUpdate Memory Transaction

The TEUpdate memory transaction shall utilize the flit's 3-bit SnpType field to provide a Length Index to preconfigured fixed granularities of TE State.

Length Index encodings 1 through 6 are configurable. Length Index encodings 0 and 7 are fixed, where 0 is defined as 64B and 7 is defined as the target's entire memory space.

Targets that support implicit TE State changes or in-band explicit TE State changes shall support this transaction with Length Index = 0 and may support a Length Index of 7 as reported in Get Target Capabilities. If the target only supports implicit TE State changes, then Length Index encodings 1 through 6 shall be reserved.

The HPA present in the TEUpdate transaction shall be decoded by the target to the correct HDM decoder and the starting HPA, HDM decoder Interleave Granularity (IG), and HDM decoder Interleave Ways (IW) are utilized by the target to change the TE State of those HPA ranges within the granularity determined from the SnpType field.

The CKID field is reserved for the TEUpdate transaction and shall be ignored by the target.

There is no mechanism for the target to reject an explicit TEUpdate transaction.

11.5.4.5.2 Implicit TE State Changes

Implicit state changes shall always occur on a cacheline write and shall not utilize Write Access Control.

When utilizing implicit TE State changes, the target shall also support the TEUpdate memory transaction with Length Index 0 to indicate a 64B length.

[Table 11-20](#) outlines the expected target behavior for implementing implicit TE State changes.

Table 11-20. Target Behavior for Implicit TE State Changes

Target's TE State Associated with Memory Access Address	TEE Intent of Memory Transaction Received by the Target	
	TEE Opcodes ¹	Non-TEE Opcodes ²
TE 0	<ul style="list-style-type: none"> Writes: <ul style="list-style-type: none"> Full cacheline write shall cause an implicit state change to TE=1 S2M NDR TEE opcode shall be returned 	<ul style="list-style-type: none"> Writes: <ul style="list-style-type: none"> No change to TE State S2M NDR non-TEE opcode shall be returned
TE 1	<ul style="list-style-type: none"> Writes: <ul style="list-style-type: none"> No change to TE State S2M NDR TEE opcode shall be returned 	<ul style="list-style-type: none"> Writes: <ul style="list-style-type: none"> Full cacheline writes shall cause an implicit state change to TE=0 S2M NDR non-TEE opcode shall be returned

1. MemWrTEE, MemWrPtlTEE.

2. MemWr, MemWrPtl.

11.5.4.5.2.1 Partial Write Handling with Implicit TE State Changes

Full cacheline writes are required to change the TE State implicitly.

Initiator-based memory encryption shall be handled as follows:

- A partial write shall be treated as an under fill read, merging of partial write data with under fill read data, followed by a full cacheline write. The under fill read shall utilize the same TEE intent as the full cacheline write that follows.

Target-based memory encryption shall be handled as follows:

- A partial write shall be treated as an under fill read, merging of partial write data with under fill read data, followed by a full cacheline write. The under fill read shall utilize the same TEE intent as the full cacheline write that follows.
- The under fill read shall follow the rules for reads and the full cacheline write shall follow the rules for writes.

11.5.4.5.3 Explicit TE State Changes

If explicit state changes are supported, the target shall support utilizing the TEUpdate memory transaction for in-band state changes and/or the CMA/SPDM secure session TSP request, Set Target TE State, for out-of-band changes. For explicit TE State changes > 64B, the target shall pre-allocate resources for a single explicit state change request to avoid head-of-line blocking.

The target shall be configured to enable explicit in-band TE State changes or explicit out-of-band TE State changes and either may be enabled individually or both may be enabled at the same time. If the host is utilizing both simultaneously, the host shall maintain coherency between them.

Explicit TE State changes shall be initiated from the host. The host shall ensure that memory affected by the TE state change is flushed from caches before initiating the explicit state change request.

The host shall be responsible for maintaining coherency for accesses to memory ranges that are also executing an explicit state change.

While the explicit TE State change request is executing, the target shall handle memory transactions as follows:

- The target shall continue to process unrelated memory transactions while the state change is executing
- For explicit TE State changes > 64B:
 - For writes to memory ranges that are undergoing the state change, the target shall drop the write and return the inverted TEE Intent in the write completion
 - For reads to memory ranges that are undergoing the state change, the target shall return all 1s and the inverted TEE Intent in the read completion

When executing out-of-band explicit TE State changes that cover a large amount of data, the target may require extra execution time and may utilize the Delayed Response to prevent request timeouts as described in [Section 11.5.5.9](#).

The target shall report optional support to sanitize the contents of memory with 0s anytime the explicit TE State change request is received. If enabled by the host, the target shall complete the overwrite of the affected range before the explicit state change is considered complete. This sanitize capability is reported in Get Target Capabilities and the host may enable its use with Set Target Configuration. Sanitizing large amounts of memory may require extra execution time and targets may utilize the Delayed Response to prevent request timeouts as described in [Section 11.5.5.9](#).

If in-band and out-of-band explicit state changes overlap, the host shall ensure that those requests have non-overlapping address ranges. Otherwise, an indeterminate result could occur. If the target can detect this overlap, the target should generate an appropriate event record to aid in debug.

For explicit updates > 64B: A single explicit state change shall be sent to every target in the same interleave set. Explicit state changes specify a starting address and length that cover the entire range to be changed. The target shall change the state for all portions of the range that land on its portion of the interleave.

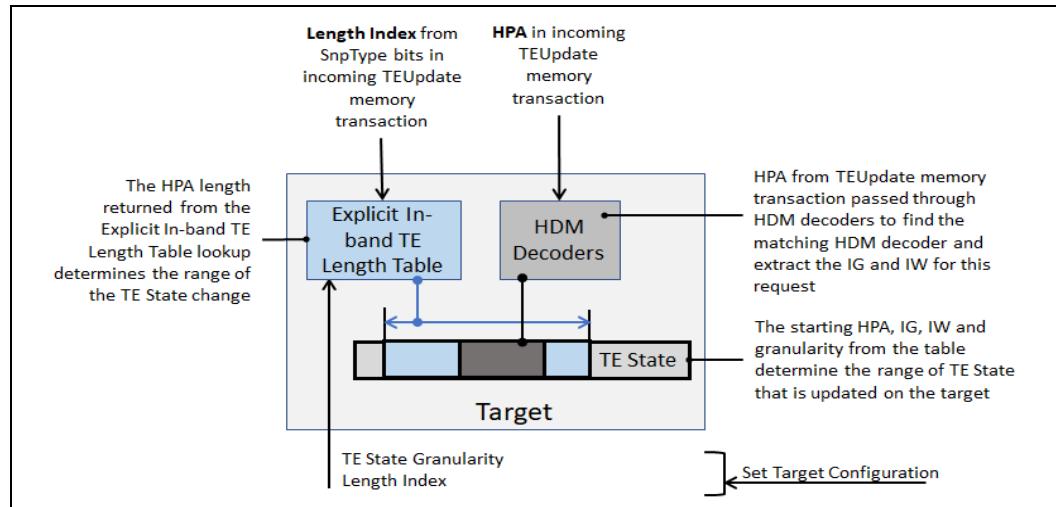
11.5.4.5.3.1 Optional Explicit In-band TE State Change

For explicit in-band TE State changes:

- The in-band mechanism shall utilize the TEUpdate memory transaction.
- The association of length index in the SnpType field to a given granularity is configured by the initiator utilizing Set Target Configuration. This is done to minimize the size of the TEUpdate flit to a single slot to minimize Transaction Layer complications.
- Length Index value of 0 is reserved for 64B state changes.
- Length Index value of 7 is reserved for state changes affecting the entire memory space of the target.
- Length Index values 1-6 are host configurable to any supported length utilizing Set Target Configuration.
- If the in-band TE State change granularity is > 64B, the host shall only issue a single explicit in-band state change request at a time.
- If the in-band TE State change granularity is 64B, the host may issue multiple explicit in-band TE State change requests to non-overlapping address ranges and the target shall queue those requests waiting to execute.

[Figure 11-27](#) outlines the association between the Explicit In-band TE State Granularity specified in Set Target Configuration and the Length Index specified in the TEUpdate transaction SnpType field.

Figure 11-27. Optional Explicit In-band TE State Change Architecture



[Table 11-21](#) outlines the expected target behavior for utilizing explicit in-band TE State changes.

Table 11-21. Target Behavior for Explicit In-band TE State Changes

Target's TE State Associated with Memory Access Address	TEE Intent of Memory Transaction Received by the Target
	TEE Opcodes ¹
TE 0	<ul style="list-style-type: none"> • TEUpdate(TE = 0): <ul style="list-style-type: none"> — No change in TE state • TEUpdate(TE = 1): <ul style="list-style-type: none"> — This shall cause an explicit state change to TE=1 for the affected memory granularity
TE 1	<ul style="list-style-type: none"> • TEUpdate(TE = 0): <ul style="list-style-type: none"> — This shall cause an explicit state change to TE=0 for the affected memory granularity • TEUpdate(TE = 1): <ul style="list-style-type: none"> — No change in TE state

1. TEUpdate.

11.5.4.5.3.2 Optional Explicit Out-of-Band TE State Change

For explicit out-of-band TE State changes:

- Out-of-band mechanism utilizes the Set TE State TSP request and supports a robust set of possible TE State change granularities reported in Get Target Capabilities that cannot be utilized with the limitations of the in-band mechanism
- Host shall only issue a single explicit out-of-band state change request at a time
- Target shall reject any request to set TE State when another TE State change request is already executing

11.5.4.5.4 Write Access Control

Table 11-22 outlines the required target behavior when Write Access Control is enabled on the target.

Table 11-22. Target Behavior for Write Access Control

Target's TE State Associated with Memory Access Address	TEE Intent of Memory Transaction Received by the Target	
	TEE Opcodes ¹	Non-TEE Opcodes ²
TE 0	<ul style="list-style-type: none"> • Writes: <ul style="list-style-type: none"> — Full/Partial cacheline write shall be dropped — S2M NDR non-TEE opcode shall be returned — Optionally log event 	<ul style="list-style-type: none"> • Writes: <ul style="list-style-type: none"> — Full/Partial cacheline write allowed — S2M NDR non-TEE opcode shall be returned
TE 1	<ul style="list-style-type: none"> • Writes: <ul style="list-style-type: none"> — Full/Partial cacheline write allowed — S2M NDR TEE opcode shall be returned 	<ul style="list-style-type: none"> • Writes: <ul style="list-style-type: none"> — Full/Partial cacheline write shall be dropped — S2M NDR TEE opcode shall be returned — Optionally log event

1. MemWrTEE, MemWrPtlTEE.

2. MemWr, MemWrPtl.

If Write Access Control is not enabled on the target, the target shall not check write requests for possible access control violations. See [Section 11.5.4.5.2](#) for implicit TE State changes and required target behavior.

If Write Access Control is enabled on the target, the target shall clear the TE State to 0 for all addressable memory in response to the Lock Target Configuration Request and before generating a Lock Target Configuration Response.

Write Access Control requires the target to also support explicit TE State changes. The target shall reject attempts to enable Write Access Control without one or more explicit TE State change mechanisms also being enabled.

The target shall not perform Write Access Control when updating MetaValue. See [Section 11.5.4.5.6](#) for requirements for handling MetaValue updates.

If enabled, the target shall perform Write Access Control when updating Extended Metadata (EMD). See [Section 11.5.4.5.7](#) for requirements for handling EMD updates.

11.5.4.5.4.1 Partial Write Handling with Write Access Control

Target-based memory encryption shall be handled as follows:

- A partial write shall be treated as an under fill read, merging of partial write data with under fill read data, followed by a full cacheline write.
- The under fill read shall follow the rules for reads and the full cacheline write shall follow the rules for writes. In case of a TEE mismatch between the TE State obtained in the under fill read and the TEE Intent of the request, the target shall drop the write.

11.5.4.5.5 Read Access Control

[Table 11-23](#) outlines the required target behavior when Read Access Control is enabled on the target.

Table 11-23. Target Behavior for Read Access Control

Target's TE State Associated with Memory Access Address	TEE Intent of Memory Transaction Received by the Target	
	TEE Opcodes ¹	Non-TEE Opcodes ²
TE 0	<ul style="list-style-type: none"> • Reads: <ul style="list-style-type: none"> — Reads shall return fixed data of all 1s — S2M DRS non-TEE opcode shall be returned — Optionally log event 	<ul style="list-style-type: none"> • Reads: <ul style="list-style-type: none"> — Allowed — S2M DRS non-TEE opcode shall be returned
TE 1	<ul style="list-style-type: none"> • Reads: <ul style="list-style-type: none"> — Allowed — S2M DRS TEE opcode shall be returned 	<ul style="list-style-type: none"> • Reads: <ul style="list-style-type: none"> — Reads shall return fixed data of all 1s — S2M DRS TEE opcode shall be returned — Optionally log event

1. MemRdTEE, MemRdDataTEE, MemRdFillTEE, and MemSpecRdTEE.

2. MemRd, MemRdData, MemRdFill, and MemSpecRd.

If Read Access Control is not enabled on the target, the target shall not check read requests for possible access control violations.

The target shall not perform Read Access Control when updating MetaValue. See [Section 11.5.4.5.6](#) for requirements for handling MetaValue updates.

If enabled, the target shall perform Read Access Control when updating Extended Metadata (EMD). See [Section 11.5.4.5.7](#) for requirements for handling EMD updates.

11.5.4.5.6 MetaValue Updates for HDM-H

MetaValue is a property of the memory address and unrelated to any of the data associated with that address. Consequently, the target shall not perform access control checks on MetaValue updates.

TEUpdate:

- TEUpdate uses the MetaValue to convey the TE State and does not update MetaValue

For all other (non-TEUpdate) transactions:

- There is no TE State associated with MetaValue
- Targets that implement access control shall ignore access control checks when updating the MetaValue and shall allow MetaValue updates, even if the associated read or write request fails access control checks.

Initiators are responsible for ensuring that changes to MetaValue do not negatively affect data coherency. How an initiator guarantees this is beyond the scope of CXL.

11.5.4.5.7 Extended Metadata Updates

Extended Metadata (EMD) is a property of the data and is updated using the same flows and transactions as data. Consequently, TE State and access control, if enabled, shall be utilized when updating EMD.

If Write Access Control checks fail, the target shall not update EMD.

If Read Access Control checks fail, the target shall return fixed data of all 1s for EMD.

11.5.4.6 Memory Encryption

Protecting data at rest in the target memory device is required for confidential computing and requires memory encryption. The TSP supports both initiator-based and target-based memory encryption capabilities, and an initiator shall utilize one or the other when adding a target to the TCB. If target-based memory encryption is not enabled by the initiator, it shall be the initiator's responsibility to utilize initiator-based memory encryption.

This version of the TSP specification is focused on direct attached CXL devices. In such a configuration, there is a single initiator for a given memory range and all transactions to that memory region shall flow through that initiator. When initiator-based encryption is utilized, it is the initiator's responsibility to ensure that the cryptographic keys are correct for all initiators and to maintain data coherency. Initiator-based encryption implementations that utilize partial writes require the initiator to perform a read of a complete cacheline, update the corresponding bytes, and write the complete cacheline back to the target. The target cannot modify any blocks that are encrypted by an initiator and thus cannot perform the RMW (read-modify-write) that is required to perform partial writes.

Target-based encryption requires Transaction Layer and Link Layer protocol changes to pass a CKID to the target so that the target can correctly choose the key that is utilized for the encryption/decryption of data associated with each transaction.

Both initiator-based and target-based memory encryption are optional. The target reports its supported memory encryption capabilities, and the initiator selects the memory encryption that it needs to utilize.

11.5.4.6.1 Initiator-based Memory Encryption

Initiator-based memory encryption may be utilized, independent of target-based memory encryption being enabled or disabled, as follows:

- Target shall support the MemRdFill memory request, which is required for deadlock prevention with partial writes and initiator-based encryption. Support for this request is a target requirement when indicating TSP Capable support in the DVSEC CXL Capability register.
- If target-based CKID memory encryption is not enabled, the CKID field in the memory transaction is reserved and ignored by the target.

11.5.4.6.2 Target-based Memory Encryption

Targets are not required to implement memory encryption. When memory encryption is implemented on the target, the following applies:

- Encryption shall be implemented using one of the Memory Encryption Algorithms Supported that are reported in Get Target Capabilities Response.
- TSP supports two target-based memory encryption mechanisms:
 - CKID-based encryption requires use of the CKID field in the Transaction Layer to identify a specific key to be utilized when encrypting/decrypting memory contents for a given transaction. This feature requires the use of TEE opcodes in the memory transaction to perform the proper CKID type checking.
 - Range-based encryption utilizes memory range registers that are configured to associate a particular encryption key with a specific memory range and does not rely on the CKID field in the transaction.
 - Host may enable CKID-based or range-based target memory encryption, but shall not enable both.
 - Target shall reject attempts to enable both target-based encryption methods.

11.5.4.6.2.1 CKID-based Memory Encryption

When CKID-based target memory encryption is implemented, the following applies:

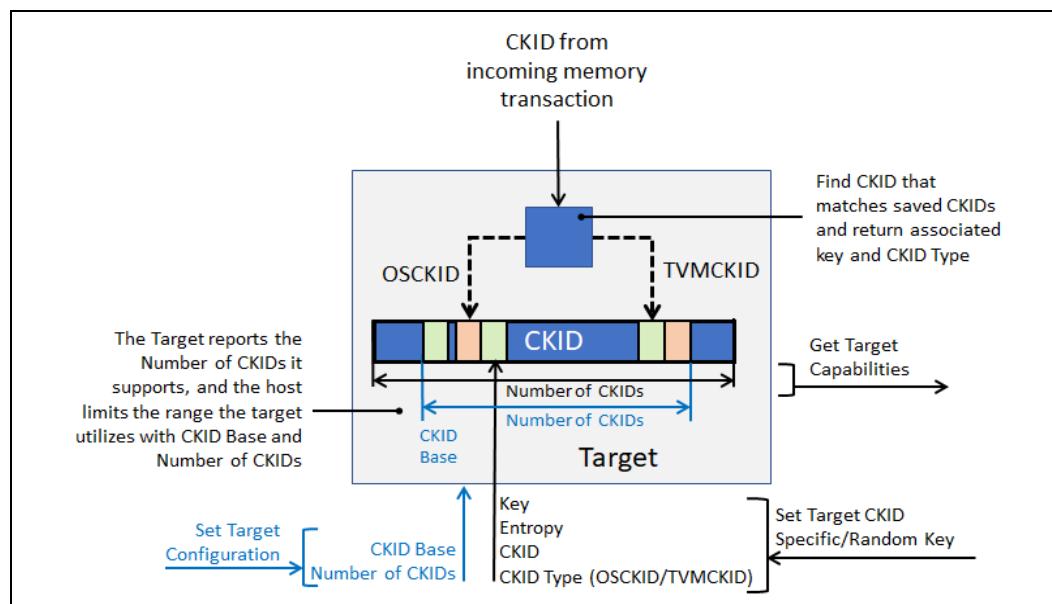
- For CKID-based encryption, the initiator and target shall utilize the CKID field in the Transaction Layer for memory requests.
- Each host Root Port has a unique 13-bit CKID value to be utilized across all targets. A host may choose to share CKID values across Root Ports.
- Target shall accept the entire range of the 13-bit CKID field defined in the Transaction Layer.
- CKIDs that are configured on each target can start and end anywhere within the CKID space supported by the protocol.
- Target reports the Number of CKIDs that it supports in Get Target Capabilities. How the target maps the supported number of CKIDs to the 13-bit CKID field in the transaction is target implementation specific.
- Number of CKIDs that the target supports may be sparsely distributed across the 13-bit range, or optionally the target may require that the CKIDs be assigned a contiguous range starting at a specific CKID Base.
- If the target does not require a CKID Base:
 - CKID specified in Set Target CKID Specific Key, Set Target CKID Random Key, and Clear Target CKID Key shall not cause the target to utilize more than the Number of CKIDs that the target supports
- If the target requires a CKID Base to be utilized:
 - Target shall indicate that a range of CKIDs using a CKID Base and Number of CKIDs is required in Get Target Capabilities
 - Host shall configure a contiguous range of CKIDs on the target by specifying the CKID Base and Number of CKIDs that the target shall utilize with Set Target Configuration. The Number of CKIDs enabled by the host shall be \leq Number of CKIDs reported by the target.
 - CKID in the Transaction Layer memory requests from the initiator shall be CKID Base \leq CKID $<$ CKID Base + Number of CKIDs configured on the target. See [Table 11-24](#) for the target behavior if the CKID in the transaction is outside the configured CKID of the target.

EVALUATION COPY

- CKID specified by the host in Set Target CKID Specific Key, Set Target CKID Random Key, and Clear Target CKID Key shall be CKID Base <= CKID < CKID Base + Number of CKIDs configured on the target.
- Initiator may configure a specific CKID to a specific initiator-supplied key by utilizing Set Target CKID Specific Key.
- Initiator may configure a specific CKID to a random target-generated key with optional initiator-supplied key entropy by utilizing Set Target CKID Random Key.
- Initiator may clear a previously configured key to allow the CKID to be recycled for another key by utilizing Clear Target CKID Key sent on the same session that was utilized to set the key. If the session utilized to set the key has terminated or closed then the target may need to be reset to break the memory encryption key to CKID association.
- Target implements the mapping of keys to CKIDs.
- Each CKID shall be a TVMCKID or OSCKID that is configured utilizing the CMA/SPDM PrimarySession or SecondarySession(s) (see [Figure 11-28](#), which describes this partitioning). See the description below on how the target utilizes the CKID Type to verify memory transactions and the behavior of the target when the verification fails.
- TSM shall be responsible for setting TVMCKIDs.
- When utilizing target CKID-based memory encryption, the initiator's transactions that contain TVMCKIDs shall utilize TEE opcodes and those that contain OSCKIDs shall utilize non-TEE opcodes. The target shall verify that the opcodes are correct for accessing each CKID Type.
- If CKID target based encryption has not yet been enabled, the CKID field in the memory transaction is reserved and ignored by the target and no encryption shall be performed by the target until it is successfully enabled.

[Figure 11-28](#) demonstrates the target mixing of TVMCKIDs and OSCKIDs in a single range of possible CKIDs that utilize the CKID Base and Number of CKIDs.

Figure 11-28. CKID-based Memory Encryption Utilizing CKID Base



The target stores the CKID Type along with the CKID and key. The target shall verify that each incoming memory transaction passes the following checks:

- The memory transaction CKID is within the valid configured range of the target
- The memory transaction has a non-TEE opcode and the CKID is an OSCKID
- The memory transaction has a TEE opcode and the CKID is a TVMCKID

If any of these checks fail, the target should provide an event log entry to aid in root-cause analysis of unexpected behavior by reporting a General Media or DRAM Event Record with Memory Event Type of CKID Violation.

[Table 11-24](#) outlines the target behavior for memory transactions that request a CKID that is outside the configured range of the target.

Table 11-24. Target Behavior for Invalid CKID Ranges

CKID	Memory Transaction Received by the Target	
	Read Opcodes	Write Opcodes
Within Configured Range	<ul style="list-style-type: none"> • Allowed 	<ul style="list-style-type: none"> • Allowed
Outside Configured Range	<ul style="list-style-type: none"> • Reads shall return fixed data of all 1's • Optionally log event 	<ul style="list-style-type: none"> • Writes shall be dropped • Optionally log event

[Table 11-25](#) outlines the target behavior for the CKID-type verification checks based on TEE Intent using TEE opcodes and non-TEE opcodes, which is described in [Section 11.5.4.5](#).

Table 11-25. Target Behavior for Verifying CKID Type

Target's CKID type Associated with CKID Contained in Memory Transaction	TEE Intent of Memory Transaction Received by the Target	
	TEE Opcodes ¹	Non-TEE Opcodes ²
OSCKID	<ul style="list-style-type: none"> • Writes: <ul style="list-style-type: none"> — Writes shall be dropped — S2M NDR non-TEE opcode shall be returned — Optionally log event • Reads: <ul style="list-style-type: none"> — Reads shall return fixed data of all 1s — S2M DRS non-TEE opcode shall be returned — Optionally log event 	<ul style="list-style-type: none"> • Writes: <ul style="list-style-type: none"> — Full/Partial cacheline write allowed — S2M NDR non-TEE opcode shall be returned • Reads: <ul style="list-style-type: none"> — Allowed — S2M DRS non-TEE opcode shall be returned
TVMCKID	<ul style="list-style-type: none"> • Writes: <ul style="list-style-type: none"> — Full/Partial cacheline write allowed — S2M NDR TEE opcode shall be returned • Reads: <ul style="list-style-type: none"> — Allowed — S2M DRS TEE opcode shall be returned 	<ul style="list-style-type: none"> • Writes: <ul style="list-style-type: none"> — Writes shall be dropped — S2M NDR TEE opcode shall be returned — Optionally log event • Reads: <ul style="list-style-type: none"> — Reads shall return fixed data of all 1s — S2M DRS TEE opcode shall be returned — Optionally log event

1. MemWrTEE, MemWrPtlTEE, MemRdTEE, MemRdDataTEE, MemRdFillTEE, and MemSpecRdTEE.

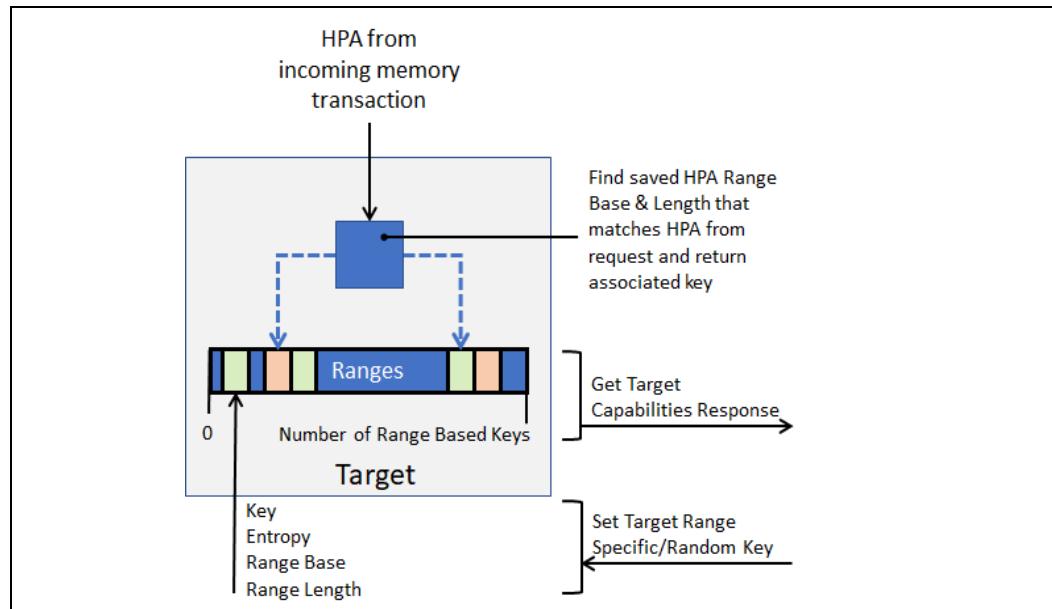
2. MemWr, MemWrPtl, MemRd, MemRdData, MemRdFill, and MemSpecRd.

11.5.4.6.2.2 Range-based Memory Encryption

- The initiator may configure a specific HPA memory range to use a specific initiator-supplied key that utilizes Set Target Range Specific Key.
- The initiator may configure a specific HPA memory range to use a random target-generated key with optional initiator-supplied key entropy that utilizes Set Target Range Random Key.
- The initiator may clear a previously configured key to allow the memory range to be recycled for another key by utilizing Clear Target Range Key. If the session utilized to set the key has terminated or closed then the target may need to be reset to break the memory encryption key to memory range association.
- The target advertises the Memory Encryption Number of Range Based Keys that it supports in Get Target Capabilities, and the initiator assigns a Range ID and HPA range at the time the key is programmed. The target shall verify that the HPA memory ranges and Range IDs do not overlap. The HPA contained within the memory request shall be compared to the programmed memory ranges by the target to retrieve the correct key to utilize in the encryption. The Range ID specified in Set Target Range Specific Key and Set Target Range Random Key shall be $0 \leq \text{Range ID} < \text{Memory Encryption Number of Range Based Keys}$.
- The PrimarySession or SecondarySession(s) shall be utilized for setting and clearing range based keys. The session utilized to set a range based key shall be the same session utilized to clear the same range based key.

Figure 11-29 outlines range-based memory encryption utilizing the HPA.

Figure 11-29. Range-based Memory Encryption



11.5.4.7 Transport Security

Transport security is optional for TSP. If supported and enabled, a CXL-approved Integrity and Data Encryption (IDE) mechanism shall be used. It is up to the TEE policy to decide whether targets that do not support transport security invalidates their use for confidential computing scenarios.

Currently hop-by-hop CXL IDE is the only defined CXL transport security mechanism, and determining the target's IDE capabilities or enabling IDE modes is done through CXL IDE-defined registers. There are no Transport Security-specific TSP interfaces defined at this time.

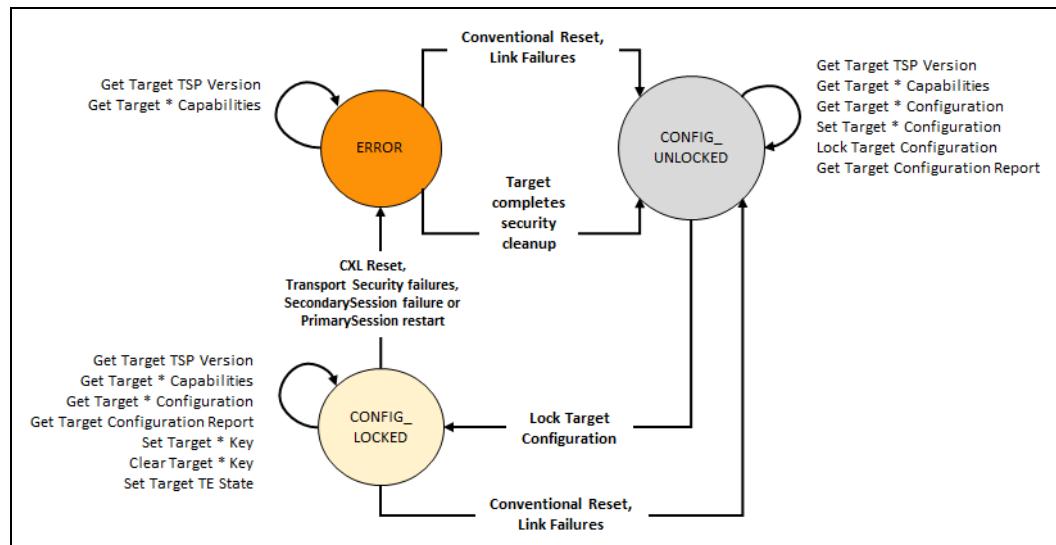
Warning: Although Transport Security is optional, disabling of Transport Security adds additional exposure to physical attacks such as device removal and manipulator-in-the-middle because the binding provided by Transport Security is not utilized. If these attacks are considered part of the threat model for the TEE and cannot be protected via other methods, then Transport Security shall be enabled for confidential computing scenarios.

11.5.4.8 Configuration

The PrimarySession shall be utilized to configure the security for each target in the CXL hierarchy that utilizes TSP, supported CXL Transport Security mechanisms, and other interfaces to setup the TCB. After the targets are correctly configured, the PrimarySession shall be utilized to lock the target to disable initiator access to registers that could cause data coherency issues, loss of data, reveal TVM data to an untrusted VM, and/or otherwise compromise the TCB. After the target is locked, trusted memory transactions are allowed, and the target or initiator shall perform access verification by utilizing the TE State.

Figure 11-30 outlines the defined target security states that are utilized by TSP.

Figure 11-30. Target TSP Security States



- CONFIG_UNLOCKED
 - Default security state after Conventional Reset (see [Section 11.5.4.8.3](#) for details).
 - TSP security configuration is performed in this state.
 - TEE opcode transactions are not allowed by the target in this state (see [Section 11.5.4.10.1](#)).
 - Non-TEE opcode transactions are allowed as discussed in [Section 11.5.4.5](#).
 - Transition to CONFIG_LOCKED state:
 - After successfully locking the target.

- CONFIG_LOCKED
 - Restrictions placed on register accesses, CCI commands to protect the TCB. See the Lock Target Configuration interface description (see [Section 11.5.5.6.7](#) and [Section 11.5.5.6.8](#)) for more details on target behavior after locking.
 - Target shall save TE State and if enabled, enforce TE State checking.
 - Assigning keys to CKIDs or memory ranges, clearing of keys allowed in this state.
 - Non-TEE opcode transactions are allowed as discussed in [Section 11.5.4.5](#).
 - TEE opcode transactions allowed as discussed in [Section 11.5.4.5](#).
 - Transition to ERROR state:
 - The following are treated as errors that shall transition the target to the ERROR state: Transport Security failures (e.g., CXL IDE becoming insecure), CXL Reset (which doesn't reset the entire device), and/or other cases in which the link can no longer be trusted or only a portion of the device is affected by the error (see [Section 11.5.4.8.3](#) for details).
 - Transition to CONFIG_UNLOCKED state:
 - Upon receipt of a Conventional Reset (see [Section 11.5.4.8.3](#) for details).
- ERROR
 - The target shall continue to protect all TVM data when in the error state.
 - After the target enters the error state, the target shall stop accepting all future TEE memory transactions. For transactions that were accepted prior to going to the error state, it is permissible to handle those transactions as normal. See [Section 11.5.4.8.3](#) for details.
 - Non-TEE opcode transactions are allowed as discussed in [Section 11.5.4.5](#).
 - Transition to CONFIG_UNLOCKED state:
 - The target shall automatically transition from the ERROR to the CONFIG_UNLOCKED state after it has cleaned up the current secure sessions and data (see [Section 11.5.4.8.3](#) for details).
 - Upon receipt of a Conventional Reset (see [Section 11.5.4.8.3](#) for details).

Use of these states and the target's behavior in each state are further detailed in [Section 11.5.5](#).

- The TSP architecture assumes that a single authority model shall be utilized for configuration and locking of the target:
 - The host owns all configuration policies
 - The host establishes and locks the security configuration for all targets within its domain
 - MH-SLDs implement separate target configurations for each host, thereby allowing each host to independently configure its SLDs

11.5.4.8.1 Locking the Target

As a pre-condition to performing the memory security checks, the DSM shall first lock the configuration to ensure that it cannot be modified during or after completion of the memory security checks. The mechanisms used by the DSM to lock the memory controller configuration, HDM decoder configuration, and other configuration context specific to the target micro-architecture are beyond the scope of this specification.

EVALUATION COPY

The target shall implement configuration and security checks that verify the locked configuration before successfully responding to the lock configuration request. Memory security checks are mechanisms that the target implements to verify whether its configuration is locked down and are acceptable to meet the confidential computing security objectives of protecting the TVM data. The TSM RoT relies on the DSM for these memory security checks.

- General Assumptions:
 - Any target operation mode that exposes internal data or allows data logging or tracing is disabled
 - Any register that can lead to data corruption shall be locked for writing.
 - Registers leading to reset events that are guaranteed to transition the DSM to ERROR or CONFIG_UNLOCKED state are not required to be locked for writing.
 - Reading of security-sensitive registers shall be blocked. Reading of non-security-sensitive registers do not need to be blocked.
 - Transport Security cannot be assumed to be enabled.
- As part of the memory security checks, the DSM shall ensure the following:
 - HDM decoders in the target are configured consistently and with no aliases. These decoders are specific to the target's implementation. An alias is present if two HPAs decode to the same DPA.
 - Memory controllers and other logic in the target are configured consistently to meet the confidential computing security objectives. Such configuration shall be specific to the target implementation. Examples of such configuration include the DIMM population registers, interleave configuration, error-detection capabilities, ECC mode configuration, debug capabilities such as error/pattern injection logic, target row refresh controls, etc.
 - HDM decoders of the LD are in a consistent state and do not have aliases or overlaps. The HDM decoders shall be locked with the Lock on Commit bit in the HDM Decoder Control register set to 1.
 - Addresses that are decoded by the HDM decoders shall not overlap with addresses that are decoded as PCIe memory space for MMIO. The target shall prevent the overlapping of PCIe BARs.
 - Error-detection capabilities required to ensure security of TVM data shall be enabled in the Uncorrectable Error Mask register of the CXL RAS Capability structure.
 - CXL link-specific registers for the enabled TSP Transport Security mechanism control registers are programmed with parameters that are validated to be safe to support the confidential computing security objectives.
 - If the target is configured to utilize Transport Security, all target CXL.cachemem links shall have Transport Security enabled to protect the security of the transport.
 - PCIe DVSEC for CXL target shall be locked by asserting CONFIG_LOCK.
 - PCIe DVSEC for Test Capability, if implemented, shall be locked by asserting TestLock and no test algorithms, test capabilities, and/or error injection methods are currently active.
 - No test algorithms, test capabilities, and/or error injection methods are currently active or configured through the compliance DOE mailbox.
 - Other target implementation-specific configuration checks as defined by the micro-architecture of the target.
 - CXL Capabilities are configured on the target to prevent the leaking of cipher text from the target after a CXL Reset, as discussed in [Section 11.5.4.8.3](#).

EVALUATION COPY

- Successfully locking the target should result in the following target behavior:
 - The TEE's memory range associated with the target is locked by making the HDM decoder configuration immutable after the target is locked.
 - The target shall limit supported TSP requests to the subset of requests that are allowed when the target is in the CONFIG_LOCKED state. See [Table 11.5.5.1](#) for the TSP requests that are allowed on a locked target.
 - Prevention of surprise changes to the target configuration that would allow unauthorized access to data that was written by a TVM, cause the target or initiator to break the data coherency model, and/or otherwise compromise TEE integrity. This may require the target to drop host writes to PCIe or CXL registers after the target is locked.
 - The Get Target Configuration Report request and response (see [Section 11.5.5.6.6](#)) shall allow the initiator to securely retrieve and verify the content of specific PCIe and CXL configuration registers, after the target is locked. This allows the TSM RoT to check the configuration, securely independent of standard insecure PCIe or CXL register accesses.
 - The following actions shall be prevented by making the associated registers immutable after the target is locked:
 - Memory aliasing
 - CXL HDM Decoder Global Control register: Changing the HDM Decoder Enable state at runtime could allow an attacker to change the address for a write transaction, potentially affecting the data coherency and/or the TE State maintained on the target.
 - CXL HDM Decoder [n] Low/High, Size Low/High, and DPA Skip Low/High Registers: Changing the HDM decoder programming at runtime could allow an attacker to change the address for a write transaction, potentially affecting the data coherency and/or the TE State maintained on the target.
 - The host shall lock all target HDM decoders that were programmed for interleave sets that are within the TEE, utilizing the Commit On Lock feature. Register writes to locked HDM decoders are dropped.
 - Writes to HDM decoders that are unlocked, not already programmed (size and skip = 0), and not considered part of the TEE, need to be allowed by the target. However, the target shall perform alias checking of those register writes to ensure that they do not alias address ranges that are already locked and considered to be in the TEE, independent of whether the host has set the Commit On Lock bit during decoder programming. The target shall treat aliased HDM decoders as a programming failure and shall behave the same way as if the Commit On Lock bit was set when the HDM decoder alias checks failed. The failed checks shall have no effect on the target, the TEE, and/or in-flight transactions between them.
 - Altering target behavior at runtime
 - CXL HDM Decoder Global Control register: Disabling the Poison On Decode Error Enable bit at runtime could cause an attacker to affect the data coherency of the target and allow the initiator to potentially consume invalid data and/or a fixed data pattern as valid data.
 - CXL Control register.
 - CXL Range 1 Base High/Low registers.
 - CXL Range 2 Base High/Low registers.
 - Altering TEE memory interleave configuration at runtime

- CXL HDM Decoder [n] Control register(s): Altering IG, IW, Range Type, BI, UIO, UIG, UIW, and/or ISP at runtime could allow an attacker to steer transactions to another target and/or memory range, potentially affecting the data coherency and/or the memory integrity of the TE State stored in the target.
- Altering link or Transaction Layer behavior at runtime
 - PCIe DVSEC for Flex Bus Port registers.
 - CXL Timeout and Isolation Control register.
 - CXL IDE Control register (if utilizing CXL IDE for Transport Security): Altering PCRC Disable or IDE.StopEnable at runtime could allow the link to operate incorrectly if there is a mismatch with the initiator. This should result in a MAC error which should cause the IDE on the link to transition to Insecure State (if IDE is enabled) and is indicated in the CXL IDE Error Status register, bit[1] and bit[7]. When IDE transitions to Insecure State, the target shall transition to the ERROR state.
- Prevention of CCI mailbox commands that could allow untrusted access to data that was written by a TVM, cause the target or initiator to break the data coherency model, and/or otherwise compromise TEE integrity (see [Section 11.5.4.9](#) for details).
- TEE opcodes shall be allowed by the target as outlined in the Access Control section (see [Section 11.5.4.5](#)).
- Enforce Read and/or Write Access Control if enabled (see [Section 11.5.4.8](#)).
- Prevention of surprise changes to the target configuration for any target implementation-specific or vendor-implementation-specific registers or CCI mailbox commands that could allow untrusted access to data that was written by a TVM, cause the target or initiator to break the data coherency model, and/or otherwise compromise TEE integrity. This requires implementation-specific analysis and is beyond the scope of the TSP.

11.5.4.8.2 Considerations for Securing the Host

- The host is responsible for securing the Root Ports to secure the TEE environment, and such mechanism is beyond the scope of the TSP.
- The host should prevent surprise changes to the host configuration that would allow unauthorized access to data that was written by a TVM, cause the target and/or host to break the data coherency model, and/or otherwise compromise TEE integrity.
- Some PCIe and CXL Root Port registers that should be protected by the host are outlined here (this is not an exhaustive list, but an example of some of the registers that affect the initiator's link to the locked target):
 - Memory aliasing
 - CXL HDM Decoder Global Control register: Changing the HDM Decoder Enable state at runtime could allow an attacker to change the address for a write transaction, potentially affecting the data coherency and/or the TE State maintained on the target.
 - CXL HDM Decoder [n] Low/High, Size Low/High, and Target List Low/High registers: Changing the HDM decoder programming at runtime could allow an attacker to change the address for a write transaction, potentially affecting the data coherency and/or the TE State maintained on the target.
 - The host shall lock all HDM decoders that were programmed for interleave sets that are within the TEE, utilizing the Commit On Lock feature. Register writes to locked HDM decoders are dropped.

- Writes to HDM decoders that are unlocked, not already programmed (size and skip = 0), and not considered part of the TEE need to be allowed by the initiator.
- Altering host behavior at runtime
 - CXL HDM Decoder [n] Global Control register: Disabling the Poison On Decode Error Enable bit at runtime could cause an attacker to affect the data coherency of the target and allow the host to potentially consume invalid data and/or a fixed data pattern as valid data.
 - CXL BI Decoder Control register: Toggling of BI Enable during runtime could affect data coherency of HDM-DB memory. If BI Decoder Commit is not toggled after reassigning bus numbers, data could be steered to the wrong location, thereby causing HDM-DB memory coherency issues.
 - CXL Cache ID RT Control register: If Cache ID RT Commit toggles at runtime and the Route Table contains invalid routing information, this could cause transactions to be steered to the wrong location, thereby causing data coherency issues.
 - CXL Cache ID Target register: Not toggling Valid and Port Number after a Route Table change could cause transactions to be steered to the wrong location, thereby causing data coherency issues.
- Altering memory interleave configuration at runtime
 - CXL HDM Decoder [n] Control register(s): Altering IG, IW, Range Type, BI, UIO, UIG, UIW, and/or ISP at runtime could allow an attacker to steer transactions to another target and/or memory range, potentially affecting the data coherency and/or the memory integrity of the TE State stored in the target.
- Altering link or Transaction Layer behavior at runtime
 - PCIe DVSEC for Flex Bus Port registers
 - CXL IDE Control register (if utilizing CXL IDE for Transport Security): Altering PCRC Disable or IDE.StopEnable at runtime could allow the link to operate incorrectly if there is a mismatch with the initiator. This should result in a MAC error which should cause the IDE on the link to transition to Insecure State (if IDE is enabled) and is indicated in the CXL IDE Error Status register, bit[1] and bit[7]. When IDE transitions to Insecure State, the target shall transition to the ERROR state.
- Configuring CXL Capabilities on the target to prevent the leaking of cypher text from the target after a CXL Reset, as discussed in [Section 11.5.4.8.3](#).

11.5.4.8.3 Reset and Error Handling Behavior of the Target

This section outlines the expected target TSP behavior to Conventional Reset, CXL Reset, Transport Security Failures, and changes in secure sessions. All other resets, including FLR, do not affect the secured link or Transport Security (i.e., CXL IDE) and therefore do not change the target's TSP security state.

11.5.4.8.3.1 Conventional Reset and Link Failures

Because a Conventional Reset or a Link Failure shall take down the CXL links to the target, all transactions shall immediately terminate. The target shall perform the following after receiving a Conventional Reset or on link failure:

- Terminate the CMA/SPDM SecondarySession(s) and PrimarySession
- Clear CMA/SPDM secure session keys to 0
- Target shall ensure that all TE State = 1 clear text data cannot be leaked
- If implementing target-based encryption:

- Clear association of CKID to encryption keys
- Clear association of HPA ranges to encryption keys
- Clear memory encryption keys
- Transition to CONFIG_UNLOCKED state

11.5.4.8.3.2 CXL Reset, Transport Security Failures, SecondarySession(s) Termination, and PrimarySession Restart

CXL Reset, Transport Security failures such as IDE going insecure, SecondarySession(s) terminating, and/or a new PrimarySession being started all affect the target but do not take down the link. For these conditions, the target shall support the following additional requirements:

- Transition to ERROR state
- Target shall stop accepting all future TEE memory transactions
 - **Writes:** TEE opcode writes shall be dropped and S2M NDR non-TEE opcode shall be returned for the write response.
 - **Reads:** TEE opcode reads return a fixed data pattern of all 1s. S2M DRS non-TEE opcode shall be returned for the read response.
- Terminate the CMA/SPDM SecondarySession(s) and PrimarySession
- Clear CMA/SPDM secure session keys to 0
- Target shall ensure that all TE State = 1 clear text data cannot be leaked
- If implementing target-based encryption:
 - Clear association of CKID to encryption keys
 - Clear association of HPA ranges to encryption keys
 - Clear memory encryption keys
- Transition to CONFIG_UNLOCKED state

Prevention of the leaking of cypher text from the target after a CXL Reset:

- Data on the target is encrypted using initiator-based or target-based memory encryption. Clear text data is protected; however, the cipher text stored on the target could be leaked after a CXL Reset, which should be avoided.
- Target should support clearing or randomizing of volatile HDM data on reset and indicate this capability by setting CXL Reset Mem Clr Capable in DVSEC CXL Capability to 1 and by setting Default Volatile HDM State after Cold Reset, Default Volatile HDM State after Warm Reset, and Default Volatile HDM State after Hot Reset to 1 in DVSEC CXL Capability3.
- Target should set Volatile HDM State after Hot Reset Configurability in DVSEC CXL Capability3 to 0 to indicate that the target is not capable of preserving volatile HDM state across Hot Reset.
- Target shall support resetting all TE State to 0 when enabling the ability to clear or randomize all data in response to a CXL Reset.
- If the target sets CXL Reset Mem Clr Capable to 1, the host should allow the target clear memory during CXL Reset by setting CXL Reset Mem Clr Enable in DVSEC CXL Status.

11.5.4.9 Component Command Interfaces

When the target is locked, Component Command Interface (CCI) commands that allow the target's configuration to be altered, new features to be set, maintenance operations to be executed, target memory partitioning changes, injecting and clearing of poison, sanitizing, and/or secure erasing shall be rejected by the target by returning Invalid Security State status.

Likewise, FM requests that can change the Dynamic Capacity configuration at runtime shall be rejected by the target by returning Invalid Security State status.

See [Section 8.2.9](#) for the following command-specific additions that are relevant to TSP support:

- Transfer FW, Activate FW
- Set Features
- Perform Maintenance
- Set Partition Info
- Inject Poison, Clear Poison
- Sanitize, Secure Erase, Passphrase Secure Erase, Security Send
- Release Dynamic Capacity

See [Section 7.6.7.6.3](#) for the following command-specific addition that is relevant to TSP support:

- Set DC Region Configuration

11.5.4.10 Dynamic Capacity

As described above, CXL security requires the device to implement configuration locking to prevent tampering with the trusted configuration at runtime. Dynamic Capacity allows for memory capacity to be added or released from one or more hosts at runtime but relies on a static maximum capacity configuration to have been configured at initialization time.

The following sections outline specific areas in which there are additional requirements for targets that implement both Dynamic Capacity and TSP.

11.5.4.10.1 TE State Changes

For Dynamic Capacity targets that implement TE State changes, there are additional responsibilities:

- Before a target adds any Dynamic Capacity to a host and before adding capacity to one host after releasing the capacity from a different host, the target shall overwrite or cryptographically clear the memory contents and reset the TE State to 0 for memory ranges described in extents that will be added to a host in response to an Add Dynamic Capacity command (see [Section 8.2.9.9.3](#)). By overwriting those ranges and resetting the TE State, the extents that will be added shall be reset to an untrusted state to prevent stale trusted data released from one host from being exposed to a trusted entity when adding that memory to another host.

11.5.4.10.2 Multiple Host Considerations

For Dynamic Capacity implementations that utilize multiple hosts, only a single host may generate transactions to a specific DPA range at any given time.

- Target shall maintain an association between the host in which the Dynamic Capacity is being added and the target that is providing the memory. The target

shall not allow other hosts to access the memory until the memory has been released from the current host, at which point the host to target association is removed and the TE State is reset as described above.

- Note that implementing Dynamic Capacity on a target already requires the target to maintain this host to target association and the target shall correctly reject transactions from hosts that do not currently own the Dynamic Capacity. This behavior is described in [Section 9.13.3](#).

11.5.5 TSP Requests and Responses

Each TSP Request sent over the secure CMA/SPDM session shall result in exactly one TSP Response, the Delayed Response if the request will take significant time to complete, or the Error Response if the request could not be executed.

11.5.5.1 TSP Request Overview

[Table 11-26](#) outlines the TSP Request payloads, defined in the sections that follow.

Table 11-26. TSP Request Overview

TSP Request Message		Message Support ¹	Payload Size	Legal TSP State	TSP Usage
Opcode	Name	HDM-H Device			
81h	Get Target TSP Version	M	4	CONFIG_UNLOCKED	Target config
82h	Get Target Capabilities		4	CONFIG_LOCKED	
83h	Set Target Configuration		C2h+	ERROR	
84h	Get Target Configuration		4	CONFIG_UNLOCKED	
85h	Get Target Configuration Report		8	CONFIG_LOCKED	
86h	Lock Target Configuration		4	CONFIG_UNLOCKED	
87h	Set Target CKID Specific Key	O - CKID-based target memory encryption	10h+	CONFIG_LOCKED	Runtime CKID-based target memory encryption
88h	Set Target CKID Random Key		10h+		
89h	Clear Target CKID Key		8		
8Ah	Set Target Range Specific Key	O - Range-based target memory encryption	20h+		Runtime range-based target memory encryption
8Bh	Set Target Range Random Key		20h+		
8Ch	Clear Target Range Key		8		
8Dh	Set Target TE State	O - Explicit TE State changes	20h+		Runtime explicit state changes
8Eh	Check Target Delayed Completion	O - Delayed completion of a request	4		Checking for completion of a long executing request

1. M = Mandatory message, O = Optional message. Targets shall return a Error Response of Unsupported Request if the target does not support the request.

11.5.5.2 TSP Response Overview

Table 11-27 outlines the TSP Response payloads, defined in the sections that follow.

Table 11-27. TSP Response Overview

TSP Response Message		Message Support ¹	Payload Size
Opcode	Name	HDM-H Device	
01h	Get Target TSP Version Response	M	5+
02h	Get Target Capabilities Response		34h
03h	Set Target Configuration Response		4
04h	Get Target Configuration Response		C0h
05h	Get Target Configuration Report Response		8+
06h	Lock Target Configuration Response		4
07h	Set Target CKID Specific Key Response	O - CKID-based target memory encryption	4
08h	Set Target CKID Random Key Response		4
09h	Clear Target CKID Key Response		4
0Ah	Set Target Range Specific Key Response	O - Range-based target memory encryption	4
0Bh	Set Target Range Random Key Response		4
0Ch	Clear Target Range Key Response		4
0Dh	Set Target TE State Response	O - Explicit TE State changes	4
0Eh	Check Target Delayed Completion Response	O - Delayed completion of a request	4
7Eh	Delayed Response		8
7Fh	Error Response	M	0Ch+

1. M = Mandatory message, O = Optional message.

11.5.5.3 Request Response and CMA/SPDM Sessions

Table 11-28 outlines which TSP-defined Request and Response payloads are allowed on a given TSP-defined CMA/SPDM session and which are prohibited.

Table 11-28. TSP Request Response and CMA/SPDM Sessions

TSP Request Response Message	SPDM PrimarySession	SPDM SecondarySession(s)	Other non-TSP-related SPDM Session
Get Target TSP Version Get Target TSP Version Response	Allowed	Allowed	Allowed
Get Target Capabilities Get Target Capabilities Response		Allowed	Allowed
Set Target Configuration Set Target Configuration Response		Prohibited	Prohibited
Get Target Configuration Get Target Configuration Response		Allowed	Prohibited
Get Target Configuration Report Get Target Configuration Report Response		Allowed	Prohibited
Lock Target Configuration Lock Target Configuration Response		Prohibited	Prohibited
Set Target TE State Set Target TE State Response		Allowed	Prohibited
Check Target Delayed Completion Check Target Delayed Completion Response		Allowed	Prohibited
Set Target CKID Specific Key Set Target CKID Specific Key Response		Allowed	Prohibited
Set Target CKID Random Key Set Target CKID Random Key Response		Allowed	Prohibited
Clear Target CKID Key Clear Target CKID Key Response		Allowed	Prohibited
Set Target Range Specific Key Set Target Range Specific Key Response		Allowed	Prohibited
Set Target Range Random Key Set Target Range Random Key Response		Allowed	Prohibited
Clear Target Range Key Clear Target Range Key Response		Allowed	Prohibited
Delayed Response		Allowed	Allowed
Error Response		Allowed	Allowed

11.5.5.4 Version

11.5.5.4.1 TSP Version Negotiation

The PrimarySession shall be utilized to perform the following process to negotiate TSP version with TSP-capable targets:

- Initiator shall send Get Target TSP Version request with Major Version Number = 1h.
- DSM shall support Get Target TSP Version request with Major Version Number = 1h and shall return Get Target TSP Version Response with a list of all supported versions.
- Initiator shall select a common (typically highest) version supported and utilize this version number in all subsequent messages to the target.

- Initiator shall not issue requests to the target other than Get Target TSP Version until the initiator has received a successful Get Target TSP Version Response and selected a common version that is supported by both the initiator and the target.

11.5.5.4.2 Get Target TSP Version

The initiator shall utilize Get Target TSP Version to discover the TSP versions that the target supports.

Possible Error Response, Error Codes:

- None

Table 11-29. Get Target TSP Version

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Get Target TSP Version = 81h.
02h	2	Reserved

11.5.5.4.3 Get Target TSP Version Response

If no error condition is detected, the DSM shall respond to the Get Target TSP Version with a Get Target TSP Version Response message.

Table 11-30. Get Target TSP Version Response

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Get Target TSP Version Response= 01h.
02h	2	Reserved
04h	1	Version Number Entry Count: The number of version entries, N, that follow. Shall be > 0.
05h	N	Version Number Entry: 8-bit version entry formatted as: <ul style="list-style-type: none"> • Bits[7:4]: Major Version Number = 1 • Bits[3:0]: Minor Version Number = 0

11.5.5 Target Capabilities

The following request and response payload defines the TSP security features that the target supports.

11.5.5.5.1 Get Target Capabilities

Any SPDM session may utilize the Get Target Capabilities request to discover the target's memory encryption, access control, and configuration capabilities.

Possible Error Response, Error Codes:

- Version Mismatch

Table 11-31. Get Target Capabilities

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Get Target Capabilities = 82h.
02h	2	Reserved

11.5.5.5.2 Get Target Capabilities Response

If no error condition is detected, the DSM shall respond to the Get Target Capabilities request with a Get Target Capabilities Response message.

Table 11-32. Get Target Capabilities Response (Sheet 1 of 3)

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Get Target Capabilities Response = 02h.
02h	2	<p>Memory Encryption Features Supported: Memory encryption related features that the target supports. Zero or more bits may be set. 1 indicates supported, 0 indicates not supported.</p> <ul style="list-style-type: none"> • Bit[0]: Encryption: When set, memory encryption for data at rest is implemented on the target. • Bit[1]: CKID-based Encryption: When set, the target supports the CKID-based TSP requests and responses for memory encryption. Encryption shall be set, and the CKID Base Required and Number of CKIDs fields shall be valid. When cleared, the target does not support the CKID field in Transaction Layer requests. • Bit[2]: Range-based Encryption: When set, the target supports the range-based TSP requests and responses. Encryption shall also be set, and Memory Encryption Number of Range Based Keys shall be valid. • Bit[3]: Initiator Supplied Entropy: The target supports initiator-supplied entropy when generating a random key. • Bit[4]: CKID Base Required: Valid only when CKID-based Encryption is set. When set, the target requires a CKID Base and Number of CKIDs to be programmed. When cleared, the target supports any CKID value within the 13-bit field. • Bits[15:5]: Reserved.
04h	4	<p>Memory Encryption Algorithms Supported: Valid only if Encryption is set in Memory Encryption Features Supported. 1 indicates supported, 0 indicates not supported. If target memory encryption is supported, one or more bits shall be set.</p> <ul style="list-style-type: none"> • Bit[0]: AES-XTS-128 • Bit[1]: AES-XTS-256 • Bits[30:2]: Reserved • Bit[31]: Vendor Specific Algorithm: When set, all other bits in this field are vendor specific
08h	2	Memory Encryption Number of Range Based Keys: Valid only if Range-based Encryption is set in Memory Encryption Features Supported. This is the maximum Range ID that can be utilized with the range-based memory encryption requests. Targets that do not support range-based memory encryption shall report 0.
0Ah	2	Reserved

Table 11-32. Get Target Capabilities Response (Sheet 2 of 3)

Byte Offset	Length in Bytes	Description
0Ch	2	<p>TE State Change and Access Control Features Supported: The TE State change and access control features that the target supports. Zero or more bits may be set. 1 indicates supported, 0 indicates not supported.</p> <ul style="list-style-type: none"> • Bit[0]: Write Access Control: When set, indicates that the target supports dropping writes that fail the verification of TEE Intent to stored TE State. When set, explicit state changes shall be supported and one or more of bits[4:3] shall also be set. • Bit[1]: Read Access Control: When set, indicates that the target supports returning all 1s for read data in response to reads that fail the verification of TEE Intent to stored TE State. When set, one or more of bits[4:2] shall also be set. • Bit[2]: Implicit TE State Change: When set, indicates that the target supports implicit TE State changes using a 64B granularity, Explicit In-band TE State Change shall be set, and Explicit In-band TE State Granularity support for 64B shall be set. • Bit[3]: Explicit Out-of-band TE State Change: When set, indicates that the target supports the CMA/SPDM out-of-band explicit Set Target TE State change message and the Supported Explicit Out-of-band TE State Granularity field shall be valid. Support is optional for targets that support implicit TE State changes or explicit in-band TE State changes. • Bit[4]: Explicit In-band TE State Change: When set, indicates that the target supports explicit TE State changes utilizing the TEUpdate memory transaction and the Supported Explicit In-band TE State Granularity field shall be valid. Support is required for targets that support implicit TE State changes and optional for targets that support explicit out-of-band TE State changes. • Bit[5]: Explicit TE State Change Sanitize: When set, indicates that the target supports overwriting data that is affected by the explicit state change with 0s when the explicit request is received and before the change is considered complete by the target. When set, one or more of bits[4:3] shall also be set. • Bits[15:6]: Reserved.
0Eh	2	Reserved
10h	4	<p>Supported Explicit Out-of-band TE State Granularity: The granularity the target supports for explicit out-of-band TE State changes and verification in powers of 2, starting with 64B. Valid only if Explicit Out-of-band TE State Change is set in TE State Change and Access Control Features Supported. One or more bits shall be set. 1 indicates target support, 0 indicates no target support.</p> <ul style="list-style-type: none"> • Bit[0]: 64B • ... • Bit[6]: 4K • ... • Bit[15]: 2MB • ... • Bit[24]: 1GB • ... • Bit[31]: 128GB
14h	4	<p>Supported Explicit In-band TE State Granularity: The granularity the target supports for explicit in-band TE State changes and verification in powers of 2 starting with 64B. Valid only if Explicit In-band TE State Change is set in TE State Change and Access Control Features Supported. One or more bits shall be set. 1 indicates target support, 0 indicates no target support.</p> <ul style="list-style-type: none"> • Bit[0]: 64B • Bit[1]: 128B • Bit[2]: 256B • Bit[3]: 512B • Bit[4]: 1KB • Bit[5]: 2KB • Bit[6]: 4KB • Bit[7]: 8KB • Bit[8]: 16KB • Bit[9]: 32KB • Bit[10]: 64KB • Bits[30:11]: Reserved • Bit[31]: The entire memory space of the device. When set, the target supports TEUpdate using Length Index 7 to change the TE State for the entire address range. When cleared, the target does not support use of Length Index value of 7.

Table 11-32. Get Target Capabilities Response (Sheet 3 of 3)

Byte Offset	Length in Bytes	Description
18h	2	Configuration Features Supported: The configuration features that the target supports. Zero or more bits may be set. 1 indicates supported, 0 indicates not supported. <ul style="list-style-type: none"> • Bit[0]: Locked Target FW Update: When set, the target supports FW updates after the target is locked. When cleared to 0, the target does not support FW updates after the target is locked. • Bit[1]: Target Supports Additional CMA/SPDM Sessions: The target supports using CMA/SPDM PSK to set up one or more SecondarySession(s). If this bit is set, then Number of Secondary Sessions shall be valid. When cleared, the target does not support secondary SPDM sessions. • Bits[15:2]: Reserved.
1Ah	2	Reserved
1Ch	4	Number of CKIDs: Total number of CKIDs that the target supports. Valid only if CKID-based Encryption is set in Memory Encryption Features Supported. Shall be ≥ 2 and $< 2^{13}$.
20h	1	Number of Secondary Sessions: Total number of additional SPDM SecondarySessions that the target supports. When valid, this shall be > 0 and ≤ 4 .
21h	13h	Reserved

11.5.5.6 Target Configuration

The following request and response payloads provide TSP configuration, locking, and register reporting of the target.

11.5.5.6.1 Set Target Configuration

The PrimarySession is utilized with the Set Target Configuration request to place the target in the preferred transport configuration. This includes providing SecondarySession CMA/SPDM PSK Key Material that shall be utilized by the Target to generate random keys for this additional session.

Possible Error Response, Error Codes:

- Version Mismatch
- Invalid Request
 - Entropy was not supplied
 - Number of CKIDs being enabled is $>$ Number of CKIDs the target reported in Get Target Configuration
 - CKID Base is $\geq 2^{13}$
 - CKID Base + Number of CKIDs $\geq 2^{13}$
 - TE State Granularity specified is not supported by the target
 - Length Indexes are not unique
 - Length Index 0 or 7 was specified but the TE State Granularity was not 0
 - Implicit TE State Change is enabled and Explicit In-band TE State Change is not enabled or no Explicit In-band TE State Granularity Entries enable Length Index 0
 - CKID-based Encryption and Range-based Encryption are both enabled
- No Privilege
 - Request was not received on the PrimarySession
- Invalid Security State
- Target not in CONFIG_UNLOCKED state

The following structure shall be utilized for associating explicit TE State lengths to specific indexes that are utilized with the TEUpdate memory transaction. Implementations that do not utilize explicit in-band TE State changes do not need to include valid information in the response payload for these entries.

Table 11-33. Explicit In-band TE State Granularity Entry

Byte Offset	Length in Bytes	Description
00h	8	TE State Granularity: The number of bytes of contiguous HPA space to which the explicit in-band TE state change will apply when the TEUpdate memory transaction is received by the target. Shall be one of the values reported by the target in Supported Explicit In-band TE State Granularity reported in Get Target Capabilities. When specifying Length Index 0 or 7, this field shall be 0 because the length is predefined. This field is ignored when Length Index is FFh.
08h	1	Length Index: The 3-bit length index that shall be utilized to represent the TE State Length in the SnpType portion of the TEUpdate memory transaction. Value shall be ≥ 0 and ≤ 7 for valid Explicit In-band TE State Granularity Entries. Each length entry specified shall utilize a unique Length Index. Value FFh is reported for unused Explicit In-band TE State Granularity Entries.
09h	7	Reserved

Table 11-34. Set Target Configuration (Sheet 1 of 4)

Byte Offset	Length in Bytes	Description
000h	1	TSP Version: V1.0 = 10h.
001h	1	Opcode: Set Target Configuration = 83h.
002h	2	Memory Encryption Features Enable: Enable the memory encryption features for the target. Zero or more bits may be set. 1 indicates to enable, 0 indicates to disable. <ul style="list-style-type: none"> • Bit[0]: Encryption: When set, memory encryption for data at rest shall be enabled on the target. When cleared, target memory encryption shall be disabled. • Bit[1]: CKID-based Encryption: When set, CKID-based encryption shall be enabled on the target, Encryption shall be set, and the CKID Base Required field shall be valid. When set, Range-based Encryption shall be cleared. When cleared, the target shall disable use of the CKID field in the Transaction Layer requests. • Bit[2]: Range-based Encryption: When set, range-based encryption shall be enabled on the target and Encryption shall also be set. When set, CKID-based Encryption shall be cleared. When cleared, the target shall disable use of range-based target memory encryption. • Bit[3]: CKID Base Required: Valid only when CKID-based Encryption is set. When set, the target shall enable a valid CKID range and CKID Base and Number of CKIDs fields shall be valid. When cleared, the target shall enable any CKID value within the 13-bit field. • Bits[15:4]: Reserved.
004h	4	Memory Encryption Algorithm Select: Valid only if Encryption is set in Memory Encryption Features Enable. Select the target memory encryption algorithm to utilize. Only one bit shall be set. 1 indicates selected, 0 indicates not selected. <ul style="list-style-type: none"> • Bit[0]: AES-XTS-128 • Bit[1]: AES-XTS-256 • Bits[30:2]: Reserved • Bit[31]: Vendor Specific Algorithm: When set, all other bits in this field are vendor specific
008h	4	Reserved

Table 11-34. Set Target Configuration (Sheet 2 of 4)

Byte Offset	Length in Bytes	Description
00Ch	2	<p>TE State Change and Access Control Features Enable: Enable the TE State change and access control features for the target. Zero or more bits may be set. 1 indicates to enable, 0 indicates to disable.</p> <ul style="list-style-type: none"> • Bit[0]: Write Access Control: When set, the target shall enable dropping writes that fail the verification of TEE Intent to stored TE State. When set, explicit state changes shall be enabled and one or more of bits[4:3] shall also be set. • Bit[1]: Read Access Control: When set, the target shall enable returning all 1s for read data in response to reads that fail the verification of TEE Intent to stored TE State. When set, one or more of bits[4:2] shall also be set. • Bit[2]: Implicit TE State Change: When set, implicit TE State changes shall be enabled on the target using 64B granularity. Explicit In-band TE State Change shall be set, and at least one Explicit In-band TE State Granularity Entry with Length Index 0 shall be enabled. When cleared, implicit TE State changes shall be disabled on the target. • Bit[3]: Explicit Out-of-band TE State Change: When set, the target shall be enabled to utilize the explicit CMA/SPDM out-of-band explicit Set Target TE State change request and Explicit Out-of-band TE State Change Granularity shall be valid. • Bit[4]: Explicit In-band TE State Change: When set, the target shall be enabled for explicit in-band TE State changes utilizing the TEUpdate memory transaction and Explicit In-band TE State Change Granularity Entries shall be valid. • Bit[5]: Explicit TE State Change Sanitize: When set, the target shall be enabled to overwrite data that is affected by the explicit state change with 0s when the explicit request is received and before the change is considered complete by the target. When set, one or more of bits[4:3] shall also be set. • Bits[15:6]: Reserved.
00Eh	2	Reserved
010h	4	<p>Explicit Out-of-band TE State Granularity: The granularity that the initiator is requesting the target to utilize for explicit out-of-band TE State changes in powers of 2, starting with 64B. Valid only if Explicit Out-of-band TE State Change is set in TE State Change and Access Control Features Enable. Only one bit shall be set. 1 indicates target shall enable, 0 indicates target shall disable.</p> <ul style="list-style-type: none"> • Bit[0]: 64B • ... • Bit[6]: 4 KB • ... • Bit[15]: 2 MB • ... • Bit[24]: 1 GB • ... • Bit[31]: 128 GB
014h	4	Reserved
018h	2	<p>Configuration Features Enable: Enable the configuration features for the target. Zero or more bits may be set. 1 indicates to enable, 0 indicates to disable.</p> <ul style="list-style-type: none"> • Bit[0]: Locked Target FW Update: When set, the target shall enable FW updates after the target is locked. When cleared, the target shall disable FW updates after the target is locked. • Bit[1]: Special Purpose Memory: When set, memory reported to the initiator should not be treated as general-purpose memory. Accelerator devices shall set this to indicate the exposed memory capacity cannot be utilized as general-purpose memory by the initiator. The target should also report EFI Memory Type and Attribute of EfiConventionalMemory EFI_MEMORY_SP for the corresponding memory ranges in the CDAT DSEMTS. When cleared, the initiator may utilize the memory capacity as general-purpose memory. • Bits[15:2]: Reserved.
01Ah	2	Reserved
01Ch	4	CKID Base: The lowest CKID that the target shall enable. Valid only if CKID Base Required is set in Memory Encryption Features Enable. Shall be < 2^13.
020h	4	Number of CKIDs: Number of contiguous CKID that the target shall enable starting at the CKID Base. Valid only if CKID Base Required is set in Memory Encryption Features Enable. Shall be <= Number of CKIDs reported by the target in Get Target Capabilities. CKID Base + Number of CKIDs shall be < 2^13.
024h	0Ch	Reserved
030h	10h	Explicit In-band TE State Granularity Entry 0
040h	10h	Explicit In-band TE State Granularity Entry 1

Table 11-34. Set Target Configuration (Sheet 3 of 4)

Byte Offset	Length in Bytes	Description
050h	10h	Explicit In-band TE State Granularity Entry 2
060h	10h	Explicit In-band TE State Granularity Entry 3
070h	10h	Explicit In-band TE State Granularity Entry 4
080h	10h	Explicit In-band TE State Granularity Entry 5
090h	10h	Explicit In-band TE State Granularity Entry 6
0A0h	10h	Explicit In-band TE State Granularity Entry 7
0B0h	10h	Reserved
0C0h	2	<p>Configuration Validity Flags: Indicators of which fields are valid in the remaining portion of this request. Zero or more bits may be set.</p> <ul style="list-style-type: none"> • Bit[0]: Secondary Session 0: When set, the Secondary Session 0 PSK Key Material field and Secondary Session 0 CKID Type field shall both be valid and shall be utilized by the target to generate a key for use when a SecondarySession0 is created utilizing CMA/SPDM PSK. When cleared, no SecondarySession0 is allowed by the target. This bit shall be set only if Get Target Capabilities, Number of Secondary Sessions is > 0. • Bit[1]: Secondary Session 1: When set, the Secondary Session 1 PSK Key Material field and Secondary Session 1 CKID Type field shall both be valid and shall be utilized by the target to generate a key for use when a SecondarySession1 is created utilizing CMA/SPDM PSK. When cleared, no SecondarySession1 is allowed by the target. This bit shall be set only if Get Target Capabilities, Number of Secondary Sessions is > 1. • Bit[2]: Secondary Session 2: When set, the Secondary Session 2 PSK Key Material field and Secondary Session 2 CKID Type field shall both be valid and shall be utilized by the target to generate a key for use when a SecondarySession2 is created utilizing CMA/SPDM PSK. When cleared, no SecondarySession2 is allowed by the target. This bit shall be set only if Get Target Capabilities, Number of Secondary Sessions is > 2. • Bit[3]: Secondary Session 3: When set, the Secondary Session 3 PSK Key Material field and Secondary Session 3 CKID Type field shall both be valid and shall be utilized by the target to generate a key for use when a SecondarySession3 is created utilizing CMA/SPDM PSK. When cleared, no SecondarySession3 is allowed by the target. This bit shall be set only if Get Target Capabilities, Number of Secondary Sessions is > 3. • Bits[15:4]: Reserved.
0C2h	0Eh	Reserved
0D0h	1	<p>Secondary Session CKID Type: The CKID Type to assign to a SecondarySession.</p> <ul style="list-style-type: none"> • Bit[0]: Secondary Session 0 CKID Type: When set, the CKID shall be considered a TVMCKID. When cleared, the CKID shall be considered an OSCKID. This field shall be valid if the Validity Flags, Secondary Session 0 is set. • Bit[1]: Secondary Session 1 CKID Type: When set, the CKID shall be considered a TVMCKID. When cleared, the CKID shall be considered an OSCKID. This field shall be valid if the Validity Flags, Secondary Session 1 is set. • Bit[2]: Secondary Session 2 CKID Type: When set, the CKID shall be considered a TVMCKID. When cleared, the CKID shall be considered an OSCKID. This field shall be valid if the Validity Flags, Secondary Session 2 is set. • Bit[3]: Secondary Session 3 CKID Type: When set, the CKID shall be considered a TVMCKID. When cleared, the CKID shall be considered an OSCKID. This field shall be valid if the Validity Flags, Secondary Session 3 is set. • Bits[7:4]: Reserved.
0D1h	0Fh	Reserved
0E0h	20h	Secondary Session 0 PSK Key Material: The CMA/SPDM PSK key material that the target shall utilize for key derivation for use when the SecondarySession0 CMA/SPDM secure session is created. This field shall be valid if the Validity Flags, Secondary Session 0 is set. When this PSK key material is used to set up the CMA/SPDM SecondarySession0, the associated CMA/SPDM PSK Hint entry in the CMA/SPDM PSK_EXCHANGE request shall be "SECONDARY_SESSION_0_PSK" string with NUL terminator.
100h	20h	Secondary Session 1 PSK Key Material: The CMA/SPDM PSK key material that the target shall utilize for key derivation for use when the SecondarySession1 CMA/SPDM secure session is created. This field shall be valid if the Validity Flags, Secondary Session 1 is set. When this PSK key material is used to set up the CMA/SPDM SecondarySession1, the associated CMA/SPDM PSK Hint entry in the CMA/SPDM PSK_EXCHANGE request shall be "SECONDARY_SESSION_1_PSK" string with NUL terminator.

Table 11-34. Set Target Configuration (Sheet 4 of 4)

Byte Offset	Length in Bytes	Description
120h	20h	Secondary Session 2 PSK Key Material: The CMA/SPDM PSK key material that the target shall utilize for key derivation for use when the SecondarySession2 CMA/SPDM secure session is created. This field shall be valid if the Validity Flags, Secondary Session 2 is set. When this PSK key material is used to set up the CMA/SPDM SecondarySession2, the associated CMA/SPDM PSK Hint entry in the CMA/SPDM PSK_EXCHANGE request shall be "SECONDARY_SESSION_2_PSK" string with NUL terminator.
140h	20h	Secondary Session 3 PSK Key Material: The CMA/SPDM PSK key material that the target shall utilize for key derivation for use when the SecondarySession3 CMA/SPDM secure session is created. This field shall be valid if the Validity Flags, Secondary Session 3 is set. When this PSK key material is used to set up the CMA/SPDM SecondarySession3, the associated CMA/SPDM PSK Hint entry in the CMA/SPDM PSK_EXCHANGE request shall be "SECONDARY_SESSION_3_PSK" string with NUL terminator.

11.5.5.6.2 Set Target Configuration Response

If no error condition is detected, the DSM shall respond to the Set Target Configuration request with a Set Target Configuration Response message.

Table 11-35. Set Target Configuration Response

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Set Target Configuration Response = 03h.
02h	2	Reserved

11.5.5.6.3 Get Target Configuration

The PrimarySession or SecondarySession(s) shall utilize the Get Target Configuration request to verify that the target is in the correct security mode after the target is locked. While it is possible to report the configuration with this request before the target is locked, the content cannot be trusted to be immutable until this request is executed after the target is successfully locked.

Possible Error Response, Error Codes:

- Version Mismatch
- No Privilege
 - The request was not received on the PrimarySession or SecondarySession(s)
- Invalid Security State
 - Target not in CONFIG_LOCKED or CONFIG_UNLOCKED state

Table 11-36. Get Target Configuration

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Get Target Configuration = 84h.
02h	2	Reserved

11.5.5.6.4 Get Target Configuration Response

If no error condition is detected, the DSM shall respond to the Get Target Configuration request with a Get Target Configuration Response message.

Table 11-37. Get Target Configuration Response (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Get Target Configuration Response = 04h.
02h	2	<p>Memory Encryption Features Enabled: The memory encryption features that are enabled for the locked target. Zero or more bits may be set. 1 indicates to enable, 0 indicates to disable.</p> <ul style="list-style-type: none"> • Bit[0]: Encryption: When set, memory encryption for data at rest is enabled on the target. • Bit[1]: CKID-based Encryption: When set, CKID-based encryption is enabled on the target, Encryption shall be set, and the CKID Base Required field shall be valid. When set, Range-based Encryption shall be cleared. When cleared, the target has disabled use of the CKID field in the Transaction Layer requests. • Bit[2]: Range-based Encryption: When set, range-based encryption is enabled on the target and Encryption shall also be set. When set, CKID-based Encryption shall be cleared. When cleared, the target has disabled use of range-based target memory encryption. • Bit[3]: CKID Base Required: Valid only when CKID-based Encryption is set. When set, the target has been enabled for a valid CKID range and CKID Base and Number of CKIDs fields shall be valid. When cleared, the target has enabled any CKID value within the 13-bit field. • Bits[15:4]: Reserved.
04h	4	<p>Memory Encryption Algorithm Selected: Valid only if Encryption is set in Memory Encryption Features Enabled. The target memory encryption algorithm that is selected. Only one bit may be set. 1 indicates selected, 0 indicates not selected.</p> <ul style="list-style-type: none"> • Bit[0]: AES-XTS-128 • Bit[1]: AES-XTS-256 • Bits[30:2]: Reserved • Bit[31]: Vendor Specific Algorithm: When set, all other bits in this field are vendor specific
08h	4	Reserved
0Ch	2	<p>TE State Change and Access Control Features Enabled: The TE State change and access control features that are enabled for the locked target. Zero or more bits may be set. 1 indicates enabled, 0 indicates disabled.</p> <ul style="list-style-type: none"> • Bit[0]: Write Access Control: When set, dropping writes that fail the verification of TEE Intent to stored TE State is enabled on the target. When set, explicit state changes shall be enabled and one or more of bits[4:3] shall also be set. • Bit[1]: Read Access Control: When set, returning all 1s for read data in response to reads that fail the verification of TEE Intent to stored TE State is enabled on the target. When set, one or more of bits[4:2] shall also be set. • Bit[2]: Implicit TE State Change: When set, the implicit TE State change feature has been enabled on the target, Explicit In-band TE State Change shall be enabled, and at least one Explicit In-band TE State Granularity Entry with Length Index 0 shall be enabled. When cleared, the implicit TE State change feature has been disabled on the target. • Bit[3]: Explicit Out-of-band TE State Change: When set, use of the explicit CMA/SPDM out-of-band Set Target TE State change request is enabled on the target and Explicit Out-of-Band TE State Change Granularity shall be valid. • Bit[4]: Explicit In-band TE State Change: When set, explicit in-band TE State changes utilizing the TEUpdate memory transaction is enabled on the target and Explicit In-band TE State Change Granularity Entries shall be valid. • Bit[5]: Explicit TE State Change Sanitize: When set, the target is enabled to overwrite data that is affected by the explicit state change with 0s when the explicit request is received and before the change is considered complete by the target. When set, one or more of bits[4:3] shall also be set. • Bits[15:6]: Reserved.
0Eh	2	Reserved

Table 11-37. Get Target Configuration Response (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
10h	4	<p>Explicit Out-of-band TE State Granularity Enabled: The granularity that has been enabled on the target to utilize for explicit TE State changes in powers of 2, starting with 64B. Valid only if Explicit Out-of-band TE State Change is set in TE State Change and Access Control Features Enabled. Only one bit shall be set. 1 indicates selected, 0 indicates not selected.</p> <ul style="list-style-type: none"> • Bit[0]: 64B • ... • Bit[6]: 4 KB • ... • Bit[15]: 2 MB • ... • Bit[24]: 1 GB • ... • Bit[31]: 128 GB
14h	4	Reserved
18h	2	<p>Configuration Features Enabled: The configuration features that are enabled for the locked target. Zero or more bits may be set. 1 indicates to enable, 0 indicates to disable.</p> <ul style="list-style-type: none"> • Bit[0]: Locked Target FW Update: When set, FW updates after the target is locked are enabled on the target. When cleared, FW updates after the target is locked are disabled on the target. • Bits[15:1]: Reserved.
1Ah	2	Reserved
1Ch	4	CKID Base: The lowest CKID that the target enabled. Valid only if CKID Base Required is set in Memory Encryption Features Enabled. Shall be < 2^{13} .
20h	4	Number of CKIDs: Number of contiguous CKID that the target enabled starting at the CKID Base. Valid only if CKID Base Required is set in Memory Encryption Features Enabled. CKID Base + Number of CKIDs shall be < 2^{13} .
24h	1	<p>Current TSP State</p> <ul style="list-style-type: none"> • 00h = CONFIG_UNLOCKED • 01h = CONFIG_LOCKED • 02h = ERROR • All other encodings are reserved
25h	0Bh	Reserved
030h	10h	Explicit In-band TE State Granularity Entry 0
040h	10h	Explicit In-band TE State Granularity Entry 1
050h	10h	Explicit In-band TE State Granularity Entry 2
060h	10h	Explicit In-band TE State Granularity Entry 3
070h	10h	Explicit In-band TE State Granularity Entry 4
080h	10h	Explicit In-band TE State Granularity Entry 5
090h	10h	Explicit In-band TE State Granularity Entry 6
0A0h	10h	Explicit In-band TE State Granularity Entry 7
0B0h	10h	Reserved

11.5.5.6.5 Get Target Configuration Report

The PrimarySession shall be utilized with the Get Target Configuration Report request to return specific CXL.mem configuration register TSP Report content that is utilized to verify the locked target's configuration. [Section 11.5.4.8](#) describes the checks that can be conducted on the TSP Report response payload that is returned from this request.

This request allows select CXL configuration register contents on the endpoint target to be returned for verification through the secure PrimarySession, and is modeled after the TDISP Get Interface Report request.

While it is possible to read the configuration with this request, before the target is locked, the content cannot be trusted to be immutable until this request is executed after the target is successfully locked.

Possible Error Response, Error Codes:

- Version Mismatch
- No Privilege
 - Request was not received on the PrimarySession or SecondarySession(s)
- Invalid Security State
 - Target not in CONFIG_LOCKED or CONFIG_UNLOCKED state

Table 11-38. Get Target Configuration Report

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Get Target Configuration Report = 85h.
02h	2	Reserved
04h	2	Offset: Offset in bytes from the start of the report to where this request message begins. For the first Get Target Configuration Report request, the initiator shall clear this field to all 0s. For non-first requests, the offset is the sum of Portion Length values reported in all the previous Get Target Configuration Report Responses.
06h	2	Length: The length of the report in bytes to be returned in the corresponding response. Length is an unsigned 16-bit integer. This value is the smaller of the following values: <ul style="list-style-type: none"> • Capacity of the initiator's internal buffer for receiving the target's report • Remainder Length of the preceding Get Target Configuration Report response If the Length is > Remainder Length, the target shall transfer the remaining Report Data, Portion Length, and a Remainder Length of 0 in the response.

11.5.5.6.6 Get Target Configuration Report Response

If no error condition is detected, the DSM shall respond to the Get Target Configuration Report request with a Get Target Configuration Report Response message.

Table 11-39. Get Target Configuration Report Response

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Get Target Configuration Report Response = 05h.
02h	2	Reserved
04h	2	Portion Length: Number of bytes of this portion of the TSP Report. This shall be less than or equal to Length received as part of the request. The target is permitted to set this field to a value that is less than the Length received in the request due to limitations on the target's internal buffer.
06h	2	Remainder Length: Number of bytes of the TSP Report that have not been sent yet after the current response. For the last response, the target shall clear this field to all 0s as an indication to the initiator that the entire TSP Report has been sent.
08h	Portion Length	Report Data: Requested contents of TSP Report.

Table 11-40 presents the TSP Report structure.

Table 11-40. TSP Report

Byte Offset	Length in Bytes	Description
00h	1	Valid TSP Report Fields: More than one bit may be set. <ul style="list-style-type: none"> Bit[0]: CXL IDE Capability Structure Valid: When set, the TSP Report contains a valid CXL IDE Capability Structure Bits[7:1]: Reserved
01h	3	Reserved
04h	3Ch	PCIe DVSEC for CXL Devices: See Section 8.1 .
40h	20h	PCIe DVSEC for Flex Bus Port: See Section 8.2 .
60h	38h	CXL Link Capability Structure: See Section 8.2.4.19 .
98h	10h	CXL Timeout and Isolation Capability Structure: See Section 8.2.4.24 .
A8h	10h + n*1Fh	CXL HDM Decoder Capability Structure: The number of HDM decoders (n) is specified in the Decoder Count field in the CXL HDM Decoder Capability register. See Section 8.2.4.20 .
B8h + n*1Fh	24h	CXL IDE Capability Structure: Structure for optionally reporting the CXL IDE Transport Security configuration. Valid if CXL IDE Capability Structure Valid is set.

11.5.5.6.7 Lock Target Configuration

The PrimarySession shall be utilized with the Lock Target Configuration request to lock the target configuration that is relevant to protecting the TEE configuration and TVM data, and perform memory security checks before responding to this request. The locked configuration includes the security configuration set, utilizing the Set Target Configuration request in addition to CXL and PCIe target registers that need to be made immutable to protect TEE integrity. This request does not lock configuration registers and other registers that are not part of protecting the TEE configuration and TVM data.

Once locked, the configuration shall be immutable until a subsequent Conventional Reset of the target.

The target shall reject requests to lock when already in the CONFIG_LOCKED state.

If Write Access Control is enabled on the target, the target shall clear the TE State to 0 for all addressable memory in response to the Lock Target Configuration Request and before generating a Lock Target Configuration Response. The target may require extra execution time to clear the initial TE State and may utilize the Delayed Response to prevent request timeouts, as described in [Section 11.5.5.9](#).

See [Section 11.5.4.8.1](#) for expected target locking behavior before successfully responding to this request.

Possible Error Response, Error Codes:

- Version Mismatch
- Invalid Security State
 - Target not in CONFIG_UNLOCKED state
- No Privilege
 - Request was not received on the PrimarySession
- Invalid Security Configuration
 - Locking security configuration failed due to memory security check failures
- Already Locked

- The target is already in CONFIG_LOCKED state

Table 11-41. Lock Target Configuration

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Lock Target Configuration = 86h.
02h	2	Reserved

11.5.5.6.8 Lock Target Configuration Response

The DSM responds with a Lock Target Configuration Response in response to Lock Target Configuration request if the target security checks and lock operations were successful, and the target transitioned to the CONFIG_LOCKED state. An Error Response shall be returned if the memory security checks failed, the configuration could not be locked, and/or other errors occurred.

See [Section 11.5.4.8.1](#) for expected target behavior before sending this successful response to the lock request.

Table 11-42. Lock Target Configuration Response

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Lock Target Configuration Response = 06h.
02h	2	Reserved

11.5.5.7 Optional Explicit TE State Change Requests and Responses

If the target supports explicit TE State changes, it shall support receiving a Set Target TE State request out-of-band that utilizes the CMA/SPDM secure PrimarySession or SecondarySession(s). The target reports its supported explicit TE State change mechanisms in Get Target Capabilities. The host that supports explicit TE State changes shall enable explicit changes, utilizing Set Target Configuration. The explicit TE State change request shall be sent by the host to the target before the memory range is to be accessed, ensuring that the target has the correct TE State to perform the access checks before receiving the memory transactions that it will verify.

11.5.5.7.1 Set Target TE State (Out-of-band)

Targets that utilize explicit TE State change notifications from the host shall implement the following request and response for updating the TE State based on memory range. The target shall change the TE State as specified for all memory locations covered by the Starting Address and Length that are relevant to the target for the interleave set configured.

If the target is enabled to sanitize memory affected by a state change, the target shall overwrite all data affected by this state change with 0s before generating the response.

This request could take a significant amount of time to complete if sanitization of a large amount of memory is required and is handled as follows:

- If the request can be completed without an excessive delay that could cause an SPDM timeout, the target shall respond with Set Target TE State Response, once the request is complete.

- If the target is not capable of executing the request due to the execution time required or does not support delaying the request's completion, it may fail this request with a Long Execution Time Error Response.
- Otherwise, if the target is capable of executing the request and the request will take a significant amount of time to complete, the target shall respond with Delayed Response with a nonzero Delay Time in microseconds (us). The host should wait the prescribed amount of time and issue the Check Target Delayed Completion request to verify the state change is complete.

If the target is already executing a state change request and another state change request is received, it shall fail the new request with Busy for Error Response.

Possible Error Response, Error Codes:

- Version Mismatch
- Invalid Request
 - Number of Memory Ranges is 0
 - One or more Memory Range Starting Address and Length is invalid for the target
 - One or more Memory Range Starting Address is not aligned to Explicit Out-of-band TE State Granularity Selected reported in Get Target Configuration Response
 - One or more Memory Range Length is not an exact multiple of the Explicit Out-of-band TE State Granularity Selected reported in Get Target Configuration Response
 - One or more Memory Range Starting Address and Length spans TEE and non-TEE ranges
- No Privilege
 - Request was not received on the PrimarySession or SecondarySession(s)
- Invalid Security State
 - Target not in CONFIG_LOCKED state
- Busy
 - Target is currently executing another state change.
- Long Execution Time
 - Target cannot execute the request due to the amount of execution time required

Table 11-43. Memory Range

Byte Offset	Length in Bytes	Description
00h	8	Starting Address: HPA to start re-initializing the TE State. This address shall be aligned to the Access Control Granularity Selected reported in Get Target Configuration Response.
08h	8	Length: The length of the memory range, in bytes, at which to re-initialize the TE State. This number shall be an exact multiple of the Access Control Granularity Selected reported in Get Target Configuration Response.

Table 11-44. Set Target TE State

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Set Target TE State = 8Dh.
02h	1	TE State: The new target TE State to set for the included memory ranges. Only one bit shall be set. <ul style="list-style-type: none"> • Bit[0]: TE State: When set to 1, the target shall set the TE State for all included memory ranges to TEE Exclusive. When cleared to 0, the target shall set the TE State for all included memory ranges to non-TEE Exclusive. • Bits[7:1]: Reserved.
03h	1	Number of Memory Ranges: The number of Memory Range structures (N) that are included in this payload. This shall be <= 32 and > 0.
04h	Ch	Reserved
10h	10h * N	List of Memory Range structures.

11.5.5.7.2 Set Target TE State Response (Out-of-band)

If no error condition is detected and the state change request has completed execution on the target, the DSM shall respond to the Set Target TE State request with a Set Target TE State Response message.

Table 11-45. Set Target TE State Response

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Set Target TE State Response = 0Dh.
02h	2	Reserved

11.5.5.8 Optional Target-based Memory Encryption Requests and Responses

The following interfaces are optionally supported by the target for use when target memory encryption is enabled. The target reports support for these interfaces in the Memory Encryption Features Supported reported in the Get Target Capabilities Response.

11.5.5.8.1 Set Target CKID Specific Key

The PrimarySession or SecondarySession(s) shall be utilized with the Set Target CKID Specific Key request to define a specific CKID as a Memory Encryption Features Supported reported and associate that CKID with specific key material. This request is utilized with CKID-based target memory encryption. Once set, the association between an initiator CKID and the target's keys are immutable and attempts to set a new key for the CKID shall fail. To change the association, the CKID shall be explicitly cleared by the initiator, utilizing Clear Target CKID Key before the CKID can be set for new keys using this request. The Attributes CKID Type shall be utilized by the target to ensure that each memory transaction TEE Intent matches the CKID Type (TVMCKID or OSCKID) as described in [Section 11.5.4.6.2.1](#).

Possible Error Response, Error Codes:

- Version Mismatch
- Unsupported Request
 - Target does not support CKID-based memory encryption

- Invalid Security State
 - Target not in CONFIG_LOCKED state
- No Privilege
 - Request was not received on the PrimarySession or SecondarySession(s).
- Invalid CKID
 - More CKIDs have been assigned to the target than the Number of CKIDs reported in Get Target Capabilities.
 - Requested CKID is outside the range of CKID Base to CKID Base + Number of CKIDs configured in Set Target Configuration.
 - CKID already has a key associated with it. Clear Target CKID Key shall be utilized to reset the CKID association before the CKID can be set again.

Table 11-46. Set Target CKID Specific Key

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Set Target CKID Specific Key = 87h.
02h	2	Reserved
04h	4	CKID: The CKID assigned by the initiator to this encryption key. The attribute CKID Type defines whether the CKID is an OSCKID or TVMCKID.
08h	1	CKID Type: The type of CKID to utilize: <ul style="list-style-type: none"> • 00h = TVMCKID • 01h = OSCKID • All other encodings are reserved
09h	6	Reserved
0Fh	1	Validity Flags: Indicators of which fields are valid in the remaining portion of this request. More than one bit may be set. <ul style="list-style-type: none"> • Bit[0]: When set, the Data Encryption Key field is valid • Bit[1]: When set, the Tweak Key field is valid • Bits[7:2]: Reserved
10h	20h	Data Encryption Key: The memory encryption key to utilize with the CKID.
30h	20h	Tweak Key: The memory encryption tweak key to utilize with the CKID. If the configured encryption algorithm does not require a tweak key, then this field shall be ignored.

11.5.5.8.2 Set Target CKID Specific Key Response

If no error condition is detected, the DSM shall respond to the Set Target CKID Specific Key request with a Set Target CKID Specific Key Response message.

Table 11-47. Set Target CKID Specific Key Response

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Set Target CKID Specific Key Response= 07h.
02h	2	Reserved

11.5.5.8.3 Set Target CKID Random Key

The PrimarySession or SecondarySession(s) shall be utilized with the Set Target CKID Random Key request to define a specific CKID as an OSCKID or TVMCKID and associate the CKID with a random target-generated key utilizing optional initiator-generated entropy. This request is utilized with CKID-based target memory encryption. Once set, the association between an initiator CKID and the target's keys are immutable and attempts to set a new key for the CKID shall fail. To change the association, the CKID shall be explicitly cleared by the initiator, utilizing Clear Target CKID Key before the CKID can be set for new keys using this request. The Attributes CKID Type shall be utilized by the target to ensure that each memory transaction TEE Intent matches the CKID Type (OSCKID or TVMCKID) as described in [Section 11.5.4.6.2.1](#).

When the target is utilizing the initiator-generated entropy, the target should generate a unique key, even if the initiator-supplied entropy is equivalent to the entropy provided in a previous request.

Possible Error Response, Error Codes:

- Version Mismatch
- Unsupported Request
 - Target does not support CKID-based memory encryption.
- Invalid Security State
 - Target not in CONFIG_LOCKED state.
- No Privilege
 - Request was not received on the PrimarySession or SecondarySession(s).
- Invalid CKID
 - More CKIDs have been assigned to the target than the Number of CKIDs reported in Get Target Capabilities.
 - Requested CKID is outside the range of CKID Base to CKID Base + Number of CKIDs configured in Set Target Configuration.
 - CKID already has a key associated with it. Clear Target CKID Key shall be utilized to reset the CKID association before the CKID can be set again.
- No Entropy
 - The target was not able to obtain enough entropy to generate a random key.

Table 11-48. Set Target CKID Random Key (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Set Target CKID Random Key = 88h.
02h	2	Reserved
04h	4	CKID: The CKID assigned by the initiator to this encryption key. The attribute CKID Type defines whether the CKID is an OSCKID or TVMCKID.
08h	1	Attributes: Additional attributes of the CKID. Only one bit shall be set. <ul style="list-style-type: none"> • Bit[0]: CKID Type: When set, the CKID is considered a TVMCKID. If cleared, the CKID is considered an OSCKID. • Bits[7:1]: Reserved.
09h	6	Reserved

Table 11-48. Set Target CKID Random Key (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
0Fh	1	Validity Flags: Indicators of which fields are valid in the remaining portion of this request. More than one bit may be set. <ul style="list-style-type: none"> Bit[0]: When set, the Data Encryption Key Entropy field is valid Bit[1]: When set, the Tweak Key Entropy field is valid Bits[7:2]: Reserved
10h	20h	Data Encryption Key Entropy: Optional initiator-supplied memory encryption key entropy for the target to utilize when generating a random data encryption key for the CKID. This field shall be ignored if Initiator Supplied Entropy is not set in Memory Encryption Features Supported reported in Get Target Capabilities Response.
30h	20h	Tweak Key Entropy: Optional initiator-supplied memory encryption tweak key entropy for the target to utilize when generating a random tweak key for the CKID. This field shall be ignored if Initiator Supplied Entropy is not set in Memory Encryption Features Supported reported in Get Target Capabilities Response.

11.5.5.8.4 Set Target CKID Random Key Response

If no error condition is detected, the DSM shall respond to the Set Target CKID Random Key request with a Set Target CKID Random Key Response message.

Table 11-49. Set Target CKID Random Key Response

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Set Target CKID Random Key Response= 08h.
02h	2	Reserved

11.5.5.8.5 Clear Target CKID Key

The PrimarySession or SecondarySession(s) shall be utilized with the Clear Target CKID Key request to clear any association on the target between the previously set CKID and random or specific keys that may have been programmed. This request is utilized with CKID-based target memory encryption. This request is utilized by the initiator to clear the association between an initiator's CKID and the target's keys and allows a CKID to be utilized with a set of new keys. Once a CKID has been cleared the CKID Type is no longer a TVMCKID and encryption utilizing that CKID is bypassed.

The target shall break the association of CKID to key and shall clear the associated key to 0.

The same SPDM session that was utilized to set the key for the CKID shall be the same session that is utilized to clear the CKID. If the SPDM session utilized to set the key has been terminated or closed, then a Conventional Reset or CXL Reset shall be utilized to clear the CKID association with the key material.

Possible Error Response, Error Codes:

- Version Mismatch
- Unsupported Request
 - Target does not support CKID-based memory encryption
- Invalid Security State
 - Target not in CONFIG_LOCKED state
- No Privilege

- Request was not received on the PrimarySession or SecondarySession(s)
- CKID was not set on the same SPDM session
- Invalid CKID
 - Requested CKID is outside the range of CKID Base to CKID Base + Number of CKIDs configured in Set Target Configuration
 - CKID is not currently programmed in the target

Table 11-50. Clear Target CKID Key

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Clear Target CKID Key = 89h.
02h	2	Reserved
04h	4	CKID: The CKID for which to clear the encryption key.

11.5.5.8.6 Clear Target CKID Key Response

If no error condition is detected, the DSM shall respond to the Clear Target CKID Key request with a Clear Target CKID Key Response message.

Table 11-51. Clear Target CKID Key Response

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Clear Target CKID Key Response = 09h.
02h	2	Reserved

11.5.5.8.7 Set Target Range Specific Key

The PrimarySession or SecondarySession(s) shall be utilized with the Set Target Range Specific Key request to associate a specific memory range with initiator-specified key material. This request is utilized with range-based target memory encryption. Once set, the association between an initiator HPA memory range and the target's keys are immutable and attempts to set a new key for the same or subset of the same HPA range shall fail. To change the association, the memory range shall be explicitly cleared by the initiator, utilizing Clear Target Range Key before the memory range can be set for new keys using this request.

Possible Error Response, Error Codes:

- Version Mismatch
- Invalid Request
 - The memory Range Base and Range Size is invalid for the target.
 - The memory Range Base is not aligned to the Range Size.
 - The Range Size is not a power of 2.
 - The Range ID is \geq Memory Encryption Number of Range Based Keys reported in Get Target Capabilities.
 - The memory range register specified by the Range ID requested is already associated with keys.

- The request attempted to change a memory range or subset of the range already configured. Clear Target Range Key shall be utilized to reset the memory range association.
- Unsupported Request
 - The target does not support range-based memory encryption.
- Invalid Security State
 - Target not in CONFIG_LOCKED state.
- No Privilege
 - The request was not received on the PrimarySession or SecondarySession(s).

Table 11-52. Set Target Range Specific Key

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Set Target Range Specific Key = 8Ah.
02h	2	Reserved
04h	4	Range ID: The range ID assigned to this encryption key. Shall be within the range specified in Get Target Capabilities Response, Memory Encryption Number of Range Based Keys.
08h	8	Range Start: HPA of the first 4-KB-aligned block within the range. <ul style="list-style-type: none"> • Bits[11:0]: Reserved • Bits[63:12]: Start HPA for the range, HPA[63:12]
10h	8	Range End: HPA of the last 4-KB-aligned block within the range. <ul style="list-style-type: none"> • Bits[11:0]: Reserved • Bits[63:12]: End HPA for the range, HPA[63:12]
18h	7	Reserved
1Fh	1	Validity Flags: Indicators of which fields are valid in the remaining portion of this request. More than one bit may be set. <ul style="list-style-type: none"> • Bit[0]: When set, the Data Encryption Key field is valid • Bit[1]: When set, the Tweak Key field is valid • Bits[7:2]: Reserved
20h	20h	Data Encryption Key: The memory encryption key to utilize with the range.
40h	20h	Tweak Key: The memory encryption tweak key to utilize with the range. If the configured encryption algorithm does not require a tweak key, then this field shall be ignored.

11.5.5.8.8 Set Target Range Specific Key Response

If no error condition is detected, the DSM shall respond to the Set Target Range Specific Key request with a Set Target Range Specific Key Response message.

Table 11-53. Set Target Range Specific Key Response

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Set Target Range Specific Key Response= 0Ah.
02h	2	Reserved

11.5.5.8.9 Set Target Range Random Key

The PrimarySession or SecondarySession(s) shall be utilized with the Set Target Range Random Key request to associate a specific memory range with initiator-specified entropy material. This request is utilized with range-based target memory encryption. Once set, the association between an initiator HPA memory range and the target's keys are immutable and attempts to set a new key for the same or subset of the same HPA range shall fail. To change the association, the memory range shall be explicitly cleared by the initiator, utilizing Clear Target Range Key before the memory range can be set for new keys using this request.

Possible Error Response, Error Codes:

- Version Mismatch
- Invalid Request
 - Memory Range Base and Range Size is invalid for the target.
 - Memory Range Base is not aligned to the Range Size.
 - Range Size is not a power of 2.
 - Range ID is >= Memory Encryption Number of Range Based Keys reported in Get Target Capabilities.
 - Memory range register specified by the Range ID is already associated with keys.
 - Request attempted to change a memory range or subset of the range already configured. Clear Target Range Key shall be utilized to reset the memory range association.
- Unsupported Request
 - Target does not support range-based memory encryption.
- Invalid Security State
 - Target not in CONFIG_LOCKED state.
- No Privilege
 - Request was not received on the PrimarySession or SecondarySession(s).

Table 11-54. Set Target Range Random Key (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Set Target Range Random Key = 8Bh.
02h	2	Reserved
04h	4	Range ID: The range ID assigned to this encryption key. Shall be in the range specified in Get Target Capabilities Response, Memory Encryption Number of Range Based Keys.
08h	8	Range Start: HPA of the first 4-KB-aligned block within the range. <ul style="list-style-type: none"> • Bits[11:0]: Reserved • Bits[63:12]: Start HPA for the range, HPA[63:12]
10h	8	Range End: HPA of the last 4-KB-aligned block within the range. <ul style="list-style-type: none"> • Bits[11:0]: Reserved • Bits[63:12]: End HPA for the range, HPA[63:12]
18h	7	Reserved

Table 11-54. Set Target Range Random Key (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
1Fh	1	Validity Flags: Indicators of which fields are valid in the remaining portion of this request. More than one bit may be set. <ul style="list-style-type: none"> Bit[0]: When set, the Data Encryption Key Entropy field is valid Bit[1]: When set, the Tweak Key Entropy field is valid Bits[7:2]: Reserved
20h	20h	Data Key Entropy: Optional initiator-supplied data key entropy that is utilized by the target when generating an encryption key for the range.
40h	20h	Tweak Key Entropy: Optional initiator-supplied memory encryption tweak key entropy that is utilized by the target when generating a tweak key for the range. If the configured encryption algorithm does not require a tweak key, then this field shall be ignored.

11.5.5.8.10 Set Target Range Random Key Response

If no error condition is detected, the DSM shall respond to the Set Target Range Random Key request with a Set Target Range Random Key Response message.

Table 11-55. Set Target Range Random Key Response

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Set Target Range Random Key Response = 0Bh.
02h	2	Reserved

11.5.5.8.11 Clear Target Range Key

The PrimarySession or SecondarySession(s) shall be utilized with the Clear Target Range Key request to clear any association between a previously set HPA memory range and random or specific keys that may have been programmed. This request is utilized with range-based target memory encryption. This request is utilized by the initiator to clear the association between an initiator's memory range and the target's keys and allows a memory range to be utilized with a new set of keys. The target shall break the association of HPA memory range to key, shall clear the associated key to 0 and memory encryption utilizing the cleared memory range shall be bypassed.

The same SPDM session that was utilized to set the key for the range shall be the same session that is utilized to clear the range. If the SPDM session utilized to set the key has been terminated or closed, then a Conventional Reset or CXL Reset shall be utilized to clear the memory range association with the key material.

Possible Error Response, Error Codes:

- Version Mismatch
- Unsupported Request
 - The target does not support range-based memory encryption
- Invalid Security State
 - Target not in CONFIG_LOCKED state
- No Privilege
 - The request was not received on the PrimarySession or SecondarySession(s)
 - The range was not set on the same SPDM session
- Invalid Request

- Range-based encryption is not supported by the target
- The range specified is not currently set

Table 11-56. Clear Target Range Key

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Clear Target Range Key = 8Ch.
02h	2	Reserved
04h	4	Range ID: The range ID assigned to this encryption key. Shall be within the range specified in Get Target Capabilities Response, Memory Encryption Number of Range Based Keys.

11.5.5.8.12 Clear Target Range Key Response

If no error condition is detected, the DSM shall respond to the Clear Target Range Key request with a Clear Target Range Key Response message.

Table 11-57. Clear Target Range Key Response

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Clear Target Range Key Response = 0Ch.
02h	2	Reserved

11.5.5.9 Optional Delayed Completion Requests and Responses

TSP provides a simple polling mechanism for requests that may take a significant amount of time to complete. If the time it takes for the target to execute the request might cause an SPDM timeout, the target may return a Delayed Response. This will notify the initiator that the request passed all the syntax checks, has started execution on the target, and will take additional time to complete.

The initiator utilizes the returned Delay Time to check on the completion of the request by sending the Check Target Delayed Completion request after waiting the prescribed amount of time. If the request has completed the target shall send a Check Target Delayed Completion Response. If the request is still executing the target shall respond with another Delayed Response with the updated time the initiator is expected to wait.

11.5.5.9.1 Delayed Response

The response that the target has started executing a request that could take a significant amount of time to complete. The target shall return the number of microseconds (us) that it expects the initiator to wait before checking the completion of the request.

Table 11-58. Delayed Response (Sheet 1 of 2)

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Delayed Response = 7Eh.

Table 11-58. Delayed Response (Sheet 2 of 2)

Byte Offset	Length in Bytes	Description
02h	2	Reserved
04h	4	Delay Time: Estimated number of microseconds (us) that the initiator should delay before checking for the completion of the long executing command. Shall be > 0.

11.5.5.9.2 Check Target Delayed Completion

This request shall only utilized by the initiator if a request returns a Delayed Response. It is utilized to verify the completion of a long executing request. If the request has completed execution, the target shall return a Check Target Delayed Completion Response. If the request is still executing, the target shall respond with Delayed Response with the new Delay Time that the initiator should wait before issuing this request again. If Delayed Response was not returned for a request, there is no delayed completion, and this request shall be failed by the target.

Possible Error Response, Error Codes:

- Version Mismatch
- Invalid Request
 - No request is outstanding that would result in a delayed completion

Table 11-59. Check Target Delayed Completion

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Check Target Delayed Completion = 8Eh.
02h	2	Reserved

11.5.5.9.3 Check Target Delayed Completion Response

If no error condition is detected and the execution of the long executing request has completed, the DSM shall respond to the Check Target Delayed Completion request with a Check Target Delayed Completion Response message. This indicates that the previously delayed completion of the request is now complete.

Table 11-60. Get Target TE State Change Completion Response

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Check Target Delayed Completion Response = 0Eh.
02h	2	Reserved

11.5.5.10 Error Response

The Error Response is permitted to be used by the target to complete any of the requests issued to the target.

Table 11-61. Error Response

Byte Offset	Length in Bytes	Description
00h	1	TSP Version: V1.0 = 10h.
01h	1	Opcode: Error Response = 7Fh.
02h	2	Reserved
04h	4	Error Code: See Table 11-62 .
08h	4	Error Data: See Table 11-62 .
0Ch	Varies	Extended Error Data: See Table 11-62 .

Table 11-62. Error Response — Error Code, Error Data, Extended Error Data

Error Code	Definition	Error Data	Extended Error Data
0000h	Reserved	0000h	None
0001h	Invalid Request: One or more fields in the request are invalid.	0000h	None
0002h	Busy: The target could not process the request but the target may be able to process the request if it is sent again in the future.	0000h	None
0003h	Unspecified: An unspecified error occurred.	0000h	None
0004h	Unsupported Request: The Message Type in the command is unsupported.	Unsupported Message Type	None
0005h	Version Mismatch: The version in the request is not supported.	Highest TSP version number that the target supports	None
0006h	Vendor Specific Error: A vendor defined error occurred.	Length of Extended Error Data	Vendor specific error data
0007h	No Privilege: The requested session ID has no privilege to generate the request.	SPDM Session ID	None
0008h	No Entropy: Target failed to generate random numbers to execute the request.	0000h	None
0009h	Invalid CKID: The request contains an invalid CKID.	0000h	None
000Ah	Invalid Security Configuration: Target security checks failed due to an invalid configuration.	Length of Extended Error Data	Vendor specific error data
000Bh	Invalid Security State: The target was not in the correct TSP state to execute the request.	0000h	None
000Ch	Long Execution Time: The target did not start the execution of the request as it may cause SMTP timeouts waiting for the completion.	0000h	None
000Dh	Already Locked: An initiator is attempting to lock an already locked target.	0000h	None

§ §

12.0 Reliability, Availability, and Serviceability

CXL RAS capabilities are built on top of PCIe*. Additional capabilities are introduced to address cache coherency and memory as listed below.

12.1 Supported RAS Features

[Table 12-1](#) lists the RAS features supported by CXL and their applicability to CXL.io vs. CXL.cache and CXL.mem.

Table 12-1. CXL RAS Features

Feature	CXL.io	CXL.cache and CXL.mem
Link CRC and Retry	Required	Required
Link Retraining and Recovery	Required	Required
eDPC	Optional	Leverage CXL.io capability CXL.cache or CXL.mem errors may be signaled via ERR_FATAL or ERR_NONFATAL and may trigger eDPC
ECRC	Optional	N/A
Hot-Plug	Not Supported in RCD mode Managed Hot-Plug is supported in CXL VH mode	Same as CXL.io
Data Poisoning	Required	Required
CXL Isolation	N/A	Optional (see Section 12.3)
Viral	N/A	Required (see Section 12.4)

12.2 CXL Error Handling

CXL Error handling can be subdivided into two parts:

- Link and Protocol Errors, which apply to the CXL component-to-component communication mechanism. These include errors detected by CXL.cache and CXL.mem protocol logic. This is further described in [Section 12.2.1](#) and [Section 12.2.2](#).
- Device Errors, which apply exclusively to the device itself. This is further described in [Section 12.2.3](#).

12.2.1 Protocol and Link Layer Error Reporting

Protocol and Link errors are detected and communicated to the Host where the errors can be exposed and handled. Errors may also be reflected to Platform software if so configured. There are no error pins that connect CXL devices to the Host. Errors are communicated between the Host and the CXL device via messages over CXL.io.

CXL Protocol and Link errors detected by components that are part of a CXL VH are escalated and reported using standard PCIe error reporting mechanisms over CXL.io as UIEs and/or CIEs. See PCIe Base Specification for details.

Reporting and logging of CXL Protocol and Link errors in RCD mode is described in this section.

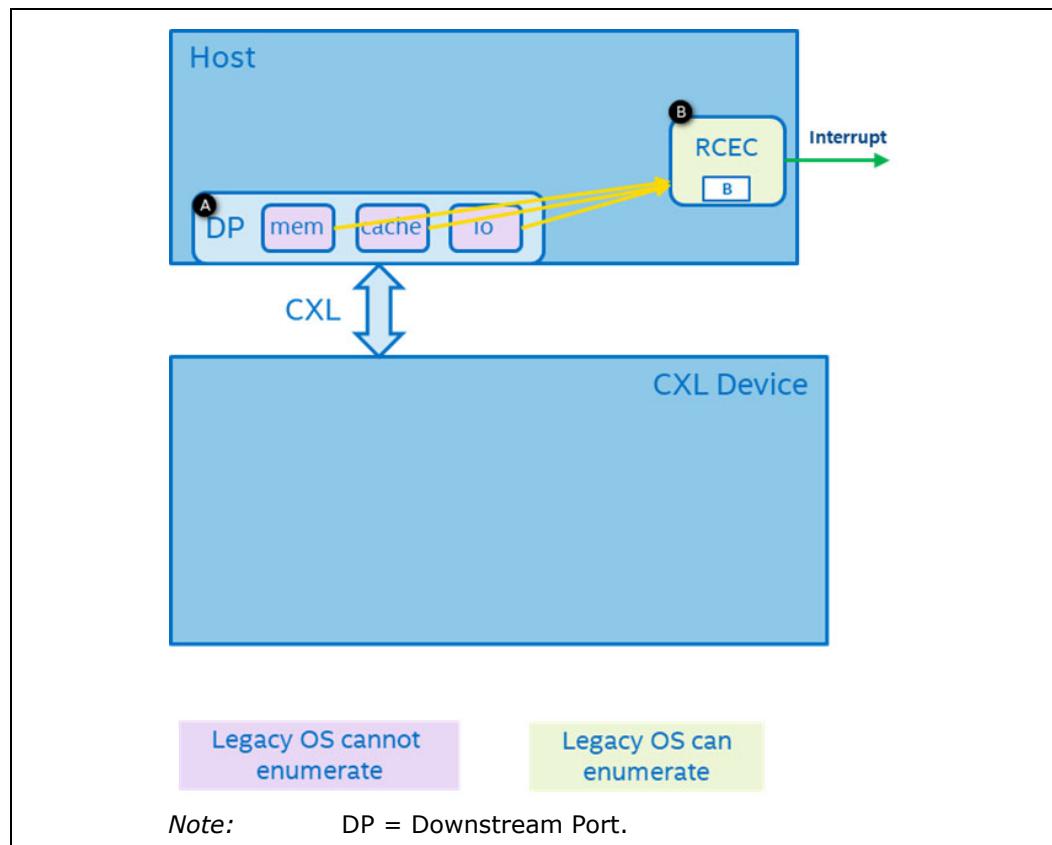
12.2.1.1

RCH Downstream Port-detected Errors

RCH Downstream Port-detected CXL Protocol errors are escalated and reported via the Root Complex error-reporting mechanisms as UIEs and/or CIEs. The various signaling and logging steps are listed below and illustrated in [Figure 12-1](#).

1. DP_A CXL.io-detected errors are logged in the local AER Extended Capability in DP_A RCRB. Software must ensure that the Root Port Control register in the DP_A AER Extended Capability is not configured to generate interrupts.
2. DP_A CXL.cache and CXL.mem log errors in the CXL RAS Capability (see [Section 8.2.4.17](#)).
3. DP_A CXL.cache, CXL.mem, or CXL.io sends error message(s) to RCEC.
4. RCEC logs UIEs and/or CIEs. The RCEC Error Source Identification register shall log the RCEC's Bus, Device, and Function Numbers because the RCH Downstream Port is not associated with one.
5. RCEC generates an MSI/MSI-X, if enabled.

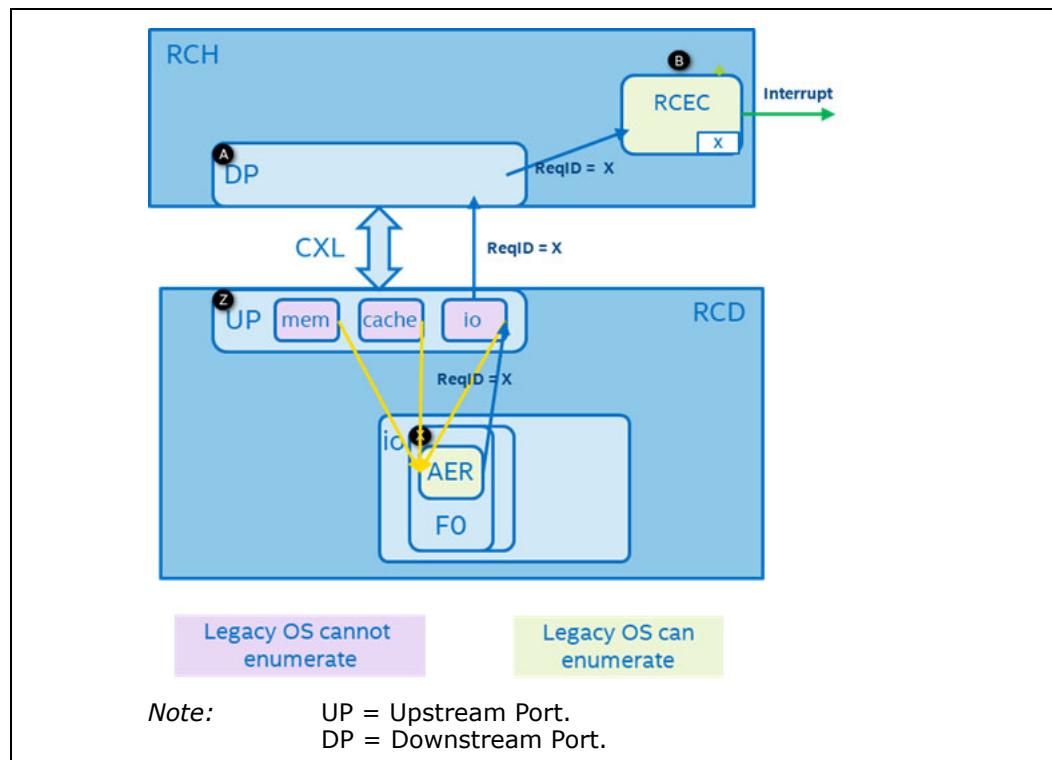
The OS error handler may begin by inspecting the RCEC AER Extended Capability and following PCIe rules to discover the error source. The RCEC Error Source Identification register is insufficient for identifying the error source. The OS error handler may rely on RDPAS structures (see [Section 9.18.1.5](#)), if present, to identify such Downstream Port(s). The Platform Software Error Handler may interrogate the Platform-specific error logs in addition to the error logs defined in PCIe Base Specification and this specification.

Figure 12-1. RCH Downstream Port Detects Error

12.2.1.2 RCD Upstream Port-detected Errors

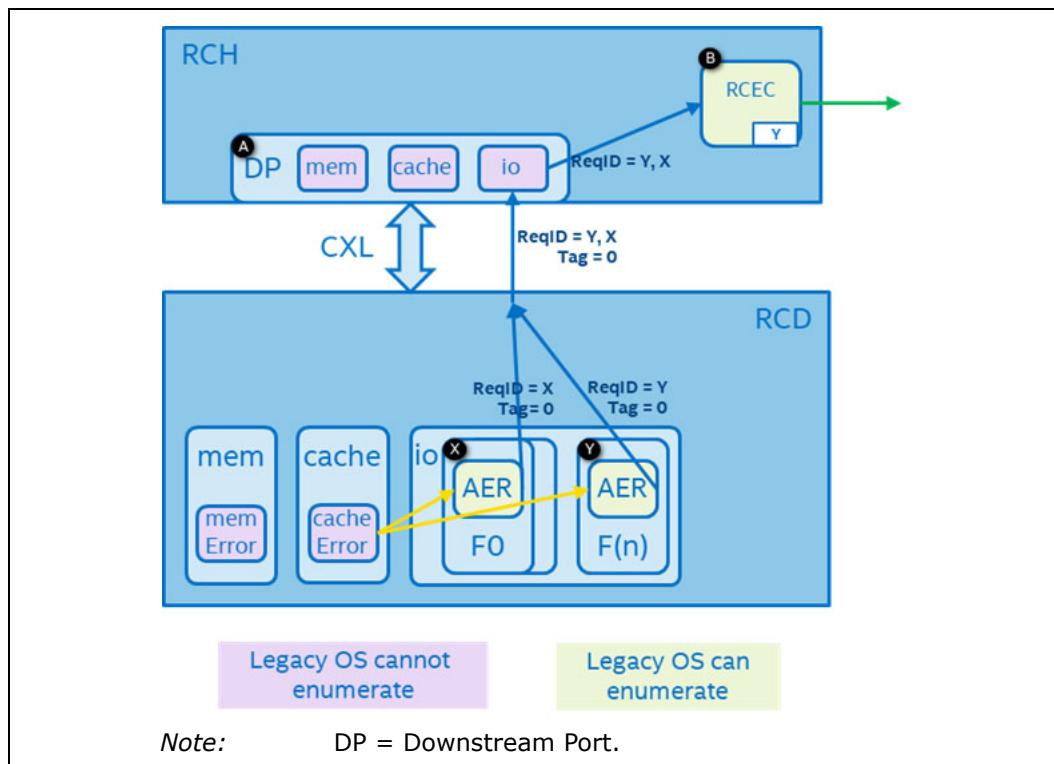
RCD Upstream Port-detected CXL protocol errors are also escalated and reported via the RCEC. The various signaling and logging steps are listed below and illustrated in Figure 12-2.

1. If a CXL.cache or CXL.mem logic block in UP_Z detects a protocol or link error, the block shall log the error in the CXL RAS Capability (see Section 8.2.4.17).
2. Upstream Port RCRB shall not implement the AER Extended Capability.
3. UP_Z sends an error message to all CXL.io Functions that are affected by this error. (This example shows a device with a single function. The message must include all the details that the CXL.io function needs for constructing an AER record.)
4. CXL.io Functions log the received message in their respective AER Extended Capability.
5. Each affected CXL.io Function sends an ERR_ message to UP_Z with its own Requester ID.
6. UP_Z forwards this Error message across the Link without logging.
7. DP_A forwards the Error message to the RCEC.
8. RCEC logs the error in the Root Error Status register and then signals an interrupt, if enabled, in accordance with PCIe Base Specification. The Error Source Identification register in the RCEC shall point to the CXL.io Function that sent the ERR_ message.

Figure 12-2. RCD Upstream Port Detects Error**12.2.1.3 RCD RCiEP-detected Errors**

CXL protocol errors detected by the RCD RCiEP are also escalated and reported via the RCEC. The various signaling and logging steps are listed below and illustrated in [Figure 12-3](#).

1. CXL.cache (or CXL.mem) notifies all affected CXL.io Functions of the errors.
2. All affected CXL.io Functions log the UIEs and/or CIEs in their respective AER Extended Capability.
3. CXL.io Functions generate PCIe ERR_ messages on the Link with Tag = 0.
4. DP_A forwards the ERR_ messages to the RCEC.
5. RCEC logs the errors in the Root Error Status register and then generates an MSI/ MSI-X, if enabled, in accordance with PCIe Base Specification.

Figure 12-3. RCD RCiEP Detects Error

12.2.1.4

Header Log and Handling of Multiple Errors

Unmasked CXL protocol and link errors are captured in the Uncorrectable Error Status register and the Correctable Error Status register (see [Section 8.2.4.17.1](#) and [Section 8.2.4.17.4](#), respectively). In the scenarios where multiple bits are set in the Uncorrectable Error Status register, the First Error Pointer field in the Error Capabilities and Control register (see [Section 8.2.4.17.6](#)), if valid, points to the first uncorrectable error that was captured. The First Error Pointer is valid if the associated bit of the Uncorrectable Error Status register is set. Otherwise, it is invalid. By definition, First Error Pointer is considered invalid if bit 5 of that field is set to 1. For certain uncorrectable errors, the specification requires that the component capture part of the message header, called Error Header, in the Header Log register. [Section 8.2.4.17.1](#) defines the format of the Error Header for each error.

If the Multiple_Header_Recording_Capability bit in the Error Capabilities and Control register (see [Section 8.2.4.17.6](#)) is set, the component is capable of recording multiple Error Headers in the order in which they are detected. If header logging resources are unavailable when an unmasked uncorrectable error is detected, the corresponding error status bit is set to 1; however, the Error Header is not recorded in the Header Log register. After software has consumed the error to which the First Error Pointer points, software writes 1 to the corresponding error status bit to indicate that. The error status bit may remain set if there was another occurrence of the same error. If any bit in the Uncorrectable Error Status register remains set after this software action, the component must atomically update the Header Log register and the First Error Pointer to point to the next recorded error. If no other error is recorded, the component shall update the First Error Pointer to an invalid value. If `Multiple_Header_Recording_Capability=1`, it is recommended that software not clear

the Status bit other than the one pointed to by the First Error Pointer. If software violates this condition, the state of the Header Log register in the presence of other recorded errors is undefined.

12.2.2 CXL Root Ports, Downstream Switch Ports, and Upstream Switch Ports

CXL protocol errors detected by CXL root ports, DSPs, and USPs are escalated and reported using PCIe error-reporting mechanisms as UIEs and/or CIEs. It is strongly recommended that CXL.cachemem protocol errors that are detected by a CXL root port be logged as CIEs or UIEs in the root port's AER Extended Capability. The Error Source Identification register logs the Bus, Device, and Function Numbers of the Root Port itself. If the CXL.cachemem protocol errors detected by a CXL root port are logged as CIEs or UIEs in an RCEC's AER Extended Capability, it is recommended that the System Firmware populate an RDPAS record (see [Section 9.18.1.5](#)) to establish the association between the RCEC and the root port.

The OS error handler may begin by inspecting the Root Port AER Extended Capability and follow PCIe rules to discover the error source. The Platform Software Error Handler may interrogate the Platform-specific error logs in addition to the error logs defined in PCIe Base Specification and this specification.

If the CXL.cachemem errors are logged in an RCEC and the CEDT includes RDPAS structures (see [Section 9.18.1.5](#)) that reference the RCEC, the OS handler may consult those RDPAS structures to locate the CXL root port that is the error source.

12.2.3 CXL Device Error Handling

Whenever a CXL device returns data that is either known to be bad or suspect, the device must ensure that the consumer of the data is made aware of the nature of the data, either at the time of consumption or prior to data consumption. This allows the consumer to take appropriate containment action.

CXL defines two containment mechanisms - poison and viral:

- **Poison:** Return data on CXL.io and CXL.cachemem may be tagged as poisoned.
- **Viral:** CXL.cachemem supports viral, which is mainly used to indicate more-severe error conditions at the device (see [Section 12.4](#)). Any data returned by a device on CXL.cachemem after the device has communicated Viral is considered suspect, even if the data is not explicitly poisoned.

A device must set the MetaField to No-Op in the CXL.cachemem return response when the Metadata is suspect.

If a CXL component is not in the Viral condition, the component shall poison the data message on the CXL interface whenever the data being included is known to be bad or suspect.

If Viral is enabled and a CXL component is in the Viral condition, it is recommended that the component not poison the subsequent data responses on the CXL.cachemem interface to avoid error pollution.

The Host may send poisoned data to the CXL-connected device. How the CXL device responds to Poison is device specific but must follow PCIe guidelines. The device must consciously make a decision about how to handle poisoned data. In some cases, simply ignoring poisoned data may lead to Silent Data Corruption (SDC). A CXL device is required to keep track of any poison data that the device receives on a 64-byte granularity.

Any device errors that cannot be handled with Poison indication shall be signaled by the device back to the Host as messages since there are no error pins. To that end, [Table 12-2](#) shows a summary of the error types and their mappings, and error reporting guidelines for devices that do not implement Memory Error Logging and Signaling Enhancements (see [Section 12.2.3.2](#)).

For devices that implement Memory Error Logging and Signaling Enhancements, [Section 12.2.3.2](#) describes how memory errors are logged and signaled. Such devices should follow [Table 12-2](#) for dealing with all non-memory errors.

Table 12-2. Device-specific Error Reporting and Nomenclature Guidelines

Error Severity	Definition/Example	Signaling Options (SW picks one)	Logging ¹	Host HW/FW/SW Response
Corrected	Memory single bit error corrected via ECC	MSI or MSI-X to Device driver	Device-specific registers	Device-specific flow in Device driver
Uncorrected Recoverable	UC errors from which the Device can recover, with minimal or no software help (e.g., error localized to single computation)	MSI or MSI-X to driver	Device-specific registers	Device-specific flow in driver (e.g., discard results of suspect computation)
Uncorrected NonFatal	Equivalent to PCIe UCNF, contained by the device (e.g., write failed, memory error that affects many computations)	MSI or MSI-X to Device Driver	Device-specific registers	Device-specific (e.g., reset affected device) flow in driver. Driver can escalate through software.
		PCIe AER Internal Error	Device-specific registers + PCIe AER	System FW/SW AER flow, ends in reset
Uncorrected Fatal	Equivalent to PCIe UCF, poses containment risk (e.g., command/parity error, Power management Unit ROM error)	PCIe AER Internal error	Device-specific registers + PCIe AER	System FW/SW AER flow, ends in reset
		AER + Viral		System FW/SW Viral flow

1. For CXL devices that implement memory error logging and signaling enhancements (see [Section 12.2.3.2](#)), the memory error logging and signaling mechanisms are defined by the CXL specification.

In keeping with the standard error logging requirements, all error logs should be sticky.

12.2.3.1 CXL.cache and CXL.mem Errors

If demand accesses to memory result in an uncorrected data error, the CXL device must return data with poison. The requester (processor core or a peer device) is responsible for dealing with the poison indication. The CXL device should not signal an uncorrected error along with the poison. If the processor core consumes the poison, the error will be logged and signaled by the Host.

Any non-demand uncorrected errors detected by a device (e.g., memory scrub logic in CXL device memory controller) that does not support the Memory Error Logging and Signaling Enhancements (see [Section 12.2.3.2](#)) will be signaled to the device driver via a device MSI or MSI-X. Any corrected memory errors will be signaled to the device driver via a device MSI or MSI-X. The driver may choose to deallocate memory pages that have repeated errors. Neither the platform firmware nor the OS directly deal with these errors. An eRCD may implement the capabilities described in [Section 12.2.3.2](#), in which case a device driver is not required.

If a CXL component is unable to positively decode a CXL.mem address, the handling is described in [Section 8.2.4.20.2](#). If a component does not implement HDM Decoders (see [Section 8.2.4.20](#)), the component shall drop such a write transaction and return all 1s in response to such a read transaction.

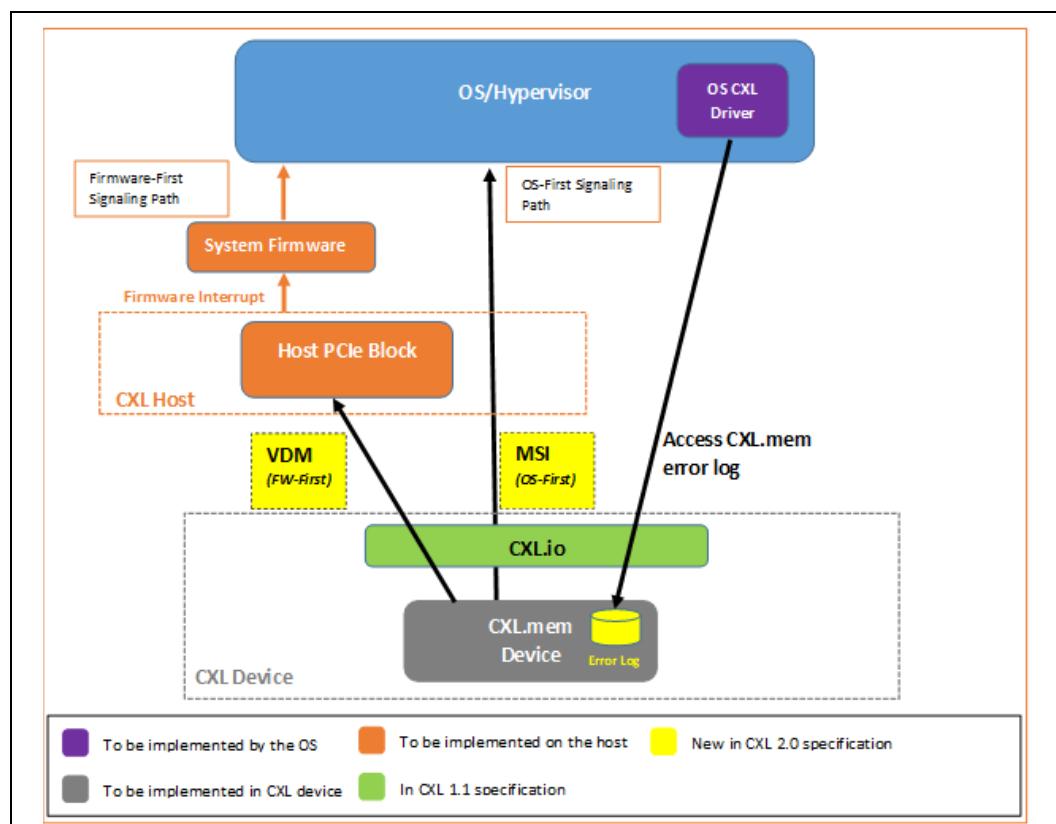
12.2.3.2 Memory Error Logging and Signaling Enhancements

Errors in memory may be encountered during a demand access or independent of any request issued to the memory. It is important to log sufficient data about such errors to enable the use of host platform-level RAS features, such as page retirement, without dependence on a driver.

In addition, general device events that are unrelated to the media, including changes in the device's health or environmental conditions detected by the device, need to be reported using the same general-event logging facility.

Figure 12-4 illustrates a use case where the two methods of signaling supported by a CXL.mem device - VDM and MSI/MSI-X – are used by a host to implement Firmware-first and OS-first error handling.

Figure 12-4. CXL Memory Error Reporting Enhancements



A CXL device that supports the Memory Error Logging and Signaling Enhancements capability must log such errors locally and expose the error log to system software via the MMIO Mailbox (see [Section 8.2.8.4.3](#)). Reading an error record from the mailbox will not automatically result in deletion of the error record on the device. An explicit clear operation is required to delete an error record from the device. To support error record access and deletion, the device shall implement the Get Event Records and Clear Event Records commands.

Both operations must execute atomically. Furthermore, all writes or updates to the error records by the CXL.mem device must also execute atomically.

Using these two operations, a host can retrieve an error record as follows:

12.2.3.3

CXL Device Error Handling Flows

RCD errors may be sourced from a Root Port (RP) or Endpoint (RCiEP). For the purpose of differentiation, RCiEP-sourced errors shall use a tag value of 0, whereas RP-sourced errors shall use a tag of nonzero value.

Errors detected by the CXL device shall be communicated to the host via PCIe Error messages across the CXL.io link. Errors that are not related to a specific Function within the device (Non-Function errors) and not reported via an MSI/MSI-X are reported to the Host via PCIe error messages where the errors can be escalated to the platform.

The Upstream Port reports non-function errors to all EPs/RCiEPs where they are logged. Each EP/RCiEP reports the non-function-specific errors to the host via PCIe error messages. Software should be aware that although an RCiEP does not have a software-visible link, the RCiEP may still log link-related errors.

At most, one error message of a given severity is generated for a multi-function 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.

Errors generated by the RCD RCiEP will be sent to the corresponding RCEC. Each RCiEP must be associated with no more than one RCEC. Errors generated by a CXL component that is part of a CXL VH shall be logged in the CXL Root Port.

12.3

Isolation on CXL.cache and CXL.mem

Isolation on CXL.cache and CXL.mem is an optional normative capability of a CXL Root Port. Such isolation halts traffic on the respective protocol. Further, once triggered, the Root Port synthesizes the response for all pending and subsequent transactions on that protocol. This is further described in [Section 12.3.1](#) and [Section 12.3.2](#), respectively.

The specification defines two trigger mechanisms:

- Link Down – If a Root Port supports CXL.cache isolation and software enables CXL.cache isolation, a Link Down condition shall unconditionally trigger CXL.cache isolation. If a Root Port supports CXL.mem isolation and software enables CXL.mem isolation, a Link Down condition shall unconditionally trigger CXL.mem isolation.
- Transaction timeout – A Root Port that supports CXL.cache isolation may be capable of being configured in such a way that a CXL.cache timeout triggers CXL.cache isolation. A Root Port that supports CXL.mem isolation may be capable of being configured in such a way that a CXL.mem timeout triggers CXL.mem isolation.

Note:

Transaction Timeout Value settings for CXL.cache and CXL.mem: The system needs to ensure that timeouts are appropriately set up. For example, a timeout should not be so short that isolation is triggered due to a non-erroneous, long-latency access to a CXL device. Software may need to temporarily disable the triggering of isolation upon timeout if one or more devices are being transitioned to a state (e.g., firmware update) where the device may violate the timeout.

The primary purpose of the isolation action is to complete pending and subsequent transactions that are associated with the isolated root port quickly, with architected semantics, after isolation is triggered. Since system memory and system caches must generally be assumed to be corrupted, software recovery generally relies on software to identify all software threads, VMs, containers, etc., whose system state might be corrupted, and then terminating them. Other software recovery mechanisms are also possible, and they are beyond the scope of this specification.

A Root Port indicates support for Isolation by implementing the CXL Timeout and Isolation Capability structure (see [Section 8.2.4.24](#)). The structure contains the capability, control, and status bits for both Transaction Timeout and Isolation on both CXL.cache and CXL.mem. Both Timeout and Isolation are disabled by default and must be explicitly and individually enabled by software for each protocol before they can be triggered. When Isolation is enabled for either CXL.cache or CXL.mem, software can optionally configure the Root Port to force a Link Down condition if the respective protocol enters Isolation.

When Isolation is entered, the Root Port, if capable, signals an MSI/MSI-X or send an ERR_COR Message if enabled. Software may also choose to rely only on mandatory synchronous exception handling (see [Section 12.3.1](#) and [Section 12.3.2](#)). Software may read the CXL Timeout and Isolation Status register to determine whether a Timeout or Isolation has occurred on CXL.cache and/or CXL.mem and if the Isolation was triggered due to a Timeout or due to a Link Down condition. The software must explicitly clear the corresponding Isolation status bits (see [Section 8.2.4.24.3](#)) for the root port to exit Isolation. The link must transition through the Link Down state before software can attempt re-enumeration and device recovery.

12.3.1

CXL.cache Transaction Layer Behavior during Isolation

This section specifies the CXL.cache Transaction Layer's behavior while the Root Port is in Isolation.

The Root Port shall handle host requests that would ordinarily be mapped to (H2D) CXL.cache messages in the following manner.

For each host snoop that would ordinarily be mapped to (H2D) CXL.cache request messages:

- If the host is tracking the device as a possible exclusive owner of the line, then data is treated as poison.
- Else if the host knows the device can only have a Shared or Invalid state for the line, then the device cache is considered Invalid (no data poisoning is needed).

IMPLEMENTATION NOTE

Exclusive vs. Shared/Invalid may be known based on an internal state within the host.

The Root Port timeout detection logic shall account for partial responses. For example, if the Root Port observes that the data is returned on the D2H Data channel in a timely manner, but no D2H Rsp was observed for a sufficient length of time, the Root Port shall treat it as a CXL.cache timeout.

12.3.2

For each pending Pull that is mapped to H2D CXL.cache Response of type *WritePull* which expects a data return, the Root Port must treat the returned data as poison.

CXL.mem Transaction Layer Behavior during Isolation

This section specifies the CXL.mem Transaction Layer's behavior while the CXL Root Port is in Isolation.

The Root Port shall handle host requests that it would ordinarily map to (M2S) CXL.mem messages in the following manner:

- For each host request that would ordinarily be mapped to CXL.mem Req and Rwd:
 - For Read transactions, the CXL Root Port synthesizes a synchronous exception response. The specific mechanism of synchronous exception response is CXL Root Port implementation specific. An example of a synchronous exception response would be returning data with Poison.
 - For non-read transactions, the CXL Root Port synthesizes a response as appropriate. The specific mechanism of the synthesized response is implementation specific. An example would be returning a completion (NDR) for a write (Rwd) transaction.

12.4

CXL Viral Handling

CXL links and CXL devices are expected to be Viral compliant. Viral is an error-containment 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 generating a Viral indication. Any entity that is capable of reporting Uncorrected_Fatal errors must also be capable of generating a Viral indication.

CXL.cache and CXL.mem are pre-enabled with the Viral concept. Viral needs to be communicated in both directions. When Viral is enabled and the Host runs into a Viral condition, the Host shall communicate Viral across CXL.cache and/or CXL.mem to all downstream components. The Viral indication must arrive before any data that may have been affected by the error (general Viral requirement). If the host receives a Viral indication from any CXL components, the Host shall propagate Viral to all downstream components.

All types of Conventional Resets shall clear the viral condition. CXL Resets and FLRs shall have no effect on the viral condition.

12.4.1

Switch Considerations

Viral is enabled on a per-vPPB basis and the expectation is that if Viral is enabled on one or more DSPs, then Viral will also be enabled on the USP within a VCS.

A Viral indication received on any port transitions that VCS into the Viral state, but does not trigger a new uncorrected fatal error inside the switch. A Viral indication in one VCS has no effect on other VCSs within the switch component. The switch continues to process all CXL.io traffic targeting the switch and forward all traffic. All CXL.cache and CXL.mem traffic sent to all ports within the VCS is considered to have the Viral bit set. The Viral indication shall propagate from an input port to all output ports in the VCS.

faster than any subsequent CXL.cache or CXL.mem transaction. The Viral bit is propagated across upstream links and links connected to SLDs with the Viral LD-ID Vector (see [Table 4-10](#)) set to 0 for compatibility with the CXL 1.1 specification.

If the switch detects an uncorrected fatal error, the switch must determine whether that error affects one or multiple VCSs. Any affected VCS enters the Viral state, sets the Viral_Status bit (see [Section 8.1.3.3](#)) to indicate that a Viral condition has occurred, asserts the Viral bit in all CXL.cache and CXL.mem traffic sent to all ports within the VCS, and then sends an AER message. The affected VCS continues to forward all CXL traffic.

Hot-remove and hot-add of devices below DSPs have no effect on the Viral state of the VCS within the switch.

If the switch has configured and enabled MLD ports, then there are additional considerations. When a VCS with an MLD port enters the Viral state, the VCS propagates the Viral indication to LDs within the MLD Component by setting the Viral bit in the Viral LD-ID Vector (see [Table 4-10](#)) for the LDs in that VCS. If an uncorrected fatal error causes one or more VCSs to enter the Viral state, then the corresponding bits in the Viral LD-ID Vector shall be set. An LD within an MLD component that has entered the Viral state sets the Viral bit in CXL.mem traffic with the Viral LD-ID Vector mask set to identify all the LD-IDs associated with all the affected VCSs. The indication from each LD-ID propagates the Viral state to all associated VCSs that have Viral containment enabled.

12.4.2 Device Considerations

Although the device's reaction to Viral is device specific, the device is expected to take error-containment actions that are consistent with Viral requirements. Mainly, the device must prevent bad data from being committed to permanent storage. If the device is connected to any permanent storage or to an external interface that may be connected to permanent storage, then the device is required to self-isolate to be Viral compliant. This means that the device has to take containment actions without depending on help from the Host.

The containment actions taken by the device must not prevent the Host from making forward progress. This is important for diagnostic purposes as well as for avoiding error pollution (e.g., withholding data for read transactions to device memory may cause cascading timeouts in the Hosts). Therefore, on Viral detection, in addition to the containment requirements, the device shall:

- Drop writes to the persistent HDM ranges on the device or connected to the device.
- Always return a Completion response.
- Set MetaField to No-Op in all responses that carry MetaField.
- Fail the Set Shutdown State command (defined in [Section 8.2.9.9.3.5](#)) with an Internal Error when attempting to change the state from "dirty" to "clean".
- Not transition the Shutdown State to "clean" after a GPF flow.
- Commit to the persistent HDM ranges any writes that were completed over the CXL interface before receipt of the viral condition.
- Keep responding to snoops.
- Complete pending writes to Host memory.
- Complete all reads and writes to Device volatile memory.

When the device itself runs into a Viral condition and Viral is enabled, the device shall:

- Set the Viral Status bit to indicate that a Viral condition has occurred

- Containment – Take steps to contain the error within the device (or logical device in an MLD component) and follow the Viral containment steps listed above.
- Communicate the Viral condition back up to CXL.cache and CXL.mem, toward the Host.
 - Viral propagates to all devices in the Virtual Hierarchy, including to the host.

Viral Control and Status bits are defined in the DVSEC (see [Chapter 3.0](#) for details).

12.5

Maintenance

Maintenance operations may include media maintenance, media testing, module testing, etc. A maintenance operation is identified by a Maintenance Operation Class and a Maintenance Operation Subclass. A Device may support one or more Maintenance Operation Subclasses related to a Maintenance Operation Class. See [Table 8-102](#).

The Device may use Event Records to notify the System Software or System Firmware about needing a maintenance operation. When the Device requires maintenance, the Maintenance Needed bit in the Event Record Flags is set to 1, while the class of recommended maintenance operation is indicated by the Maintenance Operation Class field. See [Table 8-43](#).

The Perform Maintenance command (see [Section 8.2.9.7.1](#)) initiates a maintenance operation. The maintenance operation to be executed is specified in the input payload by the Maintenance Operation Class field and the Maintenance Operation Subclass field.

12.6

CXL Error Injection

The major aim of error-injection mechanisms is to allow system validation and system firmware/software development, etc., the means to create error scenarios and error-handling flows. To this end, a CXL Upstream Port and Downstream Port are recommended to implement the following error injection hooks to a specified address (where applicable):

- One type of CXL.io UC error (optional - similar to PCIe)
 - CXL.io is always present in any CXL connection
- One type of CXL.cache UC error (if applicable)
- One type of CXL.mem UC error (if applicable)
- Link Correctable errors
 - Transient errors and
 - Persistent errors
- Returning Poison on a read to a specified address (CXL.mem only)

Error injection interfaces are documented in [Chapter 14.0](#).

§ §

13.0

Performance Considerations

CXL provides a low-latency, high-bandwidth path for an accelerator to access the system. Performance on CXL is dependent on a variety of factors. [Table 13-1](#) captures the main CXL performance attributes.

Table 13-1.

CXL Performance Attributes

Characteristic	CXL via Flex Bus (if PCIe Gen 4)	CXL via Flex Bus (if PCIe Gen 5)	CXL via Flex Bus (if PCIe Gen 6)
Width	16 Lanes		
Link Speed	16 GT/s	32 GT/s	64 GT/s
Total Bandwidth per link ¹	32 GB/s	64 GB/s	128 GB/s

1. Achieved bandwidth depends on protocol and payload size. Expect 60-90% efficiency on CXL.cache and CXL.mem. Efficiency similar to PCIe* on CXL.io.

In general, it is expected that the Upstream Ports and Downstream Ports are rate-matched. However, if the implementations are not rate-matched, it would require the faster of the implementations to limit the rate of its protocol traffic to match the slower (including bursts) whenever there is no explicit flow-control loop.

CXL allows accelerators/devices to coherently access host memory, and allows memory attached to an accelerator/device to be mapped into the system address map and to be accessed directly by the host as writeback memory. To support this, it supports Coherency models as described in [Section 2.2.1](#) and [Section 2.2.2](#).

13.1

Performance Recommendations

To minimize buffering requirements and provide good responsiveness, CXL components need to strive for low latency. Specific transaction flows merit special attention, depending on component type, to ensure the system performance is not negatively impacted.

It is recommended that components meet the latency targets listed in [Table 13-2](#). These targets are measured at the component pins in an otherwise idle system. Measurements are for average idle response times, meaning that a single message is transmitted to the component and the response is received from the component before another message is transmitted to the component. Messages used for average measurements should have their addresses randomly distributed. All the targets listed in [Table 13-2](#) assume a x16 link at full speed (64 GT/s) with IDE disabled.

For CXL devices and switches, CDAT (see [Section 8.1.11](#)) provides a mechanism for

Table 13-2. Recommended Latency Targets for Selected CXL Transactions

Case ¹	Component	Protocol	Received Message	Transmitted Message	Latency Target
1	Type 1/Type 2	CXL.cache	H2D Req Snoop (Miss Case)	D2H Resp Snoop Response	90-150 ns
2			H2D Resp WritePull	D2H Data	65 ns
3	Type 3 (DDR)	CXL.mem	M2S Req MemRd	S2M DRS MemData	80 ns
4			M2S Rwd MemWr	S2M NDR Cmp	40 ns
5	Host		S2M BISnp	M2S BIRsp	90 ns

- Case 1:** The range provided accounts for implementation trade-offs, with a dedicated CXL interface snoop filter providing the lowest latency, and snoop filters embedded in the Device cache hierarchy resulting in higher latency.

Case 2: Assumes write data is ready to transmit in a CXL output buffer.

Cases 3 and 4: Applies to Type 3 Devices using DRAM media intended to provide system-level performance comparable to DDR DIMMs. Such Devices are assumed to be relatively simple, small, and low-power, and not complex, multi-ported, or pooled-memory Devices. Noting that Case 3 is more aggressive than the targets in [Table 13-3](#), the [Table 13-3](#) targets will be lower than the Case 3 target in these Devices. Memory Devices that use slower media, such as some persistent-memory types, will have longer latencies that must be considered by system designers.

Case 5: The BISnp will cause the host to resolve coherence for an HDM-DB memory address owned by a device that sent the BISnp. The target latency for resolving coherence is for the simple case in which the host does not have the cacheline in any host-managed cache (this scope includes peer CXL devices that are using CXL.cache).

reporting both lowest latency and highest bandwidth to Host software (see ACPI Specification for the definition of the System Locality Latency and Bandwidth Information Structure). That reporting structure, however, only assists Host software. Hardware and system designers must consider the topologies and components of interest at design time. Hardware designers should consider the maximum latency that their component needs to tolerate while still maintaining high-link bandwidth, sizing outstanding requests, and data buffers accordingly.

The QoS Telemetry mechanism (see [Section 3.3.4](#)) defines a mechanism by which Hosts can dynamically adjust their request rate to avoid overloading memory devices. This mechanism is particularly important for MLD components that are shared among multiple Hosts.

At the Link Layer, the maximum loaded latencies listed in [Table 13-3](#) are recommended. If not adhered to, the link between two ports risks being throttled to less than the line rate. These recommendations apply to all CXL ports and both the CXL.cache and CXL.mem interfaces. The targets assume a x16 link with IDE disabled.

Table 13-3. Recommended Maximum Link Layer Latency Targets

Case ¹	Condition	Latency Target
1	Message received to Ack Transmitted	65 ns
2	Credit Received to Flit transmitted	50 ns

- Case 1:** Accounts for a sequence of 8 back-to-back flits with a clean CRC that needs to accumulate before the Ack can be transmitted. Applies only to links operating in 68B Flit mode and thus assumes a full link speed of 32 GT/s. Links operating in 256B Flit mode share Ack/Retry logic with PCIe and fall under any guidelines or requirements provided in PCIe Base Specification.

Case 2: In this case, the port lacks the Link Layer credits that are needed to transmit a message and then receives a credit update that enables transmission of the message. Assumes full link speed of 64 GT/s.

13.2

Performance Monitoring

Performance tuning and performance debug activities rely on performance counters that are located within different system components. This section introduces the Performance Monitoring infrastructure for CXL components.

The base hardware unit is called the CXL Performance Monitoring Unit (CPMU). A CXL component may have zero or more CPMU instances. Each CPMU instance includes one or more Counter Units. The registers associated with a CPMU are located by following the CXL Register Locator DVSEC with Register Block Identifier=4. Each CPMU instance in a Switch shall count events that are associated with a single Port. The CPMU instance associated with the Port can be identified by following the CXL Register Locator DVSEC in the Configuration Space of that Port. If a CXL multi-function device implements one or more CPMU instances, the Register Locator DVSEC that is associated with Function 0 shall point to them.

A CXL Event is defined as the occurrence of a specific component activity that is relevant to the operation of the CXL component. Events are grouped into Event Groups based on the type of activity that the Events represent. The pair <Event Vendor ID, Event Group ID> identifies an Event Group. Each Event Group can have up to 32 different types of events and each event is identified by a 5-bit Event ID. The tuple <Event Vendor ID, Event Group ID, Event ID> uniquely identifies the type of the Event. See [Table 13-5](#) for the list of CXL Events that are defined by this specification. The **Filter** column lists all the Filter ID values that are applicable to the Event Group. The Multiple Event Counting (**MEC**) column defines the Counter Unit behavior when it is configured to simultaneously count more than one event within the Event Group by setting multiple bits. If the **MEC** column indicates ADD, the Counter Unit shall *add* the occurrences of all the enabled events every clock, which may result in the Counter Data being incremented by a value of more than one within a single clock. If the **MEC** column indicates OR, the Counter Unit shall *logically or* the occurrences of all the enabled events every clock and the Counter Data shall never increment by more than one within any single clock.

None of the events defined in this specification are capable of incrementing the Counter Data by more than one per cycle. As such, software must set the Threshold field in the Counter Configuration register (see [Section 8.2.7.2](#)) to 1 when counting any event specified here.

A Counter Unit is capable of counting the occurrence of one or more events. Counter Units are capable of being configured to count a subset of the Events that the Counter Unit is capable of counting. A Counter Unit may be capable of being configured to take certain predefined actions when the count overflows. [Table 13-4](#) describes the three types of Counter Units.

Table 13-4. CPMU Counter Units

Counter Unit Type	Description
Fixed Function	Capable of counting a fixed set of one or more events within a single Event Group. Counting is halted when the Counter Unit is Frozen or Counter Enable=0.
Free-running	Capable of counting a fixed set of one or more events within a single Event Group. Not affected by freeze. The Counter Enabled bit is RO and always returns 1.
Configurable	Capable of counting any Events that are identified via CPMU Event Capabilities register (see Section 8.2.7.1.4), and may be configured by software to count a specific set of events in a specific manner. Counting is halted when the Counter Unit is Frozen or Counter Enable=0.

The CPMU register interface is defined in [Section 8.2.7](#). These registers are for Host Software or System Firmware consumption. The component must not expose CPMU registers or the underlying resources to an out-of-band agent if such an access may

interfere with the Host Software actions. Although a component may choose to implement a separate set of counters for out-of-band usage, use of such a mechanism is beyond the scope of this specification.

Table 13-5. Events under CXL Vendor ID (Sheet 1 of 3)

Event Group	Event ID	Mnemonic	Event Description	Filters	MEC ¹
00h (Base)	00h	Clock Ticks	Count the clock ticks of the always-on fixed-frequency clock that is used to increment the Counters. Every CPMU must allow counting of this event, either via a Fixed Function Counter Unit or via a Configurable Counter Unit.	None	N/A
01h-0Fh	00h - 1Fh	Reserved	Reserved	N/A	N/A
10h (CXL.cache D2H Req Channel)	D2H Req Opcode encoding	D2HReq.Opcde mnemonic	Counts the number of messages in the D2H Req message class with Opcode=Event ID that were initiated, or forwarded, or received by the component. The D2H Req message class Opcode mnemonics and the Opcode encodings are enumerated in Table 3-22 . All Event ID values corresponding to unused D2H Req message class Opcode encodings are reserved. For example, the mnemonic associated with the Event Group=10h, Event ID=01h is D2HReq.RdCurr and counts the number of RdCurr requests.	None	ADD
11h (CXL.cache D2H Rsp Channel)	D2H Rsp Opcode encoding	D2HRsp.Opcde mnemonic	Counts the number of messages in the D2H Rsp message class with Opcode=Event ID that were initiated, or forwarded, or received by the component. D2H Rsp message class Opcode mnemonics and Opcode encodings are enumerated in Table 3-25 . All Event ID values corresponding to unused D2H Rsp message class Opcode encodings are reserved. For example, the mnemonic associated with the Event Group=11h, Event ID=05h is D2HRsp.RspIHitSE and counts the number of RspIHitSE messages.	None	ADD
12h (CXL.cache H2D Req Channel)	H2D Req Opcode encoding	H2DReq.Opcde mnemonic	Counts the number of messages in the H2D Req message class with Opcode=Event ID that were initiated, or forwarded, or received by the component. The H2D Req message class Opcode mnemonics and the Opcode encodings are enumerated in Table 3-26 . All Event ID values corresponding to unused H2D Req message class Opcode encodings are reserved. For example, the mnemonic associated with the Event Group=12h, Event ID=02h is H2DReq.SnpInv and counts the number of SnpInv requests.	None	ADD
13h (CXL.cache H2D Rsp Channel)	H2D Rsp Opcode encoding	H2DRsp.Opcde mnemonic	Counts the number of messages in the H2D Rsp message class with Opcode=Event ID that were initiated, or forwarded, or received by the component. H2D Rsp message class Opcode mnemonics and Opcode encodings are enumerated in Table 3-27 . All Event ID values corresponding to unused H2D Rsp message class Opcode encodings are reserved. For example, the mnemonic associated with the Event Group=13h, Event ID=01h is H2DRsp.WritePull and counts the number of WritePull messages.	None	ADD
14h (CXL.cache Data)	00h	D2H Data	Counts the number of D2H Data messages that were initiated, or forwarded, or received by the component.	None	ADD
	01h	H2D Data	Counts the number of H2D Data messages that were initiated, or forwarded, or received by the component.	None	
	02h - 1Fh	Reserved	Reserved	N/A	N/A
15h-1Fh	00h - 1Fh	Reserved	Reserved	N/A	N/A

Table 13-5. Events under CXL Vendor ID (Sheet 2 of 3)

Event Group	Event ID	Mnemonic	Event Description	Filters	MEC¹
20h (CXL.mem M2S Req Channel)	M2S Req Opcode encoding	M2SReqOpcode mnemonic	Counts the number of messages in the M2S Req message class with Opcode=Event ID that were initiated, or forwarded, or received by the component. The M2S Req message class Opcode mnemonics and the Opcode encodings are enumerated in Table 3-35 . All Event ID values corresponding to unused M2S Req message class Opcode encodings are reserved. For example, the mnemonic associated with the Event Group=20h, Event ID=00h is M2SReq.MemInv and counts the number of MemInv requests.	Filter ID=0	ADD
21h (CXL.mem M2S Rwd Channel)	M2S Rwd Opcode encoding	M2SRwdOpcode mnemonic	Counts the number of messages in the M2S Rwd message class with Opcode=Event ID that were initiated, or forwarded, or received by the component. The M2S Rwd message class Opcode mnemonics and the Opcode encodings are enumerated in Table 3-41 . All Event ID values corresponding to unused M2S Rwd message class Opcode encodings are reserved. For example, the mnemonic associated with the Event Group=21h, Event ID=02h is M2SRwd.MemWrPtl and counts the number of MemWrPtl requests.	Filter ID=0	ADD
22h (CXL.mem M2S BIRsp Channel)	M2S BIRsp Opcode encoding	M2SBIRspOpcode mnemonic	Counts the number of messages in the M2S BIRsp message class with Opcode=Event ID that were initiated, or forwarded, or received by the component. The M2S BIRsp message class Opcode mnemonics and the Opcode encodings are enumerated in Table 3-45 . All Event ID values corresponding to unused M2S BIRsp message class Opcode encodings are reserved. For example, the mnemonic associated with the Event Group=22h, Event ID=00h is M2SBIRsp.BIRspI and counts the number of BIRspI messages.	Filter ID=0	ADD
23h (CXL.mem S2M BISnp Channel)	S2M BISnp Opcode encoding	S2MBISnpOpcode mnemonic	Counts the number of messages in the S2M BISnp message class with Opcode=Event ID that were initiated, or forwarded, or received by the component. The S2M BISnp message class Opcode mnemonics and the Opcode encodings are enumerated in Table 3-47 . All Event ID values corresponding to unused S2M BISnp message class Opcode encodings are reserved. For example, the mnemonic associated with the Event Group=23h, Event ID=00h is S2MBISnp.BISnpCur and counts the number of BISnpCur requests.	Filter ID=0	ADD
24h (CXL.mem S2M NDR Channel)	S2M NDR Opcode encoding	S2MNDROpcode mnemonic	Counts the number of messages in the S2M NDR message class with Opcode=Event ID that were initiated, or forwarded, or received by the component. The S2M NDR message class Opcode mnemonics and the Opcode encodings are enumerated in Table 3-50 . All Event ID values corresponding to unused S2M NDR message class Opcode encodings are reserved. For example, the mnemonic associated with the Event Group=24h, Event ID=02h is S2MNDR.Cmp-E and counts the number of Cmp-E messages.	Filter ID=0	ADD
25h (CXL.mem S2M DRS Channel)	S2M DRS Opcode encoding	S2MDRSOpcode mnemonic	Counts the number of messages in the S2M DRS message class with Opcode=Event ID that were initiated, or forwarded, or received by the component. The S2M DRS message class Opcode mnemonics and the Opcode encodings are enumerated in Table 3-53 . All Event ID values corresponding to unused S2M DRS message class Opcode encodings are reserved. For example, the mnemonic associated with the Event Group=25h, Event ID=00h is S2MDRS.MemData and counts the number of MemData messages.	Filter ID=0	ADD
26h - 7FFFh	00h-1Fh	Reserved	Reserved	N/A	N/A

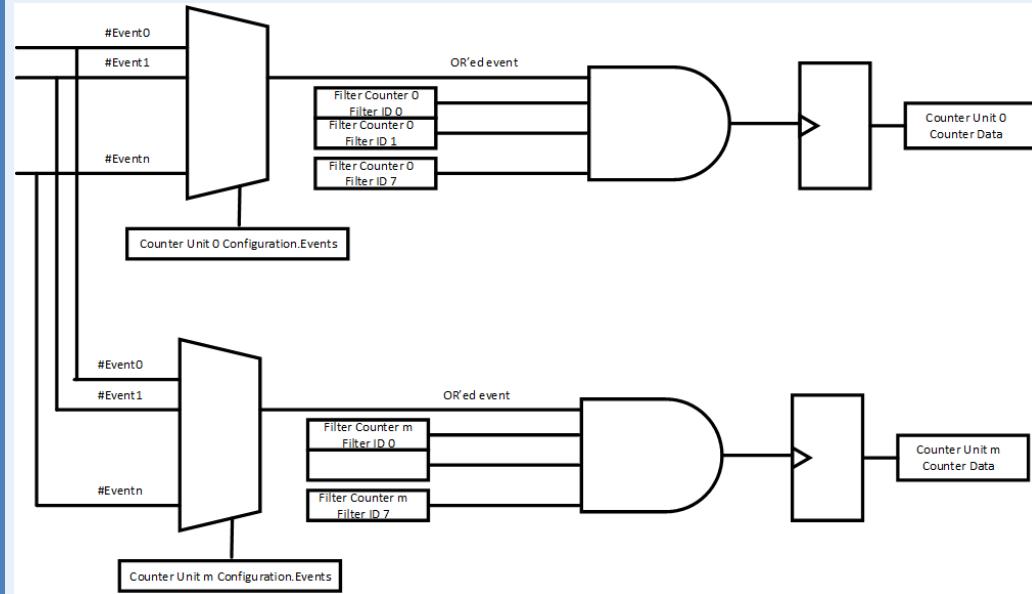
Table 13-5. Events under CXL Vendor ID (Sheet 3 of 3)

Event Group	Event ID	Mnemonic	Event Description	Filters	MEC¹
8000h (DDR Interface) ²	00h	ACT Count	Counts the number of DRAM Activate commands that were issued by the Memory Controller associated with this CPMU.	Filter ID=1	ADD
	01h	PRE Count	Counts the number of DRAM Precharge commands that were issued by the Memory Controller associated with this CPMU.	Filter ID=1	
	02h	CAS Rd	Counts the number of DRAM Column Address Strobe read commands that were issued by the Memory Controller associated with this CPMU.	Filter ID=1	
	03h	CAS Wr	Counts the number of DRAM Column Address Strobe write commands that were issued by the Memory Controller associated with this CPMU.	Filter ID=1	
	04h	Refresh	Counts the number of DRAM Refresh commands that were issued by the Memory Controller associated with this CPMU.	Filter ID=1	
	05h	Self Refresh Entry	Counts the number of Self Refresh Entry commands that were issued by the Memory Controller associated with this CPMU.	Filter ID=1	
	06h	RFM	Counts the number of Refresh Management (RFM) commands that were issued by the Memory Controller associated with this CPMU.	Filter ID=1	
	07h - 1Fh	Reserved	Reserved	N/A	N/A
8001h - FFFFh	Reserved	Reserved	Reserved	N/A	N/A

1. In the **MEC** column, ADD indicates that the Counter Unit shall add the occurrences of all the enabled events every clock, which may result in the Counter Data being incremented by a value of more than one within a single clock. In the **MEC** column, OR indicates that the Counter Unit shall logically or the occurrences of all the enabled events every clock and the Counter Data shall never increment by more than one within any single clock.
2. See JEDEC DDR5 Specification for the definition of the specific commands that are referenced in the **Event Description** column.

IMPLEMENTATION NOTE

Event Selection and Counting Summary



This diagram pictorially represents how a simple CPMU that supports a single Event Group and two Configurable Counters counts events.

1. Every clock, the events selected via the Events field in the Counter Configuration register are OR'ed together.
2. The output of step 1, labeled “OR'ed event” is subjected to various configured filters.
3. The output of step 2 is added to the Counter Data.

Note: For readability, the Threshold, Edge, and Invert controls are not shown.

§ §

14.0 CXL Compliance Testing

14.1 Applicable Devices under Test (DUTs)

The tests outlined in this chapter are applicable to all devices that support alternate protocol negotiation and are capable of CXL only or CXL and PCIe* protocols. The tests are broken into the different categories corresponding to the different chapters of CXL specification, starting with [Chapter 3.0](#).

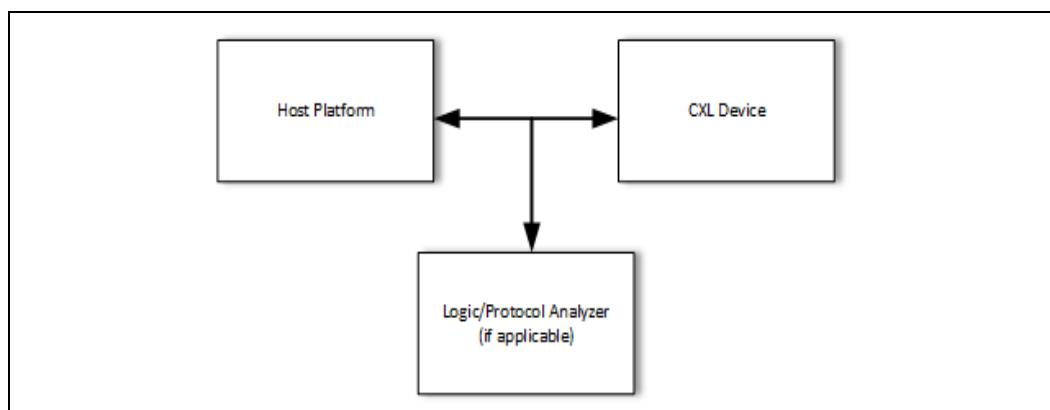
14.2 Starting Configuration/Topology (Common for All Tests)

In most tests, the initial conditions assumed are as follows (deviations from these conditions are pointed out in specific tests, if applicable): System is powered on, running in test environment OS, device-specific drivers have loaded on device, and link has trained to supported CXL modes. All error status registers should be cleared on the DUT.

Some tests make assumptions about only one CXL device being present in the system – this is identified in relevant tests. If nothing is mentioned, there is no limit on the number of CXL devices present in the system; however, the number of DUTs is limited to what the test software can support.

Certain tests may also require the presence of a protocol analyzer to monitor flits on the physical link for determining Pass or Fail results.

Figure 14-1. Example Test Topology



Each category of tests has certain device capability requirements to exercise the test patterns. The associated registers and programming is defined in the following sections.

See [Section 14.16](#) for the registers that are applicable to the tests in the following sections.

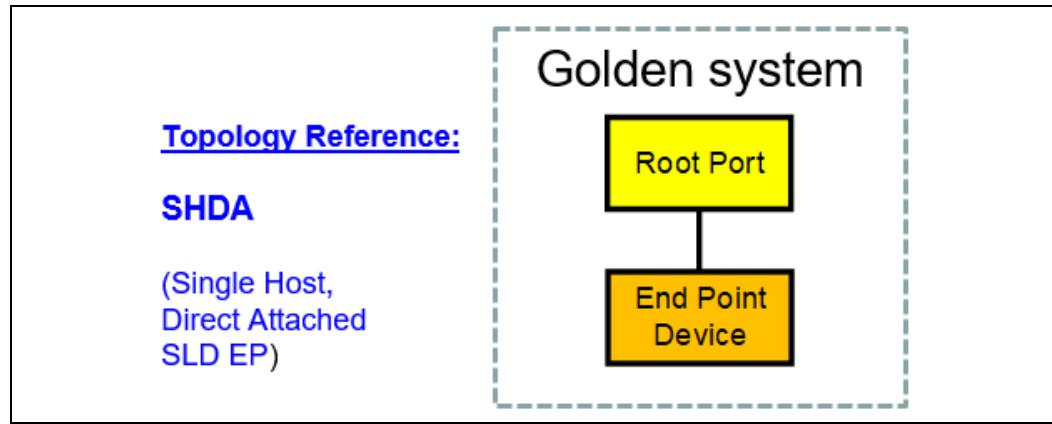
14.2.1 Test Topologies

Some tests may require a specific topology to achieve the desired requirements. Throughout this chapter there will be references to these topologies as required. This section of the document will describe these topologies at a high level to provide context for the intended test configuration.

14.2.1.1 Single Host, Direct Attached SLD EP (SHDA)

Figure 14-2 is the most direct connected topology between a root port and an endpoint device.

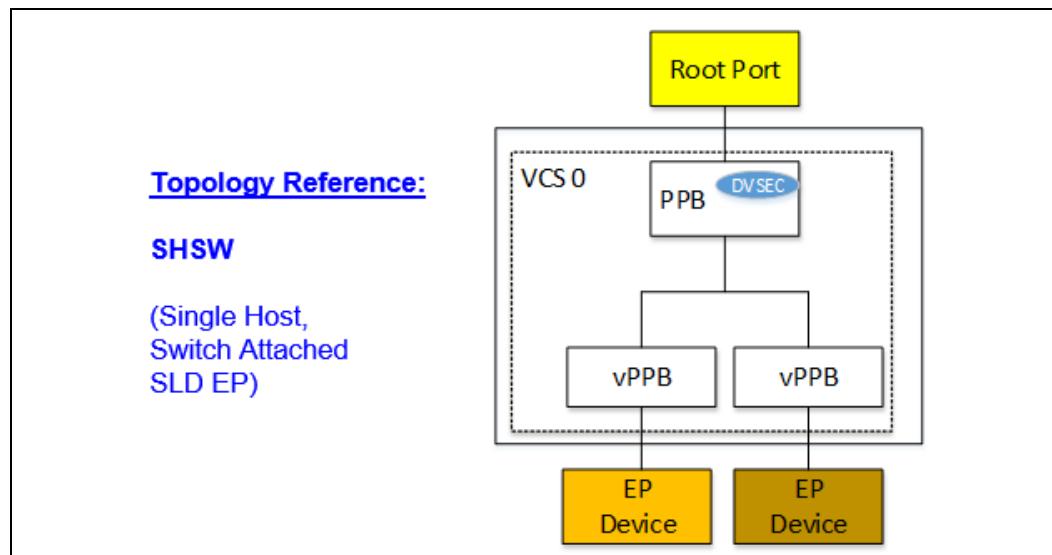
Figure 14-2. Example SHDA Topology



14.2.1.2 Single Host, Switch Attached SLD EP (SHSW)

Figure 14-3 is the initial configuration for using a CXL-capable switch in the test configurations.

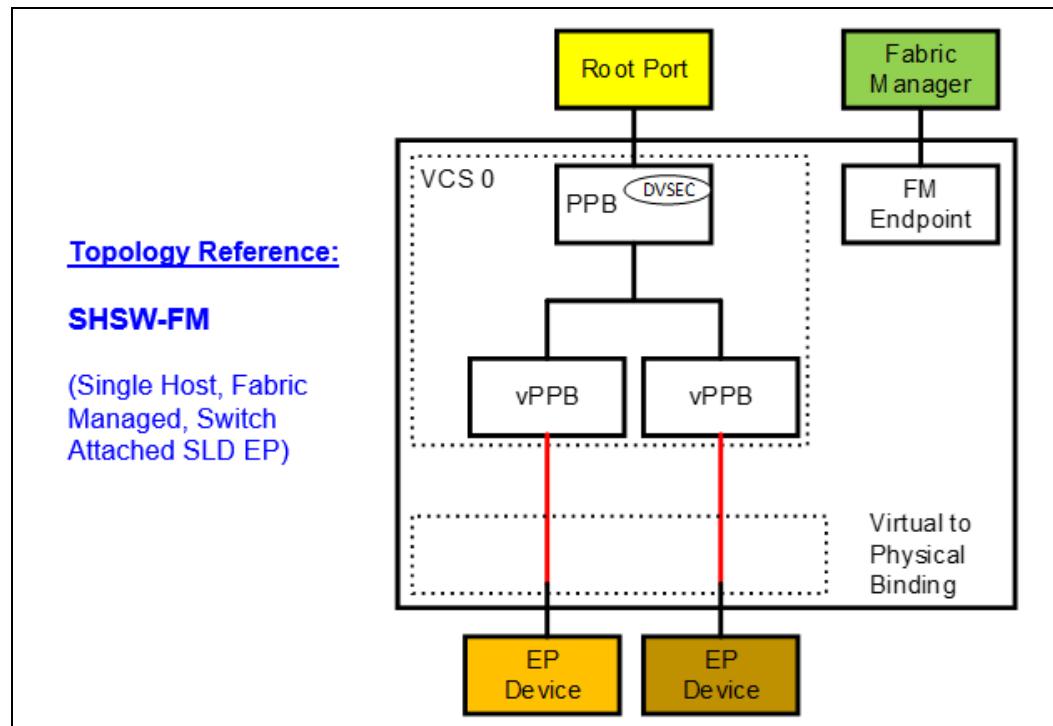
Figure 14-3. Example Single Host, Switch Attached, SLD EP (SHSW) Topology



14.2.1.3 Single Host, Fabric Managed, Switch Attached SLD EP (SHSW-FM)

Figure 14-4 shows the configuration which will use the Fabric Manager as part of the test configuration.

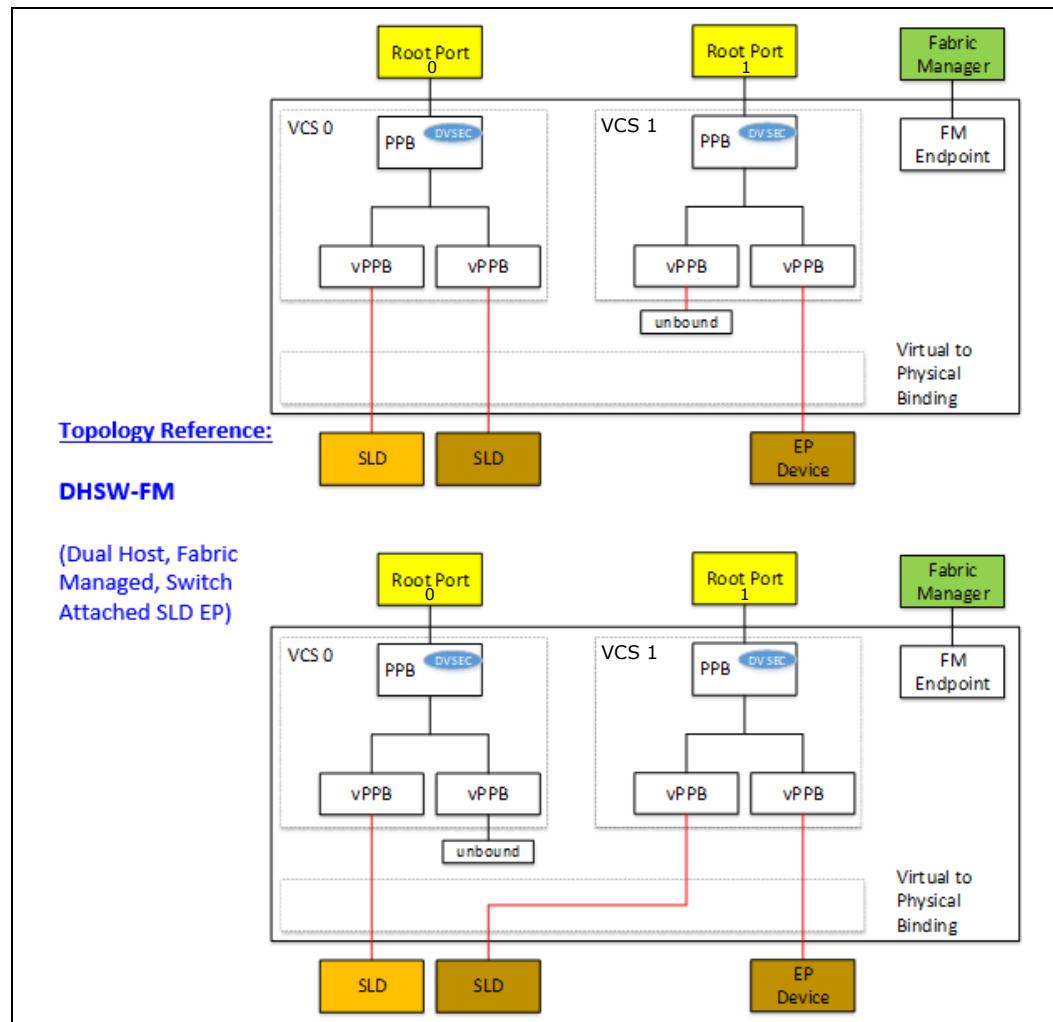
Figure 14-4. Example SHSW-FM Topology



14.2.1.4 Dual Host, Fabric Managed, Switch Attached SLD EP (DHSW-FM)

Figure 14-5 shows an example configuration topology for having dual hosts during a test.

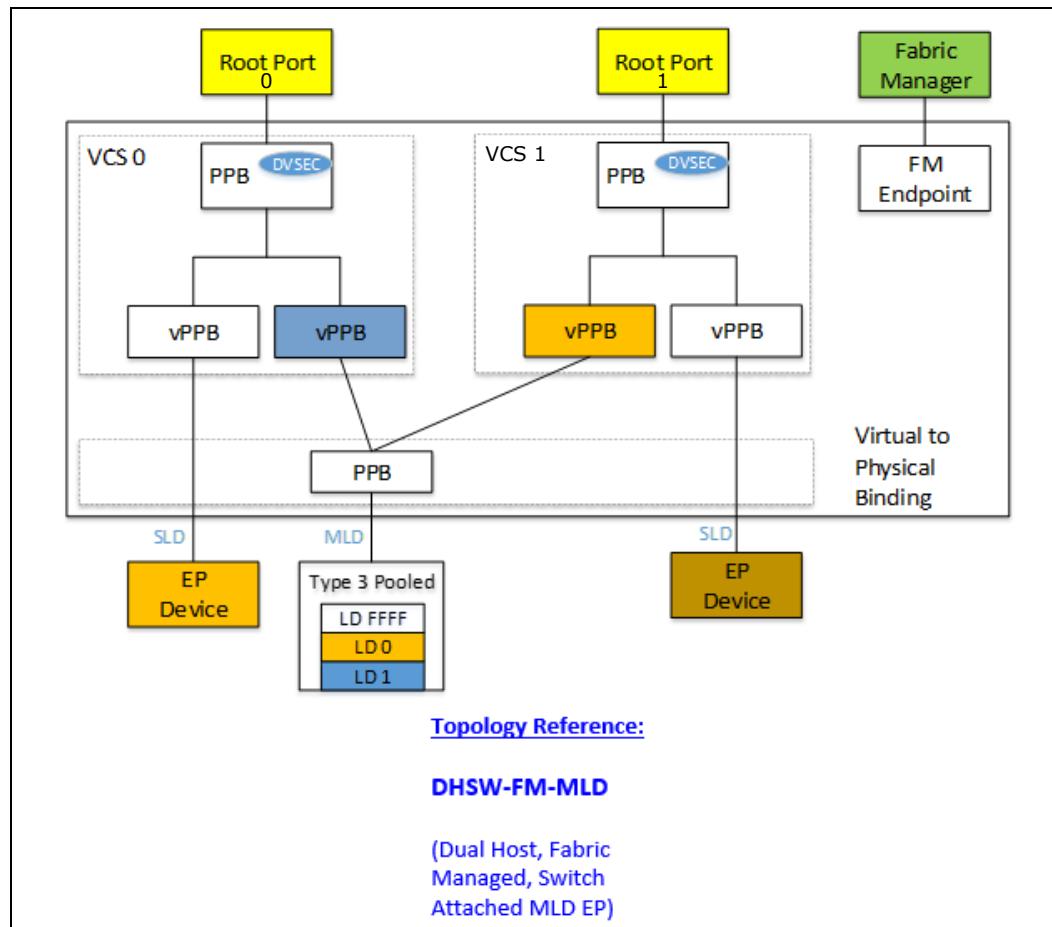
Figure 14-5. Example DHSW-FM Topology



14.2.1.5 Dual Host, Fabric Managed, Switch Attached MLD EP (DHSW-FM-MLD)

Figure 14-6 shows the topology for having dual hosts in a managed environment with multiple logical devices.

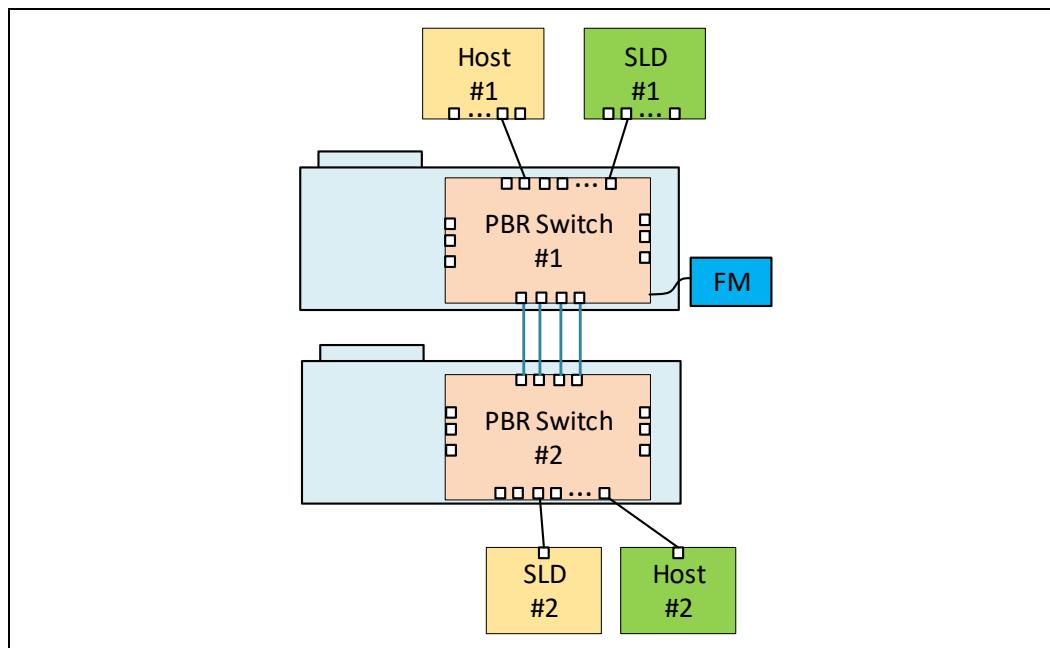
Figure 14-6. Example DHSW-FM-MLD Topology



14.2.1.6 Cascaded Switch Topologies

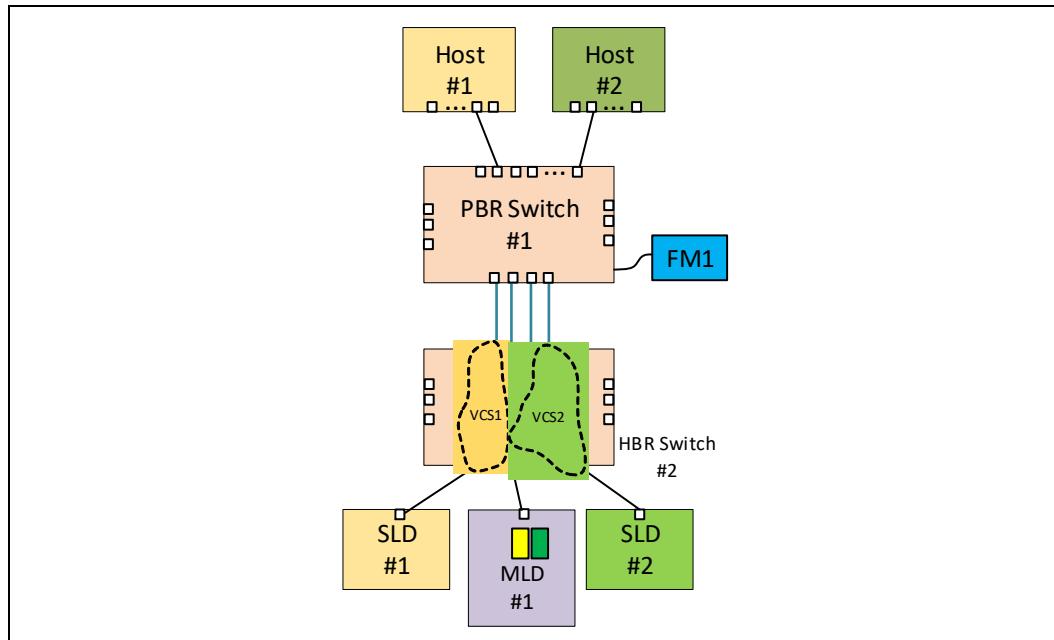
PBR switches enable cascaded and mesh topologies. Figure 14-7 shows a cascaded switch topology that is supported by PBR switches. PBR switches use PBR flits for Inter-switch links. A Fabric Manager is required to configure the fabric port routing. HBR switches may be attached to a PBR switch fabric.

Figure 14-7. Example Topology for Two PBR Switches



In a topology that has a single PBR switch and a single HBR switch (see [Figure 14-8](#)), the host devices are connected to the PBR switch and the HBR switch's Upstream Switch Ports (USPs) are connected to the PBR switch, to allow for multiple-host routing. The HBR switch configures a unique VCS for each host.

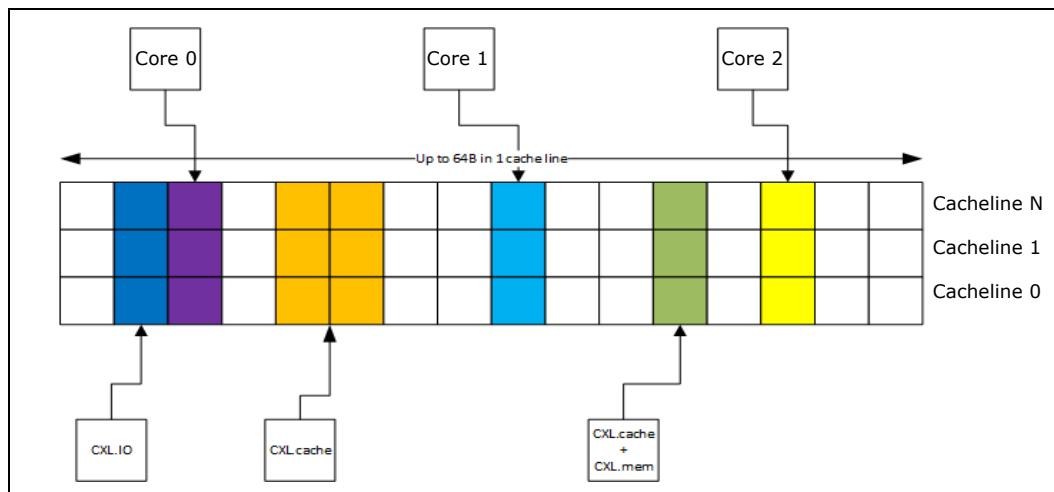
Figure 14-8. Example Topology for a PBR Switch and an HBR Switch



14.3 CXL.io and CXL.cache Application Layer/Transaction Layer Testing

14.3.1 General Testing Overview

Standard practices of testing coherency rely on “false sharing” of cachelines. Different agents in the system (e.g., cores, I/O, etc.) are assigned one or more fixed-byte locations within a shared set of cachelines. Each agent continuously executes an assigned Algorithm independently. Since multiple agents are sharing the same cacheline, stressful conflict scenarios can be exercised. [Figure 14-9](#) illustrates the concept of false sharing. This can be used for CXL.io (Load/Store semantics) or CXL.cache (caching semantics) or (CXL.cache + CXL.mem) devices (Type 2 devices).

Figure 14-9. Representation of False Sharing between Cores (on Host) and CXL Devices

This document outlines three Algorithms that enable stressing the system with false sharing tests. In addition, this document specifies the prerequisites that are needed to execute, verify, and debug runs for the Algorithms. All the Algorithms are applicable for CXL.io and CXL.cache (protocols that originate requests to the host). Devices are permitted to be self-checking. Self-checking devices must have a way to disable the checking Algorithm independent of executing the Algorithm. All devices must support the non-self-checking flow in the Algorithms outlined below. The algorithms presented for false sharing require coordination with the cache on the device (if present). Hence, it may add certain responsibility on the application layer if the cache resides there.

14.3.2 Algorithms

14.3.3 Algorithm 1a: Multiple Write Streaming

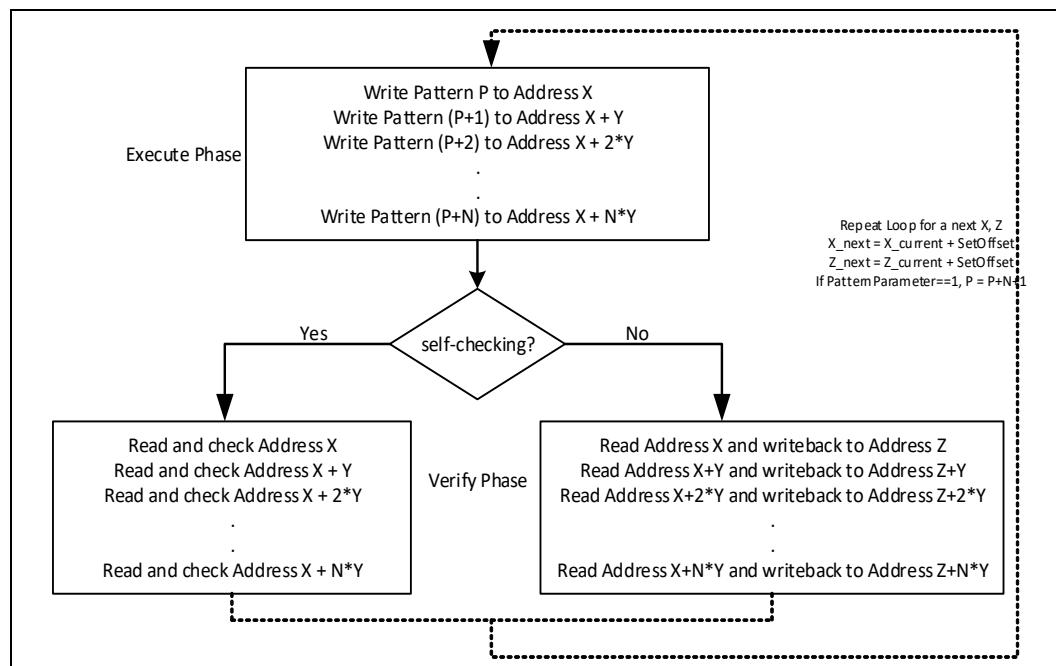
In this Algorithm, the device is setup to stream an incrementing pattern of writes to different sets of cachelines. Each set of cacheline is defined by a base address "X", and an increment address "Y". Increments are in multiples of 64B. The number of increments "N" dictates the size of the set beginning from base address X. The base address includes the byte offset within the cacheline. A pattern P (of variable length in bytes) determines the starting pattern to be written. Subsequent writes in the same set increment P. A device is required to provide a byte mask configuration capability that can be programmed to replicate pattern P in different parts of the cacheline. The programmed byte masks must be consistent with the base address.

Different sets of cachelines are defined by different base addresses (so a device may support a set like "X₁, X₂, X₃"). "X₁" is programmed by software in the base address register, X₂ is obtained by adding a fixed offset to X₁ (offset is programmed by software in a different register). X₃ is obtained by adding the same offset to X₂ and so on. Minimum support of 2 sets is required by the device. Figure 14-10 illustrates the flow of this Algorithm as implemented on the device. Address Z is the write back address where system software can poll to verify the expected pattern associated with this device, in cases where self-checking on the device is disabled. There is 1:1 correspondence between X and Z. It is the responsibility of the device to ensure that the writes in the execute phase are globally observable before beginning the verify phase. Depending on the write semantics used, this may imply additional fencing mechanism on the device to ensure the writes are globally visible before the verify phase can begin. When beginning a new set iteration, devices must also give an option

to use "P" again for the new set, or continue incrementing "P" for the next set. The select is programmed by software in "PatternParameter" field described in the register section.

Open: PatternParameter was in Table 14-41, which was removed in r3.0, v0.7. Please search the PDF for this term and determine how it and surrounding text should be revised. (Also appears in [Figure 14-10](#) and [Figure 14-11](#).)

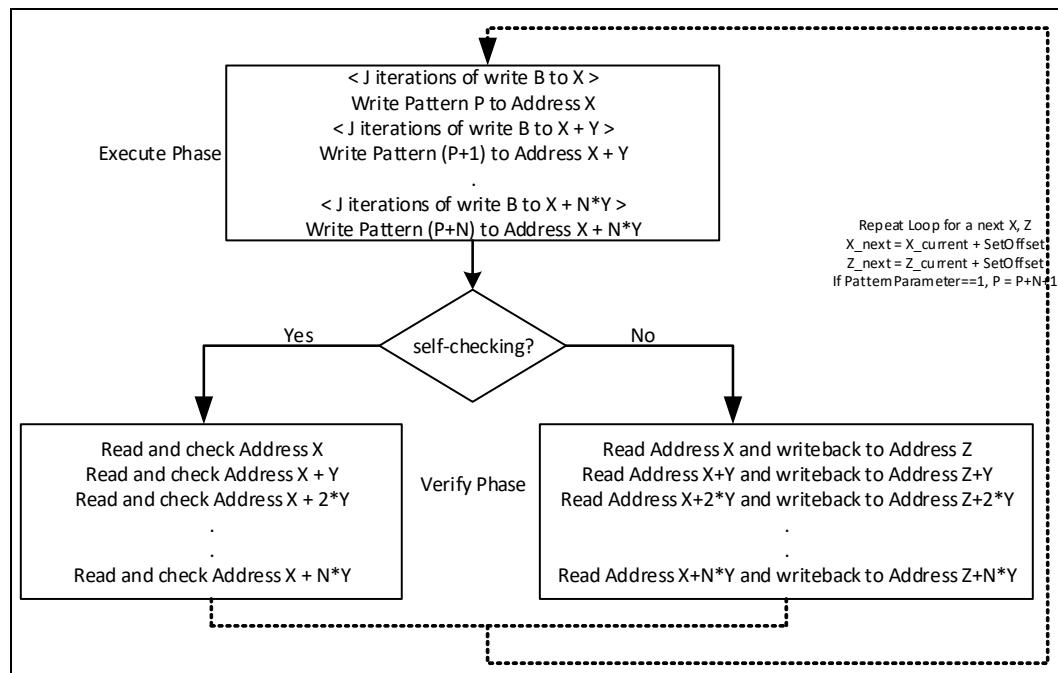
Figure 14-10. Flow Chart of Algorithm 1a



14.3.4

Algorithm 1b: Multiple Write Streaming with Bogus Writes

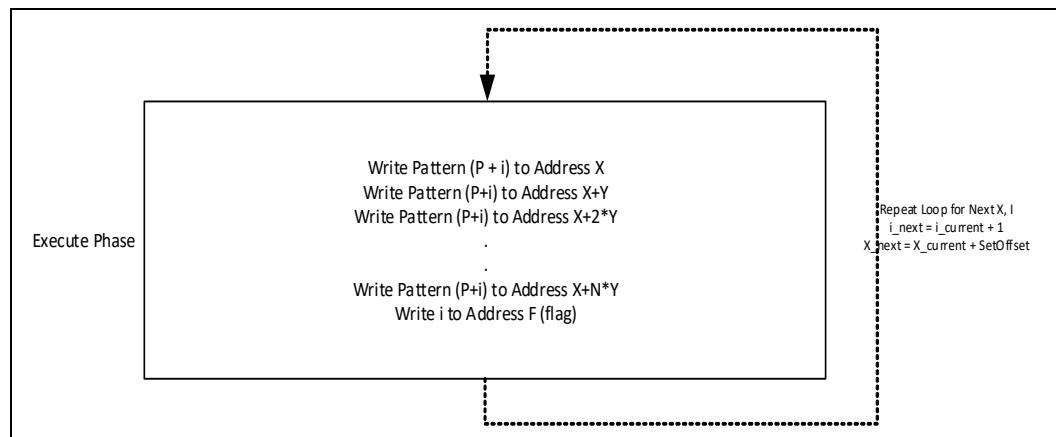
This Algorithm is a variation on Algorithm 1a, except that before writing the expected pattern to an address, the device does "J" iterations of writing a bogus pattern "B" to that address. [Figure 14-11](#) illustrates this Algorithm. In this case, if a pattern "B" is ever seen in the cacheline during the Verify phase, it is a Fail condition. The bogus writes help give a longer duration of conflicts in the system. It is the responsibility of the device to ensure that the writes in the execute phase are globally observable before beginning the verify phase. Depending on the write semantics used, this may imply additional fencing mechanism on the device to ensure the writes are globally visible before the verify phase can begin. When beginning a new set iteration, devices must also give an option to use "P" again for the new set, or continue incrementing "P" for the next set. The select is programmed by software in "PatternParameter" field described in the register section.

Figure 14-11. Flow Chart of Algorithm 1b

14.3.5

Algorithm 2: Producer Consumer Test

This Algorithm tests the scenario in which a Device is a producer and the CPU is a consumer. The Device simply executes a predetermined Algorithm of writing known patterns to a data location, followed by a flag update write. Threads on the CPU poll the flag, followed by reading the data patterns, followed by repolling the flag. This is a simple way of ensuring that the ordering rules of Producer-Consumer workloads are being followed through the stack. Device only participates in the execute phase of this Algorithm. [Figure 14-12](#) illustrates the device execute phase. The Verify phase is run on the CPU, software reads addresses in the following order [F, X, (X+Y)...(X+N*Y), F]. Knowing the value of the flag at two ends, the checker knows the range within which [X, (X+Y)...(X+N*Y)] have to be. For example, if P=0, the first read of F returns a value of 3 and the next read of F returns a value of 4, then checker knows that all intermediate values have to be either 3 or 4. Moreover, if the device is using strongly ordered semantics, then the checker should never see a transition of values from 3 to 4 (implying monotonically decreasing values for the non-flag addresses). If using CXL.cache protocol, device must ensure global observability of previous “data” writes before updating the flag. When using strongly ordered semantics, each update must be globally visible before the next write. Depending on the flow used for dirty evicts, this can be implementation specific. It is the responsibility of the device to ensure that the writes in the execute phase are globally observable before updating the flag “F”. The “PatternParameter” field is not relevant for this Algorithm. The Flag “F” should be written to Register 2: “WriteBackAddress1” in the Device Capabilities to support the Test Algorithms.

Figure 14-12. Execute Phase for Algorithm 2

14.3.6

Test Descriptions

Unless specified otherwise, the tests in this section are applicable to both 68B Flit mode and 256B Flit mode.

14.3.6.1

Application Layer/Transaction Layer Tests

The Transaction Layer Tests implicitly give coverage for Link Layer functionality. Specific error injection cases for the Link Layer are covered in [Section 14.12](#).

14.3.6.1.1

CXL.io Load/Store Test

For CXL.io, this test and associated capabilities are optional but strongly recommended. This test sets up the device to execute Algorithms 1a, 1b, and 2 in succession to stress the data path for CXL.io transactions. Configuration details are determined by the host platform testing the device. See [Section 14.16](#) for the configuration registers and device capabilities. Each run includes execute/verify phases as described in [Section 14.3.1](#).

Prerequisites:

- Hardware and configuration support for Algorithms 1a, 1b, and 2 described in [Section 14.3.1](#) and [Section 14.16](#)
- If the device supports self-checking, it must escalate a fatal system error if the Verify phase fails (see [Section 12.2](#) for specific error-escalation mechanisms)
- Device is permitted to log failing address, iteration number, and/or expected data vs. received data

Test Steps:

1. Host software will set up the device for Algorithm 1a: Multiple Write Streaming.
2. If the device supports self-checking, enable it.
3. Host software decides the test runtime and runs the test for that period of time. (The software details of this are host-platform specific, but will be compliant with the flows mentioned in [Section 14.3.1](#) and follow the configurations outlined in [Section 14.16](#).)
4. Set up the device for Algorithm 1b: Multiple Write Streaming with Bogus writes.
5. If the device supports self-checking, enable it.

6. Host software decides the test runtime and runs the test for that period of time.
7. Set up the device for Algorithm 2: Producer Consumer Test.
8. Host software decides the test runtime and runs the test for that period of time.

Pass Criteria:

- No data corruptions or system errors are reported

Fail Conditions:

- Data corruptions or system errors are reported

14.3.6.1.2 CXL.cache Coherency Test

This test sets up the device and the host to execute Algorithms 1a, 1b, and 2 in succession to stress the data path for CXL.cache transactions. This test should only be run if the device and the host support CXL.cache or CXL.cache + CXL.mem protocols. Configuration details are determined by the host platform testing the device. See [Section 14.16](#) for the configuration registers and device capabilities. Each run includes execute/verify phases as described in [Section 14.3.1](#).

Prerequisites:

- Device is CXL.cache capable
- Hardware and configuration support for Algorithms 1a, 1b, and 2 described in [Section 14.3.1](#) and [Section 14.16](#)
- If a Device supports self-checking, it must escalate a fatal system error if the Verify phase fails (see [Section 12.2](#) for specific error-escalation mechanisms)
- Device is permitted to log failing address, iteration number, and/or expected data vs. received data

Test Steps:

1. Host software will set up the device and the host for Algorithm 1a: Multiple Write Streaming. An equivalent version of the algorithm is setup to be executed by host software so as to enable false sharing of the cachelines.
2. If the device supports self-checking, enable it.
3. Host software decides the test runtime and runs the test for that period of time. (The software details of this are host-platform specific, but will be compliant with the flows mentioned in [Section 14.3.1](#) and follow the configurations outlined in [Section 14.16](#).)
4. Set up the device for Algorithm 1b: Multiple Write Streaming with Bogus writes.
5. If the device supports self-checking, enable it.
6. Host software decides the test runtime and runs the test for that period of time.
7. Set up the device for Algorithm 2: Producer Consumer Test.
8. Host software decides the test runtime and runs the test for that period of time.

Pass Criteria:

- No data corruptions or system errors are reported

Fail Conditions:

- Data corruptions or system errors are reported

14.3.6.1.3 CXL Test for Receiving GO-ERR

This test is applicable only for devices that support CXL.cache protocols. This test sets up the device to execute Algorithm 1a while mapping one of the sets of the address to a memory range that is not accessible by the device. Test system software and configuration details are determined by the host platform and are system specific.

Prerequisites:

- Device is CXL.cache capable
- Support for Algorithm 1a

Test Steps:

1. Configure device for Algorithm 1a, and set up one of the base addresses to be an address not accessible by the DUT.
2. Disable self-checking in the DUT.
3. Host software decides test runtime and runs test for that period of time.

Pass Criteria:

- No data corruptions or system errors are reported
- No fatal device errors on receiving GO-ERR
- Inaccessible memory range has not been modified by the device

Fail Conditions:

- Data corruptions or system errors reported
- Fatal device errors on receiving GO-ERR
- Inaccessible memory range modified by the device (host error)

14.3.6.1.4 CXL.mem Test

This test sets up the **host** and the device to execute Algorithms 1a, 1b, and 2 in succession to stress the data path for CXL.mem transactions. An equivalent version of the algorithm is setup to be executed by host software so as to enable false sharing of the cachelines. Test system software and configuration details are determined by the host platform and are system specific.

Prerequisites:

- Device is CXL.mem capable

Test Steps:

1. Map the device-attached memory to a test-memory range that is accessible by the host.
2. Run the equivalent of Algorithms 1a, 1b, and 2 on the host and the device targeting device-attached memory.

Pass Criteria:

- No data corruptions or system errors are reported

Fail Conditions:

- Data corruptions or system errors are reported

14.3.6.1.5 Egress Port Backpressure Test

This test applies to an MLD that supports FM API or an SLD that supports the Memory Device command set. This test sets up the device to execute Algorithms 1a, 1b, and 2 in succession to stress the data path for CXL.mem transactions. An equivalent version of the algorithm is setup to be executed by **host** software so as to enable false sharing of the cachelines. Test system software and configuration details are determined by the host platform and are system specific. NUMBER_OF_QOS_TEST_LOOPS, NUMBER_OF_CHECK_AVERAGE, and BackpressureSample Interval setting in the test steps below is decided upon by the testing platform/software.

Prerequisites:

- Device is CXL.mem capable

Test Steps:

For an MLD:

1. Through the FM API, check if Egress Port Congestion Supported is set by issuing a Get LD Info command.
2. If Egress Port Congestion Supported is enabled:
Repeat for NUMBER_OF_QOS_TEST_LOOPS:
 - a. Set the BackpressureSample Interval setting to a value between 1 -31 through the Set QoS Control command.
 - b. Set the Egress Port Congestion Enable bit through the Set QoS Control command.
 - c. Check that the Egress Port Congestion Enable bit was set successfully in the Get QoS Control Response.
 - d. Run the equivalent of Algorithms 1a, 1b, and 2 in succession on the host and the device targeting device-attached memory.
 - e. While Algorithms 1a, 1b, and 2 are running: Check the reported Backpressure Average Percentage through the Get QoS Status command and response. It should report values within the valid range which is 0 – 100. Repeat this step NUMBER_OF_CHECK_AVERAGE times at a certain interval.

For an SLD:

1. Check if Egress Port Congestion Supported is set by issuing an Identify Memory Device, and checking the corresponding Identify Memory Device Output Payload.
2. If Egress Port Congestion Supported is enabled, repeat for NUMBER_OF_QOS_TEST_LOOPS:
 - a. Set the BackpressureSample Interval setting to a value between 1 - 31 through the Set SLD QoS Control Request command.
 - b. Set the Egress Port Congestion Enable bit through the Set SLD QoS Control Request.
 - c. Check that the Egress Port Congestion Enable bit was set successfully in the Get SLD QoS Control Response.
 - d. Check the reported Backpressure Average Percentage through the Get QoS Status command and response.
 - e. Run the equivalent of Algorithms 1a, 1b, and 2 in succession on the host and the device targeting device-attached memory.
 - f. While Algorithms 1a, 1b, and 2 are running: Check the reported Backpressure Average Percentage through the Get SLD QoS Status command and response.

It should report values within the valid range which is 0 – 100. Repeat this step NUMBER_OF_CHECK_AVERAGE times at a certain interval.

Pass Criteria:

- Egress Port Congestion Enable is set after enabling it
- Backpressure Average Percentage reports valid values within 0-100.
- No data corruptions or system errors are reported while executing Algorithms 1a, 1b, and 2

Fail Conditions:

- Egress Port Congestion Enable is not set after enabling it
- Backpressure Average Percentage reports any value outside the valid 0-100 range
- Data corruptions or system errors reported while executing Algorithms 1a, 1b, and 2

14.3.6.1.6 Temporary Throughput Reduction Test

This test applies to an MLD that supports FM API or an SLD that supports the Memory Device Command set. This test sets up the device to execute Algorithms 1a, 1b, and 2 in succession to stress the data path for CXL.mem transactions. For Type 3 (MLD or SLD), it is the responsibility of the host to take care of running the algorithms as appropriate. An equivalent version of the algorithm is setup to be executed by **Host** software so as to enable false sharing of the cachelines. Test system software and configuration details are determined by the host platform and are system specific. NUMBER_OF_QOS_TEST_LOOPS in the test steps is decided upon by the testing platform/software.

Prerequisites:

- Device is CXL.mem capable

Test Steps:

For an MLD:

1. Through the FM API, check if Temporary Throughput Reduction Supported is set by issuing a Get LD Info command.
2. If Temporary Throughput Reduction Supported is enabled, repeat for NUMBER_OF_QOS_TEST_LOOPS:
 - a. Set the Temporary Throughput Reduction Enable bit by issuing the Set QoS Control command.
 - b. Check that the Temporary Throughput Reduction Enable bit was set successfully in the Get QoS Control Response.
 - c. Run the equivalent of Algorithms 1a, 1b, and 2 in succession on the host and the device targeting device-attached memory.

For an SLD:

1. Through the Memory Device Command set, check if Temporary Throughput Reduction Supported is set by issuing an Identify Memory Device, and checking corresponding Identify Memory Device Output Payload.
2. If Temporary Throughput Reduction Supported is enabled, repeat for NUMBER_OF_QOS_TEST_LOOPS:
 - a. Set the Temporary Throughput Reduction Enable bit through the Set SLD QoS Control Request.

- b. Check that the Temporary Throughput Reduction Enable bit was set successfully in the Get SLD QoS Control Response.
- c. Run the equivalent of Algorithms 1a, 1b, and 2 in succession on the host and the device targeting device-attached memory.

Pass Criteria:

- Temporary Throughput Reduction Enable is set after enabling it
- No data corruptions or system errors are reported while executing Algorithms 1a, 1b, and 2

Fail Conditions:

- Temporary Throughput Reduction Enable is not set after enabling it
- Data corruptions or system errors reported while executing Algorithms 1a, 1b, and 2

14.4 Link Layer Testing

14.4.1 RSVD Field Testing CXL.cachemem

Test Equipment:

- Exerciser

Prerequisites:

- Applicable for 68B and 256B Flit modes
- Device is CXL.cachemem capable
- CXL link is up

14.4.1.1 Device Test

Test Steps:

1. Send from host Link Layer Control.INIT.Param with all RSVD fields set to 1.
2. Wait for Control-INIT.Param from the device.
3. Wait for the Link to reach L0 state and the device is in a configured state.

Pass Criteria:

- CXL Link Layer Control and Status Register INIT_State is 11b
- Link Layer initialization is successful and Reserved fields are ignored

Fail Conditions:

- Pass criteria is not met

14.4.1.2 Host Test

Test Steps:

1. Send from device Link Layer Control.INIT.Param with all RSVD fields set to 1.
2. Wait for Link to reach L0 state.

14.4.2 CRC Error Injection RETRY_PHY_REINIT

Test Equipment:

- Protocol Analyzer
- Protocol Exerciser

Prerequisites:

- Applicable for 68B Flit mode only
- CXL Host must support Algorithm 1a
- CXL Host must support Link Layer Error Injection capabilities for CXL.cache

Test Steps:

- Setup is the same as Test [14.3.6.1.2](#).
- While a test is running, software will insert the following error injection. The Protocol Exerciser will retry the flit for at least MAX_NUM_RETRY times upon detecting a CRC error.

Table 14-1. CRC Error Injection RETRY_PHY_REINIT: Cache CRC Injection Request

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	7, CRC Injection
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	2
Dh	1	Num Bits Flipped	1
Eh	1	Num Flits Injected	1

Pass Criteria:

- Same as Test [14.3.6.1.2](#)
- Monitor and verify that CRC errors are injected (using the Protocol Analyzer), and that Retries are triggered as a result
- Five RETRY.Frame Flits are sent before RETRY.Req and RETRY.Ack (protocol analyzer)
- Check that link enters RETRY_PHY_REINIT
- Means value of NUM_Phys_Reinit_Received: Num_Phys_Reinit value reflected in the last RETRY.Req message received in CXL Link Layer Capability register is greater than 1

14.4.3

Fail Conditions:

- Same as Test [14.3.6.1.2](#)
- Link does not reach RETRY_PHY_REINIT

CRC Error Injection RETRY_ABORT**Test Equipment:**

- Protocol Analyzer
- Protocol Exerciser

Prerequisites:

- Applicable for 68B Flit mode only
- CXL device must support Algorithm 1a
- CXL device must support Link Layer Error Injection capabilities for CXL.cache

Test Steps:

1. Set up is the same as Test [14.3.6.1.2](#).
2. While a test is running, software will insert the following error injection. The Protocol Exerciser will retry the flit for at least (**MAX_NUM_RETRY** x **MAX_NUM_PHY_REINIT**) times upon detecting a CRC error:

Table 14-2. CRC Error Injection RETRY_ABORT: Cache CRC Injection Request

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	7, CRC Injection
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	2
Dh	1	Num Bits Flipped	1
Eh	1	Num Flits Injected	1

Pass Criteria:

- Same as Test [14.3.6.1.2](#)
- Monitor and verify that CRC errors are injected (using the Protocol Analyzer), and that Retries are triggered as a result
- Five RETRY.Frame Flits are sent before RETRY.Req and RETRY.Ack (protocol analyzer)
- Link retrains for MAX_NUM_PHY_REINIT number of times and fails to recover

Fail Conditions:

- Same as Test [14.3.6.1.2](#)
- Link does not reach RETRY_PHY_REINIT
- Link does not reach RETRY_ABORT

14.5 ARB/MUX

14.5.1 Reset to Active Transition

Test Equipment:

- Protocol Analyzer

Prerequisites:

- Applicable for 68B Flit mode, 256B Flit mode, and Latency-Optimized 256B Flit mode
- CXL link is not assumed to be up
- Device drivers are not assumed to have been loaded

Test Steps:

- With the link in Reset state, Link layer sends a Request to enter Active.
- ARB/MUX waits to receive indication of Active from Physical Layer.

Pass Criteria:

- ALMP Status sync exchange completes before ALMP Request{Active} sent by Local ARB/MUX (if applicable)
- Local ARB/MUX sends ALMP Request{Active} to the remote ARB/MUX
- Validate the first ALMP on the initial bring up is from the Downstream Port to the Upstream Port
- Local ARB/MUX waits for ALMP Status{Active} and ALMP Request{Active} from remote ARB/MUX
- Local ARB/MUX sends ALMP Status{Active} in response to Request
- Link transitions to Active after the ALMP handshake completes
- Link successfully enters Active state with no errors

Fail Conditions:

- Link hangs and does not enter Active state
- Any error occurs before transition to Active state

14.5.2 ARB/MUX Multiplexing

Test Equipment:

- Protocol Analyzer (used to ensure that traffic is sent simultaneously on both CXL.io and CXL.cachemem)

Prerequisites:

- Applicable for 68B Flit mode, 256B Flit mode, and Latency-Optimized 256B Flit mode
- Device is CXL.cache and/or CXL.mem capable
- Host-generated traffic or Device-generated traffic
- Support for Algorithm 1a, 1b, or 2

Test Steps:

1. Bring the link up into CXL mode with CXL.io and CXL.cache and/or CXL.mem enabled.
2. Ensure the arbitration weight is a nonzero value for both interfaces.
3. Send continuous traffic on both CXL.io and CXL.cache and/or CXL.mem using Algorithm 1a, 1b, or 2.
4. Allow time for traffic transmission while snooping the bus.

Pass Criteria:

- Data from both CXL.io and CXL.cache and/or CXL.mem are sent across the link by the ARB/MUX

Fail Conditions:

- Data on the link is only CXL.io
- Data on the link is only CXL.cache or CXL.mem (CXL.cache and CXL.mem share a single Protocol ID; see [Table 6-2](#))

Test Steps (256B Flit Mode):

1. Upstream Port sends PM state Request ALMP.
2. Wait for an ALMP Request for entry to a PM state.
3. Downstream Port rejects the request by responding Active.PMNAK Status ALMP.
4. On receiving Active.PMNAK Status ALMP, the Upstream Port must transition the corresponding vLSM to Active.PMNAK state.
5. After Active.PMNAK is observed, the Link Layer must request Active to the ARB/MUX and then wait for the vLSM to transition to Active before transmitting flits.

Pass Criteria:

- Upstream Port must continue to receive and process flits while the vLSM state is Active or Active.PMNAK
- Upstream Port must transition back to Active state
- For Upstream Ports, after the Link Layer has requested PM entry, the Link Layer must not change this request until it observes the vLSM status change to either the requested state or to Active.PMNAK or to one of the non-virtual states (LinkError, LinkReset, LinkDisable, or Reset)

Fail Conditions:

- Any system error

14.5.3 Active to L1.x Transition (If Applicable)

Test Equipment:

- Protocol Analyzer

Prerequisites:

- Applicable for 68B Flit mode, 256B Flit mode, and Latency-Optimized 256B Flit mode
- Support for ASPM L1

Test Steps:

1. Force the remote and local link layer to send a request to the ARB/MUX for L1.x state.
2. This test should be run separately for each Link Layer independently (to test one Link Layer's L1 entry while the other Link Layer is in ACTIVE), as well as both Link Layers concurrently requesting L1 entry.

Pass Criteria:

- Upstream Port ARB/MUX sends ALMP Request{L1.x}
- Downstream Port ARB/MUX sends ALMP Status{L1.x} in response
- L1.x is entered after the local ARB/MUX receives ALMP Status
- State transition doesn't occur until ALMP handshake is complete
- LogPHY enters L1 ONLY after both Link Layers enter L1 (applies to CXL mode only)

Fail Conditions:

- Error in ALMP handshake
- Protocol layer packets sent after ALMP L1.x handshake is complete (requires Protocol Analyzer)
- State transition occurs before ALMP handshake completed

14.5.4 L1.x State Resolution (If Applicable)

Test Equipment:

- Protocol Analyzer

Prerequisites:

- Applicable for 68B Flit mode, 256B Flit mode, and Latency-Optimized 256B Flit mode
- Support for ASPM L1

Test Steps:

1. Force the remote and local link layer to send a request to the ARB/MUX for **different** L1.x states.

Pass Criteria:

- Upstream Port ARB/MUX sends ALMP Request{L1.x} according to what the link layer requested
- Upstream Port ARB/MUX sends ALMP Status{L1.y} response
- The state in the Status ALMP is the more-shallow L1.y state
- L1.y is entered after the local ARB/MUX receives ALMP Status
- State transition doesn't occur until the ALMP handshake is complete
- LogPHY enters L1 ONLY after both protocols enter L1 (applies to CXL mode only)

Fail Conditions:

- Error in ALMP handshake

- Protocol layer packets sent after ALMP L1.x handshake is complete (requires Protocol Analyzer)
- State transition occurs before ALMP handshake completed

14.5.5 Active to L2 Transition

Test Equipment:

- Protocol Analyzer

Prerequisites:

- Applicable for 68B Flit mode, 256B Flit mode, and Latency-Optimized 256B Flit mode

Test Steps:

1. Force the remote and local link layer to send a request to the ARB/MUX for L2 state.

Pass Criteria:

- Upstream Port ARB/MUX sends ALMP Request{L2} to the remote vLSM
- Upstream Port ARB/MUX waits for ALMP Status{L2} from the remote vLSM
- L2 is entered after the local ARB/MUX receives ALMP Status
- If there are multiple link layers, repeat the above steps for all link layers
- Physical link enters L2
- vLSM and physical link state transitions don't occur until ALMP handshake is complete

Fail Conditions:

- Error in ALMP handshake
- Protocol layer packets sent after ALMP L2 handshake is complete (requires Protocol Analyzer)
- State transition occurs before ALMP handshake completed

14.5.6 L1 to Active Transition (If Applicable)

Test Equipment:

- Protocol Analyzer Required

Prerequisites:

- Applicable for 68B Flit mode, 256B Flit mode, and Latency-Optimized 256B Flit mode
- Support for ASPM L1

Test Steps:

1. Bring the link into L1 state.
2. Force the link layer to send a request to the ARB/MUX to exit L1.

Pass Criteria:

- Local ARB/MUX sends L1 exit notification to the Physical Layer
- Link exits L1
- Link enters L0 correctly
- 68B Flit mode
 - Status synchronization handshake completes successfully
 - Active ALMP exchange to exit vLSM L1 and transition to Active successfully
- 256B Flit mode and Latency-Optimized 256B Flit mode
 - Active ALMP request and receive Active Status ALMP to exit vLSM L1 and transition to Active

Fail Conditions:

- Link transition to L0 has not occurred
- 68B Flit mode
 - No status exchange happened or
 - Active ALMP exchange has not occurred
- 256B Flit mode
 - Active ALMP exchange has not occurred

14.5.7 Reset Entry

Prerequisites:

- Applicable for 68B Flit mode

Test Steps:

1. Initiate warm reset flow.

Pass Criteria:

- Link sees hot reset and transitions to Detect state

Fail Conditions:

- Link does not enter Detect

14.5.8 Entry into L0 Synchronization

Test Equipment:

- Protocol Analyzer

Prerequisites:

- Applicable for 68B Flit mode, 256B Flit mode, and Latency-Optimized 256B Flit mode

Test Steps:

1. Place the link into Retrain state.

2. After exit from Retrain, check Status ALMPs to synchronize interfaces across the link.

Pass Criteria:

- State contained in the Status ALMP is the same state the link was in before entry to Retrain

Fail Conditions:

- No Status ALMPs are sent after exit from Retrain state
- State in Status ALMPs different from the state that the link was in before the link went into Retrain
- Other communication occurred on the link after Retrain before the Status ALMP handshake for synchronization completed

14.5.9 ARB/MUX Tests Requiring Injection Capabilities

The tests in this section are optional but strongly recommended. The test configuration control registers for the tests in this section are implementation specific.

14.5.9.1 ARB/MUX Bypass

Test Equipment:

- Protocol Analyzer

Prerequisites:

- Applicable for 68B Flit mode and 256B Flit mode
- Device capability to force a request ALMP for any state

Test Steps:

1. Place the Link into PCIe only mode.
2. Trigger entry to Retrain State.
3. Snoop the bus and check for ALMPs.

Pass Criteria:

- No ALMPs are generated by the ARB/MUX

Fail Conditions:

- ALMP is seen on the bus when checked

14.5.9.2 PM State Request Rejection

Test Equipment:

- Protocol Analyzer

Prerequisites:

- Applicable for 68B Flit mode, 256B Flit mode, and Latency-Optimized 256B Flit mode
- Host capability to place the host into a state where it will reject any PM request ALMP

14.5.9.3

Unexpected Status ALMP

Prerequisites:

- Applicable for 68B Flit mode only
- Device capability to force the ARB/MUX to send a Status ALMP at any time

Test Steps:

1. While the link is in Active state, force the ARB/MUX to send a Status ALMP without first receiving a Request ALMP.

Pass Criteria:

- Link enters Retrain state without any errors being reported

Fail Conditions:

- No error on the link and normal operation continues
- System errors are observed

14.5.9.4

ALMP Error

Prerequisites:

- Applicable for 68B Flit mode only
- Device capability that allows the device to inject errors into a flit

Test Steps:

1. Inject a single bit error into the lower 16 bytes of a 528-bit flit.
2. Send data across the link.
3. ARB/MUX detects error and enters Retrain.
4. Repeat Steps 1-3 with a double-bit error.

Pass Criteria:

- Link enters Retrain

14.5.9.5 Recovery Re-entry

Fail Conditions:

- No errors are detected

Prerequisites:

- Applicable for 68B Flit mode only
- Device capability that allows the device to ignore ALMP State Requests

Test Steps:

1. Place the link into Active state.
2. Request link to enter Retrain State.
3. Prevent the Local ARB/MUX from entering Retrain.
4. Remote ARB/MUX enters Retrain state.
5. Remote ARB/MUX exits Retrain state and sends ALMP Status{Active} to synchronize.
6. Local ARB/MUX receives Status ALMP for synchronization but does not send.
7. Local ARB/MUX triggers re-entry to Retrain.

Pass Criteria:

- Link successfully enters Retrain on re-entry attempt

Fail Conditions:

- Link continues operation without proper synchronization

14.5.10 L0p Feature

14.5.10.1 Positive ACK for L0p

Test Equipment:

- Protocol Analyzer

Prerequisites:

- Link negotiation in 256B Flit mode is supported
- L0p feature is supported

Test Steps:

1. Get current Link Width.
2. If Link Width = 1 and Link capability > 1:
 - a. Request L0p scale up to maximum supported width.
 - b. Successful Link scale up (assuming ACK).
 - c. Continue ALMP and traffic during L0p phases as normal.

Pass Criteria:

- No packet errors

- Link Width scale up to value indicated is successful; else Link Width > 1
- Request LOp scale down to 1

Fail Conditions:

- Pass criteria is not met

14.5.10.2 Force NAK for LOp Request**Test Equipment:**

- Protocol Analyzer

Prerequisites:

- Link Negotiation in 256B Flit mode is supported
- LOp feature is supported

Test Steps:

1. For LOp request, force a NAK.

Pass Criteria:

- No change with Negotiated Link Width register

Fail Conditions:

- Up/down scaling
- Data error transfers

14.6 Physical Layer**14.6.1 Tests Applicable to 68B Flit Mode****Prerequisites:**

- Applicable only when the link is expected to train to 68B Flit mode (see [Table 6-12](#))

14.6.1.1 Protocol ID Checks**Test Equipment:**

- Protocol Analyzer

Test Steps:

1. Bring the link up to Active state.
2. Send one or more flits from the CXL.io interface, and then check for the correct Protocol ID.
3. If applicable, send one or more flits from the CXL.cache and/or CXL.mem interface, and then check for the correct Protocol ID.
4. Send one or more flits from the ARB/MUX, and then check for the correct Protocol ID.

Pass Criteria:

- All Protocol IDs are correct

Fail Conditions:

- Errors occur during test
- No communication

14.6.1.2 NULL Flit**Test Equipment:**

- Protocol Analyzer

Test Steps:

1. Bring the link up to Active state.
2. Delay flits from the Link Layer.
3. Check for NULL flits from the Physical Layer.
4. Check that NULL flits have correct Protocol ID.

Pass Criteria:

- NULL flits seen on the bus when Link Layer delayed
- NULL flits have correct Protocol ID
- NULL flits contain all zero data

Fail Conditions:

- No NULL flits are sent from the Physical Layer
- Errors are logged during tests in the CXL DVSEC Port Status register

14.6.1.3 EDS Token**Test Equipment:**

- Protocol Analyzer

Test Steps:

1. Bring the link up to Active state.
2. Send a flit with an implied EDS token.

Pass Criteria:

- A flit with an implied EDS token is the last flit in the data block
- Next Block after a flit with an implied EDS token is an ordered set (OS)
- OS block follows the data block that contains a flit with the implied EDS token

Fail Conditions:

- Errors logged during test

14.6.1.4 Correctable Protocol ID Error

This test is optional but strongly recommended.

Test Equipment:

- Protocol Analyzer

Test Steps:

1. Bring the link up to Active state.
2. Create a correctable Protocol ID framing error by injecting an error into one 8-bit encoding group of the Protocol ID such that the new 8b encoding is invalid.
3. Check that an error is logged and normal processing continues.

Pass Criteria:

- Error correctly logged in DVSEC Flex Bus Port Status register
- Correct 8-bit encoding group used for normal operation

Fail Conditions:

- No errors are logged
- Flit with error dropped
- Error causes retrain
- Normal operation does not resume after error

14.6.1.5 Uncorrectable Protocol ID Error

This test is optional but strongly recommended.

Test Equipment:

- Protocol Analyzer

Test Steps:

1. Bring the link up to Active state.
2. Create an uncorrectable framing error by injecting an error into both 8-bit encoding groups of the Protocol ID such that both 8b encodings are invalid.
3. Check that an error is logged and that the flit is dropped.
4. Link enters Retrain state.

Pass Criteria:

- Error is correctly logged in the DVSEC Flex Bus Port Status register
- Link enters Retrain state

Fail Conditions:

- No errors are logged in the DVSEC Flex Bus Port Status register

14.6.1.6 Unexpected Protocol ID

This test is optional but strongly recommended.

Test Equipment:

- Protocol Analyzer

Test Steps:

1. Bring the link up to Active state.
2. Send a flit with an unexpected Protocol ID.
3. Check that an error is logged and that the flit is dropped.
4. Link enters Retrain state.

Pass Criteria:

- Error is correctly logged in the DVSEC Flex Bus Port Status register
- Link enters Retrain state

Fail Conditions:

- No Errors are logged in the DVSEC Flex Bus Port Status register

14.6.1.7 Recovery.Idle/Config.Idle Transition to L0**Test Equipment:**

- Protocol Analyzer

Test Steps:

1. Bring the link up in CXL mode to Recovery.Idle or Config.Idle state.
2. Wait for the NULL flit to be received by the DUT.
3. Check that the DUT sends NULL flits after receiving NULL flits.

Pass Criteria:

- LTSSM transitions to L0 after 8 NULL flits are sent and at least 4 NULL flits are received

Fail Conditions:

- LTSSM remains in IDLE

14.6.1.8 Uncorrectable Mismatched Protocol ID Error

This test is optional but strongly recommended.

Prerequisites:

- Protocol ID error perception in the device Log PHY (device can forcibly react as though there is an error even if the Protocol ID is correct)

Test Steps:

1. Bring the link up to Active state.
2. Create an uncorrectable Protocol ID framing error by injecting a flit such that both 8-bit encoding groups of the Protocol ID are valid but do not match.
3. Check that an error is logged and that the flit is dropped.
4. Link enters Retrain state.

14.6.2

Drift Buffer (If Applicable)

Prerequisites:

- Drift buffer is supported

Test Steps:

1. Enable the Drift buffer.

Pass Criteria:

- Drift buffer is logged in the Flex Bus DVSEC

Fail Conditions:

- No log in the Flex Bus DVSEC

14.6.3

SKP OS Scheduling/Alternation (If Applicable)

Test Equipment:

- Protocol Analyzer

Prerequisites:

- Applicable only when the link trains to 32 GT/s or lower
- Support Sync Header Bypass

Test Steps:

1. Bring the link up in CXL mode with Sync Header Bypass enabled.
2. Check for SKP OS.

Pass Criteria:

- Physical Layer schedules SKP OS every 340 data blocks
- Control SKP OS and standard SKP OS alternate at 16 GT/s or higher
- Standard SKP OS is used only at 8 GT/s

Fail Conditions:

- No SKP OS is observed
- SKP OS is observed at an interval other than 340 data blocks

14.6.4 SKP OS Exiting the Data Stream (If Applicable)

Test Equipment:

- Protocol Analyzer

Prerequisites:

- Applicable only when the link trains to 32 GT/s or lower
- Support Sync Header Bypass

Test Steps:

- Bring the link up in CXL mode with Sync Header Bypass enabled.
- Exit Active state.

Pass Criteria:

- Physical Layer replaces SKP OS with EIOS or EIEOS

Fail Conditions:

- SKP OS is not replaced by the Physical Layer

14.6.5 Link Initialization Resolution

See [Section 14.2.1](#) for the list of configurations that are used by this test.

Test Equipment:

- Protocol Analyzer

Test Steps:

- For the DUT, set up the system as described in the **Configurations to Test** column of [Table 14-3](#).
- In each of the configurations marked “Yes” in the **Retimer Check Required (If Present)** column, if there are CXL-aware retimer(s) present in the path, ensure that bit 12 and bit 14 (in Symbols 12-14) of the Modified TS1/TS2 Ordered Set are set to 1 (as applicable). In addition, ensure that Sync Header Bypass capable/enable is set.
- Negotiate for CXL during PCIe alternate protocol negotiation.

Table 14-3. Link Initialization Resolution Table (Sheet 1 of 2)

DUT	Upstream Component	Downstream Component	Retimer Check Required (If Present)	Configurations to Test	Verify
CXL Switch	Host - CXL VH capable	DUT	Yes	SHSW	Link initializes to L0 in CXL VH mode
	Host - RCH	DUT		SHSW	Link doesn't initialize to L0 in CXL mode
	DUT	Endpoint - CXL VH capable	Yes	SHSW	Link initializes to L0 in CXL VH mode
	DUT	Endpoint - eRCD	Yes	SHSW	Link initializes to CXL VH mode

Table 14-3. Link Initialization Resolution Table (Sheet 2 of 2)

DUT	Upstream Component	Downstream Component	Retimer Check Required (If Present)	Configurations to Test	Verify
Host - CXL VH capable	DUT	Switch - CXL VH capable		SHSW	Link initializes to L0 in CXL VH mode
	DUT	Endpoint - CXL VH capable	Yes	SHDA	Link initializes to L0 in CXL VH mode
	DUT	Endpoint - eRCD	Yes	SHDA	Link initializes to L0 in RCD mode
Endpoint - CXL VH capable	Host - CXL VH capable	DUT		SHDA	Link initializes to L0 in CXL VH mode
	CXL Switch	DUT		SHSW	Link initializes to L0 in CXL VH mode
	Host - RCH	DUT	Yes	SHDA	Link initializes to L0 in RCD mode

Pass Criteria:

- For a given type of DUT (column 1), all Verify Conditions in [Table 14-3](#) are met
- For cases where it is expected that the link initializes to CXL VH mode, IO_Enabled is set and either one or both of Cache_Enabled and Mem_Enabled are set in the DVSEC Flex Bus Port Status register

Fail Conditions:

- For a given type of DUT (column 1), any of the Verify Conditions in [Table 14-3](#) are not met
- For cases where it is expected that the link initializes to CXL VH mode, neither Cache_Enabled nor Mem_Enabled are set in the DVSEC Flex Bus Port Status register

14.6.6 Hot Add Link Initialization Resolution

See [Section 14.2.1](#) for the list of configurations that are used by this test.

Test Steps:

- Set up the system as described in the **Configurations to Test** column of [Table 14-4](#).
- Attempt to Hot-Add the DUT in CXL mode in each configuration.

Table 14-4. Hot Add Link Initialization Resolution Table (Sheet 1 of 2)

DUT	Upstream Component	Downstream Component	Configurations to Test	Verify
CXL Switch	Host - CXL VH capable	DUT	SHSW	Hot-Add - Link initializes to L0 in CXL VH mode
	DUT	Endpoint - CXL VH capable	SHSW	Hot-Add - Link initializes to L0 in CXL VH mode
	DUT	Endpoint - eRCD	SHSW	Link doesn't initialize to L0 in CXL mode for Hot-Add

Table 14-4. Hot Add Link Initialization Resolution Table (Sheet 2 of 2)

DUT	Upstream Component	Downstream Component	Configurations to Test	Verify
Host	DUT	CXL Switch	SHSW	Hot-Add - Link initializes to L0 in CXL VH mode
	DUT	Endpoint - CXL VH capable	SHDA	Hot-Add - Link initializes to L0 in CXL VH mode
	DUT	Endpoint - eRCD	SHDA	Link doesn't initialize to L0 in CXL mode for Hot-Add
Endpoint - CXL VH capable	Host - CXL VH capable	DUT	SHDA	Hot-Add - Link initializes to L0 in CXL VH mode
	CXL Switch	DUT	SHSW	Hot-Add - Link initializes to L0 in CXL VH mode

Pass Criteria:

- For a given type of DUT (column 1), all Verify Conditions in [Table 14-4](#) are met
- For cases where it is expected that the link initializes to CXL VH mode, IO_Enabled is set and either one or both of Cache_Enabled and Mem_Enabled are set in the DVSEC Flex Bus Port Status register

Fail Conditions:

- For a given type of DUT (column 1), any of the Verify Conditions in [Table 14-4](#) are not met
- For cases where it is expected that the link initializes to CXL VH mode, neither Cache_Enabled nor Mem_Enabled are set in the DVSEC Flex Bus Port Status register

14.6.7 Link Speed Advertisement

Test Equipment:

- Protocol Analyzer

Prerequisites:

- Applicable only for devices that support 8 GT/s or 16 GT/s in addition to also supporting 32 GT/s

Test Steps:

- Wait for initial link training at 2.5 GT/s.
- Check speed advertisement before alternate protocol negotiations have completed (i.e., LTSSM enters Configuration.Idle with LinkUp=0 at 2.5 GT/s).

Pass Criteria:

- Advertised CXL speed is 32 GT/s until Configuration.Complete state is exited

Fail Conditions:

- Speed advertisement is not 32 GT/s

14.6.8 Link Speed Degradation - CXL Mode

Test Steps:

1. Train the CXL link up to the highest speed possible (at least 16 GT/s).
2. Degrade the Link Down to a lower CXL mode speed.

Pass Criteria:

- Link degrades to slower speed without going through mode negotiation

Fail Conditions:

- Link leaves CXL mode

14.6.9 Link Speed Degradation below 8 GT/s

Test Steps:

1. Train the CXL link up to the highest speed possible (at least 8 GT/s).
2. Degrade the Link Down to a speed below CXL mode operation.
3. Link enters Detect state.

Pass Criteria:

- Link degrades to slower speed
- Link enters Detect state

Fail Conditions:

- Link remains in CXL mode
- Link does not change speed

14.6.10 Tests Requiring Injection Capabilities

The tests in this section are optional but strongly recommended. The test configuration control registers for the tests in this section are implementation specific.

14.6.10.1 TLP Ends on Flit Boundary

Test Equipment:

- Protocol Analyzer

Prerequisites:

- Applicable only when the link trains to 68B Flit mode

Test Steps:

1. Bring the link up to Active state.
2. CXL.io sends a TLP that ends on a flit boundary.
3. Check that next flit sent by the Link Layer contains IDLE tokens, EDB, or more data.

Pass Criteria:

- TLP that ends on flit boundary is not processed until a subsequent flit is transmitted
- IDLE tokens, EDB, or more data is observed after a TLP that ends on the flit boundary

Fail Conditions:

- Errors are logged
- No IDLE, EDB, or data observed after TLP flit

14.6.10.2 Failed CXL Mode Link Up

Test Steps:

1. Negotiate for CXL during PCIe alternate protocol negotiation.
2. Hold the link at 2.5 GT/s.
3. Link transitions back to detect.

Pass Criteria:

- Link transitions back to detect after being unable to reach 8 GT/s speed
- Link training does not complete

Fail Conditions:

- Link does not transition to detect

14.6.11 Link Initialization in Standard 256B Flit Mode

Prerequisites:

- Upstream Ports and Downstream Ports support PCIe Flit mode

Test Steps:

1. Train the CXL link up at the highest possible speed.

Pass Criteria:

- Link trains to L0 state
- PCIe Flit mode is selected during training - Flit Mode Status in the Link Status 2 register is set
- DVSEC Flex Bus Port Status register has IO_Enabled set and either one or both of Cache_Enabled and Mem_Enabled are set

Fail Conditions:

- Link training is incomplete
- PCIe Flit mode is not selected during training - Flit Mode Status in the Link Status 2 register is not set
- DVSEC Flex Bus Port Status register has IO_Enabled not set
- DVSEC Flex Bus Port Status register has both Cache_Enabled and Mem_Enabled not set

14.6.12 Link Initialization in Latency-Optimized 256B Flit Mode

Prerequisites:

- Upstream Ports and Downstream Ports support PCIe Flit mode
- Upstream Ports and Downstream Ports are Latency-Optimized 256B Flit capable

Test Steps:

1. Train the CXL link up at the highest possible speed.
 - a. During link training, set the CXL Latency_Optimized_256B_Flit_Enable bit in the Downstream Port's DVSEC Flex Bus Port Control register.

Pass Criteria:

- Link trains to L0 state
- PCIe Flit mode is selected during training - Flit Mode Status in the Link Status 2 register is set
- DVSEC Flex Bus Port Status register has CXL Latency_Optimized_256B_Flit_Enabled set
- DVSEC Flex Bus Port Status register has IO_Enabled set and either one or both of Cache_Enabled and Mem_Enabled set

Fail Conditions:

- Link training is incomplete
- PCIe Flit mode is not selected during training - Flit Mode Status in Link Status 2 register is not set
- DVSEC Flex Bus Port Status register has CXL Latency_Optimized_256B_Flit_Enable not set
- DVSEC Flex Bus Port Status register has IO_Enabled not set
- DVSEC Flex Bus Port Status register has both Cache_Enabled and Mem_Enabled not set

14.6.13 Sync Header Bypass (If Applicable)

Test Equipment:

- Protocol Analyzer

Prerequisites:

- Support for Sync Header Bypass

Test Steps:

1. Negotiate for Sync Header Bypass during PCIe alternate protocol negotiation.
2. Link trains to 2.5 GT/s.
3. Transition to each of the device-supported speeds: 8 GT/s, 16 GT/s, and 32 GT/s.
4. Check for Sync headers.

Pass Criteria:

- No Sync Headers are observed after 8 GT/s transition

14.7

Switch Tests

14.7.1

Introduction to Switch Types

CXL supports two types of switches (see [Section 7.7.5](#)):

- HBR (Hierarchy Based Routing)
- PBR (Port Based Routing)

14.7.2

Compliance Testing

Compliance testing of switches requires a “Golden reference” host and endpoint devices. These are devices that have been tested and are trusted to operate in accordance with the CXL specifications.

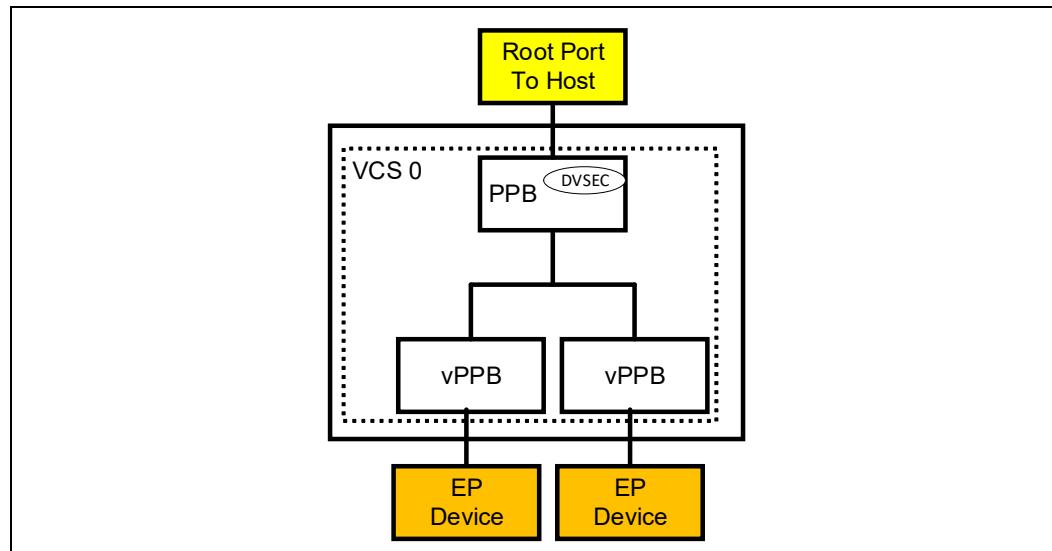
Assemble a topology to allow testing of the switches to confirm that the CXL protocol is unencumbered by the switches for interoperability, to include the following:

- Validate all EP devices and address ranges are identified and accessible to the host (root port)
- Run tests to verify that attached memory is visible to the host and operates correctly
- Testing by function
- Managed device removal
- Managed addition of devices
- Link Down testing, link recovery for switched ports
- Device reset events for individual EP devices

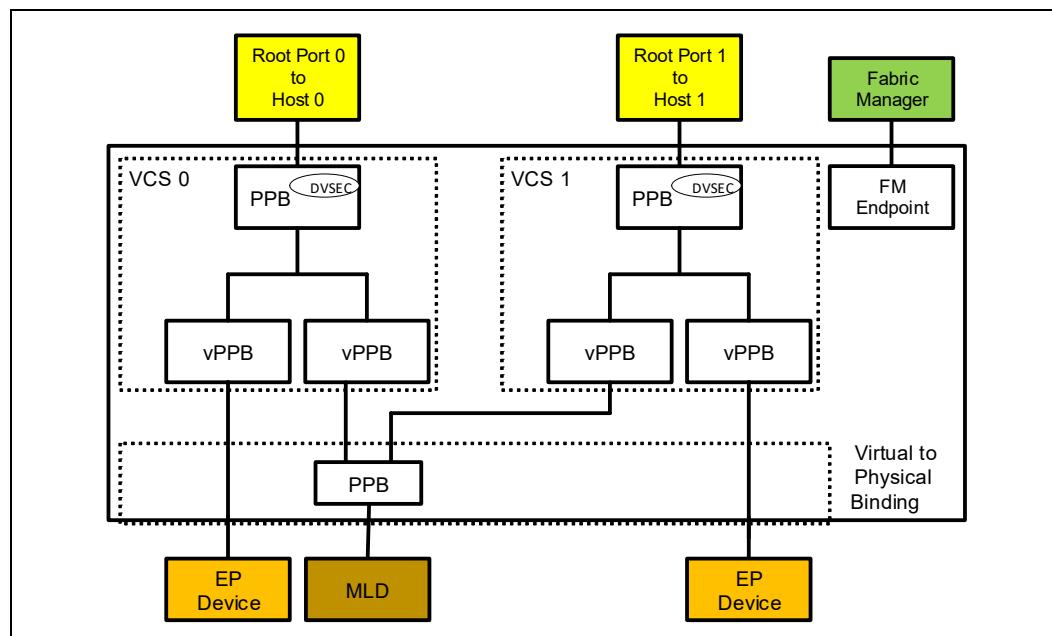
14.7.2.1

HBR Switch Assumptions

The minimum configuration for an HBR switch is not managed by an FM and is defined as one Virtual CXL Switch (VCS) that has a USP and two or more DSPs. Compliance tests for a single VCS.

Figure 14-13. Compliance Testing Topology for an HBR Switch with a Single Host

The minimum configuration for a managed switch is defined as two VCS: each VCS has one USP and two or more DSPs.

Figure 14-14. Compliance Testing Topology for an HBR Switch with Two Hosts

Known good Host devices are required to support managed Hot-Plug and managed removal of devices.

All connectors used in these tests must support Hot-Plug sideband signals.

An HBR switch that is not FM managed should have all ports bound to a VCS. An unmanaged switch cannot support unbound ports and MLDs because there is no managing function to control LD bindings.

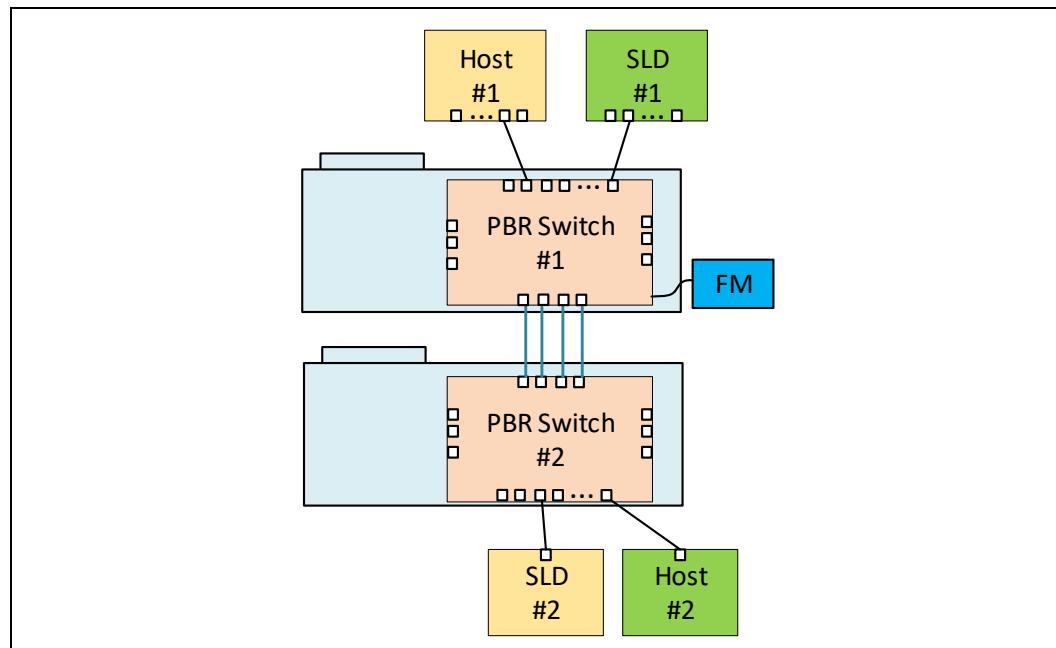
An FM-managed HBR switch should have at least two VCSs configured for these test purposes, so that interactions between hosts on different VCSs can be monitored. Devices may be connected to unbound ports for a managed switch (i.e., an unallocated resource). Unbound ports may be bound to any VCS at any time. The switch is managed by a Fabric Manager of the vendor's choice and supports MLDs.

A known good Endpoint should support Hot-Plug and should have passed previous tests in a direct attached system.

14.7.2.2 PBR Switch Assumptions

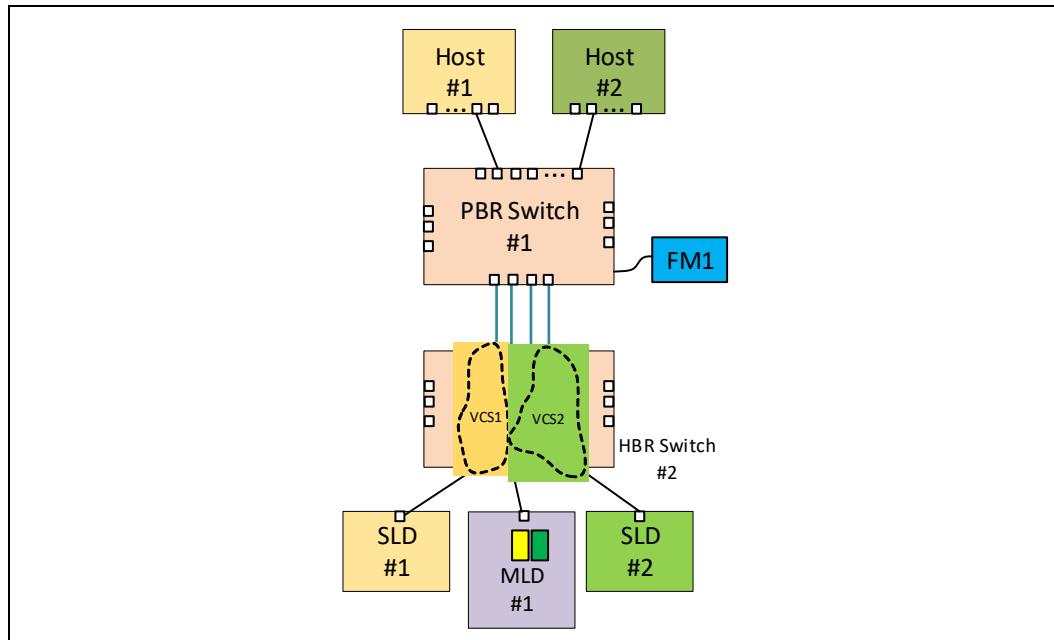
The minimum configuration for PBR switches is composed of two cascaded switches, at least one of which shall be a PBR switch. Switches shall be FM managed.

Figure 14-15. Compliance Testing Topology for Two PBR Switches



In a topology with a single PBR switch and a single HBR switch, the host devices are connected to the PBR switch and the HBR switch's USPs are connected to the PBR switch, to allow for multiple-host routing. The HBR switch configures a unique VCS for each host.

Figure 14-16. Compliance Testing Topology for a PBR Switch and an HBR Switch



14.7.3 Unmanaged HBR Switch

This is a fixed-configuration test. This test is used for an HBR switch that has the ability for bindings to be preconfigured and immediately accessible to the attached host after power-up. This test is suitable only for SLDs because MLDs require management to determine which LDs to bind to each VCS. All port bindings that define the VCS are configured and allocated at boot time without any interaction from a Fabric Manager device.

Test Steps:

1. An HBR switch that is not FM managed shall have all port bindings defined to be active at power-up.
2. An FM-managed HBR switch should be configured so that at least one port is bound to a VCS on power-up.
3. At least one SLD component shall be attached to a port.
4. Power-on or initialize the system (host, switch, and EP device).

Pass Criteria:

- Devices attached to bound ports are identified by the host at initialization without any external intervention by a Fabric Manager, if any

Fail Conditions:

- Devices attached to bound ports are not identified by the host on initialization

14.7.4 Reset Propagation

14.7.4.1 Host PERST# Propagation

HBR switch overview: If an HBR switch receives a USP PERST#, then only devices or SLDs that are bound to the VCS for that USP shall be reset; other VCSs and ports shall not be reset. For an MLD component, only LDs that are bound to the VCS that received the USP PERST# shall be reset. LDs that are bound to another VCS shall be unaffected and shall continue to operate normally.

PBR switch overview: If a PBR switch receives a PERST#, then only devices attached to ports with access to the receiving port shall be reset. No other ports shall be reset. MLDs are not supported by PBR switches. All other ports shall continue to operate normally.

14.7.4.1.1 Host PERST# Propagation to an SLD Component (HBR Switch)

Test Steps:

1. One or more SLDs are bound to a VCS.
2. Assert PERST# from the host to the USP of the VCS.

Pass Criteria:

- Switch propagates reset to all SLDs that are connected to the VCS
- All SLDs that are bound to the VCS go through a Link Down and the host unloads the associated device drivers
- Hosts and all devices that are bound to any other VCS shall continue to be connected and bound; reset events shall not occur

Fail Conditions:

- One or more SLDs that are bound to the VCS under test fails to go through a Link Down
- Hosts or SLDs that are bound to any other VCS are reset

14.7.4.1.2 Host PERST# Propagation to an SLD Component (PBR Switch)

Test Steps:

1. One or more SLDs has port access to a host.
2. PERST# is asserted by the host.

Pass Criteria:

- Switch propagates reset to all SLDs with port access to the host
- All SLD port access to the host goes through a Link Down and the host unloads the associated device drivers
- Hosts and all devices connected to other switch ports shall continue to be connected and no reset events occur

Fail Conditions:

- One or more SLDs with port access to the host under test fail to go through a Link Down
- Hosts or SLDs connected to other switch ports are reset

14.7.4.1.3 Host PERST# Propagation to an MLD Port (HBR Switch Only)

Prerequisites:

- Not applicable to PBR switches
- Switch with a minimum of two VCSs that are connected to respective Hosts
- An MLD with at least one LD that is bound to each VCS (i.e., at least two bound LDs)
- Optionally, SLDs may also be attached to each VCS

Test Steps:

1. Host 0 asserts USP PERST#.
2. Reset is propagated to all VCS 0 vPPBs.

Pass Criteria:

- Host 0 processes a Link Down for each LD that is bound to VCS 0 and unloads the associated device drivers
- All SLDs that are connected to VCS 0 go through a Link Down and Host 0 unloads the associated device drivers
- MLD remains link up
- Other hosts do not receive a Link Down for any LDs that are connected to them

Fail Conditions:

- Host 0 does not process a Link Down for the LDs and SLDs that are bound to VCS 0
- Any other host processes a Link Down for LDs of the shared MLD
- MLD goes through a Link Down

14.7.4.2 LTSSM Hot Reset

HBR switch overview: If a switch USP receives an LTSSM Hot Reset, then the USP vPPB shall propagate a reset to all vPPBs for that VCS. Other vPPBs shall not be reset. In a topology where an HBR switch is connected to a PBR switch, the USP of a VCS that is reset should reset the inter-switch link for the VCS USP.

PBR switch overview: If a PBR switch host port receives an LTSSM Hot Reset, then all switch ports with access to the host port shall be reset. No other ports shall be reset. Inter-switch links should not be reset.

14.7.4.2.1 LTSSM Hot Reset Propagation to SLDs (HBR Switch)

Test Steps:

1. One or more SLDs are bound to a VCS.
2. Initiate LTSSM Hot Reset from the host to the switch.

Pass Criteria:

- Switch propagates hot reset to all SLDs that are connected to the VCS and their links go down
- Hosts and devices bound to any other VCS must not receive the reset

Fail Conditions:

- Switch fails to send a hot reset to any SLDs that are connected to the VCS
- Hosts or devices bound to any other VCS are reset

14.7.4.2.2 LTSSM Hot Reset Propagation to SLDs (PBR Switch)**Test Steps:**

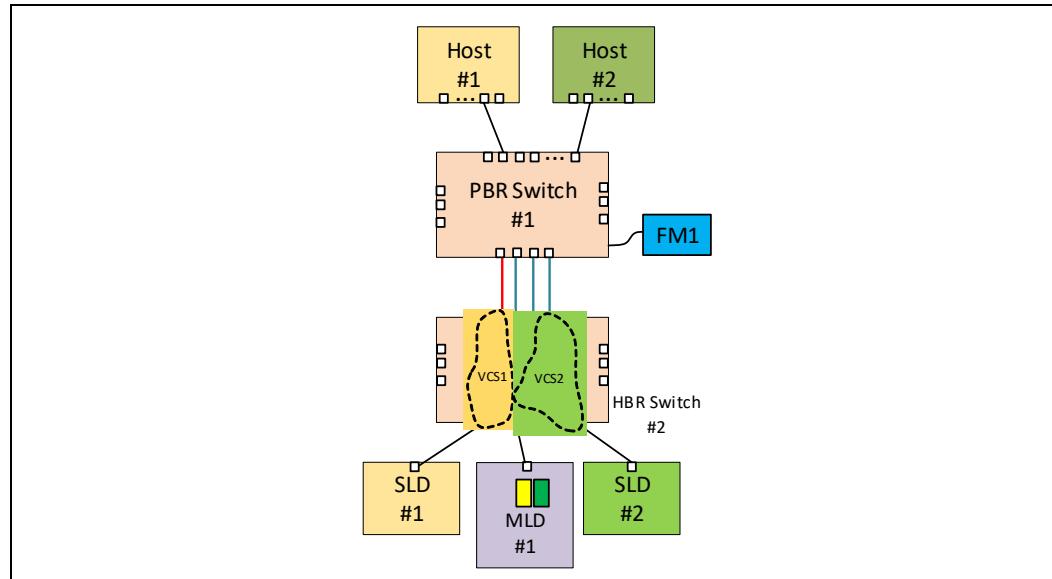
1. One or more SLDs have port access to the host port under test.
2. Initiate LTSSM Hot Reset from the host to the switch.

Pass Criteria:

- Switch propagates hot reset to all SLDs that are connected with port access to the host and their links go down
- Hosts and devices connected to other ports shall not receive a connection reset

Fail Conditions:

- Switch fails to send a hot reset to any SLDs that have port access to the host
- Hosts or devices connected to other ports are reset

14.7.4.2.3 LTSSM Hot Reset Propagation to SLDs (PBR+HBR Switch)**Figure 14-17. LTSSM Hot Reset Propagation to SLDs (PBR+HBR Switch)****Test Steps:**

1. A PBR switch and an HBR switch compose the topology, with the host connected to the PBR switch.
2. One or more SLDs have port access to the host port under test.
3. Initiate LTSSM Hot Reset from the host to the switch.

Pass Criteria:

- Switch propagates hot reset to all SLDs that are connected with port access to the host and their links go down
- The inter-switch link for the USP for the VCS of the HBR switch shall be reset (shown red in [Figure 14-17](#) (leftmost/first connecting line between the two switches), where VCS 1 received LTSSM reset)
- Hosts and devices connected to other ports shall not receive a connection reset

Fail Conditions:

- Switch fails to send a hot reset to any SLDs that have port access to the host
- Hosts or devices connected to other ports are reset

14.7.4.2.4 LTSSM Hot Reset Propagation to an MLD Component (HBR Switch Only)**Prerequisites:**

- Not applicable to PBR switches
- Switch with a minimum of two VCSs that are connected to respective Hosts
- An MLD with at least one LD that is bound to each VCS (i.e., at least two bound LDs)
- Optionally, SLDs may also be attached to each VCS

Test Steps:

1. Host 0 asserts LTSSM Hot Reset to the switch.
2. The USP propagates a reset to all vPPBs associated with VCS 0.

Pass Criteria:

- Host 0 processes a Link Down for all LDs and SLDs that are bound to VCS 0
- Host 1 does not receive a Link Down for any LDs that are bound to VCS 1

Fail Conditions:

- MLD port goes through a Link Down
- Host 1 processes a Link Down for LDs of the shared MLD
- Host 0 does not process a Link Down for any LD or SLD that is bound to VCS 0

14.7.4.3 Secondary Bus Reset (SBR) Propagation**14.7.4.3.1 Secondary Bus Reset (SBR) Propagation to All Ports of a VCS with SLD Components****Test Steps:**

1. One or more SLDs are bound to a VCS.
2. The Host sets the SBR bit in the Bridge Control register of the USP vPPB.

Pass Criteria:

- Switch sends a hot reset to all SLDs that are connected to the VCS and their links go down

- The Host processes a Link Down for all SLDs that are bound to the VCS and unloads the associated device drivers

Fail Conditions:

- Switch fails to send a hot reset to any SLDs that are connected to the VCS
- The Host fails to unload an associated device driver for a device that is connected to the VCS

14.7.4.3.2 Secondary Bus Reset (SBR) Propagation to All Ports of a VCS Including an MLD Component**Prerequisites:**

- Switch with a minimum of two VCSs that are connected to respective Hosts
- An MLD with at least one LD that is bound to each VCS (i.e., at least two bound LDs)
- Optionally, SLDs may also be attached to each VCS

Test Steps:

1. Host 0 sets the SBR bit in the Bridge Control register associated with the USP vPPB of the VCS under test.

Pass Criteria:

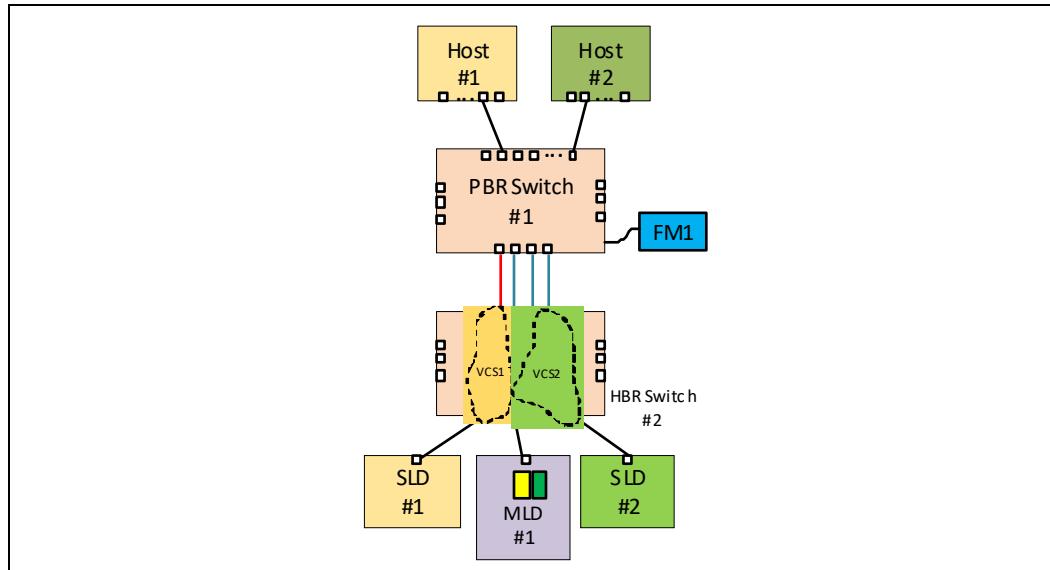
- Host 0 processes a Link Down for the LDs and SLDs that are bound to VCS 0 and unloads the associated device drivers
- MLD port remains Link Up
- Other Hosts that share the MLD are unaffected

Fail Conditions:

- MLD port goes through a Link Down
- Any other host processes a Link Down
- Host 0 does not process a Link Down for any LDs that are bound to VCS 0
- Host 0 does not process a Link Down for any SLDs that are connected to VCS 0

14.7.4.3.3 Secondary Bus Reset (SBR) Hot Reset Propagation to SLDs (PBR+HBR Switch)

Figure 14-18. Secondary Bus Reset (SBR) Hot Reset Propagation to SLDs (PBR+HBR Switch)



Test Steps:

1. A PBR switch and an HBR switch compose the topology, with the host connected to the PBR switch.
2. One or more SLDs have port access to the host port under test.
3. Initiate LTSSM Hot Reset from the host to the switch.

Pass Criteria:

- Switch propagates hot reset to all SLDs that are connected with port access to the host and their links go down
- The inter-switch link for the USP for the VCS of the HBR switch shall be reset (shown red in [Figure 14-18](#) (leftmost/first connecting line between the two switches), where VCS 1 received LTSSM reset)
- Hosts and devices connected to other ports shall not receive a connection reset

Fail Conditions:

- Switch fails to send a hot reset to any SLDs that have port access to the host
- Hosts or devices connected to other ports are reset

14.7.4.3.4 Secondary Bus Reset (SBR) Propagation to One Specific Downstream Port (SLD) (HBR Switch)

All links in the path between the host and specific SLD shall be reset.

Test Steps:

1. vPPB under test is connected to an SLD component.
2. Host sets the SBR bit in the Bridge Control register of the vPPB to be reset.

Pass Criteria:

- Host processes a Link Down for the vPPB under test and unloads the device driver
- All other ports in the VCS remain unaffected

Fail Conditions:

- Port under test does not go Link Down
- Any other port goes Link Down

14.7.4.3.5 Secondary Bus Reset (SBR) Propagation to One Specific Downstream Port (SLD) (PBR + HBR Switch)

All links in the path between the host and the specific SLD shall be reset, including the VCS USP for the VCS connected to the specific SLD being reset.

Test Steps:

1. A PBR switch and an HBR switch compose the topology, with the host connected to the PBR switch.
2. One or more SLDs have port access to the host port under test.
3. Initiate an SBR from the host to the switch for a specific SLD.

Pass Criteria:

- Host processes a Link Down for the SLD port under test
- Reset the ISL of the VCS USP containing the SLD that received the SBR
- All other ports remain unaffected

Fail Conditions:

- Port under test does not go Link Down
- ISL of the VCS USP containing the SLD that received the SBR failed to be reset
- Any other port goes Link Down

14.7.4.3.6 Secondary Bus Reset (SBR) Propagation to One Specific Shared Downstream Port (MLD) (HBR Switches Only)**Prerequisites:**

- Not applicable to PBR switches
- Switch with a minimum of two VCSs that are connected to respective Hosts
- Each VCS is bound to an LD each from the MLD component

Test Steps:

1. For the VCS under test, the host sets the SBR bit in the Bridge Control register of the vPPB bound to the LD.

Pass Criteria:

- Host processes a Link Down for the vPPB under test and unloads the device driver
- MLD port remains Link Up
- Other Hosts sharing the MLD are unaffected

Fail Conditions:

- Host processes a Link Down for the vPPB not under test
- Host does not process a Link Down for the vPPB under test
- Any switch port goes through a Link Down

14.7.5**Managed Hot-Plug - Adding a New Endpoint Device**

This test is for adding a device to a switch and then subsequently hot adding the device to a host. The host should load any relevant driver(s) and operate with the newly added device.

14.7.5.1**Managed Add of an SLD Component****14.7.5.1.1****Incremental Add of an SLD to a VCS (HBR Switch)****Prerequisites:**

- Host has completed enumeration
- Host has loaded drivers for all attached devices

Test Steps:

1. Perform a managed add of the SLD component to the port under test.
2. For an unmanaged switch, the port is already bound to a VCS.
3. For a managed switch, the FM must bind the port to a VCS.

Pass Criteria:

- Host successfully enumerates the added device and loads the driver

Fail Conditions:

- Host is unable to enumerate and fails to load the device driver for the added device

14.7.5.1.2**Incremental Add of an SLD to a VCS (PBR Switch)****Prerequisites:**

- Host has completed enumeration
- Host has loaded drivers for all attached devices

Test Steps:

1. Perform a managed add of the SLD component to the port under test.
2. FM identifies the new device and enables port routing to the required host.

Pass Criteria:

- Host successfully enumerates the added device and loads the device driver

Fail Conditions:

- Host is unable to enumerate and fails to load the device driver for the added device

14.7.5.2 Managed Add of an MLD Component (HBR Switch Only)

The Switch reports PPB-related events to the Fabric Manager using the FM API. At the time of publication there are no defined Fabric Manager reporting requirements to a user, and so parts of this test may only be observable through vendor-specific reporting.

Prerequisites:

- Not applicable to PBR switches
- Host enumeration successfully completes for all devices prior to this test
- Switch port supports MLD and is unbound (i.e., not bound to a VCS)

Test Steps:

1. Perform a managed add of the MLD to the port under test.

Pass Criteria:

- Fabric Manager identifies the device but does not bind it to any host
- Hosts are not affected by the addition of the device to an unbound port
- Hosts do not identify the added device
- Interrupts are not sent to the hosts, and the system operates normally

Fail Conditions:

- A host identifies the new device

14.7.5.3 Managed Add of an MLD Component to an SLD Port (HBR Switch Only)

This test exercises the behavior of an MLD component when plugged into an SLD port. If the MLD capability is not common to both sides, an MLD operates as an SLD component.

Prerequisites:

- Not applicable to PBR switches
- The port under test is configured as an SLD port
- Host enumeration successfully completes for all devices prior to this test

Test Steps:

1. Perform a managed add of the MLD component to the port under test.

Pass Criteria:

- Host successfully enumerates the added device and loads the driver.
- MLD component operates as an SLD (i.e., MLD capable but MLD is not enabled) and presents its full memory capacity to the host (i.e., does not divide into multiple LDs). For MH-MLDs, the component presents the full memory capacity that is allocated to the head under test.

Fail Conditions:

- Host does not identify the new device
- Host does not identify the full memory capacity of the new device

14.7.6 Managed Hot-Plug Removal of an Endpoint Device

A managed Hot-Plug remove operation requires the host to:

- Cease all read/write operations to the device
- Unload relevant drivers to allow the device to be removed

14.7.6.1 Managed Removal of an SLD Component from a VCS (HBR Switch)

Prerequisites:

- Host enumeration successfully completes for all devices prior to this test

Test Steps:

1. Perform a managed remove of the SLD component from the port under test.

Pass Criteria:

- Host recognizes the device removal and unloads the associated device driver

Fail Conditions:

- Host does not unload the device driver

14.7.6.2 Managed Removal of an SLD Component (PBR Switch)

Prerequisites:

- Host enumeration successfully completes for all devices prior to this test

Test Steps:

1. Perform a managed remove of the SLD component from the host.
2. FM removes port access for the SLD port and the host port.

Pass Criteria:

- Host recognizes the device removal and unloads the associated device driver

Fail Conditions:

- Host does not unload the device driver

14.7.6.3 Managed Removal of an MLD Component from a Switch (HBR Switch Only)

Prerequisites:

- Not applicable to PBR switches
- Host enumeration successfully completes for all devices prior to this test
- The MLD must have one or more LDs bound to the host

Test Steps:

1. Perform a managed remove of the MLD component from the port under test.
2. Fabric Manager unbinds LDs from the vPPBs of the VCS.

Pass Criteria:

- Host recognizes that the LD has been removed and unloads the associated device driver

Fail Conditions:

- Host does not recognize removal of the LD

14.7.6.4 Removal of a Device from an Unbound Port**Prerequisites:**

- Host enumeration successfully completes for all devices prior to this test
- Device is to be removed from an unbound port (i.e., not bound to any VCS)
- A device is connected to a switch, but the FM has:
 - Not bound the port to a VCS in an HBR switch, or
 - Not assigned port access to any other ports in a PBR switch

Test Steps:

1. Perform a managed remove of the device from the port under test.

Pass Criteria:

- Fabric Manager identifies that the device has been removed
- Hosts are not affected by the removal of the device from an unbound port
- Interrupts are not sent to hosts, and the system operates normally

Fail Conditions:

- A host is affected by the removal of the device

14.7.7 Bind/Unbind and Port Access Operations

HBR switches report PPB-related events to the Fabric Manager, using the FM API. Port changes on PBR switches are detected by a Fabric Manager. At the time of publication, there are no defined Fabric Manager-reporting requirements to the user, so parts of this test may only be observable through vendor-specific reporting.

Prerequisites:

- Applicable only to managed switches
- While the endpoint device remains connected to the port, the FM must:
 - Bind or unbind ports for an HBR switch, or
 - Enable or disable port access to other ports in a PBR switch

14.7.7.1 Binding and Granting Port Access of Pooled Resources to Hosts**14.7.7.1.1 Bind a Pooled SLD to a vPPB in an FM-Managed HBR Switch****Prerequisites:**

- An SLD component is connected to a switch port that is not bound to a VCS
- Fabric Manager has identified the SLD

Test Steps:

- Bind the SLD to a vPPB of the host.

Pass Criteria:

- Host recognizes the hot-added SLD and successfully enumerates the SLD
- Fabric Manager indicates that the SLD has been bound to the correct VCS

Fail Conditions:

- Host does not successfully process the SLD's managed add
- Fabric Manager does not indicate a successful bind operation

14.7.7.1.2 Assign Port Access of a Pooled SLD to a PBR Switch**Prerequisites:**

- Pooled SLD component is connected to a switch port that has not granted port access by the FM to any other ports
- Fabric Manager has identified the SLD

Test Steps:

- FM assigns port access of the SLD to the host port.

Pass Criteria:

- Host recognizes the hot-added SLD and successfully enumerates the SLD

Fail Conditions:

- Host does not successfully process the SLD's managed add

14.7.7.1.3 Binding an MLD to Two Different VCSs (HBR Switch Only)**Prerequisites:**

- Not applicable to PBR switches
- An MLD component is connected to the Switch and the Fabric Manager has identified the MLD
- MLD has two or more LDs that are not bound to any hosts

Test Steps:

- Bind one or more LDs to VCS 0.
- Bind one or more LDs to VCS 1.

Pass Criteria:

- Both hosts recognize the hot-added LDs and successfully enumerate both LDs
- Fabric Manager indicates that the LDs have been bound to the correct VCS

Fail Conditions:

- One or both hosts fail to recognize, enumerate, and load drivers for the hot-added LDs

- Fabric Manager indicates that one or more of the LDs are not bound to the correct VCSs

14.7.7.2 Unbinding Resources from Hosts without Removing the Endpoint Devices

This test takes an allocated resource and unbinds it from a host. The resource remains available, but unallocated after a successful unbind operation.

14.7.7.2.1 Unbind an SLD from a VCS (HBR Switch)

Prerequisites:

- An SLD component is bound to the vPPB of a VCS in an FM-managed switch
- Associated host loads the device driver for the SLD

Test Steps:

1. FM unbinds the SLD from the vPPB of the VCS.

Pass Criteria:

- Host recognizes the SLD's hot removal and successfully unloads the device driver
- Fabric Manager indicates that the SLD is present but has been unbound from the VCS
- SLD remains linked up

Fail Conditions:

- Host does not successfully process the SLD's managed removal
- Fabric Manager does not indicate a successful unbind operation
- SLD link goes down

14.7.7.2.2 Deallocate an SLD from a Host (PBR Switch)

Prerequisites:

- Host has port access to the SLD
- Host has loaded drivers

Test Steps:

1. FM indicates a managed removal to the host.
2. When the host completes hot removal, the FM revokes port access to the SLD, and then only the FM has access to the SLD.

Pass Criteria:

- Host recognizes the SLD's hot removal and successfully unloads the device driver
- FM indicates that the SLD is present and that there is no port access to other ports
- SLD remains linked up

Fail Conditions:

- Host does not successfully process the SLD's managed removal
- FM fails to revoke port access to the SLD for other ports

- SLD link goes down

14.7.7.2.3 Unbind LDs from Two Host VCSs (HBR Switch Only)

Prerequisites:

- Not applicable to PBR switches
- An MLD component is connected to the switch and the Fabric Manager has identified the MLD
- MLD component has LDs that are bound to two or more host VCSs

Test Steps:

1. FM unbinds the LDs from the vPPBs of the host VCSs.

Pass Criteria:

- All hosts successfully recognize the managed removal of the LDs and unload the device drivers
- FM indicates that the LDs are present but have been unbound from the VCSs
- MLD remains linked up and all other LDs are unaffected

Fail Conditions:

- One or more hosts do not successfully process the managed removal of the LDs
- FM status does not indicate a successful unbind operation
- Other LDs in the MLD are impacted

14.7.8 Error Injection

Test Equipment:

- A Jammer, Exerciser, or analyzer is required for many of these tests

Prerequisites:

- Errors are injected into the DSP; therefore, the error status registers in the associated vPPB should reflect the injected error

14.7.8.1 AER Error Injection

An MLD port must ensure that the vPPB associated with each LD that is bound is notified of errors that are not vPPB specific.

14.7.8.1.1 AER Uncorrectable Error Injection for MLD Ports

Test Equipment:

- This test requires an Exerciser if the MLD component is not capable of error injection

Prerequisites:

- vPPB of VCS 0 and vPPB of VCS 1 are each bound to LDs from the same MLD component

Test Steps:

1. Inject a CXL.io unmasked uncorrectable error into the MLD port of the Switch. The injected error should be categorized as 'supported per vPPB' per [Section 7.2.7](#).

Pass Criteria:

- The Uncorrectable Error Status register for the vPPBs that are bound to the LDs should reflect the appropriate error indicator bit
- The Uncorrectable Error Status register for the FM-owned PPB should reflect the appropriate error indicator bit

Fail Conditions:

- PPB or vPPB AER Uncorrectable Error Status does not reflect the appropriate error indicator bit

14.7.8.1.2 AER Correctable Error Injection for MLD Ports

Test Equipment:

- This test requires an Exerciser if the MLD component is not capable of error injection

Prerequisites:

- vPPB of VCS 0 and vPPB of VCS 1 are each bound to LDs from the same MLD component

Test Steps:

1. Inject a CXL.io correctable error into the MLD port of the Switch. The injected error should be categorized as 'supported per vPPB' per [Section 7.2.7](#).

Pass Criteria:

- The Correctable Error status register for the vPPBs that are bound to the LDs should reflect the appropriate error indicator bit
- The Correctable Error status register for the FM-owned PPB should reflect the appropriate error indicator bit

Fail Conditions:

- PPB or vPPB AER Correctable Error status does not reflect the appropriate error indicator bit

14.7.8.1.3 AER Uncorrectable Error Injection for SLD Ports

Test Equipment:

- This test requires an Exerciser if the SLD component is not capable of error injection

Prerequisites:

- Host enumeration successfully completes for all devices prior to this test

Test Steps:

1. Inject a CXL.io unmasked uncorrectable error into the SLD port under test.

14.7.8.1.4 AER Correctable Error Injection for SLD Ports

Test Equipment:

- This test requires an Exerciser if the SLD component is not capable of error injection

Prerequisites:

- Host enumeration successfully completes for all devices prior to this test

Test Steps:

1. Inject a CXL.io correctable error into the SLD port under test.

Pass Criteria:

- The Correctable Error status register for the vPPB that is bound to the SLD should reflect the appropriate error indicator bit

Fail Conditions:

- The vPPB AER status does not reflect the appropriate error indicator bit

14.8 Configuration Register Tests

Configuration space register cover the registers defined in [Chapter 3.0](#). These tests are run on the DUT, and require no additional hardware to complete. Tests must be run with Root/Administrator privileges. Test makes the assumption that there is one and only one CXL device in the system, and it is the DUT. This test section has granularity down to the CXL device.

See [Section 14.2.1](#) for topology definitions that are referenced in this section.

14.8.1 Device Presence

Prerequisites:

- Applicable for VH components

Test Steps:

1. If the DUT is a CXL switch:
 - a. Read the PCIe device hierarchy and filter for PCIe Upstream Port/Downstream Port of a switch.
 - b. Locate the PCIe Upstream/Downstream Port with PCIe DVSEC Capability with VID of 1E98h and ID of 0000h (PCIe DVSEC for CXL device).
 - c. Save the PCIe device location for further tests. This will be referred to in subsequent tests as the DUT.
2. If the DUT is a CXL endpoint:

- a. Read the PCI* device hierarchy and filter for PCIe Endpoint Devices.
 - b. Locate the PCIe Endpoint device with PCIe DVSEC Capability with VID of 1E98h and ID of 0000h (PCIe DVSEC for CXL device).
 - c. Save the PCIe device location for further tests. This will be referred to in subsequent tests as the DUT.
3. If the DUT is a CXL root port:
 - a. Read the PCI device hierarchy and filter for PCIe root port devices.
 - b. Locate the PCIe root port device with PCIe DVSEC Capability with VID of 1E98h and ID of 0000h (PCIe DVSEC for CXL device).
 - c. Save the PCIe device location for further tests. This will be referred to in subsequent tests as the DUT.

Pass Criteria:

- One PCIe device with CXL PCIe DVSEC Capability found

Fail Conditions:

- PCIe device with CXL PCIe DVSEC Capability not found

14.8.2 CXL Device Capabilities

Prerequisites:

- Device is CXL.mem capable

Test Steps:

1. Read the configuration space for the DUT.
2. Initialize variables with value 0.
3. Search for PCIe DVSEC for CXL device:
 - a. Read the configuration space for the DUT. Search for a PCIe DVSEC with VID of 1E98h and ID of 0000h.
 - b. Save this location as CXL_DEVICE_DVSEC_BASE.
4. Search for Non-CXL Function Map DVSEC:
 - a. Read the configuration space for the DUT. Search for a PCIe DVSEC with VID of 1E98h and ID of 0002h.
 - b. If found, save this location as NON_CXL_FUNCTION_DVSEC_BASE.
5. Search for CXL Extensions DVSEC for ports:
 - a. Read the configuration space for the DUT. Search for a PCIe DVSEC with VID of 1E98h and ID of 0003h.
 - b. If found, save this location as CXL_EXTENSION_DVSEC_BASE.
6. Search for GPF DVSEC for CXL ports:
 - a. Read the configuration space for the DUT. Search for a PCIe DVSEC with VID of 1E98h and ID of 0004h.
 - b. If found, save this location as CXL_GPF_PORT_DVSEC_BASE.
7. Search for GPF DVSEC for CXL devices:
 - a. Read the configuration space for the DUT. Search for a PCIe DVSEC with VID of 1E98h and ID of 0005h.

- b. If found, save this location as CXL_GPF_DEVICE_DVSEC_BASE.
8. Search for PCIe DVSEC for Flex Bus Port:
 - a. Read the configuration space for the DUT. Search for a PCIe DVSEC with VID of 1E98h and ID of 0007h.
 - b. If found, save this location as CXL_FLEXBUS_DVSEC_BASE.
9. Search for Register Locator DVSEC:
 - a. Read the configuration space for the DUT. Search for a PCIe DVSEC with VID of 1E98h and ID of 0008h.
 - b. If found, save this location as CXL_REGISTER_DVSEC_BASE.
10. Search for MLD DVSEC:
 - a. Read the configuration space for the DUT. Search for a PCIe DVSEC with a VID of 1E98h and ID of 0009h.
 - b. If found, save this location as CXL_MLD_DVSEC_BASE.
11. Search for Advanced Error Reporting Capability:
 - a. If found, save this location as AER_BASE.
12. Search for Table Access DOE:
 - a. Read Configuration Space for the DUT. Search for PCI Express* DVSEC with VID of 1E98h and type of 0002h.
 - b. If found, save this location as CXL_TABLE_ACCESS_DOE_BASE.
13. Verify:

Variable	Condition
CXL_DEVICE_DVSEC_BASE != 0	Always
CXL_EXTENSION_DVSEC_BASE != 0	Device is root port, Upstream Port, or Downstream Port of a switch
CXL_GPF_PORT_DVSEC_BASE != 0	Device is root port or Downstream Port of a switch
CXL_GPF_DEVICE_DVSEC_OFFSET != 0	Device is CXL.mem and supports GPF
CXL_FLEXBUS_DVSEC_BASE != 0	Always
CXL_REGISTER_DVSEC_BASE != 0	Always
CXL_MLD_DVSEC_BASE != 0	Device is MLD
AER_BASE != 0	Always
CXL_TABLE_ACCESS_DOE_BASE != 0	Always

Pass Criteria:

- Test [14.8.1](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.3 DOE Capabilities

Prerequisites:

- DOE is implemented

Test Steps:

1. Read the Configuration space for DUT.
2. Loop until end of configuration space capabilities are found.
 - a. Search for DOE mailbox:

- i. Read the configuration space for DUT. Search for a PCIe Extended Capability with type of 2Eh.
- b. If found, repeatedly issue DOE Discovery until the response contains Vendor ID = FFFFh to get the list of supported Object Protocols for this mailbox.
- c. If a response contains Vendor ID = 1E98h and Data Object Protocol = 0h:
 - i. Save Mailbox location as CXL_COMPLIANCE_DOE_MAILBOX.
- d. If a response contains Vendor ID = 1E98h and Data Object Protocol = 2h:
 - i. Save Mailbox location as CXL_CDAT_DOE_MAILBOX.

Pass Criteria:

- Test 14.8.2 passed
- Either Compliance or CDAT DOE mailbox has a valid response

Fail Conditions:

- Verify Conditions failed

14.8.4 DVSEC Control Structure

Test Steps:

1. Read the Configuration space for DUT, CXL_DEVICE_DVSEC_BASE + Offset 04h, Length 4 bytes.

2. Decode this into:

Bits	Variable
15:0	DVSEC Vendor ID
19:16	DVSEC Revision
31:20	DVSEC Length

3. Verify:

Variable	Value	Condition
DVSEC Vendor ID	1E98h	Always
DVSEC Revision	2h	Always
DVSEC Length	3Ch	Always

4. Read the Configuration space for DUT, CXL_DEVICE_DVSEC_BASE + Offset 08h, Length 2 bytes.

5. Decode this into:

Bits	Variable
15:0	DVSEC ID

6. Verify:

Variable	Value	Condition
DVSEC ID	0000h	Always

Pass Criteria:

- Test 14.8.2 passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.5 DVSEC CXL Capability

Test Steps:

1. Read Configuration Space for DUT, CXL_DEVICE_DVSEC_BASE + Offset 0Ah, Length 2 bytes.
2. Decode this into:

Bits	Variable
0:0	Cache_Capable
1:1	IO_Capable
2:2	Mem_Capable
3:3	Mem_HWInit_Mode
5:4	HDM_Count
6:6	Cache Writeback and Invalidate Capable
7:7	CXL Reset Capable
10:8	CXL Reset Timeout
14:14	Viral Capable
15:15	PM Init Completion Reporting Capable

3. Verify:

Variable	Value	Condition
IO_Capable	= 1	Always
HDM_Count	!= 11b	Always
HDM_Count	!= 00b	Mem_Capable = 1
HDM_Count	= 00b	Mem_Capable = 0
CXL Reset Timeout	> 100b	CXL Reset Capable = 1

Pass Criteria:

- Test 14.8.4 passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.6 DVSEC CXL Control

Test Steps:

1. Read the Configuration space for DUT, CXL_DEVICE_DVSEC_BASE + Offset 0Ch, Length 2 bytes.
2. Decode this into:

Bits	Variable
0:0	Cache_Enable
1:1	IO_Enable
2:2	Mem_Enable
7:3	Cache_SF_Coverage
10:8	Cache_SF_Granularity
11:11	Cache_Clean_Eviction
14:14	Viral_Enable

3. Verify:

Variable	Value	Condition
IO_Enable	== 1	Always
Cache_SF_Granularity	!= 111b	Always

Pass Criteria:

- Test [14.8.4](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.7 DVSEC CXL Lock**Test Steps:**

1. Read Configuration Space for DUT, CXL_DEVICE_DVSEC_BASE + Offset 14h, Length 2 bytes.

2. Decode this into:

Bits	Variable
0:0	CONFIG_LOCK

3. Read Configuration Space for DUT as per the following list, and then store it as a 'List of Config Lock Registers' for the next steps of this test.

Note:

These are only locked by Config Lock (see [Section 8.2.4.20.13](#)). There are other registers that are marked as RWL but a lock bit is not mentioned.

DVSEC CXL Control (Offset 0Ch)

Bits	Variable
0:0	Cache_Enable
2:2	Mem_Enable
7:3	Cache_SF_Coverage
10:8	Cache_SF_Granularity
11	Cache_Clean_Eviction
14	Viral_Enable

DVSEC CXL Range 1 Base High (Offset 20h)

Bits	Variable
31:0	Memory_Base_High

DVSEC CXL Range 1 Base Low (Offset 24h)

Bits	Variable
31:28	Memory_Base_Low

DVSEC CXL Range 2 Base High (Offset 30h)

Bits	Variable
31:0	Memory_Base_High

4. Record all read values for each variable into the 'Read Value List' – R1.

5. Write Configuration for all registers listed above in the 'List of Config Lock Registers' with inverted values.
6. Record all read values for each variable into the 'Read Value List' – R2.
7. Verify:

Variable	Value	Condition
R1	= R2	CONFIG_LOCK = 1
R1	!= R2	CONFIG_LOCK = 0

Pass Criteria:

- Test [14.8.4](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.8

DVSEC CXL Capability2

Test Steps:

1. Read the Configuration space for DUT, CXL_DEVICE_DVSEC_BASE + Offset 16h, Length 2 bytes.
2. Decode this into:

Bits	Variable
3:0	Cache Size Unit
15:8	Cache Size

3. Verify:

Variable	Value	Condition
Cache Size Unit	= 0h	Cache Capable = 0
Cache Size Unit	!= 0h	Cache Capable = 1
Cache Size Unit	> 2h	Always

Pass Criteria:

- Test [14.8.4](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.9

Non-CXL Function Map DVSEC

Test Steps:

1. Read the Configuration space for DUT, NON_CXL_FUNCTION_DVSEC_BASE + Offset 04h, Length 4 bytes.
2. Decode this into:

Bits	Variable
15:0	DVSEC Vendor ID
19:16	DVSEC Revision
31:20	DVSEC Length

3. Verify:

Variable	Value	Condition
DVSEC Vendor ID	1E98h	Always
DVSEC Revision	0h	Always
DVSEC Length	02Ch	Always

4. Read the Configuration space for DUT, Offset 08h, Length 2 bytes.

5. Decode this into:

Bits	Variable
15:0	DVSEC ID

6. Verify:

Variable	Value	Condition
DVSEC ID	0002h	Always

Pass Criteria:

- Test [14.8.2](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.10 CXL Extensions DVSEC for Ports Header

Prerequisites:

- DUT is root port, Upstream Port, or Downstream Port of a switch

Test Steps:

1. Read the Configuration space for DUT, CXL_EXTENSION_DVSEC_BASE + Offset 04h, Length 4 bytes.

2. Decode this into:

Bits	Variable
15:0	DVSEC Vendor ID
19:16	DVSEC Revision
31:20	DVSEC Length

3. Verify:

Variable	Value	Condition
DVSEC Vendor ID	1E98h	Always
DVSEC Revision	0h	Always
DVSEC Length	028h	Always

4. Read the Configuration space for DUT, CXL_EXTENSION_DVSEC_BASE + Offset 08h, Length 2 bytes.

5. Decode this into:

Bits	Variable
15:0	DVSEC ID

6. Verify:

Variable	Value	Condition
DVSEC ID	0003h	Always

Pass Criteria:

- Test [14.8.2](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.11 Port Control Override**Prerequisites:**

- DUT is root port, Upstream Port, or Downstream Port of a switch

Test Steps:

1. Read the Configuration space for DUT, CXL_EXTENSION_DVSEC_BASE + Offset 0Ch, Length 4 bytes.
2. Verify:

Bits	Value
0:0	0
1:1	0
3. Verify:
 - a. For Ports operating in PCIe mode or RCD mode:
 - i. Verify that the port's SBR functionality is as defined in PCIe Base Specification.
 - ii. Verify that the Link Disable functionality follows PCIe Base Specification.
 - b. For Ports operating in 68B Flit mode:
 - i. Verify that writing to the SBR bit in the Bridge Control register of this Port has no effect.
 - ii. Verify that writing to the Link Disable bit in the Link Control register of this Port has no effect.
4. Store '1' into Bit 0 at Offset 0Ch.
5. Verify:
 - a. For Ports operating in PCIe mode or RCD mode, verify that the port's SBR functionality is as defined in PCIe Base Specification.
 - b. For Ports operating in 68B Flit mode, verify that writing to the SBR bit in the Bridge Control register of this Port results in the port generating a hot reset.
6. Store '0' into Bit 0 at Offset 0Ch.
7. Store '1' into Bit 1 at Offset 0Ch.
8. Verify:
 - a. For Ports operating in PCIe mode or RCD mode, verify that the port's Link Disable functionality is as defined in PCIe Base Specification.
 - b. For Ports operating in 68B Flit mode, verify that writing to the Link Disable bit in the Link Control register of this Port results in the Port being able to disable and re-enable the link.

Pass Criteria:

- Test [14.8.10](#) passed

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.12 GPF DVSEC Port Capability

Prerequisites:

- DUT is a root port or a Downstream Port of a switch

Test Steps:

1. Read the Configuration space for DUT, CXL_GPF_PORT_DVSEC_BASE + Offset 04h, Length 4 bytes.
2. Decode this into:

Bits	Variable
15:0	DVSEC Vendor ID
19:16	DVSEC Revision
31:20	DVSEC Length

3. Verify:

Variable	Value	Condition
DVSEC Vendor ID	1E98h	Always
DVSEC Revision	0h	Always
DVSEC Length	010h	Always

4. Read the Configuration space for DUT, CXL_GPF_PORT_DVSEC_BASE + Offset 08h, Length 2 bytes.

5. Decode this into:

Bits	Variable
15:0	DVSEC ID

6. Verify:

Variable	Value	Condition
DVSEC ID	0004h	Always

Pass Criteria:

- Test 14.8.2 passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.13 GPF Port Phase 1 Control

Prerequisites

- DUT is a root port or a Downstream Port of a switch

Test Steps:

1. Read the Configuration space for DUT, CXL_GPF_PORT_DVSEC_BASE + Offset 0Ch, Length 2 bytes.

2. Decode this into:

Bits	Variable
11:8	Port GPF Phase 1 Timeout Scale

3. Verify:

Variable	Value	Condition
Port GPF Phase 1 Timeout Scale	< 8h	Always

Pass Criteria:

- Test [14.8.12](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.14 GPF Port Phase 2 Control

Prerequisites:

- DUT is a root port or a Downstream Port of a switch

Test Steps:

1. Read the Configuration space for DUT, CXL_GPF_PORT_DVSEC_BASE + Offset 0Eh, Length 2 bytes.

2. Decode this into:

Bits	Variable
11:8	Port GPF Phase 2 Time Scale

3. Verify:

Variable	Value	Condition
Port GPF Phase 2 Time Scale	< 8h	Always

Pass Criteria:

- Test [14.8.12](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.15 GPF DVSEC Device Capability

Prerequisites:

- Device is CXL.mem capable
- Device is GPF capable

Test Steps:

1. Read the Configuration space for DUT, CXL_GPF_DEVICE_DVSEC_BASE + Offset 04h, Length 4 bytes.

2. Decode this into:

Bits	Variable

15:0	DVSEC Vendor ID
19:16	DVSEC Revision
31:20	DVSEC Length

3. Verify:

Variable	Value	Condition
DVSEC Vendor ID	1E98h	Always
DVSEC Revision	0h	Always
DVSEC Length	010h	Always

4. Read the Configuration space for DUT, CXL_GPF_DEVICE_DVSEC_BASE + Offset 08h, Length 2 bytes.

5. Decode this into:

Bits	Variable
15:0	DVSEC ID

6. Verify:

Variable	Value	Condition
DVSEC ID	0005h	Always

Pass Criteria:

- Test [14.8.2](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.16 GPF Device Phase 2 Duration

Prerequisites:

- Device is CXL.mem capable
- Device is GPF capable

Test Steps:

1. Read the Configuration space for DUT, CXL_GPF_DEVICE_DVSEC_BASE + Offset 0Ah, Length 2 bytes.

2. Decode this into:

Bits	Variable
11:8	Device GPF Phase 2 Time Scale

3. Verify:

Variable	Value	Condition
Device GPF Phase 2 Time Scale	< 8h	Always

Pass Criteria:

- Test [14.8.15](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.17 Flex Bus Port DVSEC Capability Header

Test Steps:

1. Read the Configuration space for DUT, CXL_FLEXBUS_DVSEC_BASE + Offset 04h, Length 4 bytes.
2. Decode this into:

Bits	Variable
15:0	DVSEC Vendor ID
19:16	DVSEC Revision
31:20	DVSEC Length

3. Verify:

Variable	Value	Condition
DVSEC Vendor ID	1E98h	Always
DVSEC Revision	2h	Always
DVSEC Length	20h	Always

4. Read CXL_FLEXBUS_DVSEC_BASE + Offset 08h, Length 2 bytes.

5. Decode this into:

Bits	Variable
15:0	DVSEC ID

6. Verify:

Variable	Value	Condition
DVSEC ID	0007h	Always

Pass Criteria:

- Test 14.8.2 passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.18 DVSEC Flex Bus Port Capability

Test Steps:

1. Read the Configuration space for DUT, CXL_FLEXBUS_DVSEC_BASE + Offset 0Ah, Length 2 bytes.
2. Decode this into:

Bits	Variable
0:0	Cache_Capable
1:1	IO_Capable
2:2	Mem_Capable
5:5	68B_Flit_and_VH_Capable
6:6	CL_MLD_Capable

3. Verify:

Variable	Value	Condition
IO_Capable	1	Always
68B_Flit_and_VH_Capable	1	Always

Pass Criteria:

- Test [14.8.2](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.19 Register Locator

Test Steps:

1. Read the Configuration space for DUT, CXL_REGISTER_DVSEC_BASE + Offset 04h, Length 4 bytes.
2. Decode this into:

Bits	Variable
15:0	DVSEC Vendor ID
19:16	DVSEC Revision
31:20	DVSEC Length

3. Verify:

Variable	Value	Condition
DVSEC Vendor ID	1E98h	Always
DVSEC Revision	0h	Always

4. Read the Configuration space for DUT, CXL_REGISTER_DVSEC_BASE + Offset 08h, Length 2 bytes.

5. Decode this into:

Bits	Variable
15:0	DVSEC ID

6. Verify:

Variable	Value	Condition
DVSEC ID	0008h	Always

Pass Criteria:

- Test [14.8.2](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.20 MLD DVSEC Capability Header

Prerequisites:

- Device is MLD capable

Test Steps:

1. Read the Configuration space for DUT, CXL_MLD_DVSEC_BASE + Offset 04h, Length 4 bytes.
2. Decode this into:

Bits	Variable
15:0	DVSEC Vendor ID
19:16	DVSEC Revision
31:20	DVSEC Length

3. Verify:

Variable	Value	Condition
DVSEC Vendor ID	1E98h	Always
DVSEC Revision	0h	Always
DVSEC Length	010h	Always

4. Read the Configuration space for DUT, Offset 08h, Length 2 bytes.

5. Decode this into:

Bits	Variable
15:0	DVSEC ID

6. Verify:

Variable	Value	Condition
DVSEC ID	0009h	Always

Pass Criteria:

- Test [14.8.2](#) Device Present passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.21 MLD DVSEC Number of LD Supported

Prerequisites:

- Device is MLD capable

Test Steps:

1. Read the Configuration space for DUT, CXL_MLD_DVSEC_BASE + Offset 0Ah, Length 2 bytes.

2. Decode this into:

Bits	Variable
15:0	Number of LDs Supported

3. Verify:

Variable	Value	Condition
Number of LDs Supported	≤ 16	Always
Number of LDs Supported	$\neq 0$	Always

Pass Criteria:

- Test [14.8.20](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.22 Table Access DOE

Prerequisites:

- Device supports Table Access DOE

Test Steps:

1. For the following steps, use the DOE mailbox at CXL_CDAT_DOE_MAILBOX.
2. Issue DOE Read Entry:

Offset	Length in Bytes	Value
00h	2	1E98h
02h	1	02h
04h	2	03h
08h	1	00h
09h	1	00h
0Ah	2	0000h

3. Read Response and decode this into:

Offset	Length in Bytes	Variable
08h	1	Table Access Response Code
09h	1	Table Type

4. Verify:

Variable	Value	Condition
Table Access Response Code	0	Always
Table Type	0	Always

Pass Criteria:

- Test 14.8.3 passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.8.23 PCI Header - Class Code Register

Prerequisites:

- DUT is a CXL.mem device

Test Steps:

1. Read the Configuration space for DUT, Offset 09h, Length 4 bytes.
2. Decode this into:

Bits	Variable
7:0	Programming Interface (PI)
15:8	Sub Class Code (SCC)
23:16	Base Class Code (BCC)

3. Verify:

Variable	Value	Condition
Programming Interface (PI)	10h	Always
Sub Class Code (SCC)	02h	Always
Base Class Code (BCC)	05h	Always

Pass Criteria:

- Verify Conditions are met

Failed Conditions:

- Verify Conditions failed

14.9 Reset and Initialization Tests

14.9.1 Warm Reset Test

Prerequisites:

- DUT must be in D3 state with context flushed

Test Steps:

1. Host issues CXL PM VDM, Reset Prep (ResetType= Warm Reset; PrepType=General Prep).
2. Host waits for CXL device to respond with CXL PM VDM ResetPrepAck.

Pass Criteria:

- DUT responds with an ACK

Fail Conditions:

- DUT fails to respond to ACK

14.9.2 Cold Reset Test

Prerequisites:

- DUT must be in D3 state with context flushed

Test Steps:

1. Host issues CXL PM VDM, Reset Prep (ResetType= Warm Reset; PrepType=General Prep).
2. Host waits for CXL device to respond with CXL PM VDM ResetPrepAck.

Pass Criteria:

- DUT responds with an ACK

Fail Conditions:

- DUT fails to respond to ACK

14.9.3 Sleep State Test

Prerequisites:

- DUT must be in D3 state with context flushed

Test Steps:

1. Host issues CXL PM VDM, Reset Prep (ResetType= S3; PrepType=General Prep).
2. Host waits for the CXL device to respond with CXL PM VDM ResetPrepAck.

Pass Criteria:

- DUT responds with an ACK

Fail Conditions:

- DUT fails to respond to ACK

14.9.4 Function Level Reset Test

This test is accomplished by running the Application Layer tests as described in [Section 14.3.6.1](#), and issuing a Function Level Reset in the middle of it.

Prerequisites:

- Device supports Function Level Reset
- CXL device maintains Cache Coherency
- Hardware configuration support for Algorithm 1a described in [Section 14.3.1](#)
- If the device supports self-checking, it must escalate a fatal system error
- Device is permitted to log failing information

Test Steps:

1. Determine test runtime T, based on the amount of time available or allocated for this testing.
2. Host software sets up a Cache Coherency test for Algorithm 1a: Multiple Write Streaming.
3. If the device supports self-checking, enable it.
4. At a time between 1/3 and 2/3 of T and with at least 200 ms of test time remaining, the host initiates FLR by writing to the Initiate Function Level Reset bit.

Pass Criteria:

- System does not elevate a fatal system error, and no errors are logged

Fail Conditions:

- System error reported, logged failures exist

14.9.5 CXL Range Setup Time

Prerequisites:

- Device is CXL.mem capable
- Ability to monitor the device reset

Test Steps:

1. Reset the system, Monitor Reset until cleared.
2. Wait for 1 second.
3. Read Configuration Space for DUT, Offset 1Ch, Length 4 bytes.
4. Decode this into:

Bits	Variable
0:0	Memory_Info_Valid
1:1	Memory_Active

5. Verify:

Variable	Value	Condition
Memory_Info_Valid	1	
Memory_Active	1	Mem_HW_Init_Mode = 1

Pass Criteria:

- Test [14.8.2](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.9.6 FLR Memory

This test ensures that an FLR does not affect data in device-attached memory.

Prerequisites:

- Device is CXL.mem capable

Test Steps:

1. Write a known pattern to a known location within the HDM.
2. Host performs an FLR as defined in steps of Test [14.9.4](#).
3. Host reads the HDM's location.
4. Verify that the read data matches the previously written data.

Pass Criteria:

- HDM retains information after the FLR

Fail Conditions:

- HDM is reset

14.9.7 CXL_Reset Test

Prerequisites:

- CXL Reset Capable bit in the DVSEC CXL Capability register is set

Test Steps:

1. Determine test runtime T1 from DVSEC CXL Capability CXL Reset Timeout register.

2. Read and record value of following ROS register for step 6.

Error Capabilities and Control Register (Offset 14h)

Bits	Variable
5:0	First_Error_Pointer

Header Log Registers (Offset 18h)

Bits	Variable
511:0	Header Log

Note: Register contents may or may not be 0.

3. Set the following RWS registers to settings as per list and record the written values for step 6.

RWS Registers and settings:**Uncorrectable Error Mask Register (Offset 04h)**

Bits	Variable	Settings
11:0	Error Mask registers	Set to FFFh
16:14	Error Mask registers	Set to 111b

Uncorrectable Error Severity Register (Offset 08h)

Bits	Variable	Settings
11:0	Error Severity registers	Set to FFFh
16:14	Error Severity registers	Set to 111b

Correctable Error Mask Register (Offset 10h)

Bits	Variable	Settings
6:0	Error Mask registers	Clear to 00h

Error Capabilities and Control Register (Offset 14h)

Bits	Variable	Settings
13:13	Poison_Enabled	Set to 1

CXL Link Layer Capability Register (Offset 00h)

Bits	Variable	Settings
3:0	CXL Link Version Supported	Set to 2h
15:8	LLR Wrap Value Supported	Set to FFh

Note: Intention is to set the register to a nonzero value.

CXL Link Layer Control and Status Register (Offset 08h)

Bits	Variable	Settings
1:1	LL_Init_Stall	Set to 1

2:2	LL_Crd_Stall	Set to 1
-----	--------------	----------

CXL Link Layer Rx Credit Control Register (Offset 10h)

Bits	Variable	Settings
9:0	Cache Req Credits	Set to 3FFh
19:10	Cache Rsp Credits	Set to 3FFh
29:20	Cache Data Credits	Set to 3FFh
39:30	Mem Req_Rsp Credits	Set to 3FFh
49:40	Mem Data Credits	Set to 3FFh
59:50	BI Credits	Set to 3FFh

CXL Link Layer Ack Timer Control Register (Offset 28h)

Bits	Variable	Settings
7:0	Ack Force Threshold	Set to FFh
17:8	Ack or CRD Flush Retimer	Set to 1FFh

CXL Link Layer Defeature Register (Offset 30h)

Bits	Variable	Settings
0:0	MDH Disable	Set to 1

DVSEC CXL Control2 (Offset 10h)

Bits	Variable	Settings
4:4	Desired Volatile HDM State after Hot Reset	Set to 1 if DVSEC CXL Capability3 (Offset 38h) Bit 3 Volatile HDM State after Hot Reset – Configurability == 1

4. Set Initiate CXL Reset =1 in the DVSEC CXL Control2 register.
5. Wait for time T1.
6. Verify:
 - a. Confirm DVSEC Flex Bus Status2 CXL Reset complete is set.
 - b. ROS register values before and after CXL reset are matching.
 - c. RWS register values before and after CXL reset are matching.

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.9.8

Global Persistent Flush (GPF)

Test Equipment:

- Protocol Analyzer

14.9.8.1 Host and Switch Test

Test Steps:

1. Bring system to operating state.
2. Initiate Shut Down process.
3. Verify:
 - a. System sends a CXL GPF PM VDM Phase 1 request.
 - b. After receiving the response message from the device, the System sends a CXL GPF PM VDM Phase 2 request.
 - c. After receiving the response message, the Link transitions to the lowest-possible power state.

Pass Criteria:

- Verify that the required CXL GPF PM VDM Phase 1 request is sent
- Verify that the required CXL GPF PM VDM Phase 2 request is sent after the Phase 1 response
- Verify that the Link enters the lowest-possible power state

Fail Conditions:

- Verify Conditions failed

14.9.8.2 Device Test

Test Steps:

1. Ensure that the link between the system and the device is in an initialized state.
2. Initiate Shut Down process.
3. Verify:
 - a. Cache transactions are not initiated by the device after CXL GPF PM VDM.
 - b. Verify GPF Response message is sent by the device in Phase 1.
 - c. Verify GPF Response message is sent by the device in Phase 2.

Pass Criteria:

- Ensure that cache transactions are not initiated after the CXL GPF PM VDM in Phase 1
- Verify that the device sends a Response Message in Phase 1
- Check that the response message fields are correct
- Verify that the device sends a Response Message in Phase 2
- Verify that the Link enters the lowest-possible power state

Fail Conditions:

- Verify Conditions failed

14.9.9 Hot-Plug Test

Prerequisites:

- Device supports Hot-Plug

Test Steps:

1. Bring system to an operating state.
2. Initiate Hot-Plug remove.
3. Verify that the Hot-Plug remove process is complete.
4. Remove and then reinsert the device.
5. Initiate Hot-Plug add.
6. Verify that the Hot-Plug add process is complete.

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.9.10 Device to Host Cache Viral Injection

Prerequisites:

- Device is CXL.cache capable
- Device must support Compliance mode DOE
- Device must support Algorithm 1a

Test Steps:

1. Host software will set up the device and the host for Algorithm 1a: Multiple Write Streaming.
2. Host software decides the test runtime and runs the test for that period.
3. While a test is running, software will perform the following steps to the Device registers:
 - a. Write the Compliance Mode DOE Request register with the following values:
 - Request Code (Offset 08h) = 0Ch, Inject Viral
 - Protocol (Offset 0Ch) = 1, CXL.cache
4. Host software waits for Poll Compliance mode DOE response Viral Injection response until the following is returned from the device:
 - Request Code (Offset 08h) = 0Ch
 - Status (Offset 0Bh) = 00h

Pass Criteria:

- Host logs AER -Fatal Error

Fail Conditions:

- Host does not log AER -Fatal Error

14.9.11 Device to Host Mem Viral Injection

Prerequisites:

- Device is CXL.mem capable
- Device must support Compliance mode DOE
- Device must support Algorithm 1a

Test Steps:

1. Host software will set up the device and the host for Algorithm 1a: Multiple Write Streaming.
2. Host software decides the test runtime and runs the test for that period.
3. While a test is running, software will perform the following steps to the Device registers:
 - a. Write the Compliance Mode DOE Request register with the following values:
 - Request Code (Offset 08h) = 0Ch, Inject Viral
 - Protocol (Offset 0Ch) = 2, CXL.mem
4. Host software waits for Poll Compliance mode DOE response Viral Injection response until the following is returned from the device:
 - Request Code (Offset 08h) = 0Ch
 - Status (Offset 0Bh) = 00h

Pass Criteria:

- Host logs AER -Fatal Error

Fail Conditions:

- Host does not log AER -Fatal Error

14.10 Power Management Tests

14.10.1 Pkg-C Entry (Device Test)

This test case is optional if the device does not support generating PMReq() with memory LTR reporting.

This test case will check the following conditions:

- Device initiates PkgC entry, and reports appropriate LTR
- All PMReq() fields adhere to the CXL specification

Test Equipment:

- Protocol Analyzer (optional)

Prerequisites:

- Applicable for 68B Flit mode and 256B Flit mode
- Power Management is complete
- Credit Initialization is complete
- CXL link is up

Device Test Steps:

1. Host or Test Equipment maintains the link in an idle state, no CXL.cachemem requests are initiated by either the Host/Test Equipment or the DUT.
2. Host or Test equipment waits for the Link to enter CXL L1 Idle State.
3. Optionally, a Protocol Analyzer is used to inspect that the link enters L1 state, that the PMReq.Req is sent from the device, and that the host replies with PMReq.Rsp and PMReq.Go.

Pass Criteria:

- Link enters L1

Fail Conditions:

- Link enters L1 but PMReq.Req is missing
- LTR values in the PMReq.Req are invalid

14.10.2 Pkg-C Entry Reject (Device Test)

This test case is optional if the device does not support generating PMReq() with memory LTR reporting.

This test case will check the following conditions:

- Device initiates PkgC entry, and reports appropriate LTR
- All PMReq() fields adhere to the CXL specification
- DUT does not enter a low-power state when the Exerciser responds with Low LTR (processor busy condition)

Test Equipment:

- Exerciser

Prerequisites:

- Power Management is complete
- Credit Initialization is complete
- CXL link is up

Device Test Steps:

1. Host or Test Equipment maintains the link in an idle state, no CXL.cachemem requests are initiated by either the Host/Test Equipment or the DUT.
2. Exerciser waits for the PMReq.Req from the device.
3. Exerciser sends PMReq.Rsp that advertises Low LTR, indicating that the processor is busy.

Pass Criteria:

- Link does not enter L1

Fail Conditions:

- Device requests L1 entry
- LTR values in the PMReq.Req are invalid

14.10.3 Pkg-C Entry (Host Test)

This test case will check the following conditions:

- Host sends PMReq.Rsp when the device initiates PkgC entry, and that the host reports appropriate LTR
- All PMReq.Rsp() and PMReq.Go() fields adhere to the CXL specification

Test Equipment:

- Protocol Analyzer (optional)

Prerequisites:

- Power Management is complete
- Credit Initialization is complete
- CXL link is up

Host Test Steps:

1. Host and device maintain the link in an idle state, no CXL.cachemem requests are initiated by either the host or the device.
2. Test waits for the link to enter CXL L1 Idle State. If CXL L1 state is not entered, this test can be skipped.
3. Optionally, a Protocol Analyzer is used to inspect that the link enters L1 state, that the PMReq.Req is sent from the device, and that the host replies with PMReq.Rsp and PMReq.Go.

Pass Criteria:

- Link enters L1

Fail Conditions:

- Device sends PMReq.Req, but the host fails to respond with PMReq.Rsp and PMReq.Go
- LTR values in the PMReq.Rsp are invalid

14.11 Security

14.11.1 Component Measurement and Authentication

14.11.1.1 DOE CMA Instance

Prerequisites:

- DOE CMA is supported by at least one Function

Modes:

- CXL.io

Topologies:

- SHDA
- SHSW

- SHSW-FM

Test Steps:

1. Scan every function and read DOE CMA instances.

Pass Criteria:

- Each DOE CMA instance supports only DOE Discovery data object protocol, and CMA data object protocol

Fail Conditions:

- DOE discovery is not supported
- CMA data object is not supported

14.11.1.2 FLR while Processing DOE CMA Request

Prerequisites:

- DOE CMA is supported by at least one Function

Modes:

- CXL.io

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Send DOE CMA request.
2. Perform FLR to associated function (this should cancel the DOE request).
3. Attempt to read DOE CMA response.

Pass Criteria:

- Target Function response does not indicate that a DOE CMA response is available (the request should be canceled from the FLR)

Fail Conditions:

- Original DOE CMA request results in a response returned by the DOE CMA target function after FLR

14.11.1.3 OOB CMA while in Fundamental Reset

Prerequisites:

- OOB CMA is supported
- Platform or slot supports asserting Fundamental Reset under host software control

Note:

Known Good Host support for Fundamental Reset shall be on either a per-slot basis under Host-software control or hold all in Fundamental Reset during POST.

Modes:

- CXL.io
- OOB

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Assert Fundamental Reset on the device.
2. Perform authentication over OOB CMA.

Pass Criteria:

- Device successfully authenticates while the device is held in reset

Fail Conditions:

- Pass criteria is not met

14.11.1.4 OOB CMA while Function Gets FLR

Prerequisites:

- OOB CMA is supported
- Function 0 supports FLR

Modes:

- CXL.io
- OOB

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Clear Authenticated state over OOB with GET_VERSION request.
2. Host Issues FLR to Function 0 (Beginning a loop: Issue a single FLR with a delay until the FLR completes. Repeat.):
 - a. In parallel with the FLR loop, begin authentication with OOB (long CHALLENGE sequence beginning with GET_VERSION and calling required intermediate functions ending with CHALLENGE).
3. Host continues FLR (exit loop of FLRs once Authentication succeeds):
 - a. In parallel with FLR, verify CHALLENGE_AUTH succeeds over OOB.

Pass Criteria:

- Authentication successfully completes with FLR on device Function 0 during OOB authentication

Fail Conditions:

- OOB Authentication fails at any point using full authentication/negotiation sequence

14.11.1.5 OOB CMA during Conventional Reset

Prerequisites:

- OOB CMA supported
- Host issues Link_Disable on the device's root port to create the Conventional Reset condition

Modes:

- CXL.io
- OOB

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Host issues Link_Disable on the device's root port.
2. Perform authentication over OOB CMA (long sequence beginning with GET_VERSION, followed by intermediate requests as required and finishing with CHALLENGE).
3. Host enables Link on the device's root port.

Pass Criteria:

- Device successfully authenticates over OOB while the device's Link is in disabled state.

Fail Conditions:

- Pass criteria is not met

14.11.2 Link Integrity and Data Encryption CXL.io IDE

Use protocol analyzer to verify that link traffic is encrypted. Test is informational only if the Protocol analyzer is unavailable.

Link IDE tests are based on configuring IDE in a specific configuration, and then running a compliance test algorithm specified in Test [14.3.6.1.1](#).

Test Equipment:

- Protocol Analyzer

14.11.2.1 CXL.io Link IDE Streams Functional

Prerequisites:

-

Open: Prerequisites to be completed later.

Modes:

- CXL.io

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Establish Link IDE Streams on all links between the host and the DUT:
 - a. Disable aggregation.
 - b. Disable PCRC.
2. Start compliance test algorithm for CXL.io as defined in Test [14.3.6.1.1](#).

Pass Criteria:

- Self-checking compliance test reports that there are no errors
- CXL link remains up
- No errors are reported in the AER or IDE Status registers

Fail Conditions:

- Pass criteria is not met

14.11.2.2 CXL.io Link IDE Streams Aggregation

Prerequisites:

- Aggregation Supported bit is Set for both ports of each Link IDE Stream

Modes:

- CXL.io

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Establish Link IDE Streams on all links between the host and the DUT:
 - a. Enable aggregation.
 - b. Disable PCRC.

2. Start compliance test algorithm for CXL.io as defined in Test [14.3.6.1.1](#).
3. Cycle through the following Tx aggregation modes:
 - a. NPR/PR/CPL all set to 01b (up to 2).
 - b. NPR/PR/CPL all set to 10b (up to 4).
 - c. NPR/PR/CPL all set to 11b (up to 8).
 - d. NPR=01b, PR=10b, CPL=11b.

Pass Criteria:

- Self-checking compliance test reports that there are no errors
- CXL link remains up
- No errors are reported in the AER or IDE Status registers

Fail Conditions:

- Pass criteria is not met

14.11.2.3 CXL.io Link IDE Streams PCRC

Prerequisites:

- PCRC Supported bit is Set for both ports of each Link IDE Stream

Modes:

- CXL.io

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Establish Link IDE Streams on all links between the host and the DUT:
 - a. Disable aggregation.
 - b. Enable PCRC.
2. Start compliance test algorithm for CXL.io as defined in Test [14.3.6.1.1](#).

Pass Criteria:

- Self-checking compliance test reports that there are no errors
- CXL link remains up
- No errors are reported in the AER or IDE Status registers

Fail Conditions:

- Pass criteria is not met

14.11.2.4 CXL.io Selective IDE Stream Functional

Prerequisites:

- DOE CMA support

Modes:

- CXL.io

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Establish Selective IDE Streams on all links between the host and the DUT:
 - a. Disable aggregation.
 - b. Disable PCRC.
2. Start compliance test algorithm for CXL.io as defined in Test [14.3.6.1.1](#).

Pass Criteria:

- Self-checking compliance test reports that there are no errors
- CXL link remains up
- No errors are reported in the AER or IDE Status registers

Fail Conditions:

- Pass criteria is not met

14.11.2.5 CXL.io Selective IDE Streams Aggregation

Prerequisites:

- DOE CMA support
- Aggregation Support bit set for both ports of the Selective IDE stream

Modes:

- CXL.io

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Establish Selective IDE Streams on all links between the host and the DUT:
 - a. Enable aggregation.
 - b. Disable PCRC.

2. Start compliance test algorithm for CXL.io as defined in Test [14.3.6.1.1](#).
3. Cycle through the following Tx aggregation modes:
 - a. NPR/PR/CPL all set to 01b (up to 2).
 - b. NPR/PR/CPL all set to 10b (up to 4).
 - c. NPR/PR/CPL all set to 11b (up to 8).
 - d. NPR=01b, PR=10b, CPL=11b.

Pass Criteria:

- Self-checking compliance test reports that there are no errors
- CXL link remains up
- No errors are reported in the AER or IDE Status registers

Fail Conditions:

- Pass criteria is not met

14.11.2.6 CXL.io Selective IDE Streams PCRC

Prerequisites:

- DOE CMA support
- Aggregation Support bit is set for both ports of the Selective IDE stream

Modes:

- CXL.io

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Establish Selective IDE Streams on all links between the host and the DUT:
 - a. Disable aggregation.
 - b. Enable PCRC.
2. Start compliance test algorithm for CXL.io as defined in Test [14.3.6.1.1](#).

Pass Criteria:

- Self-checking compliance test reports that there are no errors
- CXL link remains up
- No errors are reported in the AER or IDE Status registers

Fail Conditions:

- Pass criteria is not met

14.11.3 CXL.cachemem IDE

14.11.3.1 CXL.cachemem IDE Capability (SHDA, SHSW)

This test determines whether the CXL device is capable of a secure IDE link, is configured to enable secure IDE links, and checks that the CXL IDE capability structure is read.

Prerequisites:

- Device must support CXL.cachemem IDE security
- Device must support Compliance Mode DOE and SPDM over DOE
- Host software has established a secure SPDM link to the device

Test Steps:

1. Read the CXL IDE Capability and Control structure (see [Section 8.2.4.21](#)).
2. Issue a CXL_QUERY request against the device.

Pass Criteria:

- Bit 0 of CXL IDE Capability register (CXL IDE Capable) is set
- CXL IDE Capability structure read from configuration space matches the Capability structure from CXL_QUERY RESP

Fail Conditions:

- Pass criteria is not met

14.11.3.2 Establish CXL.cachemem IDE (SHDA) in Standard 256B Flit Mode

This test verifies the device's ability to establish a CXL.cachemem IDE secure link between the downstream root port and an endpoint.

Prerequisites:

- Device supports Standard 256B Flit mode and 256B Flit mode is enabled
- Device must support CXL.cachemem IDE security
- Device must support Compliance Mode DOE and SPDM over DOE
- Host software has established a secure SPDM link to the device
- Test [14.11.3.1](#) passed

Test Steps:

1. Host software issues a CXL_GETKEY request to the endpoint and saves the Locally generated key as KEY1.
2. Host software issues a CXL_GETKEY request to the host Downstream Port, P, and saves the Locally generated key as KEY2.
3. Host software programs the endpoint keys with the following CXL_KEY_PROG requests to the Endpoint DOE mailbox. After each request, check the CXL_KP_ACK Status field for a nonzero value, and fail if found.
 - a. CXL_KEY_PROG (RxTxB=0, Use Default IV=1, KEY2).
 - b. CXL_KEY_PROG (RxTxB=1, Use Default IV=1, KEY1).

4. Host software programs the root port keys with the following CXL_KEY_PROG requests to the downstream root port. After each request, check the CXL_KP_ACK Status field for a nonzero value, and fail if found.
 - a. CXL_KEY_PROG (PortIndex = P, RxTxB=0, Use Default IV=1, KEY1).
 - b. CXL_KEY_PROG (PortIndex = P, RxTxB=1, Use Default IV=1, KEY2).
5. Host software activates the endpoint keys with the following KEY_SET_GO requests to the Endpoint DOE mailbox. After each request, check:
 - a. CXL_K_SET_GO (Skid mode, RxTxB=0).
 - b. CXL_K_SET_GO (Skid mode, RxTxB=1).
6. Host software activates the Root Downstream Port keys with the following KEY_SET_GO requests:
 - a. CXL_K_SET_GO (PortIndex= P, Skid mode, RxTxB=0).
 - b. CXL_K_SET_GO (PortIndex= P, Skid mode, RxTxB=1).

Pass Criteria:

- CXL.cachemem flits between the host and the endpoint are protected by IDE

Fail Conditions:

- CXL_KP_ACK Status field is set to a nonzero value

14.11.3.3 Establish CXL.cachemem IDE (SHSW)

This test verifies the device's ability to establish a CXL.cachemem IDE secure link between a switch's Downstream Port and the endpoint device.

Prerequisites:

- Device must support CXL.cachemem IDE security
- Device must support Compliance Mode DOE and SPDM over DOE
- Host software has established a secure SPDM link to the device
- Test [14.11.3.1](#) passed

Test Steps:

1. Host software issues a CXL_GETKEY request to the endpoint and saves the Locally generated key as KEY1.
2. Host software issues a CXL_GETKEY request to the switch USP (Port index =P, where P is the DSP that the EP is connected to) and saves the Locally generated key as KEY2.
3. Host software programs the endpoint keys with the following CXL_KEY_PROG requests to the Endpoint DOE mailbox. After each request, check the CXL_KP_ACK Status field for a nonzero value, and fail if found.
 - a. CXL_KEY_PROG (RxTxB=0, Use Default IV=1, KEY2).
 - b. CXL_KEY_PROG (RxTxB=1, Use Default IV=1, KEY1).
4. Host software programs the root port keys with the following CXL_KEY_PROG requests to the downstream root port. After each request, check the CXL_KP_ACK Status field for a nonzero value, and fail if found.
 - a. CXL_KEY_PROG (PortIndex = P, RxTxB=1, Use Default IV=1, KEY2).
 - b. CXL_KEY_PROG (PortIndex = P, RxTxB=0, Use Default IV=1, KEY2).

5. Host software activates the endpoint keys with the following KEY_SET_GO requests to the Endpoint DOE mailbox. After each request, check:
 - a. CXL_K_SET_GO (Skid mode, RxTxB=0).
 - b. CXL_K_SET_GO (Skid mode, RxTxB=1).
6. Host software activates the Root Downstream Port keys with the following KEY_SET_GO requests:
 - a. CXL_K_SET_GO (PortIndex=0, Skid mode, RxTxB=0).
 - b. CXL_K_SET_GO (PortIndex=0, Skid mode, RxTxB=1).

Open: Pass criteria/fail conditions are missing.

14.11.3.4 Establish CXL.cachemem IDE (SHDA) Latency-Optimized 256B Flit Mode

This test verifies the device's ability to establish a CXL.cachemem IDE secure link between the downstream root port and an endpoint.

Prerequisites:

- Device supports Latency-Optimized 256B Flit mode, and Latency-Optimized 256B Flit mode is enabled
- Device must support CXL.cachemem IDE security
- Device must support Compliance Mode DOE and SPDM over DOE
- Host software has established a secure SPDM link to the device
- Test [14.11.3.1](#) passed

Test Steps:

1. Host software issues a CXL_GETKEY request to the endpoint and saves the Locally generated key as KEY1.
2. Host software issues a CXL_GETKEY request to the host Downstream Port, P, and saves the Locally generated key as KEY2.
3. Host software programs the endpoint keys with the following CXL_KEY_PROG requests to the Endpoint DOE mailbox. After each request, check the CXL_KP_ACK Status field for a nonzero value, and fail if found.
 - a. CXL_KEY_PROG (RxTxB=0, Use Default IV=1, KEY2).
 - b. CXL_KEY_PROG (RxTxB=1, Use Default IV=1, KEY1).
4. Host software programs the root port keys with the following CXL_KEY_PROG requests to the downstream root port. After each request, check the CXL_KP_ACK Status field for a nonzero value, and fail if found.
 - a. CXL_KEY_PROG (PortIndex = P, RxTxB=0, Use Default IV=1, KEY1).
 - b. CXL_KEY_PROG (PortIndex = P, RxTxB=1, Use Default IV=1, KEY2).
5. Host software activates the endpoint keys with the following KEY_SET_GO requests to the Endpoint DOE mailbox. After each request, check:
 - a. CXL_K_SET_GO (Skid mode, RxTxB=0).
 - b. CXL_K_SET_GO (Skid mode, RxTxB=1).
6. Host software activates the Root Downstream Port keys with the following KEY_SET_GO requests:
 - a. CXL_K_SET_GO (PortIndex= P, Skid mode, RxTxB=0).

- b. CXL_K_SET_GO (PortIndex= P, Skid mode, RxTxB=1).

Pass Criteria:

- CXL.cachemem flits between the host and the endpoint are protected by IDE

Fail Conditions:

- CXL_KP_ACK Status field is set to a nonzero value

14.11.3.5 Establish CXL.cachemem IDE (SHDA) 68B Flit Mode

This test verifies the device's ability to establish a CXL.cachemem IDE secure link between the downstream root port and an endpoint.

Prerequisites:

- Device supports 68B Flit mode and 68B Flit mode is enabled
- Device must support CXL.cachemem IDE security
- Device must support Compliance Mode DOE and SPDM over DOE
- Host software has established a secure SPDM link to the device
- Test 14.11.3.x passed

Test Steps:

1. Host software issues a CXL_GETKEY request to the endpoint and saves the Locally generated key as KEY1.
2. Host software issues a CXL_GETKEY request to the host Downstream Port, P, and saves the Locally generated key as KEY2.
3. Host software programs the endpoint keys with the following CXL_KEY_PROG requests to the Endpoint DOE mailbox. After each request, check the CXL_KP_ACK Status field for a nonzero value, and fail if found.
 - a. CXL_KEY_PROG (RxTxB=0, Use Default IV=1, KEY2).
 - b. CXL_KEY_PROG (RxTxB=1, Use Default IV=1, KEY1).
4. Host software programs the root port keys with the following CXL_KEY_PROG requests to the downstream root port. After each request, check the CXL_KP_ACK Status field for a nonzero value, and fail if found.
 - a. CXL_KEY_PROG (PortIndex = P, RxTxB=0, Use Default IV=1, KEY1).
 - b. CXL_KEY_PROG (PortIndex = P, RxTxB=1, Use Default IV=1, KEY2).
5. Host software activates the endpoint keys with the following KEY_SET_GO requests to the Endpoint DOE mailbox. After each request, check:
 - a. CXL_K_SET_GO (Skid mode, RxTxB=0).
 - b. CXL_K_SET_GO (Skid mode, RxTxB=1).
6. Host software activates the Root Downstream Port keys with the following KEY_SET_GO requests:
 - a. CXL_K_SET_GO (PortIndex= P, Skid mode, RxTxB=0).
 - b. CXL_K_SET_GO (PortIndex= P, Skid mode, RxTxB=1).

Pass Criteria:

- CXL.cachemem flits between the host and the endpoint are protected by IDE

14.11.3.6 Locally Generate IV (SHDA)

Fail Conditions:

- CXL_KP_ACK Status field is set to a nonzero value

Prerequisites:

- Device must support CXL.cachemem IDE security
- Device must support Compliance Mode DOE and SPDM over DOE
- Host software has established a secure SPDM link to the device
- Test 14.11.3.1 passed
- Device supports Locally generated CXL.cachemem IV

Test Steps:

1. Host software issues a CXL_GETKEY request to the endpoint and saves the Locally generated key as KEY1, and the Initialization Vector as IV1.
2. Host software issues a CXL_GETKEY request to the host Downstream Port, P, and saves the Locally generated key as KEY2, and the Initialization Vector as IV2.
3. Host software programs the endpoint keys with the following CXL_KEY_PROG requests to the Endpoint DOE mailbox. After each request, check the CXL_KP_ACK Status field for a nonzero value, and fail if found.
 - a. CXL_KEY_PROG (RxTxB=0, Use Default IV=0, KEY2, IV2).
 - b. CXL_KEY_PROG (RxTxB=1, Use Default IV=0, KEY1, IV1).
4. Host software programs the root port keys with the following CXL_KEY_PROG requests to the downstream root port. After each request, check the CXL_KP_ACK Status field for a nonzero value, and fail if found.
 - a. CXL_KEY_PROG (PortIndex = P, RxTxB=0, Use Default IV=0, KEY1, IV1).
 - b. CXL_KEY_PROG (PortIndex = P, RxTxB=1, Use Default IV=0, KEY2, IV2).
5. Host software activates the endpoint keys with the following KEY_SET_GO requests to the Endpoint DOE mailbox. After each request, check:
 - a. CXL_K_SET_GO (Skid mode, RxTxB=0).
 - b. CXL_K_SET_GO (Skid mode, RxTxB=1).
6. Host software activates the Root Downstream Port keys with the following KEY_SET_GO requests:
 - a. CXL_K_SET_GO (PortIndex= P, Skid mode, RxTxB=0).
 - b. CXL_K_SET_GO (PortIndex= P, Skid mode, RxTxB=1).

Pass Criteria:

- No Failure is reported via the IDE Status register (see Section 8.2.4.22.3) or the CXL IDE Error Status register (see Section 8.2.4.22.4)
- CXL_KP_ACK response with Status=0

Fail Conditions:

- IDE Capabilities do not match
- CXL_KP_ACK response with Status!=0

14.11.3.7 Data Encryption – Decryption and Integrity Testing with Containment Mode for MAC Generation and Checking

Prerequisites:

- Host and Device are CXL.cache/CXL.mem/Both capable and enabled
- Containment mode must be enabled
- Host software has established a secure SPDM link to the device
- Test 14.11.3.2/3/4/5 passed (depends on Flit mode operation and topology)

Test Steps:

1. Enable the Containment mode of MAC generation.
2. Host Software should set up the device and the host for Algorithms 1a, 1b, and 2 to initiate traffic.
3. Enable Self-testing for checking validity of data.
4. Host software will control the test execution and test duration.

Pass Criteria:

- No Failure is reported via the IDE Status register (see [Section 8.2.4.22.3](#)) or the CXL IDE Error Status register (see [Section 8.2.4.22.4](#))

Fail Conditions:

- IDE reported failures

14.11.3.8 Data Encryption – Decryption and Integrity Testing with Skid Mode for MAC Generation and Checking

Prerequisites:

- Host and Device are CXL.cache/CXL.mem/Both capable and enabled
- Skid mode must be enabled
- Host software has established a secure SPDM link to the device
- Test 14.11.3.2/3/4/5 passed (depends on Flit mode operation and topology)

Test Steps:

1. Enable the Skid mode of MAC generation via the CXL Link Encryption Configuration registers.
2. Host Software should set up the device and the host for Algorithms 1a, 1b, and 2 to initiate traffic (see Test 14.3.6.1.2).
3. Enable Self-testing for checking validity of data.
4. Host software will control the test execution and test duration.

Pass Criteria:

- No Failure is reported via the IDE Status register (see [Section 8.2.4.22.3](#)) or the CXL IDE Error Status register (see [Section 8.2.4.22.4](#))

Fail Conditions:

- IDE reported failures

14.11.3.9 Key Refresh

Prerequisites:

- Device must support CXL.cachemem IDE security
- Device must support Compliance Mode DOE and SPDM over DOE
- Host software has established a secure SPDM link to the device
- Test 14.11.3.2/3/4/5 passed (depends on Flit mode operation and topology)

Topologies:

- SHDA

Test Steps:

1. See Test 14.11.3.2/3/4/5 (depends on Flit mode operation and topology) to set up an encrypted link between the host and the device and the initial KEY_EXCHANGE.
2. Host software sets up the Device for Algorithms 1a, 1b, and 2 to initiate traffic (see Section 14.3.6.1).
3. Enable Self-testing for checking validity of data.
4. Host software controls the test execution and test duration.
5. Move IDE to insecure state and reconfigure keys with the following steps:
 - a. Host Software/CIKMA initiates "CXL_K_SET_STOP" to Tx and Rx of both ports for transition to IDE insecure state.
 - b. If CXL.cachemem IDE Key Generation Capable=1 in QUERY_RSP, CIKMA will issue the following:
 - i. Host Software/CIKMA initiates "CXL_GETKEY" to get the locally generated keys from ports.
 - c. Host Software/CIKMA initiates "CXL_KEY_PROG" for setting up new set of Keys for Tx and Rx of ports.
 - d. Host/CIKMA initiates "CXL_K_SET_GO" to Rx, waits for successful response, and then initiates "CXL_K_SET_GO" to Tx ports to indicate/prepare for start of KEY_EXCHANGE.
6. Initiate the next set of traffic by repeating steps 1, 2, and 3.

Pass Criteria:

- No Failure is reported via the IDE Status register (see Section 8.2.4.22.3) or the CXL IDE Error Status register (see Section 8.2.4.22.4)
- CXL_KP_ACK response with Status=0

Fail Conditions:

- IDE reported failures
- CXL_KP_ACK response with Status!=0
- CXL_K_GOSTOP_ACK is not received within the specified timeout period

14.11.3.10 Early MAC Termination

Prerequisites:

- Host and Device are CXL.cache/CXL.mem/Both capable and enabled
- Skid mode must be enabled

- Host software has established a secure SPDM link to the device
- Test [14.11.3.2/3/4/5](#) passed (depends on Flit mode operation and topology)

Test Steps:

1. Host Software sets up the host and the device to initiate a number of protocol flits in the current MAC epoch that is less than Aggregation_Flit_Count via Algorithms 1a, 1b, and 2 (see Test [14.3.6.1.2](#) and Test [14.3.6.1.4](#)).
2. Device will send a TMAC LLCTRL flit.
3. Device should send “TruncationDelay” number of IDE.Idle flits.
4. Host software controls the test execution and test duration.

Pass Criteria:

- No “Truncated MAC flit check error” error is reported in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#))
- Configured number of IDLE flits is observed

Fail Conditions:

- Error is logged in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#))
- Configured number of IDE.Idle LLCTRL flits is not observed

[14.11.3.11 Error Handling](#)

[14.11.3.11.1 Invalid Keys \(Host and Device Keys Are Not Synced\)](#)

Prerequisites:

- Host and Device are CXL.cache/CXL.mem/Both capable and enabled
- Skid mode must be enabled
- Host software has established a secure SPDM link to the device
- Test [14.11.3.2/3/4/5](#) passed (depends on Flit mode operation and topology)

Test Steps:

1. Set up the device side for an invalid key via test steps mentioned in Test [14.11.3.2/3/4/5](#) with an invalid combination of KEY1 and KEY2 for the TX and RX Ports for the device.
2. Host Software sets up the device to initiate traffic via Algorithms 1a, 1b, and 2 (see [Section 14.3.6.1](#)).
3. Stop the test execution as soon as the pass criteria is achieved.

Pass Criteria:

- “Integrity Failure” Error is reported in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#))

Fail Conditions:

- No error is reported in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#))

[14.11.3.11.2 Inject MAC Delay](#)

This test checks whether the MAC for the previous epoch is received within the first 5 flits of MAC epoch.

Prerequisites:

- Host and Device are CXL.cache/CXL.mem/Both capable and enabled
- Skid mode must be enabled
- Host software has established a secure SPDM link to the device
- Test [14.11.3.2/3/4/5](#) passed (depends on Flit mode operation and topology)

Test Steps:

1. Write Compliance mode DOE with the "Inject MAC Delay" with following:

Table 14-5. Inject MAC Delay Setup

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	0Bh, Delay MAC
9h	1	Version	
Ah	2	Reserved	
Ch	1	<ul style="list-style-type: none"> • 00h = Disable • 01h = Enable 	01h
Dh	1	Mode <ul style="list-style-type: none"> • 00h = CXL.io • 01h = CXL.cache • 02h = CXL.mem 	01h or 02h

2. Host Software sets up the device to initiate traffic via Algorithms 1a, 1b, and 2 (see [Section 14.3.6.1](#)).
3. Stop test execution as soon as the pass criteria is achieved.

Pass Criteria:

- Link exits secure mode
- MAC Header not received when not expected error (Error code 100h) reported in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#))

Fail Conditions:

- Error is not logged in the IDE Error Status register (see [Section 8.2.4.22.4](#))

14.11.3.11.3 Inject Unexpected MAC**Prerequisites:**

- Host and Device are CXL.cache/CXL.mem/Both capable and enabled
- Skid mode must be enabled
- Host software has established a secure SPDM link to the device
- Test [14.11.3.2/3/4/5](#) passed (depends on Flit mode operation and topology)

Test Steps:

1. Write Compliance mode DOE with the "Inject Unexpected MAC" with following:

Table 14-6. Inject Unexpected MAC Setup

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	0Bh, Unexpected MAC
9h	1	Version	
Ah	2	Reserved	
Ch	1	<ul style="list-style-type: none"> • 00h = Disable • 01h = Insert message • 02h = Delete message 	02h
Dh	1	Mode <ul style="list-style-type: none"> • 00h = CXL.io • 01h = CXL.cache • 02h = CXL.mem 	01h or 02h

2. Host Software sets up the device to initiate traffic via Algorithms 1a, 1b, and 2 (see [Section 14.3.6.1](#)).
3. Stop test execution as soon as the pass criteria is achieved.

Pass Criteria:

- “MAC header received when not expected” error (Error code 0011h) reported in the CXL IDE Error Status register (see [Section 8.2.4.22.4](#))

Fail Conditions:

- Error is not logged in the CXL IDE Error Status register

14.11.3.11.4 Invalid CXL Query Request (SHDA)**Prerequisites:**

- Device must support CXL.cachemem IDE security
- Device must support Compliance Mode DOE and SPDM over DOE
- Host software has established a secure SPDM link to the device
- Test [14.11.3.1](#) passed

Test Steps:

1. Set up an encrypted link between the host and the device as per Test [14.11.3.2/3/4/5](#) (depends on Flit mode operation and topology).
2. Host software sets up the Device for Algorithms 1a, 1b, and 2 to initiate traffic (see [Section 14.3.6.1](#)).
3. Enable Self-testing for checking validity of data.
4. Host software controls the test execution and test duration.
5. Initiate the next set of traffic by repeating steps 1, 2, and 3.
6. Host software (CIKMA) sends a CXL QUERY Request except Protocol ID will use a nonzero value, thereby making the request invalid.

Pass Criteria:

- Response is not generated

- Invalid request is silently dropped
- Active IDE data stream should continue passing valid data/traffic
- No Failure is reported via the IDE Status register (see [Section 8.2.4.22.3](#)) or the CXL IDE Error Status register (see [Section 8.2.4.22.4](#))

Fail Conditions:

- IDE reported failures

14.11.3.11.5 Invalid CXL_KEY_PROG Request (SHDA)**Prerequisites:**

- Device must support CXL.cachemem IDE security
- Device must support Compliance Mode DOE and SPDM over DOE
- Host software has established a secure SPDM link to the device
- Test [14.11.3.1](#) passed

Test Steps:

1. Set up an encrypted link between the host and the device as per Test [14.11.3.2/3/4/5](#) (depends on Flit mode operation and topology).
2. Host software sets up the Device for Algorithms 1a, 1b, and 2 to initiate traffic (see [Section 14.3.6.1](#)).
3. Enable Self-testing for checking validity of data.
4. Host software controls the test execution and test duration.
5. Initiate the next set of traffic by repeating steps 1, 2, and 3.
6. Host software (CIKMA) sends a CXL_KEY_PROG Request except Stream ID will use a nonzero value, thereby making the request invalid.

Pass Criteria:

- Key and IV are not updated
- Device returns CXL_KP_ACK with Status=04h
- IDE stream of data should continue
- No Failure is reported via the IDE Status register (see [Section 8.2.4.22.3](#)) or the CXL IDE Error Status register (see [Section 8.2.4.22.4](#))

Fail Conditions:

- Key or IV are updated
- Successful status is returned

14.11.3.11.6 Invalid SPDM Session ID on CXL_IDE_KM for CXL_KEY_PROG Request (SHDA)

This test verifies the device's error response after the device receives CIKMA Invalid Messages for IDE.

Prerequisites:

- Device must support CXL.cachemem IDE security
- Device must support Compliance Mode DOE and SPDM over DOE
- Host software has established a secure SPDM link to the device

- Test [14.11.3.1](#) passed

Test Steps:

1. Set up an encrypted link between the host and the device as per Test [14.11.3.2/3/4/5](#) (depends on Flit mode operation and topology).
2. Host software sets up the Device for Algorithms 1a, 1b, and 2 to initiate traffic (see [Section 14.3.6.1](#)).
3. Enable Self-testing for checking validity of data.
4. Host software controls the test execution and test duration.
5. Initiate the next set of traffic by repeating steps 1, 2, and 3.
6. Host software (CIKMA) sends a CXL_KEY_PROG request with CXL_IDE_KM message header with an incorrect SPDM Session ID.

Pass Criteria:

- Response is not generated
- Invalid request is silently dropped
- Active IDE data stream should continue passing valid data/traffic
- No Failure is reported via the IDE Status register (see [Section 8.2.4.22.3](#)) or the CXL IDE Error Status register (see [Section 8.2.4.22.4](#))

Fail Conditions:

- IDE reported failures

[14.11.3.11.7 Invalid Key/IV Pair \(SHDA, SHSW\)](#)

This test verifies that the Device detects an invalid key state and does not initiate an IDE stream in response.

Prerequisites:

- Device must support CXL.cachemem IDE security
- Device must support Compliance Mode DOE and SPDM over DOE
- Host software has established a secure SPDM link to the device
- Test [14.11.3.1](#) passed
- Device supports both Locally generated CXL.cachemem IDE Key and Locally generated CXL.cachemem IV

Test Steps:

1. Host software issues a CXL_GETKEY request to the endpoint and saves the Locally generated key as KEY1.
2. Host software issues a CXL_GETKEY request to the endpoint and saves the Locally generated Initialization Vector as IV1.
3. Host software issues a CXL_GETKEY request to the host Downstream Port, P, and saves the Locally generated key as KEY2.
4. Host software issues a CXL_GETKEY request to the host Downstream Port, P, and saves the Locally generated Initialization Vector as IV2.
5. Host software programs the endpoint keys with the following CXL_KEY_PROG requests to the Endpoint DOE mailbox. After each request, check the CXL_KP_ACK Status field for a nonzero value, and fail if found.

- a. CXL_KEY_PROG (RxTxB=0, Use Default IV=0, KEY2, IV2).
- b. CXL_KEY_PROG (RxTxB=1, Use Default IV=0, KEY1, IV1).
6. Host software programs the root port keys with the following CXL_KEY_PROG requests to the downstream root port. After each request, check the CXL_KP_ACK Status field for a nonzero value, and fail if found.
 - a. CXL_KEY_PROG (PortIndex = P, RxTxB=0, Use Default IV=0, KEY1, IV1).
 - b. CXL_KEY_PROG (PortIndex = P, RxTxB=1, Use Default IV=0, KEY2, IV2).
7. EP and DSP should return ACK with Status=08h.

Pass Criteria:

- EP and DSP return CXL_KP_ACK with Status=08h at step 5

Fail Conditions:

- IDE Capabilities do not match
- Key and IV mismatch not detected at step 5

14.11.4 Certificate Format/Certificate Chain

Prerequisites:

- Certificate requirements for this test are drawn from the following external documents: SPDM 1.1, CMA ECN, PCIE-IDE ECN

Test Steps:

1. Receiver sends GET_DIGESTS to DUT.
2. Receiver verifies that the DUT responds with DIGESTS response.
3. Receiver records which Certificate Chains are populated, and then performs the following for each populated slot:
 - a. Receiver sends a series of GET_CERTIFICATE requests to read the entire certificate chain.
 - b. Receiver verifies that the DUT provides a CERTIFICATE response to each request.
4. Test Software parses Certificate Chain and verifies:
 - a. Certificate Version (should be version 2 or 3).
 - b. Serial Number.
 - c. CA Distinguished Name.
 - d. Subject Name.
 - e. Certificate Validity Dates.
 - f. Subject Public key info.
 - g. Subject Alternate Name (if implemented).
 - h. All Certificates use X.509 v3 format.
 - i. All Certificates use DER / ANS.1.
 - j. All Certificates use ECDSA / NIST* P-256.
 - k. All certificates use SHA-256 or SHA-384.
 - l. Leaf nodes do not exceed MaxLeafCertSize.

- m. Intermediate nodes do not exceed MaxIntermediateCertSize.
- n. Textual ASN.1 objects contained in certificates use UTF8String and do not exceed 64 bytes.
- o. Common names appear in every certificate.
- p. Common names use format "CXL:<vid><pid>" with VID in uppercase HEX.
- q. If VID and/or PID appears, they are consistent within a certificate chain.
- r. Organization name appears in Root Certificate in human-readable format.

Open: Pass criteria/fail conditions are missing.

14.11.5 Security RAS

14.11.5.1 CXL.io Poison Inject from Device

Prerequisites:

- CXL device must support Algorithm 1a
- CXL device must support Link Layer Error Injection capabilities

Test Steps:

1. Set up the device for Multiple Write streaming:
 - a. Write a pattern {64{8'hFF}} to cache-aligned Address A1.
 - b. Write a Compliance mode DOE to inject poison:

Table 14-7. CXL.io Poison Inject from Device: I/O Poison Injection Request

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	6, Poison Injection
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	0

- c. Write Compliance mode DOE with the following request:

Table 14-8. CXL.io Poison Inject from Device: Multi-Write Streaming Request (Sheet 1 of 2)

Data Object Byte Offset	Length in Bytes	Description	Value
00h	8	Standard DOE Request Header	
08h	1	Request Code	3, Multiple Write Streaming
09h	1	Version	2
0Ah	2	Reserved	
0Ch	1	Protocol	1
0Dh	1	Virtual Address	0
0Eh	1	Self-checking	0
0Fh	1	Verify Read Semantics	0
10h	1	Num Increments	0

Table 14-8. CXL.io Poison Inject from Device: Multi-Write Streaming Request (Sheet 2 of 2)

Data Object Byte Offset	Length in Bytes	Description	Value
11h	1	Num Sets	0
12h	1	Num Loops	1
13h	1	Reserved	
14h	8	Start Address	A1
1Ch	8	Write Address	0
24h	8	WriteBackAddress	A2 (Must be distinct from A1)
2Ch	8	Byte Mask	FFFF FFFF FFFF FFFFh
34h	4	Address Increment	0
38h	4	Set Offset	0
3Ch	4	Pattern "P"	AAh
40h	4	Increment Pattern "B"	0

Pass Criteria:

- Receiver (host) logs poisoned received error
- CXL.io IDE link state remains secured

Fail Conditions:

- Pass criteria is not met

14.11.5.2 CXL.cache Poison Inject from Device**Prerequisites:**

- Device is CXL.cache capable
- CXL device must support Algorithm 1a
- CXL device must support Link Layer Error Injection capabilities

Test Steps:

1. Set up the device for Multiple Write streaming:
 - a. Write a pattern {64{8'hFF}} to cache-aligned Address A1.
 - b. Write a Compliance mode DOE to inject poison:

Table 14-9. CXL.cache Poison Inject from Device: Cache Poison Injection Request

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	6, Poison Injection
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	0

- c. Write Compliance mode DOE with the following request:

Table 14-10. CXL.cache Poison Inject from Device: Multi-Write Streaming Request

Data Object Byte Offset	Length in Bytes	Description	Value
00h	8	Standard DOE Request Header	
08h	1	Request Code	3, Multiple Write Streaming
09h	1	Version	2
0Ah	2	Reserved	
0Ch	1	Protocol	2
0Dh	1	Virtual Address	0
0Eh	1	Self-checking	0
0Fh	1	Verify Read Semantics	0
10h	1	Num Increments	0
11h	1	Num Sets	0
12h	1	Num Loops	1
13h	1	Reserved	
14h	8	Start Address	A1
1Ch	8	Write Address	0
24h	8	WriteBackAddress	A2 (Must be distinct from A1)
2Ch	8	Byte Mask	FFFF FFFF FFFF FFFFh
34h	4	Address Increment	0
38h	4	Set Offset	0
3Ch	4	Pattern "P"	AAh
40h	4	Increment Pattern "B"	0

Pass Criteria:

- Receiver (host) logs poisoned received error
- CXL.io IDE link state remains secured

Fail Conditions:

- Pass criteria is not met

14.11.5.3 CXL.cache CRC Inject from Device**Prerequisites:**

- CXL device must support Algorithm 1a
- CXL device must support Link Layer Error Injection capabilities

Test Steps:

1. Set up the device for Multiple Write streaming:
 - a. Write a pattern {64{8'hFF}} to cache-aligned Address A1.
 - b. Write a Compliance mode DOE to inject CRC errors:

Table 14-11. CXL.cache CRC Inject from Device: Cache CRC Injection Request

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	7, CRC Injection
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	2
Dh	1	Num Bits Flipped	1
Eh	1	Num Flits Injected	1

c. Write Compliance mode DOE with the following request:

Table 14-12. CXL.cache CRC Inject from Device: Multi-Write Streaming Request

Data Object Byte Offset	Length in Bytes	Description	Value
00h	8	Standard DOE Request Header	
08h	1	Request Code	3, Multiple Write Streaming
09h	1	Version	2
0Ah	2	Reserved	
0Ch	1	Protocol	2
0Dh	1	Virtual Address	0
0Eh	1	Self-checking	0
0Fh	1	Verify Read Semantics	0
10h	1	Num Increments	0
11h	1	Num Sets	0
12h	1	Num Loops	1
13h	1	Reserved	
14h	8	Start Address	A1
1Ch	8	Write Address	0
24h	8	WriteBackAddress	A2 (Must be distinct from A1)
2Ch	8	Byte Mask	FFFF FFFF FFFF FFFFh
34h	4	Address Increment	0
38h	4	Set Offset	0
3Ch	4	Pattern "P"	AAh
40h	4	Increment Pattern "B"	0

Pass Criteria:

- Receiver (host) logs poisoned received error
- CXL.cache IDE link state remains secured

Fail Conditions:

- Pass criteria is not met

14.11.5.4 CXL.mem Poison Injection

Prerequisites:

- Device is CXL.mem capable
- CXL device must support Link Layer Error Injection capabilities

Test Steps:

1. Select a Memory target range on the Device Physical Address (DPA) that belongs to the DUT.
2. Translate the DPA to a Host Physical Address (HPA).
3. Perform continuous read/write operations on the HPA.
4. Write a Compliance mode DOE to inject Poison errors:

Table 14-13. CXL.mem Poison Injection: Mem-Poison Injection Request

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	6, Poison Injection
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	3

Pass Criteria:

- Receiver (host) logs poisoned received error
- CXL IDE link state remains secured

Fail Conditions:

- Pass criteria is not met

14.11.5.5 CXL.mem CRC Injection

Prerequisites:

- Device is CXL.mem capable
- CXL device must support Link Layer Error Injection capabilities

Test Steps:

1. Select a Memory target range on the Device Physical Address (DPA) that belongs to the DUT.
2. Translate the DPA to a Host Physical Address (HPA).
3. Perform continuous read/write operations on the HPA.
4. Write a compliance mode DOE to inject CRC errors:

Table 14-14. CXL.mem CRC Injection: MEM CRC Injection Request

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	7, CRC Injection
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	3
Dh	1	Num Bits Flipped	1
Eh	1	Num Flits Injected	1

Pass Criteria:

- Receiver (host) logs poisoned received error
- CXL IDE link state remains secured

Fail Conditions:

- Pass criteria is not met

14.11.5.6 Flow Control Injection**Prerequisites:**

- CXL device must support Algorithm 1a
- CXL device must support Link Layer Error Injection capabilities

Test Steps:

1. Set up the device for Multiple Write streaming:
 - a. Write a pattern {64{8'hFF}} to cache-aligned Address A1.
 - b. Write a Compliance mode DOE to inject poison:

Table 14-15. Flow Control Injection: Flow Control Injection Request

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	8, Flow Control Injection
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	0

- c. Write Compliance mode DOE with the following request:

Table 14-16. Flow Control Injection: Multi-Write Streaming Request

Data Object Byte Offset	Length in Bytes	Description	Value
00h	8	Standard DOE Request Header	
08h	1	Request Code	3, Multiple Write Streaming
09h	1	Version	2
0Ah	2	Reserved	
0Ch	1	Protocol	1
0Dh	1	Virtual Address	0
0Eh	1	Self-checking	0
0Fh	1	Verify Read Semantics	0
10h	1	Num Increments	0
11h	1	Num Sets	0
12h	1	Num Loops	1
13h	1	Reserved	
14h	8	Start Address	A1
1Ch	8	Write Address	0
24h	8	WriteBackAddress	A2 (Must be distinct from A1)
2Ch	8	Byte Mask	FFFF FFFF FFFF FFFFh
34h	4	Address Increment	0
38h	4	Set Offset	0
3Ch	4	Pattern "P"	AAh
40h	4	Increment Pattern "B"	0

Pass Criteria:

- Receiver (host) logs poisoned received error
- CXL.io IDE link state remains secured

Fail Conditions:

- Pass criteria is not met

14.11.5.7 Unexpected Completion Injection**Prerequisites:**

- CXL device must support Algorithm 1a
- CXL device must support Link Layer Error Injection capabilities

Test Steps:

1. Set up the device for Multiple Write streaming:
 - a. Write a pattern {64{8'hFF}} to cache-aligned Address A1.
 - b. Write a Compliance mode DOE to inject an unexpected completion error:

Table 14-17. Unexpected Completion Injection: Unexpected Completion Injection Request

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	Ah, Unexpected Completion Injection
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	0

c. Write Compliance mode DOE with the following request:

Table 14-18. Unexpected Completion Injection: Multi-Write Streaming Request

Data Object Byte Offset	Length in Bytes	Description	Value
00h	8	Standard DOE Request Header	
08h	1	Request Code	3, Multiple Write Streaming
09h	1	Version	2
0Ah	2	Reserved	
0Ch	1	Protocol	1
0Dh	1	Virtual Address	0
0Eh	1	Self-checking	0
0Fh	1	Verify Read Semantics	0
10h	1	Num Increments	0
11h	1	Num Sets	0
12h	1	Num Loops	1
13h	1	Reserved	
14h	8	Start Address	A1
1Ch	8	Write Address	0
24h	8	WriteBackAddress	A2 (Must be distinct from A1)
2Ch	8	Byte Mask	FFFF FFFF FFFF FFFFh
34h	4	Address Increment	0
38h	4	Set Offset	0
3Ch	4	Pattern "P"	AAh
40h	4	Increment Pattern "B"	0

Pass Criteria:

- Receiver (host) logs poisoned received error
- CXL.io IDE link state remains secured

Fail Conditions:

- Pass criteria is not met

14.11.5.8 Completion Timeout Injection

Prerequisites:

- CXL device must support Algorithm 1a
- CXL device must support Link Layer Error Injection capabilities

Test Steps:

1. Set up the device for Multiple Write streaming:
 - a. Write a pattern {64{8'hFF}} to cache-aligned Address A1.
 - b. Write a Compliance mode DOE to inject an unexpected completion error:

Table 14-19. Completion Timeout Injection: Completion Timeout Injection Request

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	Ah, Completion Timeout Injection
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	0

- c. Write Compliance mode DOE with the following request:

Table 14-20. Completion Timeout Injection: Multi-Write Streaming Request (Sheet 1 of 2)

Data Object Byte Offset	Length in Bytes	Description	Value
00h	8	Standard DOE Request Header	
08h	1	Request Code	3, Multiple Write Streaming
09h	1	Version	2
0Ah	2	Reserved	
0Ch	1	Protocol	1
0Dh	1	Virtual Address	0
0Eh	1	Self-checking	0
0Fh	1	Verify Read Semantics	0
10h	1	Num Increments	0
11h	1	Num Sets	0
12h	1	Num Loops	1
13h	1	Reserved	
14h	8	Start Address	A1
1Ch	8	Write Address	0
24h	8	WriteBackAddress	A2 (Must be distinct from A1)
2Ch	8	Byte Mask	FFFF FFFF FFFF FFFFh
34h	4	Address Increment	0

Table 14-20. Completion Timeout Injection: Multi-Write Streaming Request (Sheet 2 of 2)

Data Object Byte Offset	Length in Bytes	Description	Value
38h	4	Set Offset	0
3Ch	4	Pattern "P"	AAh
40h	4	Increment Pattern "B"	0

Pass Criteria:

- CXL.cache IDE link state remains secure
- Host Receiver logs link error

Fail Conditions:

- Pass criteria is not met

14.11.5.9 Memory Error Injection and Logging**Prerequisites:**

- CXL device must support Algorithm 1a
- CXL device must support Link Layer Error Injection capabilities
- CXL Type 2 device or Type 3 device must support Memory Logging and Reporting
- CXL device must support Error Injection for Memory Logging and Reporting

Test Steps:

1. Set up the device for Multiple Write streaming:
 - a. Write a pattern {64{8'hFF}} to cache-aligned Address A1.
 - b. Write a Compliance mode DOE to inject poison:

Table 14-21. Memory Error Injection and Logging: Poison Injection Request

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	6, Poison Injection
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	3

- c. Write Compliance mode DOE with the following request:

Table 14-22. Memory Error Injection and Logging: Multi-Write Streaming Request (Sheet 1 of 2)

Data Object Byte Offset	Length in Bytes	Description	Value
00h	8	Standard DOE Request Header	
08h	1	Request Code	3, Multiple Write Streaming
09h	1	Version	2
0Ah	2	Reserved	

Table 14-22. Memory Error Injection and Logging: Multi-Write Streaming Request (Sheet 2 of 2)

Data Object Byte Offset	Length in Bytes	Description	Value
0Ch	1	Protocol	3
0Dh	1	Virtual Address	0
0Eh	1	Self-checking	0
0Fh	1	Verify Read Semantics	0
10h	1	Num Increments	0
11h	1	Num Sets	0
12h	1	Num Loops	1
13h	1	Reserved	
14h	8	Start Address	A1
1Ch	8	Write Address	0
24h	8	WriteBackAddress	A2 (Must be distinct from A1)
2Ch	8	Byte Mask	FFFF FFFF FFFF FFFFh
34h	4	Address Increment	0
38h	4	Set Offset	0
3Ch	4	Pattern "P"	AAh
40h	4	Increment Pattern "B"	0

Pass Criteria:

- Receiver (host) logs error into DOE and error is signaled to the host
- CXL.cache IDE link state remains secured

Fail Conditions:

- Pass criteria is not met

14.11.5.10 CXL.io Viral Inject from Device**Prerequisites:**

- CXL device must support Algorithm 1a
- CXL device must support Link Layer Error Injection capabilities

Test Steps:

1. Set up the device for Multiple Write streaming:
 - a. Write a pattern {64{8'hFF}} to cache-aligned Address A1.
 - b. Write a Compliance mode DOE to inject poison viral.

Table 14-23. CXL.io Viral Inject from Device: I/O Viral Injection Request (Sheet 1 of 2)

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	Ch, Viral Injection

Table 14-23. CXL.io Viral Inject from Device: I/O Viral Injection Request (Sheet 2 of 2)

Data Object Byte Offset	Length in Bytes	Description	Value
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	0

c. Write Compliance mode DOE with the following request:

Table 14-24. CXL.io Viral Inject from Device: Multi-Write Streaming Request

Data Object Byte Offset	Length in Bytes	Description	Value
00h	8	Standard DOE Request Header	
08h	1	Request Code	3, Multiple Write Streaming
09h	1	Version	2
0Ah	2	Reserved	
0Ch	1	Protocol	1 CXL.io
0Dh	1	Virtual Address	0
0Eh	1	Self-checking	0
0Fh	1	Verify Read Semantics	0
10h	1	Num Increments	0
11h	1	Num Sets	0
12h	1	Num Loops	1
13h	1	Reserved	
14h	8	Start Address	A1
1Ch	8	Write Address	0
24h	8	WriteBackAddress	A2 (Must be distinct from A1)
2Ch	8	Byte Mask	FFFF FFFF FFFF FFFFh
34h	4	Address Increment	0

Pass Criteria:

- Receiver (host) logs poisoned received error
- CXL.io IDE link state remains secured

Fail Conditions:

- Pass criteria is not met

14.11.5.11 CXL.cache Viral Inject from Device

Prerequisites:

- Device is CXL.cache capable
- CXL device must support Algorithm 1a
- CXL device must support Link Layer Error Injection capabilities

Test Steps:

1. Set up the device for Multiple Write streaming:
 - a. Write a pattern {64{8'hFF}} to cache-aligned Address A1.
 - b. Write a Compliance mode DOE to inject poison viral:

Table 14-25. CXL.cache Viral Inject from Device: Cache Viral Injection Request

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	Ch, Viral Injection
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	2 CXL.cache.

- c. Write Compliance mode DOE with the following request:

Table 14-26. CXL.cache Viral Inject from Device: Multi-Write Streaming Request

Data Object Byte Offset	Length in Bytes	Description	Value
00h	8	Standard DOE Request Header	
08h	1	Request Code	3, Multiple Write Streaming
09h	1	Version	2
0Ah	2	Reserved	
0Ch	1	Protocol	2 CXL.cache
0Dh	1	Virtual Address	0
0Eh	1	Self-checking	0
0Fh	1	Verify Read Semantics	0
10h	1	Num Increments	0
11h	1	Num Sets	0
12h	1	Num Loops	1
13h	1	Reserved	
14h	8	Start Address	A1
1Ch	8	Write Address	0
24h	8	WriteBackAddress	A2 (Must be distinct from A1)
2Ch	8	Byte Mask	FFFF FFFF FFFF FFFFh
34h	4	Address Increment	0

Pass Criteria:

- Receiver (host) logs poisoned received error
- CXL.cache IDE link state remains secured

Fail Conditions:

- Pass criteria is not met

14.11.6 Security Protocol and Data Model

14.11.6.1 SPDM GET_VERSION

Prerequisites:

- SPDM version 1.0 or higher
- DOE for CMA (should include DOE Discovery Data object protocol and the CMA data object protocol)
- CMA over MCTP/SMBus for out-of-band validation should function while device is held in fundamental reset
- A fundamental link reset shall not impact the CMA connection over out-of-band
- Compliance Software must keep track of all transactions (per SPDM spec, Table 21a: Request ordering and message transcript computation rules for M1/M2) to complete the CHALLENGE request after the sequence of test assertions are complete

Modes:

- CXL.io
- OOB CMA

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Issue GET_VERSION over SPDM to target the device over DOE/CMA using HOST capabilities for SPDM version 1.0.
2. Optional OOB: Issue the Discovery command to gather version information over out-of-band.
3. Validate that the VERSION response matches the host's capabilities and meets the minimum SPDM version 1.0 requirements.
4. Optional OOB: Valid JSON file is returned from the Discovery command for version.
5. Optional: Repeat for next version of SPDM if the Responder VERSION response includes a version that is higher than 1.0 and the Requester supports the same version. The higher version is then used throughout SPDM for the remaining test assertions.

Pass Criteria:

- Shall return a VERSION response over the DOE interface (transfer is performed from the host over DOE/SPDM following the CMA interface)
- Responder answers with VERSION Request ResponseCode = 04h containing 10h, 11h, or 12h
- A valid version of 1.0, or higher version of 1.1 shall be returned in the VERSION response
- Optional OOB: JSON file shall contain a version of 1.0 or higher for SPDM for the target device

Fail Conditions:

- ErrorCode=ResponseNotReady or 100-ms timeout
- CXL Compliance test suite should error/time out after 100 ms if a VERSION response is not received
- Version is not 1.0 or higher and does not match a version on the host

14.11.6.2 SPDM GET_CAPABILITIES**Prerequisites:**

- Test steps must directly follow successful GET_VERSION test assertion following SPDM protocol

Modes:

- CXL.io
- OOB CMA

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Issue GET_CAPABILITIES over SPDM to target the device over DOE/CMA, using Host capabilities for SPDM version 1.0 or higher as negotiated in the GET_VERSION test assertion.
2. Optional OOB: Issue the Discovery command to gather capabilities information over out-of-band. Skip this step if performed in the GET_VERSION test assertion as JSON should be the same.
3. Validate that the CAPABILITIES response matches the host's capabilities and meets the minimum SPDM version 1.0 requirements.
4. Record Flags for the device capabilities and capture CTExponent for use in timeout of CHALLENGE response and MEASUREMENTS timeout.
5. Validate the CTExponent value within the range for the CMA Spec device. Crypto timeout (CT) time should be less than 2^{23} us.
6. Optional OOB: Validate JSON file that is returned from the Discovery command for capabilities. The capabilities should match those of in-band.

Pass Criteria:

- Valid CAPABILITIES response received that contains RequestResponseCode = 61h for CAPABILITIES and valid Flags (CACHE_CAP, CERT_CAP, CHAL_CAP, MEAS_CAP, MEAS_FRESH_CAP)
- Flags returned determine whether optional capability test assertions apply
- If CERT_CAP is not set, then SPDM-based test assertions end after NEGOTIATE_ALGORITHMS and there is no Certificate test supported
- Valid value for CTExponent should be populated in the CAPABILITIES response
- CTExponent Value must be less than 23
- MEAS_CAP: Confirm the Responder's MEASUREMENTS capabilities. If the responder returns:

- 00b: The Responder does not support MEASUREMENTS capabilities (i.e., the Measurement Test Assertion does not apply)
- 01b: The Responder supports MEASUREMENTS capabilities, but cannot perform signature generation (only the Measurement with Signature test assertion does not apply)
- 10b: The Responder supports MEASUREMENTS capabilities and can generate signatures (all Measurement Test Assertions apply)
- If MEAS_FRESH_CAP is set, then fresh measurements are expected on each MEASUREMENTS request and delays may be observed by Compliance Software

Fail Conditions:

- ErrorCode=ResponsNotReady or 100-ms timeout (CXL Compliance test suite should error/timeout after 100 ms if no response to GET_VERSION is received)
- Invalid Flags or no value for CTExponent
- CTExponent larger than 23

14.11.6.3 SPDM NEGOTIATE_ALGORITHMS

Prerequisites:

- Test must directly follow successful GET_CAPABILITIES test assertion following SPDM protocol

Modes:

- CXL.io
- OOB CMA

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Requester sends NEGOTIATE_ALGORITHMS, including algorithms supported by the host for MeasurementHashAlgo and BaseAsymSel.
2. Responder sends the ALGORITHMS response.

Note:

This response is the “negotiated state” for the Requester/Responder pair until a new GET_VERSION request is sent to “clear the state”.

3. Validate the ALGORITHMS response.

Pass Criteria:

- Valid ALGORITHMS response is received that contains RequestResponseCode = 63h for ALGORITHMS
- Valid fields required:
 - MeasurementSpecificationSel (bit selected should match Requester)
 - MeasurementHashAlgo (Value of 0 if measurements are not supported. If measurements are supported, only one bit set represents the algorithm. Valid

algorithms are: TPM_ALG_SHA_256, TPM_ALG_SHA_384, TPM_ALG_SHA_512, TPM_ALG_SHA3_256, TPM_ALG_SHA3_384, and TPM_ALG_SHA3_512.)

- Expected to support CXL-based algorithm TPM_ALG_SHA_256 at a minimum; PCI-SIG* CMA requires TPM_ALG_SHA_256 and TPM_ALG_SHA_384
- If CHALLENGE is supported, these fields are valid:
 - BaseAsymSel, BaseHashSel, ExtAsymSelCount, and ExtHashSelCount
- One of the following bits must be selected by the BaseAsymAlgo field for signature verification:
 - TPM_ALG_RSASSA_3072, TPM_ALG_ECDSA_ECC_NIST_P256, TPM_ALG_ECDSA_ECC_NIST_P384
 - If CHALLENGE is not supported, then this field should be 0, and Extended Algorithms will not be used in compliance testing

Fail Conditions:

- ErrorCode=ResponsNotReady or timeout (CXL Compliance test suite should error/time out after 100 ms if no response to GET_VERSION is received).
- Measurement is supported, but no algorithm is selected.
- If CHALLENGE is supported, one bit in the BaseAsymAlgo field should be set.
- Responder should match 1 ALGORITHMS capability with the Requester.
- If MEAS_CAP, CERT_CAP, and CHAL_CAP are not supported, then SPDM tests stop.
- If some options are supported, then some tests may continue.

14.11.6.4 SPDM GET_DIGESTS

Prerequisites:

- CERT_CAP=1
- Must directly follow NEGOTIATE_ALGORITHMS test assertion
- Assumes that a cached copy of the Digest or the Certificate is unavailable to the Requester

Modes:

- CXL.io

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Requester sends GET_DIGESTS.
2. Responder sends DIGESTS.
3. Requester saves the content provided by the Digest for future use. (Saved copy shall be known as cached Digest.)
4. If the Responder replies with Busy, then the Requester should repeat the test steps, starting with step 1.

Pass Criteria:

- Param2 of Digests sent by the Responder shall contain a valid Slot Mask that denotes the number of certificate chain entries in the Digest

Fail Conditions:

- Failure to return Digests or times out
- Responder always replies with Busy

14.11.6.5 SPDM GET_CERTIFICATE**Prerequisites:**

- CERT_CAP=1
- Directly follows GET_DIGESTS test assertion
- If the device supports CMA, the device must also support Certificates on Slot 0 with DOE Function 0 and from OOB

Modes:

- CXL.io
- OOB

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

- Requester requests GET_CERTIFICATE with a Param1 value of 0 for Slot 0 for DOE of Function 0. Use Offset 00h and byte length FFFFh to return the entire certificate.
- Response returns CERTIFICATE over DOE.
- Request Slot 0 Certificate over OOB method.
- Host returns CERTIFICATE over OOB.
- Verify Slot 0 Certificate matches between in-band and out-of-band.
- Requester shall save the public key of the leaf certificate, which will be used to decode DIGESTS in future test assertions.
- Use Certificate and Certificate Authority (CA).
- Verify content from Certificate Format/Certificate Chain Test Assertion. Required fields on certificate to be validated:

Open: Add supporting text for step 8 above.

Pass Criteria:

- Same as Test [14.11.4](#)

Fail Conditions:

- Certificate with validity value invalid
- Required fields are missing

- Malformed format for Subject Alternative Name
- Key verification failure
- Mismatch between in-band and out-of-band

14.11.6.6 SPDM CHALLENGE

Prerequisites:

- CERT_CAP=1 and CHAL_CAP=1 must both be supported. Test will issue a warning if both methods are not supported.
- Must follow test assertion sequence up to this point with GET_VERSION, GET_CAPABILITIES, NEGOTIATE_ALGORITHMS, GET_DIGESTS, and GET_CERTIFICATE all being successful prior to CHALLENGE. If CERT_CAP=0, GET_VERSION, GET_CAPABILITIES, NEGOTIATE_ALGORITHMS, CHALLENGE is a valid sequence.
- Compliance Software must keep track of all transactions (per SPDM spec Table 21a: Request ordering and message transcript computation rules for M1/M2) to complete the CHALLENGE request.

Modes:

- CXL.io
- OOB CMA

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Requester sends CHALLENGE using Param1=Slot0, Param2=:
 - a. 00h if MEAS_CAP = 0 (no Measurement Summary Hash).
 - b. 01h = TCB Component Measurement Hash (if device supports only this Measurement).
 - c. FFh = All measurements Hash (if device supports multiple measurements).
 - d. Nonce sent must be a random value.
2. Requester starts a timer to track CT time using CTExponent from the earlier test assertion for Capabilities.
3. Responder returns CHALLENGE_AUTH response before CT time or returns a ResponseNotReady with expected delay time:
 - a. If ResponseNotReady occurs, the Responder must wait CT time + RTT (Round Trip Time) before issuing RESPOND_IF_READY. CT time should be less than 2^{23} microseconds.
4. Record Nonce value returned by the Responder in the table for the final log report. Value should not match the Nonce sent by Requester. The Compliance Software Nonce/Token Table should contain all Nonce and Token entries for all test assertions that are performed on the device.
5. Validate the Signature of the CHALLENGE_AUTH response.
6. Repeat steps 1-4.

7. Validate that the CHALLENGE_AUTH response contains a unique Nonce Value and a valid Signature validated per SPDM spec. Compare the Nonce Value returned by the Responder to the value in the first pass of Step 4 and then validate that the non-incremented value and numbers appear random.

Pass Criteria:

- Valid CHALLENGE_AUTH response and/or valid use of delay with ResponseNotReady before successfully answering with CHALLENGE_AUTH
- Responder should be able to decode and approve CHALLENGE_AUTH as containing a valid signature based on all prior transactions
- Verification of the CHALLENGE_AUTH performed using public key of Cert Slot 0 along with a hash of transactions and signature using the negotiated algorithms from earlier Test Assertions

Fail Conditions:

- CHALLENGE_AUTH not ready by responder prior to expiration of CT time + RTT and ResponseNotReady is not sent by the Responder
- Failure of verification step for CHALLENGE_AUTH contents
- Nonce Value is not unique
- CT time longer than 2^{23} microseconds

14.11.6.7 SPDM GET_MEASUREMENTS Count

Prerequisites:

- SPDM 1.0 or higher, DOE, CMA
- MEAS_CAP = 01b or 10b
- Test assertion is valid after successful GET_VERSION, GET_CAPABILITIES, NEGOTIATE_ALGORITHMS, GET_DIGESTS, GET_CERTIFICATE, CHALLENGE
- Note that issuing GET_MEASUREMENTS resets the "transcript" to NULL

Modes:

- CXL.io
- OOB

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Responder sends GET_MEASUREMENTS response code E0h with Param2 value of 00h to request a count of the device-supported measurements.
2. Responder returns MEASUREMENTS response code 60h with a count of the supported Measurements in Param1.
3. Optional: Compare result with OOB Measurement count.

Pass Criteria:

- Responder sends valid MEASUREMENTS response that contains the count. ResponseNotReady response/delay is permitted.

Fail Conditions:

- Responder fails to respond before timeout or sends an invalid response.

14.11.6.8 SPDM GET_MEASUREMENTS All**Prerequisites:**

- SPDM 1.0 or higher, DOE, CMA
- MEAS_CAP=1
- If MEAS_FRESH_CAP=1, measurements are expected to be fresh on each MEASUREMENTS request

Modes:

- CXL.io
- OOB

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Requester issues GET_MEASUREMENTS requester response code E0h with Param2 value of FFh. If the device is capable of signatures, the request should be with signature.
2. Responder returns MEASUREMENTS response code 60h with all measurements returned. Signature is included if requested. Signature should be valid and nonce returned must be random and recorded into the Compliance Software table of values. ResponseNotReady delay is permitted within the timeout range. Any occurrence of ResponseNotReady should record a token value in the table in the Compliance Software to verify the random value.
3. Number of Measurement blocks shall match the count in the previous test assertion.
4. Repeat steps 1-3 and verify that the measurements match between the MEASUREMENTS responses.
5. OOB step if supported: QueryMeasurements using OOB script and compare the out-of-band measurement values with the in-band values.

Pass Criteria:

- Message delay with ResponseNotReady is permitted
- Measurements match between repeated responses

Fail Conditions:

- Invalid Message response or failure to respond prior to timeout
- Mismatch between measurements

14.11.6.9 SPDM GET_MEASUREMENTS Repeat with Signature

Prerequisites:

- SPDM 1.0 or higher, DOE, CMA.
- MEAS_CAP=01b or 10b
- If MEAS_FRESH_CAP is set, then additional steps could apply
- If capable of signature, then Signature is required
 - For Signature, device must support CHAL_CAP, CERT_CAP
 - Golden Host must support CMA-required BaseAsymAlgo for signature verification: TPM_ALG_RSASSA_3072, TPM_ALG_ECDSA_ECC_NIST_P256, TPM_ALG_ECDSA_ECC_NIST_P384. PCI-SIG CMA requires TPM_ALG_SHA_256 and TPM_ALG_SHA_384 for MeasurementHashAlgo

Modes:

- CXL.io
- OOB

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Requester sends GET_MEASUREMENTS (first measurement as supported by earlier test assertions for count and measurements and index to increment with each repeat of this step).
2. Request should be with signature on the last count of measurement if the device supports signature. If the device supports fresh measurements, measurements are expected to be fresh with each response.
3. Both the Requester and the Responder keep track of messages for validation of signature throughout GET_MEASUREMENTS/MEASUREMENTS for each measurement in count. On the last Measurement, the Requester issues GET_MEASUREMENTS with signature. The Responder may issue ResponseNotReady:
 - a. If ResponseNotReady is observed, validate the fields in ReponseNotReady, including Delay time value and token. Calculate the time required (see ResponseNotReady test assertion). Record the token value in the table for the final report. Token should be a random value.
 - b. Requester should RESPOND_IF_READY based on timeout value. RESPOND_IF_READY should include the same token that was sent by the Responder in ResponseNotReady.
4. Capture the Nonce value from the MEASUREMENTS response if signature is requested. Store the Nonce value in a table for logging in the final report. The value should not be a counter or increment.
5. Capture the measurement value and compare the value against the earlier MEASUREMENTS response. The value should not change after measurement.
6. Validate that the signature is the signature required for the last measurement. This step requires the requester/responder to keep track of all requested measurement

messages until the measurement requesting signature, at which time the transcript state will be cleared.

7. Repeat - Requester sends GET_MEASUREMENTS if additional measurements exist with last request including signature.
8. Repeat MEASUREMENTS request 10 times (for devices that have 1 measurement index, this is 10 MEASUREMENTS responses; for devices that have 5 measurement blocks, this is 5*10 = 50 MEASUREMENTS responses).
9. If OOB is supported, compare the Measurement with OOB.

Pass Criteria:

- Nonce Value is unique and random each time MEASUREMENTS response with signature is received.
- Value does not increment.
- Valid Measurement shall be returned and should match earlier requests for the same measurement index.
- ResponseNotReady, if required, shall include a random token value (should not be same as any nonce values).
- Requester should expect MEASUREMENTS response or another ResponseNotReady if not ready by time of expiry. Measurements are indexed blocks. During MEASUREMENTS requests for each index, requester/responder shall keep track of messages and use those in signature generation/calculation.
- Any SPDM message sent between MEASUREMENTS requests clears this calculation. Requester successfully decodes valid message with signature. Measurement values should be requested for each value supported based on response to the initial GET_MEASUREMENTS request with index list.
- ResponseNotReady is permitted if the responder is approaching CT time + RTT before MEASUREMENTS response is ready. Delay in response is permitted and should meet timeout estimated in ResponseNotReady. If ResponseNotReady occurs, Token Value should be validated to be unique compared to any occurrences during compliance testing.

Fail Conditions:

- Timeout without a ResponseNotReady or GET_MEASUREMENTS
- Signature Failure
- Failure to return measurement/index requested
- Nonce Value is a counter or not a random number
- Timeout (CT time + RTT) occurs with no ResponseNotReady
- Timeout after ResponseNotReady of Wait time + RTT
- Measurement mismatch between responses of same index or mismatch with OOB
- Token value is not random in ResponseNotReady

14.11.6.10 SPDM CHALLENGE Sequences

Prerequisites:

- SPDM 1.0 or higher, DOE, CMA

Note:

Reset does not occur between these test sequences.

- Requester sends CHALLENGE using Param1=Slot0, Param2=:

EVALUATION COPY

- 00h if MEAS_CAP = 0 (no Measurement Summary Hash)
- 01h = TCB Component Measurement Hash (if device supports only this Measurement)
- FFh = All measurements Hash (if device supports multiple measurements)

Note: Successful CHALLENGE clears the transcript as does GET_DIGESTS, GET_VERSION, and GET_MEASUREMENTS. Delays in responses that generate ResponseNotReady and RESPOND_IF_READY messages should follow SPDM spec rules for transcripts regarding occurrences of these messages.

Modes:

- CXL.io

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Requester initiates Sequence 1 and Responder answers each step (Sequence 1: GET_VERSION, GET_CAPABILITIES, NEGOTIATE_ALGORITHMS, GET_DIGESTS, GET_CERTIFICATE, CHALLENGE).
2. CHALLENGE_AUTH should pass validation.
3. Requester issues CHALLENGE.
4. CHALLENGE_AUTH should again pass validation.
5. Requester initiates Sequence 2 and Responder answers each step. Requester uses Slot 0 for GET_CERTIFICATE (Sequence 2: GET_VERSION, GET_CAPABILITIES, NEGOTIATE_ALGORITHMS, GET_CERTIFICATE ("guess" Slot 0 certificate), CHALLENGE).
6. CHALLENGE_AUTH should again pass validation.
7. Requester issues GET_DIGESTS.
8. Responder returns DIGESTS.
9. Requester initiates Sequence 3 and Responder answers each step (Sequence 3: GET_VERSION, GET_CAPABILITIES, NEGOTIATE_ALGORITHMS, GET_DIGESTS, CHALLENGE).
10. CHALLENGE_AUTH should again pass validation.
11. Requester issues GET_DIGESTS.
12. Responder returns DIGESTS.
13. Requester issues CHALLENGE.
14. Responder returns CHALLENGE_AUTH.
15. CHALLENGE_AUTH should pass validation.
16. Requester initiates Sequence 4 and Responder answers each step (Sequence 4: GET_VERSION, GET_CAPABILITIES, NEGOTIATE_ALGORITHMS, CHALLENGE).
17. CHALLENGE_AUTH should pass validation.
18. Requester initiates Sequence 5 and Responder answers each step (Sequence 5: GET_DIGESTS, GET_CERTIFICATE, CHALLENGE).

Pass Criteria:

- Responder may issue RESPOND_IF_READY during any CHALLENGE request, GET_CERTIFICATE, or GET_MEASUREMENTS. A delayed response can occur if the responder responds with ResponseNotReady (CXL Compliance test suite should error/timeout after CT time + RTT for CHALLENGE response). CT is the calculated time that is required by the responder, and is sent during GET_CAPABILITIES. CT time applies to GET_MEASUREMENTS with signature or CHALLENGE. The Requester must keep track of any timeout as described in other test assertions for SPDM.
- Each sequence results in a Valid CHALLENGE response.
- Requester shall successfully verify the fields in each CHALLENGE_AUTH.
- ErrorCode=RequestResynch is permitted by the responder should the responder lose track of transactions. If RequestResynch occurs, the Requester should send GET_VERSION to re-establish state restart test assertion at Step 1. RequestResynch is not a failure. The Test should log a warning if this occurs at the same point in each sequence or repeatedly before completing all steps.

Fail Conditions:

- Any failure to respond to CHALLENGE if the sequence is supported by CAPABILITIES in a FAIL
- CT time + RTT timeout occurs and responder does not send ResponseNotReady
- Any Invalid Response (e.g., CHALLENGE fails verify, or Digest content fails verify)

14.11.6.11 SPDM ErrorCode Unsupported Request

Prerequisites:

- SPDM 1.0 or higher, DOE, CMA

Modes:

- CXL.io

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Requester generates any SPDM message with a Request Response Code that is not listed as valid in spec. Invalid values include the following reserved values in SPDM 1.0: 0x80, 0x85 - 0xDF, 0xE2, and 0xE4 - 0xFD.

Pass Criteria:

- Responder generated error code response with unsupported request (07h)

Fail Conditions:

- No error response from responder or no response to request with any other response that is not error unsupported request

14.11.6.12 SPDM Major Version Invalid

Prerequisites:

- SPDM 1.0 or higher, DOE, CMA

Modes:

- CXL.io

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Requester generates GET_VERSION but uses 30h in the Version field.

Pass Criteria:

- Responder generated error code response with MajorVersionMismatch (41h)

Fail Conditions:

- No error response from responder or response to request with any other response that is not error MajorVersionMismatch

14.11.6.13 SPDM ErrorCode UnexpectedRequest

Prerequisites:

- SPDM 1.0 or higher, DOE, CMA

Modes:

- CXL.io

Topologies:

- SHDA
- SHSW
- SHSW-FM

Test Steps:

1. Requester generates GET_VERSION.
2. Requester generates CHALLENGE.

Pass Criteria:

- Responder generates Error Code response with UnexpectedRequest (04h)

Fail Conditions:

- No error response from responder or response to request with any other response that is not error unsupported request

14.12 Reliability, Availability, and Serviceability

RAS testing is dependent on being able to inject and correctly detect the injected errors. For this testing, it is required that the host and the device both support error injection capabilities.

Certain Device/Host capabilities of error injection are required to enable the RAS tests. First, the required capabilities and configurations are provided. Then, the actual test procedures are laid out. Since these capabilities may only be firmware accessible, currently these are implementation specific. However, future revisions of this specification may define these under an additional capability structure.

The following register describes the required functionalities. All the registers that have an "RWL" attribute should be locked when DVSEC Test Lock is set to 1.

Table 14-27. Register 1: CXL.cachemem LinkLayerErrorInjection (Sheet 1 of 2)

Bit	Attribute	Description
0	RWL	CachePoisonInjectionStart: Software writes 1 to this bit to trigger a single poison injection on a CXL.cache message in the Tx direction. Hardware must override the poison field in the data header slot of the corresponding message (D2H if device, H2D if Host). This bit is required only if CXL.cache protocol is supported.
1	RO-V	CachePoisonInjectionBusy: Hardware loads 1 to this bit when the Start bit is written. Hardware must clear this bit to indicate that it has indeed finished poisoning a packet. Software is permitted to poll on this bit to determine when hardware has finished poison injection. This bit is required only if CXL.cache protocol is supported.
2	RWL	MemPoisonInjectionStart: Software writes 1 to this bit to trigger a single poison injection on a CXL.mem message in the Tx direction. Hardware must override the poison field in the data header slot of the corresponding message. This bit is required only if CXL.mem protocol is supported.
3	RO-V	MemPoisonInjectionBusy: Hardware loads 1 to this bit when the Start bit is written. Hardware must clear this bit to indicate that it has indeed finished poisoning a packet. Software is permitted to poll on this bit to determine when hardware has finished poison injection. This bit is required only if CXL.mem protocol is supported.
4	RWL	IOPoisonInjectionStart: Software writes 1 to this bit to trigger a single poison injection on a CXL.io message in the Tx direction. Hardware must override the poison field in the data header slot of the corresponding message.
5	RO-V	IOPoisonInjectionBusy: Hardware loads 1 to this bit when the Start bit is written. Hardware must clear this bit to indicate that it has indeed finished poisoning a packet. Software is permitted to poll on this bit to determine when hardware has finished poison injection.
7:6	RWL	CacheMemCRCInjection: Software writes to these bits to trigger CRC error injections. The number of CRC bits flipped is given as follows: <ul style="list-style-type: none"> • 00b = Disable. No CRC errors are injected. • 01b = Single bit flipped in the CRC field for "n" subsequent Tx flits, where n is the value in CacheMemCRCInjectionCount. • 10b = 2 bits flipped in the CRC field for "n" subsequent Tx flits, where n is the value in CacheMemCRCInjectionCount. • 11b = 3 bits flipped in the CRC field for "n" subsequent Tx flits, where n is the value in CacheMemCRCInjectionCount. The specific bit positions that are flipped are implementation specific. This field is required if the CXL.cache or CXL.mem protocol is supported.

Table 14-27. Register 1: CXL.cachemem LinkLayerErrorInjection (Sheet 2 of 2)

Bit	Attribute	Description
9:8	RWL	<p>CacheMemCRCInjectionCount: Software writes to these bits to program the number of CRC injections. This field must be programmed by software before OR at the same time as the CacheMemCRCInjection field. The number of flits where CRC bits are flipped is given as follows:</p> <ul style="list-style-type: none"> • 00b = Disable. No CRC errors are injected. • 01b = CRC injection is only for 1 flit. The CacheMemCRCInjectionBusy bit is cleared after 1 injection. • 10b = CRC injection is for 2 flits in succession. The CacheMemCRCInjectionBusy bit is cleared after 2 injections. • 11b = CRC injection is for 3 flits in succession. The CacheMemCRCInjectionBusy bit is cleared after 3 injections. <p>This field is required if the CXL.cache or CXL.mem protocol is supported.</p>
10	RO-V	<p>CacheMemCRCInjectionBusy: Hardware loads 1 to this bit when the Start bit is written. Hardware must clear this bit to indicate that it has indeed finished CRC injections. Software is permitted to poll on this bit to determine when hardware has finished CRC injection. This bit is required if the CXL.cache or CXL.mem protocol is supported.</p>

Table 14-28. Register 2: CXL.io LinkLayer Error Injection

Bit	Attribute	Description
0	RWL	<p>IOPoisonInjectionStart: Software writes 1 to this bit to trigger a single poison injection on a CXL.io message in the Tx direction. Hardware must override the poison field in the data header slot of the corresponding message.</p>
1	RO-V	<p>IOPoisonInjectionBusy: Hardware loads 1 to this bit when the Start bit is written. Hardware must clear this bit to indicate that it has indeed finished poisoning a packet. Software is permitted to poll on this bit to determine when hardware has finished poison injection.</p>
2	RWL	<p>FlowControlErrorInjection: Software writes 1 to this bit to trigger a Flow Control error on CXL.io only. Hardware must override the Flow Control DLLP.</p>
3	RO-V	<p>FlowControlInjectionBusy: Hardware loads 1 to this bit when the Start bit is written. Hardware must clear this bit to indicate that it has indeed finished Flow Control error injections. Software is permitted to poll on this bit to determine when hardware has finished Flow Control error injection.</p>

Table 14-29. Register 3: Flex Bus LogPHY Error Injections

Bit	Attribute	Description
0	RWL	<p>CorrectableProtocolIDErrorInjection: Software writes 1 to this bit to trigger a correctable Protocol ID error on any CXL flit that is issued by the FlexBus LogPHY. Hardware must override the Protocol ID field in the flit.</p>
1	RWL	<p>UncorrectableProtocolIDErrorInjection: Software writes 1 to this bit to trigger an uncorrectable Protocol ID error on any CXL flit that is issued by the FlexBus LogPHY. Hardware must override the Protocol ID field in the flit.</p>
2	RWL	<p>UnexpectedProtocolIDErrorInjection: Software writes 1 to this bit to trigger an unexpected Protocol ID error on any CXL flit that is issued by the FlexBus LogPHY. Hardware must override the Protocol ID field in the flit.</p>
3	RO-V	<p>ProtocolIDInjectionBusy: Hardware loads 1 to this bit when the Start bit is written. Hardware must clear this bit to indicate that it has indeed finished Protocol ID error injections. Software is permitted to poll on this bit to determine when hardware has finished Protocol ID error injection. Software should only program one of the bits between the correctable, uncorrectable, and unexpected Protocol ID error injection bits.</p>

14.12.1 RAS Configuration

14.12.1.1 AER Support

Prerequisites:

- Errors must be reported via the PCI AER mechanism

- AER is as an optional Extended Capability

Test Steps:

1. Read through each Extended Capability (EC) Structure for the Endpoint, and then locate the EC structure for that type.

Pass Criteria:

- AER Extended Capability Structure exists

Fail Conditions:

- AER Extended Capability Structure does not exist

14.12.1.2 CXL.io Poison Injection from Device to Host

Prerequisites:

- CXL device must support Algorithm 1a
- CXL device must support Link Layer Error Injection capabilities for CXL.io

Test Steps:

1. Set up the device for Multiple Write streaming:
 - a. Write a pattern {64{8'hFF}} to cache-aligned Address A1.
 - b. Write a Compliance mode DOE to inject poison:

Table 14-30. CXL.io Poison Injection from Device to Host: I/O Poison Injection Request

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	6, Poison Injection
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	0

- c. Write Compliance mode DOE with the following request:

Table 14-31. CXL.io Poison Injection from Device to Host: Multi-Write Streaming Request (Sheet 1 of 2)

Data Object Byte Offset	Length in Bytes	Description	Value
00h	8	Standard DOE Request Header	
08h	1	Request Code	3, Multiple Write Streaming
09h	1	Version	2
0Ah	2	Reserved	
0Ch	1	Protocol	0
0Dh	1	Virtual Address	0
0Eh	1	Self-checking	0
0Fh	1	Verify Read Semantics	0
10h	1	Num Increments	0

Table 14-31. CXL.io Poison Injection from Device to Host: Multi-Write Streaming Request (Sheet 2 of 2)

Data Object Byte Offset	Length in Bytes	Description	Value
11h	1	Num Sets	0
12h	1	Num Loops	1
13h	1	Reserved	
14h	8	Start Address	A1
1Ch	8	Write Address	0
24h	8	WriteBackAddress	A2 (Must be distinct from A1)
2Ch	8	Byte Mask	FFFF FFFF FFFF FFFFh
34h	4	Address Increment	0
38h	4	Set Offset	0
3Ch	4	Pattern "P"	AAh
40h	4	Increment Pattern "B"	0

Pass Criteria:

- Receiver logs the poisoned received error
- Test software is permitted to read Address A1 to observe the written pattern

Fail Conditions:

- Receiver does not log the poisoned received error

14.12.1.3 CXL.cache Poison Injection**14.12.1.3.1 Device to Host Poison Injection****Prerequisites:**

- Device is CXL.cache capable
- CXL device must support Algorithm 1a
- CXL device must support Link Layer Error Injection capabilities for CXL.cache

Test Steps:

1. Set up the device for Multiple Write streaming:
 - a. Write a pattern {64{8'hFF}} to cache-aligned Address A1.
 - b. Write a Compliance mode DOE to inject poison:

Table 14-32. Device to Host Poison Injection: Cache Poison Injection Request

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	6, Poison Injection
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	1

c. Write Compliance mode DOE with the following request:

Table 14-33. Device to Host Poison Injection: Multi-Write Streaming Request

Data Object Byte Offset	Length in Bytes	Description	Value
00h	8	Standard DOE Request Header	
08h	1	Request Code	3, Multiple Write Streaming
09h	1	Version	2
0Ah	2	Reserved	
0Ch	1	Protocol	1
0Dh	1	Virtual Address	0
0Eh	1	Self-checking	0
0Fh	1	Verify Read Semantics	0
10h	1	Num Increments	0
11h	1	Num Sets	0
12h	1	Num Loops	1
13h	1	Reserved	
14h	8	Start Address	A1
1Ch	8	Write Address	0
24h	8	WriteBackAddress	A2 (Must be distinct from A1)
2Ch	8	Byte Mask	FFFF FFFF FFFF FFFFh
34h	4	Address Increment	0
38h	4	Set Offset	0
3Ch	4	Pattern "P"	AAh
40h	4	Increment Pattern "B"	0

Pass Criteria:

- Receiver (host) logs the poisoned received error
- Test software is permitted to read Address A1 to observe the written pattern

Fail Conditions:

- Receiver does not log the poisoned received error

14.12.1.3.2 Host to Device Poison Injection

This test ensures that if a CXL.cache device receives poisoned data from the host, the device returns the poison indication in the write-back phase. The Receiver on the CXL device must also log and escalate the poisoned received error.

Prerequisites:

- Device is CXL.cache capable
- CXL device must support Algorithm 1a with DirtyEvict and RdOwn semantics

Test Steps:

1. Repeatedly write a predetermined pattern to cacheline-aligned Address A1 (example pattern – all 1s – {64{8'hFF}}). A1 should belong to Host-attached memory.

- a. Write a Compliance mode DOE to the host to inject poison:

Table 14-34. Host to Device Poison Injection: Cache Poison Injection Request

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	6, Poison Injection
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	1

Pass Criteria:

- Receiver (device) logs the poisoned received error
- Test software is permitted to read Address A1 to observe the written pattern

Fail Conditions:

- Receiver does not log the poisoned received error

14.12.1.4 CXL.cache CRC Injection**14.12.1.4.1 Device to Host CRC Injection****Test Equipment:**

- Protocol Analyzer

Prerequisites:

- Device is CXL.cache capable
- CXL device must support Algorithm 1a
- CXL device must support Link Layer Error Injection capabilities for CXL.cache

Test Steps:

1. Setup is the same as Test 14.3.6.1.2.
 - a. While a test is running, software will periodically write a Compliance mode DOE to inject CRC:

Table 14-35. Device to Host CRC Injection: Cache Poison Injection Request

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	7, CRC Injection
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	1

Pass Criteria:

- Same as Test 14.3.6.1.2

- Monitor and verify that CRC errors are injected (using the Protocol Analyzer), and that Retries are triggered as a result

Fail Conditions:

- Same as Test [14.3.6.1.2](#)

14.12.1.4.2 Host to Device CRC Injection**Test Equipment:**

- Protocol Analyzer

Prerequisites:

- Device is CXL.cache capable
- CXL device must support Algorithm 1a

Test Steps:

1. Setup is the same as Test [14.3.6.1.2](#).
 - a. While a test is running, software will periodically write a Compliance mode DOE to inject CRC:

Table 14-36. Host to Device CRC Injection: Cache Poison Injection Request

Data Object Byte Offset	Length in Bytes	Description	Value
0h	8	Standard DOE Request Header	
8h	1	Request Code	7, CRC Injection
9h	1	Version	2
Ah	2	Reserved	
Ch	1	Protocol	1

2. Perform the following steps to **host** registers:
 - a. Write LinkLayerErrorInjection::CacheMemCRCInjectionCount = 3h.
 - b. Write LinkLayerErrorInjection::CacheMemCRCInjection = 2h.
 - c. Poll on LinkLayerErrorInjection::CacheMemCRCInjectionBusy:
 - If 0, Write LinkLayerErrorInjection::CacheMemCRCInjection = 0h
 - Write LinkLayerErrorInjection::CacheMemCRCInjection = 2h
 - Return to (c) to Poll

Pass Criteria:

- Same as Test [14.3.6.1.2](#)
- Monitor and verify that CRC errors are injected (using the Protocol Analyzer), and that Retries are triggered as a result

Fail Conditions:

- Same as Test [14.3.6.1.2](#)

14.12.1.5 CXL.mem Link Poison Injection

14.12.1.5.1 Host to Device Poison Injection

Prerequisites:

- Device is CXL.mem capable

Test Steps:

1. Write {64{8'hFF}} to Address $B1$ from the host. $B1$ must belong to device-attached memory.
2. Set up the **host** Link Layer for poison injection:
 - a. `LinkLayerErrorInjection::MemPoisonInjectionStart = 1h.`
3. Write {64{8'hAA}} to Address $B1$ from the host.

Pass Criteria:

- Receiver (device) logs the poisoned received error
- Test software is permitted to read Address $B1$ to observe the written pattern

Fail Conditions:

- Receiver does not log the poisoned received error

14.12.1.6 CXL.mem CRC Injection

14.12.1.6.1 Host to Device CRC Injection

Test Equipment:

- Protocol Analyzer

Prerequisites:

- Device is CXL.mem capable

Test Steps:

1. Write {64{8'hFF}} to Address $B1$ from the host ($B1$ must belong to device-attached memory).
2. Set up the **host** Link Layer for CRC injection.
 - a. `Write LinkLayerErrorInjection::CacheMemCRCInjectionCount = 1h.`
 - b. `Write LinkLayerErrorInjection::CacheMemCRCInjection = 2h.`
3. Write {64{8'hAA}} to Address $B1$ from the host.
4. Read Address $B1$ from the host, and compare to {64{8'hAA}}.

Pass Criteria:

- Read data == {64{8'hAA}}
- CRC error and Retry observed on the link (Protocol Analyzer used for observation)

Fail Conditions:

- Read data != {64{8'hAA}}

14.12.1.7 Flow Control Injection

This is an optional but strongly recommended test that is applicable only for CXL.io.

14.12.1.7.1 Device to Host Flow Control Injection

Prerequisites:

- CXL device must support Algorithm 1a
- CXL device must support Link Layer Error Injection capabilities

Test Steps:

1. Setup is the same as Test [14.3.6.1.1](#).
2. While a test is running, software will periodically perform the following steps to the Device registers:
 - a. Write LinkLayerErrorInjection::FlowControlInjection = 1h.
 - b. Poll on LinkLayerErrorInjection::FlowControlInjectionBusy:
 - If 0, Write LinkLayerErrorInjection::FlowControlInjection = 0h
 - Write LinkLayerErrorInjection::FlowControlInjection = 2h
 - Return to (b) to Poll

Pass Criteria:

- Same as Test [14.3.6.1.1](#)

Fail Conditions:

- Same as Test [14.3.6.1.1](#)

14.12.1.7.2 Host to Device Flow Control Injection

Prerequisites:

- CXL device must support Algorithm 1a

Test Steps:

1. Setup is the same as Test [14.3.6.1.1](#).
2. While a test is running, software will periodically perform the following steps to Host registers:
 - a. Write LinkLayerErrorInjection::FlowControlInjection = 1h.
 - b. Poll on LinkLayerErrorInjection::FlowControlInjectionBusy:
 - If 0, Write LinkLayerErrorInjection::FlowControlInjection = 0h
 - Write LinkLayerErrorInjection::FlowControlInjection = 2h
 - Return to (b) to Poll

Pass Criteria:

- Same as Test [14.3.6.1.1](#)

Fail Conditions:

- Same as Test [14.3.6.1.1](#)

14.12.1.8 Unexpected Completion Injection

This is an optional but strongly recommended test that is applicable only for CXL.io.

14.12.1.8.1 Device to Host Unexpected Completion Injection

Prerequisites:

- CXL device must support Algorithm 1a
- CXL device must support Device Error Injection capabilities

Test Steps:

1. Setup is the same as Test [14.3.6.1.1](#), except that Self-checking should be disabled.
2. While a test is running, software will periodically write DeviceErrorInjection::UnexpectedCompletionInjection = 1h to the Device registers.

Pass Criteria:

- Unexpected completion error is logged

Fail Conditions:

- No errors are logged

14.12.1.9 Completion Timeout

This is an optional but strongly recommended test that is applicable only for CXL.io.

14.12.1.9.1 Device to Host Completion Timeout

Prerequisites:

- CXL device must support Algorithm 1a
- CXL device must support Device Error Injection capabilities

Test Steps:

1. Setup is the same as Test [14.3.6.1.1](#).
2. While a test is running, write DeviceErrorInjection::CompleterTimeoutInjection = 1h to the Device registers.

Open: Above, referenced register and bit no longer exist (Table 14-41 was removed in r3.0, v0.7). Determine whether this overall test is still needed.

Pass Criteria:

- Completion timeout is logged and escalated to the error manager

Fail Conditions:

- No errors are logged and data corruption is seen

14.12.1.10 CXL.mem Media Poison Injection

14.12.1.10.1 Host to Memory Device Poison Injection

Prerequisites:

- Device is CXL.mem capable

Test Steps:

1. Select error injection target address Device Physical Address (DPA) that belongs to the DUT.
2. Translate the DPA to the Host Physical Address (HPA).
3. Request Poison error injection via Enable Memory Device Poison Injection DOE specifying the DPA where the error is to be injected.
4. Poll on the Poison Injection Response DOE. Successful completion status indicates that the device has injected the poison into memory.
5. Host performs a memory read at the error injection target HPA and the device responds to the read with the poison indicator set.

Pass Criteria:

- Receiver (device) logs the poisoned received error
- When injecting poison into persistent memory regions of the CXL.mem device:
 - The device shall add the new physical address to the device's poison list and the error source should be set to an injected error and reported through the Get Poison List command
 - In addition, the device should add an appropriate poison creation event to its internal Informational Event Log, update the Event Status register, and if configured, interrupt the host
 - Poison shall be persistent across warm reset or cold reset until explicitly cleared by overwriting the cacheline with new data with the poison indicator cleared

Fail Conditions:

- Receiver does not log the poisoned received error

14.12.1.11 CXL.mem LSA Poison Injection

14.12.1.11.1 Host to Memory Device LSA Poison Injection

Prerequisites:

- Device is CXL.mem capable

Test Steps:

1. Select error injection LSA byte offset that belongs to the DUT.
2. Request LSA Poison error injection via Enable Memory Device LSA Poison Injection Compliance DOE, specifying the LSA byte offset where the error is to be injected.
3. Poll on the Poison Injection Response DOE. Successful completion status indicates that the device has injected the poison into memory.
4. Host performs a GetLSA mailbox command that includes the LSA byte offset where the poison was injected into the LSA. The device responds to the read with an error in the mailbox GetLSA command and appropriate error log generation.

Pass Criteria:

- Receiver (device) errors the GetLSA command to the injected LSA byte offset
- When injecting poison into the persistent memory Label Storage Area of the CXL.mem device:
 - Device should add an appropriate poison creation event to its internal Informational Event Log, update the Event Status register, and if configured, interrupt the host
 - Poison shall be persistent across warm reset or cold reset until explicitly cleared by a SetLSA with new data that overwrites the poisoned data at the original poison injection LSA byte offset

Fail Conditions:

- Receiver does not log the poisoned received error

14.12.1.12 CXL.mem Device Health Injection

14.12.1.12.1 Host to Device Poison Injection

Prerequisites:

- Applicable only for devices that support Device Health Injection with the DOE transport
- Device is CXL.mem capable

Test Steps:

1. Request device health injection via Enable CXL.mem Device Health Injection Compliance DOE, specifying the health status field to inject.
2. Poll on the Poison Injection Response DOE. Successful completion status indicates that the device has injected the health status change into the device.
3. Host verifies device health status changes by inspecting Event Log Records and device health status changes.

Pass Criteria:

- Device notifies host of state change through appropriate Event Log Records, and the resulting change in device health can be verified through the Get Health Info command

Fail Conditions:

- Receiver does not see correct event logs or change in health status

14.13 Memory Mapped Registers

14.13.1 CXL Capability Header

Test Steps:

1. The base address for these registers is at Offset 4K from the Register Base Low and Register Base High found in the Register Locator DVSEC.
2. Read Offset 00h, Length 4 bytes.
3. Decode this into:

Bits	Variable
------	----------

15:0 CXL_Capability_ID
19:16 CXL_Capability_Version
23:20 CXL_Cache_Mem_Version
31:24 Array_Size

4. Save the Array_Size to be used for finding the remaining capability headers in the subsequent tests.

5. Verify:

Variable	Value	Condition
CXL_Capability_ID	0001h	Always
CXL_Capability_Version	1h	Always
CXL_Cache_Mem_Version	1h	Always

Pass Criteria:

- Test [14.8.2](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.2 CXL RAS Capability Header

Test Steps:

1. Find this capability by reading all the elements within the Array_Size.
2. Read this element (1 DWORD).
3. Decode this into:

Bits Variable
15:0 CXL_Capability_ID
19:16 CXL_Capability_Version
31:20 CXL_RAS_Capability_Pointer

4. Save CXL_RAS_Capability_Pointer, which is used in subsequent tests.
5. Verify:

Variable	Value	Condition
CXL_Capability_ID	0002h	Always
CXL_Capability_Version	2h	Always

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.3 CXL Security Capability Header

Test Steps:

1. Find this capability by reading all the elements within the Array_Size.
2. Read this element (1 DWORD).

3. Decode this into:

Bits	Variable
15:0	CXL_Capability_ID
19:16	CXL_Capability_Version
31:20	CXL_Security_Capability_Pointer

4. Save CXL_Security_Capability_Pointer, which is used in subsequent tests.

5. Verify:

Variable	Value	Condition
CXL_Capability_ID	0003h	Always
CXL_Capability_Version	1h	Always

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.4 CXL Link Capability Header

Test Steps:

1. Find this capability by reading all the elements within the Array_Size.

2. Read this element (1 DWORD).

3. Decode this into:

Bits	Variable
15:0	CXL_Capability_ID
19:16	CXL_Capability_Version
31:20	CXL_Link_Capability_Pointer

4. Save CXL_Link_Capability_Pointer, which is used in subsequent tests.

5. Verify:

Variable	Value	Condition
CXL_Capability_ID	0004h	Always
CXL_Capability_Version	2h	Always

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.5 CXL HDM Decoder Capability Header

Test Steps:

1. Find this capability by reading all the elements within the Array_Size.

2. Read this element (1 DWORD).

3. Decode this into:

Bits	Variable
------	----------

15:0 CXL_Capability_ID
19:16 CXL_Capability_Version
31:20 CXL_HDM_Decoder_Capability_Pointer

4. Save CXL_HDM_Decoder_Capability_Pointer, which is used in subsequent tests.

5. Verify:

Variable	Value	Condition
CXL_Capability_ID	0005h	Always
CXL_Capability_Version	3h	Always

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.6 CXL Extended Security Capability Header

Test Steps:

1. Find this capability by reading all the elements within the Array_Size.

2. Read this element (1 DWORD).

3. Decode this into:

Bits Variable
15:0 CXL_Capability_ID
19:16 CXL_Capability_Version
31:20 CXL_Extended_Security_Capability_Pointer

4. Save CXL_Extended_Security_Capability_Pointer, which is used in subsequent tests.

5. Verify:

Variable	Value	Condition
CXL_Capability_ID	0006h	Always
CXL_Capability_Version	2h	Always

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.7 CXL IDE Capability Header

Test Steps:

1. Find this capability by reading all the elements within the Array_Size.

2. Read this element (1 DWORD).

3. Decode this into:

Bits Variable
15:0 CXL_Capability_ID

19:16 CXL_Capability_Version
31:20 CXL IDE Capability Pointer

4. Save CXL IDE Capability Pointer, which is used in subsequent tests.

5. Verify:

Variable	Value	Condition
CXL_Capability_ID	0007h	Always
CXL_Capability_Version	2h	Always

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.8 CXL HDM Decoder Capability Register

Prerequisites:

- HDM Decoder Capability is implemented

Test Steps:

1. Read register, CXL_HDM_Interleave_Capability_Pointer + Offset 00h, Length 2 bytes.

2. Decode this into:

Bits	Variable
3:0	Decoder Count
7:4	Target Count

3. Verify:

Variable	Value	Condition
Decoder Count	<Dh	Always
Target Count	<9h	Always

Pass Criteria:

- [14.13.5 Device Present](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.9 CXL HDM Decoder Commit

Prerequisites:

- HDM Decoder Capability is implemented

Test Steps:

1. Program an address range in the Decoder[m+1].Base and Decoder[m+1].Size registers such that:

- Decoder[m+1].Base >= (Decoder[m].Base+Decoder[m].Size), and
 - Decoder[m+1].Base <= Decoder[m+1].Base+Decoder[m+1].Size
2. Program distinct Target Port Identifiers for Interleave Way=0 through $2^{**IW}-1$ (not applicable to Devices).
 3. Set the Commit bit in the Decoder[m+1].Control register.

Pass Criteria:

- Committed bit in the Decoder[m+1].Control register is set
- Error Not Committed bit in the Decoder[m+1].Control register is not set

Fail Conditions:

- Committed bit in the Decoder[m+1].Control register is not set within 10 ms
- Error Not Committed bit in the Decoder[m+1].Control register is set

14.13.10 CXL HDM Decoder Zero Size Commit

Prerequisites:

- HDM Decoder Capability is implemented

Test Steps:

1. Program 0 in the Decoder[m].Size register.
2. Set the Commit bit in the Decoder[m].Control register.

Pass Criteria:

- Committed bit in the Decoder[m+1].Control register is set
- Error Not Committed bit in the Decoder[m+1].Control register is not set

Fail Conditions:

- Committed bit in the Decoder[m+1].Control register is not set within 10 ms
- Error Not Committed bit in the Decoder[m+1].Control register is set

14.13.11 CXL Snoop Filter Capability Header

Test Steps:

1. Find this capability by reading all the elements within the Array_Size.
2. Read this element (1 DWORD).
3. Decode this into:

Bits	Variable
15:0	CXL_Capability_ID
19:16	CXL_Capability_Version
31:20	CXL Snoop Filter Capability Pointer

4. Save CXL Snoop Filter Capability Pointer, which is used in subsequent tests.
5. Verify

Variable	Value	Condition
CXL_Capability_ID	0008h	Always

CXL_Capability_Version	1h	Always
------------------------	----	--------

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.12 CXL Device Capabilities Array Register

This test locates all the CXL Device Capability Headers, in addition to the Verify Conditions listed below.

Test Steps:

1. The base address for this register is obtained from the Register Locator DVSEC.
2. Read Offset 00h, Length 8 bytes.
3. Decode this into:

Bits	Variable
15:0	Capability ID
19:16	Version
47:32	Capabilities Count

4. Verify:

Variable	Value	Condition
Capability ID	0000h	Always
Version	01h	Always

5. For N, where N ranges from 1 through Capabilities_Count, find each Capability Header Element by reading 2 bytes at Offset N*10h.

6. Decode this into:

Bits	Variable
15:0	CXL_Capability_ID_Arr[N]

7. If CXL_Capability_ID_Arr[N] == 0001h, save Offset N*10h as Device_Status_Registers_Capabilities_Header_Base.
8. If CXL_Capability_ID_Arr[N] == 0002h, save Offset N*10h as Primary_Mailbox_Registers_Capabilities_Header_Base.
9. If CXL_Capability_ID_Arr[N] == 0003h, save Offset N*10h as Secondary_Mailbox_Registers_Capabilities_Header_Base.
10. If CXL_Capability_ID_Arr[N] == 4000h, save Offset N*10h as Memory_Device_Status_Registers_Capabilities_Header_Base.

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.13 Device Status Registers Capabilities Header Register

Test Steps:

1. Read offset Device_Status_Registers_Capabilities_Header_Base, Length 4 bytes. Device_Status_Registers_Capabilities_Header_Base is obtained in the test performed in Test [14.13.12](#).
2. Decode this into:

Bits	Variable
15:0	CXL_Capability_ID
19:16	CXL_Capability_Version

3. Verify:

Variable	Value	Condition
CXL_Capability_ID	0001h	Always
CXL_Capability_Version	1h	Always

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.14 Primary Mailbox Registers Capabilities Header Register

Test Steps:

1. Read offset Primary_Mailbox_Registers_Capabilities_Header_Base, Length 4 bytes. Primary_Mailbox_Registers_Capabilities_Header_Base is obtained in the test performed in Test [14.13.12](#).
2. Decode this into:

Bits	Variable
15:0	CXL_Capability_ID
19:16	CXL_Capability_Version

3. Verify:

Variable	Value	Condition
CXL_Capability_ID	0002h	Always
CXL_Capability_Version	1h	Always

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.15 Secondary Mailbox Registers Capabilities Header Register

Test Steps:

1. Read offset Secondary_Mailbox_Registers_Capabilities_Header_Base, Length 4 bytes. Secondary_Mailbox_Registers_Capabilities_Header_Base is obtained in the test performed in Test [14.13.12](#).

2. Decode this into:

Bits	Variable
15:0	CXL_Capability_ID
19:16	CXL_Capability_Version

3. Verify:

Variable	Value	Condition
CXL_Capability_ID	0003h	Always
CXL_Capability_Version	1h	Always

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.16 Memory Device Status Registers Capabilities Header Register

Test Steps:

1. Read offset Memory_Device_Status_Registers_Capabilities_Header_Base, Length 4 bytes. Memory_Device_Status_Registers_Capabilities_Header_Base is obtained in the test performed in Test [14.13.12](#).
2. Find the CXL Device Capability Header register as described in Test [14.13.12](#), step 5, Length 4 bytes.
3. Decode this into:

Bits	Variable
15:0	CXL_Capability_ID
19:16	CXL_Capability_Version

4. Verify:

Variable	Value	Condition
CXL_Capability_ID	4000h	Always
CXL_Capability_Version	1h	Always

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.17 CXL Timeout and Isolation Capability Header

Prerequisites:

- Device supports 256B Flit mode and 256B Flit mode is enabled

Test Steps:

1. Find this capability by reading all the elements within the Array_Size.

2. Read this element (1 DWORD).

3. Decode this into:

Bits	Variable
15:0	CXL_Capability_ID
19:16	CXL_Capability_Version
31:20	CXL_Timeout_and_Isolation_Capability_Pointer

4. Save CXL_Timeout_and_Isolation_Capability_Pointer, which is used in subsequent tests.

5. Verify:

Variable	Value	Condition
CXL_Capability_ID	09h	Always
CXL_Capability_Version	01h	Always

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.18 CXL.cachemem Extended Register Header

Prerequisites:

- Device supports 256B Flit mode and 256B Flit mode is enabled

Test Steps:

1. Find this capability by reading all the elements within the Array_Size.

2. Read this element (1 DWORD).

3. Decode this into:

Bits	Variable
15:0	CXL_Capability_ID
19:16	CXL_Capability_Version
31:20	CXL.cachemem Extended Register Capability Pointer

4. Save CXL.cachemem Extended Register Capability Pointer, which is used in subsequent tests.

5. Verify:

Variable	Value	Condition
CXL_Capability_ID	0Ah	Always
CXL_Capability_Version	01h	Always

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.19 CXL BI Route Table Capability Header

Prerequisites:

- Device supports 256B Flit mode and 256B Flit mode is enabled

Test Steps:

1. Find this capability by reading all the elements within the Array_Size.
2. Read this element (1 DWORD).
3. Decode this into:

Bits	Variable
15:0	CXL_Capability_ID
19:16	CXL_Capability_Version
31:20	CXL BI Route Table Capability Pointer

4. Save CXL BI Route Table Capability Pointer, which is used in subsequent tests.
5. Verify:

Variable	Value	Condition
CXL_Capability_ID	0Bh	Always
CXL_Capability_Version	01h	Always

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.20 CXL BI Decoder Capability Header

Prerequisites:

- Device supports 256B Flit mode and 256B Flit mode is enabled

Test Steps:

1. Find this capability by reading all the elements within the Array_Size.
2. Read this element (1 DWORD).
3. Decode this into:

Bits	Variable
15:0	CXL_Capability_ID
19:16	CXL_Capability_Version
31:20	CXL BI Decoder Capability Pointer

4. Save CXL BI Decoder Capability Pointer, which is used in subsequent tests.
5. Verify:

Variable	Value	Condition
CXL_Capability_ID	0Ch	Always
CXL_Capability_Version	01h	Always

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.21 CXL Cache ID Route Table Header

Prerequisites:

- Device supports 256B Flit mode and 256B Flit mode is enabled

Test Steps:

1. Find this capability by reading all the elements within the Array_Size.
2. Read this element (1 DWORD).
3. Decode this into:

Bits	Variable
15:0	CXL_Capability_ID
19:16	CXL_Capability_Version
31:20	CXL Cache ID Route Table Capability Pointer
4. Save CXL Cache ID Route Table Capability Pointer, which is used in subsequent tests.
5. Verify:

Variable	Value	Condition
CXL_Capability_ID	0Dh	Always
CXL_Capability_Version	01h	Always

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.22 CXL Cache ID Decoder Capability Header

Prerequisites:

- Device supports 256B Flit mode and 256B Flit mode is enabled

Test Steps:

1. Find this capability by reading all the elements within the Array_Size.
2. Read this element (1 DWORD).
3. Decode this into:

Bits	Variable
15:0	CXL_Capability_ID
19:16	CXL_Capability_Version
31:20	CXL Cache ID Local Decoder Capability Pointer
4. Save CXL Cache ID Local Decoder Capability Pointer, which is used in subsequent tests.
5. Verify:

Variable	Value	Condition
-----------------	--------------	------------------

CXL_Capability_ID	0Eh	Always
CXL_Capability_Version	01h	Always

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.13.23 CXL Extended HDM Decoder Capability Header

Test Steps:

1. Find this capability by reading all the elements within the Array_Size.
2. Read this element (1 DWORD).
3. Decode this into:

Bits	Variable
15:0	CXL_Capability_ID
19:16	CXL_Capability_Version
31:20	CXL Extended HDM Decoder Capability Pointer
4. Save CXL Extended HDM Decoder Capability Pointer, which is used in subsequent tests.
5. Verify:

Variable	Value	Condition
CXL_Capability_ID	0Fh	Always
CXL_Capability_Version	01h	Always

Pass Criteria:

- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.14 Memory Device Tests

This section covers tests that are applicable to devices that support the CXL.mem protocol.

14.14.1 DVSEC CXL Range 1 Size Low Registers

Prerequisites:

- Not applicable to FM-owned LD
- Device is CXL.mem capable

Test Steps:

1. Read Configuration Space for DUT, CXL_DEVICE_DVSEC_BASE + Offset 1Ch, Length 2 bytes.
2. Decode this into:

Bits	Variable	
7:5	Memory_Class	
12:8	Desired_Interleave	
3. Verify:		
Variable	Value	Condition
Memory_Class	000b or 001b	Always
Desired_Interleave	00h, 01h, or 02h	Always
Desired_Interleave	03h, 04h, 05h, 06h, or 07h	CXL 2.0 and higher

Pass Criteria:

- Test [14.8.4](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.14.2 DVSEC CXL Range 2 Size Low Registers**Prerequisites:**

- Not applicable to FM-owned LD
- Device is CXL.mem capable
- HDM_Count=10b

Inputs:

- **Type:** Volatile or Non-volatile
- **Class:** Memory or Storage

Test Steps:

1. Read Configuration Space for DUT, CXL_DEVICE_DVSEC_BASE + Offset 2Ch, Length 2 bytes.
2. Decode this into:

Bits	Variable	
4:2	Media_Type	
7:5	Memory_Class	
12:8	Desired_Interleave	
3. Verify:		
Variable	Value	Condition
Media_Type	000b, 001b, or 010b	Always
Memory_Class	000b, 001b, or 010b	Always
Desired_Interleave	0h, 1h or 2h	Always
Desired_Interleave	03h, 04h, 05h, 06h, or 07h	CXL 2.0 and higher

Pass Criteria:

- Test [14.8.4](#) passed
- Verify Conditions are met

Fail Conditions:

- Verify Conditions failed

14.15 Sticky Register Tests

This section covers tests applicable to registers that are sticky through a reset.

14.15.1 Sticky Register Test

Test Steps:

1. Read and record value of following ROS registers for step 5:

Error Capabilities and Control Register (Offset 14h)

Bits	Variable
5:0	First_Error_Pointer

Header Log Registers (Offset 18h)

Bits	Variable
511:0	Header Log

Note:

Register contents may or may not be 0.

2. Set following RWS registers to settings as per list and record written values for step 5.

RWS Registers and settings:**Uncorrectable Error Mask Register (Offset 04h)**

Bits	Variable	Settings
11:0	Error Mask registers	Set to FFFh
16:14	Error Mask registers	Set to 111b

Uncorrectable Error Severity Register (Offset 08h)

Bits	Variable	Settings
11:0	Error Severity registers	Set to FFFh
16:14	Error Severity registers	Set to 111b

Correctable Error Mask Register (Offset 10h)

Bits	Variable	Settings
6:0	Error Mask registers	Clear to 00h

Error Capabilities and Control Register (Offset 14h)

Bits	Variable	Settings
13:13	Poison_Enabled	Set to 1

CXL Link Layer Capability Register (Offset 00h)

Bits	Variable	Settings
------	----------	----------

3:0	CXL Link Version Supported	Set to 2h
15:8	LLR Wrap Value Supported	Set to FFh

Note: Intention is to set the register to a nonzero value.

CXL Link Layer Control and Status Register (Offset 08h)

Bits	Variable	Settings
1:1	LL_Init_Stall	Set to 1
2:2	LL_Crd_Stall	Set to 1

CXL Link Layer Rx Credit Control Register (Offset 10h)

Bits	Variable	Settings
9:0	Cache Req Credits	Set to 3FFh
19:10	Cache Rsp Credits	Set to 3FFh
29:20	Cache Data Credits	Set to 3FFh
39:30	Mem Req_Rsp Credits	Set to 3FFh
49:40	Mem Data Credits	Set to 3FFh
59:50	BI Credits	Set to 3FFh

CXL Link Layer Ack Timer Control Register (Offset 28h)

Bits	Variable	Settings
7:0	Ack Force Threshold	Set to FFh
17:8	Ack or CRD Flush Retimer	Set to 1FFh

CXL Link Layer Defeature Register (Offset 30h)

Bits	Variable	Settings
0:0	MDH Disable	Set to 1

DVSEC CXL Control2 (Offset 10h)

Bits	Variable	Settings
4:4	Desired Volatile HDM State after Hot Reset	Set to 1 if DVSEC CXL Capability3 (Offset 38h) Bit 3 Volatile HDM State after Hot Reset – Configurability == 1

3. Issue a link Hot Reset.
4. Wait for the link to train back to CXL.
5. Verify:
 - a. ROS register values before and after link reset are matching.
 - b. RWS register values before and after reset are matching.

Pass Criteria:

- Test [14.8.2](#) passed
- Verify Conditions are met

Fail Conditions:

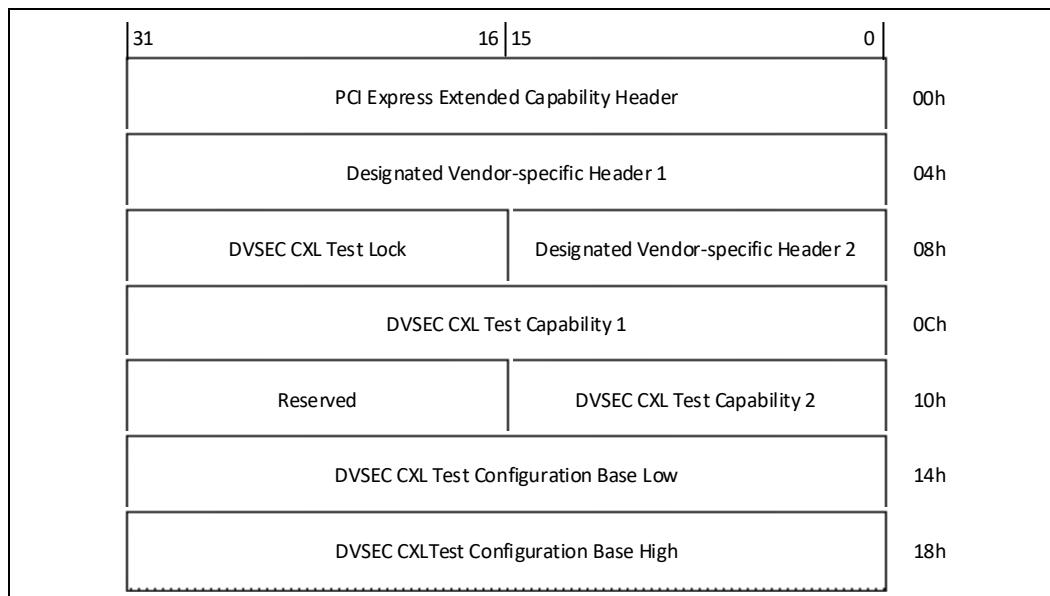
- Verify Conditions failed

14.16 Device Capability and Test Configuration Control

Implementations are expected to provision a Data Object Exchange (DOE) Interface to enable compliance capabilities.

14.16.1 CXL Device Test Capability Advertisement

Figure 14-19. PCIe DVSEC for Test Capability



To advertise Test capabilities, the standard DVSEC register fields should be set as below:

Table 14-37. DVSEC Registers

Register	Bit Location	Field	Value
Designated Vendor-Specific Header 1 (Offset 04h)	15:0	DVSEC Vendor ID	1E98h
	19:16	DVSEC Revision	0h
	31:20	DVSEC Length	1Ch
Designated Vendor-Specific Header 2 (Offset 08h)	15:0	DVSEC ID	00Ah

Table 14-38. DVSEC CXL Test Lock (Offset 0Ah)

Bit	Attribute	Description
0	RWO	TestLock: Software writes 1 to lock the relevant test configuration registers.
15:1	N/A	Reserved

Table 14-39. DVSEC CXL Test Configuration Base Low (Offset 14h)

Bit	Attribute	Description
0	RO	MemorySpaceIndicator: The test configuration registers are in memory space. Device must hardwire this bit to 0.
2:1	RO	Type <ul style="list-style-type: none"> 00b = Base register is 32 bits wide and can be mapped anywhere in the 32-bit address space 01b = Reserved 10b = Base register is 64 bits wide and can be mapped anywhere in the 64-bit address space 11b = Reserved
3	RO	Reserved: Device must hardwire this bit to 0.
31:4	RW	BaseLow: Bits [31:4] of the base address where the test configuration registers exist.

Table 14-40. DVSEC CXL Test Configuration Base High (Offset 18h)

Bit	Attribute	Description
31:0	RW	BaseHigh: Bits [63:32] of the base address where the test configuration registers exist.

14.16.2 Debug Capabilities in Device

14.16.2.1 Error Logging

The following capabilities in a device are strongly recommended to support ease of verification and compliance testing.

A device that supports self-checking must include an error status and header log register with the following fields:

Table 14-41. Register 9: ErrorLog1 (Offset 40h)

Bit	Attribute	Description
31:0	RW	ExpectedPattern: Expected data pattern as per device hardware.
63:32	RW	ObservedPattern: Observed data pattern as per device hardware.

Table 14-42. Register 10: ErrorLog2 (Offset 48h)

Bit	Attribute	Description
31:0	RW	ExpectedPattern: Expected data pattern as per device hardware.
63:32	RW	ObservedPattern: Observed data pattern as per device hardware.

Table 14-43. Register 11: ErrorLog3 (Offset 50h)

Bit	Attribute	Description
7:0	RW	ByteOffset: First byte offset within the cacheline where the data mismatch was observed.
15:8	RW	LoopNum: Loop number where data mismatch was observed.
16	RW1C	ErrorStatus: Set to 1 by device if data mismatch was observed.

14.16.2.2 Event Monitors

It is strongly recommended that a device advertise at least 2 event monitors, which can be used to count device-defined events. An event monitor consists of two 64-bit registers:

- An event controller: EventCtrl
- An event counter: EventCount

The usage model is for software to program EventCtrl to count an event of interest, and then read the EventCount to determine how many times the event has occurred. At a minimum, a device must implement the ClockTicks event. When the ClockTicks event is selected via the event controller, the event counter will increment once every clock cycle, based on the application layer's clock. Further suggested events may be published in the future. Examples of other events that a device may choose to implement are:

- Number of times a particular opcode is sent or received
- Number of retries or CRC errors
- Number of credit returns sent or received
- Device-specific events that may help visibility on the platform or with statistical calculation of performance

[Table 14-44](#) and [Table 14-45](#) list the EventCtrl and EventCount register formats, respectively.

Table 14-44. Register 12: EventCtrl (Offset 60h)

Bit	Attribute	Description
7:0	RW	EventSelect: Field that is used to select which of the available events should be counted in the paired EventCount register.
15:8	RW	SubEventSelect: Field that is used to select which sub-conditions of an event should be counted in the paired EventCount register. This field is a bitmask, where each bit represents a different condition. The EventCount should increment if any of the selected sub-conditions occurs. For example, an event might be "transactions received", with three sub-conditions of "read", "write", and "completion".
16	N/A	Reserved
17	RW	Reset: When set to 1, the paired EventCount register will be cleared to 0. Writing a 0 to this bit has no effect.
18	RW	EdgeDetect <ul style="list-style-type: none"> • 0 = Counter will increment once within each cycle that the event has occurred • 1 = Counter will increment when a 0 to 1 transition (i.e., rising edge) is detected
63:19	N/A	Reserved

Table 14-45. Register 13: EventCount (Offset 68h)

Bit	Attribute	Description
63:0	RO	EventCount: Hardware load register that is updated with a running count of the occurrences of the event programmed in the EventCtrl register. It is monotonically increasing, unless software explicitly writes it to a lower value or writes to the "Reset" field of the paired EventCtrl register.

14.16.3 Compliance Mode DOE

Device 0, Function 0 of a CXL device must support the DOE mailbox for the compliance modes to be controlled through it. The Vendor ID must be set to the CXL Vendor ID to indicate that this Object type is defined by the CXL specification. The Data Object Type must be cleared to 00h to advertise that this is a Compliance Mode type of data object.

Table 14-46. Compliance Mode – Data Object Header

Bits Location	Field	Value
15:0	Vendor ID	1E98h
23:16	Data Object Type	00h

Table 14-47. Compliance Mode Return Values

Value	Description	Value	Description
0000 0000h	Success	0000 0005h	Target Busy
0000 0001h	Not Authorized	0000 0006h	Target Not Initialized
0000 0002h	Unknown Failure	0000 0007h	Invalid Address Specified
0000 0003h	Unsupported Injection Function	0000 0008h	Invalid Injection Parameter
0000 0004h	Internal Error		

14.16.3.1 Compliance Mode Capability

Request and response pair for determining the device's compliance capabilities.

Table 14-48. Compliance Mode Availability Request

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Request Code: Value is 0, Query Capabilities.
9h	1	Version of Capability Requested: Supply 0 here for the highest supported Compliance DOE version, or specify a specific version (e.g., "3").
Ah	2	Reserved

Table 14-49. Compliance Mode Availability Response (Sheet 1 of 2)

Data Object Byte Offset	Length in Bytes	Description
00h	8	Standard DOE Request Header
08h	1	Response Code: Value is 0, Query Capabilities.
09h	1	Version of Capability Returned: Returns supported version of the Compliance mode DOE, by the spec revision number (e.g., "3").
0Ah	1	Length of Capability Package
0Bh	1	Status: See Table 14-47 for error codes.

Table 14-49. Compliance Mode Availability Response (Sheet 2 of 2)

Data Object Byte Offset	Length in Bytes	Description
0Ch	8	Available Compliance Capabilities Bitmask
14h	8	Enabled Compliance Capabilities Bitmask
1Ch	8	Compliance Capabilities Options: See Table 14-50 for Compliance Option value descriptions.

The Available Capabilities and Enabled Capabilities bitmask values correspond to the request codes of each capability. For example, bit 1 will be set if the DOE supports Request code 1, "Status", and bit 3 will be set if the DOE supports Request code 3, "Multiple Write Streaming".

Table 14-50. Compliance Options Value Descriptions

Bits	Description
15:0	Write Semantics Supported: Bitmask with the corresponding values: <ul style="list-style-type: none">Bit[0]: Set to 1 if Device supports CXL.cache and ItoMW_r opcodes as requesterBit[1]: Set to 1 if Device supports CXL.cache and WrCur opcodes as requesterBit[2]: Set to 1 if Device supports CXL.cache and DirtyEvict opcodes as requesterBit[3]: Set to 1 if Device supports CXL.cache and WOWrInv opcodes as requesterBit[4]: Set to 1 if Device supports CXL.cache and WOWrInvF opcodes as requesterBit[5]: Set to 1 if Device supports CXL.cache and WrInv opcodes as requesterBit[6]: Set to 1 if Device supports CXL.cache and CLFlush opcodes as requesterBit[7]: Set to 1 if Device supports CXL.cache and CleanEvict opcodes as requesterBit[8]: Set to 1 if Device supports CXL.cache and CleanEvictNoData opcodes as requesterBits[15:9]: Reserved
31:16	Read Semantics Supported: Bitmask with the corresponding values: <ul style="list-style-type: none">Bit[16]: Set to 1 if Device supports CXL.cache and RdCurr opcodes as requesterBit[17]: Set to 1 if Device supports CXL.cache and RdOwn opcodes as requesterBit[18]: Set to 1 if Device supports CXL.cache and RdShared opcodes as requesterBit[19]: Set to 1 if Device supports CXL.cache and RdAny opcodes as requesterBit[20]: Set to 1 if Device supports CXL.cache and RdOwnNoData opcodes as requesterBits[31:21]: Reserved
47:32	<ul style="list-style-type: none">Bit[32]: Set to 1 if Device supports CXL.cache and CacheFlushed opcodes as requesterBits[47:33]: Reserved
63:48	Reserved

The Available Compliance Capabilities and Enabled Compliance Capabilities bitmask values correspond to the request codes of each capability. For example, bit 1 will be set if the DOE supports Request code 1, "Status", and bit 3 will be set if the DOE supports Request code 3, "Multiple Write Streaming".

14.16.3.2 Compliance Mode Status

Shows which compliance mode capabilities are enabled or in use.

Table 14-51. Compliance Mode Status

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Request Code: Value is 1, Query Status.
9h	1	Version of Capability Requested
Ah	2	Reserved

Table 14-52. Compliance Mode Status Response

Data Object Byte Offset	Length in Bytes	Description
00h	8	Standard DOE Header
08h	1	Response Code: Value is 1, Query Status.
09h	1	Version of Capability Returned
0Ah	1	Length of Capability Package
0Bh	4	Capability Bitfield
0Eh	2	Cache Size
10h	1	Cache Size Units

14.16.3.3 Compliance Mode Halt All**Table 14-53. Compliance Mode Halt All**

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Request Code: Value is 2, Halt All.
9h	1	Version of Capability Requested
Ah	2	Reserved

Table 14-54. Compliance Mode Halt All Response

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Response Code: Value is 2, Halt All.
9h	1	Version of Capability Returned
Ah	1	Length of Capability Package
Bh	1	Status: See Table 14-47 for error codes.

14.16.3.4 Compliance Mode Multiple Write Streaming**Table 14-55. Enable Multiple Write Streaming Algorithm on the Device (Sheet 1 of 2)**

Data Object Byte Offset	Length in Bytes	Description
00h	8	Standard DOE Request Header
08h	1	Request Code: Value is 3, Multiple Write Streaming.
09h	1	Version of Capability Requested
0Ah	2	Reserved
0Ch	1	Protocol
0Dh	1	Virtual Address
0Eh	1	Self-checking
0Fh	1	Verify Read Semantics

Table 14-55. Enable Multiple Write Streaming Algorithm on the Device (Sheet 2 of 2)

Data Object Byte Offset	Length in Bytes	Description
10h	1	Num Increments
11h	1	Num Sets
12h	1	Num Loops
13h	1	Reserved
14h	8	Start Address
1Ch	8	Write Address
24h	8	Writeback Address
2Ch	8	Byte Mask
34h	4	Address Increment
38h	4	Set Offset
3Ch	4	Pattern "P"
40h	4	Increment Pattern "B"

Table 14-56. Compliance Mode Multiple Write Streaming Response

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Response Code: Value is 3, Multiple Write Streaming.
9h	1	Version of Capability Returned
Ah	1	Length of Capability Package
Bh	1	Status: See Table 14-47 for error codes.

If the device only supports Virtual Addresses, and the Virtual Address is cleared to 0, the return value shall be 01h “Not Authorized”. If the device only supports Physical Addresses, and the Virtual Address is set to 1, the return value shall be 01h “Not Authorized”.

14.16.3.5 Compliance Mode Producer-Consumer

Table 14-57. Enable Producer-Consumer Algorithm on the Device (Sheet 1 of 2)

Data Object Byte Offset	Length in Bytes	Description
00h	8	Standard DOE Request Header
08h	1	Request Code: Value is 4, Producer-Consumer.
09h	1	Version of Capability Requested
0Ah	2	Reserved
0Ch	1	Protocol
0Dh	1	Num Increments
0Eh	1	Num Sets
0Fh	1	Num Loops
10h	1	Write Semantics

Table 14-57. Enable Producer-Consumer Algorithm on the Device (Sheet 2 of 2)

Data Object Byte Offset	Length in Bytes	Description
11h	3	Reserved
14h	8	Start Address
1Ch	8	Byte Mask
24h	4	Address Increment
28h	4	Set Offset
2Ch	4	Pattern

Table 14-58. Compliance Mode Producer-Consumer Response

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Response Code: Value is 4, Producer-Consumer.
9h	1	Version of Capability Returned
Ah	1	Length of Capability Package
Bh	1	Status: See Table 14-47 for error codes.

If the device only supports Virtual Addresses, and the Virtual Address is cleared to 0, the return value shall be 01h "Not Authorized". If the device only supports Physical Addresses, and the Virtual Address is set to 1, the return value shall be 01h "Not Authorized".

14.16.3.6 Test Algorithm 1b Multiple Write Streaming with Bogus Writes

Table 14-59. Enable Algorithm 1b, Write Streaming with Bogus Writes (Sheet 1 of 2)

Data Object Byte Offset	Length in Bytes	Description
00h	8	Standard DOE Request Header
08h	1	Request Code: Value is 5, Test Algorithm 1b.
09h	1	Version of Capability Requested
0Ah	2	Reserved
0Ch	1	Protocol
0Dh	1	Virtual Address
0Eh	1	Self-checking
0Fh	1	Verify Read Semantics
10h	1	Num Increments
11h	1	Num Sets
12h	1	Num Loops
13h	1	Reserved
14h	8	Start Address
1Ch	8	Writeback Address
24h	8	Byte Mask

Table 14-59. Enable Algorithm 1b, Write Streaming with Bogus Writes (Sheet 2 of 2)

Data Object Byte Offset	Length in Bytes	Description
2Ch	4	Address Increment
30h	4	Set Offset
34h	4	Pattern "P"
38h	4	Increment Pattern "B"
3Ch	1	Bogus Writes Count
3Dh	3	Reserved
40h	4	Bogus Writes Pattern

Table 14-60. Algorithm1b Response

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Response Code: Value is 5, Test Algorithm 1b.
9h	1	Version of Capability Returned
Ah	1	Length of Capability Package
Bh	1	Status: See Table 14-47 for error codes.

14.16.3.7 Inject Link Poison

Table 14-61. Enable Poison Injection into

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Request Code: Value is 6, Poison Injection.
9h	1	Version of Capability Requested
Ah	2	Reserved
Ch	1	Protocol <ul style="list-style-type: none"> • 00h = CXL.io • 01h = CXL.cache • 02h = CXL.mem

Table 14-62. Poison Injection Response

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Response Code: Value is 6, Poison Injection.
9h	1	Version of Capability Returned
Ah	1	Length of Capability Package
Bh	1	Status: See Table 14-47 for error codes.

14.16.3.8 Inject CRC

Table 14-63. Enable CRC Error into Traffic

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Request Code: Value is 7, CRC Injection.
9h	1	Version of Capability Requested
Ah	2	Reserved
Ch	1	Num Bits Flipped
Dh	1	Num Flits Injected

Table 14-64. CRC Injection Response

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Response Code: Value is 7, CRC Injection.
9h	1	Version of Capability Returned
Ah	1	Length of Capability Package
Bh	1	Status: See Table 14-47 for error codes.

14.16.3.9 Inject Flow Control

Table 14-65. Enable Flow Control Injection

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Request Code: Value is 8, Flow Control Injection.
9h	1	Version of Capability Requested
Ah	2	Reserved
Ch	1	Inject Flow Control

Table 14-66. Flow Control Injection Response

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Response Code: Value is 8, Flow Control Injection.
9h	1	Version of Capability Returned
Ah	1	Length of Capability Package
Bh	1	Status: See Table 14-47 for error codes.

14.16.3.10 Toggle Cache Flush

Table 14-67. Enable Cache Flush Injection

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Request Code: Value is 9, Cache Flush.
9h	1	Version of Capability Requested
Ah	2	Reserved
Ch	1	<ul style="list-style-type: none"> • 00h = Cache Flush Disabled • 01h = Cache Flush Enabled

Table 14-68. Cache Flush Injection Response

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Response Code: Value is 9, Cache Flush.
9h	1	Version of Capability Returned
Ah	1	Length of Capability Package
Bh	1	Status: See Table 14-47 for error codes.

14.16.3.11 Inject MAC Delay

Delay MAC on IDE secure link until it no longer meets spec, flit 6+.

Table 14-69. MAC Delay Injection

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Request Code: Value is 0Ah, Delay MAC.
9h	1	Version of Capability Requested
Ah	2	Reserved
Ch	1	<ul style="list-style-type: none"> • 00h = Disable • 01h = Enable MAC Delay
Dh	1	Mode <ul style="list-style-type: none"> • 00h = CXL.io • 01h = CXL.cachemem
Eh	1	Delay: Number of flits to delay MAC. 6+ = error condition.

Table 14-70. MAC Delay Response (Sheet 1 of 2)

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Response Code: Value is 0Ah, MAC delay injection.

Table 14-70. MAC Delay Response (Sheet 2 of 2)

Data Object Byte Offset	Length in Bytes	Description
9h	1	Version of Capability Returned
Ah	1	Length of Capability Package
Bh	1	Status <ul style="list-style-type: none"> • 00h = Success • See Table 14-47 for other error codes

14.16.3.12 Insert Unexpected MAC

Insert an unexpected MAC on a non-IDE secured channel.

Table 14-71. Unexpected MAC Injection

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Request Code: Value is 0Bh, Unexpected MAC injection.
9h	1	Version of Capability Requested
Ah	2	Reserved
Ch	1	<ul style="list-style-type: none"> • 00h = Disable • 01h = Insert message • 02h = Delete message
Dh	1	Mode <ul style="list-style-type: none"> • 00h = CXL.io • 01h = CXL.cachemem

Table 14-72. Unexpected MAC Injection Response

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Response Code: Value is 0Bh, Unexpected MAC injection.
9h	1	Version of Capability Returned
Ah	1	Length of Capability Package

14.16.3.13 Inject Viral**Table 14-73. Enable Viral Injection (Sheet 1 of 2)**

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Request Code: Value is 0Ch, Inject Viral.

Table 14-73. Enable Viral Injection (Sheet 2 of 2)

Data Object Byte Offset	Length in Bytes	Description
9h	1	Version
Ah	2	Reserved
Ch	1	Protocol <ul style="list-style-type: none"> • 00h = CXL.io • 01h = CXL.cache • 02h = CXL.mem

Table 14-74. Flow Control Injection Response

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Response Code: Value is 0Ch, Inject Viral.
9h	1	Version of Capability Returned
Ah	1	Length of Capability Package
Bh	1	Status <ul style="list-style-type: none"> • 00h = Success • See Table 14-47 for other error codes

14.16.3.14 Inject ALMP in Any State

Insert an ALMP in the ARB/MUX regardless of state.

Table 14-75. Inject ALMP Request

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Request Code: Value is 0Dh, Inject ALMP in any state.
9h	1	Version of Capability Requested
Ah	2	Reserved
Ch	1	<ul style="list-style-type: none"> • 00h = Disable • 01h = Insert ALMP
Dh	3	Reserved

Table 14-76. Inject ALMP Response

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Response Code: Value is 0Dh, Inject ALMP in any state.
9h	1	Version of Capability Returned
Ah	6	Reserved

14.16.3.15 Ignore Received ALMP

Ignore the next ALMPs received.

Table 14-77. Ignore Received ALMP Request

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Request Code = 0Eh, Ignore received ALMPs.
9h	1	Version of Capability Requested
Ah	2	Reserved
Ch	1	<ul style="list-style-type: none"> • 00h = Disable • 01h = Ignore ALMPs
Dh	3	Reserved

Table 14-78. Ignore Received ALMP Response

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Response Code: Value is 0Eh, Ignore received ALMPs.
9h	1	Version of Capability Returned
Ah	6	Reserved

14.16.3.16 Inject Bit Error in Flit

Inject a single bit error into the lower 16 bytes of a 528-bit flit.

Table 14-79. Inject Bit Error in Flit Request

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Request Code: Value is 0Fh, Inject Bit Error in Flit.
9h	1	Version of Capability Requested
Ah	2	Reserved
Ch	1	<ul style="list-style-type: none"> • 00h = Disable/ no action taken • 01h = Inject single Bit error in next 528 flits
Dh	3	Reserved

Table 14-80. Inject Bit Error in Flit Response

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Response Code: Value is 0Fh, Inject Bit Error in Flit.
9h	1	Version of Capability Returned
Ah	6	Reserved

14.16.3.17 Inject Memory Device Poison

Table 14-81. Memory Device Media Poison Injection Request

Data Object Byte Offset	Length in Bytes	Description
00h	8	Standard DOE Request Header
08h	1	Request Code: Value is 10h, Memory Device Media Poison Injection.
09h	1	Version of Capability Requested
0Ah	2	Reserved
0Ch	1	Protocol • 02h = CXL.mem
0Dh	1	Reserved
0Eh	1	Action • 00h = Inject Poison • 01h = Clear Poison
0Fh	1	Reserved
10h	8	Device Physical Address: When Protocol = 2, the device shall inject poison into the media at this requested address. If this address specifies a persistent memory address, the injected poison shall persist across warm resets or cold resets. Device shall report Invalid Address Specified poison injection response status if the DPA is out of range. • Bits[5:0]: Reserved • Bits[7:6]: DPA[7:6] • Bits[15:8]: DPA[15:8] • ... • Bits[63:56]: DPA[63:56]
18h	8	Clear Poison Write Data: When Protocol = 2 and Action = 1, the device shall write this replacement data into the requested physical address, atomically, while clearing poison.

Table 14-82. Memory Device Media Poison Injection Response

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Response Code: Value is 10h, Memory Device Media Poison Injection.
9h	1	Version of Capability Returned
Ah	6	Reserved

Table 14-83. Memory Device LSA Poison Injection Request (Sheet 1 of 2)

Data Object Byte Offset	Length in Bytes	Description
00h	8	Standard DOE Request Header
08h	1	Request Code: Value is 11h, Memory Device LSA Poison Injection.
09h	1	Version of Capability Requested
0Ah	2	Reserved
0Ch	1	Protocol • 02h = CXL.mem
0Dh	1	Reserved

Table 14-83. Memory Device LSA Poison Injection Request (Sheet 2 of 2)

Data Object Byte Offset	Length in Bytes	Description
0Eh	1	Action <ul style="list-style-type: none"> • 00h = Inject Poison • 01h = Clear Poison
0Fh	1	Reserved
10h	4	LSA Byte Offset: When Protocol = 2, the device shall inject poison into the Label Storage Area of the device at this requested byte offset. Because the LSA is persistent, the injected poison shall persist across warm resets or cold resets. Device shall report Invalid Address Specified poison injection response status if the byte offset is out of range. The poison can be cleared through this interface or with SetLSA.

Table 14-84. Memory Device LSA Poison Injection Response

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Response Code: Value is 11h, Memory Device LSA Poison Injection.
9h	1	Version of Capability Returned
Ah	6	Reserved

Table 14-85. Inject Memory Device Health Enable Memory Device Health Injection (Sheet 1 of 2)

Data Object Byte Offset	Length in Bytes	Description
00h	8	Standard DOE Request Header
08h	1	Request Code: 12h, Memory Device Health Injection
09h	1	Version of Capability Requested
0Ah	2	Reserved
0Ch	1	Protocol <ul style="list-style-type: none"> • 02h = CXL.mem
0Dh	1	Injection Type <ul style="list-style-type: none"> • 00h = Error is injected immediately and remains in effect until it is cleared using this command or by a CXL warm reset or cold reset of the device • 01h = Error is not injected until after a cold reset, the injection will only occur once, and will be auto-disabled after the first occurrence
0Eh	1	Valid Device Health Injection: Indicators of which Device Health Injection fields are valid in the supplied in the payload. <ul style="list-style-type: none"> • Bit[0]: <ul style="list-style-type: none"> — 1 = Health Status Injection Enabled field shall be valid • Bit[1]: <ul style="list-style-type: none"> — 1 = Media Status Injection Enabled field shall be valid • Bit[2]: <ul style="list-style-type: none"> — 1 = Life Used Injection Enabled field shall be valid • Bit[3]: <ul style="list-style-type: none"> — 1 = Dirty Shutdown Count Injection Enabled field shall be valid • Bit[4]: <ul style="list-style-type: none"> — 1 = Device Temperature Injection Enabled field shall be valid • Bits[7:5]: Reserved

Table 14-85. Inject Memory Device Health Enable Memory Device Health Injection (Sheet 2 of 2)

Data Object Byte Offset	Length in Bytes	Description
0Fh	1	<p>Enable Device Health Injection: The device shall enable the following error injection:</p> <ul style="list-style-type: none"> • Bit[0]: Health Status Injection Enabled: <ul style="list-style-type: none"> – 0 = Device shall disable its Health Status injection – 1 = Health Status field shall be valid and the device shall enable its Health Status injection • Bit[1]: Media Status Injection Enabled: <ul style="list-style-type: none"> – 0 = Device shall disable its Media Status injection – 1 = Media Status field shall be valid and the device shall enable its Media Status injection • Bit[2]: Life Used Injection Enabled: <ul style="list-style-type: none"> – 0 = Device shall disable its Life Used injection – 1 = Life Used field shall be valid and the device shall enable its Life Used injection • Bit[3]: Dirty Shutdown Count Injection Enabled: <ul style="list-style-type: none"> – 0 = Device shall disable its Dirty Shutdown Count injection – 1 = Dirty Shutdown Count field shall be valid and the device shall enable its Dirty Shutdown Count injection • Bit[4]: Device Temperature Injection Enabled: <ul style="list-style-type: none"> – 0 = Device shall disable its Device Temperature injection – 1 = Device Temperature field shall be valid and the device shall enable its Device Temperature injection • Bits[7:5]: Reserved
10h	1	Health Status: The injected Health Status. One of the defined Get Health Info values from Section 8.2.9.9 . Return Invalid Injection Parameter for invalid or unsupported injection values.
11h	1	Media Status: The injected Media Status. One of the defined Get Health Info values from Section 8.2.9.9 . Return Invalid Injection Parameter for invalid or unsupported injection values.
12h	1	Life Used: The injected Life Used. See the Get Health Info command in Section 8.2.9.9 for legal range. Return Invalid Injection Parameter for invalid or unsupported injection values.
13h	1	Reserved
14h	4	Dirty Shutdown Count: The injected Dirty Shutdown Count. See the Get Health Info command in Section 8.2.9.9 . Return Invalid Injection Parameter for invalid or unsupported injection values.
18h	2	Device Temperature: The injected Device Temperature. See the Get Health Info command in Section 8.2.9.9 . Return Invalid Injection Parameter for invalid or unsupported injection values.

Table 14-86. Device Health Injection Response

Data Object Byte Offset	Length in Bytes	Description
0h	8	Standard DOE Request Header
8h	1	Response Code: 12h, Device Health Injection
9h	1	Version of Capability Returned
Ah	1	Length of Capability Package
Bh	1	Status: See Table 14-47 for error codes.

§ §

Appendix A Taxonomy

This appendix was included in the original release of CXL and has not been updated since. It is being included as reference for some original usage and implementation options/ideas for CXL devices. It does not cover features that were added after the initial release, including Back-Invalidate Snoop (BISnp) messages that enable new ways of handling coherence of Host-managed Device Memory (HDM), and new memory device expansion proposals around pooling and Fabric-Attached memory (FAM). See [Chapter 2.0, "CXL System Architecture,"](#) for a more-complete set of use cases.

A.1 Accelerator Usage Taxonomy

Table A-1. Accelerator Usage Taxonomy (Sheet 1 of 2)

Accelerator Type	Description	Challenges and Opportunities	CXL Support
Producer-Consumer Accelerators that don't execute against "Memory" without special requirements	<ul style="list-style-type: none"> Works on data streams or large contiguous data objects Little interaction with the host Standard P/C ordering model works well 	<ul style="list-style-type: none"> Efficient work submission Efficient exchange of metadata (flow control) 	<ul style="list-style-type: none"> Basic PCIe* CXL.io
Producer-Consumer Plus Accelerators that don't execute against "Memory" with special requirements	<ul style="list-style-type: none"> Same as above, but... P/C ordering model doesn't work well Needs special data operations such as atomics 	<ul style="list-style-type: none"> Device Coherency can be used to implement varied ordering models and special data operations 	<ul style="list-style-type: none"> CXL.cache on CXL with baseline snoop filter support CXL.io
Software-assisted SVM Memory Accelerators that execute against "Memory" with software-supportable data management	<ul style="list-style-type: none"> Local memory is often needed for bandwidth or latency predictability Little interaction with the host Data management is easily implemented in software (e.g., few and simple data buffers) 	<ul style="list-style-type: none"> Host software should be able to interact directly with accelerator memory (e.g., SVM, Google, etc.) Reduces copies, replication, and pinning Optimizing coherency impact on performance is a challenge Software can provide best optimization of coherency impact 	<ul style="list-style-type: none"> CXL Bias model with software-managed bias CXL.io CXL.cache CXL.mem
Autonomous SVM Memory Accelerators that execute against "Memory" where software-supported data management is impractical	<ul style="list-style-type: none"> Local memory is often needed for bandwidth or latency predictability Interaction with the host is common Data movement is difficult to manage in software (e.g., sparse data structures, pointer-based data structures, etc.) 	<ul style="list-style-type: none"> Host software should be able to interact directly with accelerator memory (SVM) Reduces copies, replication, and pinning Optimizing coherency impact on performance is a challenge Cannot count on software for bias management 	<ul style="list-style-type: none"> CXL Bias model with hardware-managed bias CXL.io CXL.cache CXL.mem

Table A-1. Accelerator Usage Taxonomy (Sheet 2 of 2)

Accelerator Type	Description	Challenges and Opportunities	CXL Support
Giant Cache Accelerators that execute against "Memory" where local memory and caching is required	<ul style="list-style-type: none"> Local memory is needed for bandwidth or latency predictability Data footprint is larger than local memory Interaction with the host is common Data must be cycled through accelerator memory in small blocks Data movement is difficult to manage in software 	<ul style="list-style-type: none"> Accelerator memory needs to work like a cache (not SVM/system memory) Ideally, cache misses are detected in hardware, but cache replacements can be managed in software 	<ul style="list-style-type: none"> CXL.cache on CXL with "Enhanced Directory" snoop filter support CXL.io CXL.cache
Disaggregated Memory Controller Typically for memory controllers with remote persistent memory, which may be in 2 Level-Memory or App Direct mode	<ul style="list-style-type: none"> PCIe semantics are needed for device enumeration, driver support, and device management Most operational flows rely on being able to communicate directly with a Home Device or Near Memory Controller on the Host 	<ul style="list-style-type: none"> Device needs high-bandwidth and low-latency path from memory controller to Home Device in the CPU 	<ul style="list-style-type: none"> CXL.io CXL.mem

A.2 Bias Model Flow Example – From CPU

1. Start with pages in Device Bias:
 - Pages are guaranteed to not be cached in host cache hierarchy
2. Software allocates pages from device memory:
 - a. Software pushes operands to allocated pages from the peer CPU core:
 - For example, Software may use OpenCL* API to flip operand pages to Host Bias
 - Data copies or cache flushes are not required
 - b. Host CPUs generate operand data in target pages – data arrives in an arbitrary location within the host cache hierarchy.
3. Device uses operands to generate results:
 - For example, Software may use OpenCL API to flip operand pages back to Device Bias
 - a. API call causes a work descriptor submission to the device - descriptor asks the device to flush operand pages from the host cache.
 - b. Cache flush is executed using RdOwnNoData on CXL.cache protocol (see [Table 3-22](#)).
 - c. When Device Bias flip is complete, Software submits work to the accelerator.
 - d. Accelerator executes without any host-related coherency overhead.
 - e. Accelerator dumps data to results pages.
4. Software pulls results from the allocated pages:
 - For example, Software uses OpenCL API to flip results pages to Host Bias
 - This action causes some bias states to be changed, but does not cause any coherency or cache-flushing actions
 - Host CPUs can access, cache, and share results data as needed
5. Software releases the allocated pages.

OpenCL defines a Coarse-grained buffer Shared Virtual Memory model. Under that model, memory consistency is guaranteed only at explicit synchronization points and these points provide an opportunity to perform bias flip.

Here are some example of OpenCL calls where bias flip can be performed:

- `clEnqueueSVMMap` provides host access to this buffer. Software may flip the bias from Device bias to Host bias during this call.
- `clEnqueueSVMUnmap` revokes host access to this buffer. At this point, an OpenCL implementation for a CXL device could flip the bias from Host bias to Device bias.

There are other OpenCL calls where the CPU and the Device share OpenCL buffer objects. Software could flip the bias during those calls.

A.3

CPU Support for Bias Modes

There are two envisaged models of support that the CPU would provide for Bias Modes. These are described below.

A.3.1

Remote Snoop Filter

- Remote socket-owned lines that belong to accelerator-attached memory are tracked by a Remote Snoop-Filter (SF) located in the host CPU Home Agent (HA). Remote SF does not track lines that belong to Host memory. The above removes the need for a directory in device memory. Please note this is only possible in Host Bias mode since in Device Bias mode, local/remote sockets can't cache lines that belong to device memory.
- Local socket-owned lines that belong to accelerator-attached memory will be tracked by a local SF in the host CPU Last Level Cache (LLC) controller. Please note this is only possible in Host Bias mode since in Device Bias mode, local/remote sockets can't cache lines that belong to device memory.
- Device-owned lines that belong to accelerator-attached memory (in Host Bias mode) will NOT be tracked by a local SF in the host CPU LLC controller. These will be tracked by the Device Coherency Engine (DCOH) using a device-specific mechanism (device SF). In Device Bias mode, no coherent tracking is done in the host CPU because the accesses are completed within the device and the host does not see the requests.
- Device-owned lines that belong to host memory (in Host mode or Device mode) will be tracked by a local SF in the host CPU LLC controller. This may cause the device to receive snoops through CXL (CXL.cache) for such lines.

A.3.2

Directory in Accelerator-attached Memory

- Remote socket-owned lines that belong to device memory are tracked by a directory in device memory metadata. The host HA may choose to do broadcast snooping for some cases to avoid reading the metadata.
- Local socket-owned lines that belong to device memory will be tracked by a local SF in the host CPU LLC controller. For access by device, local socket-owned lines that belong to device memory will also update the directory.
- Device-owned lines that belong to device memory will NOT be tracked by a local SF in the host CPU LLC controller. These will be tracked by the Device Coherency Engine (DCOH) using a device-specific mechanism (device SF).
- Device-owned lines that belong to host memory (in Host mode or Device mode) will be tracked by a local SF in the host CPU LLC controller. This may cause the device to receive snoops through CXL (CXL.cache) for such lines.

A.4

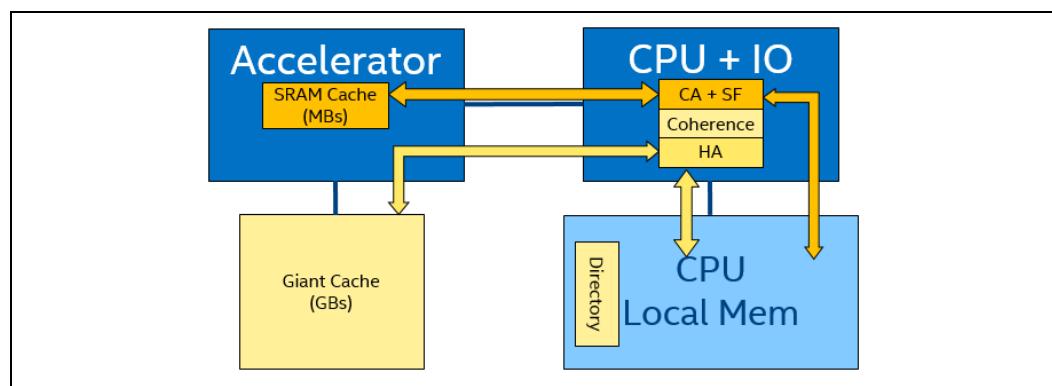
- Bias Table is located in stolen memory within the device memory and is accessed through the DCOH.

Giant Cache Model

For problems whose data sets exceed the device-attached memory size, the memory attached to the accelerator needs to be a cache, not memory:

- Typically, the full data set will reside in processor-attached memory
- Subsets of this larger data set are cycled through accelerator memory as the calculation proceeds
- For such use cases, caching is the correct solution:
 - Accelerator memory is not mapped into the system address map – data set is built up in host memory
 - Single-page table entry per page in data set – no page table manipulation as pages are cycled through accelerator memory
 - Copies of data can be created under driver control and/or hardware control with no OS intervention

Figure A-1. Profile D - Giant Cache Model



Critical issues with a Giant Cache:

- Cache is too large for tracking in the Host on-die SF
- Snoop latency for a Giant Cache is likely to be much higher than standard on-die cache snoop latency

Recommended CXL solution:

- Implements SF in processor's coherency directory (stored in DRAM ECC bits), which essentially becomes a highly scalable SF
- Minimizes impact to processor operations that are unrelated to accelerators
- Allows accelerator to access data over CXL.cache as a caching Device
- Provides support on CXL.cache to allow an accelerator to explicitly request directory snoop filtering for Giant Cache
- Processor infrastructure differentiates between low-latency and high-latency requester types
- Support for simultaneous use of a small, low-latency cache associated with the on-die snoop filter is included

§ §

Appendix B Unordered I/O to Support Peer-to-Peer Directly to HDM-DB

Revision 3.0 of the CXL specification added this appendix to document the planned CXL use of Unordered I/O (UIO) protocol for the CXL UIO Direct P2P to HDM-DB use case. At the time, UIO protocol was still under development within the PCI SIG as an ECN against the PCIe* Base specification. The UIO ECN became final in March 2023, and is available from the PCI-SIG website.

The CXL use of UIO protocol as originally described in this appendix as well as additional use cases are covered elsewhere within this version of the CXL specification. UIO protocol details should be obtained from the final UIO ECN or other PCI-SIG specifications.

§ §

Appendix C Memory Protocol Tables

To formalize the protocol expectations of the Memory Protocol, this appendix captures the allowed request encodings, states, and responses in the host and the device. In this appendix, the flows are referred to in the context of 3 HDM region types as defined in [Section 3.3](#).

This appendix uses field name abbreviations to fit into the table format captured in [Table C-1](#).

Table C-1. Field Encoding Abbreviations (Sheet 1 of 2)

Field Name	Encoding Name	Abbreviation	Notes
MetaField	Meta0-State	MS0	
	Extended Meta-State	EMS	
	Meta0-State OR No-Op	MS0-NO	
	Extended Meta-State OR No-Op	EMS-NO	For table rows that allow these two encodings in a DRS response, the No-Op encoding should be sent only when the device has an uncorrectable error on the Extended Metadata Stored such that it cannot return it.
DRS Trailer and Rwd Trailer	Extended MetaValue	EMV	Extended MetaValue in the trailer. For DRS Trailer, this state comes from EMD state stored in the device at the time the request is received. For Rwd Trailer, this is the state to become the final EMD state when the write is complete.
	Not Applicable	N/A	Indicates that no trailer is returned
	Extended MetaValue or Not Applicable	EMV-NA	Indicates that a trailer of EMV is sent, but only if the MetaField encoding is EMS. The table may allow for No-Op to be sent in MetaField, in which case no trailer is included.
MetaValue	Not Applicable	N/A	Used when MetaValue is undefined and should never be consumed, which is when MetaField is set to No-Op. The encoding rules follow the definition for Reserved which must be cleared to 0 by sender and ignored by receiver.
	Any Value	<Any>	Indicates any legal encoding that matches rules for the HDM memory region. In support of Extended Metadata with HDM-DB, the device must return the current host state (same as Meta0-State).
	Explicit No-Op	E-No-Op	This is an encoding defined for use only when EMV encoding to indicate the MetaField= No-Op equivalent coherent behavior in the device for HDM-DB.
Host State	UnChanged	UC	Used to indicate that the device should not change its bias state.

Table C-1. Field Encoding Abbreviations (Sheet 2 of 2)

Field Name	Encoding Name	Abbreviation	Notes
SnpType	Snp*	All Snoop types: SnpInv, SnpData, SnpCur	Used when a row applies to all different snoop options.
Device State	Extended Metadata	EMD	Data stored in the device for up to 32 bits of meta state. This mode is mutually exclusive with standard Meta0-State in HDM-H devices, but supplemental to coherence state in HDM-DB devices. Not applicable to HDM-D.

Note:

The term "Host State" is used in this section to describe the device tracking of the host coherence state for the address. The implementation of this may vary and has been referred to as a "Bias Table" or "Bias State" or "Directory" for HDM-D/HDM-DB. For HDM-DB more implementations are possible, including an inclusive Snoop-Filter, which use the BISnp/BIRsp channels to keep the Snoop Filter inclusive.

In the tables that follow, "Y(1)" in the Legal column indicates that the table row contents are also defined in other tables within the CXL specification and also existed in the CXL 1.1 specification. A "Y*" in the Legal column indicates that a destination must accept the message, and the rows of the table indicate the legal responses to the message. The "O" defines optional rows that may not be supported. All "O" cases will include details as to the discovery/enabling method in [Table C-2](#).

Table C-2. Optional Codes

Option Code	Option Name	Description
O-1	Cmp-M Support	Mode that can be enabled in a device. This message is optional for the host to support and is thus expected to be applied to CXL.mem traffic from a given initiator (host or peer device). For MLD/GFD, this mode could be different for each initiator.
O-2	EMD Support	This mode would apply to a given HDM region. Hosts are expected to support EMD or Traditional Metadata for each region of HDM-H. For HDM-DB, the MetaValue is treated in the same way (defining coherence state); however, the EMD value is also supported.
O-3	MemSpecRd Support	These messages are supported only on interfaces that train as 256B or that are supported by links that train as 256B Flit mode or "CXL 68B Flit and VH Capable".
O-4	TEE Requests	TSP features are enabled in the host.

C.1

HDM-D and HDM-DB Requests

Table C-3 defines messages on the request channel of CXL.mem protocol. **Table C-4** additionally defines the Forward flows that apply only for HDM-D memory regions for Type 2 devices when the devices are accessing device-attached memory. **Table C-5** defines the BISnp channel method of managing device-attached memory coherence for the HDM-DB memory region of Type 2 devices or Type 3 devices.

Table C-3. HDM-D/HDM-DB Memory Requests (Sheet 1 of 4)

Legal	Host Request				Device Response					Final Device State			Description		
	M2S Req	MetaField	MetaValue	SnpType	S2M NDR	S2M DRS	MetaField	MetaValue	DRS Trailer	Device Cache	Host State	EMD			
O-1	MemRd	MS0	A	SnpInv	Cmp-M	MemData	MS0-NO	<any>	N/A	I	A	UC	Supported for host that can accept M-state.		
O-1 & O-2		EMS					EMS-NO		EMV-NA						
Y(1)		MS0					MS0-NO		N/A						
O-2		EMS			Cmp-E		EMS-NO		EMV-NA						
N													The host wants an exclusive copy of the cacheline.		
N				SnpData											
N															
N															
N															
Y(1)		MS0													
O-2		EMS		SnpData	Cmp-S		MemData	<any>	N/A	S	S	UC	The host is requesting a shared copy of the cacheline, but Rsp types allow the device to return S-state or E-state to the host. Cmp-E response is not recommended because the host did not request this state.		
Y(1)		MS0							EMV-NA						
O-2		EMS			Cmp-E				N/A	I	A				
N									EMV-NA						
N															
Y	MemWr	MS0	I	SnpInv	Cmp	MemData	MS0-NO	<any>	N/A	I	I	UC	The host is requesting a non-cacheable but current value of the cacheline and is forcing the device to flush its cache.		
O-2		EMS							EMV-NA						
N				SnpData											
Y		MS0											The host is requesting a non-cacheable but current value of the cacheline and is thereby leaving data in the device's cache.		
O-2		EMS													
N				No-Op											

Table C-3. HDM-D/HDM-DB Memory Requests (Sheet 2 of 4)

Legal	Host Request				Device Response					Final Device State			Description	
	M2S Req	MetaField	MetaValue	SnpType	S2M NDR	S2M DRS	MetaField	MetaValue	DRS Trailer	Device Cache	Host State	EMD		
Y(1)	MemRd	No-Op	N/A	SnpInv	Cmp	MemData	MS0-NO	<any>	N/A	I	UC	UC	The host wants to read the cacheline without changing the state expected in the host cache, and the device should invalidate the cacheline from its cache.	
O-2		EMS	No-Op				EMS-NO		EMV-NA					
N		EMS-NO	N/A	SnpData										
N		EMS	E-No-Op											
Y(1)		No-Op	N/A	SnpCur	Cmp		MS0-NO	<any>	N/A	<any>	UC	UC	The host wants a current value of the cacheline without changing the state expected in the host cache.	
O-2		EMS	E-No-Op				EMS-NO		EMV-NA					
Y		No-Op	N/A	No-Op	Cmp		MS0-NO	<any>	N/A	<any>	UC	UC	The host wants the value of the memory location without snooping the device cache and without changing the cache state expected in the host cache. A use case for this would be if the host includes E-state or S-state without data so that the host is requesting data only and doesn't want to change the cache state, and because the host has E-state or S-state, the host can know that the device cache does not need to be snooped.	
O-2		EMS	E-No-Op				EMS-NO		EMV-NA					
Y		<all>	<all>	<all>	<none>		MemData-NXM	No-Op	N/A	N/A	N/A	N/A	The special case MemData-NXM response is used if the fabric or the device is unable to positively decode the Address. This is a common response type for both Type 2 devices and Type 3 devices to avoid an ambiguous case in which the host is unsure of whether the host should expect an NDR message.	

Memory Protocol Tables

Table C-3. HDM-D/HDM-DB Memory Requests (Sheet 3 of 4)

Legal	Host Request					Device Response					Final Device State			Description
	M2S Req	MetaField	MetaValue	SnpType	S2M NDR	S2M DRS	MetaField	MetaValue	DRS Trailer	Device Cache	Host State	EMD		
Y(1)	MemInv/ MemInvNT	MS0	A	SnpInv	Cmp-E	<none>	<any>	<any>	N/A	I	A	UC	The host wants ownership of the cacheline but does not require the data.	
N				SnpData										
N				SnpCur										
N				No-Op										
N			S	SnpInv						S or I	S	UC	The host wants the device to degrade to S-state in its caches, and wants the shared state for the cacheline (but does not require the data).	
Y				SnpData	Cmp-S		<any>	<any>						
N				SnpCur										
N				No-Op										
Y(1)			I	SnpInv	Cmp		<any>	<any>		I	I	UC	The host wants the device to invalidate the cacheline from its caches and does not require the data.	
N				SnpData										
N				SnpCur										
N				No-Op										
Y			No-Op	SnpInv	Cmp		<any>	<any>		I	UC	UC	The host wants the device to invalidate the cacheline from its caches and does not require the data.	
N				SnpData										
N				SnpCur										
N				No-Op										
N			EMS	N/A										

Memory Protocol Tables

Table C-3. HDM-D/HDM-DB Memory Requests (Sheet 4 of 4)

Legal	Host Request					Device Response					Final Device State			Description	
	M2S Req	MetaField	MetaValue	SnpType	S2M NDR	S2M DRS	MetaField	MetaValue	DRS Trailer	Device Cache	Host State	EMD			
N	MemRdData	<all>	N/A	SnpInv		MemData							The host wants a cacheable copy in either E-state or S-state.		
Y(1)				MS0-NO	Cmp-E		MS0	I or A	N/A	I	A				
O-2				EMS	SnpData		EMS		EMV						
Y(1)				MS0-NO			MS0	S	N/A	I or S	S				
O-2				EMS			EMS		EMV						
Y				Cmp-E			No-Op	N/A	N/A	I	A				
Y				Cmp-S						I or S	S				
N				SnpCur											
N				No-Op											
Y				<all>	<all>		MemData-NXM	No-Op	N/A	N/A	UC	UC	The special case MemData-NXM response is used if the fabric or the device is unable to positively decode the Address. This is a common response type for both Type 2 devices and Type 3 devices to avoid an ambiguous case in which the host is unsure of whether the host should expect an NDR message.		
N	MemSpecRd	No-Op	N/A	MS0-EMD	<all>	MemData									
N					Snp*										
O-3				No-Op	<none>		No-Op	<none>	<none>	<none>	N/A	UC	UC	Speculative memory read. A Demand read following this with the same address will be merged in the device. No completion is expected for this transaction. Completion is returned with demand read.	
N				A or S	<all>										
Y				MS0	I		No-Op	Cmp	<none>	No-Op	N/A	N/A	I	The host will only issue from E-state or from S-state.	
N							Snp*								
N	MemClnEvct	EMD-NO	N/A	<all>											
N		MemRdTEE/ MemRdDataTEE/ MemSpecRdTEE											TEE is not supported for HDM-DB/HDM-D in this revision of the specification.		
Sub-Table ¹		MemRdFwd	See Table C-4.												

1. Applicable only to HDM-D memory regions.

C.1.1 Forward Flows for HDM-D

Table C-4 shows the flows that can generate Mem*Fwd messages on CXL.mem from CXL.cache requests. These flows are triggered when a device issues a D2H Request to an address that is mapped with its own CXL.mem address region. This region is referred to as device-attached memory.

Note:

The HDM-D region is intended to be used only for cases in which the device or host does not support 256B Flit mode. In cases where the host, switch, and device supports 256B Flit mode, the HDM-DB must be used.

Table C-4. HDM-D Request Forward Sub-table (Sheet 1 of 2)

Legal	Device Request	Host Response on M2S Req					Final Device State		Description
		D2H Req	M2S Req	MetaField	MetaValue	SnpType	Device Cache	Host State	
Y	RdCurr	MemRdFwd	MS0	A			I	A	MemRdFwd will reflect the final cache state of the host and for RdCurr it is recommended that the host not change the cache state.
Y				S				S	
Y				I				I	
N			<all>	Snp*					
N				EMS-NO	N/A	<all>			
N			MS0	A					
Y				S			No-Op	S	The host must only be in S-state or I-state.
Y				I				I	
N				<all>	Snp*				
N				EMS-NO	N/A	<all>			
N				A					
Y	RdAny	MemRdFwd	MS0	S			No-Op	S	The host must be in S-state or I-state.
Y				I				E	
N				<all>	Snp*				
N				EMS-NO	N/A	<all>			
N			MS0	A			No-Op		
N				S					
Y				I				E	
N	RdOwn/ RdOwnNoData	MemRdFwd	<all>	Snp*					The host must be in I-state.
N				EMS-NO	N/A	<all>			

Table C-4. HDM-D Request Forward Sub-table (Sheet 2 of 2)

Legal	Device Request	Host Response on M2S Req					Final Device State		Description
	D2H Req	M2S Req	MetaField	MetaValue	SnpType	Device Cache	Host State		
N	CLFlush	MemRdFwd	MS0	A	No-Op				
N				S					
Y				I		I	S	The host must indicate invalid, but the device must assume that S-state is possible in the host as part of the CLFlush definition.	
N			EMS-NO	<all>	Snp*				
N				N/A	<all>				
N									
N	WOWrInv/ WOWrInvF	MemWrFwd	MS0	A	No-Op				
N				S					
Y				I		NC	I	The host must be in I-state.	
N			EMS-NO	<all>	Snp*				
N				N/A	<all>				
N									
N	CleanEvict/ DirtyEvict/ CleanEvictNoData	N/A	<all>	<all>	<all>			Messages are not sent to the host for device-attached memory.	
N	ItoMWr/ WrCur/ WrInv/ CacheFlushed	None	<all>	<all>	<all>			Standard CXL.cache flows are used for these requests.	

C.1.2

BISnP for HDM-DB

Table C-5 shows the flows that the device can generate. These flows are only supported to an address that is mapped in the devices own CXL.mem address region. This region is referred to as device-attached memory.

Table C-5. HDM-DB BISnP Flow (Sheet 1 of 2)

Legal	Device Request	Host Response	Final Device State			Description	
			S2M BISnP	M2S BIRsp	Device Cache	Host State ¹	
Y	BISnPCur	BIRspI	I		I		Used when the device is requesting a current copy of the cacheline but is not installing the data into its cache. Block responses are not supported.
Y		BIRspS			S		
Y		BIRspE			A		
N		BIRsp*Blk					
Y	BISnPData	BIRspI	E, S, or I	I			Used when the device is requesting Shared or Exclusive state of the cacheline.
Y		BIRspS	S or I	S			
N		BIRspE					
N		BIRsp*Blk					
Y	BISnPInv	BIRspI	E S, or I	I			Used when device is requesting Exclusive state of the cacheline including capacity evictions for device Inclusive Snoop Filter evictions. The device may choose to install only in its cache in S-state for device-specific reasons.
N		BIRspS					
N		BIRspE					
N		BIRsp*Blk					
Y	BISnPCurBlk	BIRspI	I		I		Used when device is requesting a current copy of a multiple line block but is not installing into its cache. Response is per cacheline.
Y		BIRspS			S		
Y		BIRspE			A		
Y		BIRspIBlk			I - Blk		
Y		BIRspSBlk			S - Blk		
Y		BIRspEBlk			A - Blk		
Y	BISnPDataBlk	BIRspI	E, S, or I	I			Used when device is requesting a Shared or Exclusive copy of a multiple cacheline block. Response is per cacheline.
Y		BIRspS	S or I	S			
N		BIRspE					
Y		BIRspIBlk	E, S, or I	I - Blk			
Y		BIRspSBlk	S or I	S - Blk			
N		BIRspEBlk					

Table C-5. HDM-DB BISnp Flow (Sheet 2 of 2)

Legal	Device Request	Host Response	Final Device State			Description	
			S2M BISnp	M2S BIRsp	Device Cache	Host State ¹	EMD
Y	BISnpInvBlk	BIRspI	E, S, or I	I	UC	Used when device is requesting an Exclusive copy of a multiple cacheline block. Response is per line.	
N		BIRspS					
N		BIRspE					
Y		BIRspIBlk	E, S, or I	I - Blk	UC	Used when device is requesting an Exclusive copy of a multiple cacheline block. Response is for the entire block.	
N		BIRspSBlk					
N		BIRspEBlk					

1. Host State (aka Bias State) tracking in the device can be implemented with a full Table or inclusive SF.

C.2

HDM-H Requests

HDM-H is used only by Type 3 devices for simple memory expansion. Requests from the host to this region always encode SnpType as No-Op and the host can also make arbitrary use of the MetaValue field to store 2-bit encodings, which the device should not interpret. Table C-6 captures the flows for these devices.

Table C-6. HDM-H Memory Request (Sheet 1 of 2)

Legal	Host Request				Device Response					Device State	Description
	M2S Req	MetaField	MetaValue	SnpType	S2M NDR	S2M DRS	MetaField	MetaValue	DRS Trailer		
N	<all>	<all>	<all>	Snp*							Snoop encodings are never sent to the HDM-H address region.
Y	MemRd/ MemRdTEE ¹	MS0	<any 2-bit value>	No-Op	<none>	MemData/ MemDataTEE ¹ /MemData-NXM ²	<any>	<any>	N/A	MetaValue	Read that is requesting Meta State field updates to new values.
Y		No-Op	N/A			MemData/ MemDataTEE ¹	EMS-NO	N/A		UC	Read that does not expect a Meta State field update.
O-2		EMS	N/A			MemData-NXM ²	No-Op	N/A	EMV-NA	UC	EMS in DRS indicates EMV trailer is returned. No Metadata is changed as a result of this message.
O-2	MemInv/ MemInvNT	MS0	<any 2-bit value>	Cmp	<none>	<any>	<any>	N/A	UC	MetaValue	NXM response must only return No-Op MetaField for decode errors of requests with MetaField of EMS.
Y		No-Op	N/A							MetaValue	The host wants to read the Meta State field and update it.
Y		No-Op	N/A						UC	UC	The host wants to read the Meta State field but does not want to update it.
N	MemRdData/ MemRdTEE ¹	MS0 or EMS		No-Op	No-Op	MemData/ MemDataTEE ¹ /MemData-NXM ²	MS0	I or A	N/A	A	Used for implicit directory state updates in the HDM-H address region. This is the only case in which devices with HDM-H must decode the MetaValue field.
Y								S		S	
Y							No-Op	N/A		N/A	Used for devices that do not store the MetaValue field or if the MetaValue field is corrupted.
N		MS0 or EMS	<all>								
N	MemSpecRd/ MemSpecRdTEE ¹	Snp*		No-Op	<none>	<none>	<none>	<none>	N/A	UC	Speculative memory read. A demand read that follows this with the same address will be merged in the device. No completion is expected for this transaction. Completion is returned with a demand read.
O-3		No-Op	N/A								

Table C-6. HDM-H Memory Request (Sheet 2 of 2)

Legal	Host Request				Device Response					Device State	Description
	M2S Req	MetaField	MetaValue	SnpType	S2M NDR	S2M DRS	MetaField	MetaValue	DRS Trailer		
N	MemClnEvct										
N	MemRdFwd										
N	MemWrFwd										
O-4	TEUpdate	MS0	00b (clear TE)	<Length>	Cmp	<none>	No-Op	N/A	N/A	UC	Commands used to set or clear TE state for region of memory. The TE is set when MetaValue is 01b and cleared when MetaValue is 00b. This command is optionally supported by devices that support TSP.
O-4			01b (set TE)								
N			10b or 11b								
N			No-Op or EMS								

1. TEE opcodes are optionally enabled.
2. Returned when the Switch or Device cannot positively decode that the address is HDM-H vs. HDM-D* address region. This response allows the host to complete the request by converging the flows for this Decode Miss error between the HDM-H and HDM-D*.

C.3

HDM-D/HDM-DB RwD

Table C-7 captures the Request with Data (RwD) flows for HDM-D/HDM-DB address regions used by Type 2 devices and Type 3 devices.

Table C-7. HDM-D/HDM-DB Memory RwD

Legal	Host Request					S2M NDR	Device Response			Device State			Description
	M2S RwD	MetaField	MetaValue	SnpType	RwD Trailer		S2M DRS	MetaField	MetaValue	Dev Cache	Host State	EMD ¹	
N	MemWr/ MemWrPtl	MS0-EMS	A	Snp*			No-Op	N/A EMV	Cmp	I	A	UC EMV	Snoop encodings are never sent with A-state because the host must have an exclusive copy of the cacheline.
Y(1)				MS0									The host wants to update memory and keep an exclusive copy of the cacheline.
O-2				EMS									
N		MS0-EMS	S	Snp*			No-Op	N/A EMV	Cmp	I	S	UC EMV	The host wants to update memory and keep a shared copy of the cacheline.
Y				MS0									
O-2				EMS									
Y(1)		MS0	I	SnpInv ²	N/A	Cmp	No-Op	N/A	I	I	UC	The host wants to write the cacheline back to memory and does not retain a cacheable copy. In addition, the host was not granted ownership of the cacheline before performing this write and needs the device to invalidate the device's caches before performing the writeback to memory.	
O-2		EMS			EMV								
N		MS0-EMS		SnpData									No use case.
N		MS0-EMS	I	SnpCur			No-Op	N/A EMV	Cmp	I	I	UC EMV	The host wants to update memory and will end with host caches in I-state. Use is for Dirty (M-state) Cache Evictions in the host.
Y(1)				MS0									
O-2				EMS									
N		MS0-EMS	<all>	<all>		Cmp	MS0	<all>	<any>	<any>			The device never returns the Meta State field for a write.
N		No-Op	N/A	<all>									
N		EMD	E-No-Op	<all>									The host must always define the MetaValue field for writes.
Y	BICConflict	No-Op	N/A	No-Op	N/A	BICConflict Ack	No-Op	N/A	UC	UC	UC		Used as part of the BISnp conflict handling flow.
N				Snp*									
N		MS0-EMS	<all>	<all>									
N	MemRdFill/ MemWrTEE/ MemWrPtlTEE/ MemRdFillTEE/ TEUpdate												TEE is not supported for HDM-DB or HDM-D in this revision of the specification.

1. Applicable only with EMD enabled.

2. Snoopable Writes are not supported in Shared FAM use cases.

C.4

HDM-H RwD

Table C-8 captures the Request with Data (RwD) flows for the HDM-H memory region.

Table C-8. HDM-H Memory RwD

Legal	Host Request					Device Response				Device State	Description
	M2S RwD	MetaField	MetaValue	SnpType	RwD Trailer	S2M NDR	S2M DRS	MetaField	MetaValue	Metadata/EMD ¹	
N	MemWr/ MemWrPtl/ MemWrTEE ² / MemWrPtIEE ²	MS0-EMS	<call>	Snp*			<none>				SnpType encodings are never sent to the HDM-H.
Y		MS0	<any 2-bit value>	No-Op	N/A			No-Op	N/A	MetaValue	The host wants to update memory. The host sets a value in the MetaValue field. The device optionally stores that MetaValue field value.
O-2		EMS	N/A		EMV	Cmp CmpTEE ²			EMV		Note: MemWrPtl is not expected to be used with host-side encryption.
N				No-Op		Cmp		MS0	<any>		The host never needs the Meta State field value returned from a write.
N		No-Op	N/A	<all>							The host always sends MS0 to avoid the need for a Read-modify-write in the device for the MetaValue field.
N		BIConflict	<all>	<all>	<all>						This message is not used for the HDM-H memory region.
O-4	MemRdFill/ MemRdFilIEE ²	No-Op	N/A	No-Op	N/A	<none>	MemData/ MemDataTEE ²	<any>	<any>	UC	Support for this command is required by devices that support TSP to support host-side memory encryption. The result of this read does not change meta-state and returns only data to the host. The use case is expected for partial write merging that is not possible in the device with host-side encryption.
O-4											
N				Snp*							
N		MS0	<all>								

1. EMD is applicable only if enabled.

2. TEE opcodes are optionally enabled.

§ §