



---

# Core Specification

---

Version 1.0

## DISCLAIMER

This document is provided 'as is' with no warranties whatsoever, including any warranty of merchantability, noninfringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. Gen-Z Consortium disclaims all liability for infringement of proprietary rights, relating to use of information in this document. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

Gen-Z is a trademark or registered trademark of the Gen-Z Consortium.

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

Copyright 2016-2018 by Gen-Z Consortium. All rights reserved.

# Contents

*This specification was created by developers for developers. The specification makes extensive use of cross references and fine-grain section headings to enable developers to quickly locate information and traverse the specification.*

*To quickly develop a solid understanding of Gen-Z architecture, developers need to read the Introduction and Core Architecture and Features first. With the basic architecture understood, the sequence to read material will depend upon the developer's focus, for example:*

- *Developers focused on basic memory-semantic communications, need to read sub-sections Read and Read Response and Write to understand Read and Write operations. From there read the solution-applicable sub-sections in Common End-to-end Operations. If a solution requires advanced operations, then read applicable sub-sections in Advanced End-to-end Operations.*
- *Developers focused on management, need to read sub-sections Configuration and Management Responsibilities through Control Structure Organization, then Control Space In-Band Access Operations. From there, it depends upon the specific management services being developed. Note Configuration and Management accounts for nearly half of the specification. Most of this material is organized into tables that specify the control structures used for configuration and management.*

*Developers should consult Document Organization which provides a high-level description of each section.*

<b>CONTENTS.....</b>	<b>2</b>
<b>FIGURES.....</b>	<b>12</b>
<b>TABLES.....</b>	<b>22</b>
<b>1. CORE ARCHITECTURE .....</b>	<b>30</b>
1.1. FUNCTIONAL OVERVIEW .....	30
1.2. OPERATIONAL OVERVIEW .....	32
1.3. MEMORY-SEMANTIC COMPONENTS .....	34
1.4. SPLIT MEMORY / MEDIA CONTROLLER MODEL.....	35
1.5. TOPOLOGIES.....	38
1.6. SWITCH OVERVIEW .....	40
1.7. TRANSPARENT ROUTER OVERVIEW .....	42
1.8. LATENCY DOMAINS.....	43
1.9. PROCESSOR-CENTRIC AND MEMORY-CENTRIC SOLUTIONS .....	46
1.10. PHYSICAL LAYER INDEPENDENCE.....	48
1.11. TIMER FORMULA INTERPRETATION .....	48
1.12. DOCUMENT ORGANIZATION .....	49
1.13. REFERENCE DOCUMENTS .....	50
1.14. DOCUMENTATION CONVENTIONS.....	50
<b>2. GLOSSARY.....</b>	<b>54</b>
<b>3. CORE FUNCTIONALITY .....</b>	<b>69</b>

3.1. COMMUNICATIONS.....	69
3.2. FEATURES .....	70
3.2.1. <i>Packet Types</i> .....	70
3.2.2. <i>Address</i> .....	71
3.2.3. <i>Virtual Channels</i> .....	72
3.2.4. <i>Component Identifiers</i> .....	75
3.2.5. <i>Global Component Identifiers</i> .....	75
3.2.6. <i>End-to-end Unicast Communications</i> .....	75
3.2.7. <i>End-to-end Multicast Communications</i> .....	76
3.2.8. <i>Accelerators</i> .....	77
3.2.9. <i>Access Keys</i> .....	81
3.2.10. <i>Region Key (R-Key)</i> .....	83
3.2.11. <i>Multipath</i> .....	86
3.2.12. <i>Vendor-defined Operations</i> .....	86
3.2.13. <i>Statistics</i> .....	87
3.2.14. <i>Daisy-Chain Topologies</i> .....	87
3.2.15. <i>Memory Media RAS</i> .....	92
3.3. CAPABILITIES-BASED RESOURCE ACCESS CONTROL.....	102
3.4. MEMORY MANAGEMENT .....	105
3.4.1. <i>ZMMU Page Table Structure</i> .....	109
3.4.2. <i>ZMMU PTE Caching</i> .....	113
3.4.3. <i>Memory Interleave</i> .....	114
3.4.4. <i>Requester PTE</i> .....	119
3.4.5. <i>Responder PTE</i> .....	125
<b>4. END-TO-END OPCLASSES &amp; OPCODES .....</b>	<b>128</b>
4.1. END-TO-END OPERATION CLASSES .....	128
4.2. END-TO-END PACKET OPERATION CODES .....	129
4.2.1. <i>P2P-Core OpClass Operation Codes</i> .....	130
4.2.2. <i>P2P-Coherency OpClass Operation Codes</i> .....	134
4.2.3. <i>Core 64 OpClass Operation Codes</i> .....	137
4.2.4. <i>Control OpClass Operation Codes</i> .....	138
4.2.5. <i>Atomic 1 OpClass Operation Codes</i> .....	139
4.2.6. <i>LDM 1 OpClass Operation Codes</i> .....	141
4.2.7. <i>Advanced 1 OpClass Operation Codes</i> .....	142
4.2.8. <i>Advanced 2 OpClass Operation Codes</i> .....	142
4.2.9. <i>Multicast OpClass Operation Codes</i> .....	143
4.2.10. <i>SOD OpClass Operation Codes</i> .....	143
4.2.11. <i>CTXID OpClass Operation Codes</i> .....	144
4.3. END-TO-END OPCLASSES AND OP CODE SETS .....	145
<b>5. BASE FORMATS FOR END-TO-END PACKETS .....</b>	<b>146</b>
5.1. END-TO-END GENERAL PACKET PROTOCOL FORMAT .....	146
5.2. TAG FIELD.....	154

5.3.	LPD FIELD .....	155
5.4.	EXPLICIT OPCLASS R-KEY FIELD .....	159
5.5.	EXPLICIT OPCLASS NEXT HEADER FIELD .....	159
5.6.	EXPLICIT OPCLASS MULTI-SUBNET FIELD .....	160
<b>6.</b>	<b>COMMON END-TO-END OPERATIONS .....</b>	<b>162</b>
6.1.	READ AND READ RESPONSE.....	162
6.1.1.	READ FEATURES & REQUIREMENTS .....	164
6.1.2.	READ RESPONSE FEATURES & REQUIREMENTS.....	168
6.2.	WRITE .....	171
6.3.	WRITE MSG.....	178
6.4.	WRITE POISON .....	188
6.5.	WRITE-UNDER-MASK (WUM) .....	190
6.6.	WRITE PARTIAL.....	191
6.7.	META READ RESPONSE AND META WRITE .....	194
6.8.	STANDALONE ACKNOWLEDGMENT .....	196
6.8.1.	<i>Responder Not Ready (RNR) NAK</i> .....	202
6.8.2.	<i>Forward Progress Screen (FPS)</i> .....	203
6.9.	PERSISTENT FLUSH .....	205
6.10.	CONTROL SPACE IN-BAND OPERATIONS .....	206
6.10.1.	<i>P2P-Core and P2P-Coherency In-Band Operations</i> .....	206
6.10.2.	<i>Control OpClass In-Band Operations</i> .....	211
6.10.3.	<i>Unsolicited Event (UE) Packet</i> .....	214
6.10.4.	<i>No-Op</i> .....	225
6.10.5.	<i>C-State Power Control</i> .....	226
6.10.6.	<i>R-Key Update</i> .....	230
6.10.7.	<i>Control Write MSG</i> .....	231
<b>7.</b>	<b>ADVANCED END-TO-END OPERATIONS.....</b>	<b>233</b>
7.1.	INTERRUPTS .....	233
7.2.	ATOMIC REQUEST AND RESPONSE PACKETS.....	238
7.2.1.	<i>Atomic Request Types</i> .....	241
7.2.2.	<i>Atomic Packet Formats</i> .....	244
7.3.	BUFFER REQUESTS.....	255
7.3.1.	<i>Buffer Features &amp; Requirements</i> .....	263
7.3.2.	<i>Buffer Request Types</i> .....	265
7.4.	PATTERN REQUESTS.....	277
7.4.1.	<i>Pattern Features &amp; Requirements</i> .....	278
7.4.2.	<i>Pattern Request Types</i> .....	278
7.5.	MULTI-OP REQUESTS .....	281
7.6.	VENDOR-DEFINED PACKETS .....	293
7.7.	NON-IDEMPOTENT REQUEST RELEASE (NIRR).....	297
7.8.	PRECISION TIME.....	299
7.9.	LIGHTWEIGHT NOTIFICATION (LN) .....	306

7.10. WAKE THREAD .....	313
7.11. CAPABILITIES READ .....	314
7.12. CAPABILITIES WRITE .....	315
7.13. UNICAST ENCAPSULATION PACKETS .....	315
7.14. COHERENCY-SPECIFIC OPERATIONS .....	316
7.14.1. <i>Read Exclusive</i> .....	317
7.14.2. <i>Read Shared</i> .....	318
7.14.3. <i>Release</i> .....	320
7.14.4. <i>Exclusive</i> .....	321
7.14.5. <i>Cache Line Attribute</i> .....	321
7.14.6. <i>Invalidate and Writeback</i> .....	325
7.15. COLLECTIVE OPERATIONS .....	330
7.15.1. <i>Barrier Collective</i> .....	335
7.15.2. <i>Broadcast Collective</i> .....	337
7.15.3. <i>Scatter Collective</i> .....	338
7.15.4. <i>Gather Collective</i> .....	340
7.15.5. <i>All-Gather Collective</i> .....	342
7.15.6. <i>Reduce Collective</i> .....	344
7.15.7. <i>All-Reduce Collective</i> .....	346
7.15.8. <i>Map-Reduce Collective</i> .....	348
7.15.9. <i>All-Map-Reduce Collective</i> .....	349
7.15.10. <i>Collective Result</i> .....	351
7.15.11. <i>Collective Abort</i> .....	353
7.15.12. <i>Collective Abort All</i> .....	353
7.15.13. <i>Collective Responder Count Response</i> .....	353
<b>8. CONFIGURATION AND MANAGEMENT .....</b>	<b>355</b>
8.1. CONFIGURATION AND MANAGEMENT RESPONSIBILITIES .....	355
8.2. COMPONENT MANAGEMENT .....	355
8.3. INTERFACE STATES AND DESCRIPTIONS .....	355
8.3.1. <i>Interface State Machine Terms</i> .....	357
8.4. COMPONENT STATES AND DESCRIPTIONS .....	357
8.4.1. <i>Component State Machine Terms</i> .....	360
8.5. OUT-OF-BAND MANAGEMENT .....	361
8.6. IN-BAND MANAGEMENT .....	362
8.6.1. <i>In-band Discovery and Initialization</i> .....	362
8.7. CONFIGURATION COMPLETION TIMER AND RELEASE .....	366
8.8. POWER MANAGEMENT .....	366
8.8.1. <i>Physical Layer Power Management</i> .....	367
8.8.2. <i>Component Power Management</i> .....	367
8.8.3. <i>C-State Transition and Power Management</i> .....	371
8.8.4. <i>Vendor-defined Power Management</i> .....	373
8.9. THERMAL MANAGEMENT .....	373
8.10. UUIDS .....	373

8.11.	MCTP .....	375
8.12.	CONTROL SPACE STRUCTURES .....	375
8.13.	CONTROL STRUCTURE ORGANIZATION .....	378
8.14.	CORE STRUCTURE .....	381
8.14.1.	<i>Core C-Status</i> .....	406
8.14.2.	<i>Core C-Control</i> .....	408
8.14.3.	<i>Core Structure Component CAP 1</i> .....	410
8.14.4.	<i>Core Structure Component CAP 1 Control</i> .....	415
8.14.5.	<i>Core Structure Component CAP 2</i> .....	424
8.14.6.	<i>Core Structure Component CAP 2 Control</i> .....	426
8.14.7.	<i>Core Structure Component CAP 3</i> .....	428
8.14.8.	<i>Core Structure Component CAP 3 Control</i> .....	429
8.14.9.	<i>Core Structure Component CAP 4</i> .....	430
8.14.10.	<i>Core Structure Component CAP 4 Control</i> .....	430
8.14.11.	<i>Core LPD BDF Table</i> .....	430
8.15.	OPCODE SET STRUCTURE .....	431
8.16.	INTERFACE STRUCTURE .....	441
8.16.1.	<i>Interface I-Status</i> .....	459
8.16.2.	<i>Interface I-Control</i> .....	461
8.16.3.	<i>Interface I-CAP 1</i> .....	464
8.16.4.	<i>Interface I-CAP 1 Control</i> .....	467
8.16.5.	<i>Interface I-CAP 2</i> .....	473
8.16.6.	<i>Interface I-CAP 2 Control</i> .....	473
8.16.7.	<i>Interface Error Fields</i> .....	473
8.17.	INTERFACE PHY STRUCTURE .....	476
8.18.	INTERFACE STATISTICS STRUCTURE .....	492
8.19.	COMPONENT MEDIA STRUCTURE .....	499
8.19.1.	<i>Primary Media Status</i> .....	516
8.19.2.	<i>Secondary Media Status</i> .....	518
8.19.3.	<i>Primary Media CAP 1</i> .....	520
8.19.4.	<i>Secondary Media CAP 1</i> .....	525
8.19.5.	<i>Primary Media CAP 1 Control</i> .....	529
8.19.6.	<i>Secondary Media CAP 1 Control</i> .....	532
8.20.	P2P-CORE ENCODED STRUCTURES .....	545
8.20.1.	<i>Responder Bank Structure</i> .....	545
8.20.2.	<i>Requester Bank Structure</i> .....	549
8.21.	COMPONENT ERROR AND SIGNAL EVENT STRUCTURE .....	555
8.21.1.	<i>C-Error Fields</i> .....	565
8.21.2.	<i>C-Event and I-Event Fields</i> .....	570
8.21.3.	<i>Component Error Logs</i> .....	575
8.22.	COMPONENT SWITCH STRUCTURE .....	577
8.22.1.	<i>Route Control Field</i> .....	590
8.23.	COMPONENT MULTICAST STRUCTURE .....	592
8.24.	COMPONENT EXTENSION STRUCTURE .....	601

8.25. COMPONENT STATISTICS STRUCTURE .....	602
8.26. COMPONENT IMAGE STRUCTURE .....	606
8.27. COMPONENT PRECISION TIME STRUCTURE.....	612
8.28. COMPONENT MECHANICAL STRUCTURE .....	616
8.29. COMPONENT DESTINATION TABLE STRUCTURE .....	629
8.29.1. <i>Single-Subnet Single-Route Egress Selection</i> .....	639
8.29.2. <i>Single-Subnet Multi-Route Egress Selection</i> .....	640
8.29.3. <i>Responder Egress and VC Selection</i> .....	641
8.29.4. <i>Multi-subnet Requester Egress Selection</i> .....	642
8.30. VENDOR-DEFINED STRUCTURE .....	643
8.31. COMPONENT SECURITY STRUCTURE .....	645
8.32. COMPONENT TR STRUCTURE .....	654
8.33. SERVICE UUID STRUCTURE .....	659
8.34. COMPONENT C-ACCESS STRUCTURE .....	662
8.35. COMPONENT PA (PEER ATTRIBUTE) STRUCTURE.....	665
8.36. COMPONENT EVENT STRUCTURE.....	679
8.37. COMPONENT SOD STRUCTURE.....	684
8.38. COMPONENT ATP STRUCTURE .....	687
8.39. CONGESTION MANAGEMENT STRUCTURE .....	694
8.40. COMPONENT RKD STRUCTURE .....	698
8.41. COMPONENT RE TABLE STRUCTURE .....	699
8.42. COMPONENT PM STRUCTURE .....	701
<b>9. SWITCHES .....</b>	<b>706</b>
9.1. UNICAST PACKET RELAY .....	708
9.1.1. <i>Single Subnet – Single Route</i> .....	708
9.1.2. <i>Single-Route VC Remapping</i> .....	709
9.1.3. <i>Single-Subnet Multipath Routing</i> .....	710
9.1.4. <i>Multi-Subnet Multipath Routing</i> .....	711
9.2. ROUTING TABLES AND CORRUPT PACKET RELAY .....	714
9.3. DIRECTED CONTROL SPACE PACKET RELAY .....	714
9.3.1. <i>Configuration Using Directed Packet Relay</i> .....	714
9.3.2. <i>Indirect Manager Communication</i> .....	715
<b>10. TRANSPARENT ROUTER (TR) .....</b>	<b>717</b>
10.1. TR-TO-TR COMMUNICATIONS.....	724
<b>11. LINK SPECIFICATION.....</b>	<b>726</b>
11.1. LINK STATES.....	726
11.2. LINK PROTOCOL .....	729
11.3. LINK-LOCAL PACKET PROTOCOL FORMATS.....	729
11.4. OP_CODES .....	730
11.5. LINK-LOCAL FLOW CONTROL .....	731
11.5.1. <i>Implicit Flow Control</i> .....	731

11.5.2. <i>Explicit Flow Control</i> .....	732
11.6. LINK RFC .....	740
11.7. LINK SYNCHRONIZATION (LINK SR) .....	740
11.8. LINK IDLE .....	741
11.9. STEPS FOLLOWING PHYSICAL LAYER RETRAINING.....	741
11.10. LINK RESYNCHRONIZATION .....	741
11.10.1. <i>Link Resynchronization Request (Link RR)</i> .....	743
11.10.2. <i>Link Resynchronization Pattern (Link RP)</i> .....	743
11.10.3. <i>Link Resynchronization Completion (Link RC)</i> .....	743
11.11. LINK CTL.....	744
11.12. LINK-LEVEL RELIABILITY (LLR).....	754
11.13. LINK-LOCAL PACKET VALIDATION AND PROCESSING .....	760
11.13.1. <i>Non-AFC Link-local Flow-Control Processing</i> .....	763
11.13.2. <i>Link-Local Link CTL Processing</i> .....	765
11.13.3. <i>Link-Local Synchronization Processing</i> .....	766
11.13.4. <i>Link-Local Link RP Transmission Processing</i> .....	768
11.13.5. <i>LLR Processing</i> .....	770
11.13.6. <i>Link-Local Error Retrain Processing</i> .....	772
<b>12. END-TO-END PACKET RELIABILITY / VALIDATION.....</b>	<b>774</b>
12.1. UNICAST RELIABLE DELIVERY .....	774
12.2. P2P-CORE / P2P-COHERENCY RELIABLE DELIVERY.....	776
12.3. EXPLICIT OPCLASS RELIABLE DELIVERY .....	776
12.4. NON-DETERMINISTIC REQUEST EXECUTION.....	777
12.5. NON-IDEMPOTENT REQUEST (NIR) .....	778
12.6. PACKET DATA INTEGRITY .....	783
12.6.1. PRELUDE CRC (PCRC).....	784
12.6.2. END-TO-END CRC (ECRC) .....	784
12.6.3. FLIT CRC.....	785
12.7. ERROR DETECTION AND RECOVERY .....	785
12.8. PHYSICAL LAYER ERROR HANDLING.....	785
12.9. PACKET VALIDATION AND PROCESSING .....	786
12.9.1. <i>P2P-Core Packet Processing</i> .....	788
12.9.2. <i>P2P-Coherency Packet Processing</i> .....	791
12.9.3. <i>Destination Component Packet Processing</i> .....	794
12.9.4. <i>Component Error Processing</i> .....	799
12.9.5. <i>Switch Packet Processing</i> .....	801
12.9.6. <i>Transparent Router Unicast Packet Processing</i> .....	808
12.9.7. MP AND SECURITY ERROR DETECTION .....	815
12.9.8. ACCESS KEY VALIDATION .....	818
12.9.9. ACCESS PERMISSION VALIDATION.....	821
12.9.10. TRANSIENT ERROR CLEAN-UP.....	824
12.10. ERROR CONTAINMENT .....	825
12.10.1. COMPONENT CONTAINMENT.....	826

12.10.2. INTERFACE CONTAINMENT .....	826
12.11. PACKET SCHEDULING PRECEDENCE .....	827
12.12. RESET FUNCTIONALITY .....	827
12.12.1. <i>Component Reset</i> .....	828
12.12.2. <i>Interface Reset</i> .....	830
<b>13. PHYSICAL LAYER ABSTRACTION.....</b>	<b>832</b>
13.1. REQUIRED PHYSICAL LAYER FEATURES .....	832
13.1.1. <i>Link Training</i> .....	832
13.1.2. <i>PHY States</i> .....	832
13.1.3. <i>State Transitions</i> .....	833
13.1.4. <i>Configuration Registers</i> .....	833
13.1.5. <i>Bandwidth Matching</i> .....	834
13.1.6. <i>Data Striping</i> .....	834
13.1.7. <i>Data Width</i> .....	835
13.1.8. <i>Packet Alignment</i> .....	836
13.1.9. <i>Error Management</i> .....	836
13.1.10. <i>PCIe Physical Layer</i> .....	836
13.2. OPTIONAL PHYSICAL LAYER FEATURES .....	837
13.2.1. <i>Power States</i> .....	837
13.2.2. <i>Link Width Reduction</i> .....	837
13.2.3. <i>Link Speed Reduction</i> .....	838
13.2.4. <i>Retrain State</i> .....	838
13.2.5. <i>IDLE Symbols</i> .....	838
13.2.6. <i>Interface Aggregation PLA Impacts</i> .....	838
13.2.7. <i>Flit Encoding</i> .....	838
13.2.8. <i>Flit Encoding Implementation Overview</i> .....	839
13.2.9. <i>Flit Encode</i> .....	840
13.3. PLA INTERFACE .....	842
13.4. PLA INTERFACE SIGNALS .....	843
13.5. PHYSICAL LAYER CONFIGURATION .....	849
<b>14. MULTICAST .....</b>	<b>850</b>
14.1. UNRELIABLE MULTICAST.....	852
14.1.1. <i>Unreliable Multicast Write</i> .....	852
14.1.2. <i>Unreliable Multicast Write MSG</i> .....	853
14.1.3. <i>Unreliable Multicast Encapsulation Packet</i> .....	854
14.2. RELIABLE MULTICAST.....	855
14.2.1. <i>Reliable Multicast Write and Write Persistent</i> .....	856
14.2.2. <i>Reliable Multicast Write MSG</i> .....	857
14.2.3. <i>Reliable Multicast Encapsulation Packet</i> .....	858
14.2.4. <i>Reliable Multicast Acknowledgment</i> .....	859
14.3. MULTICAST GROUP MANAGEMENT .....	863
14.3.1. <i>Multicast Group Create</i> .....	863

14.3.2. <i>Multicast Join</i> .....	863
14.3.3. <i>Multicast Leave</i> .....	863
14.3.4. <i>Multicast Prune</i> .....	863
14.3.5. <i>Multicast Group Delete</i> .....	864
<b>15. CONGESTION AND PACKET DEADLINE</b> .....	<b>865</b>
15.1. CONGESTION PROTOCOL FIELD .....	865
15.2. DEADLINE SEMANTICS .....	866
15.3. CONGESTION AND PACKET INJECTION RATE .....	869
<b>16. SECURITY</b> .....	<b>871</b>
16.1. MALICIOUS COMPONENT THREATS .....	871
16.2. MALICIOUS COMPONENT THREAT MITIGATION .....	872
16.3. ADDITIONAL PACKET VALIDATION REQUIREMENTS .....	873
16.4. CRYPTOGRAPHICALLY SECURE AUTHENTICATION .....	873
16.4.1. <i>HMAC Code Calculation</i> .....	874
16.5. ANTI-REPLAY TAG (ART) .....	874
16.5.1. <i>Sequence Number ART</i> .....	875
16.5.2. <i>Precision Time ART</i> .....	876
16.6. MITIGATING IN SITU INSERTION .....	877
16.7. TRANSACTION INTEGRITY KEYS (TIK) MANAGEMENT .....	878
<b>17. STRONG ORDERING DOMAIN (SOD)</b> .....	<b>880</b>
17.1. SOD OPERATIONS AND PACKET FORMATS .....	887
17.1.1. <i>SOD Read Request</i> .....	888
17.1.2. <i>SOD Read Response</i> .....	889
17.1.3. <i>SOD Write</i> .....	890
17.1.4. <i>SOD Write MSG</i> .....	891
17.1.5. <i>SOD Standalone Acknowledgment</i> .....	891
17.1.6. <i>SOD Encapsulation</i> .....	892
17.1.7. <i>SOD Interrupt</i> .....	893
17.1.8. <i>SOD Sync</i> .....	894
17.2. SOD OPCLASS PACKET PROCESSING .....	895
<b>18. LOGICAL PCI DEVICE (LPD)</b> .....	<b>898</b>
18.1. LPD OPERATIONAL OVERVIEW .....	898
18.2. PCIe-COMPATIBLE ORDERING (PCO) .....	899
18.2.1. <i>PCO Operation and Requirements</i> .....	900
18.2.2. <i>Deadlock Avoidance for PCO</i> .....	902
18.2.3. <i>Rules for PCO Requesters</i> .....	904
18.2.5. <i>Topology Considerations for PCO</i> .....	905
18.3. PECAM OPERATION AND REQUIREMENTS .....	908
18.4. LPD CONFIGURATION SPACE .....	914
18.4.1. <i>PCI Type 0 Configuration Space Header</i> .....	914
18.4.2. <i>PCI Type 1 Configuration Space Header</i> .....	915

18.4.3. <i>PCI Capability Structures</i> .....	915
18.4.4. <i>PCI Express Extended Capability Structures</i> .....	917
18.5. LPD MMIO SPACE.....	919
18.6. LPD ERROR HANDLING .....	922
18.7. COMPONENT LPD STRUCTURE .....	922
18.8. LPD COMPONENT REQUIREMENTS.....	929
<b>19. LOGICAL PCI HIERARCHY (LPH).....</b>	<b>930</b>
19.1. LPH OPERATIONAL OVERVIEW .....	931
19.2. LPH SUPPORT FOR PCI CONFIGURATION SPACE.....	932
19.3. LPH SUPPORT FOR PCI MMIO .....	933
19.4. LPH PIO COMMON LOGIC .....	935
19.5. LPH SUPPORT FOR DMA .....	936
19.6. LPH SUPPORT FOR PEER-TO-PEER TRAFFIC .....	937
19.7. LPH ERROR HANDLING .....	937
19.8. COMPONENT LPH STRUCTURE .....	937
19.9. LPH COMPONENT REQUIREMENTS.....	943
<b>20. ADDRESS TRANSLATION AND PAGE SERVICES.....</b>	<b>944</b>
20.1. ADDRESS TRANSLATION SERVICES OVERVIEW .....	944
20.2. PAGE SERVICES OVERVIEW.....	945
20.3. PASID .....	946
20.3.1. <i>PASID Stop Sequence</i> .....	946
20.4. COMMON ATP PACKET PROTOCOL FIELDS.....	947
20.5. ADDRESS TRANSLATION SERVICES.....	948
20.5.1. <i>Translation Request</i> .....	948
20.5.2. <i>Translation Response</i> .....	951
20.5.3. <i>Translation Invalidate Request</i> .....	956
20.5.4. <i>Translation Invalidate Response</i> .....	959
20.6. PAGE SERVICES .....	961
20.6.1. <i>PRG Request</i> .....	961
20.6.2. <i>PRG Response Notification</i> .....	965
20.6.3. <i>PRG Release</i> .....	967
20.6.4. <i>PRG Eviction Notification</i> .....	969
20.6.5. <i>Stop Marker</i> .....	969
<b>APPENDIX A CONTROL SPACE STRUCTURE ENCODINGS.....</b>	<b>972</b>
<b>APPENDIX B CRC.....</b>	<b>974</b>
<b>6-BIT CRC .....</b>	<b>974</b>
<b>24-BIT CRC .....</b>	<b>976</b>
<b>16-BIT CRC .....</b>	<b>978</b>
<b>APPENDIX C COMPONENT CLASS ENCODINGS .....</b>	<b>981</b>
<b>ACKNOWLEDGMENTS .....</b>	<b>983</b>

# Figures

FIGURE 1-1: EXAMPLE SINGLE INTERFACE REQUEST / RESPONSE EXCHANGE .....	33
FIGURE 1-2: SAMPLE SPACE OF COMPONENT TYPES .....	34
FIGURE 1-3: EXAMPLE POINT-TO-POINT TOPOLOGY WITH A MIX OF COMPONENT TYPES .....	35
FIGURE 1-4: EXAMPLE POINT-TO-POINT AND SWITCH-BASED GEN-Z TOPOLOGIES WITH A MIX OF COMPONENT TYPES .....	35
FIGURE 1-5: MONOLITHIC MEMORY / MEDIA CONTROLLER .....	35
FIGURE 1-6: SPLIT MEMORY AND MEDIA CONTROLLER .....	36
FIGURE 1-7: EXAMPLE REQUESTER-RESPONDER TOPOLOGIES .....	39
FIGURE 1-8: DISCRETE AND COMPONENT-INTEGRATED SWITCHES .....	40
FIGURE 1-9: SERIALLY CONNECTED COMPONENTS USING INTEGRATED SWITCHES .....	40
FIGURE 1-10: EXAMPLE INTEGRATED SWITCHES FOR FAN-OUT, AGGREGATE BANDWIDTH, AND RESILIENCY .....	41
FIGURE 1-11: EXAMPLE 16-INTERFACE SWITCH FOR HIGH FAN-OUT AND CAPACITY SOLUTIONS .....	41
FIGURE 1-12: TR JOINING MULTIPLE SUBNETS .....	42
FIGURE 1-13: EXAMPLE TRANSPARENT ROUTER WITH A MIX OF COMPONENT TYPES .....	43
FIGURE 1-14: EXAMPLE LATENCY DOMAINS .....	45
FIGURE 1-15: PROCESSOR-CENTRIC SOLUTION ARCHITECTURE EVOLUTION WITH GEN-Z .....	46
FIGURE 1-16: MEMORY-CENTRIC ARCHITECTURE .....	47
FIGURE 1-17: EXAMPLE MEMORY-SEMANTIC MESSAGING TOPOLOGY .....	47
FIGURE 3-1: ARCHITECTURAL LAYERS .....	69
FIGURE 3-2: EXAMPLE PIPELINED REQUEST AND RESPONSE PACKETS .....	70
FIGURE 3-3: LINK-LOCAL VS. END-TO-END PACKETS .....	70
FIGURE 3-4: COMPONENT WITH DATA AND CONTROL SPACES .....	72
FIGURE 3-5: SINGLE AND DUAL-VC COMMUNICATION .....	73
FIGURE 3-6: EXAMPLE TIME-MULTIPLEXING MULTIPLE PACKETS WITH DIFFERENT VCS .....	74
FIGURE 3-7: UNICAST PACKETS BETWEEN PROCESSORS AND MEMORY COMPONENTS .....	76
FIGURE 3-8: MULTICAST EXAMPLE .....	76
FIGURE 3-9: EXAMPLE TRANSPARENT ACCELERATOR OPERATION .....	79
FIGURE 3-10: EXAMPLE I/O-BASED ACCELERATOR .....	80
FIGURE 3-11: ACCELERATOR EXECUTABLE INITIALIZATION .....	81
FIGURE 3-12: MULTIPLE INDEPENDENT AND SHARED MEMORY ACCESS REGIONS .....	82
FIGURE 3-13: SINGLE AND MULTI-COMPONENT PAGES WITH R-KEYS .....	83
FIGURE 3-14: MULTIPLE VIRTUAL NIC SHARING A PHYSICAL NIC .....	84
FIGURE 3-15: MULTIPATH TOPOLOGY EXAMPLE .....	86
FIGURE 3-16: EXAMPLE DAISY-CHAIN TOPOLOGIES .....	88
FIGURE 3-17: DAISY CHAIN REQUEST PROCESSING FLOW .....	90
FIGURE 3-18: DAISY CHAIN RESPONSE PROCESSING FLOW .....	91
FIGURE 3-19: EXAMPLE READ MEDIA PROCESSING FLOW .....	95
FIGURE 3-20: EXAMPLE MEDIA DEVICE SPARING FLOW .....	99
FIGURE 3-21: EXAMPLE MEDIA INITIALIZATION FLOW .....	101
FIGURE 3-22: EXAMPLE FORMAT OF A HARDWARE-BASED CAPABILITY .....	103
FIGURE 3-23: EXAMPLE REQUESTER AND RESPONDER WITH CAPABILITY ACCESS TAG .....	103
FIGURE 3-24: EXAMPLE COMPONENT WITH REQUESTER ZMMU .....	105
FIGURE 3-25: EXAMPLE COMPONENT WITH RESPONDER ZMMU .....	107
FIGURE 3-26: EXAMPLE COMPONENT WITH A REQUESTER AND A RESPONDER ZMMU .....	108
FIGURE 3-27: EXAMPLE COMPONENT WITH MULTIPLE REQUESTER AND RESPONDER INSTANCES .....	109

FIGURE 3-28: ZMMU PAGE TABLE STRUCTURE .....	112
FIGURE 3-29: POINTER-PAIR TABLE ENTRY FORMAT .....	112
FIGURE 3-30: MATCHING A HIGH-BANDWIDTH REQUESTER WITH MULTIPLE LOW-BANDWIDTH RESPONDERS .....	114
FIGURE 3-31: EXAMPLE 3-WAY STACK BARBER POLE INTERLEAVE.....	117
FIGURE 3-32: EXAMPLE USE OF COMPONENT-SPECIFIC AND FABRIC INTERLEAVES IN SWITCH TOPOLOGY .....	118
FIGURE 3-33: EXAMPLE USE OF COMPONENT-SPECIFIC INTERLEAVE IN A POINT-TO-POINT TOPOLOGY .....	119
FIGURE 3-34: REQUESTER PTE FORMAT FOR DATA SPACE .....	119
FIGURE 3-35: REQUESTER PTE FORMAT FOR CONTROL SPACE .....	119
FIGURE 3-36: RESPONDER PTE FORMAT .....	125
FIGURE 5-1: P2P-CORE REQUEST GENERIC FORMAT .....	146
FIGURE 5-2: P2P-CORE RESPONSE GENERIC FORMAT.....	146
FIGURE 5-3: P2P-COHERENCY REQUEST GENERIC FORMAT .....	147
FIGURE 5-4: P2P-COHERENCY RESPONSE GENERIC FORMAT .....	147
FIGURE 5-5: EXPLICIT OPCLASS REQUEST GENERIC FORMAT .....	147
FIGURE 5-6: EXPLICIT OPCLASS RESPONSE GENERIC FORMAT .....	147
FIGURE 5-7: P2P-COHERENCY LPD FIELD LOCATION.....	156
FIGURE 5-8: EXPLICIT OPCLASS LPD FIELD LOCATION .....	156
FIGURE 5-9: LPD FIELD FORMATS .....	157
FIGURE 5-10: CONDITIONAL FIELD LOCATIONS WITHIN AN EXPLICIT OPCLASS PACKET .....	161
FIGURE 6-1: READ REQUEST WITH CORRESPONDING READ RESPONSE .....	162
FIGURE 6-2: EXAMPLE RESPONDER RESOURCE LAYOUT FOR P2P-CORE READ OFFSET REQUESTS .....	163
FIGURE 6-3: EXAMPLE P2P-CORE READ AND P2P-CORE READ OFFSET REQUEST SEQUENCE .....	163
FIGURE 6-4: LDM 1 READ REQUEST WITH CORRESPONDING LDM 1 READ RESPONSES .....	164
FIGURE 6-5: P2P-CORE READ REQUEST PACKET FORMAT.....	166
FIGURE 6-6: P2P-CORE READ OFFSET REQUEST PACKET FORMAT .....	166
FIGURE 6-7: P2P-COHERENCY READ REQUEST PACKET FORMAT .....	167
FIGURE 6-8: CORE 64 READ REQUEST PACKET FORMAT.....	167
FIGURE 6-9: LDM 1 READ REQUEST PACKET FORMAT .....	168
FIGURE 6-10: P2P-CORE READ RESPONSE PACKET FORMAT .....	170
FIGURE 6-11: P2P-COHERENCY READ RESPONSE PACKET FORMAT.....	170
FIGURE 6-12: CORE 64 READ RESPONSE PACKET FORMAT .....	170
FIGURE 6-13: LDM 1 READ RESPONSE PACKET FORMAT .....	171
FIGURE 6-14: WRITE REQUEST WITH CORRESPONDING ACKNOWLEDGMENT .....	171
FIGURE 6-15: EXAMPLE RESPONDER RESOURCE LAYOUT FOR P2P-CORE WRITE OFFSET REQUESTS...	172
FIGURE 6-16: EXAMPLE P2P-CORE WRITE AND P2P-CORE WRITE OFFSET REQUEST SEQUENCE .....	173
FIGURE 6-17: P2P-CORE WRITE REQUEST PACKET FORMAT .....	176
FIGURE 6-18: P2P-CORE WRITE OFFSET REQUEST PACKET FORMAT .....	177
FIGURE 6-19: P2P-COHERENCY WRITE REQUEST PACKET FORMAT.....	177
FIGURE 6-20: CORE 64 WRITE REQUEST PACKET FORMAT .....	178
FIGURE 6-21: EXAMPLE WRITE MSG REQUEST .....	179
FIGURE 6-22: EXAMPLE WRITE MSG WITH EMBEDDED READ PACKET EXCHANGE SEQUENCE .....	184
FIGURE 6-23: (UNRELIABLE) WRITE MSG REQUEST PACKET FORMAT.....	185
FIGURE 6-24: P2P-CORE WRITE POISON REQUEST PACKET FORMAT .....	189
FIGURE 6-25: P2P-COHERENCY WRITE POISON REQUEST PACKET FORMAT .....	189
FIGURE 6-26: CORE 64 WRITE POISON REQUEST PACKET FORMAT .....	190
FIGURE 6-27: CORE 64 WUM PACKET FORMAT.....	191
FIGURE 6-28: P2P-CORE WRITE PARTIAL PACKET FORMAT .....	193
FIGURE 6-29: P2P-COHERENCY WRITE PARTIAL PACKET FORMAT.....	193

FIGURE 6-30: CORE 64 WRITE PARTIAL PACKET FORMAT.....	194
FIGURE 6-31: P2P-CORE META WRITE PACKET FORMAT .....	195
FIGURE 6-32: P2P-COHERENCY META WRITE PACKET FORMAT.....	196
FIGURE 6-33: P2P-CORE STANDALONE ACKNOWLEDGMENT PACKET .....	197
FIGURE 6-34: P2P-COHERENCY STANDALONE ACKNOWLEDGMENT PACKET .....	197
FIGURE 6-35: CORE 64 STANDALONE ACKNOWLEDGMENT PACKET .....	198
FIGURE 6-36: P2P-CORE PERSISTENT FLUSH PACKET FORMAT .....	206
FIGURE 6-37: CORE 64 PERSISTENT FLUSH PACKET FORMAT .....	206
FIGURE 6-38: P2P-CORE CTL-READ REQUEST PACKET FORMAT.....	207
FIGURE 6-39: P2P-CORE CTL-WRITE REQUEST PACKET FORMAT .....	208
FIGURE 6-40: P2P-COHERENCY CTL-READ REQUEST PACKET FORMAT.....	208
FIGURE 6-41: P2P-COHERENCY CTL-WRITE REQUEST PACKET FORMAT .....	209
FIGURE 6-42: P2P-CORE CTL-UE REQUEST PACKET FORMAT.....	210
FIGURE 6-43: P2P-COHERENCY CTL-UE REQUEST PACKET FORMAT.....	211
FIGURE 6-44: CONTROL READ REQUEST PACKET FORMAT.....	213
FIGURE 6-45: CONTROL WRITE REQUEST PACKET FORMAT .....	214
FIGURE 6-46: EXAMPLE UNSOLICITED EVENT PACKET AND UERT OPERATION .....	216
FIGURE 6-47: CONTROL UNSOLICITED EVENT PACKET FORMAT .....	217
FIGURE 6-48: NO-OP PACKET FORMAT .....	226
FIGURE 6-49: EXAMPLE C-STATE NOTIFICATION ZERO DURATION EXCHANGE.....	228
FIGURE 6-50: EXAMPLE C-STATE NOTIFICATION WITH NON-ZERO DURATION FORMAT.....	228
FIGURE 6-51: C-STATE POWER CONTROL PACKET FORMAT .....	229
FIGURE 6-52: R-KEY UPDATE PACKET FORMAT .....	231
FIGURE 6-53: CONTROL WRITE MSG PACKET FORMAT.....	232
FIGURE 7-1: EXAMPLE INTERRUPT SET-UP AND REQUEST PACKET EXCHANGE .....	234
FIGURE 7-2: EXAMPLE INTERRUPT PROCESSING USING REQUESTER AND RESPONDER ZMMUS.....	235
FIGURE 7-3: P2P-COHERENCY INTERRUPT PACKET FORMAT .....	237
FIGURE 7-4: CORE 64 INTERRUPT PACKET FORMAT.....	238
FIGURE 7-5: EXAMPLE OF RESPONDER TARGET MEMORY ACCESS FOR ADD.....	241
FIGURE 7-6: P2P-CORE ATOMIC SINGLE ADDRESS SINGLE OPERAND PACKET FORMAT .....	246
FIGURE 7-7: P2P-COHERENCY ATOMIC SINGLE ADDRESS SINGLE OPERAND PACKET FORMAT .....	247
FIGURE 7-8: EXPLICIT ATOMIC 1 SINGLE ADDRESS SINGLE OPERAND PACKET FORMAT .....	247
FIGURE 7-9: P2P-CORE ATOMIC SINGLE ADDRESS DUAL-OPERAND REQUEST PACKET FORMAT .....	248
FIGURE 7-10: P2P-COHERENCY ATOMIC SINGLE ADDRESS DUAL-OPERAND REQUEST PACKET FORMAT .....	248
FIGURE 7-11: EXPLICIT ATOMIC 1 SINGLE ADDRESS DUAL-OPERAND REQUEST PACKET FORMAT .....	249
FIGURE 7-12: P2P-CORE ATOMIC SINGLE ADDRESS NO OPERAND PACKET FORMAT .....	249
FIGURE 7-13: P2P-COHERENCY ATOMIC SINGLE ADDRESS NO OPERAND PACKET FORMAT.....	250
FIGURE 7-14: EXPLICIT ATOMIC 1 SINGLE ADDRESS NO OPERAND PACKET FORMAT .....	250
FIGURE 7-15: P2P-CORE ATOMIC DUAL-ADDRESS NO OPERAND PACKET FORMAT.....	250
FIGURE 7-16: P2P-COHERENCY ATOMIC DUAL-ADDRESS NO OPERAND PACKET FORMAT .....	251
FIGURE 7-17: EXPLICIT ATOMIC 1 DUAL-ADDRESS NO OPERAND PACKET FORMAT .....	251
FIGURE 7-18: P2P-CORE ATOMIC VECTOR PACKET FORMAT .....	252
FIGURE 7-19: P2P-COHERENCY ATOMIC VECTOR PACKET FORMAT .....	252
FIGURE 7-20: EXPLICIT ATOMIC 1 VECTOR PACKET FORMAT .....	253
FIGURE 7-21: P2P-CORE ATOMIC RESPONSE PACKET FORMAT .....	253
FIGURE 7-22: P2P-COHERENCY ATOMIC RESPONSE PACKET FORMAT .....	253
FIGURE 7-23: EXPLICIT ATOMIC 1 RESPONSE PACKET FORMAT .....	254
FIGURE 7-24: P2P-CORE ATOMIC RESULTS RESPONSE PACKET FORMAT.....	254
FIGURE 7-25: P2P-COHERENCY ATOMIC RESULTS RESPONSE PACKET FORMAT .....	254
FIGURE 7-26: EXPLICIT ATOMIC 1 RESULTS RESPONSE PACKET FORMAT .....	255

FIGURE 7-27: BUFFER REQUEST BETWEEN TWO COMPONENTS .....	256
FIGURE 7-28: BUFFER REQUEST BETWEEN REQUESTER-LOCAL AND REMOTE BUFFERS .....	256
FIGURE 7-29: BUFFER REQUEST WITHIN A SINGLE COMPONENT .....	256
FIGURE 7-30: BUFFER VECTOR REQUEST BETWEEN TWO COMPONENTS.....	257
FIGURE 7-31: BUFFER VECTOR OPERATION SCATTER / GATHER BETWEEN TWO COMPONENTS.....	258
FIGURE 7-32: EXAMPLE BUFFER PUT REQUEST AS A SERIES OF WRITES.....	259
FIGURE 7-33: EXAMPLE BUFFER GET REQUEST AS A SERIES OF READS .....	260
FIGURE 7-34: EXAMPLE SIGNALLED BUFFER PUT (A TO B) WITH A SIGNALLED WRITE .....	261
FIGURE 7-35: EXAMPLE SIGNALLED BUFFER GET (B TO A).....	261
FIGURE 7-36: EXAMPLE DYNAMIC BUFFER ALLOCATION EXCHANGE .....	262
FIGURE 7-37: EXAMPLE DYNAMIC PUT EXCHANGE .....	262
FIGURE 7-38: EXAMPLE DYNAMIC BUFFER GET EXCHANGE .....	263
FIGURE 7-39: P2P-CORE BUFFER PUT LOCAL REQUEST PACKET FORMAT.....	269
FIGURE 7-40: BUFFER REQUEST WITH R-KEY AND / OR GLOBAL DCID B FIELD .....	270
FIGURE 7-41: (DYNAMIC) BUFFER PUT / GET REQUEST PACKET FORMAT .....	270
FIGURE 7-42: SIGNALLED (DYNAMIC) BUFFER PUT / GET REQUEST PACKET FORMAT.....	272
FIGURE 7-43: DYNAMIC BUFFER ALLOCATE REQUEST PACKET FORMAT .....	273
FIGURE 7-44: DYNAMIC BUFFER RELEASE REQUEST PACKET FORMAT .....	274
FIGURE 7-45: BUFFER PUTV / GETV REQUEST PACKET FORMATS .....	275
FIGURE 7-46: BUFFER ALLOCATE RESPONSE PACKET FORMAT.....	276
FIGURE 7-47: PATTERN APPLIED TO BUFFER A .....	277
FIGURE 7-48: PATTERN REQUEST PACKET FORMAT.....	279
FIGURE 7-49: PATTERN RESPONSE PACKET FORMAT .....	280
FIGURE 7-50: CORE 64 READ DUAL-OP REQUEST PACKET FORMAT .....	286
FIGURE 7-51: CORE 64 READ TRIPLE-OP REQUEST PACKET FORMAT .....	287
FIGURE 7-52: CORE 64 READ QUAD-OP REQUEST PACKET FORMAT .....	288
FIGURE 7-53: CORE 64 WRITE DUAL-OP REQUEST PACKET FORMAT.....	289
FIGURE 7-54: CORE 64 WRITE TRIPLE-OP REQUEST PACKET FORMAT.....	290
FIGURE 7-55: CORE 64 WRITE QUAD-OP REQUEST PACKET FORMAT .....	291
FIGURE 7-56: CORE 64 SIGNALLED WRITE / WRITE + INTERRUPT REQUEST PACKET FORMAT.....	292
FIGURE 7-57: CORE 64 WRITE-READ REQUEST PACKET FORMAT .....	292
FIGURE 7-58: CORE 64 WRITE-WAKE REQUEST PACKET FORMAT .....	293
FIGURE 7-59: P2P-CORE VENDOR-DEFINED REQUEST PACKET FORMAT .....	295
FIGURE 7-60: P2P-CORE VENDOR-DEFINED RESPONSE PACKET FORMAT .....	295
FIGURE 7-61: P2P-COHERENCY VENDOR-DEFINED REQUEST PACKET FORMAT.....	296
FIGURE 7-62: P2P-COHERENCY VENDOR-DEFINED RESPONSE PACKET FORMAT .....	296
FIGURE 7-63: EXPLICIT VENDOR-DEFINED OPCLASS PACKET FORMAT .....	297
FIGURE 7-64: CORE 64 NIRR PACKET FORMAT .....	298
FIGURE 7-65: CTXID NIRR PACKET FORMAT .....	299
FIGURE 7-66: EXAMPLE PRECISION TIME DISTRIBUTION TOPOLOGIES .....	300
FIGURE 7-67: EXAMPLE PRECISION TIME DISTRIBUTION .....	301
FIGURE 7-68: EXAMPLE PRECISION TIME REQUEST-RESPONSE EXCHANGE.....	301
FIGURE 7-69: TIMESTAMP MEASUREMENT BETWEEN T2 AND T3 .....	302
FIGURE 7-70: PRECISION TIME REQUEST FORMAT .....	304
FIGURE 7-71: PRECISION TIME RESPONSE FORMAT .....	305
FIGURE 7-72: LNREQ-LNRSP EXCHANGE USING LN READ REQUEST .....	307
FIGURE 7-73: LNREQ-LNRSP EXCHANGE USING LN WRITE REQUEST .....	308
FIGURE 7-74: LN READ REQUEST PACKET FORMAT .....	311
FIGURE 7-75: LN READ RESPONSE PACKET FORMAT.....	312
FIGURE 7-76: LN WRITE PACKET FORMAT.....	312
FIGURE 7-77: LN NOTIFICATION PACKET FORMAT .....	313

FIGURE 7-78: CORE 64 WAKE THREAD REQUEST PACKET FORMAT .....	314
FIGURE 7-79: P2P-CORE CAPABILITIES READ REQUEST PACKET FORMAT .....	315
FIGURE 7-80: UNICAST ENCAPSULATION PACKET FORMAT .....	316
FIGURE 7-81: P2P-COHERENCY READ EXCLUSIVE / EXCLUSIVE / RELEASE / INVALIDATE REQUEST PACKET FORMAT .....	317
FIGURE 7-82: CORE 64 READ EXCLUSIVE / EXCLUSIVE / RELEASE REQUEST PACKET FORMAT .....	318
FIGURE 7-83: P2P-COHERENCY READ SHARED REQUEST PACKET FORMAT .....	319
FIGURE 7-84: CORE 64 READ SHARED REQUEST PACKET FORMAT .....	320
FIGURE 7-85: P2P-COHERENCY CACHE LINE ATTRIBUTE REQUEST PACKET FORMAT .....	322
FIGURE 7-86: CORE 64 CACHE LINE ATTRIBUTE REQUEST PACKET FORMAT .....	323
FIGURE 7-87: P2P-COHERENCY CACHE LINE ATTRIBUTE RESPONSE PACKET FORMAT .....	323
FIGURE 7-88: CORE 64 CACHE LINE ATTRIBUTE RESPONSE PACKET FORMAT .....	324
FIGURE 7-89: EXAMPLE SINGLE REQUEST-RESPONSE PACKET EXCHANGE .....	326
FIGURE 7-90: EXAMPLE WRITEBACK REQUEST PACKET EXCHANGE .....	327
FIGURE 7-91: P2P-COHERENCY WRITEBACK REQUEST PACKET FORMAT .....	327
FIGURE 7-92: CORE 64 INVALIDATE REQUEST PACKET FORMAT .....	328
FIGURE 7-93: CORE 64 WRITEBACK REQUEST PACKET FORMAT .....	329
FIGURE 7-94: EXAMPLE SWITCH TOPOLOGY WITH COLLECTIVE ACCELERATORS .....	331
FIGURE 7-95: BARRIER COLLECTIVE LADDER DIAGRAM .....	336
FIGURE 7-96: BARRIER AND ABORT COLLECTIVE REQUEST PACKET FORMAT .....	337
FIGURE 7-97: BROADCAST COLLECTIVE LADDER DIAGRAM .....	337
FIGURE 7-98: BROADCAST COLLECTIVE REQUEST PACKET FORMAT .....	338
FIGURE 7-99: SCATTER COLLECTIVE LADDER DIAGRAM .....	339
FIGURE 7-100: SCATTER AND GATHER COLLECTIVE REQUEST PACKET FORMAT .....	340
FIGURE 7-101: GATHER COLLECTIVE LADDER DIAGRAM .....	341
FIGURE 7-102: ALL-GATHER COLLECTIVE LADDER DIAGRAM .....	343
FIGURE 7-103: REDUCE COLLECTIVE LADDER DIAGRAM .....	344
FIGURE 7-104: REDUCE COLLECTIVE REQUEST PACKET FORMAT .....	345
FIGURE 7-105: ALL-REDUCE COLLECTIVE LADDER DIAGRAM .....	347
FIGURE 7-106: MAP-REDUCE COLLECTIVE LADDER DIAGRAM .....	348
FIGURE 7-107: ALL-MAP-REDUCE COLLECTIVE LADDER DIAGRAM .....	350
FIGURE 7-108: COLLECTIVE RESULT REQUEST PACKET FORMAT .....	352
FIGURE 7-109: COLLECTIVE RESPONDER COUNT RESPONSE PACKET FORMAT .....	354
FIGURE 8-1: INTERFACE STATE MACHINE .....	356
FIGURE 8-2: COMPONENT STATE MACHINE .....	360
FIGURE 8-3: EXAMPLE DISCOVERY TOPOLOGIES .....	364
FIGURE 8-4: EXAMPLE MINIMALIST CONTROL SPACE STRUCTURE LAYOUT .....	376
FIGURE 8-5: EXAMPLE OF A FEATURE-RICH CONTROL SPACE STRUCTURE LAYOUT .....	376
FIGURE 8-6: CORE STRUCTURE FORMAT .....	382
FIGURE 8-7: CORE LPD BDF TABLE FORMAT .....	430
FIGURE 8-7: OPCode SET STRUCTURE FORMAT .....	433
FIGURE 8-9: EXAMPLE AGGREGATED INTERFACES .....	445
FIGURE 8-10: EXAMPLE INTERFACE GROUPS .....	445
FIGURE 8-11: EXAMPLE INTERFACE GROUP AND AGGREGATED INTERFACE POINTERS .....	446
FIGURE 8-12: INTERFACE STRUCTURE FORMAT .....	447
FIGURE 8-13: INTERFACE PHY STRUCTURE FORMAT .....	477
FIGURE 8-14: INTERFACE STATISTICS STRUCTURE FORMAT .....	494
FIGURE 8-15: COMPONENT MEDIA STRUCTURE FORMAT .....	505
FIGURE 8-16: PRIMARY AND SECONDARY MEDIA LOG ENTRY FORMAT .....	535
FIGURE 8-17: RESPONDER BANK STRUCTURE FORMAT .....	547
FIGURE 8-18: REQUESTER BANK STRUCTURE FORMAT .....	551

FIGURE 8-19: COMPONENT ERROR AND SIGNAL EVENT STRUCTURE FORMAT .....	558
FIGURE 8-20: COMPONENT ERROR LOG FORMAT – COMPONENT ERROR .....	575
FIGURE 8-21: COMPONENT ERROR LOG FORMAT – COMPONENT INTERFACE ERROR .....	576
FIGURE 8-22: LPRT / MPRT FORMAT .....	578
FIGURE 8-23: LPRT / MPRT ROUTE ENTRY ROW FORMAT .....	579
FIGURE 8-24: VCAT AND VCAT ENTRY FORMAT .....	581
FIGURE 8-25: MCPRT / MSMCPRT FORMAT .....	582
FIGURE 8-26: MCPRT / MSMCPRT ROW FORMAT .....	582
FIGURE 8-27: MVCAT FORMAT .....	583
FIGURE 8-28: COMPONENT SWITCH STRUCTURE FORMAT .....	584
FIGURE 8-29: ROUTE CONTROL FIELD .....	590
FIGURE 8-30: UNRELIABLE MULTICAST TABLE FORMAT .....	593
FIGURE 8-31: UNRELIABLE MULTICAST ROW FORMAT .....	594
FIGURE 8-32: RELIABLE MULTICAST TABLE FORMAT .....	594
FIGURE 8-33: RELIABLE MULTICAST TABLE ENTRY ROW FORMAT .....	595
FIGURE 8-34: RELIABLE MULTICAST RESPONDER TABLE .....	596
FIGURE 8-35: COMPONENT MULTICAST STRUCTURE FORMAT .....	597
FIGURE 8-36: COMPONENT EXTENSION STRUCTURE FORMAT .....	602
FIGURE 8-37: COMPONENT STATISTICS STRUCTURE FORMAT .....	603
FIGURE 8-38: COMPONENT IMAGE STRUCTURE FORMAT .....	608
FIGURE 8-39: PRECISION TIME STRUCTURE FORMAT .....	613
FIGURE 8-40: EXAMPLE MECHANICAL CONNECTIVITY .....	616
FIGURE 8-41: COMPONENT MECHANICAL STRUCTURE FORMAT .....	619
FIGURE 8-42: SSDT / MSDT FORMAT .....	631
FIGURE 8-43: SSDT / MSDT ROW FORMAT .....	631
FIGURE 8-44: REQ-VCAT AND VCAT ENTRY FORMAT .....	633
FIGURE 8-45: RSP-VCAT FORMAT .....	633
FIGURE 8-46: RIT FORMAT .....	634
FIGURE 8-47: COMPONENT DESTINATION TABLE STRUCTURE FORMAT .....	635
FIGURE 8-48: SINGLE-SUBNET SINGLE-ROUTE REQUESTER EGRESS SELECTION .....	640
FIGURE 8-49: SINGLE-SUBNET MULTI-ROUTE REQUESTER EGRESS SELECTION .....	641
FIGURE 8-50: RESPONDER EGRESS SELECTION .....	642
FIGURE 8-51: MULTI-SUBNET REQUESTER EGRESS SELECTION .....	643
FIGURE 8-52: VENDOR-DEFINED STRUCTURE FORMAT .....	644
FIGURE 8-53: VENDOR-DEFINED WITH UUID STRUCTURE FORMAT .....	645
FIGURE 8-54: COMPONENT SECURITY STRUCTURE FORMAT .....	648
FIGURE 8-55: COMPONENT TR STRUCTURE FORMAT .....	656
FIGURE 8-56: SERVICE UUID STRUCTURE FORMAT .....	661
FIGURE 8-57: COMPONENT C-ACCESS STRUCTURE FORMAT .....	664
FIGURE 8-58: PA AND SEC TABLE FORMATS .....	669
FIGURE 8-59: SSAP, MSAP, MCAP, AND MSMCAP TABLE FORMATS .....	670
FIGURE 8-60: COMPONENT PA STRUCTURE FORMAT .....	673
FIGURE 8-61: COMPONENT EVENT STRUCTURE FORMAT .....	680
FIGURE 8-62: EVENT RECORD ENTRY FORMAT .....	683
FIGURE 8-63: COMPONENT SOD STRUCTURE FORMAT .....	685
FIGURE 8-64: COMPONENT ATP STRUCTURE FORMAT .....	688
FIGURE 8-65: CONGESTION MANAGEMENT STRUCTURE FORMAT .....	695
FIGURE 8-66: COMPONENT RKD STRUCTURE FORMAT .....	698
FIGURE 8-67: COMPONENT RE TABLE STRUCTURE FORMAT .....	700
FIGURE 8-68: PERFORMANCE LOG RECORD TYPE 0 FORMAT .....	703
FIGURE 8-69: PERFORMANCE LOG RECORD TYPE 1 FORMAT .....	703

FIGURE 8-70: COMPONENT PM STRUCTURE FORMAT .....	703
FIGURE 9-1: SINGLE-SUBNET—SINGLE ROUTE .....	709
FIGURE 9-2: SINGLE ROUTE—VC REMAPPING.....	710
FIGURE 9-3: SINGLE-SUBNET—MULTIPATH SELECTION.....	711
FIGURE 9-4: MULTI-SUBNET MULTIPATH SELECTION (MSS = 1B AND HCS = 1B) .....	713
FIGURE 9-5: MULTI-SUBNET MULTIPATH SELECTION (MSS = 1B AND HCS = 0B) .....	713
FIGURE 9-6: EXAMPLE DIRECTED-CONTROL SPACE PACKET RELAY .....	715
FIGURE 9-7: EXAMPLE INDIRECT MANAGER COMMUNICATION.....	716
FIGURE 10-1: PROCESSORS COMMUNICATING THROUGH A TR.....	717
FIGURE 10-2: TWO SUBNETS CONNECTED BY A TWO TRs.....	718
FIGURE 10-3: MULTIPLE SUBNETS CONNECTED BY MULTIPLE TRs .....	719
FIGURE 10-4: TR REQUESTER PTE .....	720
FIGURE 10-5: EXAMPLE PACKET FLOW BETWEEN A REQUESTER, A TR, AND A RESPONDER .....	722
FIGURE 10-6: SERIES OF PACKETS BETWEEN A REQUESTER, A TR, AND A RESPONDER .....	723
FIGURE 10-7: ONE LOGICAL PACKET, MULTIPLE PHYSICAL PACKETS .....	724
FIGURE 10-8: EXAMPLE TR-TO-TR REQUEST-RESPONSE EXCHANGE .....	724
FIGURE 10-9: EXAMPLE TR-TO-TR REQUEST-RESPONSE EXCHANGE, REMOTE BUFFER OPERATION ..	725
FIGURE 11-1: LINK STATE MACHINE .....	728
FIGURE 11-2: 128-BIT LINK-LOCAL GENERAL PACKET FORMAT .....	729
FIGURE 11-3: IMPLICIT FLOW-CONTROL EXAMPLE.....	731
FIGURE 11-4: MID AND WRAP-AROUND FLOW-CONTROL WINDOW EXAMPLES .....	734
FIGURE 11-5: LINK SINGLE-VC FC LLR ACK FLOW-CONTROL PACKET FORMAT .....	737
FIGURE 11-6: LINK DUAL-VC FLOW-CONTROL PACKET FORMAT.....	738
FIGURE 11-7: EXAMPLE FLOW-CONTROL PACKET EXCHANGE .....	739
FIGURE 11-8: LINK IDLE PACKET FORMAT .....	741
FIGURE 11-9: EXAMPLE LINK RESYNCHRONIZATION EXCHANGE.....	742
FIGURE 11-10: LINK CTL GENERAL PACKET FORMAT.....	745
FIGURE 11-11: EXAMPLE ERROR-FREE LLR PACKET EXCHANGE.....	755
FIGURE 11-12: EXAMPLE ERROR HANDLING LLR PACKET EXCHANGE .....	756
FIGURE 11-13: LLR PACKET FORMAT .....	759
FIGURE 11-14: LINK-LOCAL PACKET VALIDATION .....	761
FIGURE 11-15: NON-AFC LINK-LOCAL FLOW-CONTROL PROCESSING.....	763
FIGURE 11-16: LINK-LOCAL LINK CTL PROCESSING .....	765
FIGURE 11-17: LINK-LOCAL SYNCHRONIZATION PROCESSING .....	766
FIGURE 11-18: LINK-LOCAL LINK RP TRANSMISSION PROCESSING.....	768
FIGURE 11-19: LLR PROCESSING.....	770
FIGURE 11-20: LINK-LOCAL ERROR RETRAIN PROCESSING .....	772
FIGURE 12-1: UNICAST PACKET REQUEST-ACKNOWLEDGMENT EXCHANGE .....	774
FIGURE 12-2: UNICAST PACKET TIMEOUT WITH RETRANSMISSION EXAMPLE .....	776
FIGURE 12-3: NON-DETERMINISTIC REQUEST RELIABLE DELIVERY EXCHANGE.....	778
FIGURE 12-4: NON-IDEMPOTENT REQUEST PACKET PROCESSING .....	780
FIGURE 12-5: EXAMPLE SUCCESSFUL NIR-RESPONSE-RELEASE EXCHANGE.....	782
FIGURE 12-6: EXAMPLE NIR-RESPONSE-RELEASE EXCHANGE WITH RETRIES.....	783
FIGURE 12-7: PACKET TYPE VALIDATION .....	787
FIGURE 12-8: P2P-CORE PACKET PROCESSING .....	789
FIGURE 12-9: P2P-COHERENCY PACKET PROCESSING .....	792
FIGURE 12-10: DESTINATION COMPONENT PACKET PROCESSING (PART 1).....	794
FIGURE 12-11: DESTINATION COMPONENT PACKET PROCESSING (PART 2).....	796
FIGURE 12-12: DESTINATION COMPONENT PACKET PROCESSING (PART 3).....	798
FIGURE 12-13: COMPONENT ERROR PROCESSING .....	800
FIGURE 12-14: SWITCH PACKET PROCESSING (PART 1) .....	802

FIGURE 12-15: SWITCH PACKET VALIDATION AND PROCESSING (PART 2) .....	805
FIGURE 12-16: SWITCH MULTICAST PROCESSING .....	807
FIGURE 12-17: TR UNICAST PACKET PART 1 VALIDATION AND PROCESSING .....	808
FIGURE 12-18: TR END-TO-END UNICAST PACKET PART 2 VALIDATION .....	810
FIGURE 12-19: TR P2P-CORE OPCLASS PACKET PROCESSING .....	812
FIGURE 12-20: TR MULTICAST PROCESSING .....	814
FIGURE 12-21: MP AND SECURITY ERROR DETECTION .....	815
FIGURE 12-22: COMPONENT ACCESS KEY VALIDATION .....	819
FIGURE 12-23: INTERFACE ACCESS KEY VALIDATION .....	820
FIGURE 12-24: ACCESS PERMISSION VALIDATION .....	821
FIGURE 12-25: TRANSIENT ERROR PACKET CLEAN-UP PROCESSING .....	824
FIGURE 13-1: PHYSICAL LAYER STATE MACHINE .....	833
FIGURE 13-2: DATA STRIPING ON A FOUR-LANE LINK WITH A 64-BIT PLA INTERFACE .....	835
FIGURE 13-3: DATA STRIPING ON A TWO-LANE LINK WITH A 128-BIT PLA INTERFACE .....	835
FIGURE 13-4: TRANSPORTATION OF FLIT ON FOUR LANE LINK .....	839
FIGURE 13-5: OPTIONAL FLIT ENCODE/DECODE LAYER .....	840
FIGURE 13-6: EXAMPLE IMPLEMENTATION OF FLIT ENCODE LAYER .....	841
FIGURE 13-7: EXAMPLE IMPLEMENTATION OF FLIT DECODE LAYER .....	842
FIGURE 13-8: PLA INTERFACE .....	843
FIGURE 13-9: PLA PHY-DOWN TO PHY-UP STATE TRANSITION .....	845
FIGURE 13-10: PLA PHY-DOWN-RETRAIN TO PHY-UP STATE TRANSITION .....	845
FIGURE 13-11: PLA TX DATA INTERFACE WITH $\langle \text{TxRequestToValidDelay} \rangle = 2$ .....	846
FIGURE 13-12: PLA TX DATA INTERFACE WITH $\langle \text{TxRequestToValidDelay} \rangle = 1$ .....	847
FIGURE 13-13: PLA TX DATA INTERFACE WITH IDLE SYMBOL SUPPORT .....	847
FIGURE 13-14: PLA RX DATA INTERFACE .....	848
FIGURE 13-15: PLA TRANSIENT ERROR .....	849
FIGURE 13-16: PLA RECEIVER RETRAIN ERRORS .....	849
FIGURE 14-1: EXAMPLE UNRELIABLE MULTICAST WITH INTERVENING COMPONENT REPPLICATION .....	850
FIGURE 14-2: UNRELIABLE MULTICAST WRITE REQUEST PACKET FORMAT .....	852
FIGURE 14-3: UNRELIABLE MULTICAST WRITE MSG REQUEST PACKET FORMAT .....	853
FIGURE 14-4: UNRELIABLE MULTICAST ENCAPSULATION REQUEST PACKET FORMAT .....	854
FIGURE 14-5: RELIABLE MULTICAST WRITE AND WRITE PERSISTENT REQUEST PACKET FORMAT .....	856
FIGURE 14-6: RELIABLE MULTICAST WRITE MSG REQUEST PACKET FORMAT .....	857
FIGURE 14-7: RELIABLE MULTICAST ENCAPSULATION REQUEST PACKET FORMAT .....	859
FIGURE 14-8: RELIABLE MULTICAST ACKNOWLEDGMENT PACKET FORMAT .....	861
FIGURE 14-9: RELIABLE MULTICAST WRITE WITH ASSOCIATED ACKNOWLEDGMENT .....	862
FIGURE 14-10: RELIABLE MULTICAST THROUGH INTERVENING COMPONENT .....	862
FIGURE 14-11: RELIABLE MULTICAST WITH LOST PACKET ERROR RECOVERY .....	862
FIGURE 15-1: CONGESTION SUB-FIELDS FORMAT .....	866
FIGURE 15-2: EXPLICIT OPCLASS PACKET FLOW AND DEADLINE UPDATES .....	868
FIGURE 16-1: ATTACK PATHOLOGIES .....	871
FIGURE 16-2: ESTABLISHING A SHARED TIK USING KEY DISTRIBUTION SERVER .....	878
FIGURE 17-1: SINGLE SOD EXAMPLE .....	880
FIGURE 17-2: MULTIPLE SOD EXAMPLE .....	880
FIGURE 17-3: REQUEST AND RESPONSE CONTEXT IDs USING SODs .....	881
FIGURE 17-4: GENERATING SOD REQUEST PACKETS .....	882
FIGURE 17-5: SOD REQUEST AND RESPONSE PACKET FLOW .....	883
FIGURE 17-6: SOD WRITE MSG PACKET SEQUENCE .....	883
FIGURE 17-7: SOD READ REQUEST .....	883
FIGURE 17-8: SOD READ REQUEST PACKET FORMAT .....	888
FIGURE 17-9: SOD READ RESPONSE PACKET FORMAT .....	890

FIGURE 17-10: SOD WRITE REQUEST PACKET FORMAT .....	890
FIGURE 17-11: SOD WRITE MSG PACKET FORMAT .....	891
FIGURE 17-12: SOD STANDALONE ACKNOWLEDGEMENT PACKET FORMAT.....	892
FIGURE 17-13: SOD ENCAPSULATION PACKET FORMAT .....	893
FIGURE 17-14: SOD INTERRUPT PACKET FORMAT.....	894
FIGURE 17-15: SOD SYNC PACKET FORMAT .....	894
FIGURE 17-16: SOD OPCLASS REQUEST PACKET PROCESSING .....	895
FIGURE 17-17: SOD OPCLASS RESPONSE PACKET PROCESSING.....	897
FIGURE 18-1: EXAMPLE SYSTEM WITH GEN-Z I/O COMPONENTS AND PCI I/O DEVICES.....	898
FIGURE 18-2: EXAMPLE SYSTEM WITH LOGICAL AND PHYSICAL PCI I/O DEVICES .....	899
FIGURE 18-3: POTENTIAL PROTOCOL DEADLOCK CASE FOR PCO .....	903
FIGURE 18-4: SIMPLE SINGLE-HOST TOPOLOGIES .....	905
FIGURE 18-5: SINGLE-HOST NON-TREE TOPOLOGIES .....	906
FIGURE 18-6: SIMPLE MULTI-HOST TOPOLOGIES .....	907
FIGURE 18-7: SINGLE-QUEUE SWITCH ARCHITECTURE.....	907
FIGURE 18-8: EXAMPLE VOQ SWITCH ARCHITECTURE .....	908
FIGURE 18-9: CONCEPTUAL PECAM ACCESS TO LOCAL PCI DEVICE CONFIGURATION SPACE .....	909
FIGURE 18-10: CONCEPTUAL EXTENSION OF PCI EXPRESS ECAM LOGIC TO SUPPORT PECAM LOGIC .....	910
FIGURE 18-11: PECAM MAPPING TO A COMPONENT AND PCI CONFIGURATION SPACE IN CONTROL SPACE .....	911
FIGURE 18-12: EXAMPLE PECAM ENUMERATION AND CONFIGURATION STEPS .....	912
FIGURE 18-13: PECAM MAPPING TO CONTROL SPACE AND DATA SPACE .....	914
FIGURE 18-14: MMIO1 AND MMIOH APERTURES IN DATA SPACE.....	920
FIGURE 18-15: PECAM MAPPING TO MULTIPLE TYPE 0 CONFIGURATION SPACE HEADERS .....	923
FIGURE 18-16: COMPONENT LPD STRUCTURE FORMAT .....	924
FIGURE 19-1: EXAMPLE GEN-Z I/O COMPONENT WITH ONE LPH .....	930
FIGURE 19-2: KEY LPH CONTROL SPACE AND DATA SPACE STRUCTURES .....	931
FIGURE 19-3: BDF NUMBER MAPPINGS BY AN LPH ECAM.....	933
FIGURE 19-4: COMPONENT LPH STRUCTURE FORMAT .....	939
FIGURE 20-1: TRANSLATION REQUEST-TRANSLATION RESPONSE PACKET EXCHANGE.....	944
FIGURE 20-2: TRANSLATION INVALIDATE REQUEST-TRANSLATION INVALIDATE RESPONSE PACKET EXCHANGE .....	945
FIGURE 20-3: PRG REQUEST-PRG RESPONSE PACKET EXCHANGE.....	946
FIGURE 20-4: P2P-COHERENCY TRANSLATION REQUEST PACKET FORMAT .....	950
FIGURE 20-5: CTXID TRANSLATION REQUEST PACKET FORMAT .....	951
FIGURE 20-6: EXAMPLE TRANSLATION RANGE SIZES VIA TRANSLATED ADDRESS AND S FIELDS .....	955
FIGURE 20-7: P2P-COHERENCY TRANSLATION RESPONSE PACKET FORMAT.....	955
FIGURE 20-8: CTXID TRANSLATION RESPONSE PACKET FORMAT.....	956
FIGURE 20-9: P2P-COHERENCY TRANSLATION INVALIDATE REQUEST PACKET FORMAT .....	958
FIGURE 20-10: CTXID TRANSLATION INVALIDATE REQUEST PACKET FORMAT .....	959
FIGURE 20-11: P2P-COHERENCY TRANSLATION INVALIDATE RESPONSE PACKET FORMAT .....	960
FIGURE 20-12: CTXID TRANSLATION INVALIDATE RESPONSE PACKET FORMAT.....	961
FIGURE 20-13: P2P-COHERENCY PRG REQUEST PACKET FORMAT .....	964
FIGURE 20-14: CTXID PRG REQUEST PACKET FORMAT .....	964
FIGURE 20-15: P2P-COHERENCY PRG RESPONSE NOTIFICATION PACKET FORMAT .....	967
FIGURE 20-16: CTXID PRG RESPONSE NOTIFICATION PACKET FORMAT .....	967
FIGURE 20-16: P2P-COHERENCY PRG RELEASE / PRG EVICTION NOTIFICATION PACKET FORMAT.....	968
FIGURE 20-11: CTXID PRG RELEASE / PRG EVICTION NOTIFICATION PACKET FORMAT .....	968
FIGURE 20-19: P2P-COHERENCY STOP MARKER PACKET FORMAT .....	970
FIGURE 20-20: CTXID STOP MARKER PACKET FORMAT .....	971

FIGURE B-1: 6-BIT CRC CALCULATION.....	975
FIGURE B-2: VC REMAPPED PCRC MODIFICATION .....	976
FIGURE B-3: 24-BIT CRC CALCULATION.....	978
FIGURE B-4: VC REMAPPED AND CONGESTION ECRC MODIFICATION.....	978
FIGURE B-5: 16-BIT FLIT CRC CALCULATION.....	980

# Tables

TABLE 3-1: DAISY CHAIN REQUEST PROCESSING DETAILS .....	90
TABLE 3-2: DAISY CHAIN RESPONSE PROCESSING DETAILS .....	91
TABLE 3-3: EXAMPLE READ MEDIA PROCESSING DETAILS .....	95
TABLE 3-4: EXAMPLE MEDIA DEVICE SPARING DETAILS .....	100
TABLE 3-5: EXAMPLE MEDIA INITIALIZATION FLOW DETAILS .....	102
TABLE 3-6: POINTER-PAIR TABLE ENTRY FIELDS .....	112
TABLE 3-7: REQUESTER PTE FIELDS .....	120
TABLE 3-8: RESPONDER PTE FIELDS .....	126
TABLE 4-1: EXPLICIT END-TO-END OPCLASS FIELD ENCODINGS .....	128
TABLE 4-2: P2P-CORE OPCLASS OPCODES .....	130
TABLE 4-3: P2P-CORE SUB-OP 1 RESPONSE OPCODES .....	132
TABLE 4-4: P2P-CORE SUB-OP 1 REQUEST OPCODES .....	132
TABLE 4-5: P2P-COHERENCY OPCLASS OPCODES .....	134
TABLE 4-6: P2P-COHERENCY SUB-OP 1 RESPONSE OPCODES .....	135
TABLE 4-7: P2P-COHERENCY SUB-OP 1 REQUEST OPCODES .....	135
TABLE 4-8: CORE 64 OPCLASS UNIQUE OPCODES .....	137
TABLE 4-9: CONTROL OPCLASS OPCODES .....	138
TABLE 4-10: ATOMICS 1 OPCLASS OPCODES .....	139
TABLE 4-11: LDM 1 OPCLASS OPCODES .....	141
TABLE 4-12: ADVANCED 1 OPCLASS OPCODES .....	142
TABLE 4-13: ADVANCED 2 OPCLASS OPCODES .....	142
TABLE 4-14: MULTICAST OPCLASS OPCODES .....	143
TABLE 4-15: SOD OPCLASS OPCODES .....	143
TABLE 4-16: CTXID OPCLASS OPCODES .....	144
TABLE 5-1: COMMON END-TO-END PACKET PROTOCOL FIELDS .....	148
TABLE 5-2: P2P-CORE PACKET PROTOCOL FIELDS .....	149
TABLE 5-3: P2P-COHERENCY PACKET PROTOCOL FIELDS .....	150
TABLE 5-4: COMMON EXPLICIT OPCLASS END-TO-END PACKET PROTOCOL FIELDS .....	150
TABLE 5-5: COMMON OPCODE-SPECIFIC BITS .....	152
TABLE 5-6: LPD-SPECIFIC FIELDS .....	157
TABLE 6-1: P2P-CORE READ OFFSET REQUEST-SPECIFIC PACKET FIELDS .....	166
TABLE 6-2: P2P-COHERENCY READ REQUEST PACKET FIELDS .....	167
TABLE 6-3: CORE 64 READ REQUEST PACKET FIELDS .....	167
TABLE 6-4: LDM 1 READ REQUEST-SPECIFIC PACKET FIELDS .....	168
TABLE 6-5: COMMON READ RESPONSE PACKET FIELDS .....	169
TABLE 6-6: P2P-CORE READ RESPONSE-SPECIFIC PACKET FIELDS .....	170
TABLE 6-7: CORE 64 READ RESPONSE-SPECIFIC PACKET FIELDS .....	170
TABLE 6-8: LDM 1 READ RESPONSE-SPECIFIC PACKET FIELDS .....	171
TABLE 6-9: P2P-CORE WRITE OFFSET REQUEST-SPECIFIC PACKET FIELDS .....	177
TABLE 6-10: P2P-COHERENCY WRITE REQUEST-SPECIFIC PACKET FIELDS .....	177
TABLE 6-11: CORE 64 WRITE REQUEST-SPECIFIC PACKET FIELDS .....	178
TABLE 6-12: (UNRELIABLE) WRITE MSG REQUEST-SPECIFIC PACKET FIELDS .....	185
TABLE 6-13: P2P-CORE WRITE POISON REQUEST-SPECIFIC PACKET FIELDS .....	189
TABLE 6-14: P2P-COHERENCY WRITE POISON REQUEST-SPECIFIC PACKET FIELDS .....	189
TABLE 6-15: CORE 64 WRITE POISON REQUEST-SPECIFIC PACKET FIELDS .....	190
TABLE 6-16: CORE 64 WUM PACKET-SPECIFIC FIELDS .....	191
TABLE 6-17: COMMON WRITE PARTIAL PACKET-SPECIFIC FIELDS .....	192
TABLE 6-18: P2P-COHERENCY WRITE PARTIAL PACKET-SPECIFIC FIELDS .....	193

TABLE 6-19: CORE 64 WRITE PARTIAL PACKET-SPECIFIC FIELDS .....	194
TABLE 6-20: P2P-CORE META WRITE PACKET-SPECIFIC FIELDS .....	195
TABLE 6-21: P2P-COHERENCY META WRITE PACKET-SPECIFIC FIELDS .....	196
TABLE 6-22: P2P-CORE STANDALONE ACKNOWLEDGMENT PACKET FIELDS .....	197
TABLE 6-23: P2P-COHERENCY STANDALONE ACKNOWLEDGMENT PACKET FIELDS .....	198
TABLE 6-24: CORE 64 STANDALONE ACKNOWLEDGMENT PACKET FIELDS .....	198
TABLE 6-25: REASON FIELD ENCODINGS .....	199
TABLE 6-26: RNR NAK TIME INTERVAL ENCODING .....	202
TABLE 6-27: P2P-CORE PERSISTENT FLUSH REQUEST PACKET FIELDS .....	206
TABLE 6-28: CORE 64 PERSISTENT FLUSH REQUEST PACKET FIELDS .....	206
TABLE 6-29: P2P-CORE CTL-READ REQUEST PACKET FIELDS .....	207
TABLE 6-30: P2P-CORE CTL-WRITE REQUEST PACKET FIELDS .....	208
TABLE 6-31: P2P-COHERENCY CTL-READ REQUEST PACKET FIELDS .....	208
TABLE 6-32: P2P-COHERENCY CTL-WRITE REQUEST PACKET FIELDS .....	209
TABLE 6-33: COMMON P2P-CORE AND PCI-COHERENCY CTL-UE REQUEST PACKET FIELDS .....	210
TABLE 6-34: P2P-CORE CTL-UE REQUEST PACKET FIELDS .....	210
TABLE 6-35: P2P-CORE CTL-UE REQUEST PACKET FIELDS .....	211
TABLE 6-36: COMMON CONTROL READ, CONTROL WRITE, AND CONTROL WRITE MSG REQUEST PACKET FIELDS .....	211
TABLE 6-37: CONTROL READ REQUEST PACKET FIELDS .....	214
TABLE 6-38: CONTROL WRITE REQUEST PACKET FIELDS .....	214
TABLE 6-39: CONTROL UNSOLICITED EVENT PACKET-SPECIFIC PACKET FIELDS .....	217
TABLE 6-40: UNSOLICITED EVENT DESCRIPTIONS .....	219
TABLE 6-41: NO-OP PACKET FIELDS .....	226
TABLE 6-42: C-STATE POWER CONTROL PACKET FIELDS .....	229
TABLE 6-43: R-KEY UPDATE PACKET FIELDS .....	231
TABLE 6-44: CONTROL WRITE MSG PACKET FIELDS .....	232
TABLE 7-1: P2P-COHERENCY INTERRUPT PACKET FIELDS .....	237
TABLE 7-2: CORE 64 INTERRUPT PACKET FIELDS .....	238
TABLE 7-3: COMMON ATOMIC PACKET-SPECIFIC FIELDS .....	244
TABLE 7-4: P2P-CORE ATOMIC SINGLE ADDRESS SINGLE OPERAND PACKET-SPECIFIC FIELDS .....	246
TABLE 7-5: P2P-COHERENCY ATOMIC SINGLE ADDRESS SINGLE OPERAND PACKET-SPECIFIC FIELDS .....	247
TABLE 7-6: EXPLICIT ATOMIC 1 SINGLE ADDRESS SINGLE OPERAND PACKET-SPECIFIC FIELDS .....	247
TABLE 7-7: P2P-CORE ATOMIC SINGLE ADDRESS DUAL-OPERAND REQUEST PACKET-SPECIFIC FIELDS .....	248
TABLE 7-8: P2P-COHERENCY ATOMIC SINGLE ADDRESS DUAL-OPERAND REQUEST PACKET-SPECIFIC FIELDS .....	248
TABLE 7-9: EXPLICIT ATOMIC 1 SINGLE ADDRESS DUAL-OPERAND REQUEST PACKET-SPECIFIC FIELDS .....	249
TABLE 7-10: P2P-CORE ATOMIC SINGLE ADDRESS NO OPERAND PACKET-SPECIFIC FIELDS .....	249
TABLE 7-11: P2P-COHERENCY ATOMIC SINGLE ADDRESS NO OPERAND PACKET-SPECIFIC FIELDS .....	250
TABLE 7-12: EXPLICIT ATOMIC 1 SINGLE ADDRESS NO OPERAND PACKET-SPECIFIC FIELDS .....	250
TABLE 7-13: P2P-CORE ATOMIC DUAL-ADDRESS NO OPERAND PACKET-SPECIFIC FIELDS .....	251
TABLE 7-14: P2P-COHERENCY ATOMIC DUAL-ADDRESS NO OPERAND PACKET-SPECIFIC FIELDS .....	251
TABLE 7-15: EXPLICIT ATOMIC 1 DUAL-ADDRESS NO OPERAND PACKET-SPECIFIC FIELDS .....	252
TABLE 7-16: P2P-CORE ATOMIC VECTOR PACKET-SPECIFIC FIELDS .....	252
TABLE 7-17: P2P-COHERENCY ATOMIC VECTOR PACKET-SPECIFIC FIELDS .....	252
TABLE 7-18: P2P-CORE ATOMIC RESPONSE PACKET-SPECIFIC FIELDS .....	253
TABLE 7-19: P2P-COHERENCY ATOMIC RESPONSE PACKET-SPECIFIC FIELDS .....	253
TABLE 7-20: EXPLICIT ATOMIC 1 RESPONSE PACKET-SPECIFIC FIELDS .....	254
TABLE 7-21: P2P-CORE ATOMIC RESULTS RESPONSE PACKET-SPECIFIC FIELDS .....	254

TABLE 7-22: P2P-COHERENCY ATOMIC RESULTS RESPONSE PACKET-SPECIFIC FIELDS .....	255
TABLE 7-23: EXPLICIT ATOMIC RESULTS 1 RESPONSE PACKET-SPECIFIC FIELDS .....	255
TABLE 7-24: P2P-CORE BUFFER PUT LOCAL PACKET-SPECIFIC FIELDS .....	269
TABLE 7-25: BUFFER REQUEST WITH GLOBAL DCID B FIELDS .....	270
TABLE 7-26: BUFFER PUT / GET PACKET-SPECIFIC FIELDS .....	271
TABLE 7-27: SIGNALLED (DYNAMIC) BUFFER PUT / GET PACKET-SPECIFIC FIELDS .....	272
TABLE 7-28: DYNAMIC BUFFER ALLOCATE PACKET-SPECIFIC FIELDS .....	273
TABLE 7-29: DYNAMIC BUFFER RELEASE PACKET-SPECIFIC FIELDS .....	275
TABLE 7-30: BUFFER PUTV / GETV PACKET-SPECIFIC FIELDS .....	276
TABLE 7-31: BUFFER ALLOCATE RESPONSE PACKET-SPECIFIC FIELDS .....	276
TABLE 7-32: PATTERN REQUEST PACKET FIELDS .....	279
TABLE 7-33: PATTERN RESPONSE PACKET FIELDS .....	280
TABLE 7-34: MULTI-OP REQUEST SUB-OPCODES .....	284
TABLE 7-35: MULTI-OP READ REQUEST PACKET OPCODES .....	284
TABLE 7-36: COMMON MULTI-OP PACKET FIELDS .....	285
TABLE 7-37: CORE 64 READ DUAL-OP REQUEST-SPECIFIC PACKET FIELDS .....	286
TABLE 7-38: CORE 64 READ TRIPLE-OP REQUEST-SPECIFIC PACKET FIELDS .....	287
TABLE 7-39: CORE 64 READ QUAD-OP REQUEST-SPECIFIC PACKET FIELDS .....	288
TABLE 7-40: CORE 64 WRITE DUAL-OP REQUEST-SPECIFIC PACKET FIELDS .....	289
TABLE 7-41: CORE 64 WRITE TRIPLE-OP REQUEST-SPECIFIC PACKET FIELDS .....	290
TABLE 7-42: CORE 64 WRITE QUAD-OP REQUEST-SPECIFIC PACKET FIELDS .....	291
TABLE 7-43: CORE 64 SIGNALLED WRITE REQUEST PACKET-SPECIFIC FIELDS .....	292
TABLE 7-44: CORE 64 WRITE-READ REQUEST PACKET-SPECIFIC FIELDS .....	293
TABLE 7-45: CORE 64 WRITE-WAKE REQUEST PACKET-SPECIFIC FIELDS .....	293
TABLE 7-46: P2P-CORE VENDOR-DEFINED REQUEST PACKET FIELDS .....	295
TABLE 7-47: P2P-CORE VENDOR-DEFINED RESPONSE PACKET FIELDS .....	295
TABLE 7-48: P2P-COHERENCY VENDOR-DEFINED REQUEST PACKET FIELDS .....	296
TABLE 7-49: P2P-COHERENCY VENDOR-DEFINED RESPONSE PACKET FIELDS .....	296
TABLE 7-50: EXPLICIT OPCLASS VENDOR-DEFINED PACKET FIELDS .....	297
TABLE 7-51: CORE 64 NIRR PACKET-SPECIFIC FIELDS .....	298
TABLE 7-52: CTXID NIRR PACKET-SPECIFIC FIELDS .....	299
TABLE 7-53: TIME TO MASTER TIME WRAP .....	304
TABLE 7-54: PRECISION TIME REQUEST PACKET-SPECIFIC FIELDS .....	304
TABLE 7-55: PRECISION TIME RESPONSE PACKET-SPECIFIC FIELDS .....	305
TABLE 7-56: COMMON LN PACKET FIELDS .....	311
TABLE 7-57: LN READ REQUEST-SPECIFIC PACKET FIELDS .....	312
TABLE 7-58: LN READ RESPONSE-SPECIFIC PACKET FIELDS .....	312
TABLE 7-59: LN WRITE REQUEST-SPECIFIC PACKET FIELDS .....	313
TABLE 7-60: LN NOTIFICATION REQUEST-SPECIFIC PACKET FIELDS .....	313
TABLE 7-61: CORE 64 WAKE THREAD REQUEST PACKET FIELDS .....	314
TABLE 7-62: P2P-CORE CAPABILITIES READ REQUEST PACKET FIELDS .....	315
TABLE 7-63: UNICAST ENCAPSULATION PACKET PROTOCOL FIELDS .....	316
TABLE 7-64: P2P-COHERENCY READ EXCLUSIVE / EXCLUSIVE / RELEASE / INVALIDATE REQUEST PACKET FIELDS .....	317
TABLE 7-65: CORE 64 READ EXCLUSIVE / EXCLUSIVE RELEASE REQUEST PACKET FIELDS .....	318
TABLE 7-66: P2P-COHERENCY READ SHARED REQUEST PACKET FIELDS .....	319
TABLE 7-67: CORE 64 READ SHARED REQUEST PACKET FIELDS .....	320
TABLE 7-68: P2P-COHERENCY CACHE LINE ATTRIBUTE REQUEST PACKET FIELDS .....	323
TABLE 7-69: CORE 64 CACHE LINE ATTRIBUTE REQUEST PACKET FIELDS .....	323
TABLE 7-70: P2P-COHERENCY CACHE LINE ATTRIBUTE RESPONSE PACKET FIELDS .....	324
TABLE 7-71: CORE 64 CACHE LINE ATTRIBUTE RESPONSE PACKET FIELDS .....	324

TABLE 7-72: P2P-COHERENCY WRITEBACK REQUEST PACKET FIELDS .....	327
TABLE 7-73: CORE 64 INVALIDATE REQUEST PACKET FIELDS .....	328
TABLE 7-74: CORE 64 WRITEBACK REQUEST PACKET FIELDS .....	329
TABLE 7-75: COMMON COLLECTIVE PACKET FIELDS .....	333
TABLE 7-76: BARRIER AND ABORT COLLECTIVE REQUEST PACKET FIELDS .....	337
TABLE 7-77: BROADCAST COLLECTIVE REQUEST PACKET FIELDS .....	338
TABLE 7-78: SCATTER AND GATHER COLLECTIVE REQUEST PACKET FIELDS .....	340
TABLE 7-79: REDUCE COLLECTIVE REQUEST PACKET FIELDS .....	345
TABLE 7-80: COLLECTIVE RESULT REQUEST PACKET FIELDS .....	352
TABLE 7-81: COLLECTIVE RESPONDER COUNT RESPONSE PACKET FIELDS .....	354
TABLE 8-1: INTERFACE STATES AND DESCRIPTIONS .....	355
TABLE 8-2: COMPONENT STATES AND DESCRIPTIONS .....	357
TABLE 8-3: CORE STRUCTURE FIELDS .....	383
TABLE 8-4: CORE C-STATUS .....	406
TABLE 8-5: CORE C-CONTROL .....	408
TABLE 8-6: COMPONENT MAXIMUM ENTRY-EXIT LATENCY AND IDLE TIME ENCODINGS .....	410
TABLE 8-7: COMPONENT CAP 1 FIELD .....	410
TABLE 8-8: COMPONENT CAP 1 CONTROL FIELD .....	415
TABLE 8-9: COMPONENT CAP 2 FIELD .....	424
TABLE 8-10: COMPONENT CAP 2 CONTROL FIELD .....	426
TABLE 8-11: COMPONENT CAP 3 FIELD .....	428
TABLE 8-12: COMPONENT CAP 3 CONTROL FIELD .....	429
TABLE 8-13: COMPONENT CAP 4 FIELD .....	430
TABLE 8-14: COMPONENT CAP 4 CONTROL FIELD .....	430
TABLE 8-15: OP CODE SET STRUCTURE FIELDS .....	434
TABLE 8-16: INTERFACE STRUCTURE FIELDS .....	447
TABLE 8-17: I-STATUS STRUCTURE FIELD .....	459
TABLE 8-18: I-CONTROL SPACE STRUCTURE FIELD .....	461
TABLE 8-19: I-CAP 1 FIELD .....	464
TABLE 8-20: I-CAP 1 CONTROL FIELD .....	467
TABLE 8-21: I-CAP 2 FIELD .....	473
TABLE 8-22: I-CAP 2 CONTROL FIELD .....	473
TABLE 8-23: I-ERROR STATUS .....	474
TABLE 8-24: I-ERROR DETECT .....	474
TABLE 8-25: I-ERROR TRIGGER .....	475
TABLE 8-26: I-ERROR FAULT INJECTION .....	476
TABLE 8-27: INTERFACE PHY STRUCTURE COMMON FIELDS .....	477
TABLE 8-28: PHY STATUS .....	479
TABLE 8-29: PHY CONTROL .....	481
TABLE 8-30: PHY CAP 1 .....	482
TABLE 8-31: PHY CAP 1 CONTROL .....	483
TABLE 8-32: PHY ENTRY AND EXIT LATENCY ENCODINGS .....	484
TABLE 8-33: PHY WORST-CASE RETRAINING TIME ENCODINGS .....	484
TABLE 8-34: PHY EVENTS .....	485
TABLE 8-35: PHY LANE STATUS .....	485
TABLE 8-36: PHY LANE CONTROL .....	486
TABLE 8-37: PHY LANE CAP .....	487
TABLE 8-38: PHY REMOTE LANE CAP .....	488
TABLE 8-39: PHY LOW POWER CONTROL .....	489
TABLE 8-40: PHY LOW POWER TIMING CAPABILITY .....	489
TABLE 8-41: PHY LOW POWER CAP .....	490

TABLE 8-42: PHY REMOTE LOW POWER CAP .....	490
TABLE 8-43: PHY UP LOW POWER CONTROL .....	491
TABLE 8-44: PHY UP LOW POWER TIMING CAPABILITY .....	491
TABLE 8-45: PHY UP LOWER POWER CAP .....	492
TABLE 8-46: PHY UP REMOTE LOWER POWER CAP .....	492
TABLE 8-47: INTERFACE STATISTICS STRUCTURE FIELDS .....	495
TABLE 8-48: COMPONENT MEDIA STRUCTURE FIELDS .....	506
TABLE 8-49: PRIMARY MEDIA STATUS .....	516
TABLE 8-50: SECONDARY MEDIA STATUS .....	518
TABLE 8-51: PRIMARY MEDIA CAP 1 FIELDS .....	520
TABLE 8-52: SECONDARY MEDIA CAP 1 FIELDS .....	525
TABLE 8-53: PRIMARY MEDIA CAP 1 CONTROL .....	529
TABLE 8-54: SECONDARY MEDIA CAP 1 CONTROL .....	532
TABLE 8-55: PRIMARY AND SECONDARY MEDIA LOG ENTRY FIELDS .....	536
TABLE 8-56: COMPONENT MEDIA STRUCTURE FAULT INJECTION FIELD .....	543
TABLE 8-57: RESPONDER BANK STRUCTURE FIELDS .....	547
TABLE 8-58: REQUESTER BANK STRUCTURE FIELDS .....	551
TABLE 8-59: COMPONENT ERROR AND SIGNAL EVENT STRUCTURE FIELDS .....	559
TABLE 8-60: C-ERROR STATUS .....	565
TABLE 8-61: C-ERROR DETECT .....	566
TABLE 8-62: C-ERROR TRIGGER .....	568
TABLE 8-63: C-ERROR FAULT INJECTION .....	569
TABLE 8-64: C-EVENT DETECT .....	570
TABLE 8-65: C-EVENT INJECTION .....	571
TABLE 8-66: I-EVENT DETECT .....	573
TABLE 8-67: I-EVENT INJECTION .....	574
TABLE 8-68: COMPONENT ERROR LOG – COMPONENT ERROR FORMAT FIELDS .....	575
TABLE 8-69: COMPONENT ERROR LOG – COMPONENT INTERFACE ERROR FORMAT FIELDS .....	576
TABLE 8-70: LPRT / MPRT ROUTE ENTRY ROW FIELDS .....	579
TABLE 8-71: VCAT FIELDS .....	581
TABLE 8-72: MCPRT / MSMCPRT ROW FIELDS .....	582
TABLE 8-73: MVCAT FIELDS .....	583
TABLE 8-74: COMPONENT SWITCH STRUCTURE FIELDS .....	584
TABLE 8-75: SWITCH CAP 1 FIELD .....	588
TABLE 8-76: SWITCH CAP 1 CONTROL FIELD .....	589
TABLE 8-77: ROUTE CONTROL SUB-FIELDS .....	590
TABLE 8-78: UNRELIABLE MULTICAST TABLE FIELDS .....	594
TABLE 8-79: RELIABLE MULTICAST TABLE FIELDS .....	595
TABLE 8-80: RELIABLE MULTICAST RESPONDER TABLE FIELDS .....	596
TABLE 8-81: COMPONENT MULTICAST STRUCTURE FIELDS .....	597
TABLE 8-82: COMPONENT EXTENSION STRUCTURE FIELDS .....	602
TABLE 8-83: COMPONENT STATISTICS STRUCTURE FIELDS .....	604
TABLE 8-84: COMPONENT IMAGE STRUCTURE FIELDS .....	608
TABLE 8-85: PRECISION TIME STRUCTURE FIELDS .....	613
TABLE 8-86: COMPONENT MECHANICAL STRUCTURE FIELDS .....	619
TABLE 8-87: MECHANICAL EVENT DETECT .....	628
TABLE 8-88: MECHANICAL EVENT INJECTION .....	629
TABLE 8-89: SSDT / MSDT ROW FIELDS .....	631
TABLE 8-90: REQ-VCAT FIELDS .....	633
TABLE 8-91: RSP-VCAT FIELDS .....	634
TABLE 8-92: RIT FIELDS .....	634

TABLE 8-93: COMPONENT DESTINATION TABLE STRUCTURE FIELDS .....	635
TABLE 8-94: VENDOR-DEFINED STRUCTURE FIELDS .....	644
TABLE 8-95: VENDOR-DEFINED WITH UUID STRUCTURE FIELDS .....	645
TABLE 8-96: COMPONENT SECURITY STRUCTURE FIELDS .....	649
TABLE 8-97: COMPONENT TR STRUCTURE FIELDS .....	656
TABLE 8-98: TR STATUS .....	659
TABLE 8-99: SERVICE UUID STRUCTURE FIELDS .....	661
TABLE 8-100: COMPONENT C-ACCESS STRUCTURE FIELDS .....	664
TABLE 8-101: PEER-ATTR FIELD .....	670
TABLE 8-102: SEC TABLE ENTRY FIELDS .....	672
TABLE 8-103: COMPONENT PA STRUCTURE FIELDS .....	673
TABLE 8-104: COMPONENT EVENT STRUCTURE FIELDS .....	680
TABLE 8-105: EVENT RECORD FIELDS .....	683
TABLE 8-106: COMPONENT SOD STRUCTURE FIELDS .....	685
TABLE 8-107: COMPONENT ATP STRUCTURE FIELDS .....	688
TABLE 8-108: CONGESTION MANAGEMENT STRUCTURE FIELDS .....	695
TABLE 8-109: COMPONENT RKD STRUCTURE FIELDS .....	698
TABLE 8-110: COMPONENT RE TABLE STRUCTURE FIELDS .....	700
TABLE 8-111: PERFORMANCE LOG RECORD FIELDS .....	702
TABLE 8-112: COMPONENT PM STRUCTURE FIELDS .....	704
TABLE 10-1: TR REQUESTER PTE FIELDS .....	721
TABLE 11-1: LINK STATES AND DESCRIPTIONS .....	726
TABLE 11-2: COMMON LINK-LOCAL PACKET PROTOCOL FIELDS .....	729
TABLE 11-3: COMMON LINK-LOCAL 128-BIT PACKET PROTOCOL FIELDS .....	729
TABLE 11-4: LINK-LOCAL PACKET OPCODES .....	730
TABLE 11-5: LINK SINGLE-VC FC LLR ACK FLOW-CONTROL PACKET-SPECIFIC FIELDS .....	738
TABLE 11-6: LINK DUAL-VC FLOW-CONTROL PACKET-SPECIFIC FIELDS .....	738
TABLE 11-7: LINK CTL PACKET FIELDS .....	745
TABLE 11-8: LINK CTL SUB-OPCODES .....	745
TABLE 11-9: LLR PACKET FIELDS .....	759
TABLE 11-10: LINK-LOCAL PACKET VALIDATION DETAILS .....	762
TABLE 11-11: NON-AFC LINK-LOCAL FLOW-CONTROL PROCESSING DETAILS .....	763
TABLE 11-12: LINK-LOCAL LINK CTL PROCESSING DETAILS .....	765
TABLE 11-13: LINK-LOCAL SYNCHRONIZATION PROCESSING DETAILS .....	767
TABLE 11-14: LINK-LOCAL LINK RP TRANSMISSION PROCESSING DETAILS .....	769
TABLE 11-15: LLR PROCESSING DETAILS .....	770
TABLE 11-16: LINK-LOCAL ERROR RETRAIN PROCESSING DETAILS .....	773
TABLE 12-1: NON-IDEMPOTENT REQUEST PACKET DETAILS .....	780
TABLE 12-2: PACKET TYPE VALIDATION DETAILS .....	787
TABLE 12-3: P2P-CORE PACKET PROCESSING DETAILS .....	790
TABLE 12-4: P2P-COHERENCY PACKET PROCESSING DETAILS .....	792
TABLE 12-5: DESTINATION COMPONENT PACKET PROCESSING PART 1 DETAILS .....	795
TABLE 12-6: DESTINATION COMPONENT PACKET PROCESSING PART 2 DETAILS .....	797
TABLE 12-7: DESTINATION COMPONENT PACKET PROCESSING PART 3 DETAILS .....	799
TABLE 12-8: COMPONENT ERROR PROCESSING DETAILS .....	801
TABLE 12-9: SWITCH PACKET PROCESSING PART 1 DETAILS .....	803
TABLE 12-10: SWITCH PACKET VALIDATION AND PROCESSING PART 2 DETAILS .....	806
TABLE 12-11: SWITCH MULTICAST PROCESSING DETAILS .....	807
TABLE 12-12: TR UNICAST PACKET PART 1 VALIDATION PROCESSING DETAILS .....	809
TABLE 12-13: TR END-TO-END UNICAST PACKET PART 2 VALIDATION DETAILS .....	811
TABLE 12-14: TR P2P-CORE OPCLASS PACKET PROCESSING DETAILS .....	813

TABLE 12-15: TR MULTICAST PROCESSING DETAILS .....	814
TABLE 12-16: MP AND SECURITY ERROR DETECTION DETAILS .....	816
TABLE 12-17: COMPONENT ACCESS KEY VALIDATION DETAILS .....	819
TABLE 12-18: INTERFACE ACCESS KEY VALIDATION DETAILS .....	821
TABLE 12-19: ACCESS PERMISSION VALIDATION DETAILS .....	822
TABLE 12-20: TRANSIENT ERROR PACKET CLEAN-UP PROCESSING DETAILS .....	825
TABLE 13-1: REQUIRED PHY STATES.....	832
TABLE 13-2: PHY LOW-POWER STATES.....	837
TABLE 13-3: CONTROL AND STATUS INTERFACE SIGNALS .....	844
TABLE 13-4: TRANSMITTER INTERFACE SIGNALS .....	845
TABLE 13-5: RECEIVER INTERFACE SIGNALS .....	847
TABLE 14-1: UNRELIABLE MULTICAST-SPECIFIC WRITE REQUEST PACKET FIELDS .....	852
TABLE 14-2: UNRELIABLE MULTICAST-SPECIFIC WRITE MSG REQUEST PACKET FIELDS .....	853
TABLE 14-3: UNRELIABLE MULTICAST ENCAPSULATION-SPECIFIC REQUEST PACKET FIELDS .....	855
TABLE 14-4: RELIABLE MULTICAST-SPECIFIC WRITE REQUEST PACKET FIELDS.....	857
TABLE 14-5: RELIABLE MULTICAST-SPECIFIC WRITE MSG REQUEST PACKET FIELDS .....	858
TABLE 14-6: RELIABLE MULTICAST ENCAPSULATION-SPECIFIC REQUEST PACKET FIELDS .....	859
TABLE 14-7: RELIABLE MULTICAST ACKNOWLEDGMENT PACKET FIELDS.....	861
TABLE 15-1: CONGESTION SUB-FIELDS .....	866
TABLE 17-1: COMMON SOD PACKET PROTOCOL FIELDS.....	887
TABLE 17-2: SOD READ REQUEST-SPECIFIC PROTOCOL FIELDS .....	888
TABLE 17-3: SOD READ RESPONSE-SPECIFIC PROTOCOL FIELDS .....	890
TABLE 17-4: SOD WRITE REQUEST-SPECIFIC PROTOCOL FIELDS .....	891
TABLE 17-5: SOD STANDALONE ACKNOWLEDGMENT PROTOCOL FIELDS.....	892
TABLE 17-6: SOD ENCAPSULATION PROTOCOL FIELDS .....	893
TABLE 17-7: SOD SYNC PROTOCOL FIELDS .....	895
TABLE 17-8: SOD OPCLASS REQUEST PACKET PROCESSING DETAILS .....	896
TABLE 17-9: SOD OPCLASS RESPONSE PACKET PROCESSING DETAILS.....	897
TABLE 18-1: SIMPLIFIED SUMMARY OF PCIe ORDERING RULES .....	899
TABLE 18-2: PECAM ADDRESS MAPPING .....	909
TABLE 18-3: PECAM ENUMERATION AND CONFIGURATION DETAILS.....	913
TABLE 18-4: PCI CAPABILITY STRUCTURE SUPPORT LEVELS .....	916
TABLE 18-5: PCI EXPRESS EXTENDED CAPABILITY STRUCTURE SUPPORT LEVELS.....	917
TABLE 18-6: COMPONENT LPD STRUCTURE FIELDS .....	924
TABLE 19-1: COMPONENT LPH STRUCTURE FIELDS .....	939
TABLE 20-1: COMMON ADDRESS TRANSLATION AND PAGE PROTOCOL FIELDS .....	947
TABLE 20-2: COMMON TRANSLATION REQUEST PACKET PROTOCOL FIELDS .....	949
TABLE 20-3: P2P-COHERENCY TRANSLATION REQUEST PACKET-SPECIFIC PROTOCOL FIELDS .....	950
TABLE 20-4: CTXID TRANSLATION REQUEST PACKET-SPECIFIC PROTOCOL FIELDS .....	951
TABLE 20-5: COMMON TRANSLATION RESPONSE PACKET PROTOCOL FIELDS .....	952
TABLE 20-6: P2P-COHERENCY TRANSLATION RESPONSE PACKET-SPECIFIC PROTOCOL FIELDS.....	956
TABLE 20-7: CTXID TRANSLATION RESPONSE PACKET-SPECIFIC PROTOCOL FIELDS .....	956
TABLE 20-8: COMMON TRANSLATION INVALIDATE REQUEST PACKET PROTOCOL FIELDS .....	957
TABLE 20-9: P2P-COHERENCY TRANSLATION INVALIDATE REQUEST PACKET-SPECIFIC PROTOCOL FIELDS .....	958
TABLE 20-10: CTXID TRANSLATION REQUEST PACKET-SPECIFIC PROTOCOL FIELDS .....	959
TABLE 20-11: P2P-COHERENCY TRANSLATION INVALIDATE RESPONSE PACKET PROTOCOL FIELDS....	960
TABLE 20-12: CTXID TRANSLATION INVALIDATE RESPONSE PACKET PROTOCOL FIELDS.....	961
TABLE 20-13: COMMON PRG REQUEST PACKET PROTOCOL FIELDS.....	962
TABLE 20-14: P2P-COHERENCY PRG REQUEST PACKET-SPECIFIC PROTOCOL FIELDS.....	964
TABLE 20-15: CTXID PRG REQUEST PACKET-SPECIFIC PROTOCOL FIELDS.....	964

TABLE 20-16: COMMON PRG RESPONSE NOTIFICATION PACKET PROTOCOL FIELDS.....	965
TABLE 20-17: P2P-COHERENCY PRG RESPONSE NOTIFICATION PACKET-SPECIFIC PROTOCOL FIELDS.....	967
TABLE 20-18: CTXID PRG RESPONSE NOTIFICATION PACKET-SPECIFIC PROTOCOL FIELDS.....	967
TABLE 20-19: P2P-COHERENCY PRG RELEASE / PRG EVICTION NOTIFICATION PACKET PROTOCOL FIELDS .....	968
TABLE 20-20: CTXID PRG RELEASE / PRG EVICTION NOTIFICATION PACKET PROTOCOL FIELDS.....	969
TABLE 20-21: P2P-COHERENCY STOP MARKER PACKET PROTOCOL FIELDS .....	970
TABLE 20-22: CTXID STOP MARKER PACKET PROTOCOL FIELDS .....	971
TABLE A-1: CONTROL SPACE STRUCTURE TYPE ENCODINGS .....	972
TABLE B-1: MAPPING 6-BIT CRC CALCULATED BITS INTO THE CRC FIELD .....	974
TABLE B-2: MAPPING 24-BIT CRC CALCULATED BITS INTO THE CRC FIELD .....	977
TABLE B-3: MAPPING 16-BIT CRC CALCULATED BITS INTO THE 16-BIT CRC FIELD .....	979
TABLE C-1: COMPONENT CLASS ENCODINGS .....	982

# 1. Core Architecture

## 1.1. Functional Overview

Gen-Z is a scalable, universal system interconnect intended to unify communications, simplify designs and solution architectures, and improve interoperability. Gen-Z uses a memory-semantic (read-write) protocol that enables multiple component types to efficiently communicate. Component types include processors, memory, storage, I/O, FPGA, GPU, GPGPU, DSP, etc. Universal communications simplifies component design and solution composition making Gen-Z applicable to multiple solution types including clients (mobility, mobile, desktops / workstations), servers, storage, embedded, and message-based communications.

Gen-Z architecture is driven by the following high-level use cases with a focus on simplifying and / or eliminating software involvement in the main data path:

- Byte-addressable memory access—Gen-Z supports volatile and non-volatile memory media accessed using processor-centric and memory-centric solution architectures.
- Block memory access—multiple operational models including traditional queue-based, variety of buffer-based (put and get), and hybrid byte and block-based.
- I/O device access—Gen-Z supports direct communications between all component types (NIC, graphics, FPGA, DSP, etc.) and eliminates the cost and complexity of traditional multi-protocol architectures. I/O components may support native Gen-Z devices to fully exploit Gen-Z architecture, or *Logical PCI Device (LPD)* or *Logical PCI Hierarchy (LPH)* that provides PCI / PCIe software compatibility and the ability to partially or fully exploit Gen-Z architecture.
- Messaging—Gen-Z supports multiple messaging solutions such as a traditional network stack using an emulated Ethernet NIC (eNIC), messaging with existing networks using gateway (e.g., a Gen-Z-to-Ethernet gateway), and ultra-low-latency messaging (high-performance computing, enterprise, and hyperscale market segments).
- Accelerator access—Gen-Z supports coherent and non-coherent communications to accelerators. Accelerators can be attached using point-to-point, meshed, or switch-based topologies.

The high-level architectural attributes are as follows.

- Gen-Z is optimized to support memory-semantic communications using a packet-based transport to exchange operations that access a component's resources (e.g., memory media devices) or cause a component to take specific actions relative to the targeted component resources. Packets are asynchronously (non-time based) exchanged between components.
- Gen-Z uses a split memory controller and media controller paradigm that hides microarchitecture details and idiosyncrasies. The split-controller model breaks the processor-memory interlock.
  - Processor memory controller issues high-level packets—reads, writes, atomics, large data movement buffer requests, etc.
  - Media controller services high-level packets and performs media-specific services and management.
  - Split controller paradigm enables memory media independence (supports volatile and non-volatile media technologies). Solutions can incorporate and evolve the memory media and ensure interoperability. Media controllers can also incorporate performance acceleration techniques (prefetch, caching, etc.) to mitigate media-specific access

variabilities as well as data-centric accelerators (embedded processor or logic) to improve overall solution performance.

- Gen-Z is also applicable to non-memory applications including I/O, storage, and message passing communications.
- Gen-Z specifies an abstract physical layer interface enabling support of multiple physical layer implementations and media that can be tailored to solution-specific needs. Physical layers can evolve or be replaced without disrupting the ecosystem or waiting for the entire ecosystem to move in lock-step.
- Gen-Z provides architectural flexibility:
  - Supports processor-centric and memory-centric architectures. Processor-centric refers to an architecture where all memory accesses are required to flow through the processor memory controller. Memory-centric refers to an architecture where components can directly access memory without flowing through the processor memory controller.
  - Uses a common data transport with application semantic overlays to support diverse component types. A common data transport simplifies solution design, enables memory-centric architecture, enables solution and component cost reductions (e.g., through Intellectual Property block re-use), and enables agile hardware development and solution composition / integration.
  - Solution-driven packaging and topologies
    - Discrete and co-packaged (2.5/3D) components
    - Point-to-point topologies (one or more links connecting components in meshed or daisy-chain topologies)
    - Switched topologies using component-integrated or discrete switches to relay packets towards their destinations.
  - Symmetric and asymmetric interfaces and links to match workload requirements
  - Real-time dynamic physical signaling rate, interface width, and link width adjustment to meet workload or environmental conditions (typically interlocked with the underlying physical layer's capabilities).
  - Volatile and non-volatile / persistent memory based on memory media and application signaling.
  - Differentiated communication services to prioritize and arbitrate data flows across shared hardware resources
  - Advanced requests, e.g., atomics, buffer movement, pattern matching / regular expression acceleration, etc. as well as transparent support for vendor-defined operations.
  - CRC-based packet data integrity combined with transparent end-to-end packet error recovery.
- Gen-Z specifies resiliency technologies:
  - Robust error detection and recovery with isolation
  - Component and resource access control with hardware-enforced isolation
  - Performance isolation
  - Mechanical serviceability
- Gen-Z assumes every component is an attack vector, hence security is built into the architecture:
  - Strong packet authentication
  - Anti-replay attack protection
  - Strong image authentication
  - Transparently transports encrypted data

- Gen-Z enables solution scalability:
  - Scalable component resource provisioning—up to  $2^{44}$  or  $2^{64}$  addressable resource per component or logical resource
  - A component may support 1-4096 component interfaces of varying widths.
  - Multipath link aggregation and resiliency
  - Support from 2 to  $2^{12}$  components per subnet
    - A subnet may be a point-to-point optimized topology. Such a topology does not use component identifiers to exchange packets.
    - A subnet may consist of point-to-point and switch-based topologies that use component identifiers to exchange packets.
  - Multiple subnets per component, e.g., a Requester such as a processor with four interfaces can be attached to a single subnet or up to four distinct subnets.
  - Subnets may be joined together to increase fan-out, provide hardware-enforced isolation, to enable diverse mechanical topologies, and to enable hardware-enforced error or failure containment.
  - Subnets can be transparently joined together such that components are unaware they are communicating across subnets. Transparency enables services to be interposed without requiring application or processor modification or coordination. For example, in storage class memory solutions where processors perform load-store operations (read-write), the architecture enables solutions to insert common and new storage-focused rich data services such as back-up, deduplication, compression, encryption, data-centric accelerator technologies, data reduction etc.
  - Subnets can be visibly joined together such that components are aware they are communicating across subnets. Solutions composed of such components can scale to  $2^{16}$  subnets.

This specification covers the core architecture and associated functionality as well as the physical layer abstraction. Additional specifications will cover specific physical layers, mechanical packaging, firmware, etc.

**Developer Note:** *The architecture mandates only a fraction of the specified functionality to ensure basic interoperability (read-write communications). So, although the architecture is extremely comprehensive and addresses a multitude of possible solutions, developers need only implement the select functionality required to provide basic interoperability and to meet their solution-specific needs.*

## 1.2. Operational Overview

At its essence, Gen-Z specifies a request-response protocol used to exchange packets between components. *Example Single Interface Request / Response Exchange* illustrates a Requester (e.g., a processor with an integrated memory controller) transmitting a request packet that results in a response packet being returned by a Responder (e.g., a memory component).

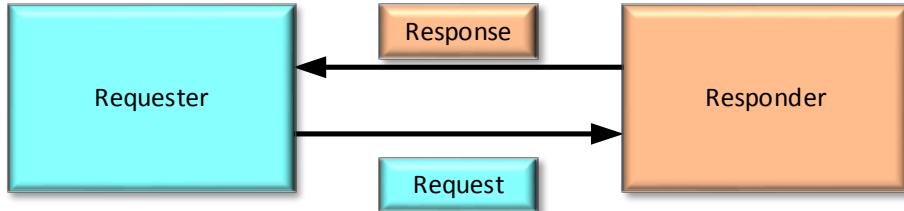


Figure 1-1: Example Single Interface Request / Response Exchange

A Requester shall:

- Generate request packets
- Enforce request packet ordering, e.g., ensure sequential consistency for Semantically Dependent request packets
- Identify the egress interface through which to transmit request packets
- If applicable, ensure end-to-end Reliable Delivery
- Make data visible upon receipt
- Validate and execute response packets
- Await receipt of a response packet (a request-specific response or a *Standalone Acknowledgment*) before making ordering decisions that depend upon the successful execution of a given request packet.

A Responder shall:

- Validate and execute request packets
- Identifying the egress interface through which to transmit response packets
- Generate response packets

The role a component plays within a solution depends upon how an operation is supported. For example, a component that generates read and write request packets and receives and executes the corresponding response packets plays a Requester role. A component that receives and executes read and write request packets and generates the corresponding response packets plays a Responder role. A component that generates and receives read and write request packets plays a Requester-Responder role. Roles are configured on a per operation basis. Throughout this document, wherever 'Requester' is used, the specification equally applies to a Requester-Responder when acting as a Requester. Similarly, wherever 'Responder' is used, the specification equally applies to a Requester-Responder when acting as a Responder.

Unless explicitly stated otherwise by this specification, a component shall be located within a single subnet.

A component contains one or more physical interfaces. Each interface represents a set of physical resources used to communicate with another interface via a link.

- A link shall consist of one or more lanes. A lane may be used to transmit or receive packet bytes. A lane is unidirectional, i.e., transmits packet bytes in only one direction at a time.
- A link may be symmetric or asymmetric. A symmetric link is one where the number of transmitter lanes and receiver lanes is equal. Conversely, an asymmetric link is one where the number of transmitter lanes and receiver lanes is not equal. An asymmetric link provides more bandwidth in a given direction, e.g., a solution with a 3:1 read-write ratio can provision 3x more receive lanes than transmit lanes which can yield ~50% increase in read bandwidth. Symmetric and asymmetric

links may be used in any topology or a portion of a topology to optimize specific paths for a given workload.

- Each link supports one or more physical layer technologies. Physical layer technology type and attributes are solution-specific. A component may support multiple links with each supporting the same or different physical layer technology.
- Depending upon the underlying physical layer capabilities, the number of lanes may be statically provisioned or dynamically adjusted on a per-link basis. Similarly, depending upon the physical layer technology, the physical signaling rate may be statically provisioned or dynamically adjusted.
- A component may support one or more interfaces. Each interface shall be used to transmit and receive packets across the directly-attached link.
  - A component may support non-homogeneous interfaces, i.e., the supported signaling rates, the number of transmit lanes, the number of receive lanes, etc. may vary on a per-interface basis

**Developer Note:** A component is either a Requester, a Responder, or a Requester-Responder. From an architectural perspective, there is a single Requester, a single Responder, or a single Requester plus a single Responder. However, implementations can contain multiple instances, e.g., one Requester and one Responder instance per component interface. These instances are considered part of the single architectural Requester or the single architectural Responder.

### 1.3. Memory-Semantic Components

The architecture enables a wide spectrum of component types to communicate using a single, common interconnect and protocol. Components communicate using memory-semantic requests with application-specific semantic overlays to derive meaning and drive type-specific actions. In essence, these requests act as high-level abstractions that hide the underlying component-specific microarchitectures and associated idiosyncrasies. The architecture specifies a mandatory set of memory-semantic requests (e.g., read and write) and an optional set of requests (e.g., Atomics and buffer movement) to maximize interoperability and differentiation.

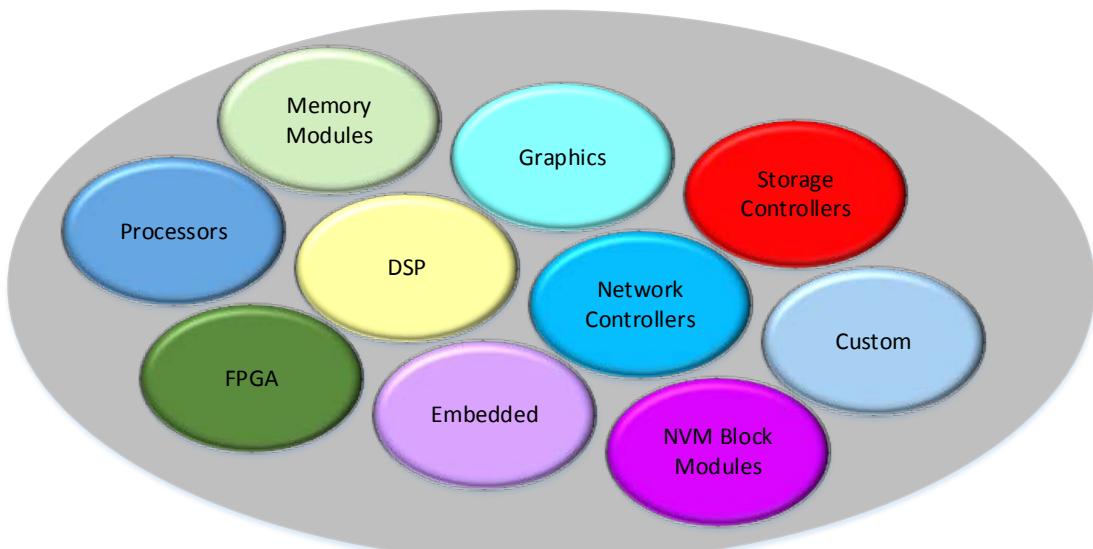


Figure 1-2: Sample Space of Component Types

Components may be interconnected through a variety of topologies driven by solution-specific needs. The simplest topologies will be point-to-point between two or more components as illustrated in *Example Point-to-point Topology with a Mix of Component Types*. The block diagram on the left illustrates a simple mobility solution in which a processor accesses memory and graphics components via point-to-point links. Similarly, a simple server solution accesses memory and network controller components via point-to-point links. Multiple topologies and solutions can be constructed using any mix of component types. Solutions may contain any mix of interconnects and protocols or may converge all communications to use Gen-Z protocols (non-coherent or coherent or a mix of non-coherent and coherent communications).

5

10

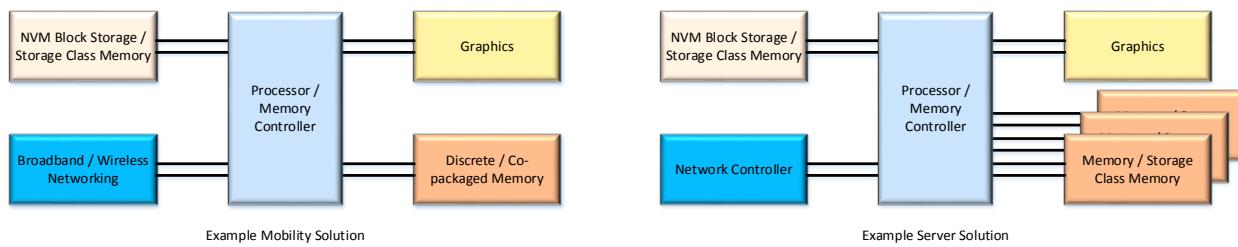


Figure 1-3: Example Point-to-point Topology with a Mix of Component Types

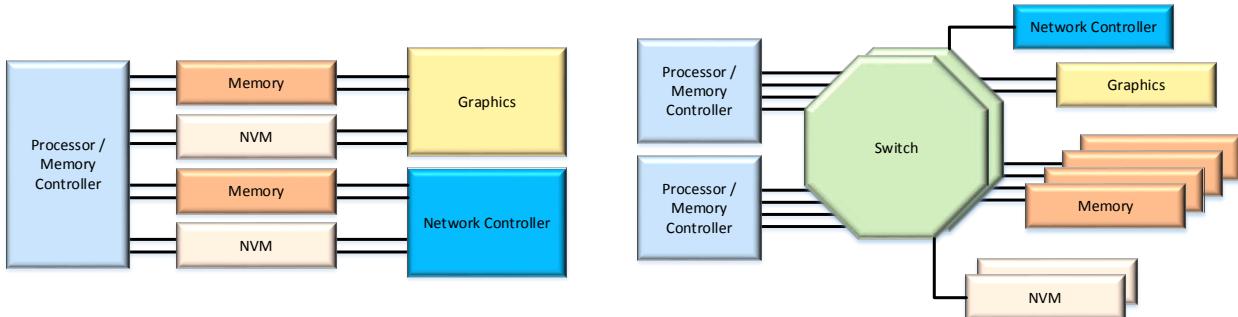


Figure 1-4: Example Point-to-point and Switch-based Gen-Z Topologies with a Mix of Component Types

15

## 1.4. Split Memory / Media Controller Model

Existing computer architectures intertwine and interlock processor memory management and media-specific logic into a single, monolithic controller.



Figure 1-5: Monolithic Memory / Media Controller

20

Gen-Z uses a split controller model as illustrated in *Split Memory and Media Controller*.



Figure 1-6: Split Memory and Media Controller

A comparison of these figures reveals:

- All media-specific logic is isolated to the combined media controller plus memory media (referred to as a memory component). The media's attributes can be completely abstracted or partially exposed depending upon solution-specific needs.
- The media-specific protocol is replaced by the general-purpose Gen-Z protocol, which is void of media-specific knowledge. For example, the protocol communicates simple Read and Write requests with sufficient detail that media-specific operations are not specified within the protocol (e.g., DRAM-based Precharge, Activate, Refresh, etc.).
- Although conceptually both use solution-specific physical interfaces, the split controller model supports multiple physical layer technologies that can be used without comprehending media-specific idiosyncrasies and constraints.
- Although both models support memory media components composed of multiple media die stacks, the split memory model enables more versatile media composition without impacting the processor memory controller or the communication protocol. For example, a media controller can transparently scale from a co-packaged solution to a solution composed of 8, 16, 32, or larger numbers of integrated and / or discrete media die stacks.
- Depending upon the media controller implementation, Gen-Z enables solutions to provision memory anywhere within a topology, to be accessed through multiple paths, to be securely accessed, to be shared by multiple peer components, to support memory interleaving, RAID, and erasure codes, etc. These capabilities enables highly resilient solutions to eliminate single points of failure and the potential for stranded memory resources.

Minimally, the memory controller is responsible for:

- Translation and processing services between processor-specific operations and Gen-Z protocol request and response packets. This includes request packet scheduling and associated ordering (strong vs. relaxed).
- Gen-Z and memory controller-specific management services, e.g., power, error, data integrity, etc.
- Path selection to a given Responder
- Multipath resiliency and performance aggregation management

Minimally, the media controller is responsible for:

- Translating protocol request packets into media-specific operations.
- Media and component-specific data-integrity, resiliency (e.g., media-specific error recovery and sparing), availability, and wear-leveling services.

- 5
- Gen-Z and media controller-specific management services, e.g., power management including refresh for volatile media (i.e., self-refresh), error, statistics, sensors, etc.
  - Media-specific internal communications, e.g., TSV (Through Silicon VIA), silicon interposers, etc.
  - Aggregation and interleaving media-specific operations across one or more media devices (if present).

A media controller may incorporate functionality and services to accelerate performance (bandwidth, latency, and memory operations per second), optimize power and thermal management, minimize data movement and optionally accelerate data manipulation, etc., for example:

- 10
- In general, a traditional DDR memory buffer chip acts as a type of simple cache through which a memory controller accesses the underlying physical media banks. The buffer chip accesses a page or row of data from multiple media devices, and subsequently services memory controller requests directly from the cached row. When the memory controller no longer requires the row, it instructs the buffer chip to write back the row to the media devices. Given its ability to abstract the underlying media, a media controller can build upon this paradigm and provision a larger, more flexible cache.
    - The media controller may advertise logical banks instead of physical banks. Depending upon the size of the cache, the number of logical banks may be many times larger than the number of physical banks, e.g., 256 logical banks compared to 16 or 32 physical banks. This enables a memory controller to have a larger number of outstanding requests per media controller. A larger number of outstanding requests can improve aggregate performance and mitigate load-to-use latency due to request pipeline effects.
    - Depending upon the cache size, a media controller may re-arrange media operations to improve latency, e.g., for DRAM media, instead of performing a Precharge prior to the next Activate, a media controller might use the added cache capacity to issue back-to-back Activates to a given physical bank with the results stored in different cached logical rows. Similarly, a media controller might issue back-to-back Precharge operations to a given physical bank or perform lazy Precharge operations whenever the physical bank is idle to avoid incurring the latency typically associated with a buffer chip-based solution.
    - A media controller may use a portion of the cache to perform volatile memory refresh when physical banks are idle. Depending upon the refresh cycle frequency, this service may be performed in the background, or combined with power optimizations to extend battery life in mobile solutions or to reduce solution power consumption and associated operating expenses.
    - A media controller may apply solution-driven heuristics or other policies to perform prefetch operations to accelerate subsequent media access. These may be combined with other functionality to improve load-to-use latency and aggregate performance.
    - A media controller may consider address spatial locality to optimize access in place of using an all-or-nothing open or closed page policy. For example, if the request is a one-shot random memory access, then the media controller can retain only the portion of the logical row to service the request, and, depending upon the media's properties, the media controller can either discard the rest of the physical row or immediately schedule a Precharge of the physical row without waiting for the memory controller.
    - Given the number of random memory access applications, a media controller may reduce the size of the physical rows to reduce data movement and provide cache optimizations.
  - In general, a media controller is co-located with the media devices and has intimate knowledge of their unique properties and organization, the media controller may:
    - Optimize / tighten media timing budgets to minimize / eliminate wait cycles

- Optimize power management to reduce or smooth power consumption
- Provide real-time thermal management
- Provide automatic wear leveling for non-volatile media
- Provide automatic or software-controlled garbage collection services for non-volatile media
- Provide data migration between multiple tiers of media or diverse types of media
- Provide data integrity and data encryption services
- Provide mailbox communication services to enable memory management
- Support Meta read and write semantics to support value-add functionality, application-level data integrity services, customized caching services, etc.
- Provide dynamic post package repair, e.g., automatic row remapping upon detecting defects or too many correctable / uncorrectable errors during power-on self-test (POST) or during run-time operation
- If a media controller supports discrete or integrated acceleration logic or processing capabilities (aka data-centric acceleration services), then the media controller can improve solution performance by reducing the amount of data moved between the memory controller and the media controller, and off-loading computation and data manipulation operations from a processor. This can reduce overall power consumption.

## 1.5. Topologies

Numerous solutions can be constructed using simple point-to-point topologies. *Example Requester-Responder Topologies* illustrates three example topologies.

- The simplest point-to-point topology (A) contains a single Requester (e.g., processor-memory controller) communicating to a single Responder (e.g., memory component). The Requester and the Responder assume any transmitted packet is destined to the component on the other end of the link, so no additional logic is required to determine if a packet is for the receiver or not.
- The second point-to-point topology (B) is a daisy-chain topology and contains a single Requester and two Responders that are daisy-chained together. A daisy-chain topology requires each Responder to examine a request packet's Tag (uniquely identifies the packet) field to determine if the request is for it or the next hop in the daisy-chain. Responders automatically forward any response packet to the next hop in the daisy-chain.
- The third topology (C) is a switch-based topology which enables a single Requester to communicate with multiple Responders all within a single switch-hop. A variety of single and multi-switch-based topologies are possible enabling communications between any combination of Requesters and Responders. A switch-based topology uses component identifiers embedded in the packets to perform packet relay and ensure packets are delivered to the correct destination component. As a result, packets that operate over switches use a different packet format than packets optimized for point-to-point and daisy-chain topologies.
- The fourth topology (D) includes point-to-point, meshed, and switch-based topologies that support any mix of component types. This example illustrates a set of accelerators that can be connected in a variety of topologies enabling solutions to scale to meet any application needs. Further, the accelerators can communicate using any mix of coherent and non-coherent packets.

Communications may occur over one or more links to increase solution flexibility and aggregate performance. Point-to-point topologies may also be constructed to connect a single Requester with one or more Responders.

Additional capacity and performance scaling can be achieved through the use of component-integrated or discrete switches. A variety of topologies can be constructed from simple linear / daisy chains to star to 3D-Torus.

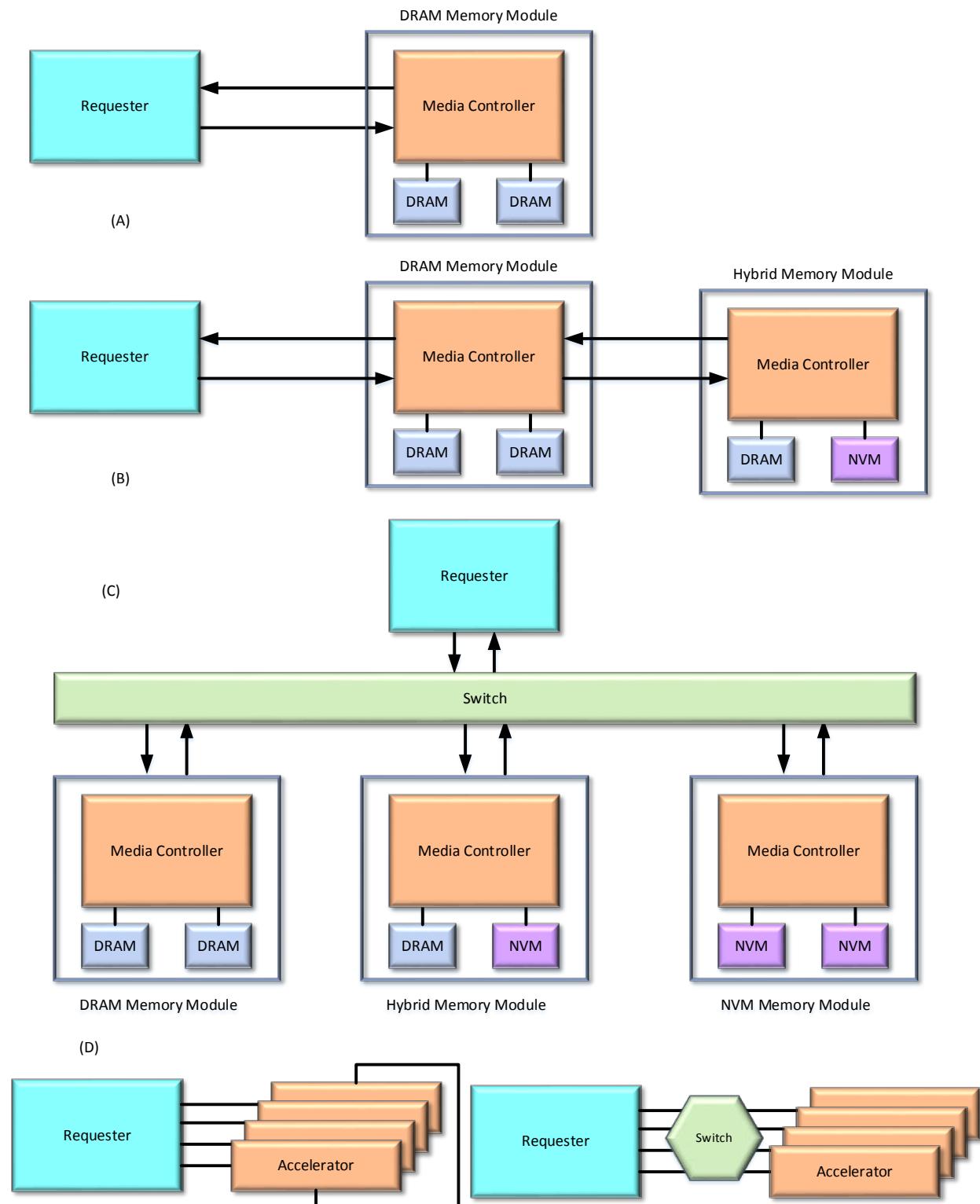


Figure 1-7: Example Requester-Responder Topologies

## 1.6. Switch Overview

Topologies requiring additional scalability may use discrete switch components or components with integrated switches (e.g., a processor or memory component with an integrated switch). Conceptually a switch contains a relay engine to forward packets between interfaces. The relay engine examines a packet's header to determine which egress interface to forward the packet. The relay engine might examine only the packet's destination component identifier or, when multiple egress interfaces are configured to reach a given destination component, the switch might examine additional protocol fields when making forwarding decisions or egress interface queuing delays.

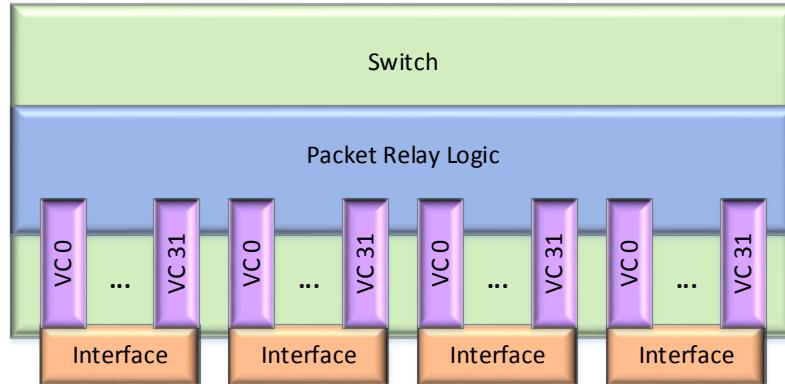


Figure 1-8: Discrete and Component-Integrated Switches

A switch may support two or more interfaces. Switch components can be serially connected as illustrated in *Serially Connected Components Using Integrated Switches*. Though not shown, the last component in the serial topology can be attached back to the Requester in a “horseshoe” arrangement or to an expansion switch to provide resiliency and additional performance (request and response packets can fully-utilize both paths).

Higher-interface-count switches (see *Switches* for detailed requirements) can be used to increase fan-out capacity and increase aggregate bandwidth, as well as provide multipath aggregation and resiliency (see *Example Integrated Switches for Fan-Out, Aggregate Bandwidth, and Resiliency* and *Example 16-Interface Switch for High Fan-Out and Capacity Solutions*).

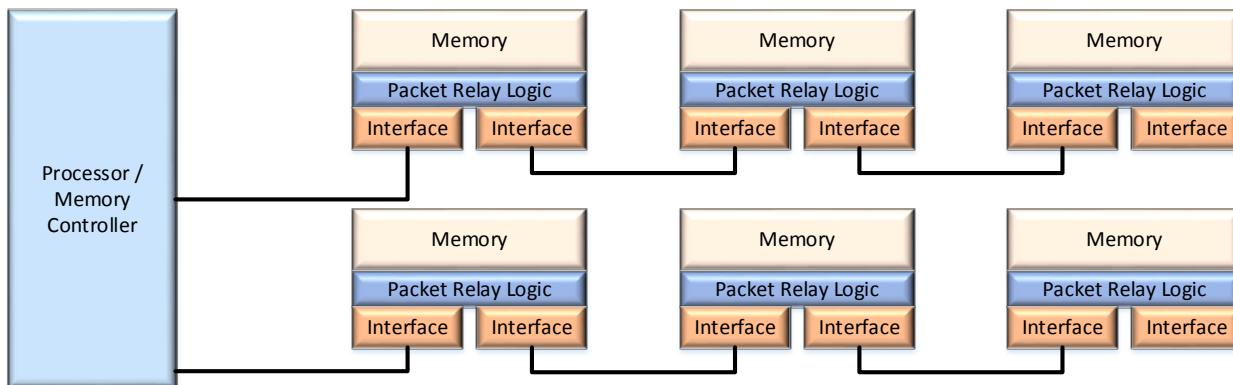


Figure 1-9: Serially Connected Components Using Integrated Switches

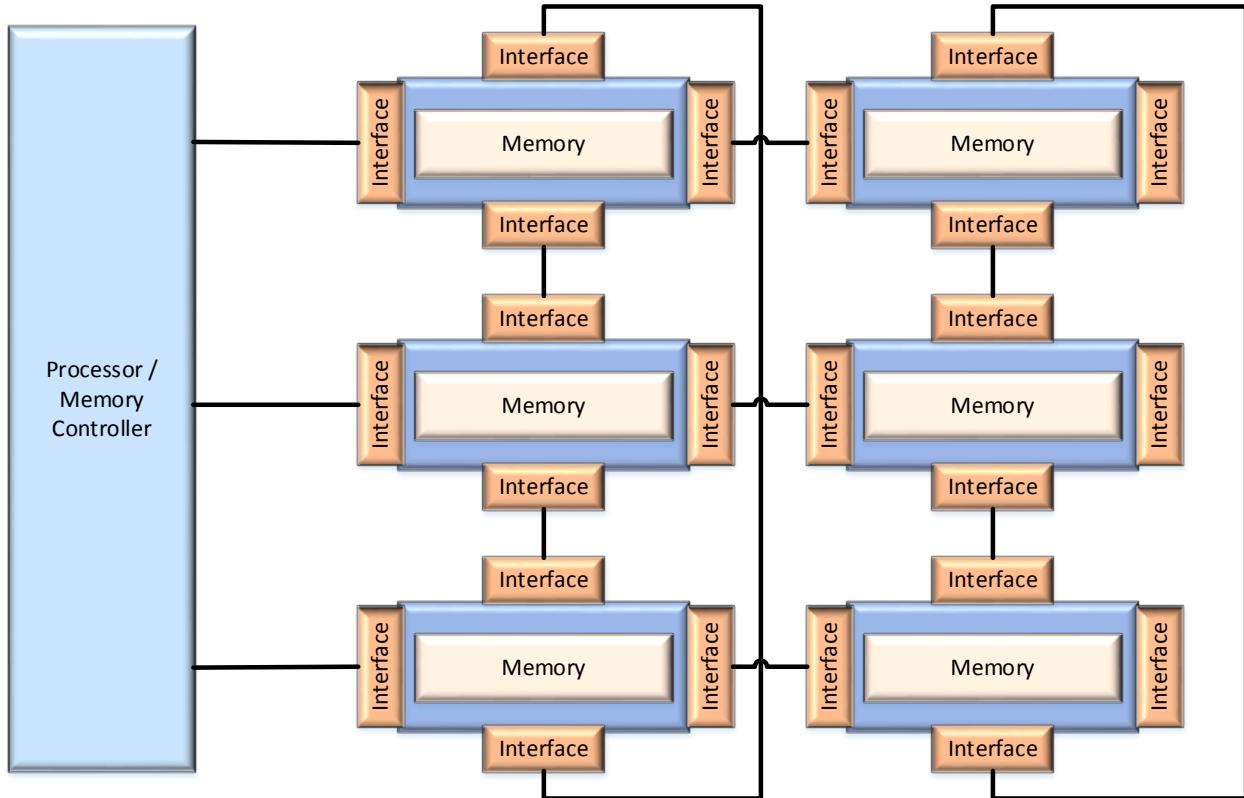


Figure 1-10: Example Integrated Switches for Fan-Out, Aggregate Bandwidth, and Resiliency

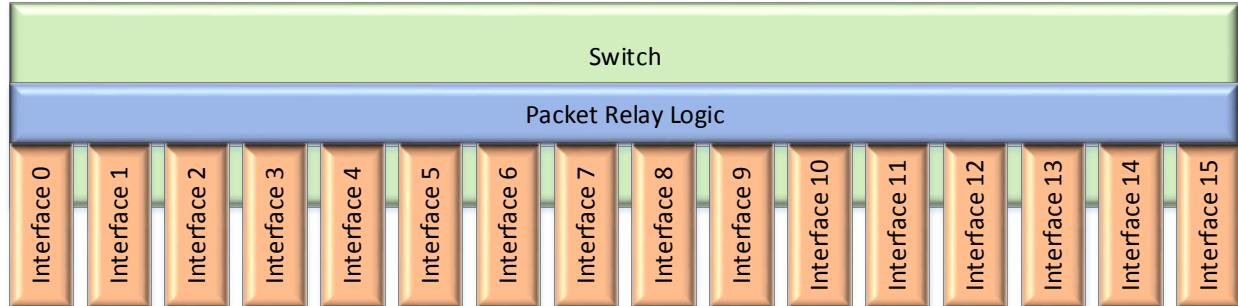


Figure 1-11: Example 16-Interface Switch for High Fan-Out and Capacity Solutions

5 Switches may be integrated into any component type including processors, memory, GPU, FPGA, I/O, etc. Integration facilitates peer-to-peer communication and reduces solution component count.

To simplify hardware, switches rely on software for configuration and management.

10 A component is configured with one or more numeric identifiers (referred to as a component identifier) that uniquely identifies the component within a given subnet. Within a single subnet, the packet's protocol header contains an identifier associated with the destination component. The switch uses this identifier to look up the egress interface to transmit the packet towards the destination component.

15 A component may support explicit multi-subnet communications. If supported, then the component is configured with a numeric identifier that uniquely identifies the subnet where a component is located (the combination of the subnet identifier and the component identifier creates a global component identifier that is unique within the multi-subnet topology). Components that support explicit multi-subnet communications use an extended protocol header that contains the destination component's

subnet identifier. A switch that supports multi-subnet packet relay uses the destination component's subnet identifier and component identifier to identify the egress interface to transmit the packet through. To connect subnets using switches requires each subnet to contain a subnet-local switch capable of multi-subnet packet relay, i.e., a single switch cannot be used to join multiple subnets.

## 5 1.7. Transparent Router Overview

A *Transparent Router (TR)* transparently joins two or more subnets together. For example, *TR Joining Multiple Subnets* illustrates a TR that connects four subnets [A-D].

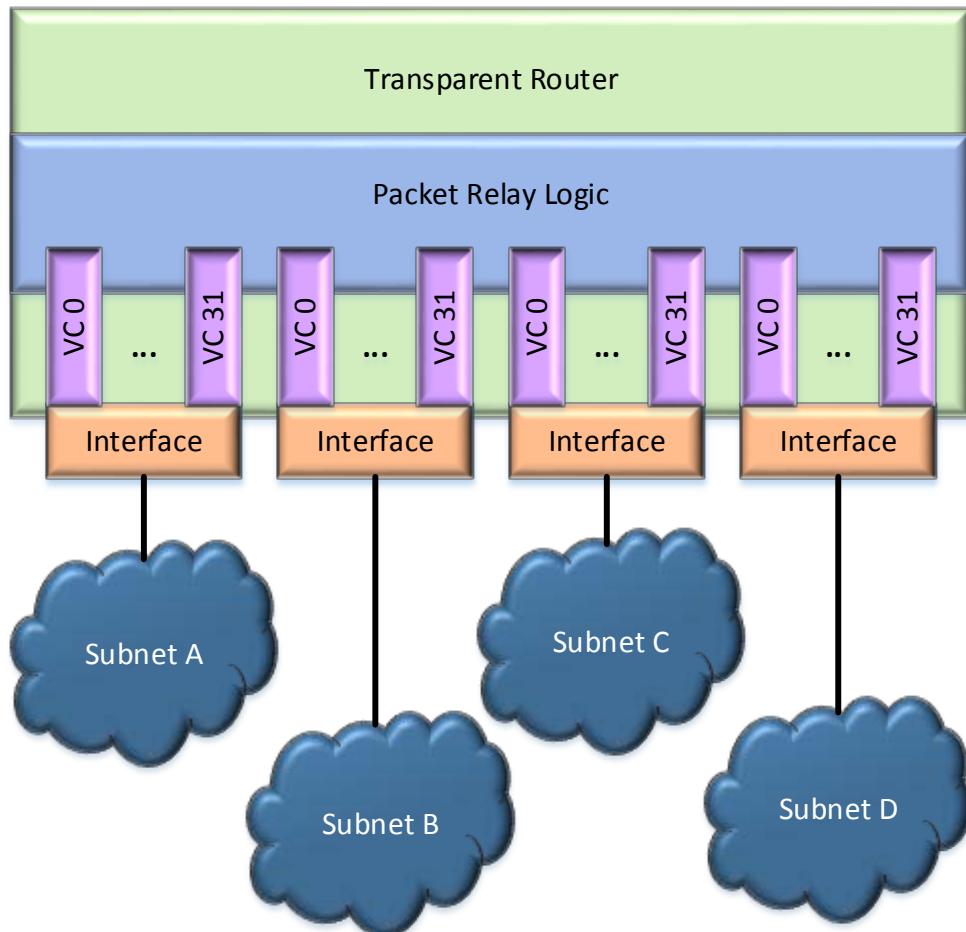


Figure 1-12: TR Joining Multiple Subnets

- 10 A TR relays packets between subnets unbeknownst to the source and destination components. To exchange packets, the TR transparently translates request and response packets between subnets without exposing each subnet's composition. This enables each subnet to be composed of any number and mix of component types. For example, *Example Transparent Router with a Mix of Component Types* illustrates two subnets. Subnet 1 contains a pair of processors and a TR that appears as a single Responder. Subnet 2 contains a mix of component types. To the subnet 2 components, the TR appears as a single Requester (on behalf of the two processors). Though the TR appears in each subnet as a distinct component, it is implemented as a single component with multiple personalities.
- 15

TR translation is address-based instead of identifier-based, and uses a memory management unit (MMU) to translate addresses that are mapped to a given Responder within the peer subnet.

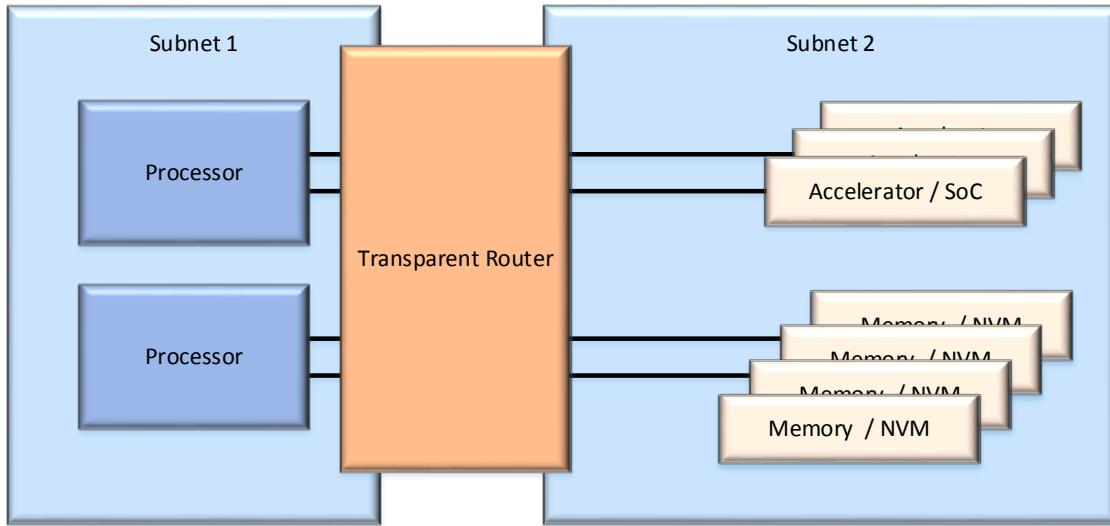


Figure 1-13: Example Transparent Router with a Mix of Component Types

5 A TR enables:

- Isolation—events and services that occur behind a TR are invisible to external processors, operating systems, application software, management, etc. For example, a subnet may provide memory resiliency or transparently migrate memory between tiers to optimize performance without processor or application coordination.
- Serviceability—components within a subnet can be mechanically serviced conceptually similar to how a storage controller enables underlying storage targets to be transparently serviced / replaced.
- Differentiated services—multiple services can be transparently instantiated or executed or performance accelerated on all or a subset of a multi-component subnet’s resources.
- Power optimization—components within a subnet can be transparently power optimized to meet environmental and dynamic workload needs.
- A TR may be integrated into any component type including processors, e.g., to take advantage of a TR-optimized MMU and reduce solution component count.
- A TR proxies request and response packets between subnets without revealing the actual source and destination components within each subnet.
- A TR may be used to transparently translate packets from one OpClass to another.
- A TR may be used to transparently perform memory interleaving on behalf of one or more Requesters.

## 1.8. Latency Domains

25 Communication latency is impacted by the physical topology. For example, a discrete or co-packaged point-to-point topology provides the lowest latency communication path between two components. In contrast, a switched-based topology incurs additional latency due to packet relay and queuing delays.

Communication latency is also impacted by the type of operations used to exchange data. For example, read and write request packets are the simplest, most efficient operations and deliver the lowest

latency. In contrast, advanced operations involving accelerators or large data movement may require additional processing time and may involve exchanging multiple request and response packets to complete.

5 A Requester is responsible for ensuring end-to-end Reliable Delivery (when applicable). A Requester maintains timers to identify request packet loss and initiate retransmission or error recovery. It is critical that Requester timers be configured to account for topology and operation-specific latencies to ensure good performance without causing unnecessary request packet retransmission. To minimize the number of timers, communicating components are conceptually organized into latency domains.

10 A latency domain defines the expected latency between communicating components when using specific request packets. For example, *Example Latency Domains* illustrates three example latency domains.

- Latency Domain 0 is a point-to-point topology between a single Requester and a single Responder, represents the simplest, lowest-latency topology.
- Latency Domain 1 is a single switch-hop topology between two Requesters. Domain 1 represents the simplest switch topology and incurs additional latency due to the packet relay and queuing delays within the directly-attached switch.
- Latency Domain 2 is a multi-switch topology containing multiple Requesters and Responders. Domain 2 represents the most diverse topology, and incurs additional latency based on how far the topology scales out.

20 If components exchange only simple request packets, then each topology may be treated as a single low-latency domain, albeit with different latencies that account for topology, packet relay, and queuing delays. If the components also exchange complex request packets that incur additional latency to complete, then each topology may be treated as two latency domains—one for simple, low-latency request packets and one for complex, non-low-latency request packets. In such cases, multiple latency domains may physically overlap one another (partial or complete). Management is responsible for determining the physical scope of a given latency domain, and for configuring the components to ensure correct operation.

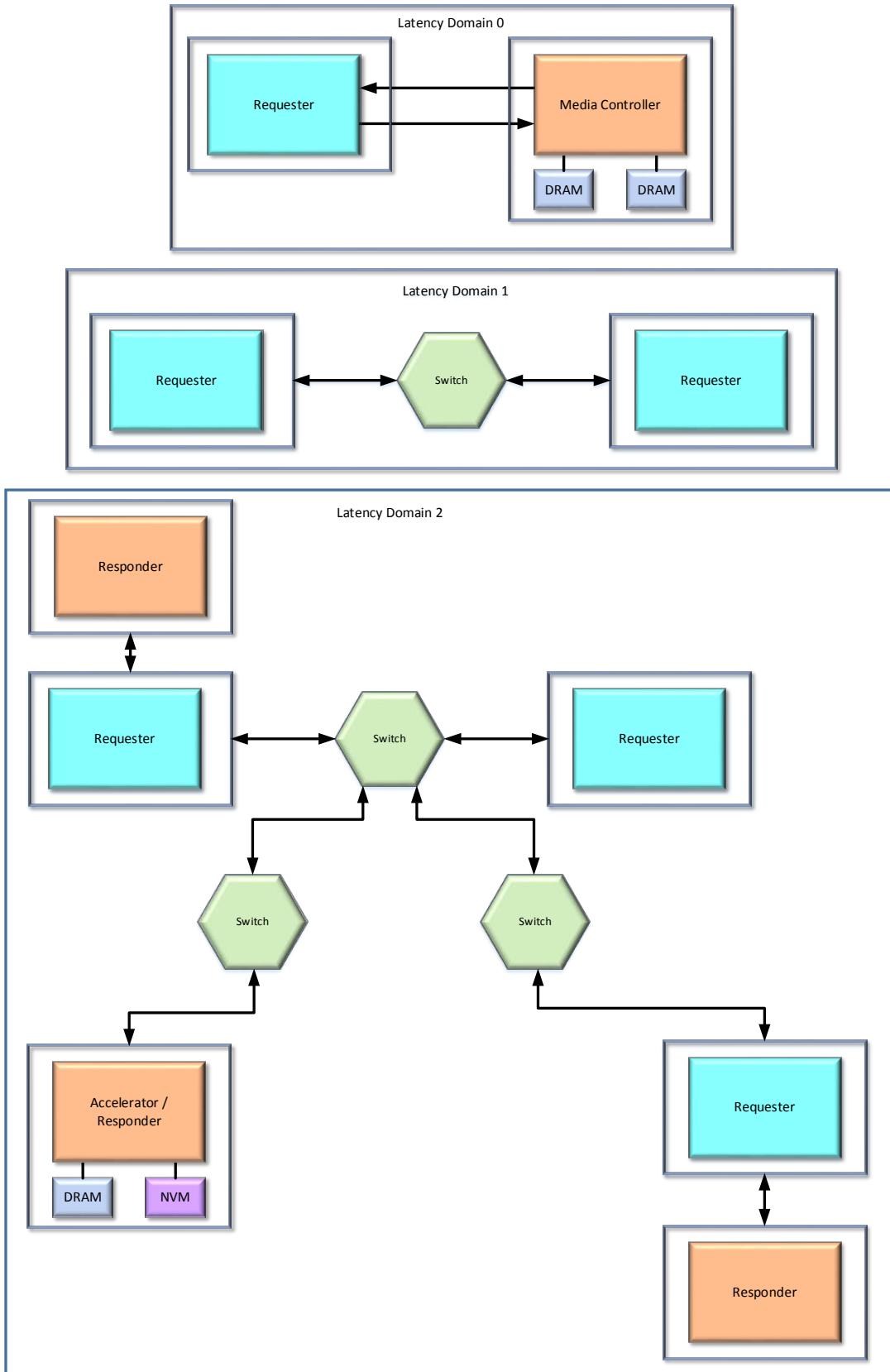


Figure 1-14: Example Latency Domains

## 1.9. Processor-Centric and Memory-Centric Solutions

In a processor-centric architecture, all components (integrated or discrete) access memory via the processor's memory controller. Such architectures require a suite of industry standard and / or proprietary interconnect technologies to provide memory access. The widespread adoption of solid-state technologies and memory-semantic communications between diverse components enable processor-centric architectures to consolidate interconnects to a single technology (*Processor-Centric Solution Architecture Evolution with Gen-Z*). Using Gen-Z simplifies designs, reduces power consumption, and improves performance by eliminating the need to provision and to translate operations between multiple interconnect technologies.

10

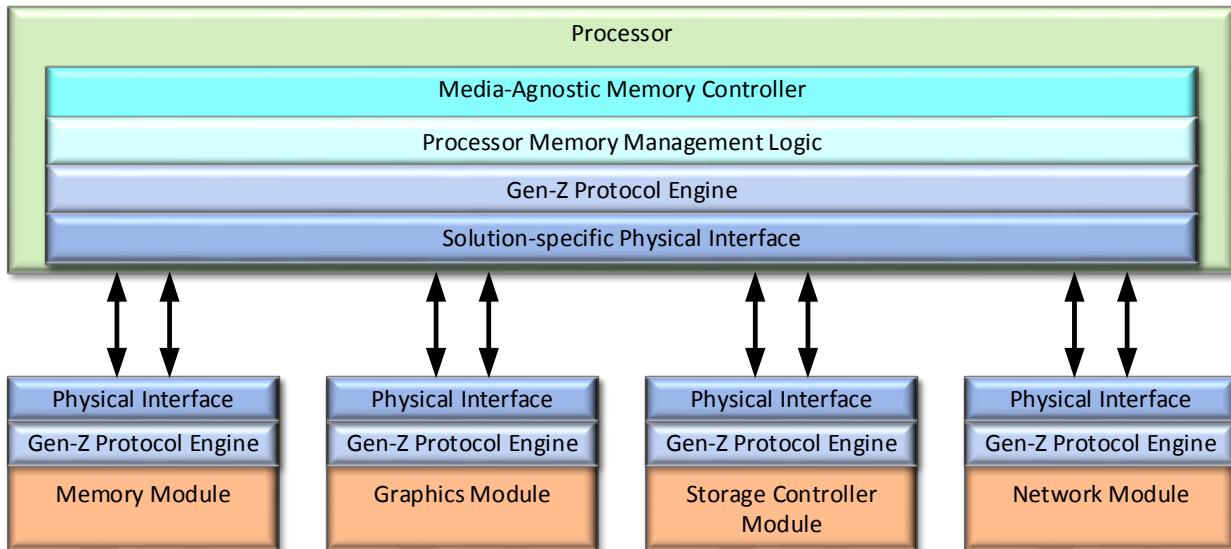


Figure 1-15: Processor-Centric Solution Architecture Evolution with Gen-Z

Gen-Z also enables memory-centric solution architectures. In a memory-centric architecture, components can bypass the processor memory controller and directly communicate with memory as well as other peer components. Direct communication reduces data movement, protocol translation complexity and latencies, power consumption, etc. For example, in *Memory-Centric Architecture*, communicating components such as a network controller and an accelerator can directly share memory without processor involvement. Similarly, a graphics controller can coherently or non-coherently communicate with an application via the processor on one interface and independently access memory on a second interface.

15

20

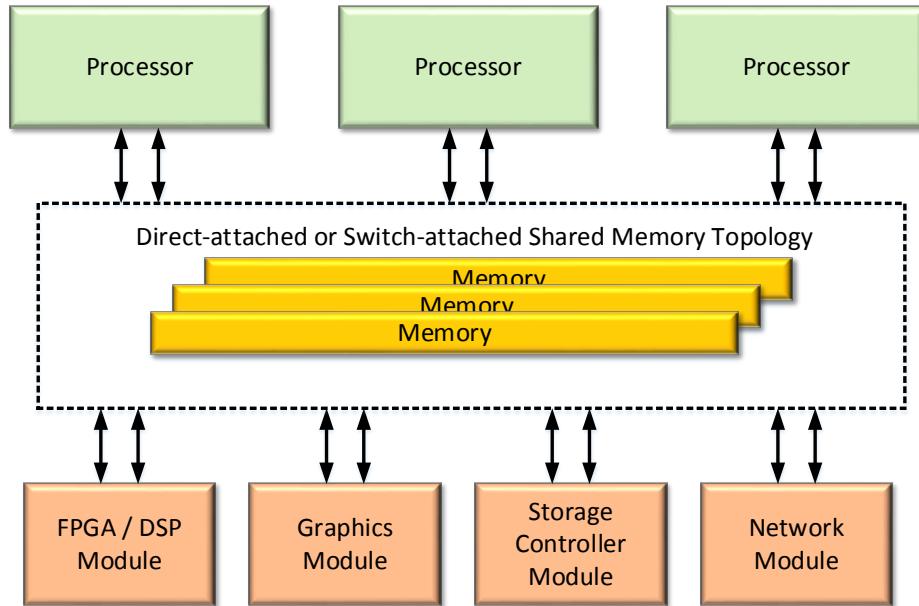


Figure 1-16: Memory-Centric Architecture

Finally, Gen-Z's packet-based protocol enables messaging solutions. *Example Memory-Semantic Messaging Topology* illustrates multiple Requesters attached to a collection of memory and switch components. Requester-Responders can communicate directly with one another using the same read and write operations used to access memory or vendor-defined requests. To minimize data movement, Requesters can use memory (discrete or co-located) as an intermediate staging area and exchange only pointers to locate a message.

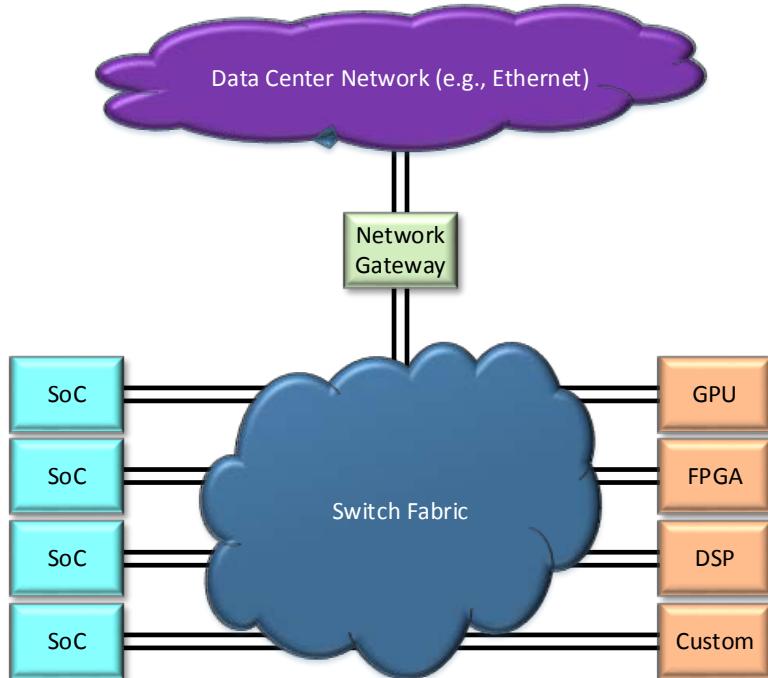


Figure 1-17: Example Memory-Semantic Messaging Topology

## 1.10. Physical Layer Independence

The physical interconnect is an independent layer in Gen-Z. Physical layer independence means all physical layer characteristics and operating semantics are isolated below the Gen-Z *Physical Layer Abstraction*. This provides multiple capabilities:

- Transparent support for existing as well as future physical layer media, e.g., copper, photonics, varying types of multi-die interposers, TSV (Through Silicon Via), etc.
- Transparent support for multiple physical layer generations, i.e., new physical layer media generations can be inserted into solutions without requiring modifications to higher-level functional blocks.
- Tailored support for solution-specific physical layers and associated media without impacting higher-level functional blocks.
- The ability to support automatic, real-time power optimizations based on workload needs, e.g., the ability to burst data across a link and then automatically power down a subset of lanes based on the transmit queue depth or to only power up a subset of the lanes to meet packet latency or throughput requirements.

## 1.11. Timer Formula Interpretation

Various timers are specified within this document. Although timers vary in purpose, they share a common formula template and interpretation to calculate the specified time duration:

$$\text{Timer} = \text{Base } \text{xs}, +\text{N\%} / -\text{M\%}$$

Where:

- Base = the minimum timer value, e.g., 1
- xs = unit of measured time, e.g., ns represents nanoseconds
- +N% = the maximum resolution variance above the calculated Timer value
- -M% = the minimum resolution variance below the calculated Timer value

The following example illustrates how the formula for Timer is interpreted:

$$\text{Timer} = 100 \text{ ns} *, +25\% / -0\%$$

The calculated Timer = 100 ns. Due to variances in designs, timer resolution may vary. In this example, the timer shall never expire in less than 100 nanoseconds (-0%) and never expire in more than 125 nanoseconds (+25%). Any fractions are rounded up to the nearest whole unit of time, e.g., nearest nanosecond.

The above example illustrates a timer where the action that is the result of a timeout never occurs in less time than the calculated timer value. For example, packets are never retransmitted in less than the nominal value and have no detrimental side effects if retransmitted slightly later than the calculated value. In contrast, the following example illustrates a timer where the action as a result of a timeout should not occur in more time than the calculated value.

$$\text{Timer} = 5000 \text{ ns}, +5\% / -25\%$$

The calculated Timer = 5000 nanoseconds. Due to variances in designs, the maximum timer is 5250 ns and the minimum timer is 3750 ns.

## 1.12. Document Organization

This document covers the core architecture and functionality. The document is organized to allow developers to focus on the subset of sections required to implement a given solution.

The sections are as follows:

- Section 1 *Core Architecture*: High-level overview of the architecture and associated features.
- Section 2 *Glossary*
- Section 3 *Core Functionality*: Blend of informative and normative material covering key architectural features such as memory semantic components, split controller model, virtual channels, etc.
- Section 4 *End-to-end OpClasses & OpCodes*: Covers end-to-end operation classes and associated operation codes. Operation classes are used to organize operations to simplify solution design and interoperability. Solutions can be constructed using the bare minimum operation classes and operation codes or can include any mix of operation classes and codes required to meet solution-specific needs.
- Section 5 *Base Formats for End-to-end Packets*: Covers general end-to-end packet formats and common protocol fields.
- Section 6 *Common End-to-end Operations*: Covers the end-to-end operations common to all solutions. These operations include Read Requests, Read Responses, Write Requests, Acknowledgements, and *In-band Management* packets used for configuration and management.
- Section 7 *Advanced End-to-end Operations*: Covers end-to-end operations that can be selectively used by solutions. These operations include Atomics, Interrupts, Buffer operations, Pattern operations, vendor-defined, etc.
- Section 8 *Configuration and Management*: Covers configuration and management including discovery and configuration, state machines, configuration management structures, etc. Solutions can be constructed using only a small number of mandatory structures to any mix of advanced structures depending upon the services and robustness of information they wish to communicate.
- Section 9 *Switches*: Covers switch requirements and semantics. Switches are optional and their use is driven by solution-specific needs.
- Section 10 *Transparent Router (TR)*: Covers transparent router requirements and semantics. Transparent routers are optional and their use is driven by solution-specific needs.
- Section 11 *Link Specification*: Covers link operation, communications, protocol, and error handling. In general, this section is applicable to all solutions.
- Section 12 *End-to-End Packet Reliability / Validation*: Covers end-to-end packet Reliable Delivery, data integrity, packet validation, etc. In general this section is applicable to all solutions.
- Section 13 *Physical Layer Abstraction*: Covers the physical layer abstraction (PLA). Physical layer abstraction enables the architecture to support multiple, solution-specific physical layers without permeating the rest of the architecture. This abstraction is applicable to all solutions.
- Section 14 *Multicast*: Covers multicast operation, requirements, and semantics. Multicast is optional functionality primarily applicable to switch-based topologies.
- Section 15 *Congestion and Packet Deadline*: Covers congestion management operation, requirements and semantics. Congestion management is optional functionality applicable to switch-based topologies.
- Section 16 *Security*: Covers all specified security requirements and semantics. Security is optional functionality applicable to any solution.

- Section 17 *Strong Ordering Domain (SOD)*: Covers end-to-end packet strong ordering packet exchange that uses sequence numbers. SODs are intended for less common solutions.
- Section 18 *Logical PCI Device (LPD)*: Covers the methods and configuration structure that enable any component to be seen by unmodified software as a conventional PCI device. This section covers how a component can scale up the number of LPD functions (e.g., to support direct hardware access in virtualization solutions) as well as how a component may be simultaneously shared by multiple processors with minimal added complexity.
- Section 19 *Logical PCI Hierarchy (LPH)*: Covers the methods and configuration structure that enable any component to be seen by unmodified software as presenting a small native PCIe hierarchy. This section covers how a component can serve as a logical Host Bridge for one or more hierarchies, as well as how a component supporting multiple hierarchies may present each hierarchy to a different host with minimal added complexity.
- Section 20 *Address Translation and Page Services*: Specifies the protocol and semantics required to translate addresses and manage on-demand paging.
- Appendices: Appendices contain material required by a subset of the developers. Unless explicitly stated otherwise by this specification, appendices are normative.

Solutions are constructed using this Core Specification in conjunction with mechanical and physical specifications:

- Gen-Z mechanical specifications cover instantiating mechanical, power, and thermal solutions. Existing industry mechanical specifications are leveraged to ease adoption and new specifications have been developed to advance the state of the art within the industry and maximize volume adoption of a universal mechanical solution across multiple interconnect technologies.
- *Gen-Z Physical Specification* covers mapping Gen-Z's physical layer abstraction as defined in the *Gen-Z Core Specification* to new or augmented industry standard physical layers.

### 1.13. Reference Documents

*Gen-Z Physical Specification*, <http://genzconsortium.org>

*Gen-Z Mechanical Form Factor Specifications*, <http://genzconsortium.org>

*PCI Express® Base Specification*, <http://www.pcisig.com>

*PCI Firmware Specification*, <http://www.pcisig.com>

*PCI Code and ID Assignment Specification*, <http://www.pcisig.com>

*Advanced Configuration and Power Interface (ACPI)*, <http://www.uefi.org>

*Management Component Transport Protocol (MCTP)*, <http://www.dmtf.org>

*ITU-T X.667*, <https://www.itu.int/>

### 1.14. Documentation Conventions

#### **Shall, Should, May, and Can**

This specification adheres to Section 13.1 of the IEEE Specifications Style Manual, which dictates use of the words 'shall', 'should', 'may', and 'can' in the development of documentation, as follows:

- The word *shall* is used to indicate mandatory requirements strictly to be followed in order to conform to the Specification and from which no deviation is permitted (*shall* equals *is required to*).
- The use of the word *must* is deprecated, and shall not be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.
- The use of the word *will* is deprecated, and shall not be used when stating mandatory requirements; *will* is only used in statements of fact.
- The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).
- The word *may* is used to indicate a course of action permissible within the limits of the Specification (*may* equals *is permitted*).
- The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

## Capitalization

Some terms are capitalized to distinguish their definition in the context of this document from their common English meaning. Words not capitalized and not defined within this document's Glossary have their common English meaning.

Bit fields are capitalized to improve legibility.

The first letter of a term composed of multiple words concatenated as follows (A\_B\_C) is capitalized.

Operation Code and Operation Class names are capitalized.

## Numbers and Number Bases

Unless explicitly stated otherwise by this specification, numerical values without qualifiers are decimal. This specification uses the following qualifiers:

- Hexadecimal numbers are written with a '0x' prefix followed by a mix of digits 0 through 9 and / or upper case English letters A through F, e.g., 0xFFFF or 0x80. Hexadecimal numbers larger than four digits are represented with a space dividing each group of four digits, as in 0x1E FFFF EFFF.
- Binary numbers are written with a lower case 'b' suffix, e.g., 1001b and 10b. Binary numbers larger than four digits are written with a space dividing each group of four digits, as in 1000 0101 0010b.

A dash between two numbers represents a range of values, e.g., 0-7 represents zero to seven inclusive.

A colon between two numbers represents a range of bits, e.g., Bits 7:0 represents a binary value of bits seven to zero inclusive.

## Standard Units of Time

Unit	Description
ms	Millisecond— $10^{-3}$ seconds

$\mu\text{s}$	Microsecond— $10^{-6}$ seconds
$\text{ns}$	Nanosecond— $10^{-9}$ seconds
$\text{ps}$	Picosecond— $10^{-12}$ seconds
$\text{fs}$	Femtosecond— $10^{-15}$ seconds

## Reserved

The following applies to the term ‘Reserved’:

- The contents, state, or information are not specified at this time.
- Any field, feature, capability, etc. marked ‘Reserved’ is subject to change.
- Components shall not use any field, feature, capability, etc. that is marked ‘Reserved’.
- All RsvdP control structure fields are reserved for future read-write implementations. Field is read-only and shall return zero when read. Software shall preserve the value read for subsequent writes.
- All RsvdZ control structure fields are reserved for future RW1C implementations. Field is read-only and shall return zero when read. Software shall use 0x0 for writes.
- All Reserved control structure fields shall return 0x0 when read. Software written to this specific specification shall not write to Reserved control fields. Hardware shall ignore all writes to Reserved control structure fields. These fields are reserved for future implementations.
- Reserved encodings of any control structure, packet fields, etc. shall not be used.
- Any implementation dependence on a ‘Reserved’ field value, or encoding will result in a non-compliant implementation. The functionality of such an implementation cannot be guaranteed in this or any future revision of this specification.
- A Reserved protocol field shall be transmitted as zero and ignored upon receipt.

## Read-only Control Structure Fields

Hardware shall ignore writes to read-only control structure fields.

## Write-only Control Structure Fields

If a write-only Control Space field is read, then the read shall return zero, and shall not be treated as an error.

## Developer Notes

Developer Notes are informative. They are included for clarification and illustration only. These notes are delineated by: **Developer Note:** *Text*

## State Machine Conventions

State machines describe the behavior of a component, an interface, a link, or the physical layer. A state machine does not imply the internal design or implementation. State machines use the following conventions:

- Each state machine is represented by a rectangular box.
  - The top section of the box contains the state name.
  - The bottom section of the box contains the actions which occur in the state.
- Transition arrows indicate state transitions which are made when the expression next to the arrow is satisfied.
- A transition arrow which does not originate in a state indicates a global transition. Such a transition will occur regardless of the current state.
- If no exit condition for state is met, then the state machine remains in the current state.
- A logical “OR” is represented by a “+”.
- A logical “AND” is represented by a “\*”.
- If a conflict occurs, state machine figures take precedence over descriptive text.

### Flow Chart Conventions

Flow charts illustrate packet processing, validation, etc. Flow charts use the following conventions:

- A ‘Go to’ represents a jump to another flow chart to complete processing.
- A ‘Perform XXX ( )’ represents a branch to a flow chart that performs additional processing before returning to continue processing in the current flow chart. Conceptually, this is similar to invoking a sub-function.

### Table Entry Conventions

Unless explicitly stated within this specification, common table entries are defined as follows.

Table Entry	Description
M	Mandatory—the field or associated semantic is treated as a ‘shall’.
O	Optional—the field or associated semantic is treated as a ‘shall if implemented’.
MC	Mandatory Conditional—if the described condition is true, then the field or associated semantic is treated as a ‘shall’.

### Endian

This specification uses little endian bit and byte order.

## 2. Glossary

Term	Definition
$\oplus$	Symbol for concatenate
<b>Access Key (AKEY)</b>	An identifier within the protocol header used to isolate packet exchanges to only those components that share a common Access Key.
<b>ACK</b>	A response packet that indicates the corresponding request packet was successfully validated and executed.
<b>acknowledgment</b>	Throughout this specification, an acknowledgment refers to either a specific response corresponding to a specific request or to a <i>Standalone Acknowledgment</i> .
<b>address</b>	A resource location communicated as a fixed-length sequence of bits treated as an unsigned integer.
<b>aggregation</b>	Aggregation refers to grouping multiple interfaces into a single active interface (this is the antithesis of interface bifurcation where a single link is split into multiple distinct interfaces).
<b>BER (Bit Error Ratio)</b>	The number bits with errors divided by the total number of bits that have been transmitted, received or processed over a given time period. The ratio is expressed as 10 to the negative power, e.g., $10^{-15}$ .
<b>BIST</b>	Built-in Self-Test
<b>Caching Agent</b>	A Caching Agent is logic within a component that initiates Gen-Z coherency request packets to an address space. A Caching Agent can maintain copies of data in its own cache structure, and it can provide copies to other Caching Agents.

<b>CID (Component Identifier)</b>	A value that uniquely identifies a component within a single subnet.
<b>component</b>	A component represents a collection of one or more resources and accompanying logic connected by Gen-Z links that enables it to exchange or relay end-to-end packets with other components.
<b>Control Space</b>	An addressable resource containing management structures. Control Space may also contain component-specific resources outside of this specification's scope, e.g., accelerator executables, component-specific resources, etc. Control Space may be accessed by in-band packets or an <i>Out-of-band Management</i> interface.
<b>Data Space</b>	An addressable resource upon which operations such as read and write request packets are executed. One or more resource types may be provisioned depending upon the component type, e.g., memory media.
<b>DCID</b>	Destination Component Identifier
<b>DGCID</b>	Destination Global Component Identifier
<b>DIMM</b>	Dual Inline Memory Module
<b>DR (Directed Relay)</b>	Directed Relay is used by a switch to relay a Control Space request packet to a specific switch egress interface.
<b>DRAM</b>	Dynamic Random Access Memory
<b>DSID</b>	Destination Subnet Identifier

<b>ECRC (End-to-end CRC)</b>	An N-bit cyclic redundancy check used to validate the integrity of an end-to-end packet.
<b>ECRC Error</b>	The value in the received packet's ECRC field does not match the ECRC dynamically generated by the receiving interface.
<b>End-to-end</b>	Operations or events that occur between a Requester component and a Responder component irrespective of the number of intervening hops.
<b>EXE (Execution Error)</b>	<p>An uncorrectable error designation that indicates that though a packet was successfully validated, an unrecoverable error was detected while executing the operation indicated by the packet. For example, the target media failed, an internal processing element failed, a thermal or power event prevented execution, etc.</p> <p>Execution errors indicate whether they are fatal (the component is non-operational in whole or part), or non-fatal (the component is operational but cannot successfully execute operation associated with the packet).</p>
<b>Flit</b>	Flow-control digit—unit of flow-control. A Flit may require one or more physical layer clock cycles to transfer across a link.
<b>Flow-Control Transmission Timer (FCTT)</b>	A timer used to schedule the transmission of link-local flow-control packets.
<b>FPS (Forward Progress Screen)</b>	Mechanisms used to ensure that request and response packets move toward successful receipt and execution.
<b>GiB (Gibibyte)</b>	GiB represents $2^{30}$ bytes.
<b>GCID (Global Component Identifier)</b>	A value that uniquely identifies a component within a multi-subnet topology.
<b>Global DCID (Destination Component Identifier)</b>	A Global DCID refers to the destination component in a multi-subnet topology that uses non-transparent routers to exchange packets between subnets.

	<p>In unicast packets, a Global DCID is used by a destination component to validate correct packet delivery.</p> <p>In a non-transparent router, a Global DCID is used to perform packet relay between subnets.</p>
<b>Global Multicast Group Identifier (GMGID)</b>	<p>An identifier associated with a multi-subnet multicast group.</p>
<b>Global SCID (Source Component Identifier)</b>	<p>A Global SCID refers to the component that first injected the end-to-end packet into a multi-subnet topology that uses non-transparent routers to exchange packets between subnets.</p>
<b>HMAC</b>	<p>Hash-based Message Authentication Code—see <i>Security</i></p>
<b>Home Agent</b>	<p>A Home Agent is logic within a component that maintains coherency for an address space, e.g., providing data and ownership responses to manage Gen-Z coherency request packets and conflicts between Caching Agents. A Home Agent manages a portion of the coherent address space.</p>
<b>HwInit (Hardware Initialized)</b>	<p>An attribute where the field is hardware initialized, and subsequently treated as read-only. The hardware-initialized field contains a fixed, read-only value that can be changed only by a <i>Full Component Reset</i> or <i>Full Interface Reset</i> (depending upon the field's location). A HwInit field is valid when <i>Core C-Status HwInit Valid</i> = 1b.</p>
<b>idempotent</b>	<p>Idempotent request packets are request packets that may be retransmitted and executed without changing the execution result. For example, if a Read Response packet was lost, a Requester could retransmit the Read request packet and expect the new Read Response packet to return an equally-valid payload as the lost Read Response packet (if no intervening update operations occurred, then the same payload).</p> <p>In contrast, an Atomic Add operation is not idempotent as re-executing the add portion of the request packet would change the result.</p>
<b>interface</b>	<p>The physical point at which a single Gen-Z link attaches to a component.</p>

internal error	An error internal to a component error whose root cause is not based on a packet, configuration setting, or any other action as specified in this document, i.e., any error not defined by this specification.
Interface Group	A set of interfaces whose configuration and / or operation can be intertwined, e.g., a set of interfaces that can be aggregated into a single interface.
KiB (Kibibyte)	KiB represents $2^{10}$ bytes.
lane	A point-to-point signal channel that connects one transmitter and one receiver. Transmitters send electrical or photonic signals to receivers.  In SerDes-based implementations, a lane represents a differential pair.
Lightweight Notification (LN)	A protocol that supports notifications to components via a hardware mechanism when registered LN Blocks of interest are updated.
link	A set of one or more lanes used to exchange packets between two interfaces.
link-local	Packets or events associated with the operation of a single link.
Link-level Reliability Timer (LLRT)	A timer used when scheduling LLR acknowledgments
LN Requester (LNREQ)	An LNREQ is a Requester-Responder component that transmits LN request packets to register LN Blocks, and receives LN Notification packets when registered LN Blocks are updated.
LN Responder (LNRSP)	An LNRSP is a Requester-Responder component that executes LN Block registration request packets, and transmits LN Notifications when registered LN Blocks are updated.
Logical PCI Device (LPD)	A portion of a component that supports conventional PCI functional semantics for I/O device discovery, driver association, and basic configuration with I/O infrastructure.

<b>Logical PCI Hierarchy (LPH)</b>	A portion of a component that presents a PCIe hierarchy with native PCIe devices for I/O device discovery, driver association, and basic configuration with I/O infrastructure.
<b>LSB</b>	Least Significant Bit is the bit position in a binary number having the lowest value. This is also known as the low-order bit.
<b>Max_Packet_Payload</b>	Indicates the maximum number of bytes (payload) that a packet can contain.
<b>MCAP</b>	The Multicast Access and Peer attribute table is used to generate and validate multicast end-to-end packets in a directly-attached subnet topology.
<b>MCTP</b>	Management Component Transport Protocol is specified by the DMTF. See <a href="http://www.dmtf.org">www.dmtf.org</a>
<b>MGID (Multicast Group Identifier)</b>	<p>A 12-bit value that identifies a multicast group within a subnet.</p> <p>An MGID is unique per subnet.</p>
<b>MiB (Mebibyte)</b>	MiB represents $2^{20}$ bytes.
<b>MP (Malformed Packet)</b>	An uncorrectable error designation that indicates a packet was not properly formed per the packet formation rules defined within this specification.
<b>MRL (Mechanical Retention Latch)</b>	A physical mechanism used to hold a physical mechanical module in place.
<b>MSAP</b>	The Multi-Subnet Access and Peer attribute is used to generate and validate unicast end-to-end packets in a multi-subnet topology.
<b>MSB</b>	Most Significant Bit is the bit position in a binary number having the greatest value. This is also known as the high-order bit.

MSMCAP	The Multi-Subnet Multicast Access and Peer attribute table is used to generate and validate multicast end-to-end packets in a multi-subnet topology.
multicast	<p>A facility by which a packet that targets a single MGID is delivered to multiple components.</p> <p>Multicast is referred to as unreliable multicast if multicast request packets are not acknowledged by the destination components.</p> <p>Multicast is referred to as reliable multicast if multicast request packets are acknowledged by the destination components.</p>
multicast group	A collection of interfaces that receive multicast packets sent to a single MGID.
multi-subnet packet	An explicit OpClass packet that contains an <i>Explicit OpClass Multi-subnet Field</i> .
multi-subnet packet relay	Switch-based packet relay of a multi-subnet packet.
multi-subnet switch	A switch capable of relaying multi-subnet packets.
NAI	Non-operational, aggregated interface—an interface that has been combined with one or more interfaces that have been configured to operate as a single interface. Though its resources (e.g., transmit and receive lanes, flow-control buffers, etc.) can be used by the SAI, the NAI cannot be used for any purpose.
NAK	A response packet that indicates the corresponding request packet failed packet validation or execution. The response packet indicates the failure Reason.
NIR	Non-idempotent request

<b>Non-transient condition</b>	A condition that impacts component or application operation, e.g., unsafe thermal operating conditions, uncorrectable media errors, etc.
<b>non-transient error</b>	An error that cannot be transparently recovered by Requester packet retransmission. Example non-transient errors include UR, MP, and EXE errors.
<b>Null</b>	A pointer field that is set to 0x0.
<b>nonce</b>	An arbitrary number that used only once
<b>NVM</b>	Non-volatile memory—memory whose state continues to exist after power is removed.
<b>operation</b>	An exchange of one or more request packets to provide a given service. Operations may be normative or vendor-defined to enable customized services.
<b>Outstanding Request Packet</b>	<p>A request packet is <i>outstanding</i> whenever a Requester has issued it, and is awaiting the receipt of a required response packet.</p> <p>Request packets for which no response packet is expected, e.g., an unreliable request packet or an Unsolicited Event (UE) Packet are never outstanding.</p> <p>Request packets requiring multiple response packets are <i>outstanding</i> until all required response packets have been received by the Requester. For example, <i>Non-Deterministic Request Execution</i> packets, LDM Reads, <i>Reliable Multicast</i>, etc.</p>
<b>P2P</b>	Point-to-point
<b>P2P-Core</b>	Unicast packets optimized for point-to-point communications.

<b>packet</b>	Smallest unit of information transmitted between two components.
<b>packet relay</b>	If a unicast end-to-end packet, then packet relay is the act of forwarding a packet from an ingress interface to an egress interface within a single component.  If a multicast end-to-end packet, then packet relay is the act of forwarding a packet from an ingress interface to one or more egress interfaces within a single component.
<b>PASID</b>	Process Address Space Identifier—a 20-bit value used to identify an application process or thread.
<b>payload</b>	The data, not including protocol fields, carried in a single end-to-end packet.
<b>PCI Express® (PCIe® )</b>	PCI Express is an I/O interconnect and protocol specified by the PCI-SIG. <a href="http://www.pcisig.com">http://www.pcisig.com</a>
<b>PCO</b>	PCIe Compatible Ordering
<b>PCRC (Prelude CRC)</b>	CRC used to protect a consistent set of bits within every packet format.
<b>PECAM Requester</b>	A Requester that supports PECAM logic.
<b>persistent</b>	An attribute of memory where the data state continues to exist after power is removed.
<b>PLA</b>	Physical Layer Abstraction
<b>Poison</b>	Indicates the underlying data at a specified address has suffered an uncorrectable error.

<b>POST (Power-On Self-Test)</b>	Hardware component diagnostic service that is executed prior to computer boot up. In general, this service covers power supplies, memory, system interconnects, etc.
<b>Power-cycle</b>	Power loss followed by Power-up
<b>Power-up</b>	Power being applied to a component
<b>PTD (Precision Time Domain)</b>	A set of components synchronized to a common Master Time.
<b>PTE</b>	Page Table Entry
<b>PRG (Page Request Group)</b>	A set of correlated page requests initiated by a Requester Context Identifier. Correlated page requests are identify by a PRG Index.
<b>PRI (Page Request Interface)</b>	Logic associated with a Requester context identifier that initiates page requests and processes page response packets. See <i>Address Translation and Page Services</i> for additional details.
<b>receiver</b>	Refers to the interface consuming a packet (link-local or end-to-end packet).
<b>Region Key (R-Key)</b>	An opaque object associated with a page. An R-Key is used to validate page access authorization, e.g., to prevent erroneous software or hardware access, and as such, should not be considered a security mechanism.  An R-Key may also be used to accelerate address translation. R-Keys are only present on select packet formats.
<b>Reliable Delivery</b>	Uncorrupted, exactly-once packet delivery.

<b>request packet</b>	A packet that is transmitted by a component to initiate or continue an interaction with one or more other components
<b>Requester</b>	<p>A component that generates request packets, other than a <i>CTL-UE Packet</i> or an <i>Unsolicited Event (UE) Packet</i>, and receives and executes any corresponding response packets.</p> <p>Throughout this document, wherever 'Requester' is used, the specification equally applies to a Requester-Responder when acting as a Requester.</p>
<b>Responder</b>	<p>A component that receives and executes request packets and generates any corresponding response packets.</p> <p>Throughout this document, wherever 'Responder' is used, the specification equally applies to a Requester-Responder when acting as a Responder.</p>
<b>response packet</b>	A packet that is transmitted by a component to a second component in direct response to the receipt of a request packet from the second component
<b>RNR (Responder Not Ready)</b>	An acknowledgment packet with a Reason value that indicates the Requester was unable to execute the request packet and that it can retransmit the request packet after the indicated period of time has elapsed.
<b>RO (Read-only)</b>	<p>An attribute where a field is read-only, i.e., it cannot be modified.</p> <p>If the feature associated with the field is unsupported, then the field shall be hardwired to 0b.</p>
<b>ROS (Read-only Sticky)</b>	<p>An attribute where a field is RO with sticky semantics.</p> <p>If the feature associated with the field is unsupported, then the field shall be hardwired to 0b.</p>
<b>RW (Read-Write)</b>	<p>An attribute where a field is read-write, i.e., the field can be modified.</p> <p>If the feature associated with the field is unsupported, then the field shall be hardwired to 0b.</p>
<b>RW1C (Read-Write 1 to Clear)</b>	<p>A bit field attribute where a read returns the bit's value, and writing a value of 1b to the bit clears its value.</p> <p>Unless explicitly stated otherwise by this specification, writing a value of 0b shall have no effect on the field.</p> <p>If the feature associated with the field is unsupported, then the field shall be hardwired to 0b.</p>

<b>RW1CS (Read-Write 1 to Clear, Sticky)</b>	A bit field attribute of RW1C with sticky semantics. If the feature associated with the field is unsupported, then the field shall be hardwired to 0b.
<b>RWS (Read-Write Sticky)</b>	A bit field attribute of RW with Sticky semantics. If the feature associated with the field is unsupported, then the field shall be hardwired to 0b.
<b>SAI</b>	Single, aggregated interface—the sole operational interface among the set of interfaces that have configured to operate as a single interface.
<b>scheduled</b>	Within this specification's context, refers to the moment when a packet is placed on an interface transmission queue. There is no guarantee that the moment a packet is placed on the transmission queue is the same as the moment when the packet transmission on the link is begun.
<b>semantically dependent</b>	One or more request packets whose execution is determined by the execution of prior request packets. For example, the result of a read request packet may depend upon the success of a prior write request packet to the same location.
<b>SCID</b>	Source Component Identifier
<b>SGCID</b>	Source Global Component Identifier
<b>SID</b>	A value that uniquely identifies a subnet within a multi-subnet topology that is explicitly addressed within a packet.
<b>SSAP</b>	The Single Subnet Access and Peer attribute table is used to generate and validate unicast end-to-end packets in a directly-attached subnet topology.
<b>SSID</b>	Source Subnet Identifier
<b>spatial locality</b>	A point-to-point optimized packet attribute that indicates one or more subsequent packets will access the same address range.

	<p>A component uses this attribute to optimize the underlying resource access to minimize latency, data movement, power consumption, etc.</p>
<b>sticky</b>	<p>Bit or field attribute where the value is preserved across a <i>Warm Component Reset</i> or <i>Warm Interface Reset</i>.</p>
<b>stomp</b>	<p>The act of replacing an end-to-end packet's ECRC field content with a dynamically-generated stomped ECRC value.</p>
<b>stomped ECRC</b>	<p>A stomped ECRC is used to indicate to the destination component or intermediate components that a packet data integrity error was previously detected.</p> <p>If an intervening or destination component detects a stomped ECRC, then it handles the packet differently than a packet with a corrupted ECRC field.</p>
<b>subnet</b>	<p>A collection of components that share a single component identifier space. A subnet can be trivial (e.g., a single processor and a single memory component), or it can be quite extensive and diverse (e.g., one or more processors, multiple memory components, and a number of diverse component types).</p> <p>A single physical component may include connections to more than one subnet. Unless the component supports packet relay between these subnets, the subnets are isolated from one another (e.g., a series of point-to-point subnets).</p> <p>A subnet may have a numeric label that explicitly identifies the subnet (referred to as a subnet identifier). This identifier is used to construct a Global CID. Such subnets are connected using switches that support multi-subnet packet relay.</p> <p>A subnet may be transparently accessed without requiring the use of a numeric label. Such subnets are connected using <i>Transparent Router (TR)</i>.</p>
<b>switch</b>	<p>A component or component-integrated functionality that performs packet relay between component interfaces.</p>

tag	An identifier used to correlate request packets with response packets.
TiB (Tebibyte)	TiB represents $2^{40}$ bytes.
transient condition	A temporary condition that could impact component or application operation, e.g., temporary resource shortage, thermal spikes, correctable media errors, etc.
transient error	An error that is transparently recovered through packet retransmission. Example transient errors include CRC errors, and physical layer-specific errors (e.g., symbol encoding errors).
transmitter	Refers to the interface injecting a packet (link-local or end-to-end packet).
transparent router	A component or component-integrated functionality that performs packet relay between two subnets unbeknownst to the source and destination components.
uninitialized state	State where all resources and / or Control Space structures are returned to their initial power-up contents. With the exception of hardware-initialized fields, all fields and volatile media are cleared (zeroed).
Unit Interval (UI)	UI is the time interval taken to transmit one bit (logical 0 or 1) on a single lane of a link. Although UI is independent of link width, UI duration changes with link speed changes. For multilevel signaling, this is the time taken to send the minimum possible number of bits divided by the minimum number of bits sent (e.g. if two bits are sent over time T, then the UI is $T / 2$ ).
unsolicited event packet	A request packet that provides an asynchronous indication of a change in operational state. For example, a component failure or new component detection.
UP (Unexpected Packet)	An uncorrectable error designation that indicates a component received a packet that was not expected, e.g., a packet that was incorrectly relayed to the component

<b>UR (Unsupported Request or Response)</b>	An uncorrectable error designation that indicates that although the packet was delivered to the correct destination and passes data integrity validation, one or more fields within the packet indicates an unsupported capability.
<b>UUID</b>	Universally Unique Identifier as specified in ITU-T X.667. UUID are used to identify components, identify vendor-specific capabilities, etc.
<b>VC<sub>k</sub> MIN</b>	The minimum number of explicit flow-control credits to receive at least one packet of the largest supported packet type (protocol plus associated payload) on VC <sub>k</sub> .
<b>VET</b>	Variable request execution timer—a timer used to ensure reliability of request packets whose individual execution times will vary as a function of the amount of work performed, e.g., as a function of the amount of data moved
<b>Vendor Defined Operation (VDO)</b>	An operation whose details are outside of this specification's scope, e.g., a designer (vendor) creates a custom operation for a particular solution.
<b>VC (Virtual Channel)</b>	<p>A mechanism to create multiple logical packet streams that are time-multiplexed across a single physical link.</p> <p>Each VC represents an independent explicit flow control domain.</p> <p>Components that support multiple VCs use one or more distinct VCs for request packets, and one or more distinct VCs for response packets. This delineation is used to prevent deadlocks due to request-response dependencies.</p>
<b>WO (Write-only)</b>	<p>An attribute where a field is write-only. Reads of a write-only field shall return 0x0.</p> <p>If the feature associated with the field is unsupported, then the field shall be hardwired to 0b.</p>

### 3. Core Functionality

The architectural layers within the specification's scope are:

- Physical Layer Abstraction Interface—the architecture defines a *Physical Layer Abstraction* to isolate physical layer-specific impacts.
- Packet—smallest unit of information transmitted between two components. A packet is self-describing, i.e., it contains multiple protocol fields that dictate how it is to be processed as well as to provide data integrity. The architecture supports multiple packet types generally categorized as link-local and end-to-end.

Components, e.g., memory controllers and memory components, implement these layers in addition to their chosen physical layer to facilitate interoperable communications.

This section covers the core functionality and operational principles and semantics associated with these layers. Where beneficial, this chapter illustrates how different component types may use this architecture to create optimal solutions.

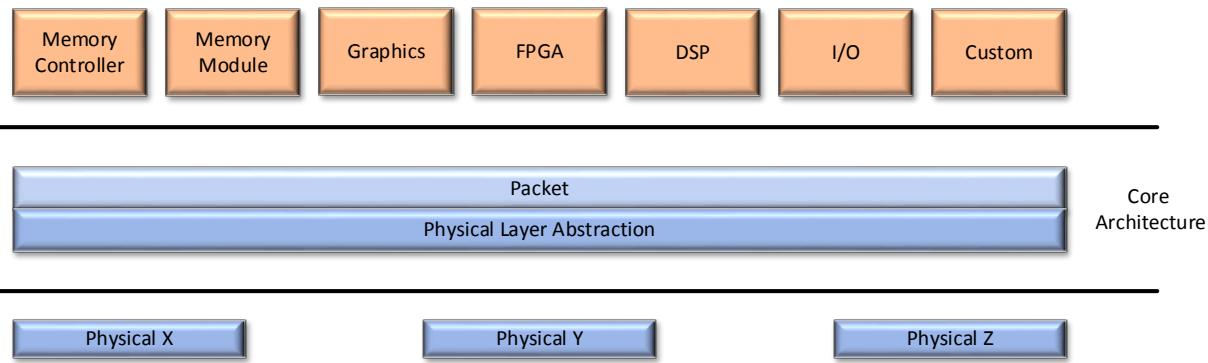


Figure 3-1: Architectural Layers

#### 3.1. Communications

Gen-Z uses a non-blocking, asynchronous protocol to exchange self-describing packets between two or more components. This provides numerous capabilities:

- Optimized, power-proportional communications (real-time, variable service time). Physical layer signaling rates and link widths may be dynamically tuned to meet real-time workload needs. Being asynchronous, the protocol tolerates variable response latency due to dynamic operating conditions.
- Request and response packet pipelining to increase bandwidth and reduce effective per-packet latency. Components can support whatever number of in-flight packets are required to meet a given solution's performance requirements—see *Example Pipelined Request and Response Packets*.
- Transparent support of multiple memory media types including DRAM, Flash, STTRAM, ReRAM, MRAM, etc.
- Transparent support of diverse topologies—point-to-point, meshed, switch, and multi-subnet—with associated variable response times as a function of topology and load.
- Transparent recovery of transient errors and events.
- Improved resiliency including multipath and transparent end-to-end recovery.

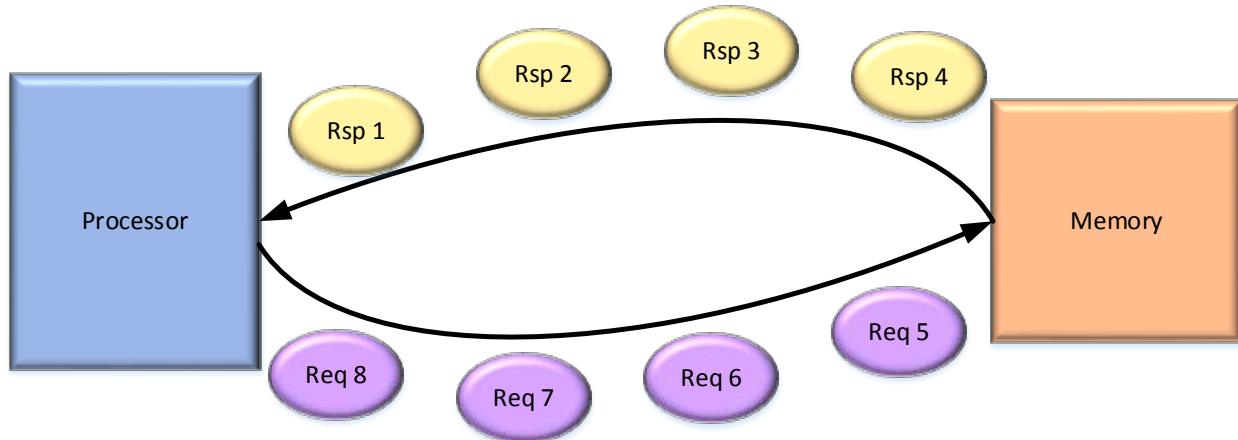


Figure 3-2: Example Pipelined Request and Response Packets

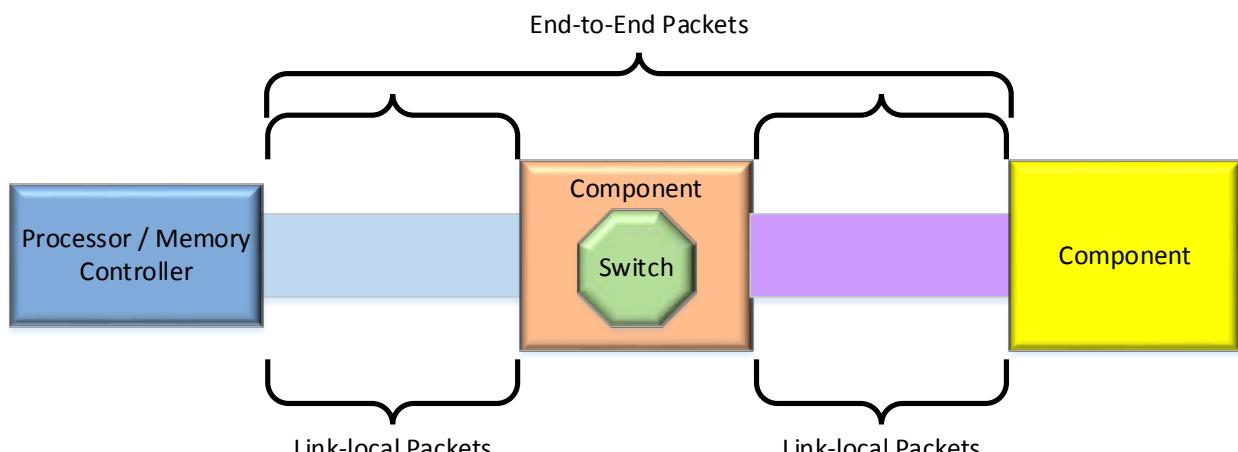
- 5 The architecture allows any-to-any communication between component types, as well as allows any component type to issue and process any packet type.

## 3.2. Features

This section covers various architectural features and associated requirements and specifications.

### 3.2.1. Packet Types

- 10 There are two primary packet types—link-local and end-to-end. As their names imply, link-local packets operate on a per link basis between two interfaces and end-to-end packets operate between any two components across one or more intervening links. As illustrated in *Link-local vs. End-to-end Packets*, there are two independent link-local domains (light blue and light purple) and there are three possible end-to-end domains (blue-orange, blue-yellow, and orange-yellow).



15

Figure 3-3: Link-local vs. End-to-end Packets

Link-local packets communicate link-specific control information, e.g., per-link flow-control credits, link initialization, link synchronization, etc. Link-local packets operate completely independent of end-to-end packets.

5 End-to-end packets communicate component-specific data and control information. At a minimum all components support a common set of request packet types—read, read response, write, acknowledgement, and control—to ensure interoperability.

### 3.2.2. Address

Many request packets contain an Address field. The Address field identifies an addressable resource within the destination component.

10 The packet's Address field is acted upon only by the destination component; directly-attached and packet-relay components do not examine the packet's Address field to make packet forwarding decisions.

The following are the features and requirements associated with an Address:

- 15 • A request packet's Address field shall target a Data Space or a Control Space resource (see *Component with Data and Control Spaces*).
- A component shall support a single Data Space. The Data Space is intended for application use.
  - o All non-*In-band Management* packets shall operate on the component's Data Space.
  - o A request packet may access the entire Data Space independent of the ingress interface.
  - o The maximum size of Data Space depends upon the OpClass:
    - 20 ▪ P2P-Core OpClass supports up  $2^{44}$  bytes per logical bank. A request packet's Address field represents a byte offset from the indicated logical bank's byte 0.
    - P2P-Coherency OpClass and explicit OpClasses support a  $2^{64}$  bytes per component.
- 25 • A component shall support a single Control Space that shall be accessible through *In-band Management* or *Out-of-band Management*.
  - o The Control Space shall contain Control Space structures.
  - o The Control Space may contain implementation-specific structures and data as well as space for executables, working set space, address translation structures and logic, etc.
  - o Unless explicitly stated by this specification, a Control Space address shall have integer 30 single-byte resolution, and shall represent a byte offset from Control Space byte 0.
  - o The maximum size of Control Space depends upon the OpClass:
    - P2P-Core OpClass supports up to  $2^{44}$  bytes.
    - P2P-Coherency OpClass and Control OpClass support up to  $2^{52}$  bytes.
- 35 • An *In-band Management* request packet may access the entire component's Control Space independent of the ingress interface on which it arrived.

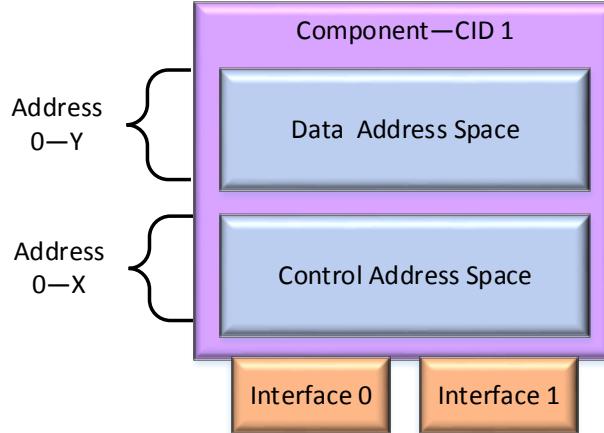


Figure 3-4: Component with Data and Control Spaces

- A component may support address resource mapping services, for example, an address translation table or equivalent. Such services may be used to bypass failed resources (e.g., due to yield, failure, or fatigue), to provide wear-leveling services, to enable components to exchange packets without requiring additional translation, to transparently enable or facilitate services (e.g., see *Accelerators*), and so forth.
- An address may be associated with a physical resource, e.g., memory. An address may be associated with a non-physical resource, e.g., associated with an address range used to exchange messages such as inter-process communications.
- An address may be interpreted:
  - As a zero-based offset where resources are accessed starting at byte 0 through the maximum supported address.
  - As a non-zero-based address where the destination component supports a memory management unit that translates the packet address field into a local address. This translation is outside of this specification's scope.

### 3.2.3. Virtual Channels

Virtual Channels (VC) provide a mechanism to create multiple logical streams across a single physical link. Multiple VCs enable packet differentiation, supporting the following objectives:

1. To shape and differentiate packets to meet service rate requirements.
2. To enhance resource usage diversity to increase performance by reducing head-of-line blocking and / or cross path blocking.
3. To facilitate multipath and dynamic routing
4. To prevent deadlocks due to:
  - a. Request-response packet dependencies.
  - b. Deadlock patterns formed by overlapping communication paths between components
5. To support unique ordering and relay policies required for *PCIe-Compatible Ordering (PCO)*.

The following are the features and requirements associated with VCs:

- Each VC logically represents a set of per-interface, per-direction transmit and receive buffers and associated tracking logic and resources.
- Each VC identifies a unique flow-control domain in each direction, across the interface. See *Link-Local Flow Control* for details on features and requirements.

- VCs are delineated by consecutively increasing identifiers, VC0, VC1, etc. and communicated via the packet's VC field.
- Each interface shall support VC0. A single VC interface (see *Single and Dual-VC Communication*) represents the simplest implementation and is associated with a component that only acts as a Requester or only as a Responder.

5

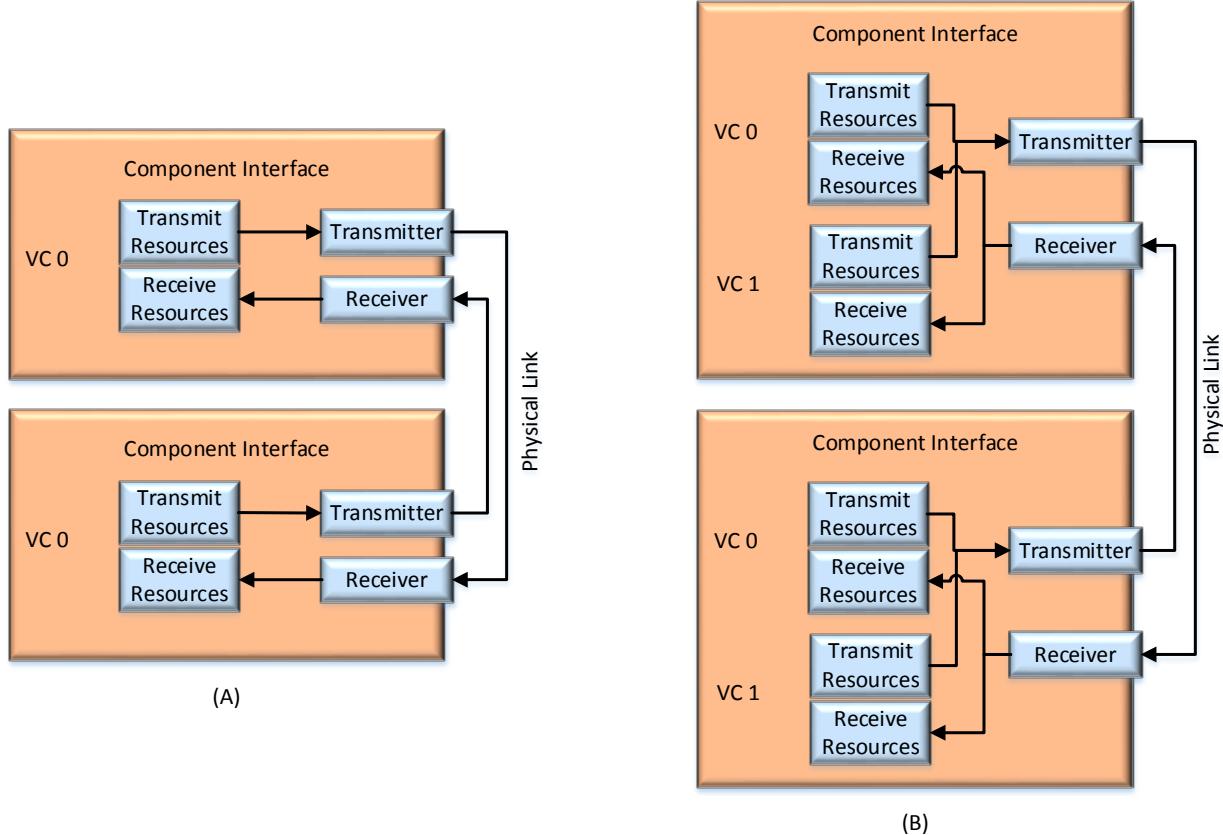


Figure 3-5: Single and Dual-VC Communication

- Each interface may support additional VCs.
  - Interfaces that support the P2P-Core OpClass shall support a single, implicit VC0.
  - Interfaces that support the P2P-Coherency OpClass may support up to 4 VCs.
  - Interfaces that support explicit OpClasses may support up to 32 VCs.
  - VC shall be implemented in contiguously increasing order, VC0, VC1 ... VC[n].
  - The transmitting component determines the VC that a packet uses and marks the packet with that VC number.
  - When two interfaces connected by a link support disparate VC numbers, the number of enabled VCs is limited to the lowest common VC set. For example, if one interface supports VC0-VC3 and the other connected interface supports VC0-VC2, then VC3 is not enabled. Software is responsible for ensuring operations are correctly mapped to the enabled VCs to ensure interoperability and avoid communication deadlock.
  - End-to-end packets from different VCs are time-multiplexed onto the physical link. *Example Time-Multiplexing Multiple Packets with Different VCs* illustrates a quad-VC solution where each VC transmits a set of packets (T) and receives a set of packets (R). These packets are interleaved on the physical lanes.

10

15

20

- VC arbitration may be applied at each egress interface to control individual VC service rate and egress interface transmission order.
  - All VCs shall be periodically serviced to ensure all packets make forward progress.
  - Due to the variety of design options and solution needs, VC arbitration is outside of this specification's scope.
- For VCs with PCO disabled, VC configuration shall ensure that no interface or link shall carry, on the same VC in the same direction, intermixed request and response packets.
- For VCs with PCO enabled, VC configuration shall follow the rules in *PCO Operation and Requirements*.

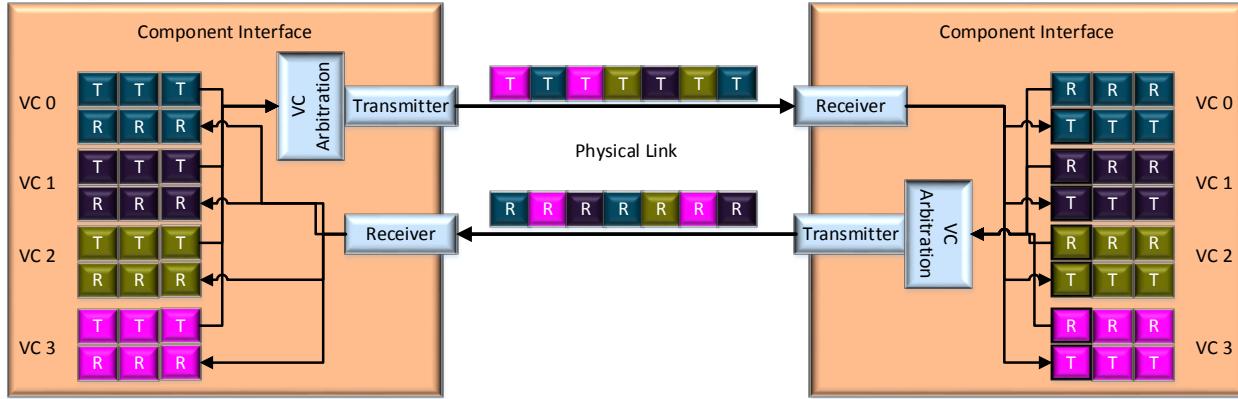


Figure 3-6: Example Time-Multiplexing Multiple Packets with Different VCs

- A component that acts only as a Requester or acts only as a Responder may implement a single VC (VC0).
- A component that acts only as a Requester and supports explicit OpClasses shall implement at least two VCs (one for request packets and one for response packets).
- A component that acts as a Requester-Responder shall implement at least two VCs.
  - A component that acts as a Requester-Responder and supports explicit OpClasses shall implement at least three VCs—two that are associated with request packets and one that is associated with response packets.
  - A component supporting PCO communications should implement at least as many VCs as recommended in *PCO Operation and Requirements*.
- Unless explicitly stated otherwise by this specification, a component that acts only as a Responder shall impose only request-upon-response packet dependencies.
  - Request-upon-response packet dependency is defined as a Responder shall not be able to receive new request packets until it has link resources available to transmit response packets. A Responder controls this by limiting the return of *Explicit Flow Control* credits.
- A single-VC component shall use VC0 to exchange request and response packets.
  - In point-to-point topologies, single-VC component request and response packets shall travel in opposite directions.
  - In switch-based topologies, components shall be configured to ensure that no single-VC interface on any component carries request and response packets that travel in the same directions.
- Components that support packet relay may support VC remapping, i.e., the VC contained in the packet's VC field may be remapped to a different VC. VC remapping can be used to prevent deadlocks and to reduce head-of-line blocking by distributing packets across many VCs, e.g., distributing packets from multiple two-VC components attached to a 16-VC component.

- See additional VC requirements relative to flow control found in *Link-Local Flow Control*.
- See additional VC requirements in *Configuration and Management*.
- See additional VC requirements relative to routing deadlock avoidance in *Topology Considerations for PCO and Routing Deadlock Avoidance for PCO*.

### 5 3.2.4. Component Identifiers

A Component Identifier (CID) is assigned to a component to uniquely identify a component within a single subnet that supports any explicit OpClass (see *End-to-end OpClasses & OpCodes*).

The following are the features and requirements associated with a CID:

- A CID shall not be configured if a component supports only point-to-point optimized packet formats that do not contain source or destination component identifier fields.
- A CID may be any unsigned integer from 0 to  $(2^{12}-1)$ , inclusive.
- A CID shall be subnet unique, i.e., no two components within the same subnet shall be assigned the same CID.
- Software shall configure one CID, and may configure multiple CIDs to a component.
- A SCID (Source CID) is used to identify the source component that first transmitted a request or response packet within the present subnet topology.
- A DCID (Destination CID) is used to identify the destination component within the present subnet topology that a request or response packet is intended to reach.
- A component whose CID is configurable shall support the entire CID range.

### 20 3.2.5. Global Component Identifiers

A Global Component Identifier (GCID) uniquely identifies a component that explicitly communicates with peer components within a multi-subnet topology.

The following are the features and requirements associated with a GCID:

- A GCID shall be a 28-bit identifier.
- A GCID shall be created by concatenating a subnet identifier with a configured CID, i.e.,  $GCID = Subnet\ Identifier \parallel CID$ .
- A GCID shall be unique within a multi-subnet topology, i.e., no two components regardless of where they are located within the topology shall be configured with the same subnet identifier and the same CID.
- A SGCID (Source GCID) is used to identify the source component that first transmitted a request or response packet within a multi-subnet topology.
- A DGCID (Destination GCID) is used to identify the destination component within a multi-subnet topology that a request or response packet is intended to reach.
- By definition, a component that supports multiple CIDs or SIDs supports multiple GCIDs.

### 35 3.2.6. End-to-end Unicast Communications

Unicast communication is the exchange of end-to-end packets between a source component and a single destination component. Unless specifically stated otherwise in this specification, all operations and packets shall use unicast communications.

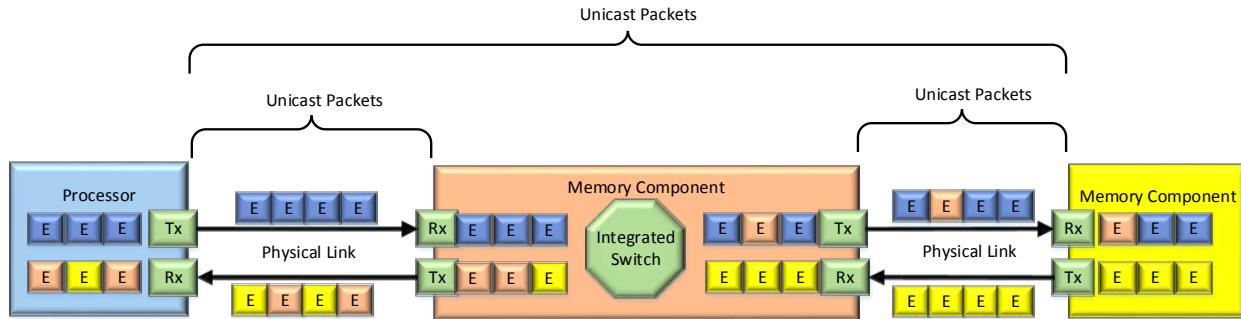


Figure 3-7: Unicast Packets between Processors and Memory Components

The following are the features and requirements associated with unicast packets:

- End-to-end (E) unicast packets may flow between any two components, transparently crossing intervening components. The color of each packet indicates the component that originally transmitted the packet.
- Unicast packets may flow across any VC. If multiple VCs are supported, packets may be assigned to different VCs based on packet type, e.g., request packets on VC1 and response packets on VC0.
- A switch shall not relay point-to-point optimized packets (see *End-to-end OpClasses & OpCodes*).
- Intervening components should examine only those packet fields required to relay the packet and ensure correct communications; unless explicitly stated otherwise by this specification, they may ignore all other fields and payload (if present).

### 3.2.7. End-to-end Multicast Communications

Multicast is an optional one-to-multiple / multiple-to-multiple communication paradigm designed to simplify and improve communication efficiency between participating components. As illustrated in *Multicast Example*, packet 16 (T16) is transmitted by the processor and subsequently replicated by multicast switch components (discrete or integrated) and ultimately relayed to all participating components.

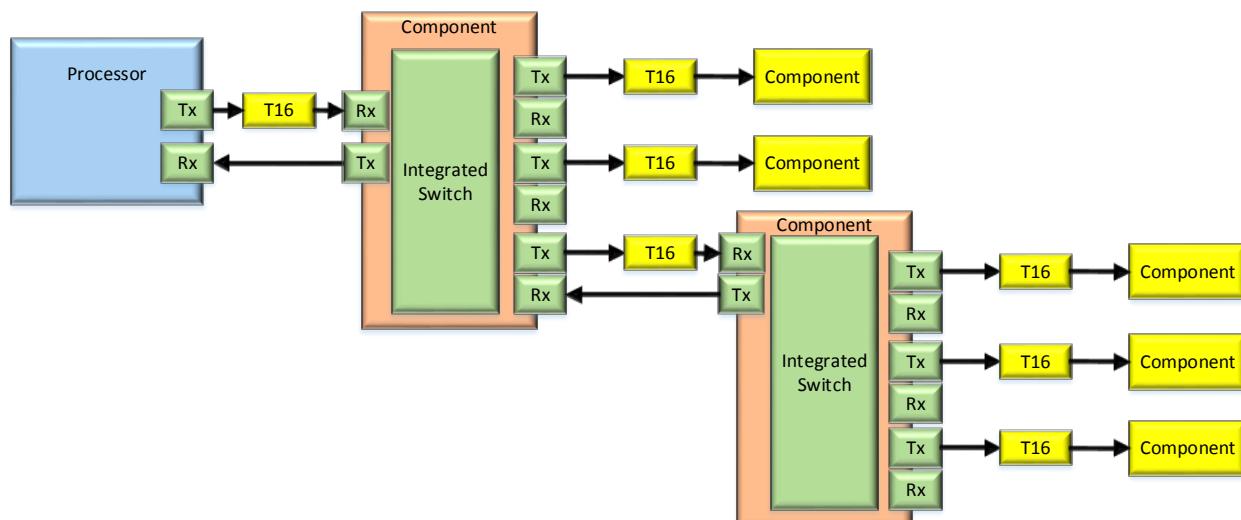


Figure 3-8: Multicast Example

5 Gen-Z supports two types of multicast—reliable and unreliable. Reliable multicast ensures Reliable Delivery to each receiving component. This requires each receiving component to generate a unicast acknowledgment, and the use of a multicast packet retransmission and recovery algorithm if an error is encountered (Requesters are responsible for error handling and recovery). Unreliable multicast does not ensure Reliable Delivery as intervening and receiving components may silently discard packets as conditions warrant.

10 Though multicast is optional to support, Requesters that need to interact with management services where the specific management component is unknown should support the ability to transmit unreliable multicast request packets to the management multicast group. Similarly, Requesters that support *Collective Operations* should support the ability to transmit multicast request packets to communicate with collective accelerators. Finally, independent of multicast packet replication and relay support, switches shall support the ability to relay multicast packets through the Default Multicast Egress Interface (see *Switches*).

### 3.2.8. Accelerators

15 Accelerators are integrated or discrete computation and data-transformation engines. Accelerators may be application-visible (e.g., computational off-load) or application-transparent (e.g., automatic memory compression, cryptography services, data classification / analytics, etc.). Using abstraction techniques and well-defined management primitives and operational semantics enables developers to incorporate accelerators into any Gen-Z solution or component type.

#### 20 **3.2.8.1. Application Transparent Accelerator Operation**

25 *Example Transparent Accelerator Operation (A)* illustrates application-transparent acceleration based on the accessed resource. In this example, the request packet's Address field or operation type are used to identify the accelerator instance. The write packet's payload is transparently executed upon by the accelerator. A similar sequence is performed when generating a Read Response packet; the protocol fields within the Read request packet are processed to determine the accelerator instance and the underlying application resources, which are processed with the results returned in the Read Response packet payload.

30 *Example Transparent Accelerator Operation (B)* illustrates application-transparent acceleration to provide computational offload. In this example, the SoC contains a Home Agent, and the accelerator contains a Caching Agent. A Home Agent is logic within a component that maintains a portion of the coherent address space associated with the coherent memory resource (e.g., a set of pages), providing data and ownership responses to Gen-Z coherency request packets. A Caching Agent is logic within a component that initiates Gen-Z coherency request packets to an address space, and can maintain copies of data in its own cache structure.

35 In this example, in order for the accelerator to access the coherent memory resource, it issues a mix of coherent and non-coherent read and write request packets based on which address space pages permit coherency request packet access (see *Memory Management*). In many solutions, only a small subset of pages require coherency, e.g., to pass data pointers or to access control plane logic; a significant percentage of the data is accessed using non-coherent read and write request packets.

40 *Example Transparent Accelerator Operation (B)* also illustrates a non-coherent memory resource that can be directly accessed by the SoC and the accelerator. In this example, the non-coherent memory

resource contains the application data set that can be directly accessed by the accelerator. To identify the specific data for the accelerator to access, the following steps are taken:

1. OS / middleware configures an address range associated with the coherent memory resource to permit coherent access. This address range can be used to communicate control information or application data.
2. The application / middleware communicates a subset of this address range to the accelerator, e.g., a small number of pages that contain control structures used to manage application data.
3. The application / middleware communicates the address of a pointer that indicates where to locate the next set of application data to be processed.
4. Using this pointer, the Caching Agent within the accelerator performs a *Read Shared* to acquire a shared copy of the cache line and a current copy of the pointer value. The SoC's Home Agent updates its cache line tracking state to reflect the Caching Agent that has Shared access.
5. When the application / middleware has new data to be processed by the accelerator, it updates the pointer value. In order to coherently update the pointer, the Home Agent issues an Invalidate request to the accelerator's Caching Agent. The Caching Agent evicts the cache line from its local cache, and acknowledges the request packet.
10. Upon receiving the Invalidate acknowledgment, the Home Agent permits the cache line to be updated.
15. Upon acknowledging the Invalidate request packet, the accelerator issues a new *Read Shared* to acquire a Shared copy of the cache line and a current copy of the pointer value. Using the pointer value, the accelerator accesses the application data. The application data could be in the coherent memory resource (accessible using any mix of coherent or non-coherent operations) or in the non-coherent memory resource (accessible using non-coherent operations).
20. When the accelerator has completed its work, it issues an *Exclusive* request packet to a coherent cache line in the coherent memory resource in order to transmit a coherent write request packet to a flag that indicates completion.
25. Once the page(s) that contain the pointer and completion flag are set up, the application / middleware can transparently accelerate data located in the coherent memory resource or the non-coherent memory resource using any mix of coherent and non-coherent operations by repeating steps 4-8.
30. Further, this can be applied to any number of memory components, or any mix of component types.

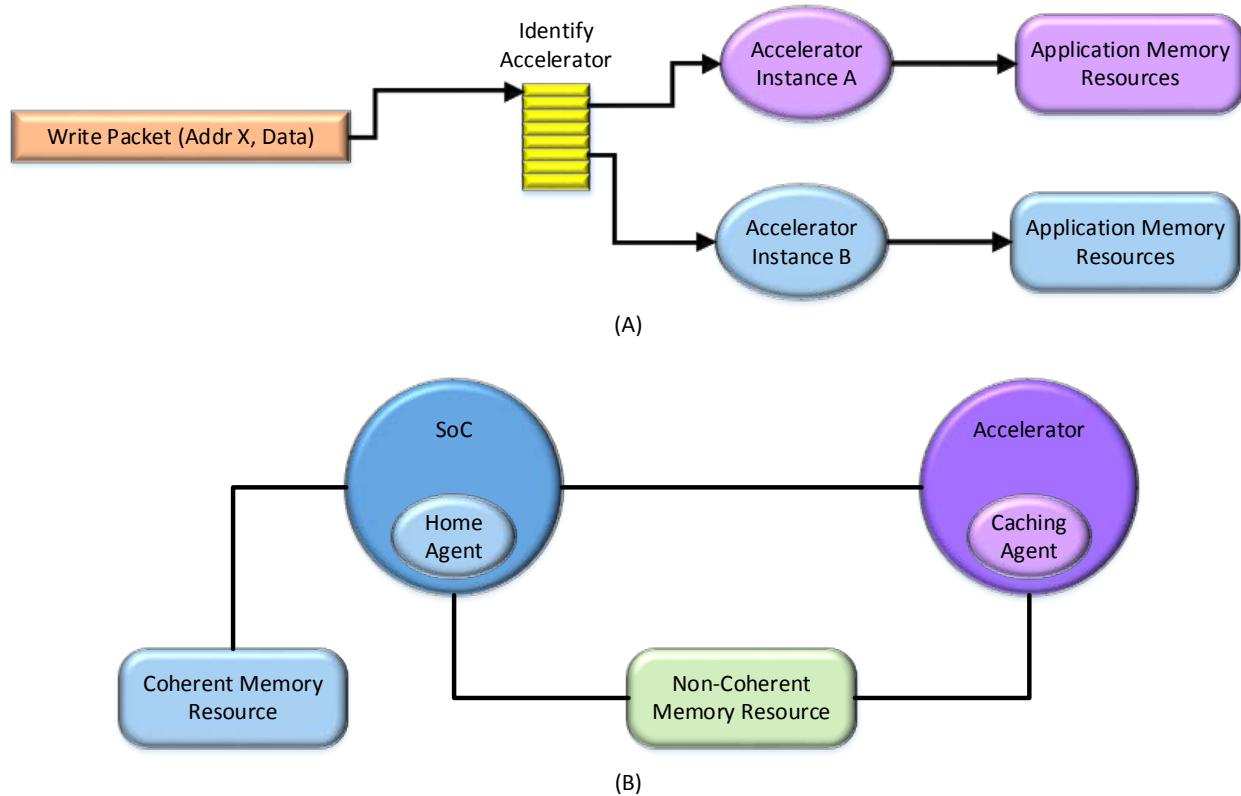


Figure 3-9: Example Transparent Accelerator Operation

### 3.2.8.2. *Non-Transparent (Application Visible) Accelerator Operation*

A non-transparent accelerator is not associated with a specific subset of a component's Data Space, nor is it constrained to a single component. Most non-transparent accelerators use a work queue model. In a work queue model, the application communicates with the non-transparent accelerator through one or more work queues (WQ) and a completion queues (CQ). The application posts a series of work requests to the WQ, and the accelerator posts a series of completion events to the CQ with optional interrupt generation as illustrated in *Example I/O-based Accelerator*. To post a work request, the application writes to an accelerator-specific address, and the application reads completions from a different accelerator-specific address.

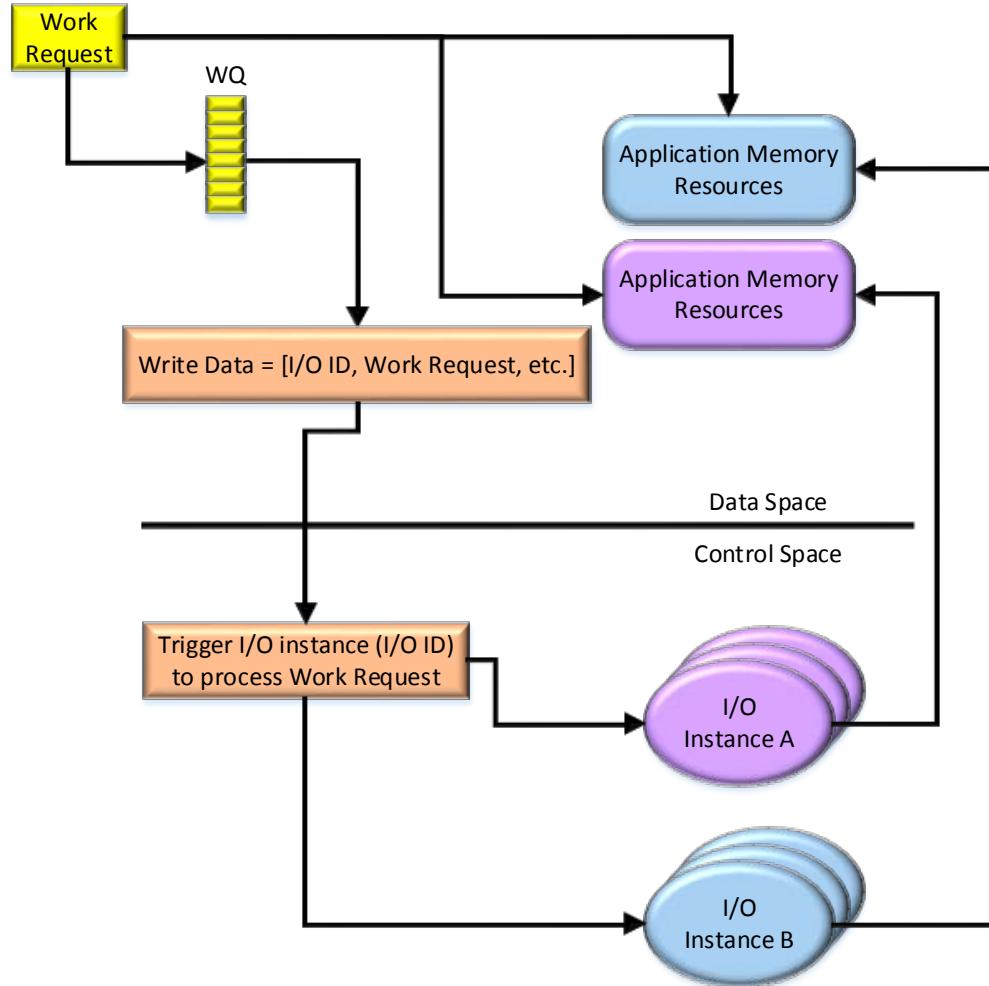


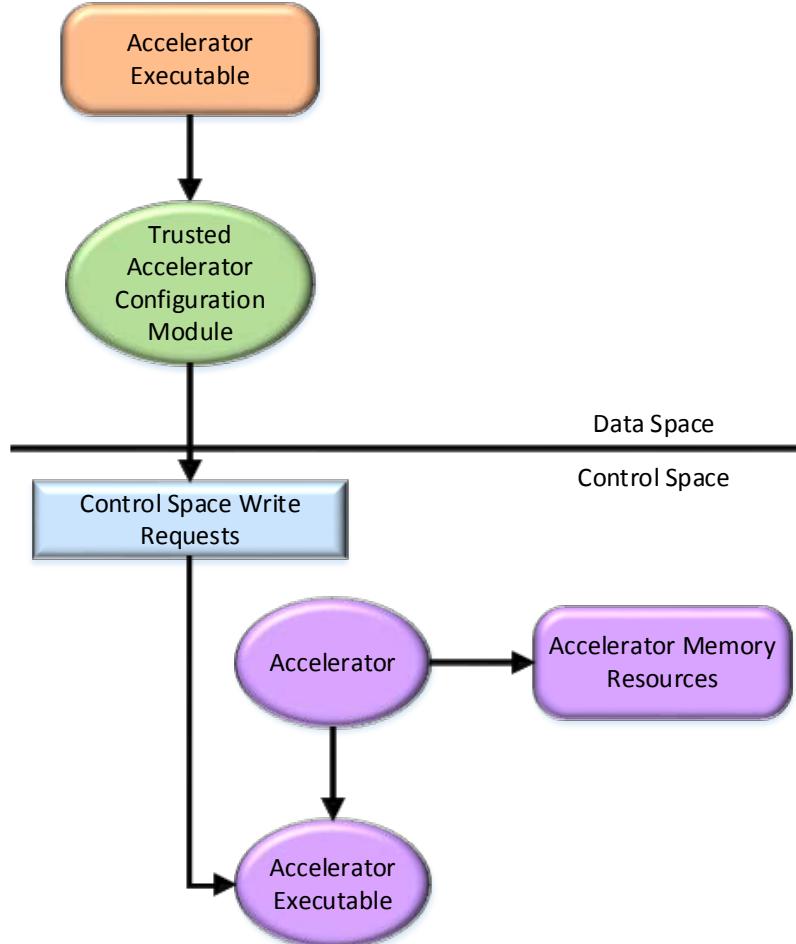
Figure 3-10: Example I/O-based Accelerator

### 3.2.8.3. Accelerator Features and Requirements

The following are the features and requirements associated with accelerators:

- Accelerators may be composed of any mix of computational and data transformation engines, including but not limited to: processors (SoC, GPU, NPU), FPGA, DSP, ASIC, state machines, custom logic, I/O devices, etc. Accelerators may be implemented as discrete components or integrated within other components such as memory or storage components.
- Accelerators may communicate using any mix of coherent and non-coherent request and response packets. For example, pages within a Responder can be configured to permit coherency request packets to target a page using coherent and non-coherent read and write operations at the Requester's discretion. In many applications that use coherency, only a small fraction of the data needs to be exchanged using coherency operations.
- Accelerators may be initialized in hardware, initialized via software, or a combination thereof. Trusted management modules may initialize or download software through a sequence of control write requests or through mechanisms outside of this specification's scope. For example, *Accelerator Executable Initialization* illustrates a sequence of Control Space Write Requests to 'download' an accelerator executable.

- Accelerators may operate on Data Space memory resources, on Control Space memory resources, or a combination of Data Space and Control Space resources.
- Accelerators may be dynamically and / or application-transparently enabled or disabled through mechanisms outside of this specification's scope.



5

Figure 3-11: Accelerator Executable Initialization

### 3.2.9. Access Keys

An Access Key is an opaque identifier used to provide hardware-enforced component isolation between components that support explicit OpClasses. Each Access Key defines an access control domain that is conceptually similar to a virtual network, e.g., *Multiple Independent and Shared Memory Access Regions* (A) illustrates four independent component pairs—a processor and a memory component—sharing a common discrete switch. Only components with the same Access Key can communicate with one another, though all exchange packets through the same switch. (B) illustrates the same components divided into two independent access control domains. As before, only components with the same Access Key can communicate with one another, though all exchange packets through the same switch.

10

15

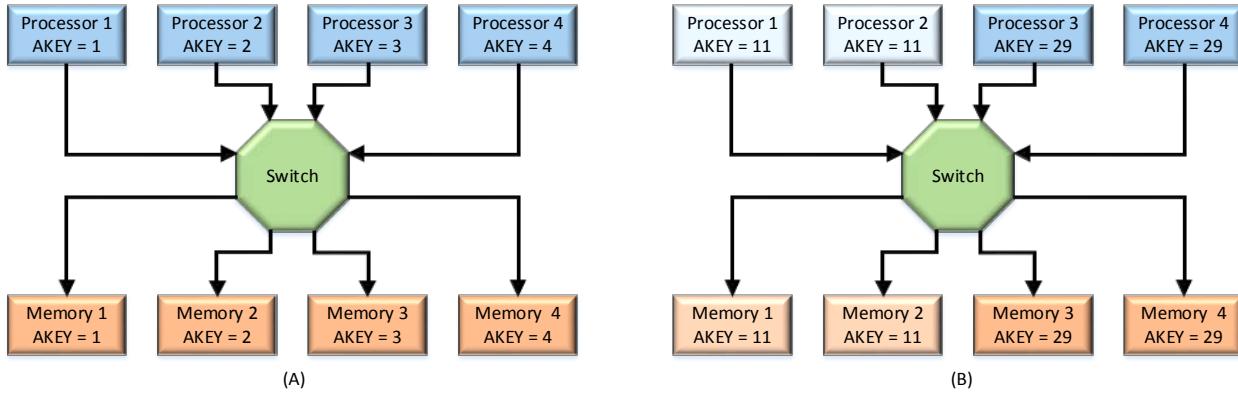


Figure 3-12: Multiple Independent and Shared Memory Access Regions

The following are the features and requirements associated with Access Keys:

- Access Keys may be used in any topology and with any component mix.
- The Access Key field shall be present only in explicit OpClass end-to-end packets (see *Base Formats for End-to-end Packets*).
- Each Access Key is an unsigned integer that is associated with a given Access Key Space. An Access Key Space may span one or more subnets. Access Key Space management is outside of this specification's scope.
- Each Access Key Space contains  $2^6$  Access Keys (0 to  $(2^6 - 1)$ ).
- Components that support Access Keys shall support the entire Access Key Space, i.e., any valid Access Key may be used at any given time if Access Keys are enabled.
- The Default Access Key shall be 0x0.
  - By default, Access Key validation shall be disabled. All components shall be implicitly associated with the Default Access Key.
  - The Default Access Key shall be implicitly associated with any packet that does not contain the Access Key field.
  - If Access Key validation has not been configured and enabled, then the Default Access Key shall be copied to the explicit OpClass packet Access Key field.
- Zero or more Access Keys may be configured per Requester or Responder component. If configured, an Access Key shall apply to the entire component.
- Zero or more Access Keys may be configured per component ingress or egress interface. If configured, an Access Key may be configured on multiple component interfaces.
- Access Keys are configured through the *Component PA (Peer Attribute) Structure* and the *Interface Structure*. The *Component PA (Peer Attribute) Structure* may be used to provide additional per peer component access control independent of Access Keys.
- Access Key validation involves verifying whether an end-to-end packet may be transmitted or received based on matching one of the component's configured Access Keys and / or the ingress and egress interface Access Keys. Access Key validation errors are classified as Access Errors (AE). If Access Key validation is enabled,
  - Access Key validation shall not be performed on any interface enabled to exchange P2P-Core, P2P-Coherency, or P2P-Vendor-defined OpClass packets; these packets are optimized for point-to-point topologies, and do not contain an Access Key field.
  - Access Key validation and error handling shall be performed on all explicit OpClass packets as specified in *Access Key Validation*.

- If a request packet requires Reliable Delivery, then the Access Key used in a request packet shall be used in the corresponding response packet.

**Developer Note:** Though the number of Access Keys is limited, developers can use a combination of Access Keys, packet relay table configuration, and per component access control to support robust hardware-enforced isolation between components.

5

### 3.2.10. Region Key (R-Key)

A Region Key (R-Key) is an opaque identifier used to enforce page isolation within a component. *Single and Multi-Component Pages with R-Keys* (A) illustrates a memory component with multiple pages. A subset of these memory pages (pages 1 and 2) are configured with unique non-Default R-Key values. Only packets whose R-Key matches the target page's configured R-Key are executed. *Single and Multi-Component Pages with R-Keys* (B) illustrates how this concept is expanded to cover multiple memory components each containing a similar set of pages that are configured with the same R-Key values.

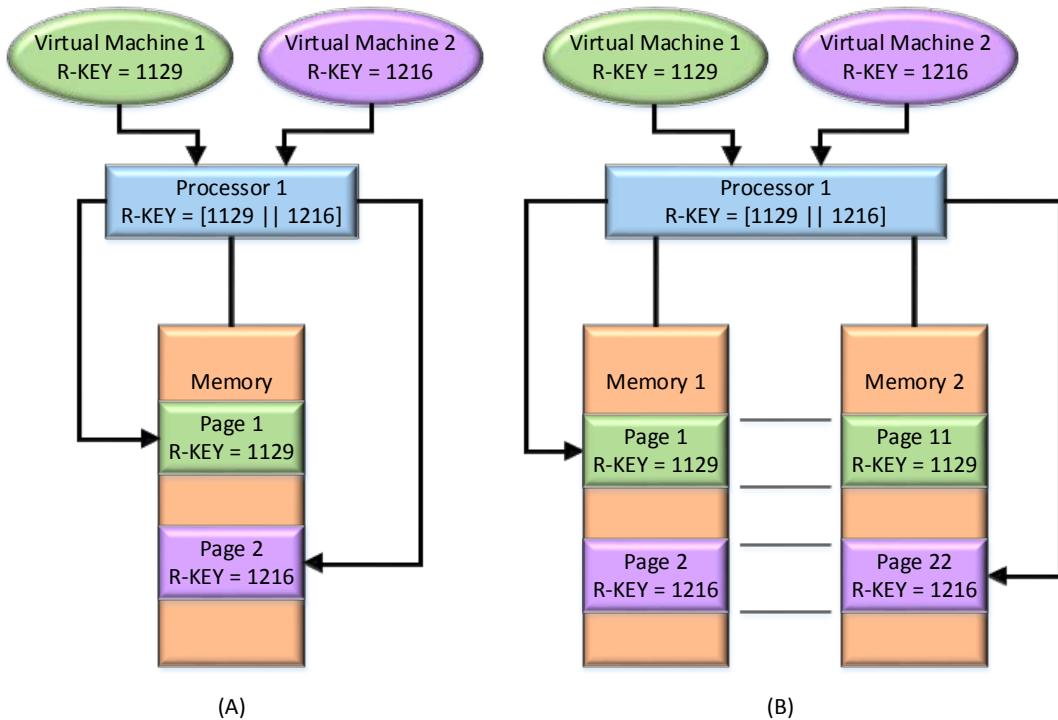


Figure 3-13: Single and Multi-Component Pages with R-Keys

**Multiple Virtual NIC Sharing a Physical NIC** illustrates four processors connected to a discrete switch, which in turn, is connected to a shared I/O device that is connected to an external network, e.g., Ethernet.

- The I/O device supports four virtual network interface controllers (vNIC).
- Each vNIC is associated with one or more distinct pages.
- Each page is configured with a unique non-Default R-Key.
- Processor request packets for a given vNIC contain the associated page's configured R-Key.

15

20

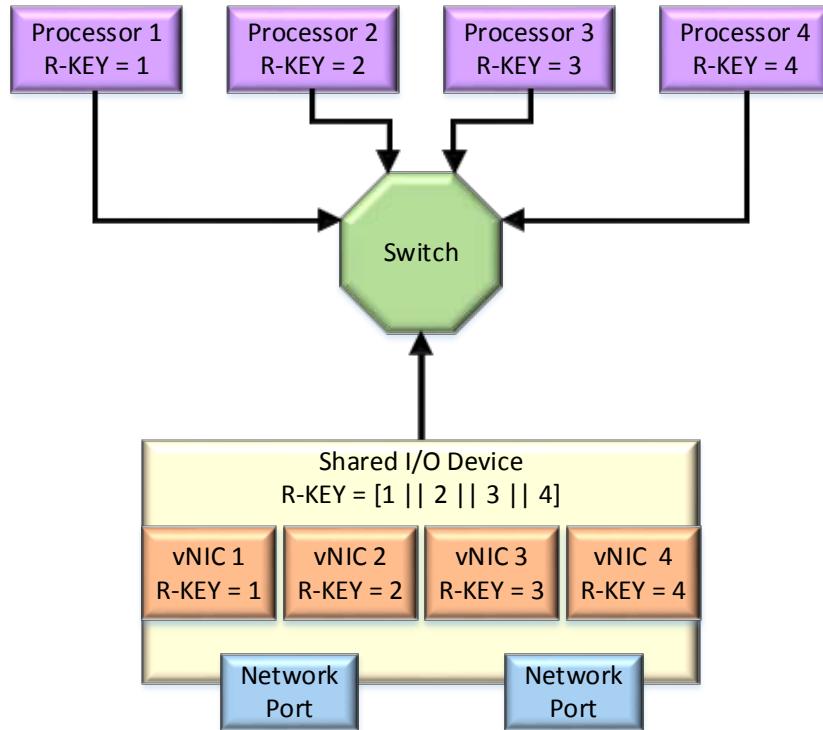


Figure 3-14: Multiple Virtual NIC Sharing a Physical NIC

The following are the features and requirements associated with R-Keys:

- R-Keys may be supported by any component type.
- The Default R-Key shall be 0x0.
- Only select explicit OpClass request packet contain the R-Key field.
  - Explicit OpClass response packets do not contain the R-Key field.
  - Non-explicit OpClass request packets do not contain the R-Key field. These packets may access only pages that do not support R-Keys or are configured with the Default R-Key.
  - If a given page is configured with a non-default R-Key, then access by a non-explicit OpClass request packet shall trigger an Access Error.
- Each R-Key is an unsigned 32-bit integer that is associated with a given R-Key Space.
  - An R-Key Space may span multiple subnets.
  - R-Key Space management is outside of this specification's scope.
  - Each R-Key Space contains  $2^{32}$  R-Keys (0 to  $(2^{32} - 1)$ ).
  - Components that support R-Keys shall support the entire R-Key Space, i.e., any R-Key may be used at any given time if R-Keys are enabled.
- An R-Key may be associated with read-only or read-write access control.
- If a Requester supports a ZMMU, then an R-Key is configured within the component's *Requester PTE*, and is used to create the request packet. If a Responder supports a ZMMU, then a read-only R-Key and a read-write R-Key are configured within the component's *Responder PTE* for each page that is made visible to Requesters.
  - If a component does not support a ZMMU, then R-Key configuration and management is outside of this specification's scope.
- A component may support any mix of packet types and resources as long as all request packets targeting a given page with a non-Default R-Key configured contain the corresponding R-Key.

**Developer Note:** Though R-Key management is outside of this specification's scope, some solutions use de facto schemes. For example, if an operating system uses 32-bit process or thread identifiers, then the identifiers associated with the process or thread that created or controls the page can be used as an R-Key. Similarly, a 16-bit random number can be concatenated with a 16-bit process or thread identifier to create an R-Key. Use of such identifiers can simplify management as well as facilitate the use of virtual addresses when combined with the packet's Address field. See R-Key Domain (RKD) for additional requirements when constructing an R-Key.

### 3.2.10.1. R-Key Domain (RKD)

The R-Key space may be partitioned into a maximum of 4096 sub-spaces. A sub-space is referred to as an R-Key Domain (RKD). Since R-Keys are managed through means outside of this specification's scope, management may use a *Component RKD Structure* to prevent an untrusted application from issuing request packets with R-Keys from an unauthorized RKD. Restricting R-Key use to only an authorized RKD:

- Simplifies R-Key revocation by enabling management to revoke RKD authorization from one or more Requesters without affecting the authorization of other Requesters.
- Simplifies R-Key management of Responders that are shared by multiple Requesters, e.g., each Requester can be assigned to a unique RKD.
- Enables a service to use a dedicated RKD to access resources on its clients without enabling those clients to use that same RKD to access related resources on other clients of the service.
- Enables a fabric manager to independently control through *In-band Management* which R-Keys an untrusted Requester can transmit in its request packets.
- Enables a fabric manager to delegate responsibilities requiring special access permissions to a resource manager without giving it the full access permissions of a fabric manager, notably when accessing structures in Control Space that are protected by a *Component C-Access Structure*.

The following are the features and requirements associated with R-Key Domains:

- If a Requester supports transmitting R-Keys in request packets, then the Requester shall support RKDs.
- If enabled in the *Component RKD Structure*, then a Requester shall validate if the R-Key in any request packet containing an R-Key field is within an authorized RKD. The Requester uses the upper twelve bits of the R-Key as an index into the RKD Authorization Bit array. If the corresponding  $Bit_k == 1b$ , then the request packet may be transmitted; if not, then the request packet shall not be transmitted, and the Requester shall handle this as a component-local AE.
- The *Component RKD Structure* supports the concept of "trusted threads", which is a platform-specific mechanism for hardware to identify specific processor execution threads as being trusted. If supported by the platform and enabled in the *Component RKD Structure*, request packets generated on behalf of trusted threads may access any RKD, independent of RKD Authorization bit values.

**Developer Note:** Examples of trusted thread implementations include the System Management Mode (SMM) on Intel® processors and the TrustZone® technology on ARM® processors.

### 3.2.11. Multipath

The architecture supports a diverse spectrum of topologies. Some topologies support multiple paths between components. *Multipath Topology Example* illustrates two paths between components A and D. Solutions can take advantage of multiple paths to increase aggregate bandwidth, reduce the probability of jitter / congestion, improve resiliency, etc.

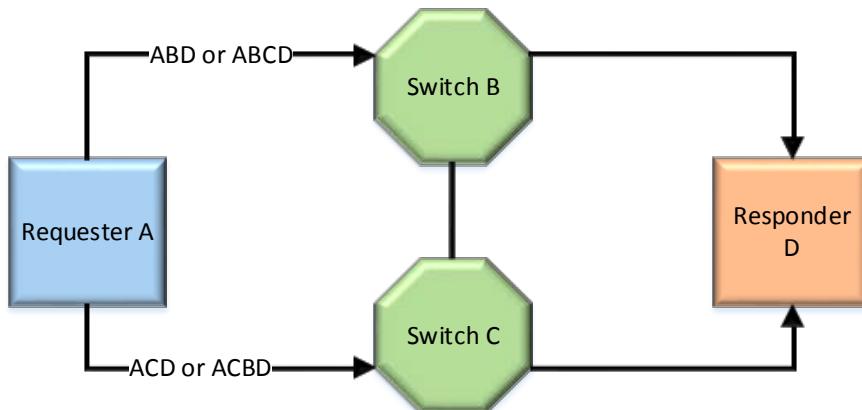


Figure 3-15: Multipath Topology Example

The following are the features and requirements associated with multipath topologies:

- All or a subset of the paths between any two components may be used to exchange packets.
- Each interface may be associated with one or more paths between components (*Multipath Topology Example* illustrates multiple paths between components A and D—ABD, ABCD, ACD, ACBD).
- A Requester shall use the *Component Destination Table Structure* to select the request packet's VC and egress interface.
- A Responder shall use the *Component Destination Table Structure* to select the response packet's VC and egress interface.
- At the Requester's discretion, a Requester may retransmit a request packet on any available interface that can reach the destination.
- Path failure may be communicated using an *Unsolicited Event (UE) Packet*, e.g., a component detects a failed interface and notifies management. Path failure may be indicated by a Maximum Request Packet Retransmission Exceeded event.

### 3.2.12. Vendor-defined Operations

Vendor-defined operations enable customized operations and architectural extensions. Vendor-defined packets contain a common subset of protocol fields to enable these packets to be relayed across a topology without intervening components being aware of their unique or customized nature.

Vendor-defined operations can be simple or complex, based on solution needs. For example, compound operations can be constructed to reduce communication overheads, improve efficiency, and reduce power consumption.

To enable high protocol efficiency and maximize flexibility, the architecture supports multiple ways to exchange vendor-defined OpCodes—a Point-to-Point (P2P) Vendor-defined implicit OpClass, a set of explicit vendor-defined OpClasses, and vendor-defined OpCodes within the P2P-Core and P2P-

Coherency OpClasses. This approach eliminates the need to transport vendor-unique identifiers within each packet and enables flexible solution composition and management.

### 3.2.13. Statistics

Statistics provide insight into solution operation and component behavior, which can be extremely useful for solution and component development as well as deployment. The architecture specifies statistics applicable to any component. Component vendors may also specify custom statistics using vendor-defined Control Space structures. In addition to statistics, the architecture supports performance markers which enable performance tools to conduct “life of a packet” monitoring.

### 3.2.14. Daisy-Chain Topologies

Requesters and Responder memory components optimized to support point-to-point topologies may also support daisy-chain topologies by incorporating additional interfaces and packet processing logic (this differs from a linear switch topology illustrated in *Serially Connected Components Using Integrated Switches* which relies on different packet processing and relay logic—see *Switches*). For example, *Example Daisy-Chain Topologies* illustrates three example daisy-chain topologies. *Example Daisy-Chain Topologies* (A) illustrates a single Requester and two Responders. Requester X is directly attached to a single Responder (A) which forwards packets to or from the second Responder (B) across a single link. The basic operation is as follows:

- Request packets flow from Requester (X) Interface 0 to Responder (A) Interface 0. If a given request is for Responder (A), then it validates and executes the request and returns the corresponding response on Interface 0. If the request is not for Responder (A), e.g., the target resource (memory) does not correlate to any Responder (A) resources, then (A) forwards the packet to (A) Interface 1 for transmission to Responder (B) Interface 0.
- Responder (B) performs the same steps and either executes the request or forwards it to (B) Interface 1 to the next component (if present). If (B) generates a response packet, it transmits the packet to Responder (A) Interface 1, which forwards the packet to (A) Interface 0, which in turn transmits the packet to Requester (X).

*Example Daisy-Chain Topologies* (B) illustrates how a daisy chain topology can be extended further to provide higher-capacity solutions. In this example, a single Requester interface can be used to access four daisy-chained Responders.

*Example Daisy-Chain Topologies* (C) illustrates a daisy-chain topology consisting of two Requesters and three Responders, e.g., a pair of controllers (e.g., storage or processors) sharing a set of memory components.

- Request packets flow from Requester X to Responders A-C with response packets flowing back to Requester X. Similarly, request packets flow from Requester Y to Responders C-A with response packets flowing back to Requester Y.
- If Requester X needs to transmit a request packet to Requester Y, then Requester Y is a Requester-Responder and is configured with an encoded Tag range that enables Responder C to determine when to forward request and response packets.

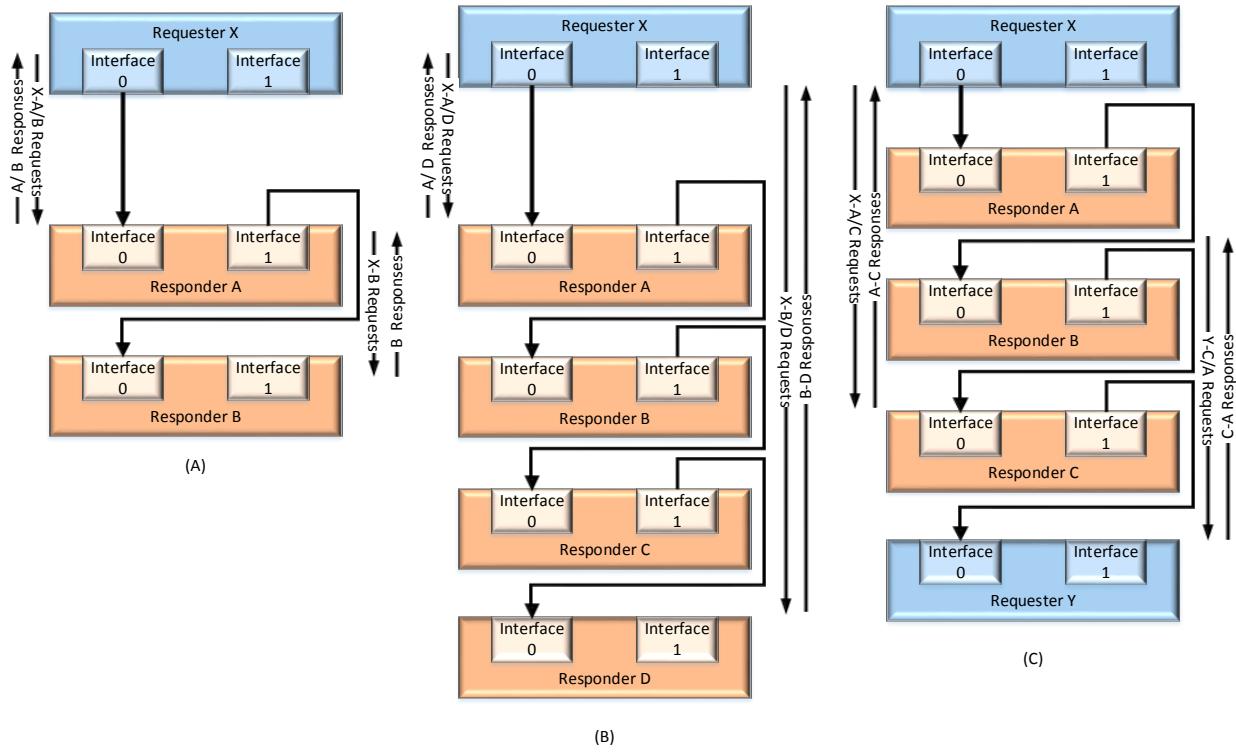


Figure 3-16: Example Daisy-Chain Topologies

To support daisy-chain topologies:

- A component shall support and have enabled the P2P-Core OpClass on each component interface used to attach daisy-chain components.
- A daisy-chain may contain a single Requester and multiple Responders. If a daisy-chain contains a single Requester, then the Requester shall be the first component in the daisy-chain, and the Responders shall be serially-linked to compose the rest of the daisy chain. Only Responders or Requester-Responders configured to operate only as Responders may be serially-linked to the first Requester.
- A daisy-chain may contain no more than two Requesters. If a daisy-chain contains two Requesters then one Requester shall be provisioned as the first component at each end of the daisy-chain, and the Responders shall be serially-linked between the two Requesters. Only Responders or Requester-Responders configured to operate only as Responders may be serially-linked between the two Requesters. If the Requesters are Requester-Responders, then they may exchange request and response packets with one another through the daisy-chain.
- A Requester shall support a single interface per daisy-chain. A Requester may support multiple component interfaces and, therefore, may support multiple daisy-chains.
- Each Responder shall support two interfaces per daisy-chain, one to transmit packets between each component pair (Requester and Responder or Responder and Responder). A Responder may support multiple daisy-chains.
- A Requester may be any component type. A Responder may be any memory component type, i.e., a component that contains addressable media that is organized into logical banks and logical rows.
- A Requester shall not transmit unreliable request packets to any Responder within a daisy-chain topology, e.g., a P2P-Core Unreliable Write.

- Management shall set *Interface I-CAP 1 Control Omit P2P Standalone Acknowledgment* = 0b in every Responder within a daisy-chain topology.
- For each daisy-chain, a Requester shall limit the maximum number of request packets to the smallest Max Requests supported by any component in the daisy chain. For example, if Responder<sub>0</sub> Max Requests = 256 and Responder<sub>1</sub> Max Requests = 128, then the Requester cannot have more than 128 request packets. Software configures the maximum allowed via *Core Structure Max Requests*.
  - If a daisy-chain supports a Requester at each end, then the sum of the maximum number of request packets values configured for the Requesters shall total less than or equal to the smallest *Core Structure Max Requests* value of all Responders on the chain.
- A request packet flows from the first Responder to the next until it reaches its destination or an error is detected. *Daisy Chain Request Processing Flow* illustrates the conceptual packet processing used by each Responder to determine whether the request packet is for it or to transmit it to the next component in the daisy-chain.
  - A Responder compares the request packet's Tag field (a unique identifier) to the range of Tags assigned to the Responder. If the request packet's Tag falls within the range, then the Responder validates and executes the request packet. If not, then the Responder shall forward the request packet through the configured daisy-chain interface in the order that it was received.
  - If the last component in the daisy-chain determines the request packet is not for it, then it shall silently discard the request packet.
- A response packet flows from the Responder that executed the request packet or detected an error towards the Requester. *Daisy Chain Response Processing Flow* illustrates the conceptual response packet processing used by each Responder to forward the response packet and ultimately to the Requester which validates and executes it.
- When multiple packets are available for transmission, a daisy-chain component should use round-robin arbitration between packets it generated and packets to be forwarded on behalf of a peer daisy-chain component.
- Components used in a daisy-chain topology shall support and be configured to use *Explicit Flow Control*.

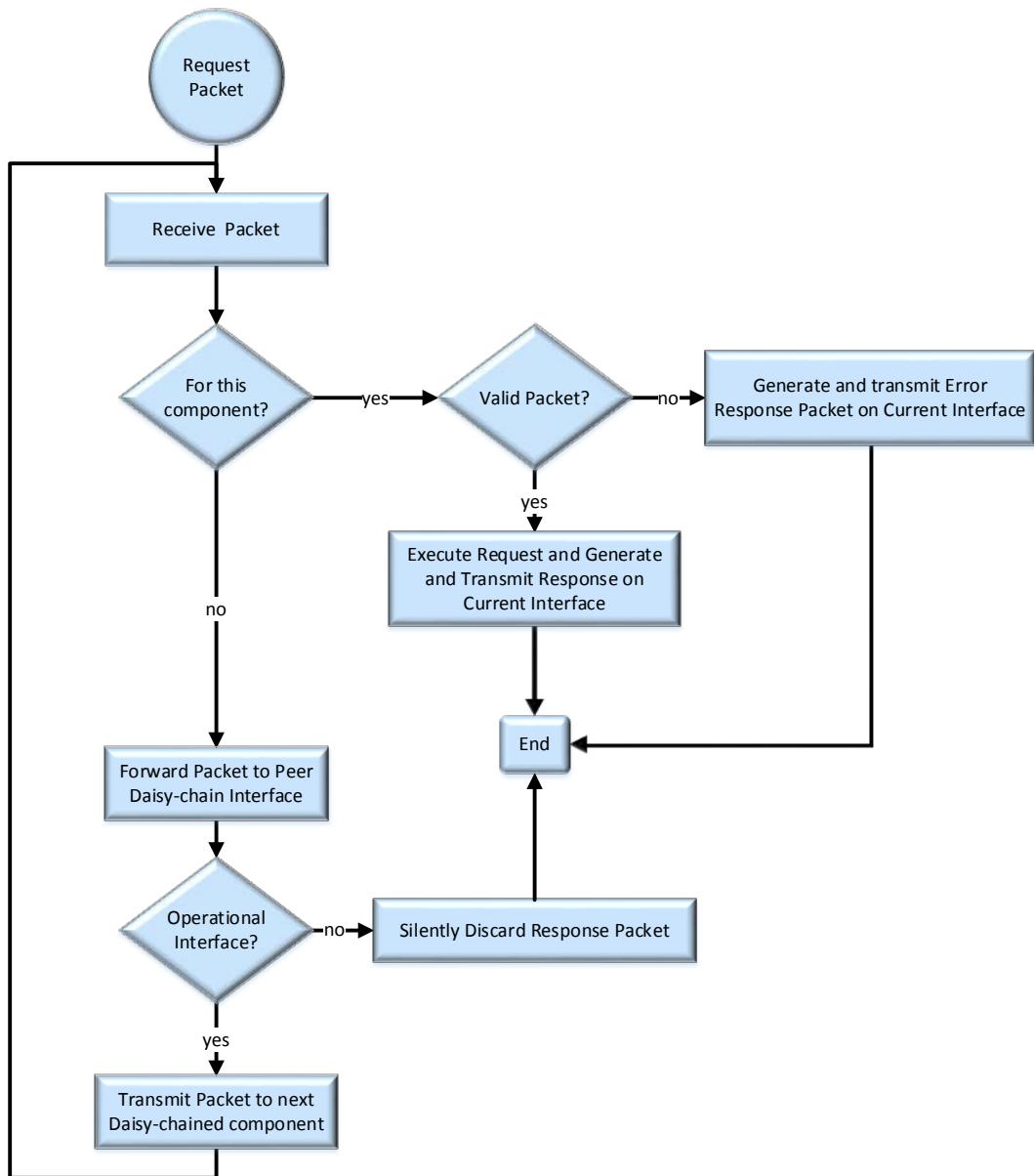


Figure 3-17: Daisy Chain Request Processing Flow

Table 3-1: Daisy Chain Request Processing Details

Step	Explanation
<b>For this component?</b>	Does the packet's Tag correspond to the component's encoded Tag range? If so, then the packet is for this component; if not, then forward the packet to the peer daisy-chain interface.
<b>Operational Interface?</b>	If <i>Interface I-CAP 1 Control Daisy-chain Peer Interface Configured</i> == 1b, and the interface identified by the <i>Interface Structure</i> Peer Daisy Interface ID field is in the I-Up state, then forward the packet to the peer interface and transmit the

Step	Explanation
	packet to the next component. If not, then silently discard the packet.
Valid Packet?	If packet validation fails, then generate a request-specific error response packet, else execute the request packet, and generate a request-specific response packet.

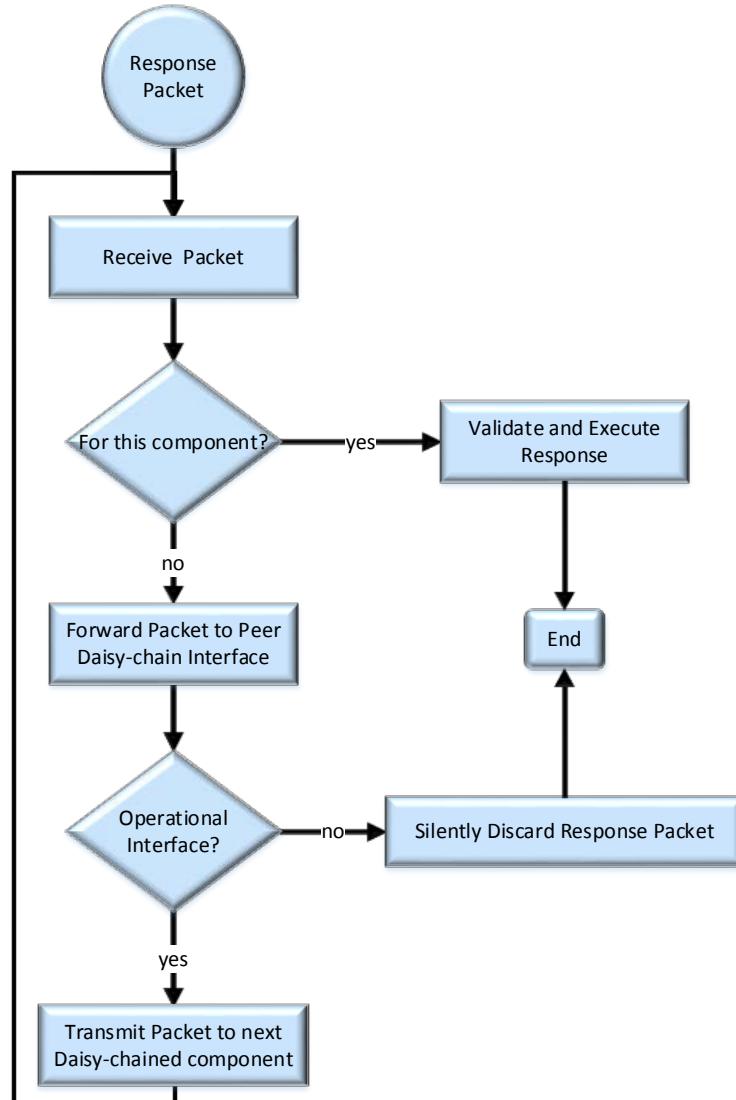


Figure 3-18: Daisy Chain Response Processing Flow

Table 3-2: Daisy Chain Response Processing Details

Step	Explanation
<b>For this component?</b>	Is the component the originating Requester? If not, then forward the packet to the peer daisy-chain interface. If so, then validate the packet and execute the response packet.

Step	Explanation
<b>Operational Interface?</b>	<p>If the components support <i>In-band Management</i> and the component is the Requester at the head of the daisy chain, then consume the received <i>CTL-UE Packet</i> (response packet), else forward the packet to the peer daisy-chain interface.</p> <p>Is the peer configured daisy-chain interface configured and is it in the I-Up state? If not, then silently discard the packet; else, transmit the packet to the next daisy-chained component.</p>

### 3.2.15. Memory Media RAS

Components with addressable memory media may support additional resiliency capabilities to increase data protection and availability. These capabilities fall into the following basic categories:

- Error detection codes are used to detect one or more unexpected bit transitions. Error correction codes are used to restore these transitions to the original values. If successful, these transitions are referred to as correctable errors. If unsuccessful, these transitions are referred to as uncorrectable errors.
- Media scrubbing is used to detect correctable errors and to attempt to correct the errors. There are two types of scrubbing—demand scrubbing is a reactive process when an error is detected during media access, and patrol scrubbing is a background process that reads the media to proactively identify and correct errors (if possible).
- Spare media device provisioning is used to dynamically-substitute a failing or failed media device with another pre-provisioned media device. When a media device has reached a given error threshold, the media device is deemed to be failing. When this occurs, the media controller remaps a spare media device in place of the failing media device, and begins migrating the data to the spare. In contrast, if a media device fails, then depending upon the media controller implementation, the robustness of the error correction capabilities, and possibly the use of erasure codes spanning multiple Responders, a media controller could remap a spare media device, and rebuild the contents of the failed media device.
- In addition to media errors, the infrastructure surrounding the media is subject to faults. Fault management covers a range of media and media infrastructure elements including active and backup power, thermal events, media lifetime endurance events, etc.
- Media initialization covers all aspects of volatile and non-volatile media initialization including the media error detection and correction, media device sparing prior to application use, and sanitation and / or erasure of media contents to prevent data leakage between different applications or customers.

Given the complexity, cost, and potential performance impacts, resiliency requirements, capabilities, and enablement will vary by solution, and even by solution instance. As a result, management is instrumental in delivering working solutions. So, though a media controller abstracts the underlying media, management still needs to comprehend the relevant media attributes and manage the media controller capabilities. A component may surface media and media controller attributes and capabilities through a *Component Media Structure* accessible only to management. Within this structure, there are a number of fields specific to media resiliency, for example:

- Maximum number of errors that the component can detect

- Maximum number of errors that the component can correct
- Correctable and uncorrectable error thresholds
- Poison support and enablement
- Media scrubbing support and configuration
- Spare media device support and configuration
- Row remapping support and configuration
- Media error and event logs

Management uses this structure to determine the component's capabilities, to enable media resiliency services, to monitor media error and events, and to take actions when policies outside of this specification's scope dictate. If the component preserves this structure in non-volatile media, then this structure can play a role in error and failure root cause analysis which is critical to determining manufacturing defect escape rates, identifying component shipping and handling issues, resolving component warranty issues, etc.

### **3.2.15.1. Media Error Detection and Correction**

*Example Read Media Processing Flow* illustrates how media error detection and correction could be done in a component during a media read operation. This example also illustrates how correctable and uncorrectable errors could be handled, and how the events and results could be surfaced to management and the application.

With this example in mind, the following are the media error detection and correction requirements:

- A component that contains addressable media should support the *Component Media Structure*. The *Component Media Structure* is used to surface relevant media attributes and manage media controller capabilities without requiring media-specific knowledge and controls.
- A component shall support power-on self-test (POST) to ascertain media state and to report any defects, failures, etc. Reporting should be through the log entries within the *Component Media Structure* or may be through proprietary means.
- A component may support error detection.
  - Error detection types and methods are outside of this specification's scope. These may be industry standard or proprietary.
  - A component may report the number of errors it can detect per media device.
  - A component shall detect at least the number of errors communicated in the *Component Media Structure*.
- A component may support error correction.
  - Error correction types and methods are outside of this specification's scope. These may be industry standard or proprietary.
  - A component may report the number of errors it can correct per media device.
  - A component shall correct at least the number of errors communicated in the *Component Media Structure*.
- If an uncorrectable error is detected, then the component may poison the error's location.
- Field replaceable memory components should log memory failure events that caused a correctable or uncorrectable error in a non-volatile component-resident resource. This enables subsequent root cause analysis to identify the failed resources to determine patterns and warranty responsibility.
- Erasure codes operating across multiple components are outside of this specification's scope. Such functionality may be used to protect against complete component failure, and to rebuild

data post a component service event. Erasure codes also enable a subset of components that support NVM media to be safely moved from one platform to another, and the application data to be transparently rebuilt.

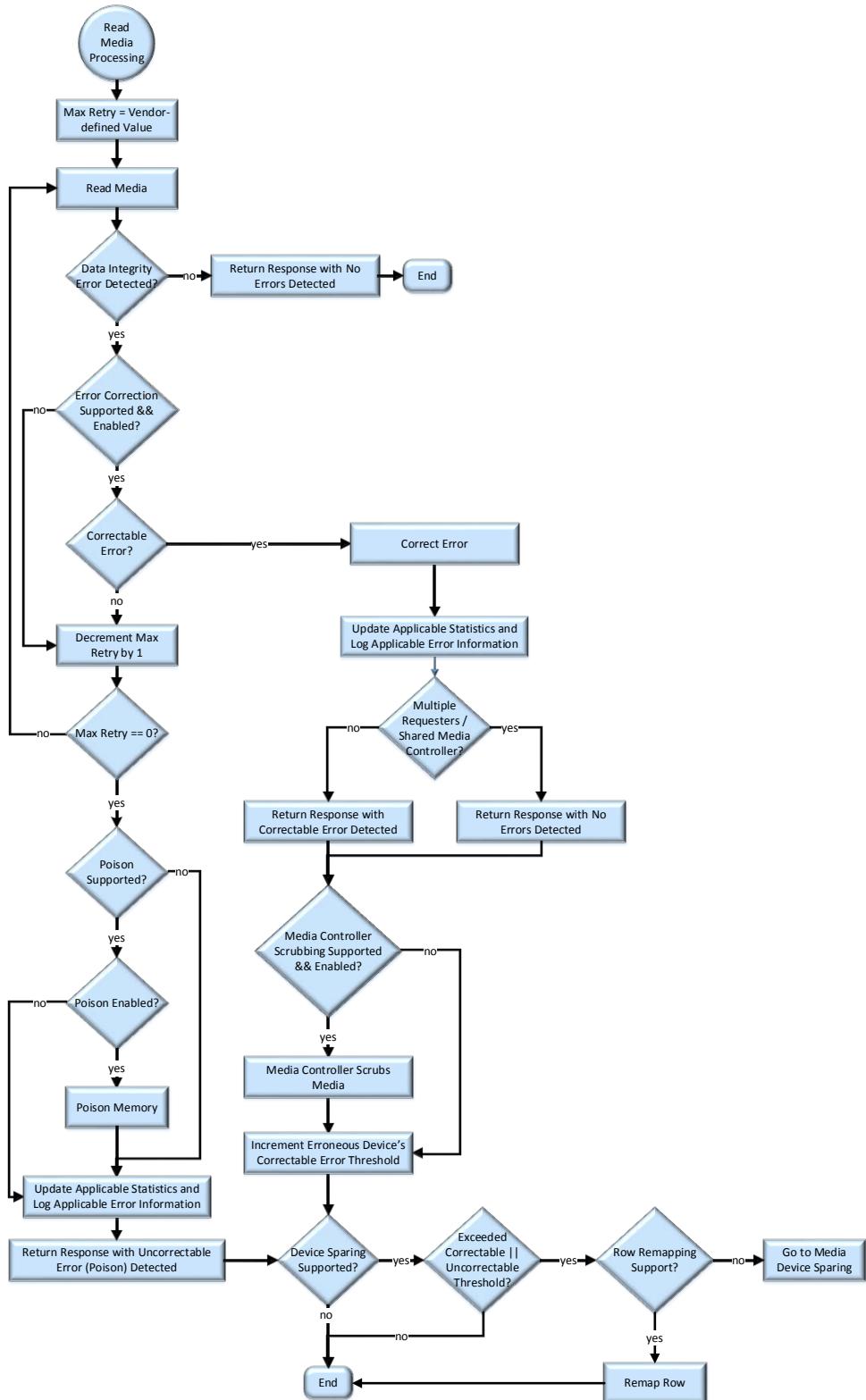


Figure 3-19: Example Read Media Processing Flow

Table 3-3: Example Read Media Processing Details

Step	Explanation
<b>Data Integrity Error Detected?</b>	If a data integrity error was not detected, then return the request-specific response packet with the data.
<b>Error Correction Supported and Enabled?</b>	If media controller error correction is unsupported or not enabled, then decrement the Max Retry counter and reread the media.
<b>Correctable Error?</b>	Is the data integrity error correctable or uncorrectable?
<b>Correctable: Multiple Requesters / Shared Media Controller?</b>	If media is access by multiple requesters or is shared by multiple services, then return the request-specific response packet with no errors detected, else return the response with correctable error detected.
<b>Correctable: Media Controller Scrubbing Supported and Enabled?</b>	If the media controller supports and has enabled demand scrubbing, then scrub the impacted media range.
<b>Uncorrectable: Max Retry == 0?</b>	If the number of read media attempts has not exceeded the maximum number of retries, then read the media again. The maximum number of read media attempts is implementation-specific.
<b>Uncorrectable: Poison Supported?</b>	Is media poison supported?
<b>Uncorrectable: Poison Enabled?</b>	If (Primary Media and <i>Primary Media CAP 1 Control Poison Forwarding Enabled == 1b</i> ) or (Secondary Media and <i>Secondary Media CAP 1 Control Poison Forwarding Enabled == 1b</i> ), then update the applicable statistics, log the error information, and return the request-specific response packet with (Reason = Poison Data (PD) Detected).
<b>(Un-)Correctable: Device Sparing Supported?</b>	If the media controller does not support media device sparing, then take no further actions.
<b>(Un-)Correctable: Exceeded Threshold?</b>	If the number of errors has exceeded the corresponding error threshold, then if row remapping support, then remap the row, else if media device sparing support, then spare the media device.

### 3.2.15.2. Memory Media Scrubbing

Any component type may support memory media scrubbing. Memory media scrubbing is performed using the following basic steps:

1. Reading a memory range
2. Detecting if the memory range has suffered an error
3. Using error correction logic to fix the erroneous bits
4. Writing the corrected data back to the memory range

5. Re-reading the memory range to verify if the error still present.
- If the error is still present and the component provisions spare row remapping resources (aka run-time post-package repair), then the component should substitute a new spare row for the impacted row, and, if possible, then the component should correct and copy the data to the new row.
  - If the error is still present and the component provisions spare media device resources, then the component may perform media device sparing.
  - If the error is still present and current memory resiliency has been reduced to an unacceptable level (e.g., no redundancy is available), then the component may inform management to take additional memory resiliency steps, e.g., evacuating the impacted memory ranges and scheduling a component service event.
  - Though re-reading the memory range may reveal no error, subsequent reads may detect the same error type with the underlying resource. If the same error type is repeatedly detected, then the component may take one of two steps:
    - If the component support row remapping, then the component should perform row remapping on the impacted row(s).
    - If the component provisions spare media device resources, then the component should perform media device sparing on the impacted media device. The repeated error threshold is outside of this specification's scope.

**Developer Note:** *Research has demonstrated that proactive row remapping when a component first detects an uncorrectable error or repeated correctable errors can reduce application downtime and the potential for silent data corruption. With ever-growing memory capacities, developers are strongly encouraged to support row remapping capabilities.*

There are two fundamental types of scrubbing:

- Demand scrubbing when correctable errors are detected during memory media access. *Example Read Media Processing Flow* illustrates a read media processing example where demand scrubbing might be performed.
  - If demand scrubbing is supported, then upon detecting an error, the Responder shall perform demand scrubbing on the impacted memory range transparent to all Requesters.
    - If demand scrubbing detects an excessive number of correctable errors, management should be informed to allow additional resiliency-related actions to be taken.
  - If demand scrubbing is unsupported or disabled and the component is accessed only by a single Requester, then the Responder shall inform the Requester that a correctable error was detected as part of the response packet.
  - If demand scrubbing is unsupported or disabled and the component is accessed by multiple Requesters, then the Responder should inform management. Management is responsible for taking any additional resiliency actions. The Responder shall not inform any individual Requester as part of the response packet.
- Patrol scrubbing is performed as a background activity during idle memory media periods.
  - If supported, a Responder shall perform patrol scrubbing at the configured frequency. The configured frequency is the time required for the entire memory range under the control of a given Responder to be scrubbed.

If uncorrectable errors are detected and the component supports poison, then the memory range shall be poisoned.

- A Responder shall poison a memory range through component-specific methods.
- A Requester may poison a memory range using a *Write Poison*. When a Responder executes a Write Poison request packet, it poisons a memory range using the same component-specific method as it uses when it detects an uncorrectable error and it poisons the impacted memory range.
- If supported, each poison event should be logged, and management should be informed to enable any additional resiliency actions to be taken.
- If an already poisoned memory range is detected during demand or patrol scrubbing, no actions shall be taken.
- Requester behavior when poisoned data is accessed during normal media access, i.e., non-scrubbing operation, is outside of this specification's scope.

**Developer Note:** *Software is responsible for comprehending a Requester's behavior should it access a poison / uncorrectable error memory range. Prior to memory allocation, software examines each Responder's primary media and secondary media (if applicable) log records to identify bad pages, and using policy outside of this specification's scope, it takes steps such as software page remapping to deal with these bad pages. It is critical that Responders provision a sufficient number of log entries to ensure software has a complete picture of the media state.*

### 3.2.15.3. Media Device Sparing

A component may support one or more media devices, which are abstracted through a component-specific mapping service. Abstraction enables a component to provision additional media devices beyond what is required to deliver the component's advertised capacity, e.g., if N media devices are required to deliver a given capacity, then a component may provision N+1, N+2, etc. media devices. These additional media devices may be treated as spares, which may be used to replace media devices which have exceeded their correctable or uncorrectable error thresholds.

Media device sparing uses the following basic steps (see *Example Media Device Sparing Flow*):

1. When a threshold is reached, the media controller stops accessing the bad media device.
2. Media controller maps a spare media device in place of the bad media device.
3. Media controller migrates data from the bad media device to the new media device, and rebuilds any data and restored data integrity using the new media device.

The following are the media device sparing requirements:

- If supported, a component shall support at least one spare media device.
  - A component may support more than one spare media device. The supported number is communicated *Component Media Structure*.
- If a component has completed media device sparing, then it shall retain all such information and state through power-cycle and component reset events.
- Independent of any errors or media device sparing events, if a component was operational prior to a power-cycle or component event, then it should be operational post a power-cycle or component event.
- Memory component performance may degrade during data migration. Once migration is complete, performance should return to its prior level.

- If a component is unable to replace a failing media device (e.g., no available or suitable spare devices), then performance may degrade. The component should inform management of the situation.

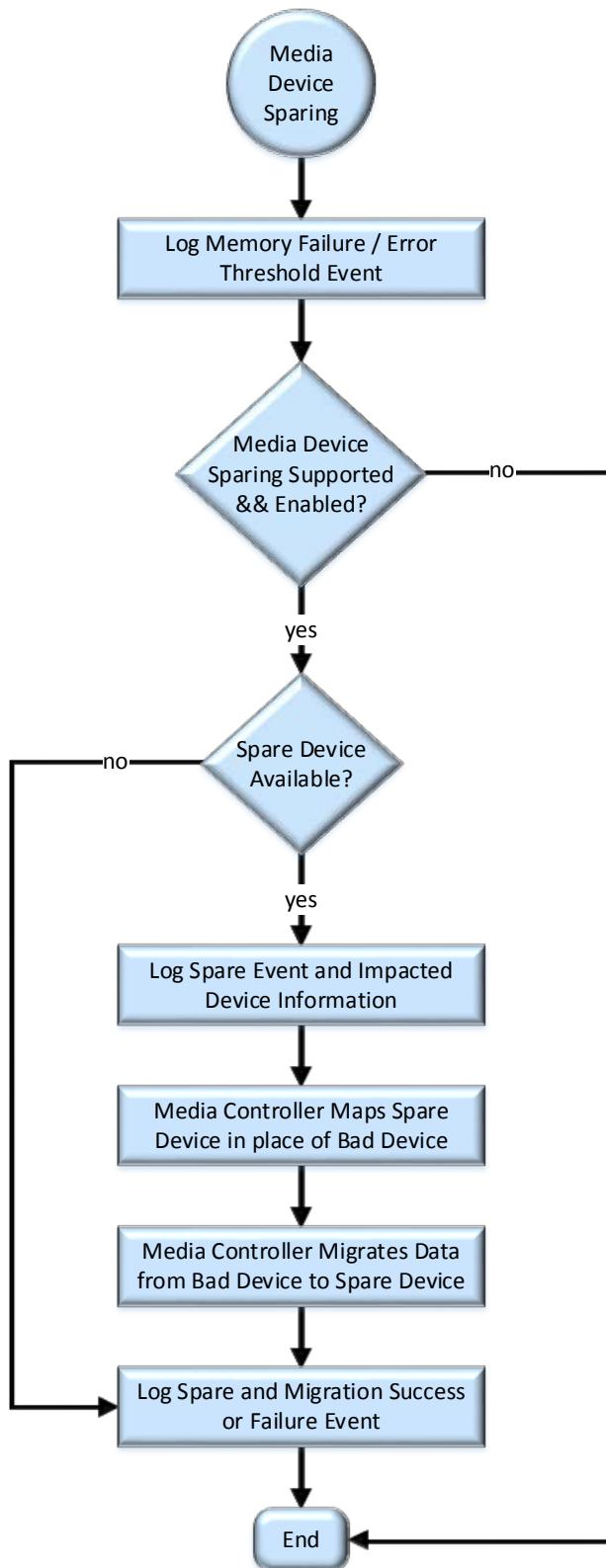


Figure 3-20: Example Media Device Sparing Flow

Table 3-4: Example Media Device Sparing Details

Step	Explanation
<b>Device Sparing Enabled?</b>	If device sparing is not enabled, then take no further actions.
<b>Spare Device Available?</b>	<p>If a spare device is not available, then take no further actions.</p> <p>If available, then:</p> <ol style="list-style-type: none"> <li>1. Log the spare event and the impacted device information.</li> <li>2. Map out the bad device and map in the spare in its place.</li> <li>3. Migrate data from the bad to the spare device. Migration may occur in the background. Until migration is completed, performance may degrade.</li> <li>4. Log success or failure. Failure includes the lack of a spare device.</li> </ol>

### 3.2.15.4. Fault Management

Fault management covers how a component and the underlying media detect, handle, and report faults. Faults may occur for a number of reasons, e.g., due to manufacturing defects, changes in environmental conditions (power, thermal, etc.), degradation or failure of the component logic or media data integrity, and so forth. The *Component Media Structure* is the primary vehicle to configure, manage, and log media and media-related component faults. If the component supports the *Component Error and Signal Event Structure*, then media-related faults and events may be raised to management using one or more the configured methods.

### 3.2.15.5. Media Initialization

10 Outside of vendor quality validation during manufacturing, media initialization within a platform is the first step to determine the media's quality, to identify faults and failures (media device, logic, etc.), and to ensure the media contents are set to what was expected. To assess the media quality, and identify faults and failures, the media initialization process (see *Example Media Initialization Flow*) involves performing similar steps as involved in normal operation media read steps.

15 The primary difference is the “initialize media range” step:

- If the media is volatile, then the initialization typically will be to set the contents of the range to zero.
- If the media is non-volatile, then it depends upon how the media is being used and what has been configured in the *Component Media Structure*.
  - o If the non-volatile media is being used as non-persistent memory, then the contents of the range are set to the configured sanitation and erase level.
  - o If the non-volatile media is being used as persistent memory, and if the stored data is to be preserved, then the initialization may involve reading and validating the memory range using media controller data integrity methods. If the stored data is to be overwritten, then the contents of the range are set to the configured sanitation and erase level.

20

25

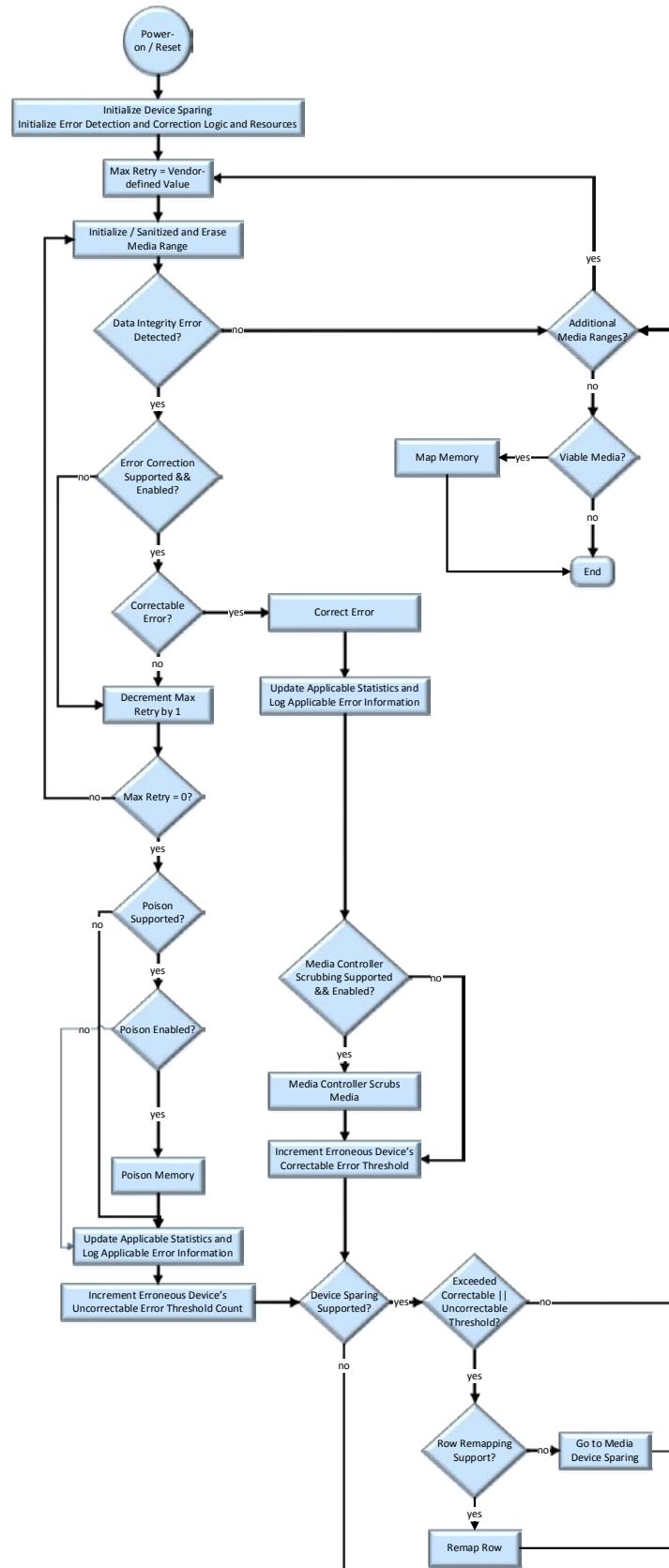


Figure 3-21: Example Media Initialization Flow

Table 3-5: Example Media Initialization Flow Details

Step	Explanation
<b>Initial Device Sparing</b>	If the component supports device sparing, then initialize the logic and associated resources as part of power-on self-test (POST).
<b>Initialize Error Detection and Correction Logic and Resources</b>	If the component supports error detection and correction, then initialize the logic and associated resources as part of POST.
<b>Data Integrity Error Detected?</b>	If a data integrity error was not detected, then continue to the next media range (if available).
<b>Error Correction Supported and Enabled?</b>	If media controller error correction is unsupported or not enabled, then decrement the Max Retry counter and reread the media.
<b>Correctable Error?</b>	Is the data integrity error correctable or uncorrectable?
<b>Correctable: Media Controller Scrubbing Supported and Enabled?</b>	If the media controller supports and has enabled demand scrubbing, then scrub the impacted media range.
<b>Uncorrectable: Poison Supported?</b>	Is media poison supported?
<b>Uncorrectable: Poison Enabled?</b>	Is media poison enabled? If so, update the applicable statistics, log the error information, and
<b>(Un-)Correctable: Device Sparing Supported?</b>	If the media controller does not support media device sparing, then take no further actions, and move to the next media range (if available).
<b>(Un-)Correctable: Exceeded Threshold?</b>	If the number of errors has exceeded the corresponding error threshold and if row remapping is supported, then remap the row, else if media device sparing support, then spare the media device.
<b>Viable Media?</b>	Does the media contain addressable media ranges that can be mapped into a Requester (e.g., a processor)? If yes, then map the memory.

### 3.3. Capabilities-based Resource Access Control

Many security defects and vulnerabilities stem from referencing pointers that are outside the bounds of data buffers. Critical system software is often written in assembly and/or low-level programming languages which lack pointer bounds checking. Moving to a high-level language is a potential solution but is often not practical for reasons such as performance.

‘Fat pointers’, which associate bounds with every pointer and check these bounds on every memory access, can enforce spatial safety and eliminate these violations. However, fat pointers are not extensively used in practice because of overhead.

5 Capability-based addressing, which is also referred to as ‘capabilities,’ extends the fat pointer concept by encoding access rights in the form of ‘handles’ to memory. A handle is a protected (i.e. unforgeable) object created only via the use of a privileged instruction or process such that a system can rely entirely on capabilities to manage memory. Processes can share the same address space as each process can access only those objects it has capabilities to. In a well implemented capability-based system, processes can restrict only access rights in a capability but can never expand them. This makes sharing objects as simple as giving each process a capability to the object.

10 Capabilities can be implemented in software or hardware. Hardware-based capabilities are usually simpler, more secure and better performing. *Example Format of a Hardware-based Capability* illustrates the format of an example hardware-based capability used in the capability hardware enhanced RISC instructions (CHERI<sup>1</sup>) architecture.

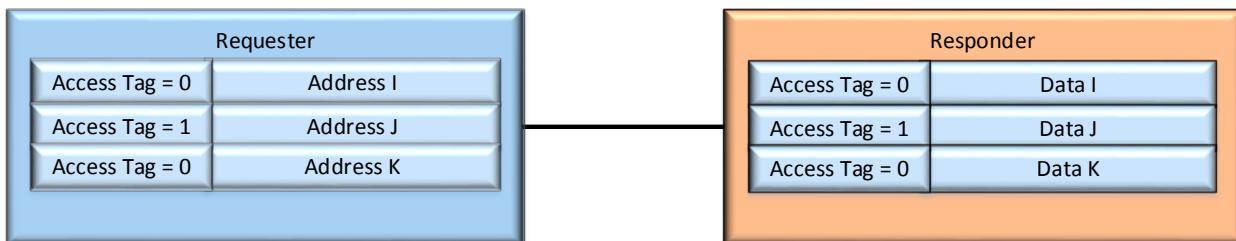


Figure 3-22: Example Format of a Hardware-based Capability

15 In *Example Format of a Hardware-based Capability*, the 64-bit Address field is the virtual address of the object pointed to by the capability while the 64-bit Offset field gives the extent of the object. The 4-bit Permissions field specifies the set of operations that can be performed by capability. Finally, the 1-bit Guard Tag (GT) identifies valid capabilities and is used to prevent the forgery of capabilities. Guard Tags are usually managed by hardware and cannot be directly manipulated by software.

20 In Gen-Z, capabilities-based resource access control uses an access token to determine Requester access permission. Rather than use an explicit token, e.g., a random value associated with a given resource, Gen-Z embodies the token within the access mechanism itself. For example, if a Requester transmits a capabilities request packet with an address associated with a capability-controlled resource, then this indicates that the Requester has access permission to the resource. *Example Requester and Responder with Capability Access Tag* illustrates an example Requester and Responder with capabilities-based resource access control:

- 25
- This example uses a single-bit access tag where 1b indicates capability access control and 0b indicates no capability access control.
  - In this example, a set of Requester addresses I-K have been configured such that address J requires capability access control and addresses I and K do not. Similarly, the Responder has been configured such that data I-K have been configured such that data J requires capability access control and data I and K do not.
  - When the Responder receives a *Capabilities Read* or a *Capabilities Write* request packet, it examines the access tag associated with the data to determine what action to take.



35 Figure 3-23: Example Requester and Responder with Capability Access Tag

<sup>1</sup> See <https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-850.pdf>

The following are the features and requirements associated with capabilities-based access control:

- Any component type may support capabilities-based resource access control.
- If a component supports *Capabilities Read* or a *Capabilities Write* operations, then it shall support both operation types.
- The size and contents of a capability access tag are outside of this specification's scope. Similarly, Requester and Responder management of the access tag is outside of this specification's scope.
- If a Responder receives a Read, Write, Capabilities Read, or Capabilities Write request packet, then the Responder shall take the following actions:
  - If a Read request packet is received, then the Responder shall execute the request independent of the access tag value.
  - If a Write request packet is received, then the Responder shall verify that there is no access tag associated with the targeted resource, or if an access tag is associated, then the access tag indicates the targeted resource is not capabilities data. If capabilities data, then the Responder shall generate a *Standalone Acknowledgment* (Reason = Access Error (AE)—Invalid Capabilities Access).
  - If a *Capabilities Read* request packet is received, then the Responder shall verify that an access tag is associated with the targeted resource and that the access tag indicates the targeted resource is capabilities data, e.g., is set to 1b if using a single-bit access tag. If the access tag is present and indicates the targeted resource is capability data, then the Responder shall execute the request packet and transmit a corresponding Read Response packet. If the access tag is not present or, if present, indicates the targeted resource is not capability data, then the Responder shall generate a *Standalone Acknowledgment* (Reason = Access Error (AE)—Invalid Capabilities Access).
  - If a *Capabilities Write* request packet is received, then the Responder shall verify that there is an access tag associated with the targeted resource. If the access tag is not present, then the Responder shall generate a *Standalone Acknowledgment* (Reason = Access Error (AE)—Invalid Capabilities Access). If the access tag is present, then the Responder shall execute the *Capabilities Write* request packet and update the access tag to indicate this resource contains capability data. The Responder shall ensure updates to the targeted resource and to the access tag are done atomically, i.e., execution of another request packet to the same resource is delayed until the updates to both are successfully completed.
- If a Requester supports *Capabilities Read* and *Capabilities Write* operations, then it shall take the following actions:
  - If a Read Response packet is received in response to a Read request packet, then the Requester shall update the access tag, if present, to indicate this is non-capability data.
  - If a Read Response packet is received in response to a *Capabilities Read* request packet, then the Requester shall update the access tag to indicate this is capability data.
  - When performing a write operation, the Requester shall verify if an access tag is associated with the targeted resource and that the access tag indicates the targeted resource is protected. If protected, then the Requester shall generate a *Capabilities Write* request packet. If the access tag is not present or indicates the targeted resource is not protected, then the Requester shall generate a Write request packet.
  - As with Read and Write operations, a Requester shall be responsible for sequential *Capabilities Read* and *Capabilities Write* request packet execution consistency.

### 3.4. Memory Management

To enable application-transparent access to a Responder's addressable resources, a Requester maps each Responder's addressable resources into the Requester's memory address space. For example, *Example Component with Requester ZMMU* illustrates a Requester (Component 0) and two Responders (components A and B) with addressable resources. In this figure, each Responder's Data Space is partitioned into three memory pages. Each memory page is mapped to a Requester page through the Requester's memory management unit (MMU) or logic that supports equivalent operational semantics. When an application on the Requester allocates memory, the allocated memory is mapped to a series of pages with some pages being mapped to a given Responder's memory page. Once resources are allocated and mapped, an application uses load-store / read-write operations that are transparently translated using the Requester's MMU and translation logic into Gen-Z read and write request packets to access Responder resources.

**Developer Note:** To improve readability and to simplify the concepts, the term ZMMU is used throughout the rest of this material to represent any implementation option that provides the operational memory management semantics described in this specification. Multiple implementations are possible including enhancing a processor MMU. To maximize hardware-software interoperability, if a non-MMU implementation is used, developers are strongly encouraged to support the same page sizes, alignments, page-packing granularity of mixed page sizes, interleave, etc. as the described in this specification.

**Developer Note:** A traditional MMU refers to a page-granular address-indexed lookup table. Depending upon the size of a component's memory address space, a ZMMU implementation can use a sparse in-DRAM page table and an on-chip cache. To avoid forward progress dependencies / prevent deadlock, all MMU resources are component-local.

**Developer Note:** If a component will contain an MMU, then to deliver optimal performance, developers are strongly encouraged to integrate Requester ZMMU and Responder ZMMU logic and capabilities within the component's MMU.

**Developer Note:** Point-to-point or daisy-chain-attached memory management used to support P2P-Core OpClass or P2P-Coherency OpClass-based solutions relies exclusively upon Requester-specific decoding and interleaving which are outside of this specification's scope.

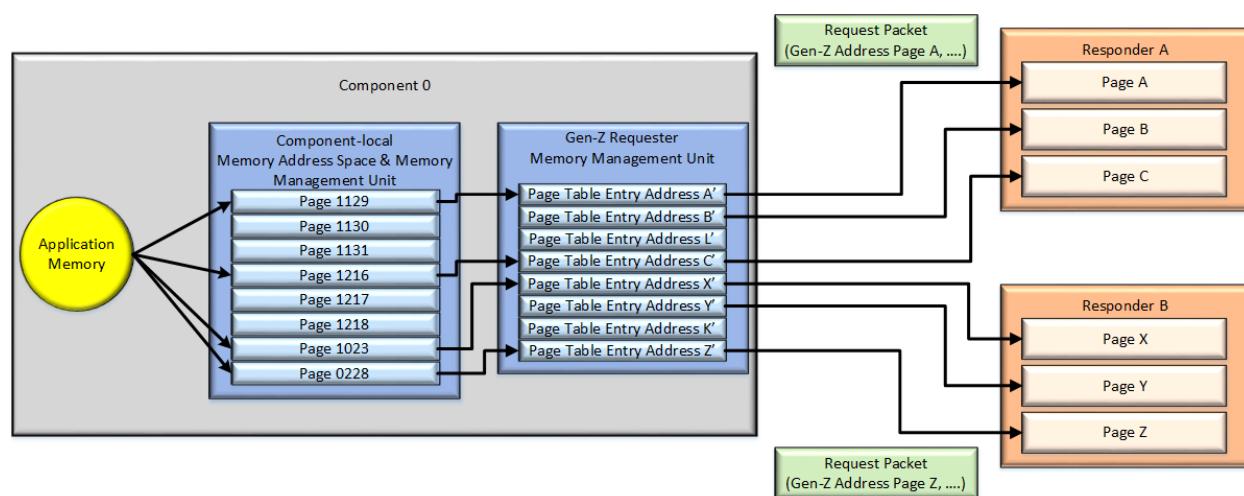


Figure 3-24: Example Component with Requester ZMMU

Key aspects to note regarding a Requester ZMMU and its operation:

- A Requester ZMMU is configured by trusted software.
- A Requester ZMMU maps the addressable resources, e.g., DRAM or byte-addressable NVM, of one or more Responders into the Requester's local memory address space.
- A Requester ZMMU maps a Responder's Data Space.
- If *In-band Management* is supported, a Requester ZMMU maps a Responder's Control Space.
- If supported, a Requester ZMMU maps addresses associated with advanced operations, services, or resources of one or more Responders into the Requester's local memory address space. For example, a Requester ZMMU may facilitate buffer operations, unicast and multicast messages, etc.
- The size or scale of a Requester ZMMU will vary depending upon the size and amount of the Requester's local memory space that can be mapped to Responders.
- The Requester's local memory address space is accessed through a Requester-specific MMU (or equivalent). The Requester-specific MMU surfaces local memory addresses as virtual memory addresses to applications.
- If a Requester supports *Region Key (R-Key)*, then each Requester ZMMU leaf PTE contains a single R-Key. The R-Key will be the read-only R-Key or the read-write R-Key of the mapped Responder resource that corresponds to the Requester's access permission.
- Each Requester ZMMU leaf PTE contains additional elements and page attributes used to construct request packets and fill in request-specific protocol fields.
- Based on policies outside of this specification's scope, when applications allocate memory, trusted software allocates memory that maps Requester-local memory address space to Responder-specific addressable resources. Once allocated, the application transparently accesses the memory using load-store / read-write semantics to the allocated Requester-local memory address space. In *Example Component with Requester ZMMU*, the application issues a store operation to an address that resolves to page 1129. The component-specific MMU determines that the address is associated with a Gen-Z-accessed resource, and initiates a Requester ZMMU hardware table walk to locate the associated leaf Page Table Entry (PTE). When the Requester ZMMU leaf PTE is located, the Requester uses the PTE's contents to identify the Responder component and to translate the application's address into a Responder-specific address. Using the PTE contents, the operation type, and the Responder-specific information derived from other Gen-Z configuration structures, etc., the Requester creates and transmits a request packet to the Responder.

Depending upon the component type and the robustness of its memory management, a Responder may contain a Responder ZMMU. A Responder ZMMU translates a request packet's Address into a Responder-local page that is mapped to a Responder-local resource. *Example Component with Responder ZMMU* illustrates a Responder that contains a ZMMU used to access multiple addressable pages.

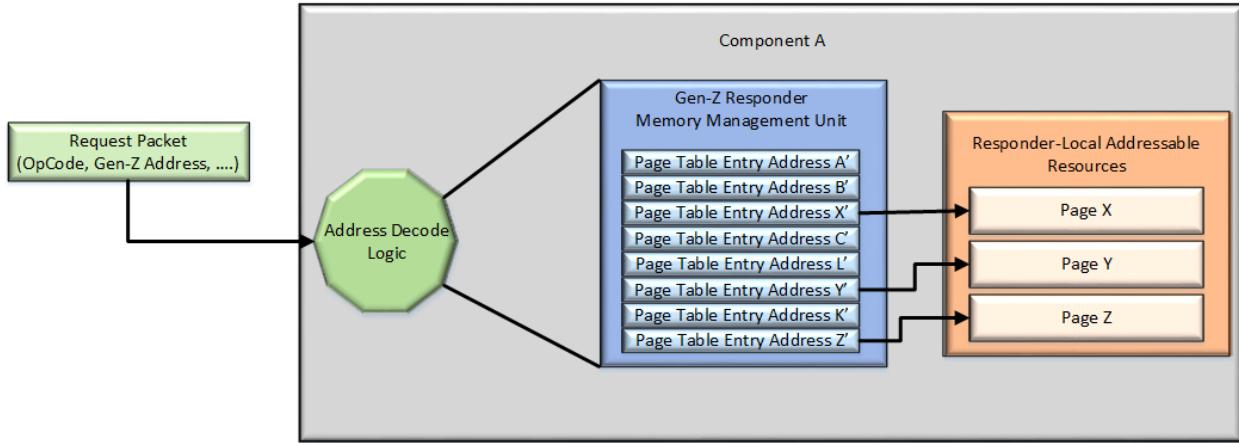


Figure 3-25: Example Component with Responder ZMMU

Key aspects to note regarding a Responder ZMMU and its operation:

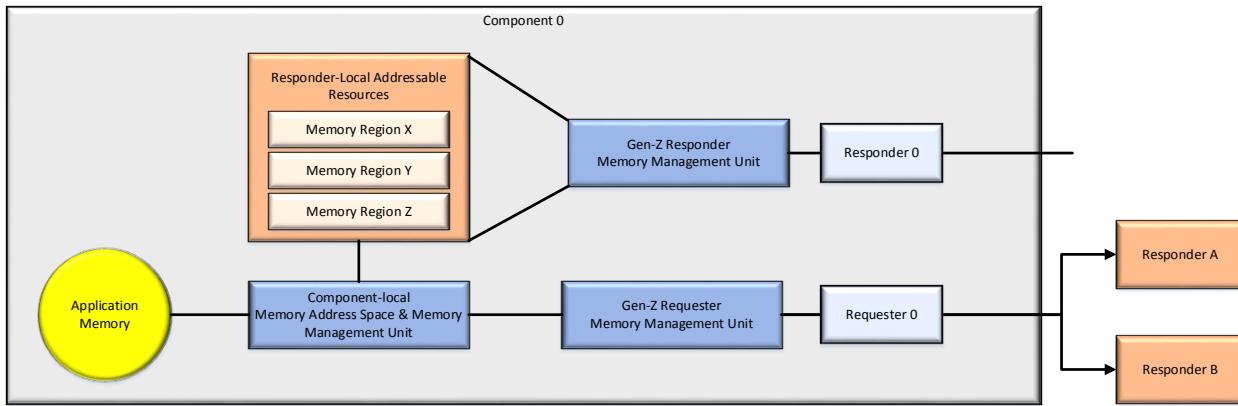
- A Responder ZMMU is configured by trusted software.
- A Responder ZMMU is used to translate a request packet's Address field into a Responder-local address. This Responder-local address is used to access the target resource. For example, in *Example Component with Requester ZMMU*, the Requester maps page 1023 to Responder memory page X'. The Requester ZMMU is used to translate and construct the request packet's Address field to target X'. Upon receipt, the Responder decodes the request packet's Address and initiates a Responder ZMMU hardware table walk to locate the associated Responder leaf PTE. When the Responder leaf PTE is located, the Responder uses the information within the PTE to identify the Responder-local resources, validate access permission, and execute the request packet.
- A Responder may support multiple types of Responder-local resources, e.g.,
  - A physical resource such as memory or a cache
  - An address range used to exchange memory-semantic messages between any component type
  - An address range used to explicitly or transparently invoke accelerator operations
  - And so forth.
- The size or scale of a Responder ZMMU will vary depending upon the size and amount of the Responder-local addressable resources that can be mapped to Requesters. In many cases, the amount of provisioned addressable resources will be a fraction of a Requester's memory address space, e.g., a Responder might contain multi-TiBs of memory capacity and a Requester might support a multi-PiB memory space.
- If a Responder supports *Region Key (R-Key)*, then each leaf PTE contains two R-Keys used to control access to the associated page—a read-only R-Key and a read-write R-Key.
- Each leaf PTE contains additional elements and page attributes used to validate and execute the request packet.

**Developer Note:** Given the operational similarities between a Requester ZMMU and a Responder ZMMU, much of the underlying ZMMU IP can be re-used to implement Requester and Responder ZMMUs. For example:

- The Responder's address decoding logic / Responder ZMMU interprets incoming request packet addresses (Responder-local application address or request packet Address field) as sequences of

5 *pages based on high-order bits. Each page represents a separate decoded region with potentially different page attributes.*

- *Requester-local and Responder-local addresses that correspond to the same page share the same page attributes.*
- *Requester ZMMU and Responder ZMMU page table layout designs are common in terms of page size, alignment, and page-packing restrictions. However, a Requester leaf PTE will have a different format and capabilities than a Responder leaf PTE. Further, Requester leaf PTE format will differ based on interleaved and non-interleaved support.*
- *The Requester and Responder ZMMUs in communicating components need to support mutually-compatible page sizes.*
- *The Requester and Responder ZMMUs may be sparse, i.e., storing address decode information only for accessible memory pages.*
- A Requester-Responder component may contain a Requester ZMMU and a Responder ZMMU.



15 Figure 3-26: Example Component with a Requester and a Responder ZMMU

Developer Note: Though the underlying ZMMU IP can be re-used, if a component supports a Requester ZMMU and a Responder ZMMU, then the component needs to implement distinct and independent logic and resources. This is necessary because Requester and Responder ZMMUs operate independently and concurrently, because Requester and Responder ZMMUs operate on non-equivalent input address spaces, because the leaf PTE formats differ, and because the trusted software used to manage each ZMMU could differ or require a unique execution environment with different access permission.

Developer Note: Responders will vary in their capabilities and needs. Though possibly less flexible, an alternative to a Responder ZMMU is to mask the request packet's Address field and decode the results to one of the supported PTE sizes (to improve software interoperability, the Responder leaf PTE format can still be used).

To improve scalability and performance, a component may contain multiple Requester and / or Responder protocol engine instances, e.g., one instance per component interface (an instance is an implementation-specific set of logic and associated resources that are outside of this specification's scope). As *Example Component with Multiple Requester and Responder Instances* illustrates multiple Requester instances share the same Requester ZMMU, which enables any Requester instance to generate a request packet on behalf of the component. Similarly, multiple Responder instances share the same Responder ZMMU, which enables any Responder instance to validate and execute any request packet and generate a response packet (if required).

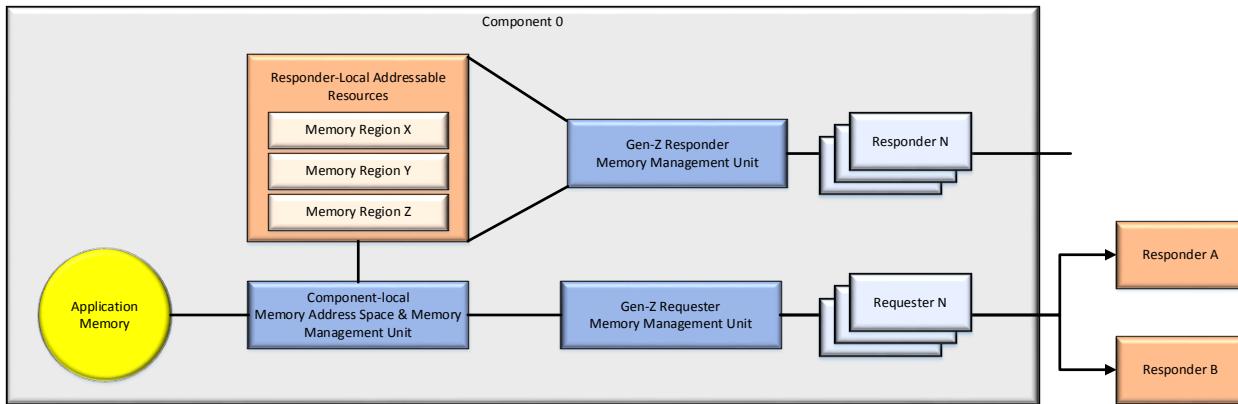


Figure 3-27: Example Component with Multiple Requester and Responder Instances

A component may support a vendor-specific implementation of a ZMMU or its equivalent. If the component supports a vendor-specific implementation, then to ensure software-hardware interoperability, at a minimum, the component shall support at least the same minimum set of page sizes, and page-packing restrictions shall not be any more stringent than as specified below. Vendor-specific implementations may support additional page sizes, support a different hierarchical page table layout, support different caching characteristics, support on-chip, SRAM-only page table, etc.

A component may support a Requester and / or a Responder ZMMU as specified in this document.

#### 3.4.1. ZMMU Page Table Structure

The ZMMU page table structure is used for both Requester and Responder ZMMUs. As such, Requester and Responder ZMMUs support common page sizes, alignment and packing restrictions. Where they differ is the leaf nodes, referred to as Page Table Entries (PTE). A Requester PTE contains fields necessary to generate a request packet to one or more Responders, and a Responder PTE contains fields necessary to validate and execute a request packet relative to Responder-specific resources.

*ZMMU Page Table Structure* illustrates the conceptual structure of a ZMMU page table layout in DRAM. Multiple levels or tiers of sub-tables form a hierarchy. A page table walk follows a series of one or more pointers to a leaf PTE which describes the actual address mappings. Each level in a page table walk uses a different subset of the address bits to index into the corresponding sub-table. Each sub-table entry contains a pointer to the next sub-table or a leaf PTE.

- If a component supports a Requester ZMMU and a Responder ZMMU, then the two ZMMUs shall be independent, i.e., two separately provisioned and managed page tables.
- To improve interoperability, ZMMU and non-ZMMU address decoding schemes shall support at least the following page sizes:
  - 32 MiB, naturally-aligned with the base address that is a 32 MiB multiple
  - 1 MiB, naturally-aligned with the base address that is a 1 MiB multiple
  - 64 KiB, naturally-aligned with the base address that is a 64 KiB multiple
  - 4 KiB, naturally-aligned with the base address that is a 4 KiB multiple
- For page sizes smaller than 32 MiB, each naturally-aligned 16 MiB region shall support a single page size.
- The page table and its sub-tables are component-specific resources that are accessed through the component's local address space, i.e., these resources are provisioned and accessed independent of Gen-Z.

- The Base Address CSR is a component-specific register used to locate the page table.
- Pointers to sub-tables within the page table are stored as component-local addresses.
- Pointer widths support 64-bit addresses. Unused high-order bits may be ignored.
- The Input Address into a Requester ZMMU is the Requester-local address.
- The Input Address into a Responder ZMMU is the contents of the request packet's Address field.
- A subset of the Input Address bits is defined as an address range. In the following descriptions, an address range may be in one of two formats:
  - MSB:LSB, e.g., 32:12. This format points to a 128-bit table entry.
  - If the table entry is a pointer-pair (see *Pointer-Pair Table Entry Format*), then the format is extended to be MSB:LSB, S where S == 0b indicates Pointer 0 and S == 1b indicates Pointer 1 shall be used to continue to the table walk. For example, a third-level subtable is indexed by address bits A[38:25]. If the table entry is a pointer-pair, then A[24] == 0b indicates Pointer 0 and A[24] == 1b indicates Pointer 1 is used.
- An implementation performs a 128-bit read at the address indexed by bits MSB:LSB to determine whether the entry is valid and what type of entry—PTE or a pair of pointers.
  - If ET == 1b, then this is a PTE entry.
  - If the ET, PT0, and PT1 fields are set to 0x0, then this is an invalid entry.
- The page table structure is hierarchical:
  - A first-level subtable is indexed by:
    - Bits 63:53, S[52] which produces two 64-bit pointers where S[52] determines which pointer to follow to the base address of a second-level sub-table.
  - A second-level subtable is indexed by:
    - Bits 51:39 point to a PTE describing a 512 GiB page
    - Bits 51:39, S[38] which produces two 64-bit pointers where S[38] determines which pointer to follow to the base address of a third-level sub-table.
  - A third-level subtable is indexed by:
    - Bits 37:25 point to a PTE describing a 32 MiB page
    - Bits 37:25, S[24] which produces two 64-bit pointers where S[24] determines which pointer to follow to the base address of a fourth-level sub-table. Each pointer is associated with a metadata describing the page size for the 16 MiB region. Each 16 MiB region shall contain only a single page size, i.e., no intermixing of finer granularity pages sizes.
  - A fourth-level subtable is indexed by one of three sets of bits which point to a PTE entry:
    - If the PTE points to a 1 MiB page, then Bits 23:20 are used to index the table.
    - If the PTE points to a 64 KiB page, then Bits 23:16 are used to index the table.
    - If the PTE points to a 4 KiB page, then Bits 23:12 are used to index the table.
    - All PTE entries within a given fourth-level table shall describe pages of identical size.
  - A first-level subtable may contain fewer entries than the subsequent tables. If present, a first-level subtable should be structured similarly to a second-level subtable.
    - Table Address[63:0] = {CSR[63:20], 3'b0, Input Address[63:53], 7'b0}
    - A first-level subtable shall contain only pointer-pairs.
  - A second-level subtable should contain 8192 table entries. A second-level table may contain an arbitrary mix of pointer-pairs and PTEs. When dereferencing a pointer from a first-level subtable (or from the Base Address CSR if the first-level subtable is unsupported):
    - Sub-Table Address[63:0] = {Pointer[63:20], Input Address[51:39], 7'b0}

- If the pointer points to a PTE and the PTE is cached, then the cache tags shall be encoded in such a way that Input Address[38:12] are treated as "don't care" bits in subsequent cache lookups (see *ZMMU PTE Caching*)
- A third-level subtable should contain 8192 table entries. A third-level table may contain an arbitrary mix of pointer-pairs and PTEs. When dereferencing a pointer from a second-level subtable:
  - Sub-Table Address[63:0] = {Pointer[63:20], Input Address[37:25], 7'b0}
  - Pointer[19:11] shall be ignored.
  - If PA points to a PTE and the PTE is cached, then the cache tags shall be encoded in such a way that Input Address[24:12] are treated as "don't care" bits in subsequent cache lookups (see *ZMMU PTE Caching*).
- A fourth-level subtable shall contain only PTEs. The number of PTE table entries depends upon the page size.
  - If the page size is 4 KiB, then there are 4096 table entries. When dereferencing a pointer from a third-level subtable:
    - PTE address[63:0] = {Pointer[63:19], Input Address[23:12], 7'b0}
    - Pointer[18:11] shall be ignored.
  - If the page size is 64 KiB, then there are 256 table entries. When dereferencing a pointer from a third-level subtable:
    - PTE address[63:0] = {Pointer[63:15], Input Address[23:16], 7'b0}
    - Pointer[14:11] shall be ignored.
    - Input Address[15:12] shall be ignored.
    - If the resulting PTE is cached, then the cache tags shall be encoded in such a way that Input Address[15:12] are treated as "don't care" bits in subsequent cache lookups (see *ZMMU PTE Caching*).
  - If the page size is 1 MiB, then there are 16 table entries. When dereferencing a pointer from a third-level subtable:
    - PTE address[63:0] = {Pointer[63:11], Input Address[23:20], 7'b0}
    - Input Address[19:12] shall be ignored.
    - If the resulting PTE is cached, then the cache tags shall be encoded in such a way that Input Address[19:12] are treated as "don't care" bits in subsequent cache lookups (see *ZMMU PTE Caching*).
- The described page table walk may be reduced in size. For example, the first-level lookup may be omitted in components that support an input address that is less than or equal to 52 bits. In this case, there is only one second-tier table, and the Base Address CSR is a component-specific register used to locate the second-tier page table. Or, one or more table tiers may be implemented as a static table lookup using on-chip SRAM, relying on in-DRAM tables only for the larger, subsequent table entries.

In *ZMMU Page Table Structure*, the yellow boxes represent PTEs. These are 128-bit descriptors of the memory page they represent. The orange boxes illustrate the sequence of pointers and eventual PTE traversed during a page walk to locate a PTE for a 4 KiB page in a 64-bit input address space.

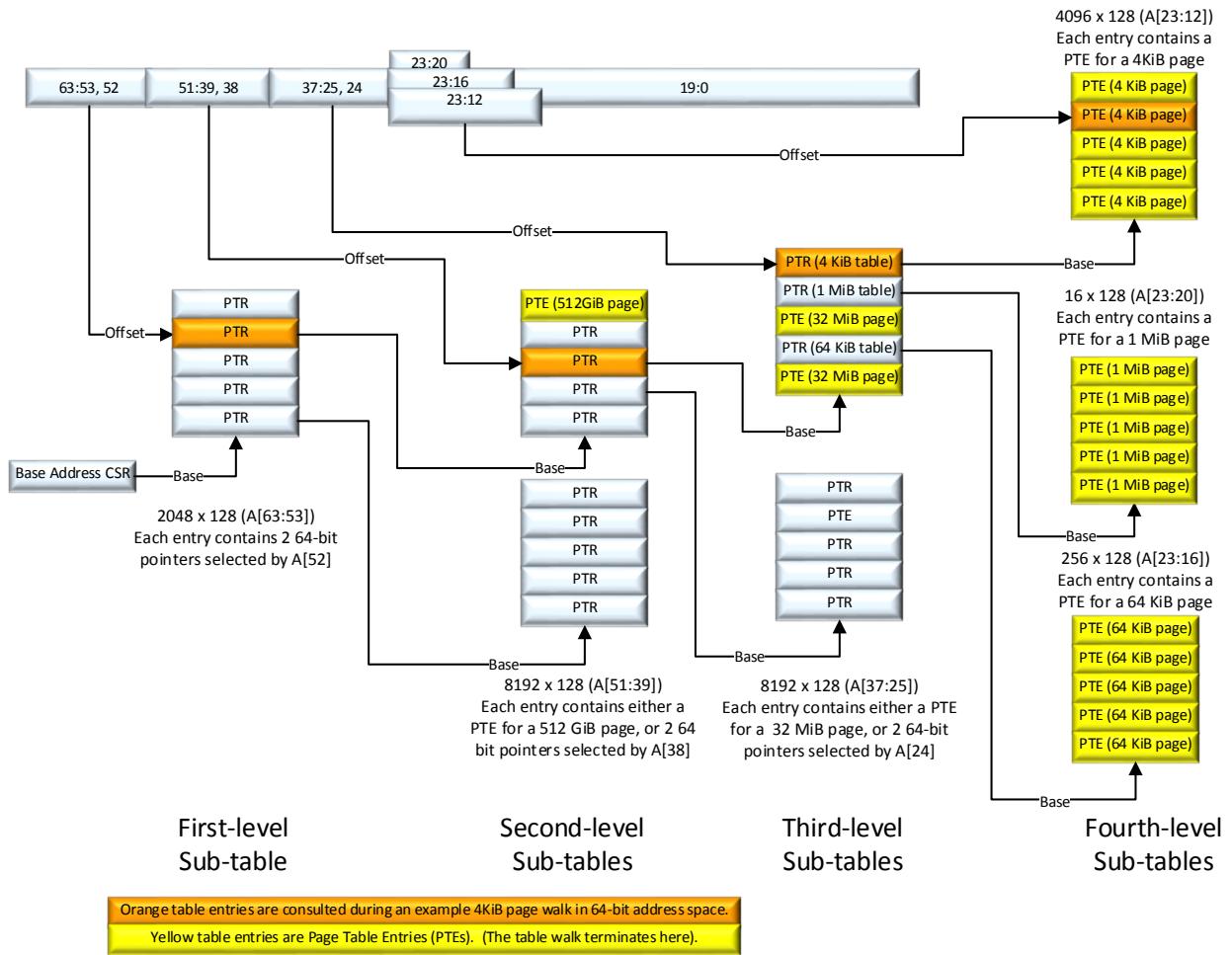


Figure 3-28: ZMMU Page Table Structure

Figure 3-29 illustrates a pointer-pair table entry. Each pointer represents an effective 64-bit pointer, with implicit bits [10:0] equal to 0x0.

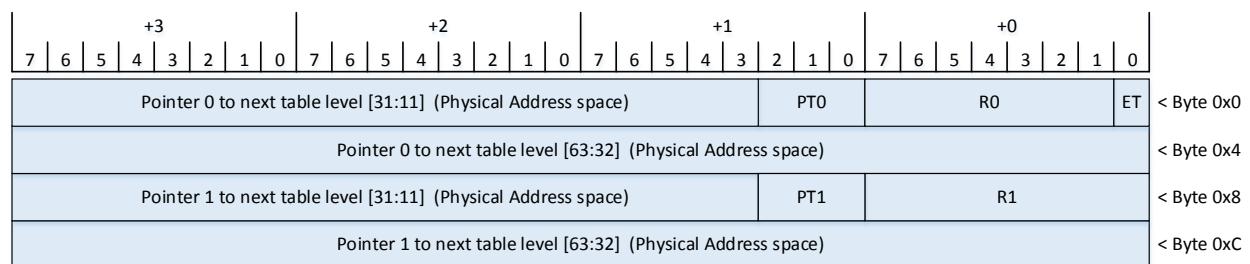


Figure 3-29: Pointer-Pair Table Entry Format

Table 3-6: Pointer-Pair Table Entry Fields

Field Name	Size (bits)	Description
ET	1	Entry Type 0b—Pointer-pair 1b—PTE

Field Name	Size (bits)	Description
PT0, PT1		If ET, PT0, and PT1 are equal to 0x0, then this indicates an invalid table entry. Invalid table entries may be inserted in any table tier where sparse population is desired.
	3	Pointer Type 0x0—Invalid pointer 0x1—Points to next-level table. Used only in first and second-level tables 0x2—Points to a 4096-entry table of PTEs describing 4 KiB pages. Pointer[18:11] shall be 0x0. Used only in third-level tables. 0x3—Points to a 256-entry table of PTEs describing 64 KiB pages. Pointer[14:11] shall be 0x0. Used only in third-level tables. 0x4—Points to a 16-entry table of PTEs describing 1 MiB pages. Used only in third-level tables. 0x5-0x7—Reserved
Pointer 0	53	If valid, points to subtable or a PTE
Pointer 1	53	If valid, points to subtable or a PTE
R0	7	Reserved
R1	8	Reserved

### 3.4.2. ZMMU PTE Caching

To mitigate the latency associated with PTE lookup, a ZMMU may contain an on-chip cache of recently-accessed PTEs. To ensure that a given input address matches the correct cache entry without a priori knowledge of which page size might match, cache tagging and indexing need to take into account the pattern of “don’t care” address bits. This requires a ternary-CAM matching scheme, where certain tag bits can match a 0, a 1, or an X (don’t care). The X state is associated with those bits not participating in the match (e.g., Address[19:12] in a cache entry representing a 1 MiB page). Each cache stored tag contains an encoded page size to determine the pattern of X-bits to apply. In a fully-associative cache, this ternary behavior is sufficient to handle mixed page-size matches. In a set-associative cache, it is also necessary to exclude any bits that may contain X’s from the cache index (typically a hash function), and to increase the number of ways to compensate for any increase in index collisions.

Certain bits within A[45:12] are AND-masked to zero before accessing media for an interleaved region. Masking is applied only to the underlying media access, and not to the page table walk, the PTE cache index hash, or the stored CTAGs. The Responder ZMMU acts upon the pre-masked address to ensure that Responder R-Key validation is applied to pages sharing common boundaries with the Requester’s R-Key understanding.

**Developer Note:** A set associative cache for solutions that support very large page sizes (e.g., 512 GiB) require very large ways to avoid thrashing. For such solutions, developers are encouraged to support a

fully-associative caching or a series of independent caches (one per supported page size) and set-associative caching.

### 3.4.3. Memory Interleave

A non-interleaved memory page associates a single Responder with all of the addresses corresponding to a single contiguous page. Non-interleaved regions are used by unicast and multicast messaging applications, by interrupt services, by management services to map Control Space, by memory and storage applications that do not benefit from interleaving, and so forth.

An interleaved memory page associates multiple Responders with the addresses corresponding to a single contiguous page. Interleaved regions are used to increase performance (lower effective latency and increase aggregate bandwidth). For example, *Matching a High-bandwidth Requester with Multiple Low-bandwidth Responders* illustrates a high-bandwidth Requester that communicates with multiple low-bandwidth Responders. Interleaving request packets across multiple Responders ensures even spatially-local access patterns are effectively distributed.

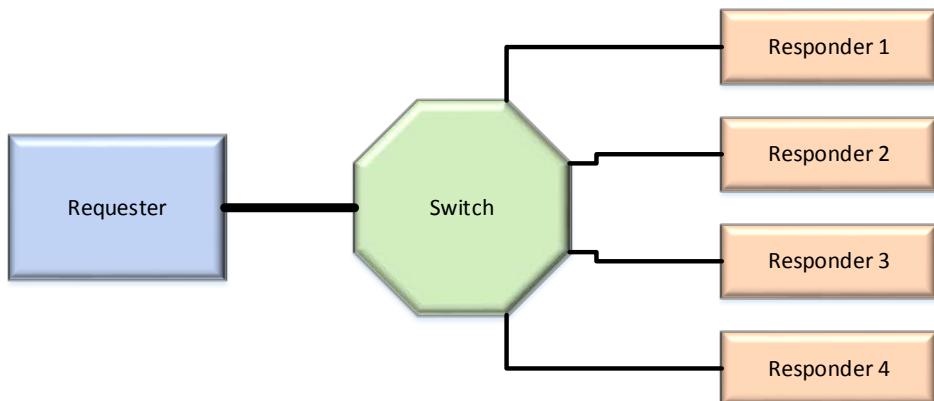


Figure 3-30: Matching a High-bandwidth Requester with Multiple Low-bandwidth Responders

When address decode indicates that an interleave group is being accessed, the address decode selects from a set of Responder CIDs stored in the interleave table. Gen-Z uses a stacked barber pole interleave scheme to enable an arbitrary number of Responders to participate in a given interleave group. An N-way stacked barber pole maps sequential addresses across N Responders in such a way as to balance access bandwidth across all Responders even in the presence of spatially-local address patterns (e.g., sequential address). A stacked barber pole uses modulo-N arithmetic on the input address to select the Responder. Modulo-N arithmetic uses a subset of the low-order address bits instead of the entire address. The resulting interleave occupies a set of N discrete contiguous regions in the input address space. Though each region matches the size of a single participating Responder, the region is interleaved across all participating Responders.

If supported, only a Requester may support an Interleave Table:

- Each Interleave Table entry describes a single interleave group.
- The number of Interleave Table entries is implementation-specific.
- ILTE represents an index into the interleave table and identifies a single interleave group.
- Each Interleave Table entry consists of the following fields:
  - A 6-bit *max\_way* field, where (*max\_way* + 1) is the maximum number of Responders that may participate in an interleave group.
  - A *valid* bit determines if the table entry is valid.

- An ordered list of Responder CIDs. Valid entries are 0 to *max\_way*.
- A *Responder SID*. All Responders shall be located within the same subnet, i.e., multi-subnet interleave groups are unsupported.
- A 16-bit *module\_size* field which indicates the size in GiB of the interleave range provisioned per Responder.
- A 2-bit *interleave\_granularity* field:
  - 0x0—64-byte interleave granule
  - 0x1—256-byte interleave granule
  - 0x2-0x3—Reserved

10 **Developer Note:** Though the number of Interleave Table entries is implementation-specific, if supported, developers are strongly encouraged to support at least 128 entries.

If the Interleave Table is supported, then the Requester shall perform the interleave calculation as follows:

```

15 // ILTE is the interleave-table index (e.g., from the Requester's ZMMU PTE)
  valid = intlv_table[ILTE].valid;
  if (valid == 0) error ("Invalid interleave table entry");

20 ways = intlv_table[ILTE].max_way + 1;
  module_size = intlv_table[ILTE].size;
  nb_size = clogb2 (module_size) + 30      // +30 because module_size is in GiB
  nb_intlv_hi = clog2 (max_way + 1);        //clog2(N) = number address bits for memory size (N)
  nb_intlv_lo = 4;
  LZA = Requester_ZMMU_translated (input_address);

25 // Calculate the target address using the interleave table entry
  intlv_hi_mask = (1LL << nb_intlv_hi) - 1;
  intlv_hi = (LZA >> nb_size) & intlv_hi_mask;
  intlv_lo_mask = (1LL << nb_intlv_lo) - 1;
  intlv_lo = (LZA >> 6) & intlv_lo_mask;
  cid_index = (intlv_hi + intlv_lo) % ways;
  Responder_CID = intlv_table[ILTE].dcid[cid_index];
  Responder_SID = intlv_table[ILTE].dsid;
  Req_Addr_Mask = (1LL << (nb_size + nb_intlv_hi)) - 1;
  Target_Address = LZA & Req_Addr_Mask;

```

35 *Example 3-way Stack Barber Pole Interleave* illustrates a 3-way interleave. The left-hand column represents the output address from the Requester ZMMU. The Requester ZMMU page mappings are set to create a set of contiguous regions whose number is equal to the number of Responders participating in the interleave group and whose size represents the size of the contributed Responder addressable region. Each page mapping begins on the next available power-of-two-aligned starting address. In this example, there are three Responders, each contributing 1.5 TiB of addressable memory to the interleave group. The ZMMU maps the region into three 1.5 TiB regions with one at address 0, one at address 2 TiB, and one at address 4 TiB.

40 The middle three columns represent the three Responder address maps. Address maps preserve page boundaries enabling Responders that support a ZMMU to perform R-Key validation. Each Responder

contributes one third of the media to each of the three regions, and the request packets are distributed across all three. Each colored box represents a single interleave granule, where the size of each granule is either 64 bytes or 256 bytes depending upon the interleave granularity setting in the interleave table. In this example, if 64-byte granule is used, then accesses to address 0 go to Responder 0, accesses to address 0x40 go to Responder 1, accesses to address 0x80 go to Responder 2, and so forth. For illustrative purposes, each granule is labeled as follows: X..Y where X identifies the interleave range number and Y identifies the offset within the range.

The right three columns illustrate how each Responder may “compress out” sparseness by packing the regions into contiguous Responder-local addresses. This is accomplished by masking the high-order address bits in a programmable MSB:LSB range to zero (this impacts only Responders participating in a given interleave group). If a Responder supports a Responder ZMMU (e.g., to perform R-Key validation), then the masking shall be done subsequent to ZMMU lookup.

The following pseudo code illustrates the bits that need to be masked in the Responder as a function of the interleave-related constants derived in the Requester pseudo code:

```
15      Rsp_Addr_Mask = (1LL << nb_size) - 1;  
      Responder-local address = Address && Rsp_Addr_Mask;
```

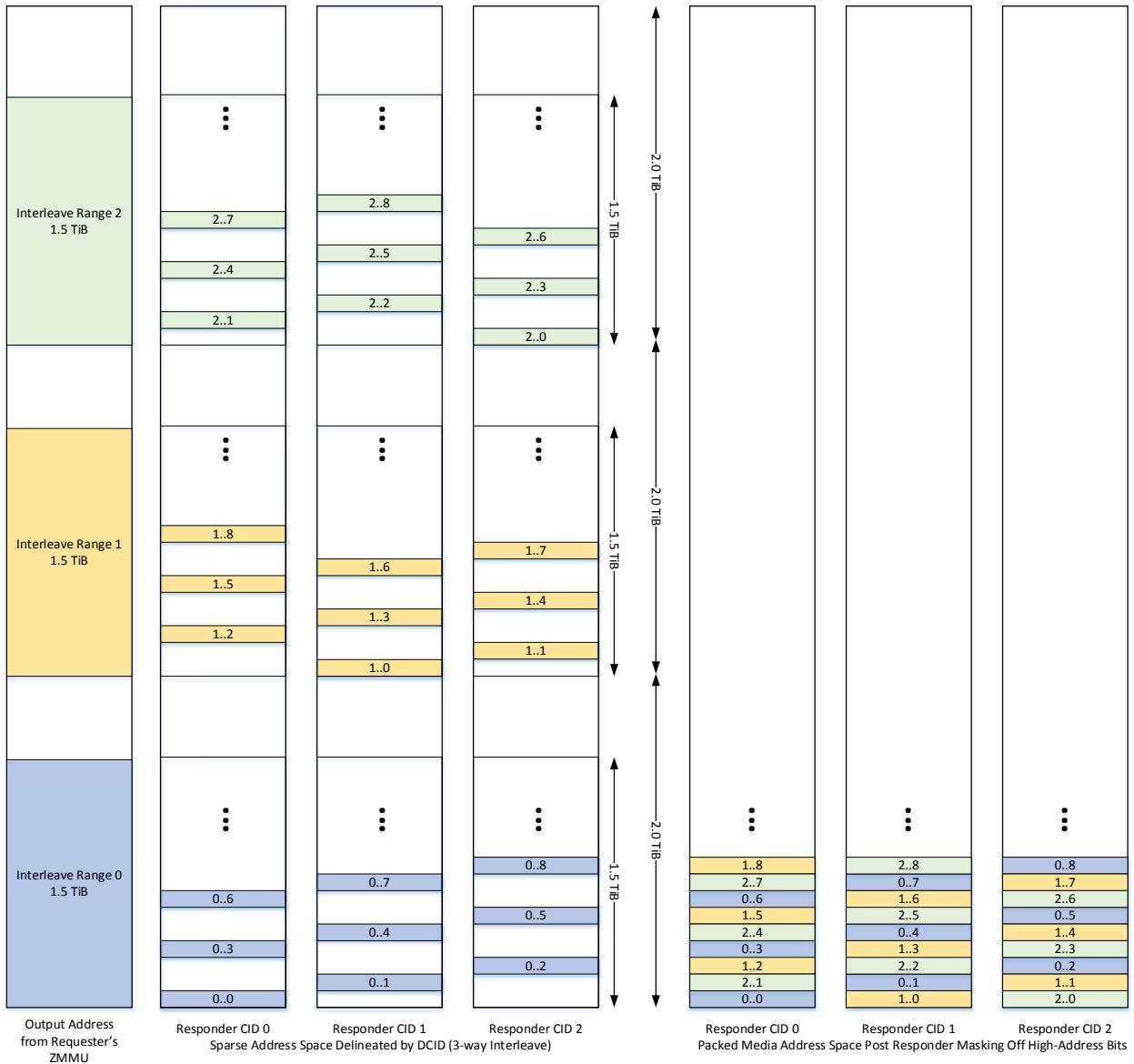


Figure 3-31: Example 3-way Stack Barber Pole Interleave

### 3.4.3.1. Component-specific and Gen-Z Interleave Relationship

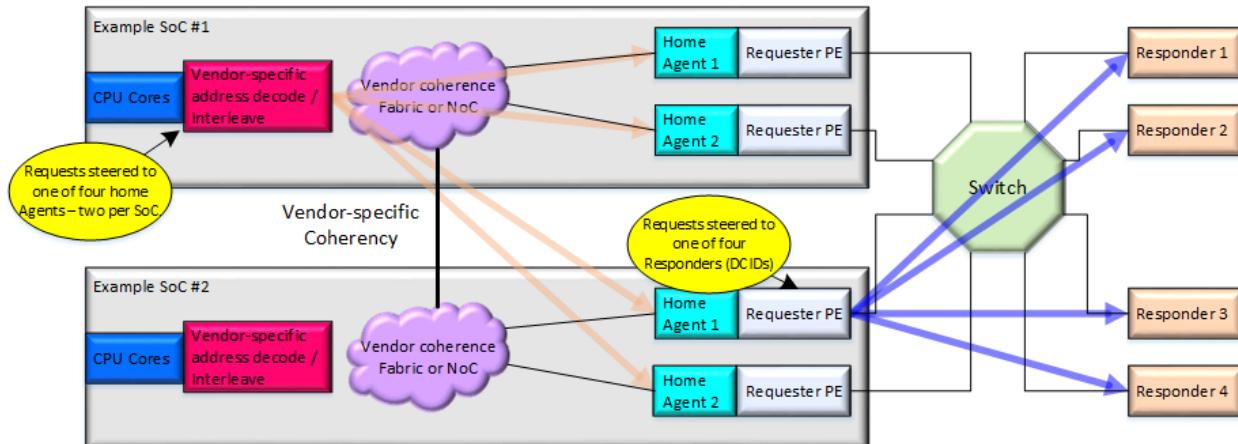
A component such as a SoC may impose component-specific requirements and constraints on interleaves. For example, interleave between SoC home agents, vendor-specific coherency links, cache line slices, etc. *Example Use of Component-specific and Fabric Interleaves in Switch Topology* illustrates two SoCs connected through a vendor-specific coherency interconnect.

- Each SoC is attached to a Gen-Z switch which is attached to four Responders.
- The orange arrows illustrate where component-specific decode and interleave are applied. This example illustrates that any CPU core can reach any home agent through the SoC-specific coherency fabric.
- The blue arrows illustrate where Gen-Z architected decode and interleave are applied. This example illustrates that any Requester PE (protocol engine) can reach any Responder. Each

Requester PE uses a combination of the Requester ZMMU and the Gen-Z Interleave Table (if supported) to identify the Responder and generate a request packet.

5 When processing a request destined to a Gen-Z Responder, the SoC shall apply the component-specific decode and interleave prior to applying the Gen-Z decode and interleave. This enables the two sets of decode and interleave to operate independently.

For non-interleaved regions, a component may apply component-specific decode and interleave to use multiple home agents and Requester PE to target the same Responder through multiple component interfaces and paths between the two components.



10 Figure 3-32: Example Use of Component-specific and Fabric Interleaves in Switch Topology

15 **Developer Note:** Point-to-point or daisy-chain-attached (P2P-Core OpClass) memory relies exclusively upon Requester-specific decoding and interleaving (Example Use of Component-specific and Fabric Interleaves in Switch Topology orange arrows). Data placement is a direct function of the Requester-specific decode / interleave. This enables low-latency access by eliminating the need for decode and interleave methods as specified by this document (blue arrows). Requester-specific decode and interleave can impact interoperability in multi-Requester daisy-chain topologies if disparate Requesters are used.

20 **Developer Note:** Though Example Use of Component-specific and Fabric Interleaves in Switch Topology illustrates a 1:1 relationship between home agents and Requester PEs, this is strictly for illustrative purposes. Implementations can support 1:1, N:1, and N:N relationships. Requester PEs can support any relationship by making each Requester PE capable of handling any outbound request to any address. Further, by enabling any-to-any communications, solutions can avoid crossing the vendor-specific coherency interconnect which in some situations could improve overall solution performance.

25 **Example Use of Component-specific Interleave in a Point-to-Point Topology** illustrates two SoCs connected through a vendor-specific coherency interconnect. Each SoC is attached to two of the four Responders using point-to-point links. The orange arrows illustrate where component-specific interleaves are applied. Since the Responders are point-to-point-attached, the component-specific interleave shall be the only interleave applied when processing a request destined to a Responder.

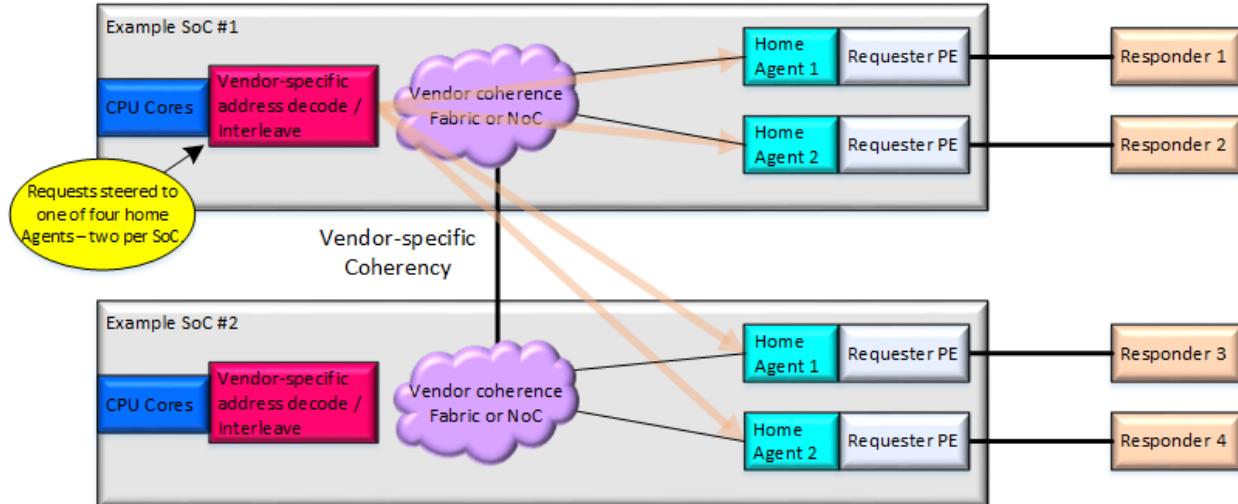


Figure 3-33: Example Use of Component-specific Interleave in a Point-to-Point Topology

### 3.4.4. Requester PTE

A Requester PTE is used when generating a request packet destined for a Responder (a Requester PTE can be mapped to a memory interleave group or a multicast group, each containing multiple Responders). Requester ZMMU support, supported page sizes, and the composition of a Requester PTE is indicated by the *Core Structure* ZMMU ATTR field.

*Requester PTE Format for Data Space* illustrates a Requester PTE used to access Data Space.

Local Destination	TC	Write Mode	NSE	LPE	PEC	PS	PME	WPE	SKE	CE	CCE	RKE	DT	ST	D-ATTR	ET
ADDR [63:12]																
PASID	Security Key ID										Global Destination					
R-Key																

Figure 3-34: Requester PTE Format for Data Space

*Requester PTE Format for Control Space* illustrates a Requester PTE used to access Control Space. The primary differences between the two Requester PTE formats is the Control Space format has a 52-bit effective address field and a DR-Interface field which is used to support *Directed Control Space Packet Relay*.

Local Destination	TC	Write Mode	PEC	PS	PME	WPE	SKE	CE	CCE	RKE	DT	ST	D-ATTR	ET	
DR-Interface	ADDR [51:12]														
PASID	Security Key ID										Global Destination				
R-Key															

Figure 3-35: Requester PTE Format for Control Space

15

Table 3-7: Requester PTE Fields

Field Name	Size (bits)	Access	Description
ET  Local Destination  D-ATTR	1	-	<p>Entry Type</p> <p>0b—Pointer-pair table entry 1b—PTE table entry</p>
	12	RW	The contents of this field depend upon the D-ATTR field value.
	3	RW	<p>This field indicates how to interpret the Local Destination and Global Destination fields.</p> <p>0x0—Invalid Entry 0x1—Interleave Table Index 0x2—Unicast Single-subnet Responder CID. Explicit OpClass packets shall set GC = 0b to indicate that the Responder is located in the same subnet as the Requester. The Local Destination field shall contain the Responder's CID. The Global Destination field shall be Reserved.</p> <p>0x3—Unicast multi-subnet Responder GCID. If present, explicit OpClass packets shall set the GC bit to 1b to indicate that the Responder is located in a different subnet from the Requester. The Local Destination field shall contain the Responder's CID. The Global Destination field shall contain the Responder's SID. The combination of the Local Destination and the Global Destination equal the Responder's GCID. A Requester shall use explicit OpClass request packets to communicate with the peer component.</p> <p>0x4—Multicast Single-subnet Group Identifier. Multicast OpClass packets shall set GC = 0b to indicate the multicast group spans only the local subnet. The Location Destination field shall contain the local subnet's Multicast Group Identifier. The Global Destination field shall be Reserved.</p> <p>0x5—Multicast Multi-subnet Group Identifier. Multicast OpClass packet shall set GC = 1b to indicate the multicast group spans multiple subnets. The Local Destination field shall contain the Multicast Group Identifier. The Global Destination field shall contain the Global Multicast Group Prefix. The combination of the Local Destination and the Global Destination field equals the GMGID (Global Multicast Group Identifier).</p> <p>0x6—Local Destination and Global Destination fields shall be Reserved. The Responder is point-to-point attached and one of</p>

Field Name	Size (bits)	Access	Description
Global Destination			<p>the point-to-point optimized OpClasses is used for communication.</p> <p>0x7—Reserved</p>
	Variable	RW	If present, then the Global Destination field semantics are determined by the D-ATTR field.
	1	RW	<p>Space Type—Determines the PTE format and if the page is associated with Control Space or Data Space.</p> <p>0b—Control Space PTE 1b—Data Space PTE</p>
DT	1	RW	<p>The use of this bit depends upon the value of the ST bit:</p> <p>Control Space PTE: DR Control</p> <p>0b—Value of the DR bit in Control Space request packets shall be 0b, specifying CID-based relay.</p> <p>1b—Value of the DR bit in Control Space request packets shall be 1b, specifying Directed Relay. The DR-Interface field shall be configured with the switch egress interface used to relay packets to the destination component.</p> <p>Data Space PTE: Target Cache</p> <p>0b—Indicates the page does not support coherency or the page supports coherency where caching is not a desired side-effect of the access. Coherency-specific request packets shall not target this page. Request packets that contain the Target Cache (TC) bit shall set TC = 0b.</p> <p>1b—Indicates the page supports coherency operations and that cache allocation is a desired side-effect of the access. Coherency-specific request packets may target this page. Request packets that contain the TC bit may set TC = 1b.</p>
Write Mode	3	RW	<p>Determines how a store / write operation is translated into a Gen-Z write request packet and the associated semantics.</p> <p>0x0—Writes to this page shall generate a write request packet and the application processing engine (e.g., a processor core) does not stall waiting for the corresponding <i>Standalone Acknowledgment</i> to arrive.</p> <p>0x1—Writes to this page shall generate a write request packet and the application processing engine stalls until the corresponding <i>Standalone Acknowledgment</i> arrives. If this page is associated with persistent media, then software is responsible for initiating a <i>Persistent Flush</i> request packet</p>

Field Name	Size (bits)	Access	Description
			<p>whenever it needs to ensure the payloads of all prior write request packets are persistent.</p> <p>0x2—Writes to this page shall generate a Write Persistent request packet and the application processing engine shall stall until the corresponding <i>Standalone Acknowledgment</i> arrives.</p> <p>0x3—Writes to this page shall not be acknowledged, i.e., if an explicit OpClass packet, then the Requester shall set UR = 1b.</p> <p>0x4—Writes to this page shall generate a <i>SOD Write</i> request packet and the application processing engine (e.g., a processor core) does not stall waiting for the corresponding <i>SOD Standalone Acknowledgment</i>.</p> <p>0x5—Writes to this page are translated into native Interrupt request packets. This PTE format does not support LPD / LPH interrupt request packet generation. Reads to this page shall return 0x0.</p> <p>0x6-0x7—Reserved</p>
PME	1	RW	<p>Performance Marker (PM) Enabled</p> <p>0b—Disabled. If an explicit OpClass packet contains the PM bit, then shall set PM = 0b when accessing this page.</p> <p>1b—Enabled. If an explicit OpClass packet contains the PM bit, then shall set PM = 1b when accessing this page.</p>
WPE	1	RW	<p>Write Poison Enabled</p> <p>0b—The Requester shall not generate <i>Write Poison</i> request packets to this page. If a <i>Write Poison</i> request targets this page, then the Requester shall initiate component-specific local error handling.</p> <p>1b—The Requester may generate <i>Write Poison</i> request packets to this page.</p>
RKE	1	RW	<p>R-Key Enabled</p> <p>0b—Disabled. If an explicit OpClass request packet is used to access this page and it contains the RK bit, then shall set RK = 0b. The Requester PTE R-Key field shall be Reserved.</p> <p>1b—Enabled. If an explicit OpClass request packet is used to access this page and it contains the RK bit, then shall set RK = 1b and the R-key field shall be included within the request packet. The Requester PTE R-Key field shall be copied to the R-key field in applicable explicit OpClass packets.</p>

Field Name	Size (bits)	Access	Description
<b>NSE</b>	1	RW	No-Snoop Enabled—Unless indicated by other means, this field indicates how the NS bit within request packets is to be set.  0b—NS = 0b 1b—NS = 1b
<b>LPE</b>	1	RW	LP Enable—if an explicit OpClass packet is used to communicate with the Responder, then this bit determines how to set the LP bit.  0b—Shall set LP = 0b 1b—Shall set LP = 1b
<b>CE</b>	1	RW	Capabilities Enable—Determines if the Requester may transmit <i>Capabilities Read</i> and <i>Capabilities Write</i> request packets to this page.  0b—Disabled 1b—Enabled
<b>Traffic Class (TC)</b>	4	RW	Traffic Class—if the component supports the <i>Component Destination Table Structure</i> , then the Traffic Class acts as an input in the VC-to-egress interface selection process.  Depending upon the topology and routing algorithm, multiple VCs may be associated with a given Traffic Class.  Traffic Class management and selection are outside of this specification’s scope.
<b>SKE</b>	1	RW	Security Key Enable—if the component supports page-level encryption, then this bit determines if the Security Key ID field is valid.  0b—Disabled 1b—Enabled
<b>PS</b>	1	RW	PASID Enable—if the PTE contains a PASID field, then this bit indicates if the field is valid and whether the component may use the PASID field when accessing this page.  0b—Disabled 1b—Enabled
<b>PEC</b>	1	RW	Processor Exception Control—if D-ATTR == Invalid Entry, the PTE is invalid for other reasons, or a generated Request is unable to be completed successfully, then this bit determines the local processor exception semantics.  0b—Invoke platform-specific local processor error handling; e.g., some form of synchronous or asynchronous processor exception like an interrupt, trap, machine check, instruction abort, etc.

Field Name	Size (bits)	Access	Description
CCE	1	RW	1b—Reads return all 1s and Writes are Silently Dropped without necessarily invoking any form of processor exception
			Cache Coherence Enabled 0b—Disabled: Coherency-specific request packets shall not target this page, and the TC field in applicable request packets shall be set to 0b.
			1b—Enabled: Coherency-specific request packets may target this page and the TC bit in applicable request packets may be set to 1b.
Security Key ID	Variable	RW	Security Key ID—if the component supports page-level encryption, then this field contains the security key identifier associated with this page.
PASID	Variable	RW	If present and PS == 1b, then this field contains the process space identifier associated with this page.
ADDR	Variable	RW	<p>The use of this field depends upon the value of the Space Type bit (bit 0 in the Page Attributes field), which determines whether this PTE generates Control Space or Data Space request packets.</p> <p>Control Space PTE: 12 bits for DR-Interface; up to 40 bits for Control Space Address (52-bit address space)</p> <p>DR-Interface[11:0] = ADDR[63:52]</p> <p>Address bits in the Input Address are replaced by their counterparts from this field to create the request packet's Address field. For example:</p> <p>Page size = 4 KiB:</p> <p>Address[51:12] = ADDR[51:12] Address[11:0] = Input Address[11:0]</p> <p>Page size = 64 KiB:</p> <p>Address[51:16] = ADDR[51:16] Address[15:0] = Input Address[15:0]</p> <p>Page size = 32 MiB:</p> <p>Address[51:25] = ADDR[51:25] Address[24:0] = Input Address[24:0]</p> <p>Data Space PTE: up to 52 bits for Data Space Address (64-bit address space)</p>
DR-Interface	12	RW	If D-ATTR ≠ 0x6 and DT == 1b, then this field contains the switch egress interface identifier used to access a destination

Field Name	Size (bits)	Access	Description
ADDR (Data Space Page)			<p>component when performing <i>Directed Control Space Packet Relay</i> operations. This field shall be configured by software prior to setting the Page Attributes DR Control bit to 1b.</p> <p>If D-ATTR == 0x6 and DT == 1b, then this field contains the interface identifier of the Requester interface used to exchange <i>In-band Management</i> request and response packets supported by point-to-point optimized OpClasses.</p>
R-Key	52	RW	<p>Address bits in the Input Address are replaced by their counterparts from this field to create the request packet's Address field. For example:</p> <p>Page size = 4 KiB:</p> $\text{Address}[63:12] = \text{ADDR}[63:12] \text{ Address}[11:0] = \text{Input Address}[11:0]$ <p>Page size = 64 KiB:</p> $\text{Address}[63:16] = \text{ADDR}[63:16]$ $\text{Address}[15:0] = \text{Input Address}[15:0]$ <p>Page size = 32 MiB:</p> $\text{Address}[63:25] = \text{ADDR}[63:25]$ $\text{Address}[24:0] = \text{Input Address}[24:0]$
	32	RW	If RKE == 1b, then this field contains the R-Key to use when generating applicable explicit OpClass packets. See <i>Region Key (R-Key)</i> .

### 3.4.5. Responder PTE

A Responder PTE is used to validate and execute request packets. Though Responder PTE contents and structure may be component-specific, *Responder PTE Format* illustrates an example Responder PTE. Responder ZMMU support, supported page sizes, and the composition of a Responder PTE is indicated by the *Core Structure* ZMMU ATTR field.

5

**Developer Note:** A Responder PTE does not contain a Responder-local address field. Instead, a Responder may provision one or more BARs used to map a request packet's Address field into a Responder-local address.

PASID	RK_MGR_SID	RK_MGR_CID	PS	SKE	CE	CC	RKE	PA	RKMGR	ET
RW R-Key		RO R-Key				Security Key ID				

Table 3-8: Responder PTE Fields

Field Name	Size (bits)	Access	Description
ET	1	-	Entry Type 0b—Pointer-pair table entry 1b—PTE table entry
RKMGR	2	RW	If a Responder supports <i>Region Key (R-Key)</i> and <i>R-Key Update</i> request packets, then this field determines how to interpret the contains the R-Key Manager CID and R-Key Manager SID fields (if present). 0x0—Unsupported 0x1—Valid R-Key Manager CID 0x2—Valid R-Key Manager CID & R-Key Manager SID 0x3—Reserved
PA	1	RW	Persistence Access—Determines if the page is accessed as persistent media. 0b—Accessed as non-persistent media. 1b—Accessed as persistent media. Write Persistent and Persistent Flush request packets may target this page.
RKE	1	RW	R-Key Enabled 0b—Disabled. The RO R-Key and the RW R-Key fields shall be Reserved. R-Key validation shall not be performed on request packets that access this page. 1b—Enabled. The RO R-Key and the RW R-Key fields shall contain valid R-Key values, and R-Key validation shall be performed if enabled
CCE	1	RW	Cache Coherence Enabled—Determines if the page may be accessed by coherency-specific request packets or request packets with Target Cache (TC) = 1b. 0b—Disabled: Coherency-specific request packets that target this page shall be handled as a MP error, and the Target Cache (TC) bit in applicable request packets shall be ignored by the Responder. 1b—Enabled: Coherency-specific request packets may target this page, and the Target Cache (TC) bit in applicable request packets shall be acted upon by the Responder.
CE	1	RW	Capabilities Enable—Determines if the page may be accessed by <i>Capabilities Read</i> and <i>Capabilities Write</i> request packets. 0b—Disabled: <i>Capabilities Read</i> and <i>Capabilities Write</i> request packets shall not access this page. Upon receipt of a

Field Name	Size (bits)	Access	Description
			capabilities request packet that targets this page, the Responder shall handle this as a MP error. 1b—Enabled: <i>Capabilities Read</i> and <i>Capabilities Write</i> request packets may access this page.
SKE	1	RW	Security Key Enable—if the component supports page-level encryption, then this bit determines if the Security Key ID field is valid. 0b—Disabled 1b—Enabled
PS	1	RW	PASID Enable—if the PTE contains a PASID field, then this bit indicates if the field is valid and whether the component may use the PASID field when accessing this page. 0b—Disabled 1b—Enabled
RK_MGR_CID	12	RW	If present, then this field contains the CID of the component that is authorized to transmit <i>R-Key Update</i> request packets to perform high-speed R-Key updates for this page.
RK_MGR_SID	Variable	RW	If present, then this field contains the SID of the component that is authorized to transmit <i>R-Key Update</i> request packets to perform high-speed R-Key updates for this page.
Security Key ID	Variable	RW	Security Key ID—if the component supports page-level encryption, then this field contains the security key identifier associated with this page.
PASID	Variable	RW	If present and PS == 1b, then this field contains the process space identifier associated with this page.
RO R-Key	32	RW	If RKE == 1b, then this field contains the Read-only R-Key to use when validating access permission. See <i>Region Key (R-Key)</i> .
RW R-Key	32	RW	If RKE == 1b, then this field contains the Read-Write R-Key to use when validating access permission. See <i>Region Key (R-Key)</i> .

## 4. End-to-end OpClasses & OpCodes

Components interact with one another through end-to-end request and response packets. Each request or response packet is associated with an operation class (OpClass) and an operation code (OpCode).

This section specifies the different OpClasses and their associated OpCodes used in end-to-end packets.

5 A component may support one or more OpClasses based on its solution-specific needs. Cooperating components need to support at least one common OpClass to ensure interoperability.

### 4.1. End-to-end Operation Classes

To minimize resource requirements and provide flexibility and extensibility, request and response packets are organized into operation classes (OpClass).

- 10 • The architecture supports a total of thirty five OpClasses.
- o A component may support the P2P-Core OpClass. The P2P-Core OpClass is enabled on a per component interface basis. If enabled, then the corresponding interface shall not exchange packets associated with any other OpClass.
  - o A component may support the P2P-Coherency OpClass. The P2P-Coherency OpClass is enabled on a per component interface basis. If enabled, then the corresponding interface shall not exchange packets associated with any other OpClass.
  - o A component may support an implicit P2P Vendor-defined OpClass. An implicit P2P Vendor-defined OpClass is enabled on a per component interface basis. If enabled, then the corresponding interface shall not exchange packets associated with any other OpClass.
  - o All other OpClasses are explicitly identified by the packet's OCL field. *Explicit End-to-end OpClass Field Encodings* specifies the OpClass and corresponding OCL encoding. Packets associated with any explicit OpClass may be exchanged on any interface not enabled for the P2P-Core, P2P-Coherency, or P2P-Vendor-defined OpClasses. Explicit OpClasses may be used in point-to-point, meshed, switch (single or multi-subnet), and *Transparent Router (TR)* topologies.
  - o Each component shall support at least one of the following OpClasses: P2P-Core, P2P-Coherency, P2P-Vendor-defined, or Core 64. A component may support additional OpClasses.

20 25 20 25 Table 4-1: Explicit End-to-end OpClass Field Encodings

OpClass (OCL) Name	OpClass Abbreviation	OCL Encoding	Description
Core 64	Core 64	0x0	Used to support memory, I/O, and storage operations as well as to support <i>Logical PCI Device (LPD)</i> .
Control	CTL	0x1	Used to support <i>In-band Management</i> .
Atomic 1	ATOMIC1	0x2	Used to exchange Atomic request and response packets.
Large Data Move 1	LDM1	0x3	Used to perform large data movement operations.

OpClass (OCL) Name	OpClass Abbreviation	OCL Encoding	Description
Advanced 1	ADV1	0x4	Used to support pattern / regular expression acceleration and lightweight notification applications.
Advanced 2	ADV2	0x5	Used to support packet encapsulation and precision time applications.
Reserved OpClasses	-	0x6-0x14	Reserved OpClasses
Context ID	CTXID	0x15	Used to support operations that use a context identifier in place of an address to identify a target resource.
Multicast	MULT	0x16	Used to support Multicast solutions.
Strong Order Domain	SOD	0x17	Used to support strongly-ordered packet communications.
Vendor-Defined	VDOPC	0x18- 0x1F	<p>Used to exchange customized explicit OpClass operations between cooperating components (P2P-Core and P2P-Coherency OpClasses also support a limited number of vendor-defined OpCodes).</p> <p>A component may support 0-8 vendor-defined OpClasses.</p> <p>Vendor-defined Explicit OpClass packets are defined in <i>Vendor-Defined Packets</i>.</p>

## 4.2. End-to-end Packet Operation Codes

An operation code (OpCode) identifies the operation and packet type. The following tables provide a list of OpCodes associated with each implicit and explicit OpClass.

The OpCode encodings in each OpClass are partitioned into a response operation range and a request operation range.

- OpCode encodings 0x0-0x3 shall be associated with response operations.
- OpCode encodings 0x4-0x1F shall be associated with request operations.

OpCode Table Legend:

- ID = Idempotent request
- MO = Mandatory / Optional operation
  - M = Mandatory (unconditional)
    - A Requester shall be capable of generating the corresponding request packet and receiving and executing the corresponding response packet.
    - A Responder shall be capable of receiving and executing the corresponding request packet and generating the corresponding response packet.

10

15

- MC = Mandatory (conditional)—If the described condition indicated is true, then the operation shall be treated as Mandatory.
- O = Optional
  - A Requester may support this operation. If supported, a Requester shall be capable of generating the corresponding request packet and receiving and executing the corresponding response packet.
  - A Responder may support this operation. If supported, a Responder shall be capable of receiving and executing the corresponding request packet and generating the corresponding response packet.
- RK = Indicates if the corresponding packet contains the RK bit used to indicate if a *Region Key (R-Key)* field is present.
  - N = RK bit is not present
  - R = RK bit is present and the R-Key field may match RO R-Key or RW R-Key
  - W = RK bit is present and the R-Key field shall match the RW R-Key
- ND = indicates if execution of the operation may take a non-deterministic time to execute, e.g., a 1 GiB Buffer Put operation will take considerably longer to complete than a simple read operation.

#### 4.2.1. P2P-Core OpClass Operation Codes

The P2P Core OpClass is used to connect a Requester (e.g., a processor) and one or more memory components (Responders) in a point-to-point or daisy-chain topology.

Table 4-2: P2P-Core OpClass OpCodes

OpCode Name	OpCode Encoding	M O	N D	Description
<b>Standalone Acknowledgment</b>	0x0	M	N	See <i>Standalone Acknowledgment</i>
<b>Read Response</b>	0x1	M	N	Read Response. See <i>Read and Read Response</i>
<b>Reserved</b>	0x2	-	-	Reserved Response OpCode
<b>SubOp 1 Response</b>	0x3	-		Response packet contains a Sub-OPC field that identifies the specific response type. See <i>P2P-Core Sub-Op 1 Response OpCodes</i>
<b>Write</b>	0x4	M	N	See <i>Write Requests</i>
<b>Write Offset</b>	0x5	M C	N	Write Request—Write payload bytes at the indicated offset (See <i>Write</i> )  If a component supports any P2P-Core Read Offset and Write Offset operation, then it shall support all Read Offset OpCodes and the Write Offset OpCode.
<b>Write Persistent</b>	0x6	O	N	Write Persistent—See <i>Write</i>
<b>Write Poison</b>	0x7	O	N	See <i>Write Poison</i>
<b>Persistent Flush</b>	0x8	O	N	See <i>Persistent Flush</i>

OpCode Name	OpCode Encoding	M O	N D	Description
<b>Capabilities Read</b>	0x9	O	N	See <i>Capabilities Read</i>
<b>Capabilities Write</b>	0xA	O	N	See <i>Capabilities Write</i>
<b>Buffer Put Local</b>	0xB	O	Y	See <i>Buffer Requests</i>
<b>Read 16</b>	0xC	O	N	Fixed-sized Read Request—16 bytes. See <i>Read and Read Response</i>
<b>Read 32</b>	0xD	O	N	Fixed-sized Read Request—32 bytes. See <i>Read and Read Response</i>
<b>Read 64</b>	0xE	M	N	Fixed-sized Read Request—64 bytes. See <i>Read and Read Response</i>
<b>Read 128</b>	0xF	O	N	Fixed-sized Read Request—128 bytes. See <i>Read and Read Response</i>
<b>Read 256</b>	0x10	O	N	Fixed-sized Read Request—256 bytes. See <i>Read and Read Response</i>
<b>Read Offset 32</b>	0x11	M C	N	<p>Fixed-sized Read Request—32 bytes at the indicated offset. See <i>Read and Read Response</i></p> <p>If a component supports any P2P-Core Read Offset and Write Offset operation, then it shall support all Read Offset OpCodes and the Write Offset OpCode.</p>
<b>Read Offset 64</b>	0x12	M C	N	<p>Fixed-sized Read Request—64 bytes at the indicated offset. See <i>Read and Read Response</i></p> <p>If a component supports any P2P-Core Read Offset and Write Offset operation, then it shall support all Read Offset OpCodes and the Write Offset OpCode.</p>
<b>Read Offset 128</b>	0x13	M C	N	<p>Fixed-sized Read Request—128 bytes at the indicated offset. See <i>Read and Read Response</i></p> <p>If a component supports any P2P-Core Read Offset and Write Offset operation, then it shall support all Read Offset OpCodes and the Write Offset OpCode.</p>
<b>Read Offset 256</b>	0x14	M C	N	<p>Fixed-sized Read Request—256 bytes at the indicated offset. See <i>Read and Read Response</i></p> <p>If a component supports any P2P-Core Read Offset and Write Offset operation, then it shall support all Read Offset OpCodes and the Write Offset OpCode.</p>
<b>Reserved</b>	0x15-0x1B	-	-	Reserved Request OpCodes
<b>Meta 32 Write</b>	0x1C	O	N	See <i>Meta Read Response and Meta Write</i>

OpCode Name	OpCode Encoding	M O	N D	Description
Meta 64 Write Write Partial SubOp 1 Request	0x1D	O	N	See <i>Meta Read Response and Meta Write</i>
	0xE	M	N	See <i>Write Partial</i>
	0x1F	O	N	Request packet contains a Sub-OPC field that identifies the specific request type. See <i>P2P-Core Sub-Op 1 Request OpCodes</i>

Table 4-3: P2P-Core Sub-Op 1 Response OpCodes

OpCode Name	OpCode Encoding	I D	M O	N D	Description
Atomic Response Atomic Results Response Reserved Vendor-defined Response [0-15]	0x0	-	O	N	This OpCode is used to complete a P2P-Core Atomic request without returning the results. A P2P-Core Atomic Response may be returned for any P2P-Core Atomic request type.
	0x1	-	O	N	This OpCode is used to complete a P2P-Core Atomic request and return results.
	0x2-0x5F	-	-	-	Reserved Sub-Op 1 Response OpCodes
	0x60-0x7F	-	O	-	Vendor-defined Response OpCodes [0-15]. See <i>Vendor-Defined Packets</i>

Table 4-4: P2P-Core Sub-Op 1 Request OpCodes

OpCode Name	OpCode Encoding	I D	M O	N D	Description
Add Sum Swap CAS CAS Not Equal Logical OR	0x0	N	O	N	This OpCode is used to request a single add operation.
	0x1	N	O	N	This OpCode is used to request the data at two memory locations be added together.
	0x2	N	O	N	This OpCode is used to request a single unconditional swap operation.
	0x3	N	O	N	This OpCode is used to request a single compare-and-swap if equal operation.
	0x4	N	O	N	This OpCode is used to request a single compare-and-swap if not equal operation.
	0x5	N	O	N	This OpCode is used to request a bitwise logical OR operation.

OpCode Name	OpCode Encoding	I D	M O	N D	Description
Logical XOR	0x6	N	O	N	This OpCode is used to request a bitwise logical XOR operation.
Logical AND	0x7	N	O	N	This OpCode is used to request a bitwise logical AND operation.
Load Max	0x8	N	O	N	This OpCode is used to request a single load maximum operation. The operand may be signed or unsigned.
Load Min	0x9	N	O	N	This OpCode is used to request a single load minimum operation. The operand may be signed or unsigned.
Test-Zero-and-Modify	0xA	N	O	N	This OpCode is used to request a single test-zero-and-modify operation.
Increment Bounded	0xB	N	O	N	This OpCode is used to request a single increment bounded operation.
Increment Equal	0xC	N	O	N	This OpCode is used to request a single increment equal operation.
Decrement Bounded	0xD	N	O	N	This OpCode is used to request a single decrement bounded operation.
Compare Store Twin	0xE	N	O	N	This OpCode is used to request a single compare and store twin operation.
Atomic Vector Sum Request	0xF	N	O	N	This OpCode is used to request multiple sum operations.
Atomic Vector Logical Request	0x10	N	O	N	This OpCode is used to request multiple logical operations.
Atomic Fetch	0x11	N	O	N	This OpCode is used to atomically read the addressed resource.
Reserved	0x12-0x50	-	-	-	Reserved Sub-Op 1 Request OpCodes
CTL-Read	0x51	N	M C	N	If a component supports <i>In-band Management</i> , then it shall support CTL-Read. See <i>CTL-Read and CTL-Write Packets</i> .
CTL-Write	0x52	N	M C	N	If a component supports <i>In-band Management</i> , then it shall support CTL-Write. See <i>CTL-Read and CTL-Write Packets</i> .
CTL-UE	0x53	N	M C	N	If a component supports <i>In-band Management</i> , then it shall support CTL-UE. See <i>CTL-UE Packet</i> .
Reserved	0x54-0x5F	-	-	-	Reserved Sub-Op 1 Request OpCodes

OpCode Name	OpCode Encoding	I D	M O	N D	Description
Vendor-defined Request [0-31]	0x60-0x7F	-	O	-	Vendor-defined Request OpCodes [0-31]. See <i>Vendor-Defined Packets</i>

#### 4.2.2. P2P-Coherency OpClass Operation Codes

Table 4-5: P2P-Coherency OpClass OpCodes

OpCode Name	OpCode Encoding	M O	N D	Description
Standalone Acknowledgment	0x0	M	N	See <i>Standalone Acknowledgment</i>
Read Response	0x1	M	N	See <i>Read and Read Response</i>
Cache Line Attribute Response	0x2	O	N	See <i>Cache Line Attribute</i>
SubOp 1 Response	0x3	-	-	Response packet contains a Sub-OPC field that identifies the specific response type. See <i>P2P-Coherency Sub-Op 1 Response OpCodes</i>
Write	0x4	M	N	<i>Write</i>
Read	0x5	M	N	See <i>Read and Read Response</i>
Read Exclusive	0x6	O	N	See <i>Read Exclusive</i>
Read Shared	0x7	O	N	See <i>Read Shared</i>
Release	0x8	O	N	See <i>Release</i>
Invalidate	0x9	O	N	See <i>Invalidate and Writeback</i>
Writeback	0xA	O	N	See <i>Invalidate and Writeback</i>
Cache Line Attribute Request	0xB	O	N	See <i>Cache Line Attribute</i>
Exclusive	0xC	O	N	See <i>Exclusive</i>
Write Poison	0xD	O	N	See <i>Write Poison</i>
Interrupt	0xE	O	N	See <i>Interrupts</i>
Reserved	0xE-0x1B	-	-	Reserved Request OpCode
Meta 32 Write	0x1C	O	N	See <i>Meta Read Response and Meta Write</i>
Meta 64 Write	0x1D	O	N	See <i>Meta Read Response and Meta Write</i>
Write Partial	0x1E	M	N	See <i>Write Partial</i>

OpCode Name	OpCode Encoding	M O	N D	Description
SubOp 1 Request	0x1F	O	N	Request packet contains a Sub-OPC field that identifies the specific request type. See <i>P2P-Coherency Sub-Op 1 Request OpCodes</i>

Table 4-6: P2P-Coherency Sub-Op 1 Response OpCodes

OpCode Name	OpCode Encoding	I D	M O	N D	Description
Atomic Response	0x0	-	O	N	This OpCode is used to complete an Atomic request without returning the results.  An Atomic Response may be returned for any Atomic request type.
Atomic Results Response	0x1	-	O	N	This OpCode is used to complete an Atomic request and return results.
Translation RSP	0x2	-	O	-	See <i>Translation Response</i>
Translation Invalidate RSP	0x3	-	O	-	See <i>Translation Invalidate Response</i>
Reserved	0x4-0x5F	-	-	-	Reserved Sub-Op 1 Response OpCodes
Vendor-defined Response [0-31]	0x60-0x7F	-	O	-	Vendor-defined Response OpCodes [0-31]. See <i>Vendor-Defined Packets</i>

Table 4-7: P2P-Coherency Sub-Op 1 Request OpCodes

OpCode Name	OpCode Encoding	I D	M O	N D	Description
Add	0x0	N	O	N	This OpCode is used to request a single add operation.
Sum	0x1	N	O	N	This OpCode is used to request the data at two memory locations be added together.
Swap	0x2	N	O	N	This OpCode is used to request a single unconditional swap operation.
CAS	0x3	N	O	N	This OpCode is used to request a single compare-and-swap if equal operation.
CAS Not Equal	0x4	N	O	N	This OpCode is used to request a single compare-and-swap if not equal operation.
Logical OR	0x5	N	O	N	This OpCode is used to request a bitwise logical OR operation.

OpCode Name	OpCode Encoding	I D	M O	N D	Description
Logical XOR	0x6	N	O	N	This OpCode is used to request a bitwise logical XOR operation.
Logical AND	0x7	N	O	N	This OpCode is used to request a bitwise logical AND operation.
Load Max	0x8	N	O	N	This OpCode is used to request a single load maximum operation. The operand may be signed or unsigned.
Load Min	0x9	N	O	N	This OpCode is used to request a single load minimum operation. The operand may be signed or unsigned.
Test-Zero-and-Modify	0xA	N	O	N	This OpCode is used to request a single test-zero-and-modify operation.
Increment Bounded	0xB	N	O	N	This OpCode is used to request a single increment bounded operation.
Increment Equal	0xC	N	O	N	This OpCode is used to request a single increment equal operation.
Decrement Bounded	0xD	N	O	N	This OpCode is used to request a single decrement bounded operation.
Compare Store Twin	0xE	N	O	N	This OpCode is used to request a single compare and store twin operation.
Atomic Vector Sum Request	0xF	N	O	N	This OpCode is used to request multiple sum operations.
Atomic Vector Logical Request	0x10	N	O	N	This OpCode is used to request multiple logical operations.
Atomic Fetch	0x11	N	O	N	This OpCode is used to atomically read the addressed resource.
Reserved	0x12-0x50	-	-	-	Reserved Sub-Op 1 Request OpCodes
CTL-Read	0x51	N	M C	N	If a component supports <i>In-band Management</i> , then it shall support CTL-Read. See <i>CTL-Read and CTL-Write Packets</i> .
CTL-Write	0x52	N	M C	N	If a component supports <i>In-band Management</i> , then it shall support CTL-Write. See <i>CTL-Read and CTL-Write Packets</i> .
CTL-UE	0x53	N	M C	N	If a component supports <i>In-band Management</i> , then it shall support CTL-UE. See <i>CTL-UE Packet</i> .
Translation Req	0x54	Y	O	Y	See Translation Request

OpCode Name	OpCode Encoding	I D	M O	N D	Description
PRG Req	0x55	Y	O	Y	See <i>PRG Request</i>
PRG Response Notification	0x56	-	O	N	See <i>PRG Response Notification</i>
Translation Invalidate Req	0x57	Y	O	Y	See <i>Translation Invalidate Request</i>
Stop Marker Req	0x58	Y	O	N	See <i>Stop Marker</i>
PRG Release	0x59	Y	O	N	See <i>PRG Release</i>
PRG Eviction Notification	0x5A	Y	O	N	See <i>PRG Eviction Notification</i>
Reserved	0x5B-0x5F	-	-	-	Reserved Sub-Op 1 Request OpCodes
Vendor-defined Request [0-31]	0x60-0x7F	-	O	-	Vendor-defined Request OpCodes [0-31]. See <i>Vendor-Defined Packets</i>

#### 4.2.3. Core 64 OpClass Operation Codes

Table 4-8: Core 64 OpClass Unique OpCodes

OpCode Name	OpCode Encoding	I D	M O	R K	N D	Description
Standalone Acknowledgment	0x0	-	M	N	N	See <i>Standalone Acknowledgment</i>
Read Response	0x1	-	M	N	N	Read Response. See <i>Read and Read Response</i>
Cache Line Attribute Response	0x2	-	O	N	N	See <i>Cache Line Attribute</i>
Reserved	0x3	-	-	-	-	Reserved Response OpCode
Write	0x4	Y	M	W	N	See <i>Write</i>
Reserved	0x5-0x6	-	-	-	-	Reserved Request OpCodes
Write Poison	0x7	Y	O	W	N	See <i>Write Poison</i>
Interrupt	0x8	Y	O	W	N	See <i>Interrupts</i>
Persistent Flush	0x9	Y	O	N	N	See <i>Persistent Flush</i>
Invalidate	0xA	Y	O	R	N	See <i>Invalidate and Writeback</i>
Writeback	0xB	Y	O	W	N	See <i>Invalidate and Writeback</i>
Read Exclusive	0xC	Y	O	R	N	See <i>Read Exclusive</i>

OpCode Name	OpCode Encoding	I D	M O	R K	N D	Description
Read Shared	0xD	Y	O	R	N	See <i>Read Shared</i>
Release	0xE	Y	O	R	N	See <i>Release</i>
Cache Line Attribute Request	0xF	Y	O	R	N	See <i>Cache Line Attribute</i>
Wake Thread	0x10	Y	O	N	N	See <i>Wake Thread</i>
Capabilities Read	0x11	Y	O	R	N	See <i>Capabilities Read</i>
Capabilities Write	0x12	Y	O	W	N	See <i>Capabilities Write</i>
Write Partial	0x13	Y	M	W	N	See <i>Write Partial</i>
Exclusive	0x14	Y	O	W	N	See <i>Exclusive</i>
Reserved	0x15-0x19	-	-	-	-	Reserved Request OpCodes
Multi-Op	0x1A	-	O	R/ W	N	See <i>Multi-Op Request</i>
Read	0x1B	Y	M	R	N	Variable-sized Read Request. See <i>Read and Read Response</i>
Write-Under-Mask	0x1C	Y	O	W	N	See <i>Write-Under-Mask (WUM)</i>
NIR Request	0x1D	Y	O	N	N	See <i>Non-Idempotent Request Release (NIRR)</i>
Reserved	0x1E-0x1F	-	-	-	-	Reserved Request OpCodes

#### 4.2.4. Control OpClass Operation Codes

Table 4-9: Control OpClass OpCodes

OpCode Name	OpCode Encoding	I D	M O	R K	N D	Description
Standalone Acknowledgment	0x0	-	M	N	N	See <i>Standalone Acknowledgment</i>
Read Response	0x1	-	M	N	N	See <i>Read and Read Response</i>
Reserved	0x2	-	-	-	-	Reserved Response OpCode
Reserved	0x3	-	-	-	-	Reserved Response OpCode
Write	0x4	Y	M	W	N	See <i>Control Read and Control Write Packets</i>
Reserved	0x5-0x7	-	-	-	-	Reserved Request OpCodes
Interrupt	0x8	Y	O	W	N	See <i>Interrupts</i>
No-Op	0x9	Y	M	N	N	See <i>No-Op</i>

OpCode Name	OpCode Encoding	I D	M O	R K	N D	Description
Unsolicited Event	0xA	Y	M	N	N	See <i>Unsolicited Event (UE) Packet</i>
C-State Power Control	0xB	Y	O	N	N	See <i>C-State Power Control</i>
R-Key Update	0xC	Y	O	N	N	See <i>R-Key Update</i>
Reserved	0xD-0x1A	-	-	-	-	Reserved Request OpCodes
Read	0x1B	Y	M	R	N	See <i>Control Read and Control Write Packets</i>
Reserved	0x1C-0x1E	-	-	-	-	Reserved Request OpCodes
Control Write MSG	0x1F	Y	O	N	N	See <i>Control Write MSG</i>

**Developer Note:** Solutions that operate in larger topologies and use network services, e.g., traditional networking or network boot services, are strongly encouraged to support the Control Write MSG operation. In many solutions, this operation will be required to initialize the fabric and services.

#### 4.2.5. Atomic 1 OpClass Operation Codes

Table 4-10: Atomics 1 OpClass OpCodes

OpCode Name	OpCode Encoding	I D	M O	R K	N D	Description
Atomic Response	0x0	-	O	N	N	This OpCode is used to complete an Atomic request without returning the results.  An Atomic Response may be returned for any Atomic request type.
Atomic Results Response	0x1	-	O	N	N	This OpCode is used to complete an Atomic request and return results.
Reserved	0x2-0x3	-	-	-	-	Reserved Response OpCodes
Add	0x4	N	O	W	N	This OpCode is used to request a single add operation.
Sum	0x5	N	O	W	N	This OpCode is used to request the data at two memory locations be added together.
Swap	0x6	N	O	W	N	This OpCode is used to request a single unconditional swap operation.
CAS	0x7	N	O	W	N	This OpCode is used to request a single compare-and-swap if equal operation.

OpCode Name	OpCode Encoding	I	M	R	N	D	Description
CAS Not Equal	0x8	N	O	W	N		This OpCode is used to request a single compare-and-swap if not equal operation.
Logical OR	0x9	N	O	W	N		This OpCode is used to request a bitwise logical OR operation.
Logical XOR	0xA	N	O	W	N		This OpCode is used to request a bitwise logical XOR operation.
Logical AND	0xB	N	O	W	N		This OpCode is used to request a bitwise logical AND operation.
Load Max	0xC	N	O	W	N		This OpCode is used to request a single load maximum operation. The operand may be signed or unsigned.
Load Min	0xD	N	O	W	N		This OpCode is used to request a single load minimum operation. The operand may be signed or unsigned.
Test-Zero-and-Modify	0xE	N	O	W	N		This OpCode is used to request a single test-zero-and-modify operation.
Increment Bounded	0xF	N	O	W	N		This OpCode is used to request a single increment bounded operation.
Increment Equal	0x10	N	O	W	N		This OpCode is used to request a single increment equal operation.
Decrement Bounded	0x11	N	O	W	N		This OpCode is used to request a single decrement bounded operation.
Compare Store Twin	0x12	N	O	W	N		This OpCode is used to request a single compare and store twin operation.
Atomic Vector Sum Request	0x13	N	O	W	N		This OpCode is used to request multiple sum operations.
Atomic Vector Logical Request	0x14	N	O	W	N		This OpCode is used to request multiple logical operations.
Atomic Fetch	0x15	N	O	W	N		This OpCode is used to atomically read the addressed resource.
Reserved	0x16-0x1F	-	-	-	-		Reserved Request OpCodes

#### 4.2.6. LDM 1 OpClass Operation Codes

Table 4-11: LDM 1 OpClass OpCodes

OpCode Name	OpCode Encoding	I D	M O	R K	N D	Description
<b>LDM 1 Read Response</b>	0x0	Y	O	N	N	LDM 1 Read Response. See <i>Read and Read Response</i>
<b>Buffer Allocate Response</b>	0x1	-	O	N	N	See <i>Buffer Requests</i>
<b>Reserved</b>	0x2-0x3	-	-	-	-	Reserved Response OpCodes
<b>LDM 1 Read</b>	0x4	Y	O	R	Y	LDM 1 Read Request. See <i>Read and Read Response</i>
<b>Buffer Put</b>	0x5	N	O	W	Y	See <i>Buffer Requests</i>
<b>Buffer Get</b>	0x6	N	O	R	Y	See <i>Buffer Requests</i>
<b>Buffer Putv</b>	0x7	N	O	W	Y	See <i>Buffer Requests</i>
<b>Buffer Getv</b>	0x8	N	O	R	Y	See <i>Buffer Requests</i>
<b>Signaled Buffer Put</b>	0x9	N	O	W	Y	See <i>Buffer Requests</i>
<b>Signaled Buffer Get</b>	0xA	N	O	W	Y	See <i>Buffer Requests</i>
<b>Dynamic Buffer Put</b>	0xB	N	O	W	Y	See <i>Buffer Requests</i>
<b>Dynamic Buffer Get</b>	0xC	N	O	R	Y	See <i>Buffer Requests</i>
<b>Signaled Dynamic Buffer Put</b>	0xE	N	O	W	Y	See <i>Buffer Requests</i>
<b>Signaled Dynamic Buffer Get</b>	0xF	N	O	W	Y	See <i>Buffer Requests</i>
<b>Dynamic Buffer Allocate</b>	0x10	N	O	W	Y	See <i>Buffer Requests</i>
<b>Dynamic Buffer Release</b>	0x11	N	O	W	Y	See <i>Buffer Requests</i>
<b>Reserved</b>	0x12-0x1F	-	-	-	-	Reserved Request OpCodes

#### 4.2.7. Advanced 1 OpClass Operation Codes

Table 4-12: Advanced 1 OpClass OpCodes

OpCode Name	OpCode Encoding	I D	M O	R K	N D	Description
<b>Pattern Response</b>	0x0	-	O	N	-	See <i>Pattern Requests</i>
<b>Lightweight Notification Response</b>	0x1	-	-	-	-	See <i>Lightweight Notification (LN)</i>
<b>Reserved</b>	0x2-0x3	-	-	-	-	Reserved Response OpCodes
<b>Pattern Set</b>	0x4	N	O	W	Y	See <i>Pattern Requests</i>
<b>Pattern Count</b>	0x5	N	O	R	Y	See <i>Pattern Requests</i>
<b>Pattern Match</b>	0x6	N	O	R	Y	See <i>Pattern Requests</i>
<b>Reserved</b>	0x7-0x1E	-	-	-	-	Reserved Request OpCodes
<b>Lightweight Notification Request</b>	0x1F	-	O	W	N	See <i>Lightweight Notification (LN)</i>

#### 4.2.8. Advanced 2 OpClass Operation Codes

Table 4-13: Advanced 2 OpClass OpCodes

OpCode Name	OpCode Encoding	I D	M O	R K	N D	Description
<b>Encapsulation Response</b>	0x0	N	O	N	N	See <i>Unicast Encapsulation Packets</i> . Contains an encapsulated response packet.
<b>Precision Time Response</b>	0x1	-	O	N	N	See <i>Precision Time</i>
<b>MACK</b>	0x2	-	O	N	N	Reliable Multicast Acknowledgment—See <i>Reliable Multicast Acknowledgment</i>
<b>Reserved</b>	0x3	-	-	N	-	Reserved Response OpCodes
<b>Encapsulation Request</b>	0x4	Y	O	N	N	See <i>Unicast Encapsulation Packets</i> . Contains an encapsulated request packet.
<b>Reserved</b>	0x7-0x1E	-	O	-	Y	Reserved Request OpCodes
<b>Precision Time Request</b>	0x1F	-	O	N	N	See <i>Precision Time</i>

#### 4.2.9. Multicast OpClass Operation Codes

Table 4-14: Multicast OpClass OpCodes

OpCode Name	OpCode Encoding	I D	M O	R K	N D	Description
Reserved	0x0-0x1	-	-	-	-	Reserved Response OpCodes
Vendor-defined Response	0x2-0x3	-	-	-	-	Vendor-defined multicast response operations
Unreliable Encapsulation Request	0x4	-	-	-	-	Encapsulation Request—See <i>Multicast</i>
Write	0x5	Y	O	W	N	Write Request—See <i>Multicast</i>
Reliable Encapsulation Request	0x6	-	-	-	-	Encapsulation Request—See <i>Multicast</i>
Write Persistent	0x7	Y	O	W	N	Write Persistent Request—See <i>Multicast</i> Shall be applicable only to reliable multicast groups.
Reserved	0x8-0x19	-	-	-	-	Reserved Request OpCodes
Vendor-defined	0x1A-0x1D	-	-	-	-	Vendor-defined multicast request operations
Unreliable Write MSG	0x1E	Y	O	N	Y	Unreliable Write MSG Request—See <i>Multicast</i>
Write MSG	0x1F	Y	O	N	Y	Write MSG Request—See <i>Multicast</i>

#### 4.2.10. SOD OpClass Operation Codes

Table 4-15: SOD OpClass OpCodes

OpCode Name	OpCode Encoding	I D	M O	R K	N D	Description
SOD ACK	0x0	-	O	N	N	See <i>SOD Standalone Acknowledgment</i>
SOD Read Response	0x1	-	O	N	N	See <i>SOD Read Request</i>
SOD Encapsulated Response	0x2	-	-	-	-	See <i>SOD Encapsulation</i> .
Reserved	0x3	-	-	-	-	Reserved Response OpCode
SOD Write	0x4	Y	O	W	N	See <i>SOD Write</i>

OpCode Name	OpCode Encoding	I D	M O	R K	N D	Description
SOD Sync	0x5	-	-	-	-	See <i>SOD Sync</i>
SOD Write Persistent	0x6	Y	O	W	N	See <i>SOD Write</i>
Reserved	0x7	-	-	-	-	Reserved Request OpCode
SOD Interrupt	0x8	-	O	W	N	See <i>SOD Interrupt</i>
Reserved	0x9-0x1A	-	-	-	-	Reserved Request OpCodes
SOD Read	0x1B	Y	O	R	N	See <i>SOD Read Request</i>
SOD Encapsulated Request	0x1C		O	N	N	See <i>SOD Encapsulation</i>
Reserved	0x1D-0x1F	-	-	-	-	Reserved Request OpCodes

#### 4.2.11. CTXID OpClass Operation Codes

Table 4-16: CTXID OpClass OpCodes

OpCode Name	OpCode Encoding	I D	M O	R K	N D	Description
Reserved	0x0	-	-	-	-	Reserved Response OpCode
Translation RSP	0x1	-	O	N	-	See <i>Translation Response</i>
Translation Invalidate RSP	0x2	-	O	N	-	See <i>Translation Invalidate Response</i>
Collective Response	0x3	-	O	N	-	See <i>Collective Operations</i>
Translation Req	0x4	Y	O	Y	Y	See <i>Translation Request</i>
PRG Req	0x5	Y	O	N	Y	See <i>PRG Request</i>
PRG Response Notification	0x6	-	O	N	N	See <i>PRG Response Notification</i>
Translation Invalidate Req	0x7	Y	O	N	Y	See <i>Translation Invalidate Request</i>
Stop Marker Req	0x8	Y	O	N	N	See <i>Stop Marker</i>
PRG Release	0x9	Y	O	N	N	See <i>PRG Release</i>
PRG Eviction Notification	0xA	Y	O	N	N	See <i>PRG Eviction Notification</i>
Collective Request	0xB	N	O	W	Y	See <i>Collective Operations</i>

OpCode Name	OpCode Encoding	I D	M O	R K	N D	Description
Reserved	0xC-0x1C	-	-	-	-	Reserved Request OpCodes
CTXID NIR Request	0x1D	Y	O	N	N	See <i>Non-Idempotent Request Release (NIRR)</i>
Unreliable Write MSG	0x1E	Y	O	N	N	See <i>Write MSG</i>
Write MSG	0x1F	Y	O	N	N	See <i>Write MSG</i>

## 4.3. End-to-end OpClasses and OpCode Sets

To establish operation-level interoperability, a component communicates its supported OpClasses and OpCode sets through the *OpCode Set Structure*. Prior to enabling peer components to communicate, software examines the supported OpClasses and OpCode sets and enables only those that are commonly supported by the components. This is accomplished as follows:

For each OpClass, there are two corresponding OpCode sets defined—supported and enabled. The supported OpCode set is a bit vector that indicates the supported OpCode within a given class and the enabled OpCode set is a bit vector that indicates the enabled OpCode after a component has been initialized.

- 5 For a given OpClass, each bit within an OpCode set vector is defined as follows:
- Within the supported OpCode vector, Bit<sub>k</sub> = 0b indicates the corresponding OpCode is unsupported; Bit<sub>k</sub> = 1b indicates support.
  - Within the enabled OpCode vector, Bit<sub>k</sub> = 0b indicates the corresponding OpCode is not enabled; Bit<sub>k</sub> = 1b indicates enabled.
- 10 Within the supported and enabled bit vectors, bits corresponding to mandatory OpCode within a given OpCode class shall be hardwired to 1b. Individual optional OpCode may be disabled if the request or response is not required by a given solution or if software detects a given supported Bit<sub>k</sub> is not consistently set across all components.

15 Additional details are found in section *OpCode Set Structure*.

## 5. Base Formats for End-to-end Packets

This section describes the general packet formats used by different end-to-end OpClasses. Individual request and response packets may use minor variations of these formats such as an overlay of OpCode-specific fields to meet request or response packet-specific needs. These variations are covered within each request and response packet's description in subsequent sections.

### 5.1. End-to-end General Packet Protocol Format

Although packet formats vary by OpCode, the most commonly used OpCodes cover Read and Write requests and response packets. These request and response packets are supported in all packet formats. The following figures illustrate generic packet format versions associated with specific OpClasses. Although these packet formats share a number of common fields, field presence and attributes vary.

An end-to-end packet shall be transmitted starting at bit 0 and consecutively transmitted through bit N (the last bit in the packet).

A subset of bit positions in every end-to-end and link-local packet are protected by a separate CRC referred to as the PCRC. The protected bit locations are identical in all packet formats and are distributed to improve burst error and lane failure detection. For clarity, the bits protected by the PCRC and the PCRC itself are colored orange. The fields protected by the PCRC vary by packet type.

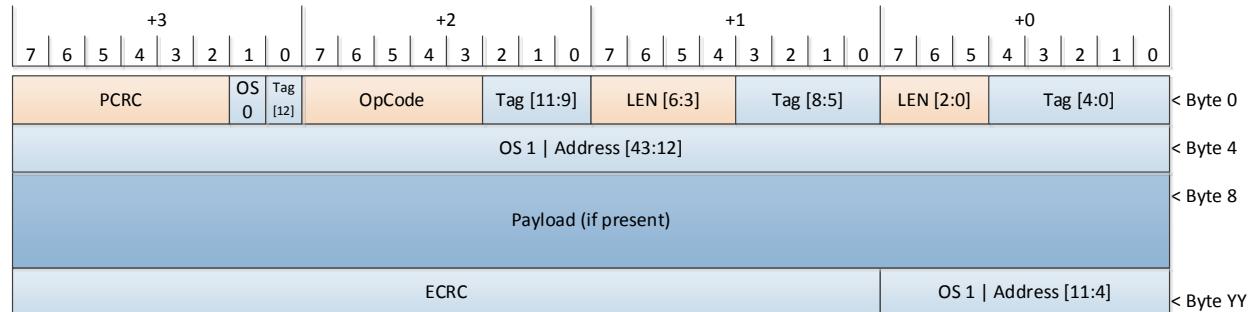


Figure 5-1: P2P-Core Request Generic Format

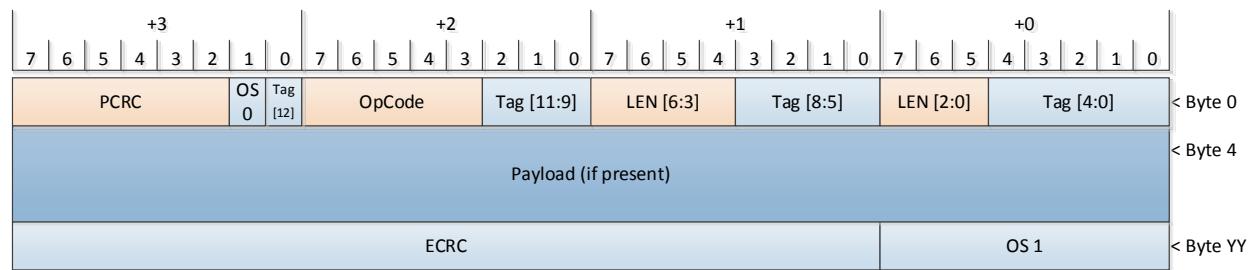
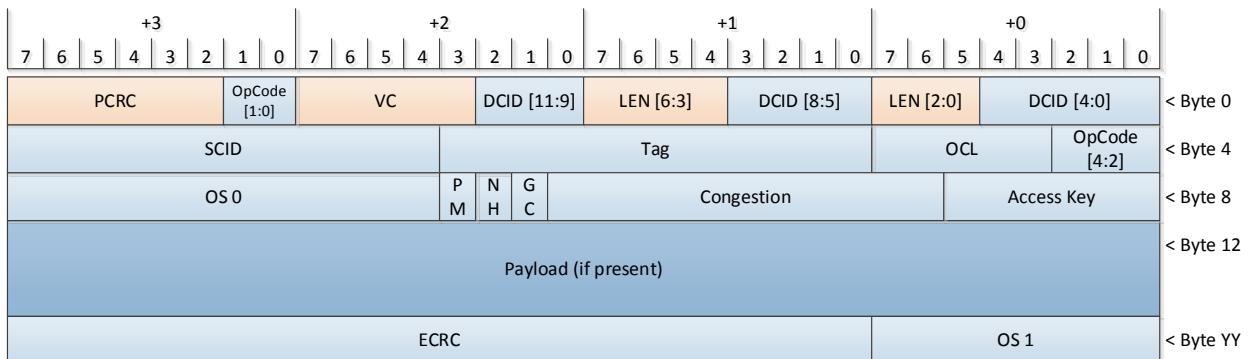
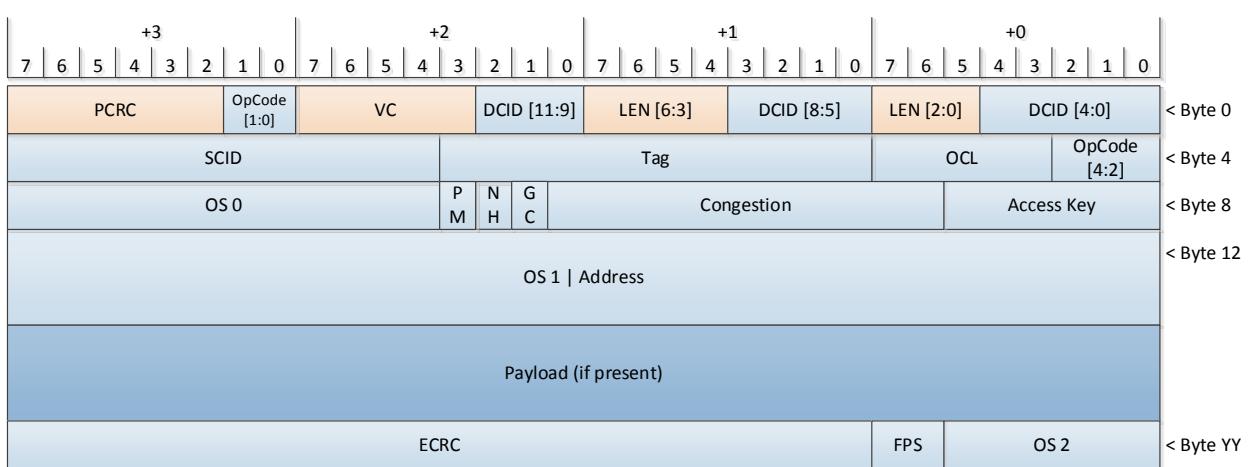
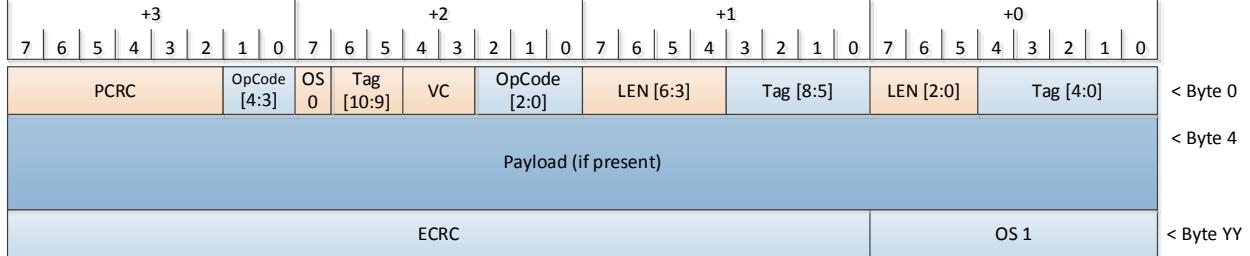
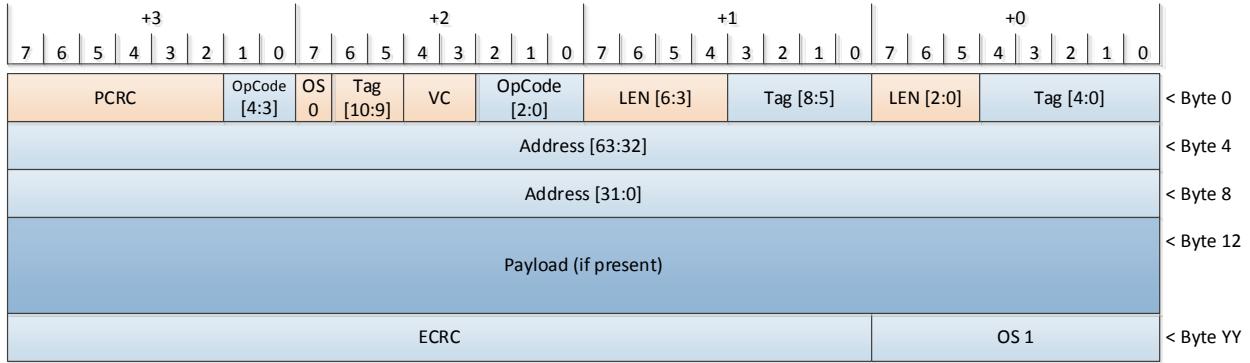


Figure 5-2: P2P-Core Response Generic Format



Unless explicitly stated otherwise by this specification, to maximize interoperability, all end-to-end packet formats shall contain the protocol fields defined in *Common End-to-end Packet Protocol Fields*.

Table 5-1: Common End-to-end Packet Protocol Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>Length</b>	LEN	7	<p>Packet length including protocol plus payload (if present) is calculated as:</p> $\text{Packet length} = \text{LEN} * 4 \text{ bytes}$ <p>Maximum end-to-end packet length is 504 bytes.</p> <p>If Length <math>\neq 0x7F</math>, then the packet is an end-to-end packet.</p> <p>If Length == 0x7F, then the packet is a link-local packet.</p>
<b>Operation Code</b>	OpCode / OC	5	Field indicates the packet's request or response type
<b>Prelude CRC</b>	PCRC	6	<p>Integrity field that protects select protocol fields that determine packet type and length. The protected fields vary by packet type but are always at the same bit locations.</p> <p>PCRC shall be dynamically generated at the source component, and validated at the destination component. PCRC shall be validated at each intermediate link receiver interface. See <i>Prelude CRC (PCRC)</i>.</p>
<b>Payload</b>	-	-	OpCode-specific payload if present.
<b>Pad CNT</b>	-	2	<p>The Pad CNT field shall reflect the number of pad bytes (0-3) appended to the payload to ensure 4-byte packet alignment.</p> <p>Each pad byte shall be Reserved.</p> <p>Pad bytes shall not be written to the destination resource.</p> <p>If a packet contains multiple Pad CNT fields, then Pad CNT [n] shall correspond to field [n].</p> <p>For Atomic request packets that contain multiple operands and a single Pad CNT field, then the Pad CNT field shall apply to all operands.</p>
<b>End-to-end CRC</b>	ECRC	24	<p>End-to-end Packet Data integrity field for P2P-Core / explicit OpClass packets</p> <p>ECRC shall be dynamically generated at the source component, and validated at the destination component. See <i>End-to-end CRC (ECRC)</i>.</p>

Field Name	Field Abbreviation	Size (bits)	Description
LPD Indicator	LP	1	<p>Indicates if the <i>LPD Field</i> is present within the packet</p> <p>0b—Not Present 1b—Present</p> <p>If LP == 1b in a received request packet, then the corresponding response packet shall set LP = 1b and contain the relevant LPD-specific fields.</p>
OpCode-specific Reserved	OS N	-	OpCode-specific field
	R[n]	-	<p>Reserved</p> <p>A Reserved protocol field shall be transmitted as zero and ignored upon receipt.</p>

Table 5-2: P2P-Core Packet Protocol Fields

Field Name	Field Abbreviation	Size (bits)	Description
Address	-	Varies	<p>The Address field identifies byte 0 of the indicated logical bank.</p> <p>Some request packets support a 44-bit effective address with integer 16-byte resolution. Other request packets support a 44-bit address with integer 1-byte resolution.</p>
Tag	-	13	Tag is an identifier used to associate a request with a response or acknowledgment, e.g., a Read Request with a Read Response or a Write Request with a <i>Standalone Acknowledgment</i> .
	-	7	Field indicates the packet's request or response sub-operation type. Sub-OPC is present when the packet's OpCode is set to SubOp 1 Response or SubOp 1 Request.
Spatial Locality	S	1	<p>A Requester that supports P2P-Core Read Offset and Write Offset operations shall use this field to inform the Responder when to cache or release the resource associated with a given logical row. For example, if a Responder supports a 1 KiB logical row size, then its media is logically partitioned into integer 1 KiB subranges. Request packets that target the same 1 KiB subrange are referred to as being spatially local.</p> <p>A Responder communicates the logical row size in the <i>Responder Bank Structure</i>.</p>

Field Name	Field Abbreviation	Size (bits)	Description
			<p>A Requester uses the logical row size configured in the <i>Requester Bank Structure</i> that corresponds to the Responder.</p> <p>This field is present only in P2P-Core Read, Write, Read Offset, and Write Offset request packets. See <i>Read and Read Response</i> and <i>Write</i> for additional details on how to use this field.</p> <p>0b—If the subrange is presently cached, then the Responder shall release the cached subrange.</p> <p>1b—If the subrange is not cached, then the Responder shall cache the targeted subrange. If already cached, then the Responder shall take no spatial locality action.</p>

Table 5-3: P2P-Coherency Packet Protocol Fields

Field Name	Field Abbreviation	Size (bits)	Description
Virtual Channel Address Tag Sub-OPC	VC	2	Virtual Channel Identifier
	Address	64	The Address field identifies byte 0 of the destination memory range.
	Tag	11	Tag is an identifier used to associate a request with a response or acknowledgment, e.g., a Read Request with a Read Response or a Write Request with a <i>Standalone Acknowledgment</i> .
	-	7	Field indicates the packet's request or response sub-operation type. Sub-OPC is present when the packet's OpCode is set to SubOp 1 Response or SubOp 1 Request.

Table 5-4: Common Explicit OpClass End-to-end Packet Protocol Fields

Field Name	Field Abbreviation	Size (bits)	Description
Virtual Channel Destination CID Address	VC	5	Virtual Channel Identifier
	DCID	12	CID of the component that this packet is intended for within the present subnet.
	-	-	The DCID within a request packet shall be reflected in the SCID in the corresponding response packet (if applicable).
	-	-	The Address field identifies byte 0 of the target data.

Field Name	Field Abbreviation	Size (bits)	Description
Payload			Address field size is determined by the specific request type.
	-	-	<p>OpCode-specific payload.</p> <p>Payload length is fixed (OpCode-specific) or variable. Unless explicitly stated otherwise by this specification, the payload length (PL) is calculated as follows:</p> $PL = \text{Packet Length} - (\text{total protocol bytes}) - \text{Pad CNT}$
Forward Progress Screen	FPS	2	<p>If the request packet is a retransmitted request packet due to a <i>Responder Not Ready (RNR)</i> NAK, then this indicates the FPS Epoch. See <i>Forward Progress Screen (FPS)</i>.</p> <p>0x0—No Epoch 0x1—Epoch 0 0x2—Epoch 1 0x3—Reserved</p> <p>If the request packet format is used for reliable and unreliable versions of an operation, then the FPS field shall be Reserved in the unreliable version.</p>
Tag	-	12	Tag is an identifier used to associate a request packet with a response packet, e.g., a Read Request with a Read Response or a Write Request with a <i>Standalone Acknowledgment</i> .
Source CID	SCID	12	<p>CID of the component that initially injected this packet into the present subnet.</p> <p>The SCID within a request packet shall be reflected in the DCID in the corresponding response packet (if applicable).</p>
OpClass Label	OCL	5	Operation Class Label identifies the explicit OpClass required to decode the OpCode field and packet format –See <i>End-to-end Operation Classes</i>
GC	-	1	<p>The GC bit indicates if the <i>Explicit OpClass Multi-subnet Field</i> is present in the packet.</p> <p>0b—<i>Explicit OpClass Multi-subnet Field</i> is not present 1b—<i>Explicit OpClass Multi-subnet Field</i> is present</p>
Next Header	NH	1	<p>The NH bit indicates if the <i>Explicit OpClass Next Header Field</i> is present in the packet.</p> <p>0b—Next Header field is not present 1b—Next Header field is present</p>

Field Name	Field Abbreviation	Size (bits)	Description
R-Key Indicator	RK	1	<p>Indicates if the packet contains the <i>Region Key (R-Key)</i> field</p> <p>0b—R-Key field is not present 1b—R-Key field is present</p>
Access Key	AKey	6	See <i>Access Keys</i> .
Congestion	-	10	See <i>Congestion Management</i>
Translated Address	TA	1	<p>Indicates if the address field(s) within a request packet was previously translated by a <i>Translation Request</i>. This field is present in select request packet formats.</p> <p>0b—Untranslated Address 1b—Translated Address</p>
Performance Marker	PM	1	<p>The Performance Marker (PM) bit is used to indicate whether to gather performance information relative to this request or response packet. For example, a switch, TR, Requester, or Responder may be capable of gathering performance information on a per packet basis. If capable and the packet's PM == 1b, then it will collect the relevant packet-specific data. The collected data is gathered (in-band or out-of-band) and processed by a performance management tool. See <i>Core Structure Component CAP 2 Performance Marker Support</i> and <i>Core Structure Component CAP 2 Control Performance Marker Enable</i> fields.</p> <p>A Responder shall reflect a request packet's PM bit in the corresponding response packet.</p> <p>0b—Do Not Collect Data 1b—Collect Data</p>
Unreliable	UR	1	<p>If no error is detected, then this indicates if the request packet is to be acknowledged</p> <p>0b—Acknowledged 1b—Not acknowledged</p> <p>If UR == 1b, then shall set PU = 0b.</p>

Table 5-5: Common OpCode-specific Bits

Field Name	Field Abbreviation	Size (bits)	Description
Target Cache	TC	1	Target Cache—If the Responder supports a cache, then this bit indicates whether to place the packet's payload (if present) in the cache and to invoke Responder-specific coherency actions

Field Name	Field Abbreviation	Size (bits)	Description
			<p>to access the resource. If the Responder does not support a cache, then this bit shall be ignored.</p> <p>0b—Payload may be placed within a non-cache Responder resource</p> <p>1b—Payload shall be placed within the Responder’s cache prior to generating a response packet. Placing data within the cache resource shall make the data visible within the Responder’s cache coherency protocol. If the request packet does not contain a payload field or the payload size is zero and the request packet’s Address is associated with a cached resource, then the Responder shall perform the Responder-specific coherency actions to access the resource. If the Requester is not permitted to modify the cached resource or the Responder does not contain the Home Agent responsible for the cached resource, then the Responder shall handle this as a MP error.</p> <p>If TC == 1b, then shall set NS = 0b.</p>
Release Cache	RC	1	<p>Release Cache—If the Responder supports a cache, the request packet’s Address is associated with a cached resource, and the Requester had exclusive access, then upon successfully validating and executing the request packet, the Responder shall perform the Responder-specific coherency actions to release exclusive access to the resource. If none of these conditions are true, then this bit shall be ignored.</p> <p>0b—No Cache Impact 1b—Release Cached Resource</p>
No Snoop	NS	1	<p>No Snoop</p> <p>If the packet contains payload, then this bit is interpreted as follows:</p> <p>0b—If supported by the destination component, cache coherency shall be applied to the packet’s payload.</p> <p>1b—If supported by the destination component, cache coherency may be applied to the packet’s payload.</p>
Persistence Update	PU	1	<p>Indicates if persistence is required prior to the request packet being acknowledged or a request-specific response packet being returned.</p> <p>0b—Not Persistent 1b—Persistent</p> <p>If PU == 1b, then shall set UR = 0b.</p>

## 5.2. Tag Field

A Tag is an identifier used to correlate request packets with response packets. For example, a Tag is used to correlate read request packets with Read Response packets or a write request packet with an acknowledgment packet.

5 The following are the features and requirements associated with Tags:

- A Tag is associated with a given Tag space.
  - P2P-Core packets support a  $2^{13}$  Tag space. Each Requester interface represents a distinct Tag Space.
    - If a request packet has a corresponding response packet (e.g., a Read Request-Read Response), then there shall be one Outstanding Request Packet with a given Tag on an interface.
  - P2P-Coherency packets support a  $2^{11}$  Tag space. Each Requester interface represents a distinct Tag Space.
    - If a request packet has a corresponding response packet (e.g., a Read Request-Read Response), then there shall be one Outstanding Request Packet with a given Tag on an interface.
  - Explicit OpClass packets support a  $2^{12}$  Tag space per configured CID or GCID (components that support multiple CIDs or GCIDs support multiple Tag spaces).
    - For each additional configured CID or GCID, a component shall support an additional Tag Space.
    - Packet uniqueness is delineated by the packet's [SCID, DCID, Tag] or [SGCID, DGCID, Tag].
    - The Tag within a request packet that does not require a response packet may be re-used in multiple request packets (e.g., unreliable request packets).
    - If the request packet is Non-idempotent Request (NIR), then the Tag shall not be used in another request packet until a *Standalone Acknowledgment* is received that corresponds to the *Non-Idempotent Request Release (NIRR)* packet which completes the operation.
    - That Tag within an Outstanding Request Packet shall not be re-used.
    - If a request packet does not require a response packet and the Tag correlates to a single multi-request-packet operation (e.g., an Unreliable Write MSG), then the Requester should cycle through multiple Tag values for a given [SCID, DCID, Tag] or [SGCID, DGCID, Tag] before reusing the Tag value. The minimum number of unique Tag values the Requester will guarantee to use before reusing a Tag is indicated by *Core Structure Min Tag Cycle*.
  - A Requester may support any non-empty subset of the Tag space. The number of supported Tags has no impact on interoperability.
  - Within each Tag Space, the Tag within a request packet that requires a response packet shall not be reused in a request packet that does not require a response packet. Similarly, the Tag within a request packet that does not require response packet shall not be reused in a request packet that does require a response packet.
- If a component interface supports the P2P-Core OpClass, then the Tag used in request packets transmitted on this interface shall be encoded as specified in *P2P-Core Encoded Structures*.
- Unless explicitly stated otherwise by this specification, Tags used in P2P-Core Coherency and explicit OpClass packets shall have no specified meaning; all Tags shall be available for use and all

5 Responders shall support the entire Tag space. Requester Tag management is outside of this specification's scope.

- The Tag within a request packet shall be reflected in the corresponding response packet.
- Some explicit OpClass request packets may require multiple response packets to fulfill. Each response packet shall reflect the request packet's Tag.

10 **Developer Note:** *If the specification does not require the Tag to be encoded, then implementation-specific encoding can be used. For example, a Tag can be encoded to steer request packets to a given egress interface or to steer response packets to a given protocol engine instance. Steering can be accomplished by applying a N-bit mask to the upper Tag bits to select an egress interface or protocol engine, while leaving the lower bits to identify the specific request processing resource. Such encoding has no impact on interoperability as Responders are required to support the entire Tag space.*

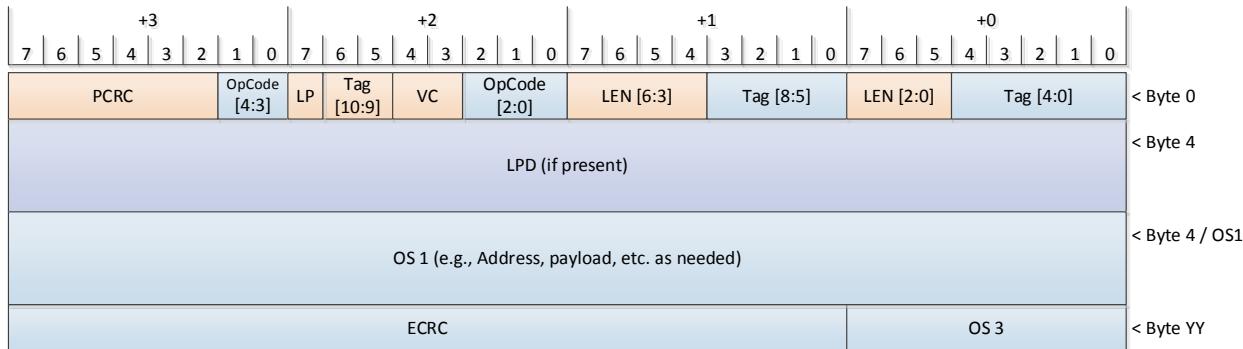
### 5.3. LPD Field

15 The LP bit is present in select request and response packets. This field indicates whether an additional LPD protocol field is included within the packet. The LPD field provides additional information required to support *Logical PCI Device (LPD)* and *Logical PCI Hierarchy (LPH)* communication.

20 The following are the features and requirements associated with the LPD field:

- The LPD field may be present only in request and response packets that contain the LP bit.
  - A Requester or a Responder may include an LPD field only if the component supports *Logical PCI Device (LPD)* or *Logical PCI Hierarchy (LPH)* communications.
    - If LP == 1b, then the packet shall contain a LPD field.
    - If LP == 0b, then the packet shall not contain a LPD field.
  - A component shall validate the LP and LPD fields only if *Logical PCI Device (LPD)* or *Logical PCI Hierarchy (LPH)* communication is enabled.
  - If a component does not support LPD communication, then LPD or LPH field validation shall not be performed irrespective of the presence and setting of a packet's LP bit.
- *Logical PCI Device (LPD)* and *Logical PCI Hierarchy (LPH)* support and enablement are managed through the *Core Structure Component CAP 2* and *Core Structure Component CAP 2 Control*.
- For P2P-Coherency OpClass packets, the LP and LPD fields shall be located as illustrated in *P2P-Coherency LPD field Location*.
- For explicit OpClass packets, the LP and LPD fields shall be located as illustrated in *Explicit OpClass LPD field Location*.
- LPD field format and size are determined by the LPD Type protocol field.
  - If LPD Type == 0x0, then the LPD shall use the *LPD Field Formats* LPD Type = 0x0 format.
  - If LPD Type == 0x1, then the LPD shall use the *LPD Field Formats* LPD Type = 0x1 format.
  - If LPD Type == 0x2, then the LPD shall use the *LPD Field Formats* LPD Type = 0x2 format.
  - If LPD Type == 0x3, then the LPD shall use the *LPD Field Formats* LPD Type = 0x3 format.
- If *Core Structure Component CAP 2 Host LPD Communications Support* == 1b or the component supports peer-to-peer LPD communications, then the component shall support the following LPD field formats: LPD Type 0, LPD Type 1, and LPD Type 2.
- If *Core Structure Component CAP 2 Host LPD Communications Support* == 0b, then the component shall support the following LPD field formats LPD Type 0, and may support LPD Type 1 and LPD Type 2.

- *Core Structure Component CAP 2 LPD Field Type 3 Support* indicates if a component supports LPD field format LPD Type 3, and *Core Structure Component CAP 2 Control LPD Field Type 3 Enable* determines if this format can be used.



5

Figure 5-7: P2P-Coherency LPD field Location

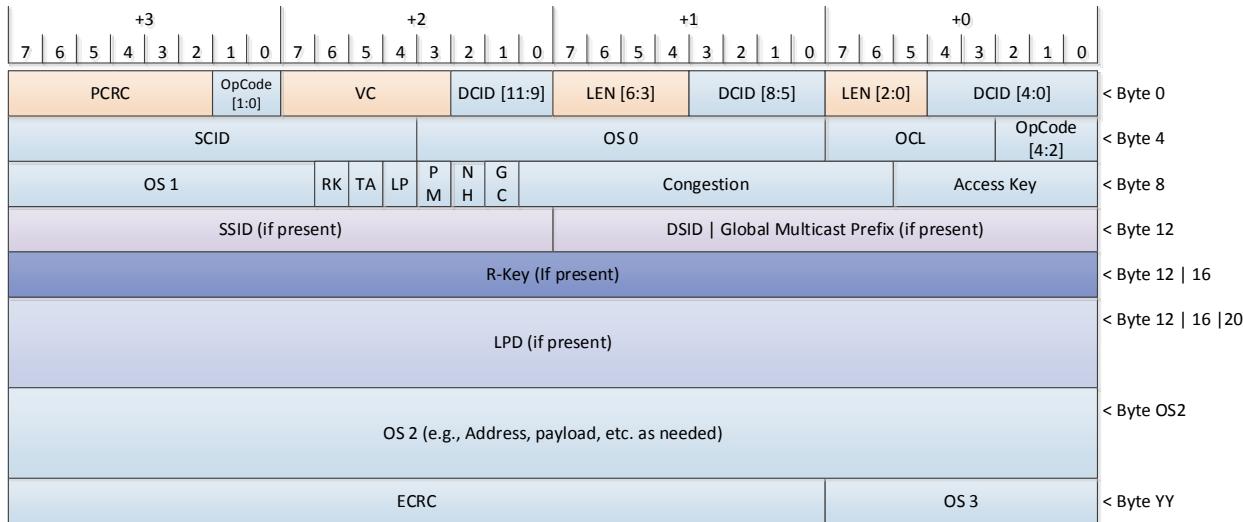


Figure 5-8: Explicit OpClass LPD field Location

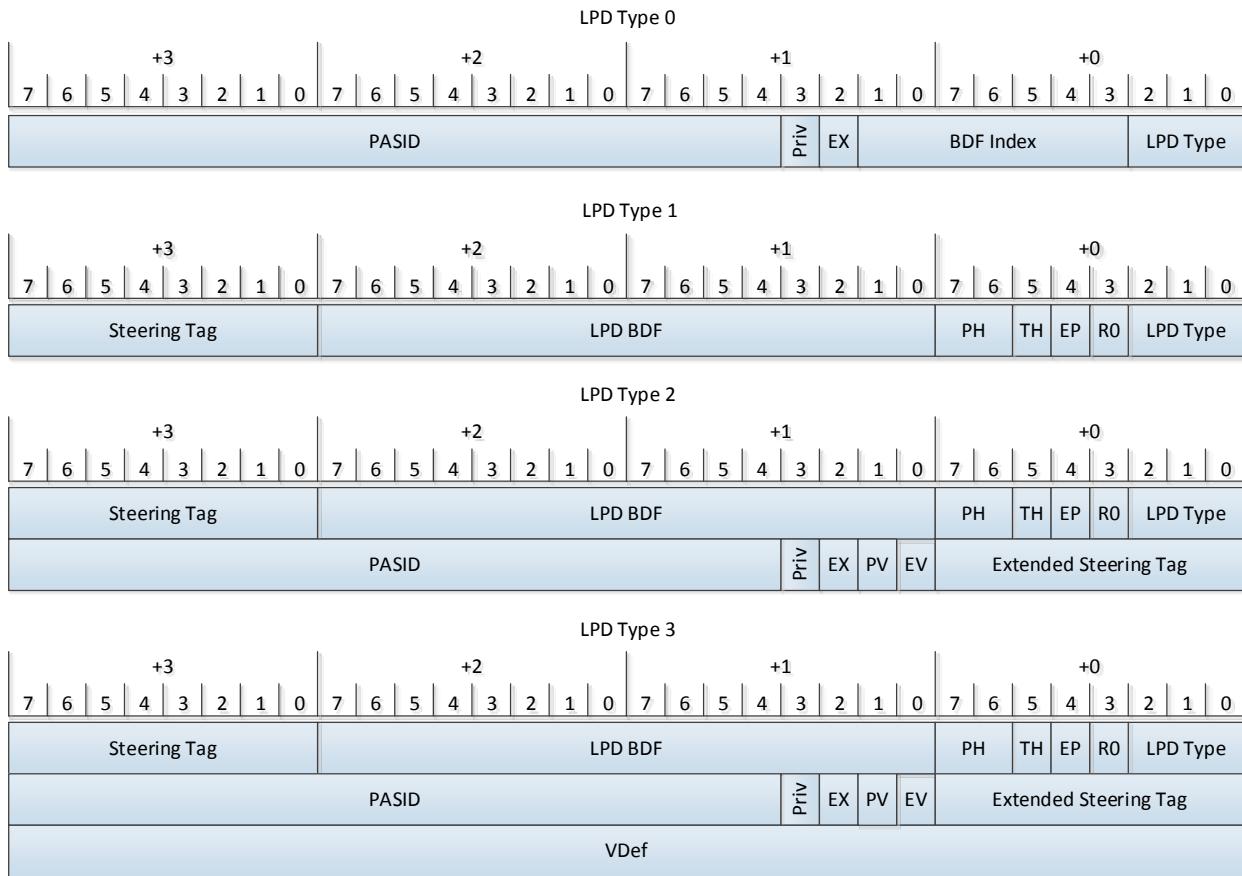


Figure 5-9: LPD Field Formats

Table 5-6: LPD-specific Fields

Field Name	Size (bits)	Description
<b>LPD Type</b>	3	Indicates the LPD field type. 0x0—LPD Type 0 0x1—LPD Type 1 0x2—LPD Type 2 0x3—LPD Type 3 0x4-0x7—Reserved
<b>BDF Index</b>	7	Used to linearly index the <i>Core LPD BDF Table</i> to locate the BDF value associated with this LPD packet.
<b>PV</b>	1	Indicates if the PASID field contains a valid value. 0b—Invalid 1b—Valid
<b>EX</b>	1	The PASID Execute Requested bit shall be as specified in the <i>PCI Express Base Specification</i> .

Field Name	Size (bits)	Description
<b>Priv</b>	1	The PASID Privileged Mode Requested bit shall be as specified in the <i>PCI Express Base Specification</i> .
<b>PASID</b>	20	The PASID field shall be as specified in the <i>PCI Express Base Specification</i> . If PV == 0b, then the PASID field shall contain a valid PASID, else this field shall be Reserved.
<b>EP</b>	1	EP (Error Poisoned) field shall be as specified in the <i>PCI Express Base Specification</i> .
<b>TH</b>	1	The TLP Hints bit shall be as specified in the <i>PCI Express Base Specification</i> .
<b>PH</b>	2	PH (Processing Hints) field shall be as specified in the <i>PCI Express Base Specification</i> .
<b>LPD BDF</b>	16	<p>Concatenation of the function's [bus number, device number, function number], which is also referred to as a BDF.</p> <p>If the function does not support ARI (Alternative Routing ID) or it is not enabled, then the LPD BDF field shall be as follows:</p> <ul style="list-style-type: none"> <li>• Byte 0, bits [2:0] shall contain a 3-bit function number.</li> <li>• Byte 0, bits [7:3] shall contain a 5-bit device number.</li> <li>• Byte 1 shall contain an 8-bit bus number.</li> </ul> <p>If the function supports ARI and it is enabled, then the LPD BDF field shall be as follows:</p> <ul style="list-style-type: none"> <li>• Byte 0 shall contain an 8-bit function number.</li> <li>• Byte 1 shall contain an 8-bit bus number.</li> </ul> <p>In a request packet, the LPD BDF field contains the Requester ID.</p> <p>In a response packet, the LPD BDF field contains the Completer ID.</p>
<b>Steering Tag</b>	8	Steering Tag field shall be as specified in the <i>PCI Express Base Specification</i> .
<b>Extended Steering Tag</b>	8	The Extended Steering Tag field is equivalent to the TPH TLP Prefix ST (15:8) sub-field, and shall be as the sub-field is specified in the <i>PCI Express Base Specification</i> .
<b>EV</b>	1	Indicates if the Extended Steering Tag field contains a valid value. 0b—Invalid 1b—Valid
<b>VDef</b>	32	If present, then the <i>Core Structure</i> C-UUID is used to identify the contents of this field.
<b>R0</b>	1	Reserved

## 5.4. Explicit OpClass R-Key Field

The RK bit is present in select explicit OpClass request packets. This field indicates whether an additional R-Key protocol field is included within the packet. An R-Key provides fine-grain resource access control key to ensure erroneous software or hardware does not incorrectly access the underlying resource.

The following are the features and requirements associated with the R-Key field:

- The R-Key field may be present only in explicit OpClass request packets that contain the RK bit.
  - A Requester may Set a request packet's RK bit and include an R-Key field only if the component supports and enables R-Keys.
    - If RK == 1b, then the request packet shall contain an R-Key field, and the R-Key field shall contain the R-Key value associated with the target page.
    - If RK == 0b, then the request packet shall not contain an R-Key field.
  - A Responder shall validate the RK and R-Key fields if R-Keys are supported and enabled. A Responder shall validate that the request packet's R-Key field matches the R-Key configured for the page identified by the packet's Address field.
    - If the page's R-Key is the Default R-Key, then R-Key packet validation shall not be performed.
    - If the page's R-Key field is not the Default R-Key, then R-Key packet validation shall be performed.
      - If a request packet does not contain the RK bit or the RK == 0b, and the page is protected by a non-Default R-Key, then packet R-Key validation shall fail.
  - If a Responder does not support or R-Key validation is not enabled, then R-Key packet validation shall not be performed irrespective of the presence or setting of a request packet's RK bit.
- R-Key management is outside of this specification's scope.
  - Requesters and Responders coordinate page addresses and associated R-Keys using policies and methods outside of this specification's scope.
- *Core Structure Component CAP 2 R-Key Support* and *Core Structure Component CAP 2 Control R-Key Enable* are used to indicate support and enable R-Key functionality.
- If a component supports a ZMMU, then see *Memory Management* for additional details on ZMMU and R-Key interactions.

The RK and R-Key fields shall be located as illustrated in *Conditional Field Locations within an Explicit OpClass Packet*. If GC == 0b, then the R-Key field shall be located as byte 12 instead of byte 16.

## 35 5.5. Explicit OpClass Next Header Field

The NH bit is present in explicit OpClass packets. This field indicates whether the Next Header protocol header is included within the packet. The purpose of the Next Header field is configurable, and allows multiple semantics to be attached based on solution needs, e.g., the Next Header field can carry timestamps or security information to enable components to authenticate that the packet was transmitted by the authorized source and was not tampered with during transit.

The following are the features and requirements associated with the Next Header field if supported and enabled:

- All transmitted explicit OpClass packets shall set NH to 1b and include a Next Header field.
- The NH bit shall be located as illustrated in *Conditional Field Locations within an Explicit OpClass Packet*.
- The Next Header field shall be populated based on the enabled Next Header capabilities.
  - Management shall enable a single Next Header capability for populating and validating the Next Header[63:0] subrange. If a Next Header capability for this subrange is not enabled, then this subrange shall be Reserved.
  - Management shall enable a single Next Header capability for populating and validating the Next Header[127:63] subrange. If a Next Header capability for this subrange is not enabled, then this subrange shall be Reserved.
  - Management shall enable a single Next Header capability for populating and validating Next Header[127:0]. If enabled, management shall not enable any other Next Header capabilities.
- If the Next Header field is enabled for all explicit OpClass packets, then a Requester shall Set the NH bit and include the Next Header field in all explicit OpClass packets, and a Responder shall validate that NH == 1b and that the Next Header field is present in all OpClass packets.
- If the Next Header field is selectively enabled for Control OpClass packets, then a Requester shall set NH == 1b and include the Next Header field in all Control OpClass packets, and a Responder shall validate that NH == 1b and that the Next Header field is present in all Control OpClass packets.

## 5.6. Explicit OpClass Multi-subnet Field

The GC bit is present in all explicit OpClass packets. The GC bit indicates whether the multi-subnet field is present. The multi-subnet field contains the source subnet identifier and the destination subnet identifier / Global Multicast Prefix fields. Subnet identifiers (SID) are used to construct *Global Component Identifiers*.

If supported and enabled, then the following are the features and requirements associated with SID and the Source SID (SSID) and the Destination SID (DSID) / Global Multicast Prefix fields:

- All components within the same subnet shall share the same SID.
  - A SID shall be a value between 0 and  $(2^{16} - 1)$  inclusive.
  - Though the architecture supports up to  $2^{16}$  SIDs, multi-subnet switch components may provisioned fewer SIDs. Software shall ensure that all configured SIDs are less than or equal to the maximum number of SIDs supported by any multi-subnet switch used to relay packets between the corresponding subnets.
- The SSID and the DSID fields shall be located as illustrated in *Conditional Field Locations within an Explicit OpClass Packet*.
- The Global CID field (GC) within an explicit OpClass packet indicates if the multi-subnet fields are present within a packet. If GC == 1b, then the SSID and the DSID fields shall be present in an explicit OpClass packet.
  - The SSID and DSID fields shall be supported only if a component supports explicit communication with peer components in multi-subnet topologies. If unsupported, then the GC bit shall be set to 0b in all applicable explicit OpClass packets.
- The SSID shall identify the subnet where the source component is located, i.e., the component that first transmitted the packet (unicast or multicast). The SSID within a request packet shall be reflected in the DSID in the corresponding response packet (if applicable).

- The DSID shall identify the subnet where the destination component is located. The DSID within a request packet shall be reflected in the SSID in the corresponding response packet (if applicable).
- If the packet indicates multicast, then in place of the DSID field, the field shall contain the Global Multicast Prefix. The Global Multicast prefix is concatenated with a MGID to create a GMGID. Using a GMGID enables a multicast group to span multiple subnets. See *Multicast*.

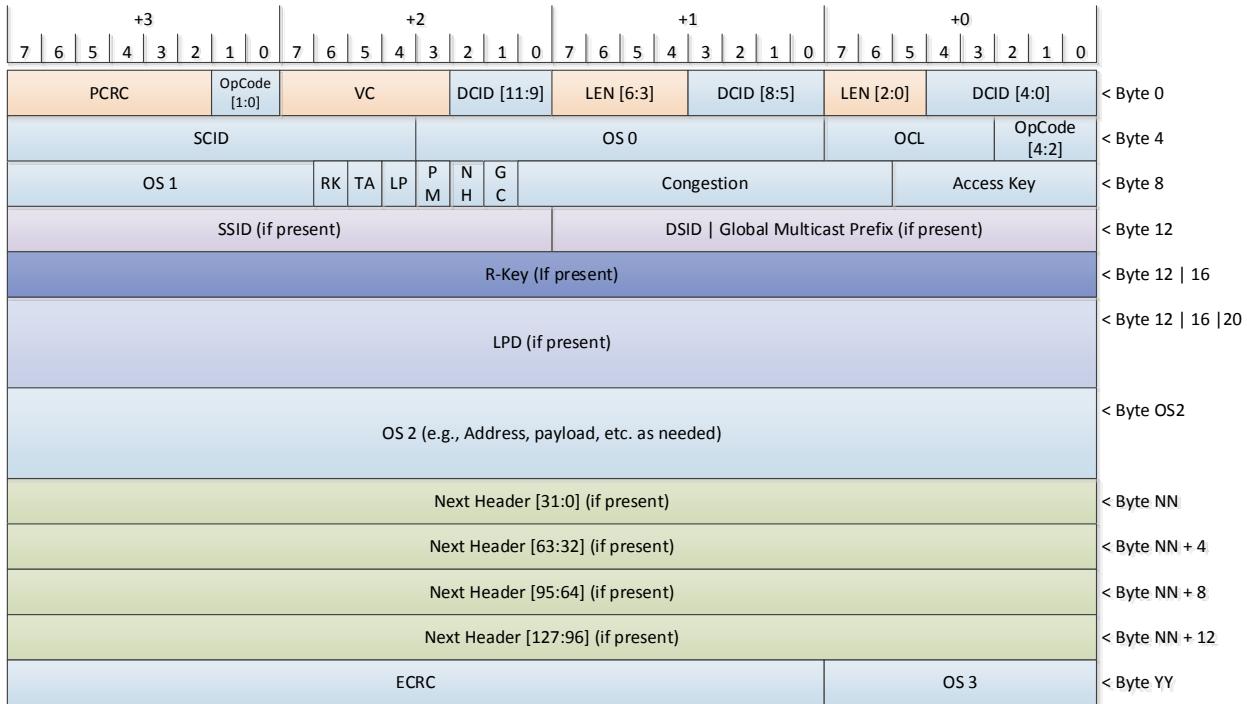


Figure 5-10: Conditional Field Locations within an Explicit OpClass Packet

**Developer Note:** The OS2 field within Conditional Field Locations within an Explicit OpClass Packet will vary by operation type and may be composed of multiple elements or sub-fields, e.g., at a minimum, the OS2 field in a Core 64 Write request packet would contain the Address field and the payload field.

# 6. Common End-to-end Operations

This section covers common end-to-end request and response packets applicable to all solutions.

## 6.1. Read and Read Response

A Requester uses a Read request packet to read N bytes of data starting at the indicated address. The Responder executes the Read request packet, and returns the data in one or more Read Response packets. *Read Request with Corresponding Read Response* illustrates an example Read Request—Read Response packet sequence between a Requester (e.g., a processor) and a Responder (e.g., a memory component):

1. The Requester generates a Read 64 request targeting address X and tags it with Tag 11.
2. The Responder decodes address X to locate the underlying physical resource.
3. The Responder moves 64 bytes of data from the underlying physical memory resource, and generates a Read Response packet.
4. The Requester decodes Tag 11 to locate internal resource Y and places the data at Y.

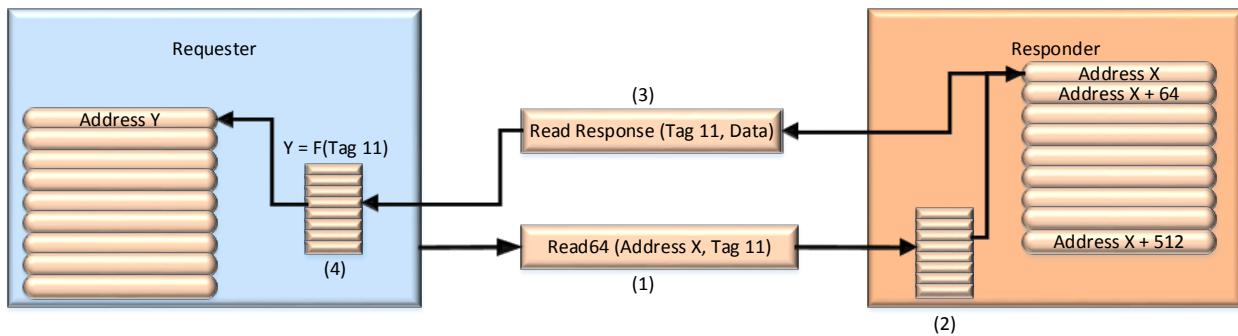


Figure 6-1: Read Request with Corresponding Read Response

Components that support the P2P-Core OpClass may use P2P-Core Read Offset requests to further optimize communication. P2P-Core Read Offset Request is predicated on the Requester comprehending the Responder's logical resource layout. *Example Responder Resource Layout for P2P-Core Read Offset Requests* illustrates an example memory component that is organized into a set of logical banks. Each logical bank is composed of a set of logical rows (the size of a logical row is indicated in the *Responder Bank Structure*). Once a logical row is cached, a P2P-Core Read Offset Request can target any 32-byte-aligned region within the cached logical row by providing the offset from the first byte of the cached logical row.

*Example P2P-Core Read and P2P-Core Read Offset Request Sequence* illustrates a series of P2P-Core Read and P2P-Core Read Offset requests. In this example, a Requester first issues a P2P-Core Read Request with the Spatial Locality (S) = 1b. This informs the Responder that the logical row identified by this request will be subsequently accessed by P2P-Core Read Offset or P2P-Core Write Offset request packets. The Responder caches the logical row to optimize access. The Requester then issues P2P-Core Read Offset request packets with the S = 1b. When the Requester determines the cached logical row is no longer needed, it issues the last P2P-Core Read Offset or Write Offset request packet with S = 0b. This informs the Responder to release the cached logical row (e.g., to write back the contents to the underlying media if needed).

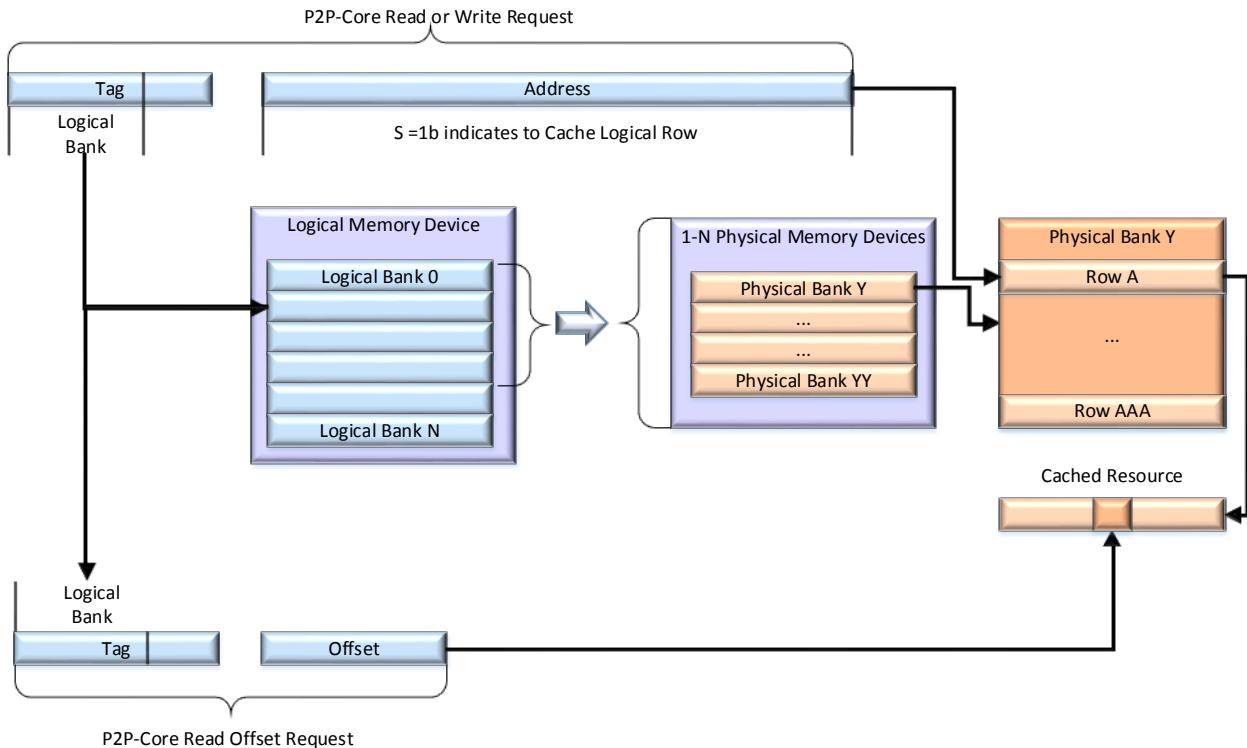


Figure 6-2: Example Responder Resource Layout for P2P-Core Read Offset Requests

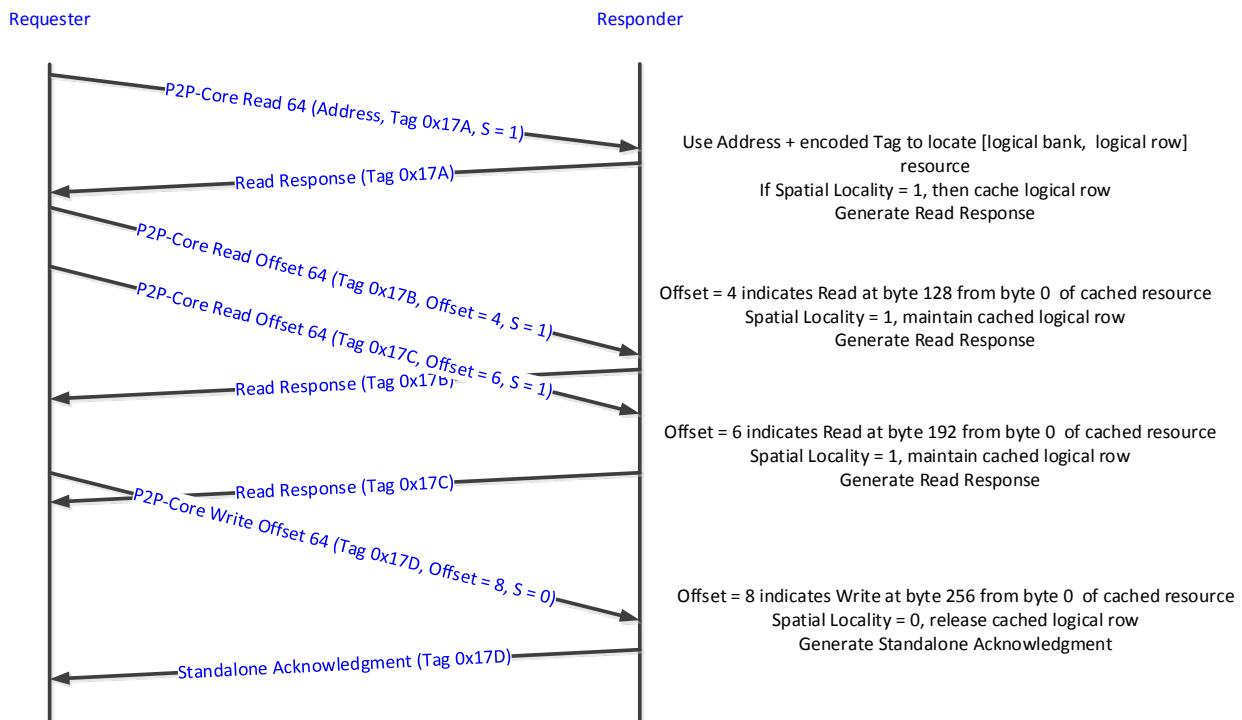
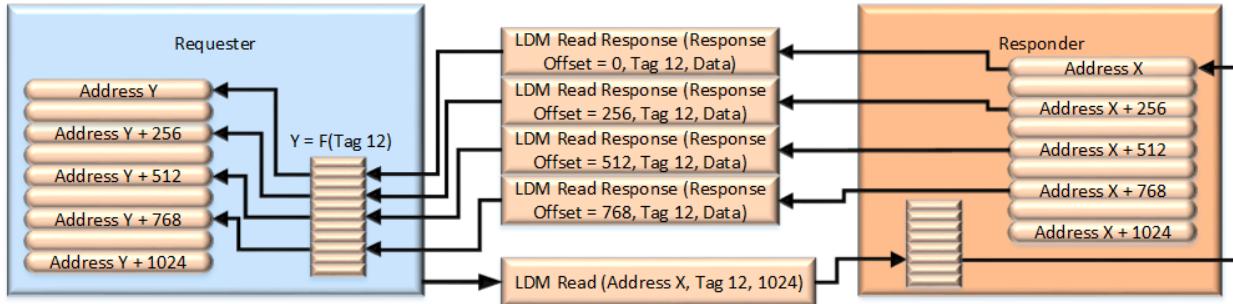


Figure 6-3: Example P2P-Core Read and P2P-Core Read Offset Request Sequence

- 5 Components that support the LDM 1 OpClass may support read requests larger than *Core Structure Max\_Packet\_Payload*. *LDM 1 Read Request with Corresponding LDM 1 Read Responses* illustrates an example single LDM 1 Read Request—Multiple LDM 1 Read Response sequence:

- 5
1. A Requester generates a LDM 1 Read request packet for 1024 bytes targeting address X and tags it with Tag 12.
  2. The Responder decodes address X to locate the underlying physical memory resource.
  3. If `Max_Packet_Payload == 256 bytes`, then the Responder generates four LDM 1 Read response packets of 256 bytes each. The first response sets `Response_Offset = 0`, the second to 256, the third to 512, and the fourth to 768.
  4. For each response packet, the Requester decodes Tag 12 to locate internal resource Y and places the data payload based on the `Response_Offset`.



10 Figure 6-4: LDM 1 Read Request with Corresponding LDM 1 Read Responses

### 6.1.1. Read Features & Requirements

The following are the features and requirements associated with Read Requests:

- 15
- Unless explicitly stated otherwise by this specification, a component that supports non-coherent read request packets shall support the P2P-Core OpClass or the Core 64 OpClass. Support may be as a Requester, a Responder, or as a Requester-Responder.
    - If a component supports P2P-Core Read Offset operations, then it shall support P2P-Core Write Offset operations.
    - *Transparent Router (TR)* shall not support the P2P-Core Read Offset operation.

20

  - Read request packet formats:
    - P2P-Core Read Request packet shall use the *P2P-Core Read Request Packet Format*.
    - P2P-Core Read Offset packet format shall use the *P2P-Core Read Offset Request Packet Format*.
    - P2P-Coherency Read Request packet format shall use the *P2P-Coherency Read Request Packet Format*.
    - Core 64 Read Request packet format shall use the *Core 64 Read Request Packet Format*.
    - LDM 1 Read Request packet format shall use the *LDM 1 Read Request Packet Format*.

25

  - Read request acknowledgments:
    - A P2P-Core Read request packet and a P2P-Core Read Offset request packet shall be acknowledged by a P2P-Core Read Response packet, or by P2P-Core *Standalone Acknowledgment* in case of error.
    - A P2P-Coherency Read request packet shall be acknowledged by a P2P-Coherency Read Response packet, or by a P2P-Coherency *Standalone Acknowledgment* in case of error.
    - A Core 64 Read request packet shall be acknowledged by a Core 64 Read Response packet, or by a Core 64 *Standalone Acknowledgment* in case of error.
    - A LDM 1 Read request packet shall be acknowledged by one or more LDM 1 Read Response packets, or by a single Core 64 *Standalone Acknowledgment* in case of error.

30

- LDM 1 Read request packets are completed by multiple LDM 1 Read Response packets. Each LDM 1 Read Response packet shall contain `Max_Packet_Payload` bytes of payload. Response packets should be returned such that response 0 contains bytes 0 through  $(\text{Max_Packet_Payload} - 1)$ , response 1 contains bytes `Max_Packet_Payload` through  $(2 * \text{Max_Packet_Payload} - 1)$ , and so forth.
- P2P-Core Read request packets are fixed-sized reads. The respective OpCode indicates the read size.
- P2P-Coherency Read request packet reads 1 to 256 bytes of data starting at the indicated address.
- A Core 64 Read request packet reads from 0 to `Max_Packet_Payload` bytes of data starting the indicated address.
  - A Core 64 Read request packet may read zero bytes. Upon receipt, the Responder shall validate and execute the request packet.
    - Even though no data is being read, the Responder shall perform access control and access permission validation steps.
    - If successfully validated and executed, then the Responder shall generate a Core 64 Read Response packet with a zero-length payload field.
    - If the request packet's Address is associated with a cached resource, then the Responder shall perform the Responder-specific coherency actions to access the resource.
- A LDM 1 Read request reads from 1 to  $2^{32}$  bytes of data starting at the indicated address. For LDM 1 Read requests larger than 4096 bytes, the Read Length shall be an integer `Max_Packet_Payload` quantity of data. When processing a LDM 1 Read request packet, Requesters and Responders shall take the steps specified in *Non-Deterministic Request Execution*.
- If a Requester or Responder supports P2P-Core Read, then:
  - Requesters shall encode the Tag field in all P2P-Core Read Request and P2P-Core Read Offset request packets as described in *Tag Field* and *Responder Bank Structure*. The Tag field is used to identify the logical bank targeted by a given request.
  - Responders shall interpret the Tag field as encoded in all P2P-Core Read and P2P-Core Read Offset request packets as described in *Tag Field*.
- A component may support the P2P-Core Read Offset operation. If supported, then:
  - Responders shall support the *Responder Bank Structure*.
  - Responders shall support a non-zero logical row size.
  - If a Responder receives a P2P-Core Read or P2P-Core Write request packet with `S = 1b`, then the Responder shall cache the corresponding logical row.
    - A P2P-Core Read Offset or P2P-Core Write Offset request packet shall target only an address plus target data length that exists within a single logical row.
  - A Requester shall transmit a P2P-Core Read or P2P-Core Write packet with `S = 1b` targeting a given logical bank prior to transmitting a P2P-Core Read Offset or P2P-Core Write Offset request packet targeting the same logical bank.
    - Each Read Offset and Write Offset request packet that targets a given logical bank shall implicitly target the previously cached logical row.
    - If a P2P-Core Read Offset or Write Offset request packet is received and there is no previously cached resource, then the Responder shall return a *Standalone Acknowledgment* indicating a MP error.
  - The Requester shall set and clear the Spatial Locality bit within P2P-Core Read, P2P-Core Read Offset, P2P-Core Write, and P2P-Core Write Offset request packets to indicate when a Responder shall cache or release a cached logical row subrange.

- If a subsequent P2P-Core Read or Write request packet targets the same logical bank but not the same logical row (i.e., the Address field maps to a different logical row), then the Responder shall automatically release the previously cached logical row, and cache the new logical row if the packet's S == 1b.
- If a Responder is accessed by two Requesters (daisy-chain topology with a Requester at each end of the daisy-chain), then the Requesters are responsible for coordinating access to prevent a cached logical row from being prematurely released. Requester coordination is outside of this specification's scope.
- The Offset field within the P2P-Core Read Offset request packet represents an integer multiple of 32-byte-offset from byte 0 of the subrange that was cached by a prior P2P-Core Read or P2P-Core Write request packet with S = 1b.
  - A 32-byte request shall target an integer multiple of 32 bytes from byte 0 of the cached logical row.
  - A 64-byte request shall target an integer multiple of 64 bytes from byte 0 of the cached logical row.
  - A 128-byte request shall target an integer multiple of 128 bytes from byte 0 of the cached logical row.
  - A 256-byte request shall target an integer multiple of 256 bytes from byte 0 of the cached logical row.
- The Requester may interleave any number of P2P-Core Read Offset or P2P-Core Write Offset request packets to the same cached logical row.
  - Individual request packets within a sequence of request packets with S = 1b may be any supported size and valid offset.

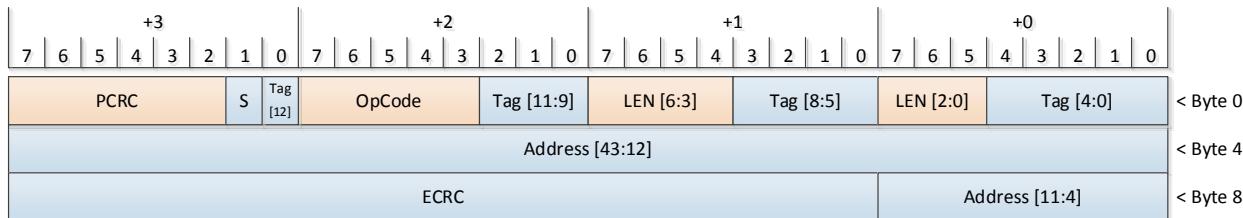


Figure 6-5: P2P-Core Read Request Packet Format

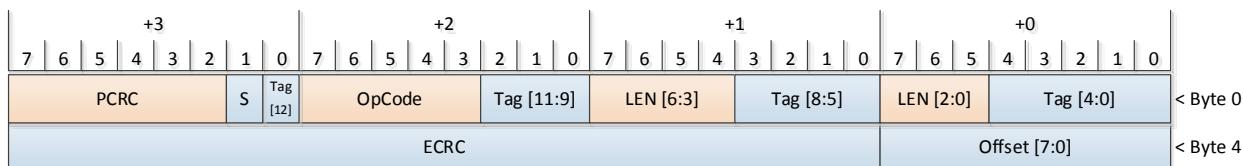


Figure 6-6: P2P-Core Read Offset Request Packet Format

Table 6-1: P2P-Core Read Offset Request-specific Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
Offset	Offset	8	<p>Offset is used to calculate the start of an integer 32-byte-aligned subrange within a previously cached logical row targeted by the P2P-Core Read Offset Request. To calculate the first byte of an offset location,</p> $\text{Byte address} = \text{Offset} * 32$

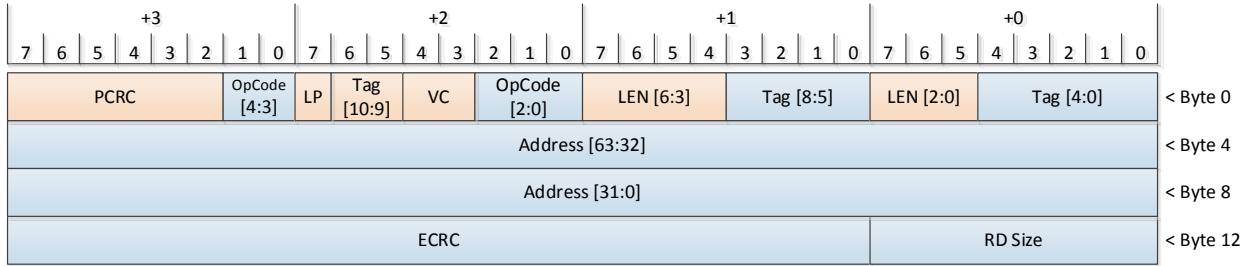


Figure 6-7: P2P-Coherency Read Request Packet Format

Table 6-2: P2P-Coherency Read Request Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
RD Size	-	8	Number of bytes to be read If RD Size == 0x0, then read 256 bytes

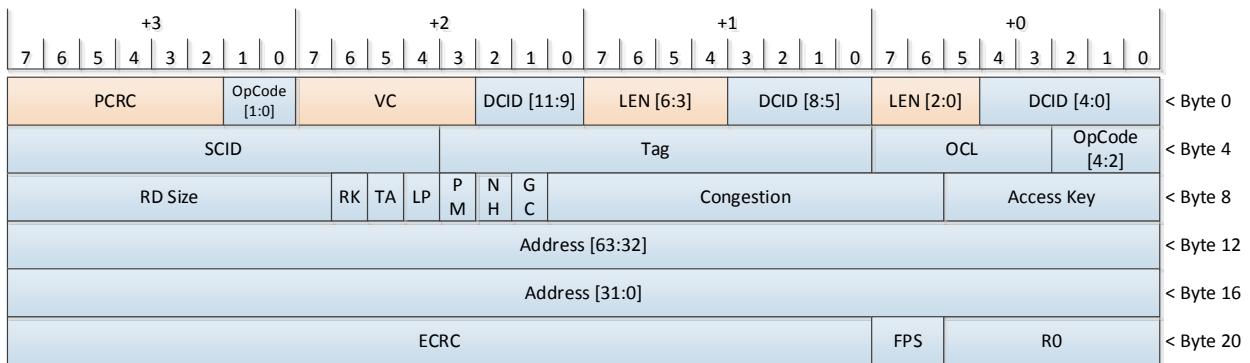


Figure 6-8: Core 64 Read Request Packet Format

Table 6-3: Core 64 Read Request Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
RD Size	-	9	Number of bytes to be read Maximum number of bytes to read shall be Max_Packet_Payload.
RO	-	6	Reserved

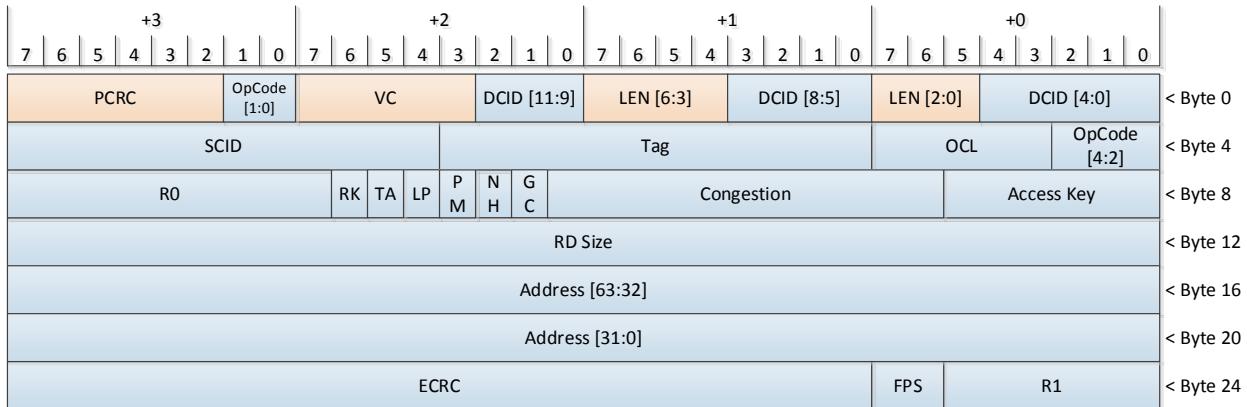


Figure 6-9: LDM 1 Read Request Packet Format

Table 6-4: LDM 1 Read Request-specific Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
RD Size	-	32	Number of bytes to be read If Read Size == 0x0, then $2^{32}$ bytes are to be read.
RO	-	9	Reserved
R1	-	6	Reserved

## 6.1.2. Read Response Features & Requirements

The following are the features and requirements associated with Read Response packets:

- Unless explicitly stated otherwise by this specification, each component shall support Read Response packets associated with at least one of the following OpClasses: the P2P-Core OpClass or the Core 64 OpClass. Support may be as a Requester, a Responder, or as Requester-Responder.
- Read Response packet formats:
  - P2P-Core Read Response packet shall use the *P2P-Core Read Response Packet Format*.
  - P2P-Coherency Read Response packet shall use the *P2P-Coherency Read Response Packet Format*.
  - Core 64 Read Response packet shall use the *Core 64 Read Response Packet Format*.
    - If the OpCode == Control Read Response, then the LP bit shall be Reserved.
  - LDM 1 Read Response packet shall use the *LDM 1 Read Response Packet Format*.
- A P2P-Core Read Response packet shall be returned for each successfully executed P2P-Core Read, P2P-Core Read Offset, or P2P-Coherency Read (non-coherent read) request packet.
- A P2P-Coherency Read Response packet shall be returned for each successfully executed P2P-Coherency *Read Exclusive* or *Read Shared* request packet.
- A single Core 64 Read Response packet shall be returned for each successfully executed Core 64 Read request packet. If a zero-length Core 64 Read request packet was successfully executed, then a zero-length Core 64 Read Response payload shall be returned.
- One or more LDM 1 Read Response packets shall be returned for each successfully executed LDM 1 Read request packet.

- 5
  - o If a LDM 1 Read request packet RD Size is less than or equal to 128 bytes, then a single LDM 1 Read Response packet shall be returned and the payload shall contain RD Size bytes.
  - o If a LDM 1 Read request packet RD Size is greater than 128 bytes, then multiple LDM 1 Read Response packets may be returned.
    - The Tag field within each Core 64 Read Response packet shall contain the Core 64 Read request packet's Tag field.
    - The retransmission timer associated with a given LDM 1 Read request shall be reset each time a corresponding LDM 1 Read Response packet is successfully received and executed.
- 10
  - A Requester shall not take any action that is dependent on the successful completion of the Read Request until the receipt and validation of corresponding Read Response packets.
  - If the ECRC field in a response is stomped by the Responder, then the Responder should generate a new response to complete the request unless the condition that caused the field to be stomped precludes generating a new response packet or generating the response packet would exceed the worst-case read latency (P2P-Core / P2P-Coherency OpClasses) or Responder Deadline (explicit OpClasses). A Responder shall stomp the ECRC field if it detects or suffers an error during response packet creation or transmission.
  - A Read Response packet (of any type) shall not be acknowledged.
  - A Read Response packet may contain a Meta field. Meta field presence is indicated by the MS field. See *Meta Read Response and Meta Write*.
- 15
- 20

Table 6-5: Common Read Response Packet Fields

Field Name	Size (bits)	Description
<b>Meta</b>	-	Meta Data—If the Responder is unable to interpret the contents of this field, then it shall treat this field as Reserved. Meta field presence and size is indicated by the MS field.
<b>RRSP Reason</b>	4	The Read Response Reason communicates additional information relative to the associated request packet's execution. Only the following Read Response Reasons may be returned in an Atomic response packet; all other conditions shall be returned using a <i>Standalone Acknowledgment</i> . <ul style="list-style-type: none"> <li>0x0—No Error</li> <li>0x1-0xB—Reserved</li> <li>0xC—Data Correctable Error Warning</li> <li>0xD—Data Uncorrectable Error Detected</li> <li>0xE—Poison Data (PD) Failure</li> <li>0xF—Exclusive Granted (Applicable only to P2P-Coherency and Core 64 Read Response packets)</li> </ul>
<b>MS</b>	2	Indicates if the Meta field is present and its size <ul style="list-style-type: none"> <li>0x0—Not present</li> <li>0x1—32-bit Meta field size</li> <li>0x2—64-bit Meta field size</li> <li>0x3—Reserved</li> </ul>

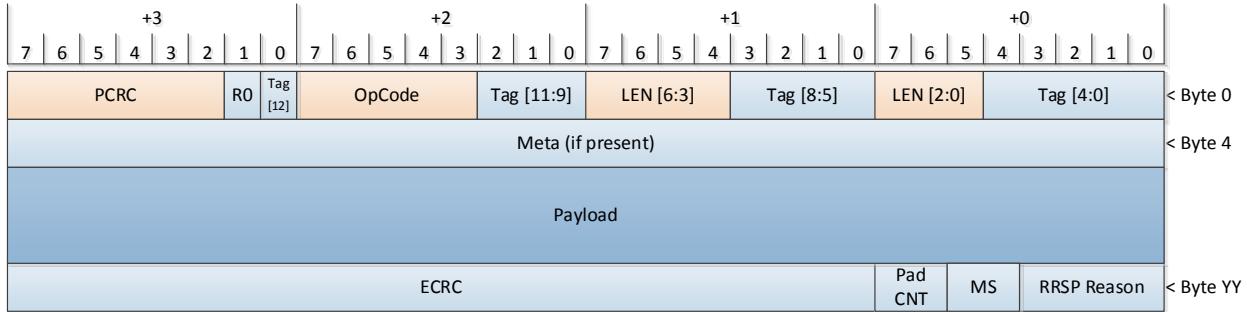


Figure 6-10: P2P-Core Read Response Packet Format

Table 6-6: P2P-Core Read Response-specific Packet Fields

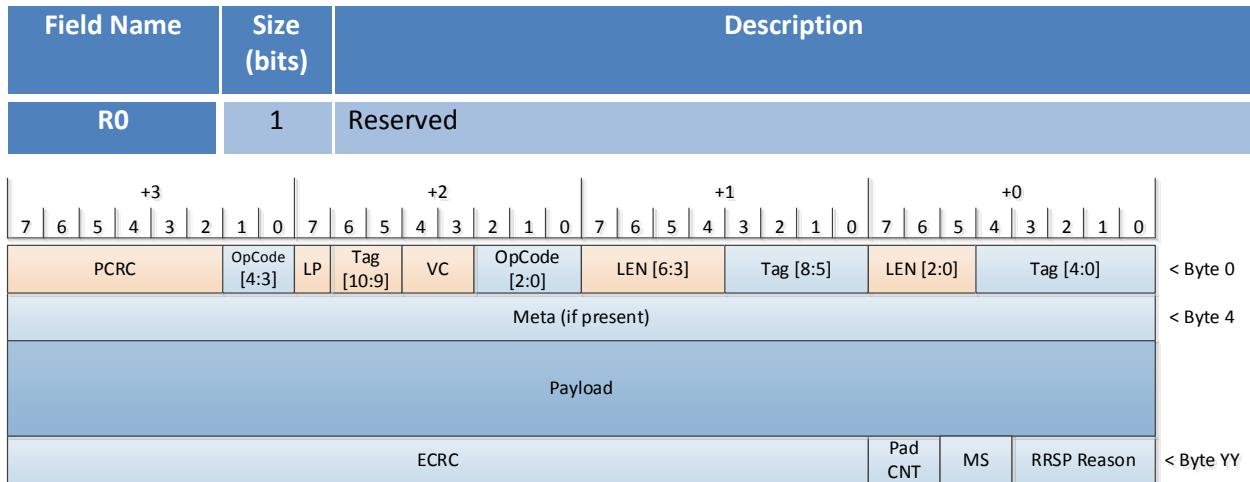


Figure 6-11: P2P-Coherency Read Response Packet Format

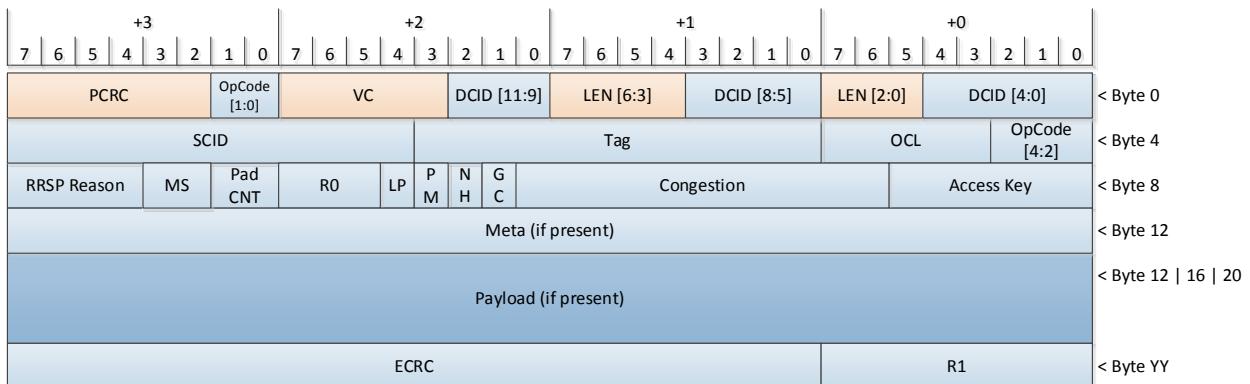


Figure 6-12: Core 64 Read Response Packet Format

Table 6-7: Core 64 Read Response-specific Packet Fields

Field Name	Size (bits)	Description									
<b>R0</b>	3	Reserved									
<b>R1</b>	8	Reserved									

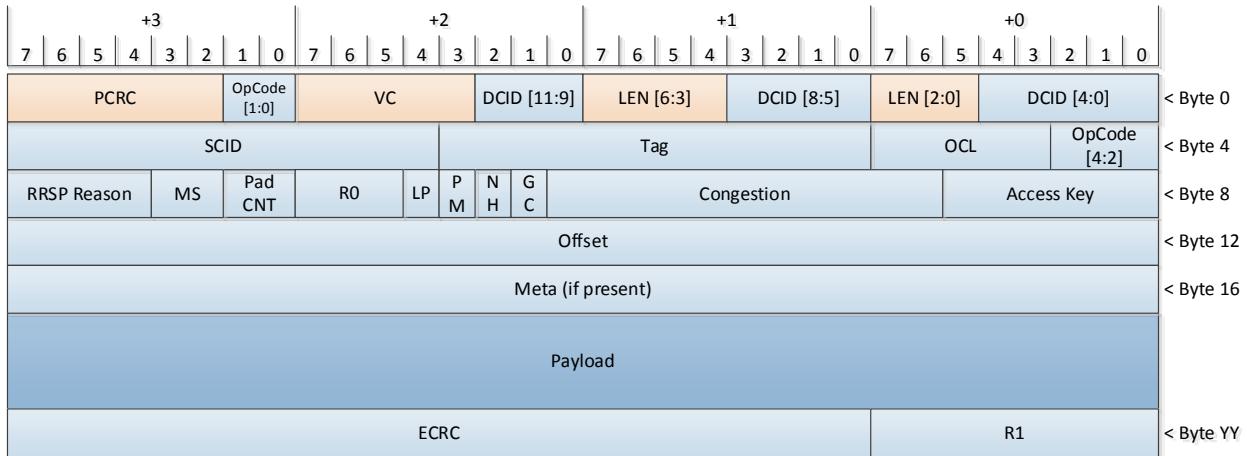


Figure 6-13: LDM 1 Read Response Packet Format

Table 6-8: LDM 1 Read Response-specific Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
Offset	OF	32	Byte offset from byte 0 of the buffer associated with the Tag where the Payload is to be placed.
R0	-	3	Reserved
R1	-	8	Reserved

## 6.2. Write

A Write request is used to write data of a specified size to the indicated address at the destination component. *Write Request with Corresponding Acknowledgment* illustrates an example packet exchange sequence:

1. A Requester generates a Write request with 64 bytes of data payload extracted from internal resource Y. The Requester sets the request packet to target address X and tags it with Tag 13.
2. The Responder decodes address X to locate the underlying physical memory resource.
3. The Responder places 64 bytes of data from the packet's payload at the specified address of the underlying physical memory resource.
4. The Responder generates a *Standalone Acknowledgment*.

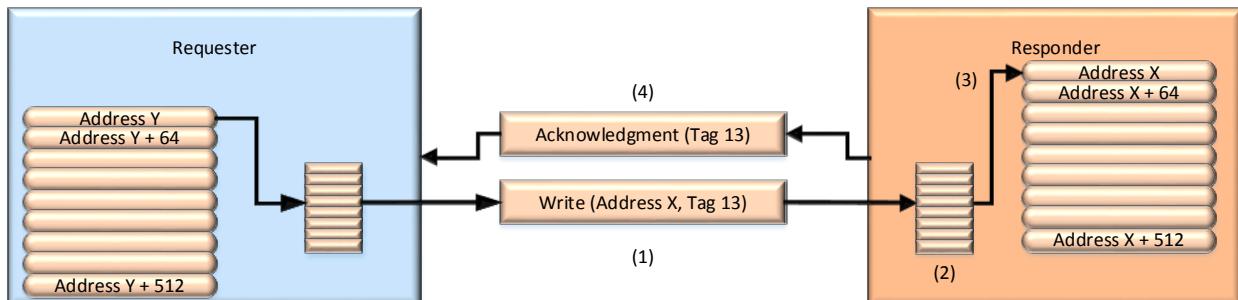


Figure 6-14: Write Request with Corresponding Acknowledgment

Components that support the P2P-Core OpClass may use P2P-Core Write Offset Requests to further optimize communication. *Example Responder Resource Layout for P2P-Core Write Offset Requests* illustrates an example memory component that is organized into a set of logical banks. Each logical bank is composed of a set of logical rows (the size of a logical row is indicated in the *Responder Bank Structure*). Once a logical row is cached, a P2P-Core Write Offset Request can target any 32-byte aligned region within the cached logical row by providing the offset from the first byte of the cached logical row.

*Example P2P-Core Write and P2P-Core Write Offset Request Sequence* illustrates a series of P2P-Core Write and P2P-Core Write Offset Requests. In this example, a Requester first issues a P2P-Core Write request packet with the Spatial Locality (S) = 1b. This informs the Responder that the logical row identified by this request will be subsequently accessed by P2P-Core Read Offset or P2P-Core Write Offset request packets. The Responder caches the logical row to optimize access. The Requester then issues P2P-Core Read Offset and P2P-Core Write Offset request packets with S = 1b. When the Requester determines the cached logical row is no longer needed, it issues the last P2P-Core Read Offset or Write Offset request packet with S= 0b. This informs the Responder to release the cached logical row (e.g., to write back the contents to the underlying media if needed).

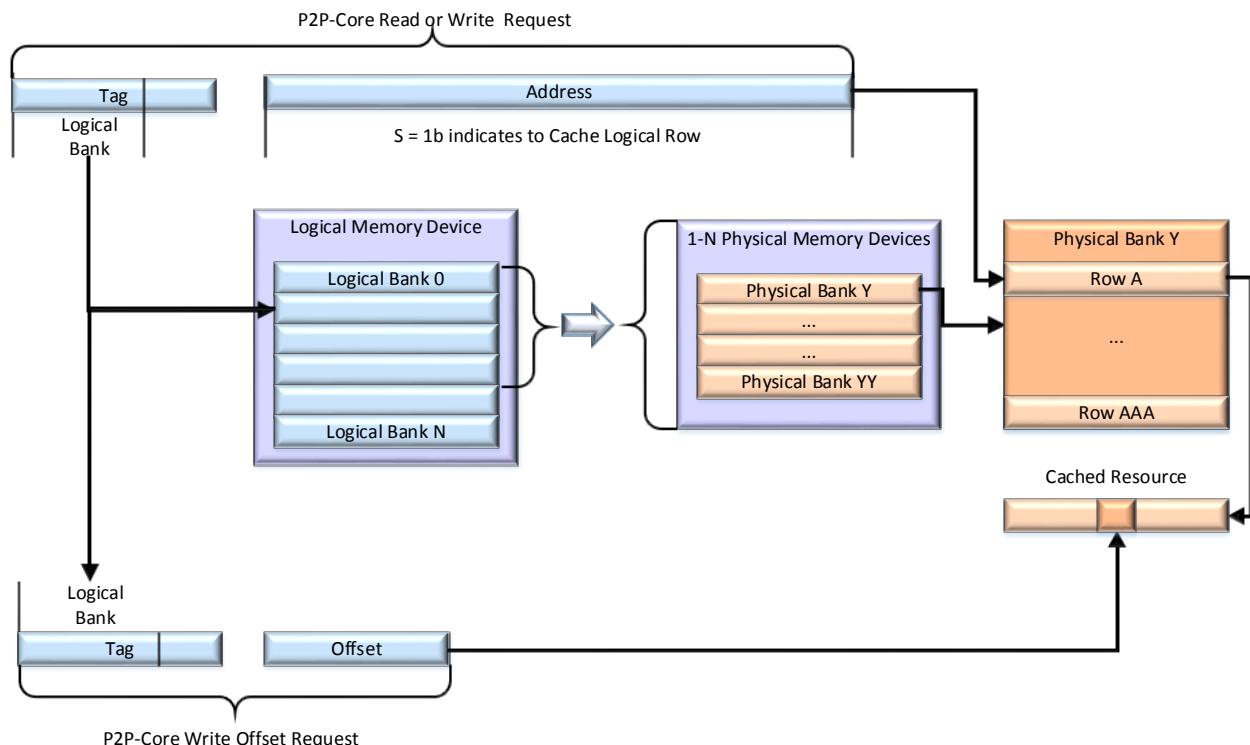


Figure 6-15: Example Responder Resource Layout for P2P-Core Write Offset Requests

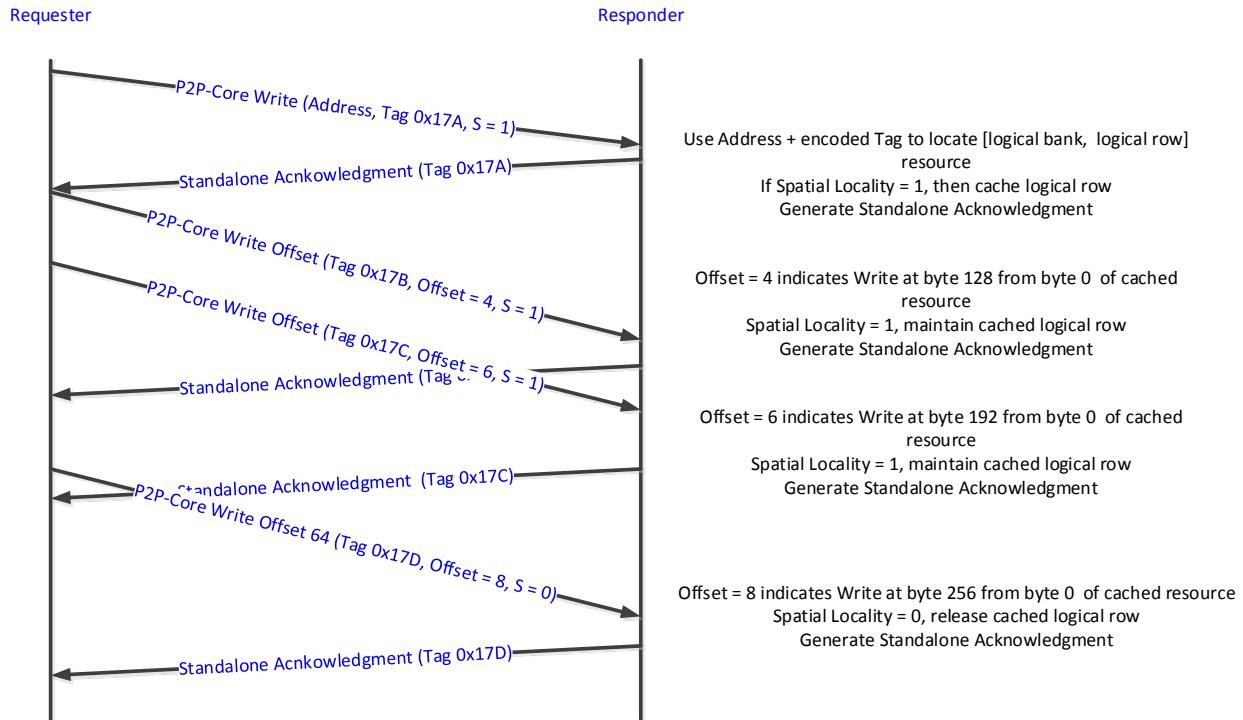


Figure 6-16: Example P2P-Core Write and P2P-Core Write Offset Request Sequence

The following are the features and requirements associated with Write Requests:

- Unless explicitly stated otherwise by this specification, a Requester or a Responder that supports non-coherent read requests shall support the P2P-Core OpClass or the Core 64 OpClass. Support may be as a Requester, a Responder, or as a Requester-Responder.
- Unless explicitly stated otherwise by this specification, each component shall support Write Requests associated with at least one of the following OpClasses: the P2P-Core OpClass, or the Core 64 OpClass. Support may be as a Requester, a Responder, or as Requester-Responder.
  - If a component supports P2P-Core Write Offset operations, then it shall support P2P-Core Read Offset operations.
  - A *Transparent Router (TR)* shall not support the P2P-Core Write Offset operation.
- Unless explicitly stated otherwise by this specification, all P2P-Core Write Request requirements shall apply to a P2P-Core Write Persistent Request.
- Write Packet formats:
  - P2P-Core Write request packets shall use the *P2P-Core Write Request Packet Format*.
  - P2P-Core Write Offset request packets shall use the *P2P-Core Write Offset Request Packet Format*.
  - P2P-Coherency Write request packets shall use the *P2P-Coherency Write Request Packet Format*.
  - Core 64 Write packets shall use the *Core 64 Write Request Packet Format*.
- Explicit OpClass write request packets may be either a unicast or a multicast end-to-end packet. Unicast request packets use the packet formats specified in this section. Multicast request packets use Multicast OpClass and the packet formats specified in *Multicast*.
- Unless explicitly stated otherwise by this specification, a Responder shall transmit a *Standalone Acknowledgment* for each received Write request packet.

- An unreliable Core 64 write request packet shall not be acknowledged, i.e., it is semantically analogous to a posted write. Core 64 uses the UR bit to indicate reliability (see *Common OpCode-specific Bits*).
  - If UR == 1b, LP == 0b, and an error is detected, then the packet shall be silently discarded. Errors may be surfaced through means outside of this specification's scope, or, if supported, through an *Unsolicited Event (UE) Packet* based on *Component Error and Signal Event Structure* configuration.
  - If UR == 1b, LP == 1b, and an error is detected, then the packet shall be discarded, and the error shall be surfaced through an *Unsolicited Event (UE) Packet* based on *Component Error and Signal Event Structure* configuration.
- If *Interface I-CAP 1 Control Omit P2P Standalone Acknowledgment* == 1b, then the Responder shall not transmit a *Standalone Acknowledgment* for a successfully validated and executed P2P-Core Write, P2P-Core Write Offset, or a P2P-Coherency Write request packet that does not contain the PU field or PU == 0b.
  - If a P2P-Core Write, P2P-Core Write Offset, or P2P-Coherency Write request packet suffers a non-transient error and *Component Containment* has not been enabled, then the Responder shall return a *Standalone Acknowledgment* (Reason = error detected).
  - Upon detecting a non-transient error, the Responder should stop executing new request packets that modify Data Space resources. If *In-band Management* is supported, then the Responder shall execute new P2P-Core or P2P-Coherency *CTL-Read and CTL-Write Packets*.
- If a component does not support P2P-Core Write Persistent but requires persistent write semantics, then the component shall support P2P-Core *Persistent Flush*. When persistence is required, the Requester transmits a P2P-Core *Persistent Flush* request packet which ensures the payloads of all previously transmitted P2P-Core Write and Write Offset request packets are persistent prior to being acknowledged (unlike other request packet formats, the P2P-Core Write and Write Offset request packet formats do not contain the PU bit to indicate when persistence is required).
- If UR == 0b or PU == 1b, then a Core 64 Write request packet shall be acknowledged by a Core 64 *Standalone Acknowledgment*.
- A Responder shall not schedule a *Standalone Acknowledgment* until a P2P-Core Write Persistent or persistent Core 64 Write request packet has been successfully executed and the packet's payload has reached a processing state such that the update to the underlying media is guaranteed to succeed or an error has been detected.
- A Responder shall not schedule a *Standalone Acknowledgment* until a Core 64 Write request packet that requires Reliable Delivery has been successfully executed and the packet's payload has reached a processing state such that the data is visible.
  - If a Responder supports cache coherency, then write data that resides in a cache and has not been flushed back to underlying media may be treated as visible.
- A Responder shall not modify the addressed resource until the request packet has been successfully received, and validated. If a request packet experiences a subsequent execution error, then the addressed resource contents may be non-deterministic.

- If the addressed media targeted by a given request packet has entered Fatal Media Error Containment, then this request packet shall not be executed and a *Standalone Acknowledgment* (Reason = Fatal Media Error Containment Triggered) shall be returned.
- The payload length of P2P-Core Write request packets shall be an integer 16-byte multiple. The maximum Payload length shall be less than or equal to *Max\_Packet\_Payload*. Payload length is calculated as follows:
  - $\text{Payload Length} = \text{Packet Length} - (\text{total protocol bytes})$
- A P2P-Coherency or Core 64 request packet may carry 0 to *Max\_Packet\_Payload* bytes. If the payload field size is not an integer 4-byte multiple, then 1-3 pad bytes shall be appended such that the payload field size is an integer 4-byte multiple that does not exceed *Max\_Packet\_Payload*. The Pad CNT field shall reflect the number of pad bytes [0-3]. Payload length shall be calculated as follows:
 
$$\text{Payload Length} = \text{Packet Length} - (\text{total protocol bytes}) - \text{Pad CNT}$$
- If a Responder supports hardware-enforced cache coherency, then upon receipt of any type of write request packet with NS = 0b, the Responder shall perform Responder-specific cache coherency actions, e.g., writeback dirty data and invalidate clean copies.
- If a Responder supports a cache resource, then upon receipt of any type of write request packet with TC == 1b, the Responder shall place the packet's payload in the cache. Placement may require the cache to evict existing cache lines to free up space and, if supported, the Responder performing cache coherency actions.
  - If a Responder's cache agent determines that the Requester is not permitted to modify the addressed cache line (e.g., is not the current owner), then the Responder shall handle this as an "Access Error (AE)—Invalid Access Permission".
- Any type of write request packet may contain a zero-byte payload. Upon receipt, the Responder validates and executes the request packet.
  - Even though there is no payload data, the Responder shall perform access control and access permission validation steps.
  - If a Responder supports cache coherency, then upon receipt of a zero-byte write request packet, the Responder shall perform a cache coherency snoop which will cause the associated cache line to be flushed. If the cache line's home is the Requester, then this shall cause the cache line to be written back to the Requester.
- If a Requester or Responder supports P2P-Core Write, then:
  - Requesters shall encode the Tag field in all P2P-Core Write Request and P2P-Core Write Offset request packets as described in *Tag Field* and *Responder Bank Structure*. The Tag field is used to identify the logical bank targeted by a given request.
  - Responders shall interpret the Tag field as encoded in all P2P-Core Write and P2P-Core Write Offset request packets as described in *Tag Field*.
- A component may support the P2P-Core Write Offset operation. If supported, then:
  - Responders shall support the *Requester Bank Structure*.
  - Responders shall support a non-zero logical row size.
  - If a Responder receives a P2P-Core Read or P2P-Core Write request packet with S = 1b, then the Responder shall cache the corresponding logical row.
    - A P2P-Core Read Offset or P2P-Core Write Offset request packet shall target only an address plus target data length that exists within a single logical row.
  - A Requester shall transmit a P2P-Core Read or P2P-Core Write packet with S = 1b targeting a given logical bank prior to transmitting a P2P-Core Read Offset or P2P-Core Write Offset request packet targeting the same logical bank.

- Each Read Offset and Write Offset request packet that targets a given logical bank shall implicitly target the previously cached logical row.
- If a P2P-Core Read Offset or Write Offset request packet is received and there is no previously cached resource, then the Responder shall return a *Standalone Acknowledgment* indicating a MP error.
- The Requester shall set and clear the Spatial Locality bit within P2P-Core Read, P2P-Core Read Offset, P2P-Core Write, and P2P-Core Write Offset request packets to indicate when a Responder shall cache or release a cached logical row.
  - If a subsequent P2P-Core Read or Write request packet targets the same logical bank but not the same logical row (i.e., the Address field maps to a different logical row), then the Responder shall automatically release the previously cached logical row, and cache the new logical row if the packet's S == 1b.
  - If a Responder is accessed by two Requesters (daisy-chain topology with a Requester at each end of the daisy-chain), then the Requesters are responsible for coordinating access to prevent a cached logical row from being prematurely released. Requester coordination is outside of this specification's scope.
- The Offset field within the P2P-Core Read Offset request packet represents an integer multiple of 32-byte-offset from byte 0 of the subrange that was cached by a prior P2P-Core Read or P2P-Core Write request packet with S = 1b.
  - A 32-byte request shall target an integer multiple of 32 bytes from byte 0 of the cached logical row.
  - A 64-byte request shall target an integer multiple of 64 bytes from byte 0 of the cached logical row.
  - A 128-byte request shall target an integer multiple of 128 bytes from byte 0 of the cached logical row.
  - A 256-byte request shall target an integer multiple of 256 bytes from byte 0 of the cached logical row.
- The Requester may interleave any number of P2P-Core Read Offset or P2P-Core Write Offset request packets to the same cached logical row.
  - Individual request packets within a sequence of request packets with S = 1b may be any supported size and valid offset.

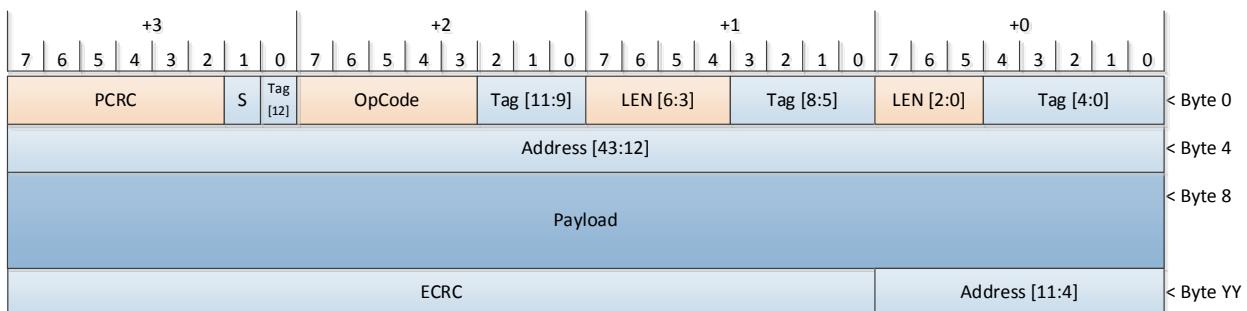


Figure 6-17: P2P-Core Write Request Packet Format

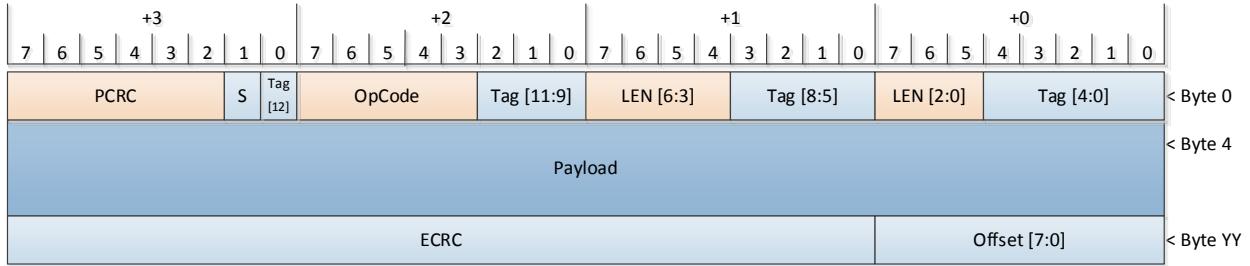


Figure 6-18: P2P-Core Write Offset Request Packet Format

Table 6-9: P2P-Core Write Offset Request-specific Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>Offset</b>	Offset	8	<p>Offset is used to calculate the start of an integer 32-byte-aligned subrange within a previously cached logical row targeted by the P2P-Core Write Offset Request. To calculate the first byte of an offset location,</p> $\text{Byte address} = \text{Offset} * 32$

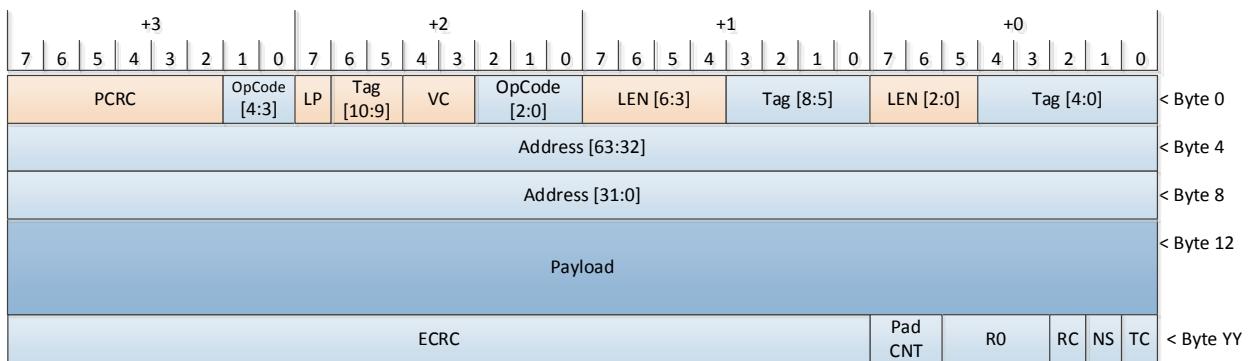


Figure 6-19: P2P-Coherency Write Request Packet Format

Table 6-10: P2P-Coherency Write Request-specific Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>R0</b>	-	3	Reserved

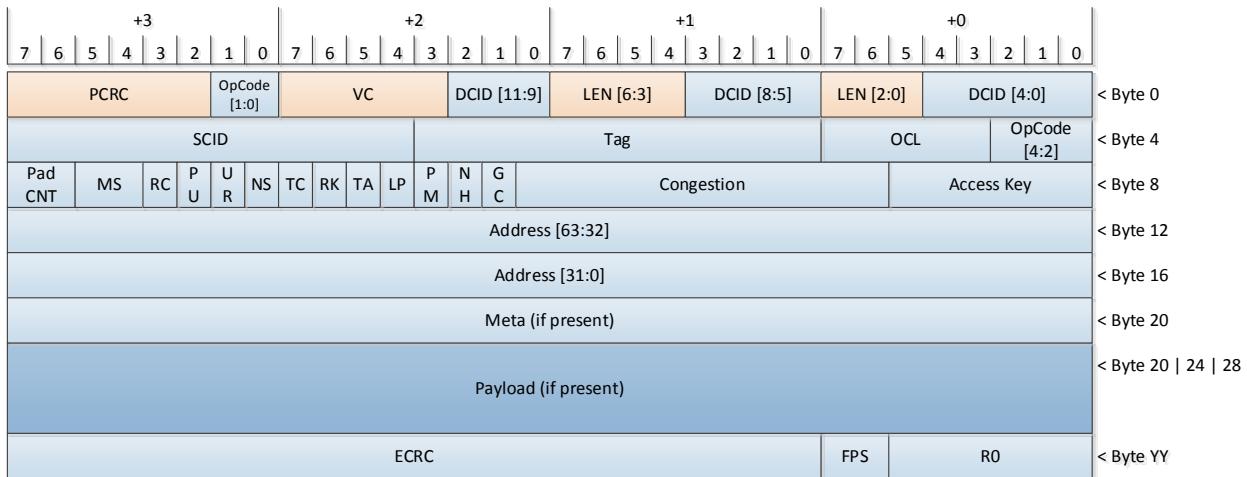


Figure 6-20: Core 64 Write Request Packet Format

Table 6-11: Core 64 Write Request-specific Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>Meta Size</b>	MS	2	Indicates if the Meta field is present and its size (see <i>Meta Read Response and Meta Write</i> ) 0x0—No present 0x1—32-bit Meta field size 0x2—64-bit Meta field size 0x3—Reserved
<b>Meta</b>	-	-	Meta Data—if the Responder is unable to interpret the contents of this field, then it shall treat this field as Reserved.
<b>RO</b>	-	6	Reserved

### 6.3. Write MSG

Conceptually, a Write MSG is a series of write requests that target a single anonymous buffer at the Responder. *Example Write MSG Request* illustrates a series of Write MSG request packets to move 1024 bytes.

Although this type of request does not change the actual number of packets required to move the data compared to a series of write request packets, Write MSG request packets can simplify communications between cooperating components, and improve underlying solution efficiency, e.g., messaging or memory migration services. Write MSG request packets can also be used to target Responder-managed buffers where the actual destination physical buffers are not visible to the Requester, i.e., are anonymous.

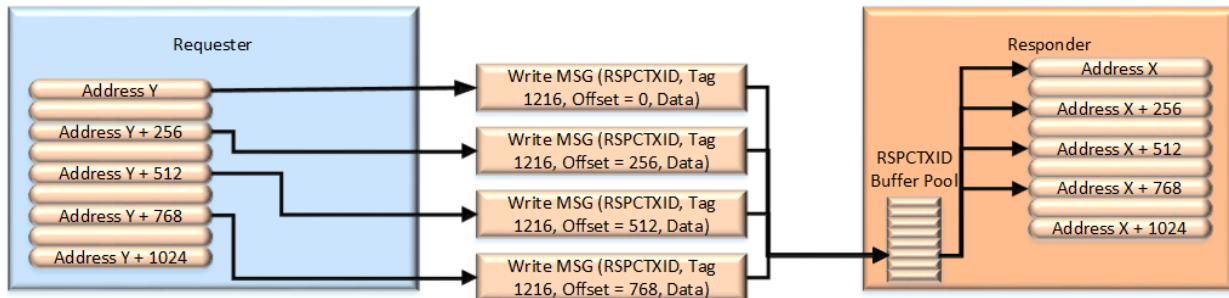


Figure 6-21: Example Write MSG Request

The following are the features and requirements associated with Write MSG Requests:

- Any component type may support Write MSG request operations.
- Write MSG and Unreliable Write MSG request packets shall use the *(Unreliable) Write MSG Request Packet Format*.
- Write MSG request packets may be either a unicast or a multicast end-to-end packet. Unicast request packets use the packet formats specified in this section. Multicast request packets use Multicast OpClass and the packet formats specified in *Multicast*.
- A Write MSG request packet shall be acknowledged by a *Core 64 Standalone Acknowledgment*.
  - A Write MSG request packet transmitted to a reliable multicast group shall be acknowledged as specified in *Multicast*.
  - If the addressed media targeted by a given request packet has entered Fatal Media Error Containment, then this request packet shall not be executed. If a reliable Write MSG, then a *Standalone Acknowledgment* (Reason = Fatal Media Error Containment Triggered) shall be returned.
  - A Responder shall not schedule a *Standalone Acknowledgment* until a reliable Write MSG request packet has been successfully executed and the packet's payload has reached a processing state such that the data is visible.
    - If a Responder supports cache coherency, then the write data may reside in cache and not yet be flushed back to the underlying media.
- Unreliable Write MSG request packets shall not be acknowledged. If a packet validation or execution error occurs, then the packet shall be silently discarded and the Responder shall silently discard all packets associated with a series of Unreliable Write MSG packets and release the underlying resources. Errors may be surfaced through means outside of this specification's scope.
- Write MSG request packets within a given series of request packets may arrive out of order.
- If multipath is supported, Write MSG request packets within a given series of request packets may be transmitted and received on multiple interfaces.
- A Responder shall maintain a timer based on *Core Structure UNRSP* that tracks the forward progress of a Write MSG operation. If the timer expires before a new request packet associated with a given Write MSG operation is received, then the Responder shall silently discard the Write MSG operation (all Write MSG tracking logic associated with the discarded operation shall be reset, and if a receive data buffer may was updated by prior Write MSG request packets associated with this operation, then the buffer contents may be in a non-deterministic state; it does not need to be released).
  - If an unreliable unicast Write MSG operation, then the Responder may generate a *Standalone Acknowledgment* (Reason = Resource Release). This is an optimization

intended to reduce Requester recovery time. If unsupported, then the Requesters shall silently discard the *Standalone Acknowledgment*

- If a reliable unicast Write MSG operation, then the Responder shall generate a *Standalone Acknowledgment* (Reason = Resource Release) to indicate the operation was abandoned. The Responder shall release non-idempotent results resources.

5     ● **Unicast Write MSG-specific requirements:**

- A Responder shall be responsible for determining when all request packets in an Unreliable Write MSG operation have been received, e.g., the Responder can calculate the number of request packets required to complete a Write MSG operation and track when all of the corresponding data has been successfully received. Upon receipt of the request packet that completes an Unreliable Write MSG operation, the Responder initiates component-local Write MSG operation completion processing.
- A Responder may generate a *Standalone Acknowledgment* (Reason = Resource Release) once an Unreliable Write MSG is completed or is no longer being tracked, e.g., due to a component-local event such as expiration of a fail-safe timer. Receipt does not acknowledge success or failure of the corresponding Write MSG operation; it indicates only that the Requester may reuse the packet identification protocol field values in a new unreliable request packet. Transmission of a *Standalone Acknowledgment* is not required for correctness; it is an optimization intended to reduce Requester resource tracking requirements. If unsupported, Requesters shall silently discard the *Standalone Acknowledgment*.
- A Requester shall be responsible for determining when all request packets in a given reliable Write MSG operation have been successfully received.
  - Upon receipt of the request packet that completes a reliable Write MSG operation, the Responder initiates component-local processing to complete the Write MSG operation.
  - Reliable Write MSG operations are non-idempotent, and shall be handled as specified in *Non-idempotent Request (NIR)*.
  - Though a Responder may delay component-local Write MSG operation completion processing until receipt of the corresponding *Non-Idempotent Request Release (NIRR)* packet, to reduce end-to-end application latency, a Responder should track when all of the corresponding data has been successfully received, and, upon receipt of all data, initiate component-local completion processing.
  - If a reliable Write MSG request packet fails due to a non-transient error, then the Requester shall transmit, if possible, a *Non-Idempotent Request Release (NIRR)* packet to release the Responder's resources, and shall fail the operation.

10    ● **Multicast Write MSG-specific requirements:**

- A Responder shall be responsible for determining when all request packets in an unreliable multicast Write MSG operation have been received, e.g., the Responder can calculate the number of request packets required to complete an unreliable multicast Write MSG operation and track when all of the corresponding data has been successfully received. Upon receipt of the request packet that completes an unreliable multicast Write MSG operation, the Responder performs component-local processing to complete the unreliable multicast Write MSG operation.

**Developer Note:** A Responder takes the same steps to determine when the unreliable multicast Write MSG operation is completed and performs essentially the same component-local processing steps upon completion as it does for an unreliable unicast Write MSG.

- 5     ○ A Requester shall be responsible for determining when all request packets in a Write MSG operation associated with a reliable multicast group have been successfully received.
  - Upon receipt of the request packet that completes a reliable Write MSG operation, the Responder performs component-local processing to complete the Write MSG operation.
  - If a Responder detects missing or out-of-order request packets in a reliable Write MSG operation associated with a reliable multicast group, then the Responder shall transmit a *Reliable Multicast Acknowledgment* packet indicating the last successfully received request packet and shall silently discard all subsequently received multicast request packets whose sequence number does not match the next expected sequence number for this multicast group.
- 10    ● If the payload field size is not an integer 4-byte multiple, then 1-3 pad bytes shall be appended such that the payload field size does not exceed Max\_Packet\_Payload and the Pad CNT field shall reflect the number of pad bytes
  - A Write MSG or Unreliable Write MSG may have a non-zero Pad CNT only in the last request packet in a message.
  - Payload length shall be calculated as follows:  
Payload Length = Packet Length – (total protocol bytes) – Pad CNT
- 15    ● A Write MSG request packet may contain a zero-byte payload. Upon receipt, the Responder validates and executes the request packet.
  - Even though there is no payload data, the Responder shall perform access control and access permission validation steps.
  - Successful execution of a zero-byte payload (Unreliable) Write MSG request packet shall consume a posted Responder receive buffer.
  - If the request packet requires Reliable Delivery, then the Responder shall generate a *Standalone Acknowledgment*.
- 20    ● If data to be transmitted is larger than Max\_Packet\_Payload, then a series of Write MSG request packets is used to transmit the data. In a series, the payload field size of the first and all intermediate Write MSG request packets shall be of Max\_Packet\_Payload.
- 25    ● The payload field size of a single Write MSG request packet operation or of the last Write MSG request packet in a series may be less than or equal to Max\_Packet\_Payload.
- 30    ● Each Write MSG request packet in the series shall use a unique WOFF. The Responder shall use WOFF to calculate where to place the payload relative to byte zero of the anonymous buffer.
- 35    ● If the Write MSG request packet was successfully received and executed, then:
  - The Responder shall transmit a Core 64 *Standalone Acknowledgment* (Reason = No Error Write MSG).
  - The Request-specific (RS) field shall be set to the WOFF of the corresponding request packet. The Requester uses the combination of the Tag and the WOFF to determine each individual request packet's success.
  - The Responder shall set the RNR / QD field to indicate the queue depth at the time the acknowledgment was scheduled.

- If a Responder supports hardware-enforced cache coherency, then upon receipt of a Write MSG request packet with NS = 0b, the Responder shall perform Responder-specific cache coherency actions, e.g., writeback dirty data and invalidate clean copies.
- Once an anonymous buffer is posted for Write MSG use, the buffer contents may be non-deterministic.
- Write MSG request packets are transmitted on behalf of a Requester Context Identifier (REQCTXID) located at the Requester to a Responder Context Identifier (RSPCTXID) located at the Responder. A Write MSG request packet targets an anonymous buffer associated with a RSPCTXID.
  - Anonymous buffer management is outside of this specification's scope.
  - If a Responder's anonymous buffers are temporarily exhausted, then the Responder may silently discard Write MSG request packets for a new message, or if a reliable Write MSG operation, it may return a *Responder Not Ready (RNR) NAK*. A Responder shall not return an RNR NAK for an Unreliable Write MSG request packet.

**Developer Note:** *Applications often rely on a Responder to temporarily buffer messages destined for a RSPCTXID without a posted receive buffer or whose receive buffer cannot be identified due to missing the application-specific receive tag. Further, applications often rely on Responders to notify an application when a message arrives so that it can post a receive buffer of the correct size and have the hardware copy the message to the receive buffer.*

- A series of Write MSG request packets used to transmit a single message shall contain the same REQCTXID, RSPCTXID, and Tag field values.
- If an application-specific receive tag needs to be transmitted, then shall set RT = 1b and the RCV-Tag field shall be present. The RCV-Tag field shall be present only in the first Write MSG request packet.
  - A RCV-Tag field contains an application-specific receive tag. A receive tag is one input used by an application to select a completed Write MSG receive buffer at the Responder.
  - If a Responder does not support application-specific receive tag processing and the RCV-Tag field is present, then the Responder shall ignore the RCV-Tag field.
- The maximum message size (Max\_MSG\_Size) shall be  $(2^{11} * \text{Max_Packet_Payload})$  bytes.
- If the message size == Max\_MSG\_Size, then the MSGSZ field shall be set to 0x0.
- If the message size is less than Max\_MSG\_Size, then:
  - MSGSZ = (message size DIV Max\_Packet\_Payload).
  - If the remainder is non-zero, then MSGSZ shall be incremented by one.

**Developer Note:** *A Responder uses the MSGSZ field to locate a posted receive buffer capable of containing the message. For example, if a single receive buffer size is posted for a given RSPCTXID whose size is greater than or equal to the MSGSZ, then the Responder can select any receive buffer. In contrast, if a multiple receive buffer sizes are posted, then it can select a "best-fit" receive buffer.*

- If the MSGSZ is greater than the maximum message size supported by the RSPCTXID, then the Responder may handle this using a temporary anonymous buffer as described in an earlier Developer Note or the Responder shall handle this as a UR error and shall set the Request-specific field to the 'Insufficient Responder Resource' encoding as specified in *Reason Field Encodings*. The maximum message size is known through means outside of this specification's scope.

- Each RSPCTXID may be independently targeted by multiple REQCTXIDs. A Responder delineates Write MSG operations by the [SCID | SGCID, REQCTXID, RSPCTXID, Tag].
- At any given time, a Requester shall have no more than one Write MSG operation in-progress per [DCID | DGCID, REQCTXID, RSPCTXID, Tag].
- Applications that typically exchange relatively small messages post receive buffers that are associated with a RSPCTXID. Upon receipt, the Responder selects a buffer and places the Write MSG payload. In some solutions, applications are unable to post a sufficient number of receive buffers or use a variety of receive buffer sizes that are impractical to delineate in hardware. In such solutions, an application may post multiple relatively small receive buffers and use the Write MSG ER field to delineate between application data (ER = 0b) and an embedded read request operation (ER = 1b). By using an embedded read request, an application can avoid posting large receive buffers or having to perform local copy operations to place data into the target application buffer.
  - If an embedded read request, then the payload field contains six fields:
    - The target read data buffer length in bytes
    - An application-supplied opaque handle that uniquely identifies the embedded request at the Requester.
    - The address of the read data buffer located at the Requester that is to be read. This address is used in subsequent Read or LDM 1 Read request packet's Address field, i.e., the Responder does not use this address with its Requester ZMMU. If multiple Read or LDM 1 Read request packets are required to complete the embedded read operation, then the request packet's Address field shall contain an offset from the address of the data buffer. If the Requester supports zero-based addressing (*Core Structure Component CAP 1 Address Field Interpretation == 0b*), then the address represents the offset from byte 0 of the Requester's Data Space. If the Requester supports non-zero-based addressing (*Core Structure Component CAP 1 Address Field Interpretation == 1b*), then the address is one comprehended by the Requester
    - The R-Key to use in the subsequent Read or LDM 1 Read request packets. If the R-Key == Default R-Key, then shall set RK = 0b in the request packets.
    - The address of the completion location that is updated upon completing the embedded read operation. The Requester sets the CA bit to indicate if a 64-bit Atomic Add (increment by 1) is to be used to update this location.
    - The R-Key to use in a subsequent Atomic Add to update the completion location.
  - A Requester may transmit additional bytes beyond the indicated six fields, up to a total of Max\_Packet\_Payload bytes (inclusive of the six fields). A Responder shall treat these additional bytes as Reserved, and should deliver these additional bytes, if applicable, to the application for processing.
  - The ER and CA bits are present in unicast Write MSG and *SOD Write MSG* request packet formats.
  - Acknowledgment of a reliable Write MSG or *SOD Write MSG* shall be independent of initiating the embedded read request.

**Developer Note:** *An embedded read does not change Write MSG operation; it impacts only how the payload is interpreted by the application. The simplest way to inform an application is to return the ER and CA values in the corresponding receive buffer completion notification. The application then interprets the payload copied to the receive buffer to initiate read operations directly or to program a local hardware data mover.*

Upon embedded read completion, the application informs the peer component that the data buffer is no longer being accessed relative to this sequence of operations. Though the figure illustrates an Atomic Add (indicated by the CA bit), an application could use other techniques at its discretion.

5

To perform the embedded read operation, developers are strongly encouraged to implement a series of Read or LDM 1 Read request packets using data mover logic within the Responder.

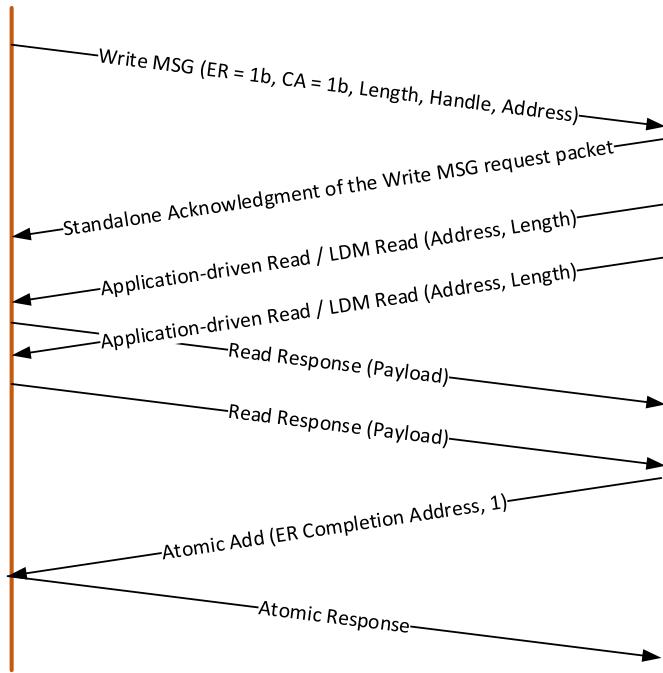
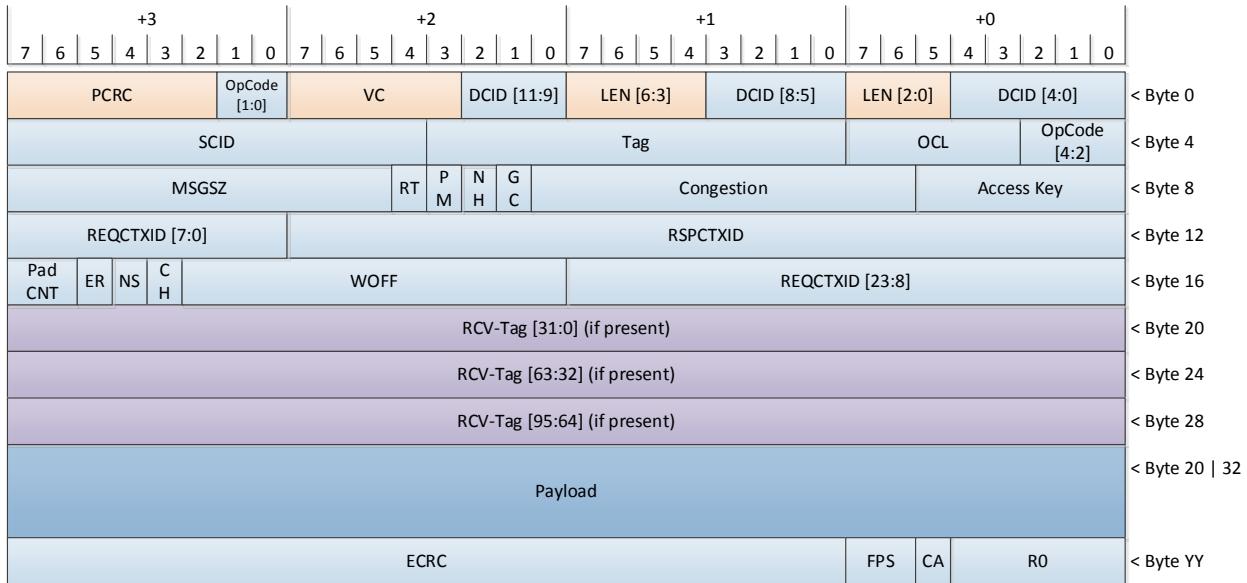
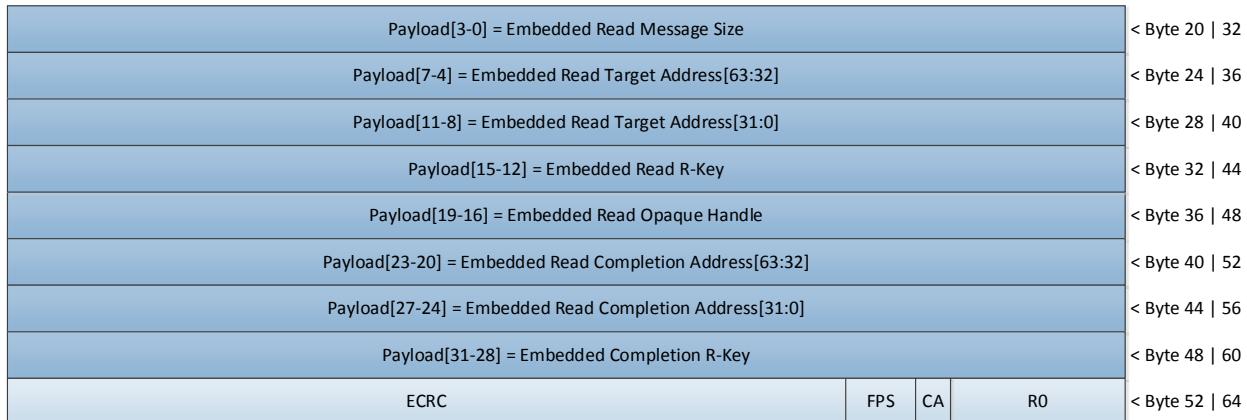


Figure 6-22: Example Write MSG with Embedded Read Packet Exchange Sequence



If ER == 0b, then Payload is Variable-sized Application Data



If ER == 1b, then Payload contains Embedded Read Fields

Figure 6-23: (Unreliable) Write MSG Request Packet Format

Table 6-12: (Unreliable) Write MSG Request-specific Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
RT	-	1	RT indicates if the RCV-Tag field is present. 0b—RCV-Tag shall not be present 1b—RCV-Tag shall be present. Only the first packet within a series of Write MSG request packets shall have RT= 1b.
RSPCTXID	-	24	Identifies the Responder context used to locate an anonymous destination receive buffer at the Responder. RSPCTXID 0x0 shall be used only in Control OpClass Write MSG request packets. RSPCTXID 0x0 is used as a well-known identifier to bootstrap management services.

Field Name	Field Abbreviation	Size (bits)	Description
REQCTXID	-	24	Control OpClass Write MSG request packets may use non-zero RSPCTXIDs.
			Identifies the Requester context used to generate the Write MSG request packet at the Requester. REQCTXID 0x0 shall be used only in Control OpClass Write MSG request packets. Control OpClass Write MSG request packets may use non-zero REQCTXIDs.
Message Size	MSGSZ	11	<p>Maximum message size targeted by this request. The actual message size shall be less than or equal to:</p> <p>If (MSGSZ == 0x0) then</p> $\text{Size} = \text{Max\_MSG\_Size} * \text{Max\_Packet\_Payload}$ <p>else</p> $\text{Size} = \text{MSGSZ} * \text{Max\_Packet\_Payload}$ <p>The MSGSZ field in each Write MSG or Unreliable Write MSG request packet associated with a given message shall contain the same value.</p>
Write Offset	WOFF	11	<p>Write Offset is used to calculate the byte offset where the payload data is placed relative to byte zero of the anonymous buffer.</p> <p>If WOFF == 0, then the payload data shall be placed starting at Byte zero of the anonymous buffer Else the payload data shall be placed starting at (Byte 0 + (WOFF * Max_Packet_Payload))</p>
ER	-	1	<p>Indicates how to interpret the payload field as application data or as an embedded read request.</p> <p>If ER == 0b, then the payload field shall be treated as application data.</p> <p>If ER == 1b, then the payload field shall be 160 bits in length and shall be interpreted as follows:</p> <ul style="list-style-type: none"> <li>• Payload[3-0] shall be the size of the message to be read by the Responder using a series of Core 64 Read or LDM 1 Read request packets</li> <li>• Payload[11-4] shall be the address of byte 0 of the data to be read. The address shall be bit-wise placed as follows: <ul style="list-style-type: none"> <li>◦ Address[63:32] in Payload[11-8]</li> <li>◦ Address[31:4] in Payload[15-12]</li> </ul> </li> <li>• Payload[15-12] shall be the R-Key to use in the subsequent Core 64 Read or LDM 1 Read request</li> </ul>

Field Name	Field Abbreviation	Size (bits)	Description
			<p>packets. If the R-Key == Default R-Key, then the subsequent Read or LDM 1 Read request packet may set RK = 0b (if RK is present).</p> <ul style="list-style-type: none"> <li>• Payload[19-16] shall be a Requester-supplied opaque handle that uniquely identifies this embedded read request. Through means outside of this specification's scope, the Responder may use this handle to inform the Requester when the embedded read request has been completed.</li> <li>• Payload[27-20] shall be the address of byte 0 the completion location to be updated once the embedded read has been completed. The address shall be bit-wise placed as follows: <ul style="list-style-type: none"> <li>◦ Address[63:32] in Payload[23-20]</li> <li>◦ Address[31:4] in Payload[27-24]</li> </ul> </li> <li>• Payload[31-28] shall be the R-Key to use to update the completion location. If the R-Key == Default R-Key, then the subsequent update request packet may set RK = 0b (if RK is present).</li> </ul> <p>0b—Application Data 1b—Embedded Read</p>
Receive Tag	RCV-Tag	96	If present, then the RCV-Tag contains an application-specific value that is used to select a specific Responder buffer.
CH	-	1	<p>CH indicates if the completion handler associated with the RSPCTXID shall be invoked upon successfully completing this Write MSG operation. If a completion handler is not associated with the RSPCTXID or cannot be invoked for any reason, then this field shall be ignored.</p> <p>A completion handler is a component-local mechanism, e.g., a local interrupt that is associated with the RSPCTXID through a mechanism that is outside of this specification's scope.</p> <p>0b—Do Not Invoke 1b—Invoke</p>
CA	-	1	<p>If ER == 1b and CA == 1b, then upon completing the embedded read operation and if supported, then the Responder shall perform a 64-bit Atomic Add (see <i>Atomic Request and Response Packets</i>) at the provided completion location. If a Responder is unable to perform the Atomic Add, then it shall return a <i>Standalone Acknowledgment</i> (Reason = UR.)</p> <p>If CA == 0b, then the Responder shall take no action upon completing the embedded read operation.</p>

Field Name	Field Abbreviation	Size (bits)	Description
R0	-	5	Reserved

## 6.4. Write Poison

A Write Poison request is used to write poisoned data to a specified address and ensures the Responder is aware that the data is poisoned. Poison implementation within a Responder is outside of this specification's scope.

5 The following are the features and requirements associated with Write Poison packets:

- P2P-Core Write Poison packets shall use the P2P-Core Write *P2P-Core Write Poison Request Packet Format*.
- P2P-Core Write Poison packets shall use the P2P-Core Write *P2P-Coherency Write Poison Request Packet Format*.
- 10 • Core 64 Write Poison packets shall use the Core 64 Write *Core 64 Write Poison Request Packet Format*.
- A Responder shall transmit a *Standalone Acknowledgment* for each Write Poison request packet.
- 15 • If the addressed resource is a non-volatile media, then the Responder shall not transmit the *Standalone Acknowledgment* until the poison request packet has been successfully executed and the poison processing state is guaranteed to succeed or an error has been detected.
- Unless explicitly stated otherwise by this specification, a Responder shall transmit a *Standalone Acknowledgment* for each received Write Poison request packet.
  - o If *Interface I-CAP 1 Control Omit P2P Standalone Acknowledgment* == 1b, then the Responder shall not transmit a *Standalone Acknowledgment* for a successfully validated and executed P2P-Core Write Poison or a P2P-Coherency Write Poison request packet that does not contain the PU field or PU == 0b; if PU == 1b, then the Responder shall transmit a *Standalone Acknowledgment*.
- 20 • If the Poison Range is greater than the Responder's *Core Structure Component CAP 2 Poison Granularity Support*, then the Responder shall not execute the request packet; the Responder shall return a *Standalone Acknowledgment* (Reason = MP).
- 25 • A Write Poison Request shall not be acknowledged until the request has been successfully executed or an error has been detected.
- A Responder shall not update the underlying media until the packet has been successfully validated. If a packet experiences a subsequent execution error then the underlying media contents may be non-deterministic.
- 30 • A Write Poison shall be a unicast end-to-end request packet.
- If the addressed media targeted by a Write Poison request packet has entered Fatal Media Error Containment, then the Responder shall not execute the request packet; the Responder shall return a *Standalone Acknowledgment* (Reason = Fatal Media Error Containment Triggered).
- 35 • If a Responder supports a cache resource, then upon receipt of a Write Poison request packet with TC == 1b, the Responder shall poison the corresponding cache line.
  - o If a Responder's cache agent determines that the Requester is not permitted to poison the addressed cache line (e.g., is not the current owner), then the Responder shall handle this as an "Access Error (AE)—Invalid Access Permission".

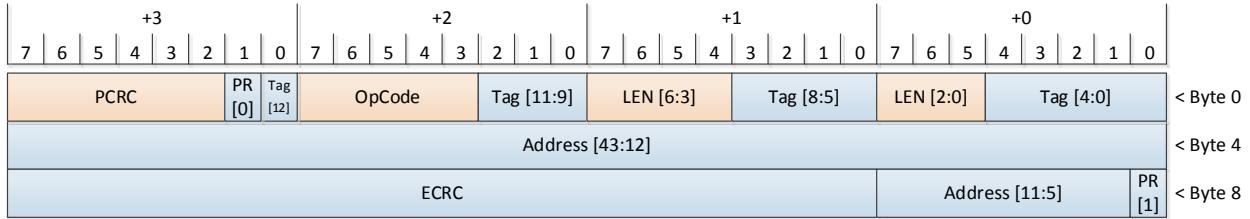


Figure 6-24: P2P-Core Write Poison Request Packet Format

Table 6-13: P2P-Core Write Poison Request-specific Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
PR	Poison Range	2	Number of bytes to be poisoned starting with the byte identified by the packet's Address 0x0—16 bytes 0x1—32 bytes 0x2—64 bytes 0x3—256 bytes

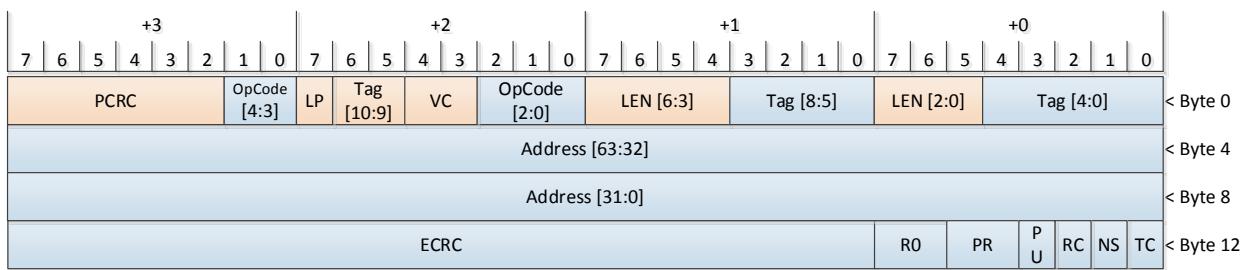


Figure 6-25: P2P-Coherency Write Poison Request Packet Format

Table 6-14: P2P-Coherency Write Poison Request-specific Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
PR	Poison Range	2	Number of bytes to be poisoned starting with the byte identified by the packet's Address 0x0—16 bytes 0x1—32 bytes 0x2—64 bytes 0x3—256 bytes
R0	-	2	Reserved

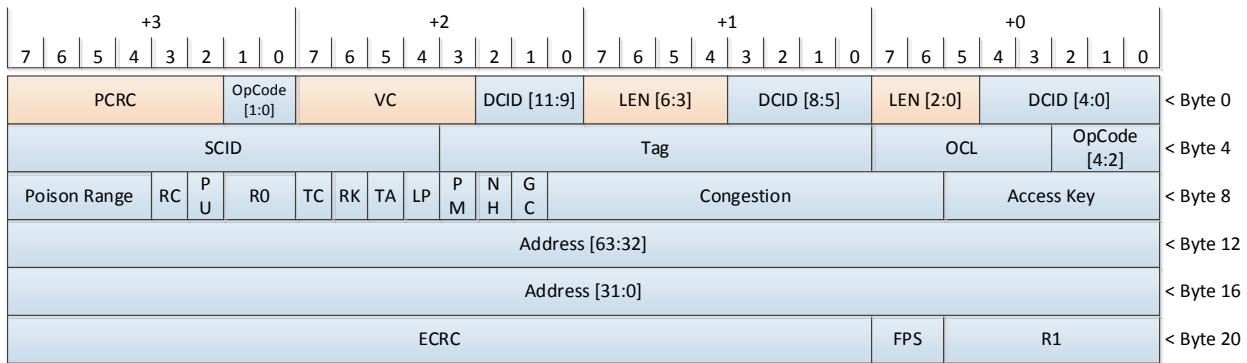


Figure 6-26: Core 64 Write Poison Request Packet Format

Table 6-15: Core 64 Write Poison Request-specific Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>Poison Range</b>	-	4	Number of bytes to be poisoned starting with the byte identified by the packet's Address 0x0—16 bytes 0x1—32 bytes 0x2—64 bytes 0x3—128 bytes 0x4—256 bytes 0x5—512 bytes 0x6—1024 bytes 0x7—2048 bytes 0x8—4096 bytes 0x9-0xF—Reserved
<b>R0</b>	-	4	Reserved
<b>R1</b>	-	5	Reserved

## 6.5. Write-Under-Mask (WUM)

A Write-Under-Mask (WUM) request is used to selectively update a set of bits within a two-byte subrange at the indicated Address.

The following are the features and requirements associated with WUM packets:

- Core 64 WUM request packets shall use the *Core 64 WUM Packet Format*.
- The Responder shall return a Core 64 *Standalone Acknowledgment* for each Core 64 WUM request packet.
- Address[0] shall be set to 0b.
- The Responder modifies the two-byte subrange as follows:
  - Old = Read(Data)
  - New = (Old &  $\sim$ Mask) | (Operand & Mask)
  - Write(Address, New)

- If a Responder supports a cache resource, then upon receipt of a WUM request packet with TC == 1b, the Responder shall place the packet's payload in the cache. Placement may require the cache to evict existing cache lines to free up space and, if supported, then the Responder performing cache coherency actions.
  - If a Responder's cache agent determines that the Requester is not permitted to modify the addressed cache line (e.g., is not the current owner), then the Responder shall handle this as an "Access Error (AE)—Invalid Access Permission"

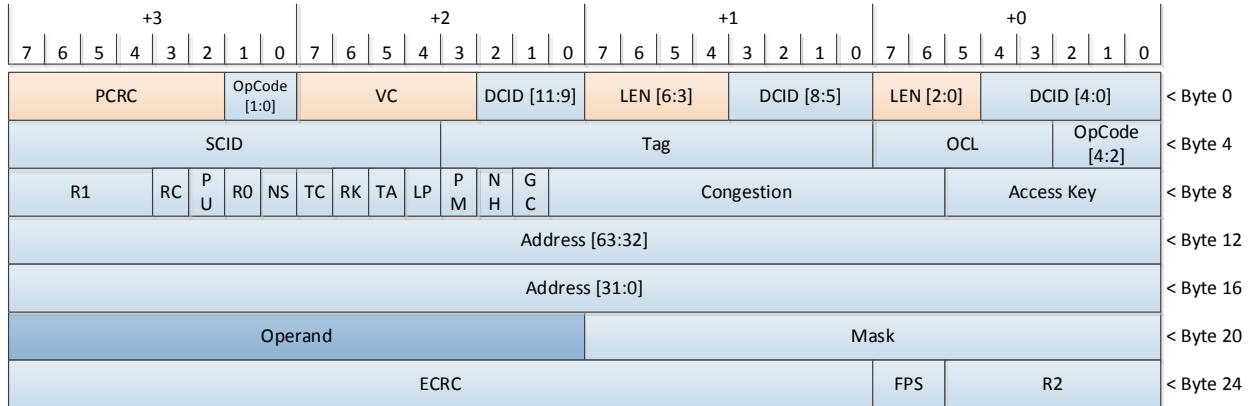


Figure 6-27: Core 64 WUM Packet Format

Table 6-16: Core 64 WUM Packet-specific Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>Mask</b>	-	16	Mask indicates which bits within the two-byte subrange are to be modified.
<b>Operand</b>	-	16	Operand contains the updated bit values within the two-byte subrange.
<b>R0</b>	-	1	Reserved
<b>R1</b>	-	4	Reserved
<b>R2</b>	-	6	Reserved

## 6.6. Write Partial

A Write Partial request is used to selectively update a set of bytes within a 64-byte subrange starting at the indicated Address. Write Partial eliminates the need to explicitly perform read-modified-write operations.

The following are the features and requirements associated with Write Partial packets:

- P2P-Core Write Partial request packets shall use the *P2P-Core Write Partial Packet Format*.
- P2P-Coherency Write Partial request packets shall use the *P2P-Coherency Write Partial Packet Format*.
- Core 64 Write Partial request packets shall use the *Core 64 Write Partial Packet Format*.

15

- Unless explicitly stated otherwise by this specification, a Responder shall transmit a *Standalone Acknowledgment* for each received Write request packet.
  - If *Interface I-CAP 1 Control Omit P2P Standalone Acknowledgment* == 1b, then the Responder shall not transmit a *Standalone Acknowledgment* for a successfully validated and executed P2P-Core Write Partial or P2P-Coherency Write Partial request packet with PU == 0b.
    - If a P2P-Core Write Partial or P2P-Coherency Write Partial request packet suffers a non-transient error and *Component Containment* has not been enabled, then the Responder shall return a *Standalone Acknowledgment* (Reason = error detected).
    - Upon detecting a non-transient error, the Responder should stop executing new request packets that modify Data Space resources. If *In-band Management* is supported, then the Responder shall execute new P2P-Core or P2P-Coherency *CTL-Read and CTL-Write Packets*.
- If UR == 0b or PU == 1b in a Core 64 Write Partial request packet, then the Responder shall transmit a Core 64 *Standalone Acknowledgment* for each Core 64 Write Partial request packet.
- Write Partial request packets shall target integer 64-byte address multiples, i.e., the address field within a P2P-Core request packet shall set bits [1:0] to zero, and within P2P-Coherency and Core 64 request packets, shall set bits [5:0] to zero.
- The Payload field shall contain 64 bytes of data. Only bytes indicated by the Partial Bit Mask shall be considered valid; all other bytes are “don’t care”.
- The Responder modifies the addressed 64-byte subrange as follows:
  - For each  $Bit_k == 1b$  within the Partial Bit Mask, Payload Byte K shall be written to (Address + K).
  - For each  $Bit_k == 0b$  within the Partial Bit Mask, (Address + K) shall not be modified and Payload Byte K shall be ignored.
- If a Responder supports a cache resource, then upon receipt of a Write Partial request packet with TC == 1b, the Responder shall place the packet’s payload in the cache. Placement may require the cache to evict existing cache lines to free up space and, if supported, then the Responder performing cache coherency actions.
  - If a Responder’s cache agent determines that the Requester is not permitted to modify the addressed cache line (e.g., is not the current owner), then the Responder shall handle this as an “Access Error (AE)—Invalid Access Permission”.

Table 6-17: Common Write Partial Packet-specific Fields

Field Name	Size (bits)	Description
Partial Bit Mask	64	Bit mask that indicates which bytes within the payload are written to the addressed resource.

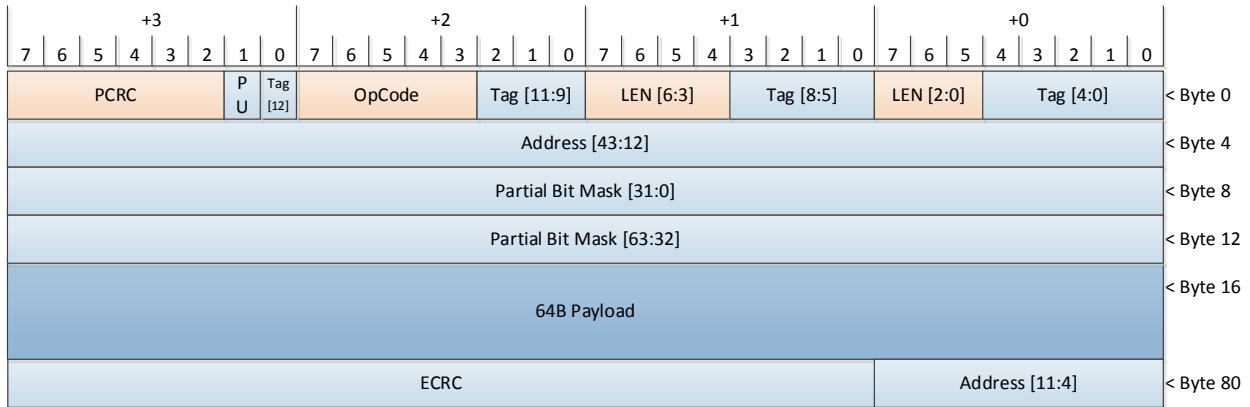


Figure 6-28: P2P-Core Write Partial Packet Format

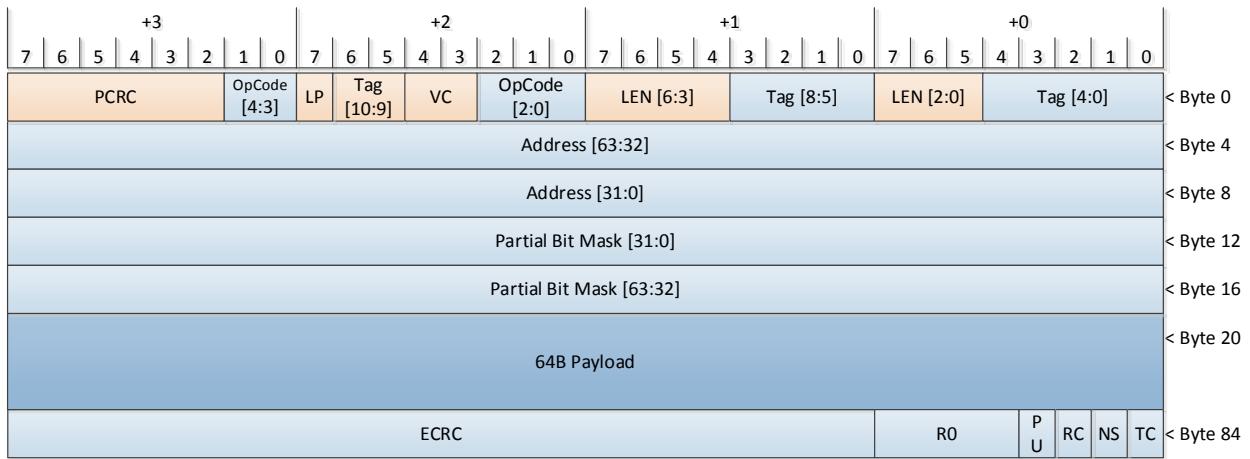


Figure 6-29: P2P-Coherency Write Partial Packet Format

Table 6-18: P2P-Coherency Write Partial Packet-specific Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>R0</b>	-	4	Reserved

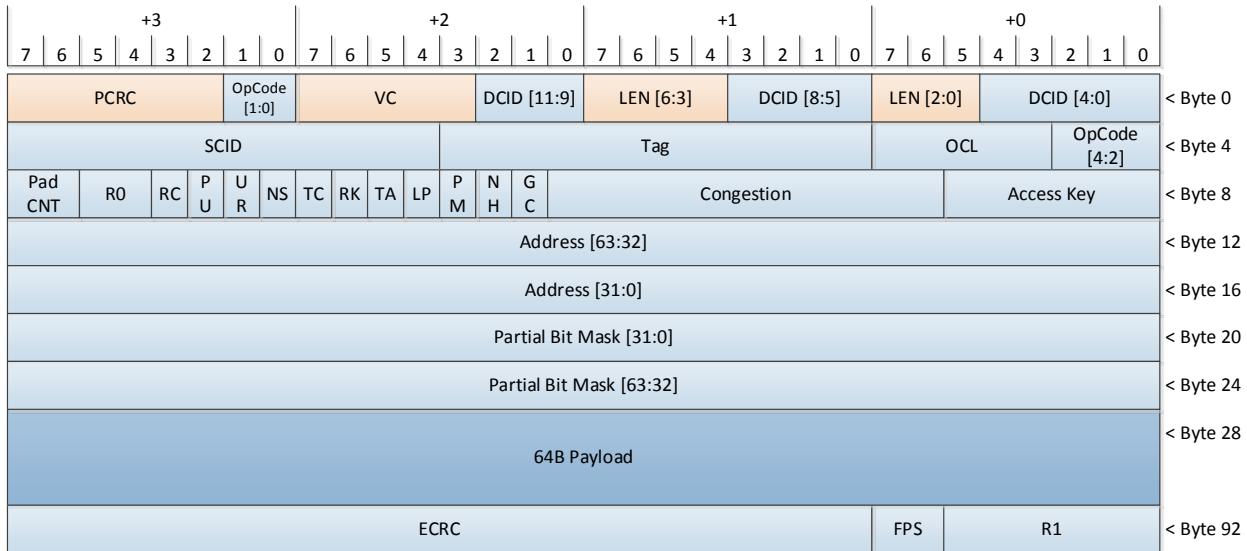


Figure 6-30: Core 64 Write Partial Packet Format

Table 6-19: Core 64 Write Partial Packet-specific Fields

Field Name	Field Abbreviation	Size (bits)	Description
R0	-	2	Reserved
R1	-	6	Reserved

## 6.7. Meta Read Response and Meta Write

5 Meta Read and Meta Write request packets enable a Requester to transmit a quantity of Meta data in conjunction with the respective read or write operation. Meta data communicates additional information to a Responder to facilitate solution operation. For example, a Requester moving a 4 KiB block of data wants to send an additional 64-bit block-level data integrity value could transmit 15 256-byte Write request packets and 1 256-byte Meta Write request packet to complete the data transfer and communicate the data integrity value. Upon receipt of the Meta Write request packet, the

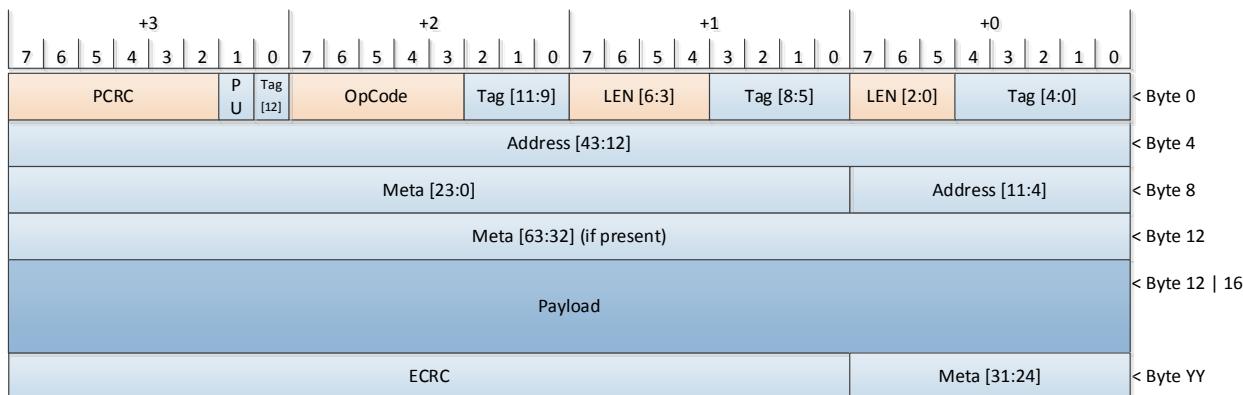
10 Responder uses the 64 bits of Meta data to perform block-level data integrity. Similarly, a solution could use a mix of Read Response / LDM 1 Read Response packets and Meta Read Response packets / LDM 1 Read Response packets with the Meta field present to efficiently communicate additional Responder information.

15 Use of Meta data is not constrained to block-level data integrity; there are many possible use cases, e.g., the Meta field can communicate control information to a memory module that contains a DRAM cache and SCM or Flash media. The Meta field in a Meta Write request packet could guide the media controller on what data to cache in the DRAM or guide cache prefetch algorithms to reduce subsequent read or write latency and associated jitter. Similarly, the Meta field in a Meta Read Response packet could provide media status such as read and write NVM endurance levels, remaining time at current workload level until garbage collection is required, etc.

20 The following are the features and requirements associated with Meta Read and Meta Write packets:

- A P2P-Core Read Response packet indicates if the Meta field is present (see *P2P-Core Read Response Packet Format*).
- A P2P-Coherency Read Response packet indicates if the Meta field is present (see *P2P-Coherency Read Response Packet Format*).
- 5 • A Core 64 Read Response packet indicates if the Meta (MS) field is present (see *Core 64 Read Response Packet Format*).
- A LDM 1 Read Response packet indicates if the Meta (MS) field is present (see *LDM 1 Read Response Packet Format*).
- P2P-Core Meta Write request packets shall use the *P2P-Core Meta Write Packet Format*.
- 10 • P2P-Coherency Meta Write request packets shall use the *P2P-Coherency Meta Write Packet Format*.
- A Core 64 Write request packet indicates if the Meta (MS) field is present (see *Core 64 Write Request Packet Format*).
- 15 • Unless explicitly stated otherwise by this specification, a Meta Read Response packet semantics shall be as a Read Response packet as specified in *Read and Read Response*.
- Unless explicitly stated otherwise by this specification, a Meta Write request packet semantics shall be as a Write request packet specified in *Write*.
- The contents and semantics of the packet's Meta field are outside of this specification's scope, and may be vendor-defined or based on a de facto or industry standard specification.

20 **Developer Note:** Developers are strongly encouraged to provision a Vendor-Defined Structure with UUID associated with the Component Media Structure, or to use the Core Structure C-UUID or one of UUIDs within the Service UUID Structure to identify the supported Meta field contents and semantics. Identification is indicated by Core Structure Component CAP 2 Meta Read-Write Support.



25 Figure 6-31: P2P-Core Meta Write Packet Format

Table 6-20: P2P-Core Meta Write Packet-specific Fields

Field Name	Size (bits)	Description
Meta	-	Meta Data—if the Responder is unable to interpret the contents of this field, then it shall treat this field as Reserved. Meta field size is indicated by the OpCode.

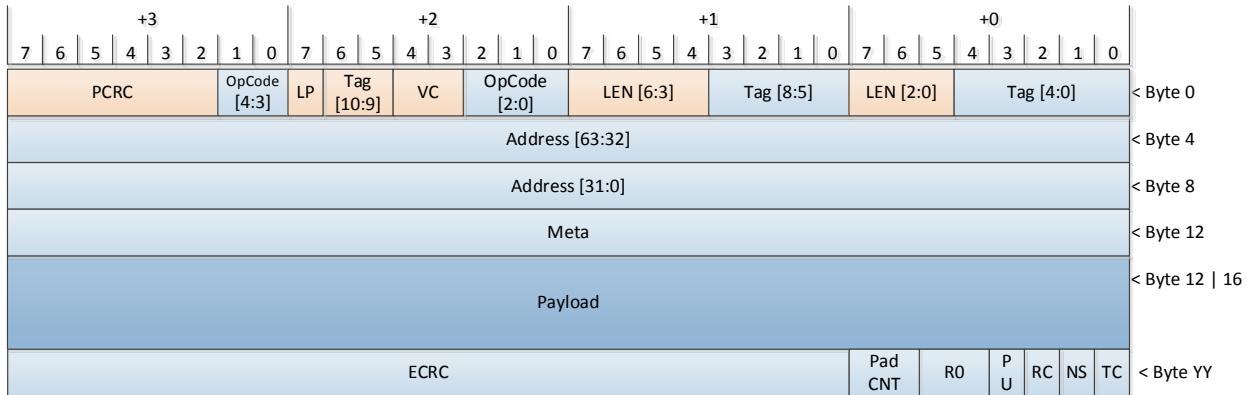


Figure 6-32: P2P-Coherency Meta Write Packet Format

Table 6-21: P2P-Coherency Meta Write Packet-specific Fields

Field Name	Size (bits)	Description
Meta	-	Meta Data—If the Responder is unable to interpret the contents of this field, then it shall treat this field as Reserved. Meta field size is indicated by the OpCode.
R0	2	Reserved

## 6.8. Standalone Acknowledgment

An acknowledgment packet is used to positively or negatively acknowledge a request packet. A positive acknowledgment communicates that the request packet identified by the Tag was successfully received, validated, and executed. A negative acknowledgment communicates that an issue was detected (validation or execution) with the request packet identified by the Tag.

The following are the features and requirements associated with Acknowledgments:

- All components shall support unicast Standalone Acknowledgment.
  - A component may support unicast Standalone Acknowledgment for a non-deterministic request. See *Non-Deterministic Request Execution*.
  - Interfaces that support P2P-Core, P2P-Coherency, or P2P-Vendor-defined may be configured to omit transmitting a Standalone Acknowledgment when no error is detected for specific request packet types that access Data Space. See *Interface I-CAP 1 Control Omit P2P Standalone Acknowledgment*.
- Packet formats:
  - P2P-Core Standalone Acknowledgment shall use the *P2P-Core Standalone Acknowledgment Packet*, and shall be used only in response to P2P-Core request packets.
  - P2P-Coherency Standalone Acknowledgment shall use the *P2P-Coherency Standalone Acknowledgment Packet*, and shall be used only in response to P2P-Coherency request packets.
  - Core 64 Standalone Acknowledgment shall use the *Core 64 Standalone Acknowledgment Packet*. Core 64 Standalone Acknowledgment is used to acknowledge or communicate an error for a variety of explicit OpClass request packets.

- Standalone Acknowledgments shall not be acknowledged.
- Requesters shall examine the Reason field.
  - (Reason = No Error) indicates that the request packet was successfully validated and executed without any other secondary conditions detected.
  - A non-zero Reason value indicates that the request was not executed, or that the request packet was executed but a secondary condition was detected. The Reason field shall indicate the highest-precedence error or secondary condition that was detected.
  - All Requesters and Responders shall support the *Responder Not Ready (RNR) NAK*. A Responder may return a *Responder Not Ready (RNR) NAK* for any request packet that it is presently unable to execute due to a transient operating condition, e.g., due to Responder-specific resource shortages.
- Requesters shall silently discard Standalone Acknowledgments do not correspond to any Outstanding Request Packets.
- If a Standalone Acknowledgment is used to indicate the successful execution of a request packet, then a Requester shall await receipt of a Standalone Acknowledgment before making ordering decisions that depend upon the successful execution of a request packet.
- If a Requester receives a P2P-Core or P2P-Coherency Standalone Acknowledgment with a Reason that indicates an error occurred and the packet's Tag does not correspond to any Outstanding Request Packet, then the Requester shall invoke response packet error processing. If the component supports the *Component Error and Signal Event Structure*, then the configured actions that correspond to the error shall be taken. This situation can occur if *Interface I-CAP 1 Control Omit P2P Standalone Acknowledgment == 1b*.
- If *Interface I-CAP 1 Control Omit P2P Standalone Acknowledgment == 1b* and a Requester used the same Tag value on the same interface in multiple request packets and subsequently receives a NAK, then the NAK may apply to any of the request packets that used the NAK's Tag value.

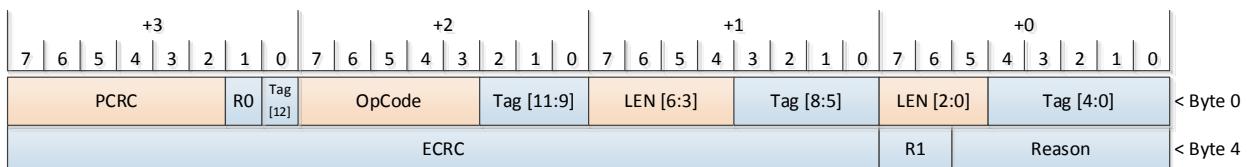


Figure 6-33: P2P-Core Standalone Acknowledgment Packet

Table 6-22: P2P-Core Standalone Acknowledgment Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
Reason	-	6	Acknowledgment Reason Code. See <i>Reason Field Encodings</i> .
R0	-	1	Reserved
R1	-	2	Reserved

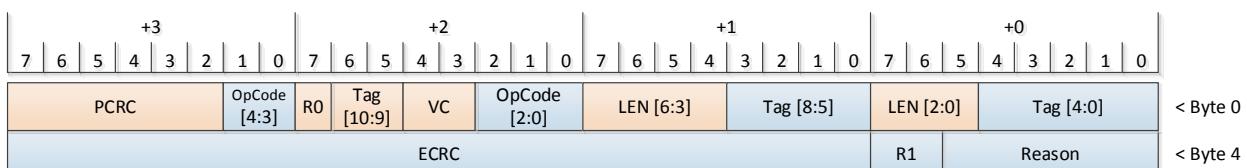


Figure 6-34: P2P-Coherency Standalone Acknowledgment Packet

Table 6-23: P2P-Coherency Standalone Acknowledgment Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
Reason	-	6	Acknowledgment Reason Code. See <i>Reason Field Encodings</i> .
R0	-	1	Reserved
R0	-	2	Reserved

Figure 6-35: Core 64 Standalone Acknowledgment Packet

Table 6-24: Core 64 Standalone Acknowledgment Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
Reason	-	6	Acknowledgment Reason Code. See <i>Reason Field Encodings</i> .
RNR   QD	-	3	If (Reason == RNR 0 or RNR 1), then this field contains RNR Time Interval—see <i>Responder Not Ready (RNR) NAK</i> .  If (Reason == No Error Write MSG) and this field is non-zero, then this field shall indicate the RSPCTXID queue depth, i.e., the relative number of anonymous buffers posted to the RSPCTXID receive queue represented as a percentage of the maximum queue depth. Queue depth can be returned through a Requester completion queue to be acted upon by software. Queue depth is the approximate percentage at the time the acknowledgment was scheduled. QD is encoded as follows:  0x0—Empty Queue 0x1—Less than or equal to 12.5% full 0x2—Less than or equal to 25% full 0x3—Less than or equal to 50% full 0x4—Less than or equal to 62.5% full 0x5—Less than or equal to 75% full 0x6—Less than or equal to 87.5% full 0x7—Greater than 87.5% full

Field Name	Field Abbreviation	Size (bits)	Description
Request-specific	RS	11	<p>This field is used to convey request-specific information in response to select request packets, e.g., the Write MSG request packet.</p> <p>If the contents of this field are not specified by the corresponding request packet, then this field shall be Reserved.</p>

Reason Field Encodings specifies encodings for Standalone Acknowledgments as well as a variety of response packet types that contain a similar field.

Unless explicitly stated otherwise by this specification, a Reason code may be used in a response packet.

Reason Class Legend:

- NE—No Error
- TE—Transient Error
- NTE—Non-Transient Error
- TC—Transient Condition
- NTC—Non-transient Condition

Table 6-25: Reason Field Encodings

Encoding	Reason Class	Description
0x0	NE	No Error (NE). Request was successfully executed. No errors, conditional issues, etc. were detected.
	NTE	Unexpected Packet (UP) Error—Shall not be returned in any response packet
	NTE	Unsupported Request (UR) Error
	NTE	Malformed Packet (MP) Error
	TE	Packet Execution Error (EXE) Non-Fatal. The Responder experienced a recoverable execution error while processing this specific request. The Requester may retry the request.
	NTE	Packet Execution Error (EXE) Fatal. The Responder experienced an unrecoverable execution error while processing this specific request. The Requester should not retry the request.
	NTE	Access Error (AE)—Invalid Access Key
	NTE	Access Error (AE)—Invalid Access Permission
	NTC	<i>Component Containment Triggered (CCT)</i>
	NTC	<i>Interface Containment Triggered (ICT)</i>
0xA	TC	RNR 0 (Responder Not Ready 0)—see <i>Responder Not Ready (RNR) NAK</i> .

Encoding	Reason Class	Description
0xB		Reason equal to RNR 0 shall be returned only in a <i>Standalone Acknowledgment</i> or <i>SOD Standalone Acknowledgment</i> .
	TC	RNR 1 (Responder Not Ready 1)—see <i>Responder Not Ready (RNR) NAK</i> . Reason equal to RNR 1 shall be returned only in a <i>Standalone Acknowledgment</i> or <i>SOD Standalone Acknowledgment</i> .
0xC	TC	Data Correctable Error Warning (DCE)—the request packet was successfully executed, however, the addressed media associated with this response suffered a correctable error. The media's error condition was corrected by the Responder.
0xD	NTC	Data Uncorrectable Error Detected (DUC)—the request packet was successfully executed, however, the addressed media associated with this response suffered an uncorrectable error. The media's error condition could not be corrected by the Responder.
0xE	NTC	Poison Data (PD) Detected—the request packet was successfully executed, however, the response packet's payload contains Poisoned data.
0xF	NTE	Poison Data (PD) Failure—request packet execution failed due to Poison data detection.
0x10	TC	In-progress Sanitize and Erase (SE) Operation—the request packet was not executed. The Requester should not retransmit the request packet until the SE operation has completed. See <i>Component Media Structure</i> .
0x11	NTE	Fatal Media Error Containment Triggered (FMCT)—the request packet was not executed. See <i>Component Media Structure</i> .
0x12	TC	Emergency Power Reduction (EPR)—the request packet was not executed due to component Emergency Power Reduction. See <i>Component Power Management</i> .
0x13	NTC	Insufficient Space (ISP)—the request packet was not executed due to insufficient resources to successfully allocate the requested buffer size
0x14	NTC	Abort Transition (ATRANS)—the Requester shall abort its requested C-State transition (See <i>C-State Power Control</i> ), and transition to the C-State returned in the Request-specific field.
0x15	NTE	Unsupported Service for Addressed Resource (US)—the request packet failed due to the addressed resource does not support the requested service
0x16	TC	Insufficient Responder Resources (IRR)—the request packet failed due to a lack of Responder resources to execute the request packet, e.g., an anonymous buffer of the requested size is not available.
0x17	NE	Exclusive Granted (EG)—though Read Shared was requested, the Responder has granted the Requester exclusive access. If the Responder receives a subsequent <i>Read Shared</i> request packet, it may demote the cache line state

Encoding	Reason Class	Description
0x18		to shared (see <i>Cache Line Attribute</i> ) or invalidate the cache line (see <i>Invalidate and Writeback</i> ).
0x18	NE	Unable to Grant Exclusive / Shared Access (UGES)—the Responder successfully validated the request packet, but was not able to grant exclusive or shared ownership of the addressed cache line to the Requester.
0x19	NE	Wake Failure (WF)—the Responder failed to awaken the indicated process or thread.
0x1A	TE	SOD Transient Error (SODTE)—the Responder detected an out-of-order packet. The <i>SOD Standalone Acknowledgment</i> indicates the last successfully received packet. The Requester should begin packet retransmission.
0x1B	NTE	Access Error (AE)—Invalid Capabilities Access
0x1C	NE	Resource Release (RR)—the Responder received an unreliable unicast request packet and is notifying the Requester that it may release and reuse any resources associated with this request packet, e.g., a Tag value. This Reason code does not acknowledge the success or failure of the associated unicast request packet.
0x1D	NTC	Component Power-off Transition (CPWRT)—the request packet was not executed due to component power being turned off.
0x1E	NE	No Error Write MSG (NEMSG)—The Write MSG request packet was successfully received and executed. The RS and RNR / QD fields are not Reserved, i.e., they contain valid information.
0x1F	NTC	Media Endurance Exceeded (MEDEE)—the request packet was not executed due to media wear exceeding its maximum read or write endurance.
0x20	NE	No Error, Contact Home Agent for Cache Line (NEHA)—the Requester should contact the Home Agent to acquire the cache line.
0x21	NTC	Persistent Flush Update Failure (PBUF)—the Responder was unable to execute the <i>Persistent Flush</i> request packet due to execution failure of a previously received request packet that modified a persistent addressable resource. Prior to transmitting the <i>Persistent Flush</i> request packet, the Requester needs to clear <i>Core C-Status Cannot Execute Persistent Flush</i> .
0x22	NTE	Maximum Request Packet Retransmission Exceeded (MTRANSE)—Shall not be returned in any response packet
0x23-0x3E	-	Reserved
0x3F	NE	Non-Deterministic Acknowledgment (ND ACK)—Indicates that the Responder has received and validate the request packet, and will transmit a second <i>Standalone Acknowledgment</i> or request-specific response packet upon executing the corresponding request packet.

### 6.8.1. Responder Not Ready (RNR) NAK

An RNR NAK is a *Standalone Acknowledgment* or a *SOD Standalone Acknowledgment* with (Reason = RNR 0 or RNR 1). An RNR NAK indicates a transient condition (e.g., Responder-specific resource shortage) that prevents the Responder from executing the explicit OpClass request packet within the expected time frame and temporary fabric backpressure is insufficient to resolve the transient condition.

The following are the features and requirements associated with RNR NAKs:

- A Responder may return an RNR NAK for any supported explicit OpClass request packet. A Responder shall not return an RNR NAK for any P2P-Core or P2P-Coherency request packet.
- An RNR NAK reflects a transient operating condition, and shall not trigger error handling and recovery.
- Upon receiving an RNR NAK, if the Requester intends to retransmit the request, then:
  - The Requester shall wait at least the indicated RNR time interval before retransmitting the explicit OpClass request packet.
  - A Requester shall restart the retransmission timer when retransmitting an explicit OpClass request packet that previously received an RNR NAK.
  - An RNR NAK may indicate any time interval specified in *RNR NAK Time Interval Encoding*.
    - The Responder determines the time interval through implementation-specific means. For example, a Responder might select an encoding based on the explicit OpClass request packet's type, the number of request packets being processed, or other operating conditions.
  - A Requester shall set the explicit OpClass request packet's FPS field based on whether the *Standalone Acknowledgment* (Reason == RNR 0 or RNR 1). See *Forward Progress Screen (FPS)*.

Table 6-26: RNR NAK Time Interval Encoding

Encoded Time	Time (ns)
0x0	0 (immediate retransmission is permitted)
0x1	100
0x2	250
0x3	500
0x4	1000
0x5	5000
0x6	10000
0x7	100000

## 6.8.2. Forward Progress Screen (FPS)

A FPS is used to ensure application forward progress in the face of activity patterns that could otherwise result in resource starvation or live lock due to non-deterministic forward-progress in the fabric protocol. Non-deterministic forward-progress results when a Requester suffers repeated *Responder Not Ready (RNR) NAK* responses to one request packet while request packets from the same or other Requesters do not.

If a Responder returns an RNR NAK due to request-specific resource shortages that prevent execution of an explicit OpClass request packet, e.g., resources associated with non-idempotent request packet processing, then the Responder shall support at least one FPS.

If a Responder never returns an RNR NAK, then the Responder shall not support a FPS and shall ignore the FPS field in any request packet.

Requesters shall set the FPS field in request packets as follows:

- If a Requester transmits a new request packet or retransmits a request packet for any reason other than receipt of an RNR NAK, then the Requester shall set FPS = No Epoch.
- If a Requester retransmits a request packet due to receiving an RNR NAK with (Reason == RNRO), then the Requester shall set FPS = Epoch 0.
- If a Requester retransmits a request packet due to receiving an RNR NAK with (Reason == RNR1), then the Requester shall set FPS = Epoch 1.

A Responder may support zero or more FPSs. If a Responder supports FPS, then:

- Each FPS shall be associated with one or more request-specific resource types, e.g., to support non-idempotent operations such as Atomics, a Responder provisions resources to hold Atomic execution results that are returned upon receipt of a retransmitted request packet.
- A Responder shall maintain an implementation-specific Current Epoch and a Prior Epoch value for each implemented FPS.
  - The Current Epoch and the Prior Epoch shall alternate between Epoch 0 and Epoch 1 such that the Current Epoch is never equal to the Prior Epoch.
  - Unless explicitly stated otherwise by this specification, a Current Epoch and Prior Epoch pair may be implemented per request-specific resource type, for any subset of request-specific resource types, or for all request-specific resource types.
- A Responder shall reserve a subset of the request-specific resources used to execute request packets. If a Responder supports multiple request-specific resource types, then the Responder shall reserve a subset of each type. For example, if a Responder supports Atomics and Buffer operations, then the Responder provisions a set of resources to hold Atomic execution results and a set of resources to hold Buffer execution results.
  - The reserved resources shall be non-zero and may equal the entire request-specific resources. Reserved resource management is outside of this specification's scope.
  - Available reserved or unreserved resources shall be used to execute request packets whose FPS field corresponds to the Prior Epoch.
    - If there are no reserved or unreserved resources available to execute the request packet, then unless explicitly stated otherwise by this specification, the Responder shall transmit an RNR NAK.
  - Only available unreserved resources shall be used to execute request packets whose FPS field corresponds to the Current Epoch, or whose FPS field == No Epoch.

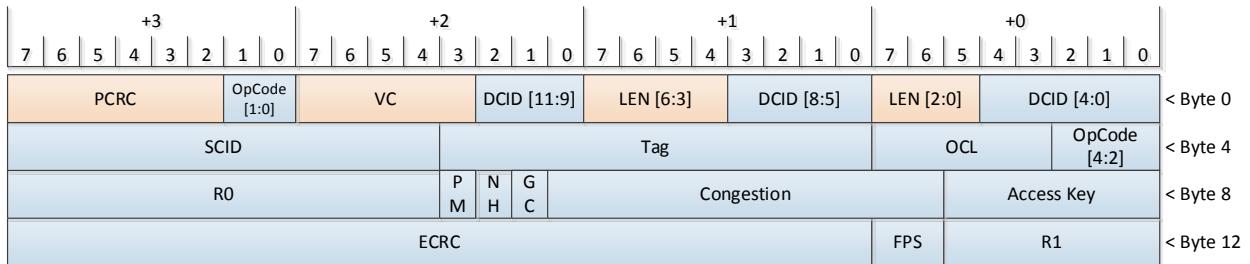
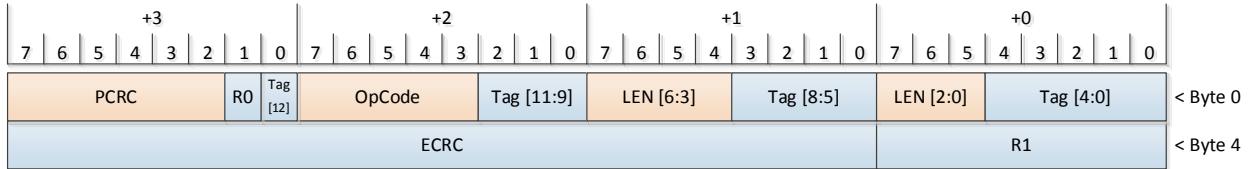
- If there are no unreserved resources available to execute the request packet, then unless explicitly stated otherwise by this specification, the Responder shall transmit an RNR NAK.
    - The unreserved resources may be used without restriction at all times.
- If a Responder transmits an RNR NAK for a request packet with FPS == No Epoch and the Current Epoch == Epoch 0, then the Responder shall set (Reason = RNR 0).
- If a Responder transmits an RNR NAK for a request packet with FPS == No Epoch and the Current Epoch == Epoch 1, then the Responder shall set (Reason = RNR 1).
- If a Responder transmits an RNR NAK for a request packet with FPS == Epoch 0, then the Responder shall set (Reason = RNR 0).
- If a Responder transmits an RNR NAK for a request packet with FPS == Epoch 1, then the Responder shall set (Reason = RNR 1).
- A Responder shall maintain one FPS Timer per FPS. Each Current Epoch / Prior Epoch pair shall be associated with a single FPS Timer.
  - A FPS Timer shall run continuously while the component is in the C-Up or the C-LP state.
  - Upon FPS Timer expiration:
    - If Current Epoch == Epoch 0, then the Current Epoch and Prior Epoch values shall be updated as follows: Current Epoch = Epoch 1 and Prior Epoch = Epoch 0
    - If Current Epoch == Epoch 1, then the Current Epoch and Prior Epoch values shall be updated as follows: Current Epoch = Epoch 0 and Prior Epoch = Epoch 1
  - A FPS Timer shall be reset to zero and restarted whenever:
    - Current Epoch is updated
    - Upon receipt of a request packet with FPS == Prior Epoch that cannot be executed due to insufficient request-specific resources.
  - The FPS Timer shall not be modified upon receipt of a request packet with FPS == No Epoch or FPS = Current Epoch.
- For FPSs other than a PCO FPS:
  - The FPS Timer shall be configured using the *Core Structure* FPST field. If multiple FPS Timers are supported, then the FPS Timers use the same FPST.
  - The FPST shall be configured with a value that is greater than the sum of the maximum Requester retransmission timer of any peer Requester plus the maximum RNR NAK Time Interval returned in any Responder RNR NAK response packet (see *Core Structure* Max RNR) plus the maximum one-way transmission time of a request packet from any peer Requester.
- For a PCO FPS (See *Protocol Deadlock Avoidance for PCO*), the PCO FPS Timer shall be configured using the *Core Structure* PCO FPST field. The PCO FPST shall be configured with a value that is greater than the sum of:
  - the maximum one-way transmission time of a PCO RNR NAK to reach a peer Requester via a PCO control VC, plus
  - the maximum time for a Requester to process a PCO RNR NAK and retransmit the request packet, plus
  - the maximum one-way transmission time of a PCO request packet from a peer Requester via a PCO traffic VC.

## 6.9. Persistent Flush

A Persistent Flush request is used to ensure that all prior updates (e.g., writes, Atomics, etc.) to a given Responder's addressable persistent resources have been successfully completed.

The following are the features and requirements associated with Persistent Flush packets:

- 5     • A Persistent Flush request packet may be supported by any component that supports addressable persistent resources.
- 10    • Persistent Flush request packet formats:
  - o A P2P-Core Persistent Flush packet shall use the *P2P-Core Persistent Flush Packet Format*.
  - o A Core 64 Persistent Flush packet shall use the *Core 64 Persistent Flush Packet Format*.
- 15    • Persistent Flush responses:
  - o The Responder shall return a *P2P-Core Standalone Acknowledgment* for each P2P-Core Persistent Flush request packet.
  - o The Responder shall return a *Core 64 Standalone Acknowledgment* for each Core 64 Persistent Flush request packet.
- 20    • A Persistent Flush request packet shall be applicable only to successfully validated and executed request packets that modify addressable persistent resources up to the receipt of the Persistent Flush request packet.
  - o The Responder shall generate a *Standalone Acknowledgment* only after all updates to addressable persistent resources have reached a processing state such that they are guaranteed to successfully update the persistent media.
    - If a Responder determines that the time to validate the Persistent Flush request packet and successfully update the persistent media will exceed the worst-case packet processing time (*Core Structure* WRLAT for a P2P-Core Persistent Flush, and *Core Structure* Responder Deadline for a Core 64 Persistent Flush), then the Responder shall generate a *Standalone Acknowledgment* (Reason = ND ACK) to acknowledge that the request packet was received and validated but not executed. Once the Responder has successfully updated the addressable persistent resources, then the Responder shall generate a second *Standalone Acknowledgment* indicating success.
  - o If a previously received request packet that modifies addressable persistent resources failed validation or execution, then the Responder shall set *Core C-Status* Cannot Execute Persistent Flush = 1b. If *Core C-Status* Cannot Execute Persistent Flush == 1b, then upon receipt of a Persistent Flush request packet, the Responder shall not execute the Persistent Flush request packet, and shall return a *Standalone Acknowledgment* (Reason = Persistent Flush Update Failure).
- 25    • If a Requester retransmits a Persistent Flush request packet, then the Responder shall validate and re-execute the request packet as though it were a new request packet, i.e., it includes previously received request packets that modify data at the time the retransmitted request packet is received.
- 30    • A Requester shall generate a separate P2P-Core Persistent Flush request packet for each Responder within a daisy-chain topology that requires a Persistent Flush to be executed.
- 35
- 40



## 5 6.10. Control Space In-Band Operations

### 6.10.1. P2P-Core and P2P-Coherency In-Band Operations

The P2P-Core and P2P-Coherency OpClasses are optimized for point-to-point topologies, hence, support a subset of the operations supported by the Control OpClass, which supports all topologies.

#### 6.10.1.1. **CTL-Read and CTL-Write Packets**

10 Components that support *In-band Management* shall support CTL-Read and CTL-Write request packets to access Control Space structures.

The following are the features and requirements associated with CTL-Read and CTL-Write packets:

- Packet formats:

- P2P-Core CTL-Read request packet shall use the *P2P-Core CTL-Read Request Packet Format*.
- P2P-Core CTL-Write request packet shall use the *P2P-Core CTL-Write Request Packet Format*.
- P2P-Coherency CTL-Read request packet shall use the *P2P-Coherency CTL-Read Request Packet Format*.
- P2P-Coherency CTL-Write request packet shall use the *P2P-Coherency CTL-Write Request Packet Format*.
- CTL-Read and CTL-Write request packets shall use byte-level addressing. Unused upper address bits shall be set to zero.
- The payload size in CTL-Write request packets that target tables and structures that are located through a Control Space structure shall be an integer 4-byte multiple.
- A P2P-Core CTL-Write request packet shall be acknowledged by a *Standalone Acknowledgment*.
- A P2P-Coherency CTL-Write request packet shall be acknowledged by a *Standalone Acknowledgment*.
- A P2P-Core CTL-Read request packet shall be acknowledged by a P2P-Core Read Response packet or by a *Standalone Acknowledgment* in case of error.
- A P2P-Coherency CTL-Read request packet shall be acknowledged by a P2P-Coherency Read Response packet or by a *Standalone Acknowledgment* in case of error.
- CTL-Read and CTL-Write packets shall use VC0 for request packets and response packets.
- Unless explicitly specified otherwise in this specification, a CTL-Read shall be as specified for non-CTL *Read and Read Response*, and a CTL-Write request shall be as specified for a non-CTL *Write*.

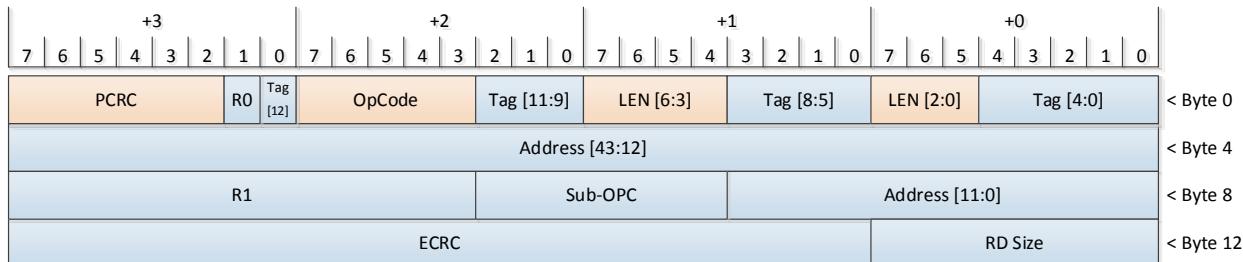


Figure 6-38: P2P-Core CTL-Read Request Packet Format

Table 6-29: P2P-Core CTL-Read Request Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>Read Size</b>	RD Size	8	Number of bytes to read at the addressed location. RD Size == 0x0 shall indicate 256 bytes is to be read.
<b>RO</b>	-	1	Reserved
<b>R1</b>	-	13	Reserved

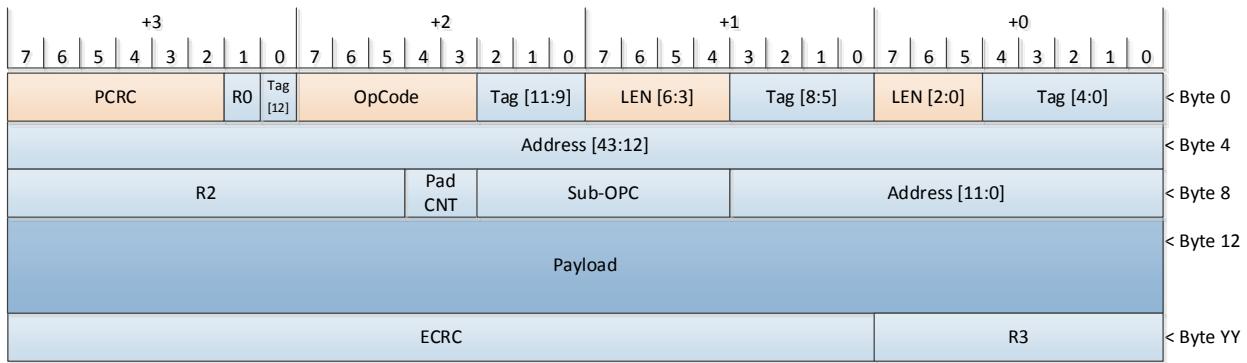


Figure 6-39: P2P-Core CTL-Write Request Packet Format

Table 6-30: P2P-Core CTL-Write Request Packet Fields

Field Name	Size (bits)	Description
<b>R0</b>	1	Reserved
<b>R1</b>	11	Reserved
<b>R2</b>	8	Reserved

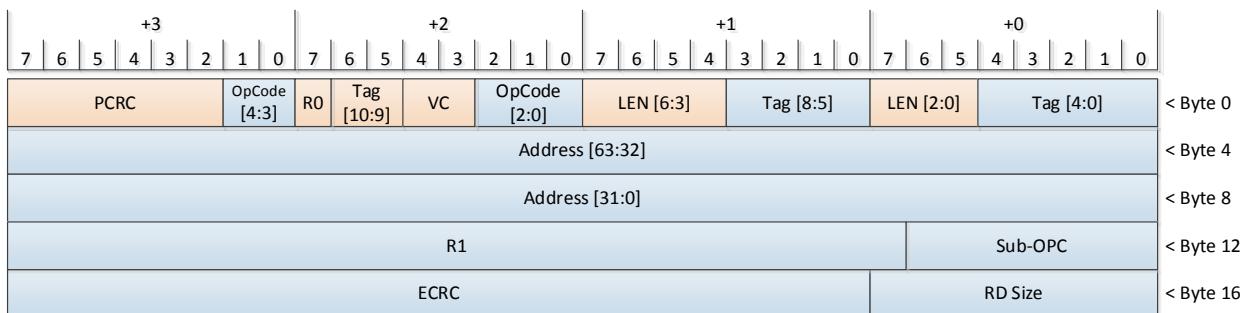


Figure 6-40: P2P-Coherency CTL-Read Request Packet Format

Table 6-31: P2P-Coherency CTL-Read Request Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>Read Size</b>	RD Size	8	Number of bytes to read at the addressed location. RD Size == 0x0 shall indicate 256 bytes is to be read.
<b>R0</b>	-	1	Reserved
<b>R1</b>	-	25	Reserved

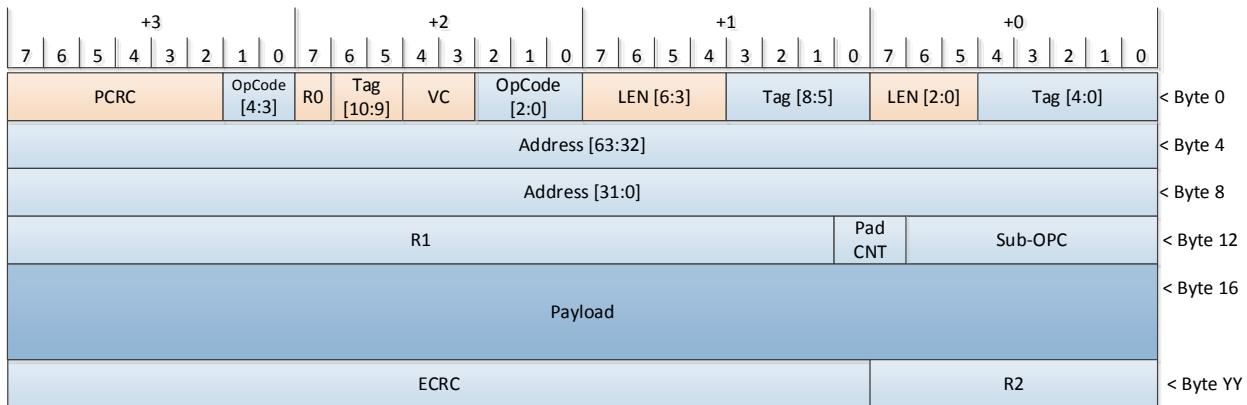


Figure 6-41: P2P-Coherency CTL-Write Request Packet Format

Table 6-32: P2P-Coherency CTL-Write Request Packet Fields

Field Name	Size (bits)	Description
R0	1	Reserved
R1	23	Reserved
R2	8	Reserved

### 6.10.1.2. CTL-UE Packet

An Unsolicited Event (UE) packet is used to communicate events to management, e.g., firmware. Events are generated for a variety of reasons including specific packet processing error events, component failure detection, new component discovery, etc. Management is responsible for gathering information from the component's Control Space structures to determine root cause and to take corrective action.

The following are the features and requirements associated with CTL-UE packets:

- Components that support *In-band Management* and either the P2P-Core or the P2P-Coherency OpClass shall support CTL-UE.
- P2P-Core CTL-UE packet shall use the *P2P-Core CTL-UE Request Packet Format*.
- P2P-Coherency CTL-UE packet shall use the *P2P-Coherency CTL-UE Request Packet Format*.
- P2P-Core and P2P-Coherency CTL-UE packets shall use VC0.
- Unless explicitly stated otherwise by this specification, a P2P-Core or P2P-Coherency CTL-UE packet shall comply with the features and requirements common to CTL-UE and Control UE packets (see *Unsolicited Event (UE) Packet*).
- P2P-Core CTL-UE packets shall be transmitted only by Responders, and shall be forwarded by all intermediate Responders within a daisy-chain towards the Requester using the interface identified by the *Interface Structure Peer Daisy Interface ID*.

Table 6-33: Common P2P-Core and PCI-Coherency CTL-UE Request Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
Event	-	8	Unsolicited Event type that triggered the Unsolicited Event packet. See <i>Unsolicited Event Descriptions</i> .
Event ID	-	16	Monotonically increasing unsigned integer (modulo $2^{16}$ ) used to uniquely identify each new Unsolicited Event.
Event-specific Field	-	32	An Unsolicited Event packet can communicate additional information to assist management. If an Unsolicited Event packet does not communicate additional information, then this field shall be Reserved. All unused Event-specific Field bits shall be Reserved.
Interface ID	-	8	If the event is associated with a specific component interface, then this field shall contain the interface identifier. Components that support CTL-UE are capable of reporting events that occur on Interface IDs [0-255].
Interface ID Valid	IV	1	Indicates if the Interface ID field contains an Interface ID or is Reserved. 0b—Interface ID field is Reserved 1b—Interface ID field contains an interface identifier

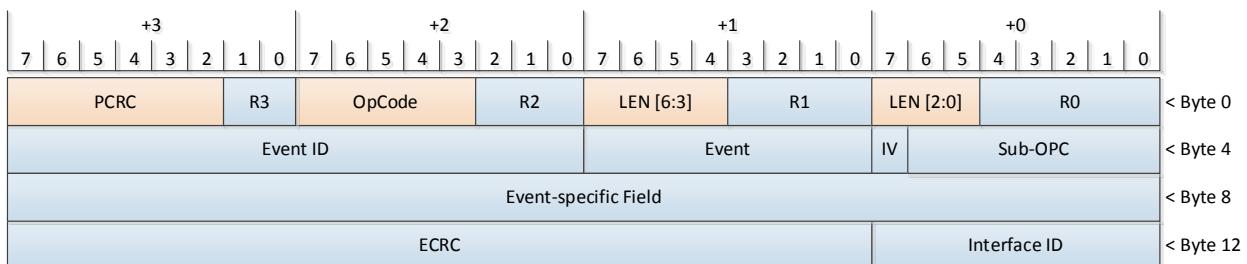


Figure 6-42: P2P-Core CTL-UE Request Packet Format

Table 6-34: P2P-Core CTL-UE Request Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
R0	-	5	Reserved
R1	-	4	Reserved
R2	-	3	Reserved
R3	-	2	Reserved

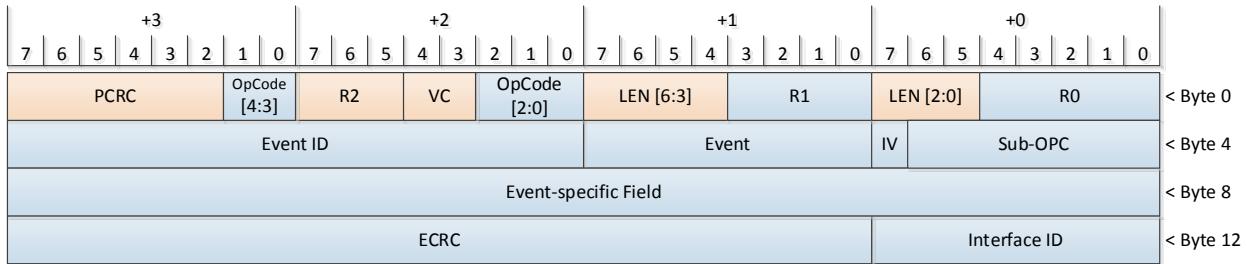


Figure 6-43: P2P-Coherency CTL-UE Request Packet Format

Table 6-35: P2P-Core CTL-UE Request Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
R0	-	5	Reserved
R1	-	4	Reserved
R2	-	3	Reserved

## 6.10.2. Control OpClass In-Band Operations

5 The following are the features and requirements associated with Control OpClass packets:

- Control OpClass packets shall access only the component's Control Space.
- Control OpClass packets shall use the OpCodes specified in *Control OpClass OpCodes*.
- Control Read Response packets shall be as specified in *Read and Read Response* and use the respective Core 64 packet format.
- Control Acknowledgments shall be as specified in *Standalone Acknowledgment* and use the Core 64 packet format.
- Control Interrupts shall be as specified in *Interrupts* and use the Core 64 packet format.
- Depending upon the underlying implementation and the request or response type, some Control OpClass request packets can take longer to execute. The component indicates the maximum control request processing time (*Core Structure CRPTO*).
- If a component supports the Control OpClass, then it shall support the *Component Error and Signal Event Structure*. This structure contains a set of fields that identify the component egress interface(s) and the corresponding VC for that interface that can be used to transmit Control OpClass packets.
  - o A Control OpClass packet may be received on any component interface.

Table 6-36: Common Control Read, Control Write, and Control Write MSG Request Packet Fields

Field Name	Size (bits)	Description
DR-Interface	12	In a Control Read or a Control Write request packet, this is the identifier of the egress interface to which the packet shall be relayed after the packet has been validated by the receiving packet-relay component.

Field Name	Size (bits)	Description
		<p>In a Control Write MSG, this is the identifier of the egress interface to which the packet shall be relayed after the packet has been validated by the receiving packet-relay component, or this is the identifier of the ingress interface where the Control Write MSG was received by a switch and is being communicated to management for subsequent message exchange.</p> <p>While in the C-CFG state, the destination component shall ignore the DR-Interface field.</p> <p>See <i>Directed Control Space Packet Relay</i>.</p>
<b>Payload</b>	-	<p>Payload shall be placed such that data byte 0 is placed at payload byte 0, data byte 1 at payload byte 1, etc. If the data size is not an integer 4-byte multiple, then the payload field shall be padded to ensure an integer 4-byte size.</p>
<b>DR</b>	1	<p>Directed Relay—Indicates whether <i>Directed Control Space Packet Relay</i> is to be performed by a switch addressed by the packet's DCID.</p> <p>For components in the C-CFG state, the DR setting is used to select the VC and egress interface for response packets.</p> <p>0b—Do not perform <i>Directed Control Space Packet Relay</i>, and select response packet's VC and egress interface using the <i>Component Destination Table Structure</i>.</p> <p>1b—Perform <i>Directed Control Space Packet Relay</i>, and select response packet's VC and egress interface without using the <i>Component Destination Table Structure</i>.</p>
<b>MGR-UUID</b>	128	<p>A UUID used to uniquely identify the manager that issued the request packet.</p> <p>If the Responder's <i>Core Structure</i> MGR-UUID is non-zero, then the Responder shall validate that the request packet's MGR-UUID value matches, else the request packet shall be handled as a MP.</p>

### 6.10.2.1. Control Read and Control Write Packets

The following are the features and requirements associated with Control Read and Control Write packets:

- Packet formats:
  - Control Read request packet shall use the *Control Read Request Packet Format*.
  - Control Read Response packet shall use the *Core 64 Read Request Packet Format*.
  - Control Write request packet shall use the *Control Write Request Packet Format*.
- Control Read and Control Write request packets shall use byte-level addressing. Unused upper address bits shall be set to zero.

- The payload size in Control Write request packets that target tables and structures that are located through a Control Space structure shall be an integer 4-byte multiple. For example, updates to packet relay tables, destination tables, etc. are performed using 4 byte, 8 byte, etc. sized payloads. Due to their potential size, such tables could be implemented using some form of RAM, and integer 4-byte multiple updates eliminates the need for the hardware implementation to perform read-modify-write operations.
  - A Control Write request packet shall be acknowledged by a Control *Standalone Acknowledgment*.
  - A Control Read request packet shall be acknowledged by a Core 64 Read Response packet or by a Control *Standalone Acknowledgment* in case of error.
  - The Directed Interface field (DR-Interface) is used to indicate the switch egress interface to be targeted by this packet. This is only used during initial configuration of components attached to a switch. See *Directed Control Space Packet Relay*.
  - If the component has not been configured or initialized, then
    - If the component is in a point-to-point topology, then Control Read, Control Read Response, and Control Write packets shall use VC0 for request packets and response packets.
    - If the component is in a switch-based topology and the directly-attached switch acts as a proxy to exchange packets to or from the component, then request packets destined for the component shall target the switch and the switch shall relay response packets as though they originated from the switch. This may entail modifying the response packet's SCID, SSID, and VC fields to those used by the switch when generating a response packet.
  - Unless explicitly stated otherwise by this specification, a Control Read and a Control Read Response shall be as specified for non-Control *Read and Read Response*, and a Control Write shall be as specified for a non-Control *Write*.
  - Control Read and Control Write request packets shall not be used to perform *Indirect Manager Communication*.

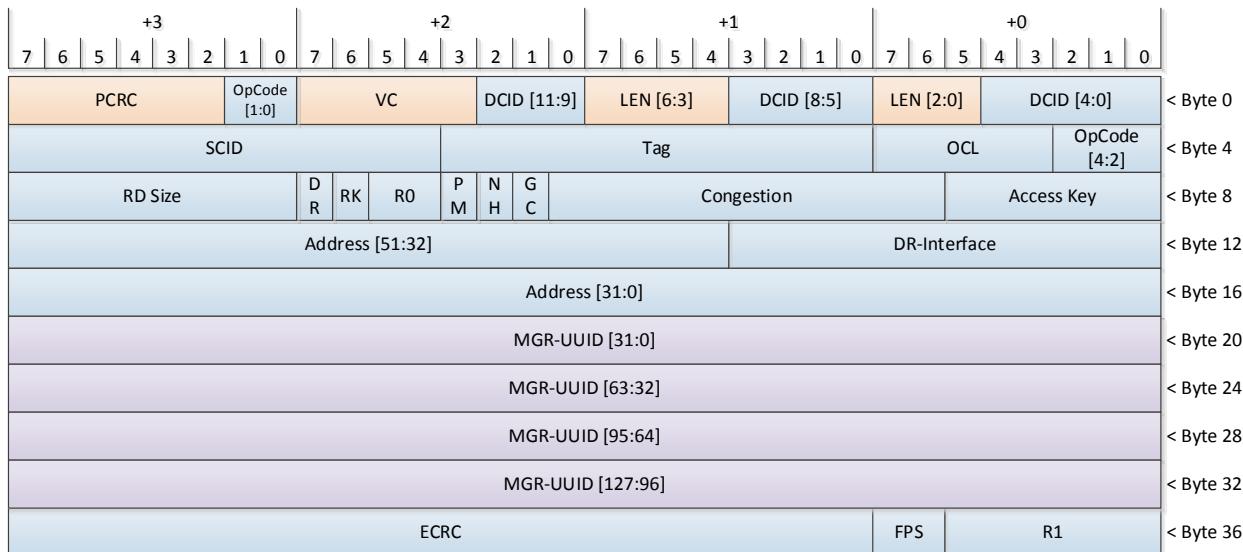


Figure 6-44: Control Read Request Packet Format

Table 6-37: Control Read Request Packet Fields

Field Name	Field Abbreviation	Size (bits)	Bit Location / Field Value	Description
Read Size	RD Size	8	-	Number of bytes to read at the addressed location. RD Size == 0x0 shall indicate 256 bytes is to be read.
R0	-	2	-	Reserved
R1	-	6	-	Reserved

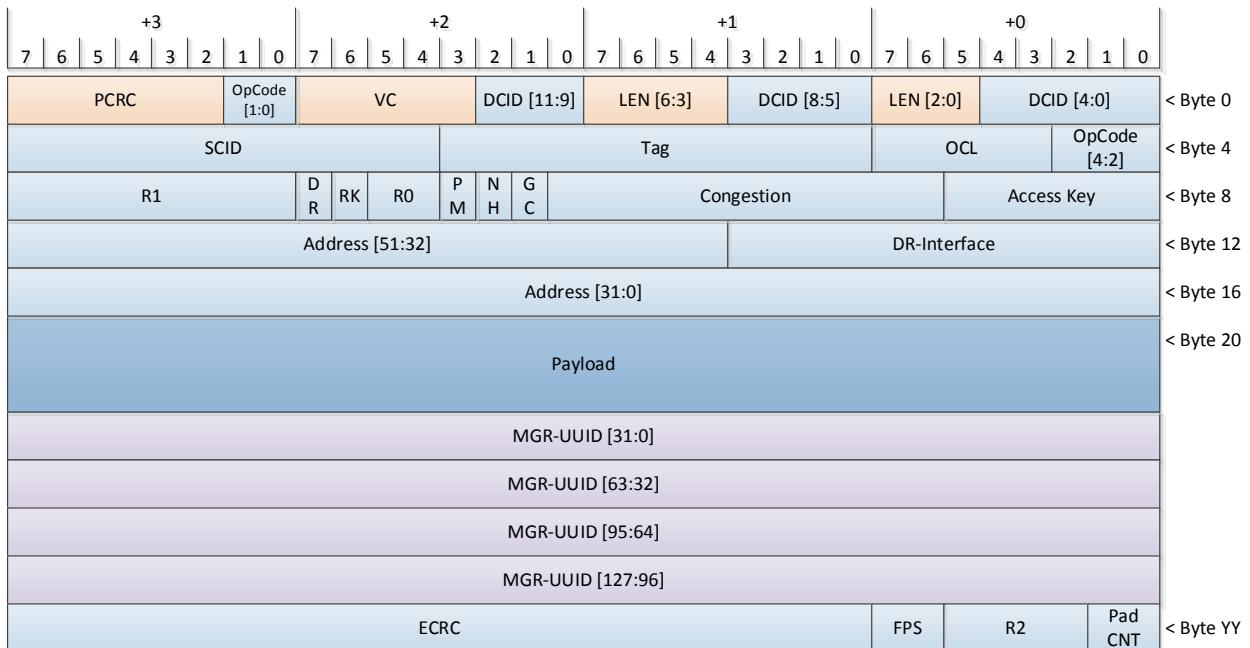


Figure 6-45: Control Write Request Packet Format

Table 6-38: Control Write Request Packet Fields

Field Name	Field Abbreviation	Size (bits)	Bit Location / Field Value	Description
R0	-	2	-	Reserved
R1	-	8	-	Reserved
R2	-	4	-	Reserved

### 6.10.3. Unsolicited Event (UE) Packet

An Unsolicited Event (UE) packet is a request packet that is used to communicate events to a designated manager, e.g., firmware, fabric manager, etc. Events are generated for a variety of reasons including specific packet processing error events, component failure detection, new component discovery, etc. Management is responsible for gathering information from the component's Control Space structures to determine root cause and to take corrective action.

The following are the features and requirements common to *CTL-UE Packet* and Control UE packets (within this sub-section, UE packets refers to either a CTL-UE or a Control UE packet):

- If a component supports *In-band Management* and is configured as such, the component shall support UE packets. If a component does not support *In-band Management* or is not configured to use *In-band Management*, then the component shall not generate UE packets.
- UE packets may be transmitted at any time that the component is in a state that supports CTL-UE or Control OpClass packet transmission.
  - Control OpClass UE packets may be transmitted on any egress interface identified by a valid *Component Error and Signal Event Structure* MGMT Interface ID using the corresponding MGMT VC.
- UE packets shall not be acknowledged at the protocol level.
- Each new UE packet generated by a component shall be assigned an Event Identifier. Upon component initialization, the component shall set the Event Identifier field in the first UE packet to 1. Event Identifiers shall be monotonically increasing modulo  $2^{16}$ .
- UE packets shall be periodically transmitted until management clears the event (see *Example Unsolicited Event Packet and UERT Operation*). The steps are as follows:
  - A component shall maintain a single Unsolicited Event Retransmission Timer (*Core Structure* UERT).
    - A component shall start the UERT whenever a UE packet is scheduled.
    - A component shall stop the UERT whenever the *Core C-Control* Halt UERT field == 1b.
      - Upon Halt UERT == 1b, the component shall stop retransmitting the present UE packet and shall clear the associated tracking logic.
      - If another event is pending, then the component shall schedule a new UE packet and start the UERT.
    - If the UERT timeout expires and Halt UERT  $\neq$  1b, then the component shall retransmit the outstanding UE packet and restart the UERT.
      - If multiple paths exist to a given manager, then the component should retransmit the UE packet on an alternative path.
      - A component shall retransmit a UE packet no fewer than 4 times and no more than 32 times before targeting the next configured manager.
        - If after 32 attempts per configured manager without a *Core C-Control* Halt UERT field being set to 1b, then the component shall not retransmit the UE packet, and shall clear event tracking logic.

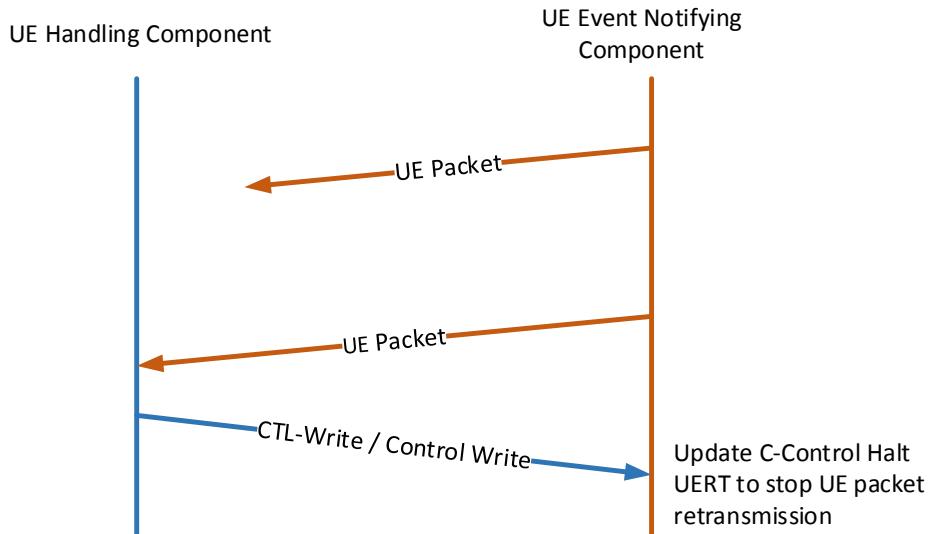


Figure 6-46: Example Unsolicited Event Packet and UERT Operation

- Upon receipt of an Unsolicited Event packet, the manager's component shall take the following steps:
  - If the UE packet fails packet validation, then it shall be silently discarded.
  - If a new or duplicate UE packet is successfully validated, then the manager shall halt UE packet generation (*Core C-Control Halt UERT*) at the source component. Due to variable event management processing time, management should halt UE packet generation without waiting for event management processing to be completed.
    - A duplicate is a UE packet from the same source component with an Event ID that was previously processed by the manager. The manager is responsible for delineating new from duplicate events.
- If a manager is unable to execute and handle a UE packet (e.g., due to temporary resource exhaustion), then it may silently discard a UE Packet.
- A component shall have at most one outstanding UE packet.
  - If a component simultaneously detects multiple events that require notification, then the component should select the event with the highest precedence to be handled first. If multiple events of equal precedence are detected, then event selection is at the component's discretion.
  - New events shall not preempt an outstanding UE packet. New events are communicated in highest precedence order after the outstanding UE packet is cleared.
    - A component should provision sufficient resources to avoid spurious event loss.
    - A component should have sufficient resources or internal tracking logic to support at least one event for each event precedence level, plus at least one event per supported component interface.
- If supported, then upon receipt of an Unsolicited Event packet, the component shall take the actions configured in the *Component Event Structure*.
- A component may generate a UE packet based on methods outside of this specification's scope or, if the component supports the *Component Error and Signal Event Structure*, then based on the structure's C-Error Signal Target, C-Event Signal Target, or I-Event Signal Target fields.

The following are the features and requirements associated with UE packets:

- If the component supports UE packets, then the component shall support the *Component Error and Signal Event Structure*.
- UE packets may be transmitted at any time that the component is in a state that supports Control OpClass packet transmission.
- P2P-Core CTL-UE packet shall use the *P2P-Core CTL-UE Request Packet Format*.
- P2P-Coherency CTL-UE packet shall use the *P2P-Coherency CTL-UE Request Packet Format*.
- Control UE packets shall use the *Control Unsolicited Event Packet Format*.
- If the component supports Control UE packets, then UE packets may be configured to target different management components based on the configured resource manager.
  - The following managers may be configured:
    - A Primary Manager identified by *Core Structure* PMCID.
    - A Primary Fabric Manager identified by *Core Structure* [PFMCID, PFMSID].
    - A Secondary Fabric Manager identified by *Core Structure* [SFMCID, SFMSID].
    - An Error Manager identified by *Component Error and Signal Event Structure* [Error MGR CID, Error MGR SID].
    - An Event Manager identified by *Component Error and Signal Event Structure* [Event MGR CID, Event MGR SID].
    - A Power Manager identified by *Core Structure* [PWR MGR CID, PWR MGR SID].
    - A Mechanical Manager identified by *Component Mechanical Structure* [Mechanical MGR CID, Mechanical MGR SID].
    - A Media Manager identified by *Component Media Structure* [PWR MGR CID, PWR MGR SID].
  - For each manager, there is a corresponding bit mask that indicates which MGMT VC and MGMT Interface ID may be used to transmit a UE packet to that manager.

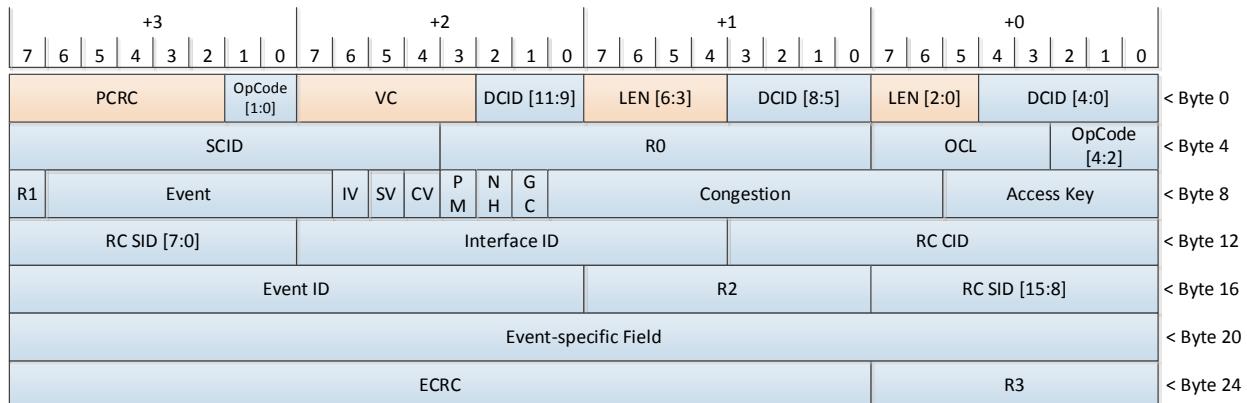


Figure 6-47: Control Unsolicited Event Packet Format

Table 6-39: Control Unsolicited Event Packet-specific Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
Event ID	-	16	Monotonically increasing unsigned integer (modulo $2^{16}$ ) used to uniquely identify each new Unsolicited Event. The [SCID   Global SCID, Event ID] uniquely identifies the event.
Root Cause CID	RC CID	12	For events triggered by the receipt of a non-Unsolicited Event packet, the RC CID contains the SCID of the received packet.

Field Name	Field Abbreviation	Size (bits)	Description
RC SID	-	16	<p>This enables management to identify the component and take appropriate action.</p> <p>If communications with a given destination are no longer possible, then the RC CID shall contain the destination's CID. This enables management to validate the path between the two components and take appropriate action.</p>
			<p>For events triggered by the receipt of a non-Unsolicited Event packet with GC == 1b, the RC SID contains the packet's SSID. This enables management to identify the offending component and take appropriate action.</p> <p>If communications with a given destination are no longer possible, then the RC SID shall contain the destination's SID. This enables management to validate the path between the two components and take appropriate action.</p>
			<p>If the received packet's GC == 0b, then the RC SID shall be Reserved.</p>
Interface ID	-	12	If the event is associated with a specific component interface, then this field shall contain the interface identifier.
Event	-	8	Unsolicited Event type that triggered the Unsolicited Event packet. See <i>Unsolicited Event Descriptions</i> .
Event-specific Field	-	32	<p>An Unsolicited Event packet can communicate additional information to assist management.</p> <p>If an Unsolicited Event packet does not communicate additional information, then this field shall be Reserved.</p> <p>All unused Event-specific Field bits shall be Reserved.</p>
CID Valid	CV	1	<p>Indicates if the RC CID field contains a CID or is Reserved.</p> <p>0b—RC CID field is Reserved</p> <p>1b—RC CID field contains a CID</p>
SID Valid	SV	1	<p>Indicates if the RC SID field contains a SID or is Reserved.</p> <p>0b—RC SID field is Reserved</p> <p>1b—RC SID field contains a SID</p>
Interface ID Valid	IV	1	<p>Indicates if the Interface ID field contains an Interface ID or is Reserved.</p> <p>0b—Interface ID field is Reserved</p> <p>1b—Interface ID field contains an interface identifier</p>
R0	-	12	Reserved
R1	-	1	Reserved

Field Name	Field Abbreviation	Size (bits)	Description
R2	-	8	Reserved
R3	-	8	Reserved

### 6.10.3.1. Unsolicited Event (UE) Encodings

Unsolicited events are classified as follows:

1. Critical—these events include failure conditions, containment events, etc.
2. Serious—these events include unrecoverable errors, access permission violations, excessive retry events, malicious behavior detected, etc.
3. Resource—these events include new resource (e.g., a new component) discovery, service events, low-power entry, low-power exit, etc.
4. Media—these events include changes in media state, endurance, spare exhaustion, media service requirements, etc.
5. Recoverable—these events include recoverable errors, etc.
6. Vendor-defined—these events include Vendor-defined C-Event and I-Events

The *Unsolicited Event Descriptions* table is used to encode the Event field in Control UE packets and P2P-Core and P2P-Coherency *CTL-UE Packets*. CTL-UE packets do not contain a RC CID or RC SID fields, hence, actions that refer to copying to these packet fields shall be ignored.

Table 6-40: Unsolicited Event Descriptions

Event Encoding	Precedence	Entity	Description
0x0	Recoverable	Component	Recoverable Protocol Error Event Detected Copy packet's SCID to the RC CID field Copy packet's SSID to the RC SID field (if applicable) Copy the protocol error Reason (see <i>Reason Field Encodings</i> .) to Event-specific Field[5:0].
			Unrecoverable Protocol Error Event Detected. Copy packet's SCID to the RC CID field Copy packet's SSID to the RC SID field (if applicable) Copy the protocol error Reason (see <i>Reason Field Encodings</i> .) to Event-specific Field[5:0]. If (Reason == MP) and the root cause is an invalid HMAC or ART, then report this event as "Possible Malicious Packet Detected" instead.
0x2	Serious	Component	Possible Malicious Packet Detected See <i>Security</i> .

Event Encoding	Precedence	Entity	Description
0x3			Copy packet's SCID to the RC CID field Copy packet's SSID to the RC SID field (if applicable)
	Serious	Interface	Interface Error Detected Copy impacted interface's identifier to the Interface ID field Copy the value K to Error-specific Field[3:0] where K is the bit location (Bit <sub>K</sub> ) in the <i>I-Error Detect</i> field.
0x4	Critical	Component	<i>Component Containment</i> Triggered If the error was triggered by a received packet, then: Copy packet's SCID to the RC CID field Copy packet's SSID to the RC SID field (if applicable)
	Critical	Interface	<i>Interface Containment</i> Triggered Copy impacted interface's identifier to the Interface ID field
0x5	Critical	Interface	Active Link Failure Detected Copy impacted interface's identifier to the Interface ID field
	Critical	Interface	<i>Full Interface Reset</i> Event Copy impacted interface's identifier to the Interface ID field
0x7	Serious	Interface	<i>Warm Interface Reset</i> Event Copy impacted interface's identifier to the Interface ID field
	Serious	Interface	<i>Link RFC</i> packet received Copy impacted receiving interface's identifier to the Interface ID field
0x9	Resource	Component	New Peer Component Detected ( <i>Link RFC</i> packet received) Copy impacted receiving interface's identifier to the Interface ID field
	Serious	Component	Unable to Communicate with an Authorized Destination This may be detected by the component exceeding its maximum request packet retransmission counter or all interfaces used to communicate with the destination transitioning to I-Down. Copy destination's CID to the RC CID field

Event Encoding	Precedence	Entity	Description
0xB			Copy destination's SID to the RC SID field (if applicable)
	Serious	Component	<p>Excessive RNR NAK responses.</p> <p>Bits [4:0] contain the OpClass of the request receiving RNR NAK responses.</p> <p>Bits [9:5] contain the OpCode of the request receiving RNR NAK responses</p> <p>Copy Responder's CID to the RC CID field</p> <p>Copy Responder's SID to the RC SID field (if applicable)</p>
0xC			
	Serious	Interface	<p>Buffer Overflow Detected</p> <p>Copy packet's SCID to the RC CID field</p> <p>Copy packet's SSID to the RC SID field (if applicable)</p> <p>Copy impacted interface's identifier to the Interface ID field</p>
0xD			
	Critical	Component	<p>Fatal Media Error Containment Triggered Event</p> <p>If a primary or secondary media log entry was generated, set the Event-specific field as follows:</p> <p>Bit [0] = 1b if a primary log entry was generated, else Bit [0] = 0b.</p> <p>Bit [1] = 1b if a secondary entry was generated, else Bit [1] = 0b.</p> <p>Bits [13:2] = log entry index</p>
0xE			
	Media	Component	<p><i>Component Media Structure</i> Plog Event. Bits [3:0] of the Event-specific field indicate the log severity:</p> <p>0x0—Reserved</p> <p>0x1—Critical Event</p> <p>0x2—Caution event</p> <p>0x3—Non-critical Event</p> <p>0x4-0xF—Reserved</p> <p>Bits [19:4] of the Event-specific field contains the index of the corresponding log entry</p>
0xF			
	Media	Component	<p><i>Component Media Structure</i> Slog Event. Set the Event-specific field as follows:</p> <p>Bits [3:0] indicate the log severity:</p> <p>0x0—Reserved</p>

Event Encoding	Precedence	Entity	Description
			0x1—Critical Event 0x2—Caution event 0x3—Non-critical Event 0x4-0xF—Reserved  Bits [19:4] indicate the index of the corresponding log entry
0x10	Serious	Component	Invalid Component Image Detected
0x11	Critical	Component	Component Thermal Shutdown Detected
0x12	Resource	Component	Attention Button Pressed Event Detected—See <i>Component Mechanical Structure</i>  Copy impacted interface's identifier to the Interface ID field
0x13	Serious	Component	Power Fault Event Detected—See <i>Component Mechanical Structure</i>  Copy impacted interface's identifier to the Interface ID field
0x14	Resource	Component	MRL Sensor Change Event Detected—See <i>Component Mechanical Structure</i>  Copy impacted interface's identifier to the Interface ID field
0x15	Resource	Component	Dynamic Insertion Removal Event Detected—See <i>Component Mechanical Structure</i>  Copy impacted interface's identifier to the Interface ID field
0x16	Resource	Component	Controller CMD Completion Event Detected—See <i>Component Mechanical Structure</i>  Copy impacted interface's identifier to the Interface ID field
0x17	Resource	Component	Component Low-power Entry Event
0x18	Resource	Component	Component Low-power Exit Event
0x19	Resource	Component	Component Deep Low-power Entry Event
0x1A	Resource	Component	Component Deep Low-power Exit Event
0x1B	Resource	Component	Peer Component Deep Low-power Entry Detected event

Event Encoding	Precedence	Entity	Description
0x1C			Copy impacted interface's identifier to the Interface ID field
	Resource	Component	Emergency Power Reduction Triggered. See <i>Component Power Management</i>
0x1D	Resource	Component	Component Power-off Transition Completed Event
	Resource	Component	Component Power Restoration Event
0x1F	Serious	Interface	Interface Performance Degradation Event
			Copy impacted interface's identifier to the Interface ID field
0x20	Serious	Interface	Interface Service Performance Degradation Event
			Copy impacted interface's identifier to the Interface ID field
0x21	Serious	Media	Media Maintenance Required Event (see <i>Component Media Structure</i> ). Set the Event-specific field as follows:  Bit [0] = 0b to indicate primary media event Bit [0] = 1b to indicate secondary media event Bits [16:1] = Number of $\mu$ s until serious performance degradation at current load.
0x22	Serious	Media	Media Maintenance Override Event (see <i>Component Media Structure</i> ). Set the Event-specific field as follows:  Bit [0] = 0b to indicate primary media event Bit [0] = 1b to indicate secondary media even
0x23	Recoverable	Interface	Exceeded Transient Error Threshold Event  Copy impacted interface's identifier to the Interface ID field
0x24	Vendor-defined	Component	Vendor-Defined C-Event  Copy vendor-defined information to the Event-specific Field[31:0].
0x25	Vendor-defined	Interface	Vendor-Defined I-Event  Copy vendor-defined information to the Event-specific Field[31:0].

Event Encoding	Precedence	Entity	Description
0x26	Serious	Component	<p>Non-Fatal Internal Component Error Event—see <i>Component Error and Signal Event Structure</i>.</p> <p>Set the Event-specific field as follows:</p> <p>Bit [0] = 1b if a log entry was created, else set to 0b.</p> <p>Bits [12:1] = log entry index</p>
0x27	Critical	Component	<p>Fatal Internal Component Error Event—see <i>Component Error and Signal Event Structure</i>.</p> <p>Set the Event-specific field as follows:</p> <p>Bit [0] = 1b if a log entry was created, else set to 0b.</p> <p>Bits [12:1] = log entry index</p>
0x28	Critical	Component	<p>Component Thermal Performance Throttle Detected—component reduced performance due to thermal operating conditions (Exceed Upper Thermal Limit or Caution Thermal Limit)</p> <p>Set bits [6:0] of the Event-specific field to indicate the current maximum performance as a percentage of the component's nominal maximum performance. Valid values are [0-100].</p>
0x29	Critical	Component	Component Thermal Throttle Restoration Detected—component restored performance
0x30	Media	Component	Primary Media Maintenance Required Event Detect
0x31	Media	Component	Primary Media Maintenance Override Event Detect
0x32	Media	Component	Secondary Media Maintenance Required Event Detect
0x33	Media	Component	Secondary Media Maintenance Override Event Detect
0x34	Resource	Component	<p>Mechanical Dynamic Module Insertion Event</p> <p>Copy impacted interface's identifier to the Interface ID field</p>
0x35	Resource	Component	<p>Mechanical Dynamic Module Removal Event</p> <p>Copy impacted interface's identifier to the Interface ID field</p>
0x36	Resource	Component	<p>Mechanical Attention Button Pressed</p> <p>Copy impacted interface's identifier to the Interface ID field</p>

Event Encoding	Precedence	Entity	Description
0x37	Resource	Component	Mechanical Attention Indicator on Continuously Event Copy impacted interface's identifier to the Interface ID field
0x38	Resource	Component	Mechanical Module Power Up Event Copy impacted interface's identifier to the Interface ID field
0x34	Resource	Component	Mechanical Module Power Off Event Copy impacted interface's identifier to the Interface ID field
0x39	Resource	Component	Mechanical MRL Open Event Copy impacted interface's identifier to the Interface ID field
0x3A	Resource	Component	Mechanical MRL Close Event Copy impacted interface's identifier to the Interface ID field
0x3B	Resource	Component	Mechanical Module Indicator Activated Event Copy impacted interface's identifier to the Interface ID field
0x3C-0xEF	-	-	Reserved
0xF0-0xFF	-	Vendor-defined	Vendor-defined Events—Precedence is at the component's discretion.

#### 6.10.4. No-Op

A No-Op request is used to verify a component or a path to a component is operational.

The following are the features and requirements associated with No-Op packets:

- Any component type may support a No-Op packet.
- A No-Op packet shall use the *No-Op Packet Format*.
- A No-Op packet shall have no impact to component internal resources (e.g., memory).
- A No-Op packet shall be acknowledged by a Control *Standalone Acknowledgment*.
- Control No-Op request packets shall be unicast packets.

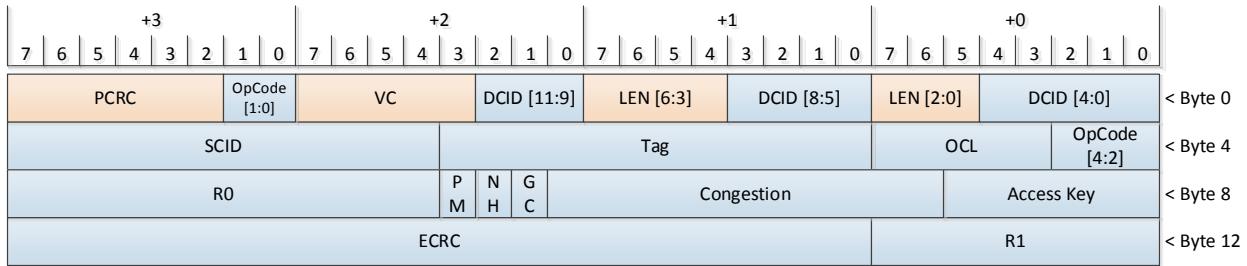


Figure 6-48: No-Op Packet Format

Table 6-41: No-Op Packet Fields

Field Name	Size (bits)	Description
<b>R0</b>	12	Reserved
<b>R1</b>	8	Reserved

### 6.10.5. C-State Power Control

A C-State Power Control request packet is used to mitigate solution disruptions due to component C-State transitions and component power level consumptions. C-State Power Control enables:

- Peer component notification that a component is auto-transitioning to a new C-State.
  - To mitigate solution disruption, prior to the auto-transitioning to a new C-State, a Responder may notify each peer Requester and / or management component. For example, based on preconfigured parameters, after a Responder has been idle for a period of time, it automatically transitions to a lower C-State to conserve power.
- Request a peer component to transition to a new C-State or power level.
  - For example, a Requester requests a Responder to transition from the C-LP state to the C-Up state to restore normal operations.
- Peer component notification that a component is modifying its power consumption.
  - For example, a component can automatically adjust its power consumption to match its workload. A component uses this request to inform peer components or management of the change to ensure the change can be accommodated or does not negatively impact upcoming workload behavior.
- Request a peer component to adjust its power consumption.
  - For example, request a component to dynamically reduce power as a function of its configured maximum power, or to inform a component it may consume additional power up to its configured maximum.

The following are the features and requirements associated with C-State Power Control packets:

- Any component type may support the C-State Power Control packet.
- The C-State Power Control packet shall use the *C-State Power Control Packet Format*.
- A C-State Power Control request packet shall be acknowledged by a Control OpClass *Standalone Acknowledgment*.
- If a component transmits a C-State Power Control packet to inform a peer that it is going to perform a C-State transition, then the peer may require the transition to be aborted. If the

transition is to be aborted, then the Request-specific field within the *Standalone Acknowledgment* (Reason = Abort Transition) shall contain the C-State to which the Requester shall transition:

- 5
  - 0x0—C-Up
  - 0x1—C-LP
  - 0x2—C-DLP
- If the component transitions to a C-State that does not support Control OpClass packet exchange and the Duration field is zero, then see *Component Power Management*.
  - If the Duration field is non-zero, then it indicates how long the component shall remain in this state until it is required to automatically return to its prior state.
- 10          • If the PID field is non-zero, then:
  - The component shall adjust its present power consumption to be less than or equal to the indicated percentage of its maximum power consumption.
  - The Duration field indicates how long the component shall operate at the new power consumption level until it automatically returns to its prior level.
- 15          • C-State and power consumption levels may be changed at the same time. If so, then the Duration field applies to both.

Component power management may be assigned to a resource manager or operating system distinct from the Primary Manager, Primary Fabric Manager, or Secondary Fabric Manager. The component where the resource manager resides is identified by *Core Structure PWR MGR CID / PWR MGR SID*, and *Core Structure Component CAP 1 Control Power Manager Enable* indicates if these fields have been configured.

20          In *Example C-State Notification Zero Duration Exchange* the purple component notifies the orange component that it is transitioning for an unbound duration from the C-Up to the C-LP state. Once the acknowledgment is received, the component initiates the transition. If the purple component were communicating with multiple peers, then it would repeat this exchange with each peer (these exchanges should be done in parallel to reduce the total time to complete the transition).

25          Though not illustrated, if the orange component needed to transmit a non-Control OpClass end-to-end packet, then it would first transmit a C-State Power Control packet to request the purple component to transition to C-Up, and once acknowledged, the orange component could continue normal operations.

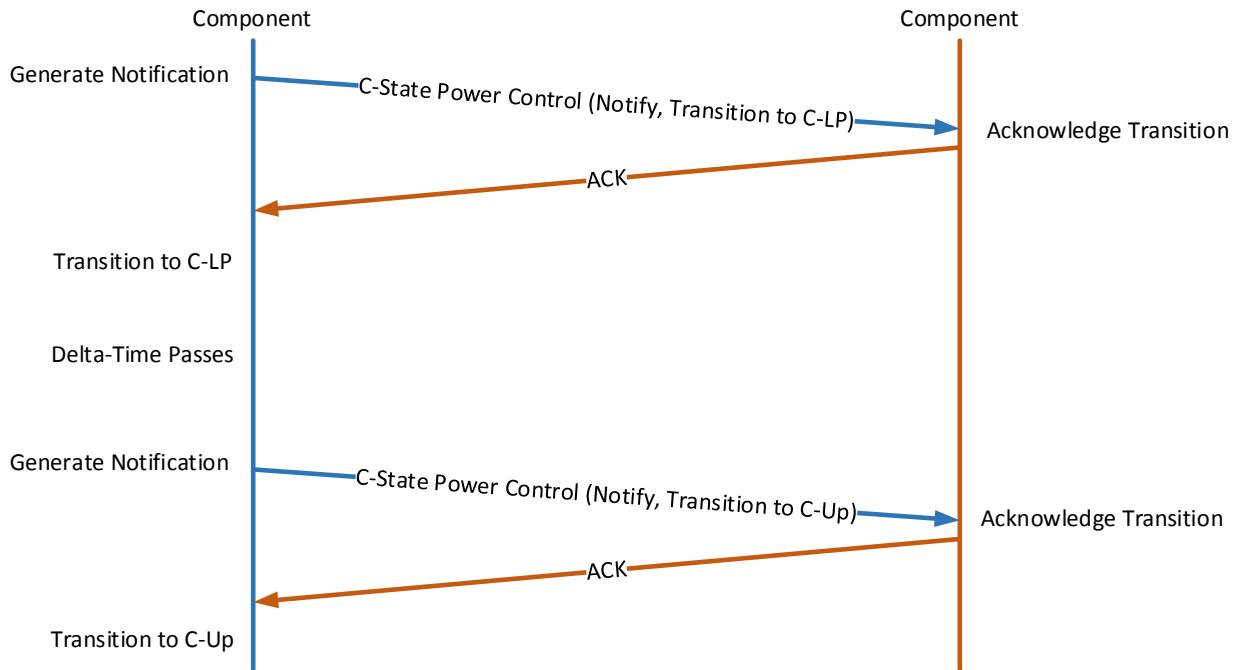


Figure 6-49: Example C-State Notification Zero Duration Exchange

In *Example C-State Notification with Non-Zero Duration Format* the purple component notifies the orange component that it is transitioning for a 100  $\mu$ s duration from the C-Up to the C-DLP state. Once the acknowledgment is received, the component initiates the transition. The purple component initiates a local, component-specific timer. When the timer expires, the component transitions to C-Up, and then transmits a C-State Power Control packet to notify the orange component that it is fully operational.

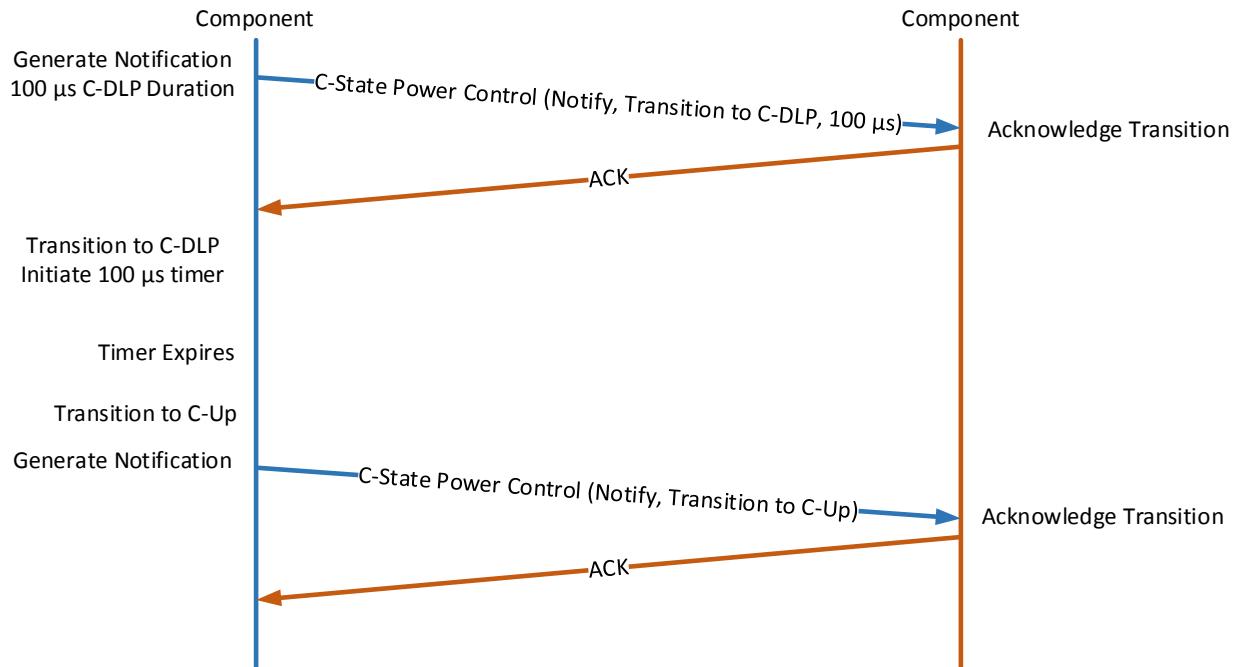


Figure 6-50: Example C-State Notification with Non-Zero Duration Format

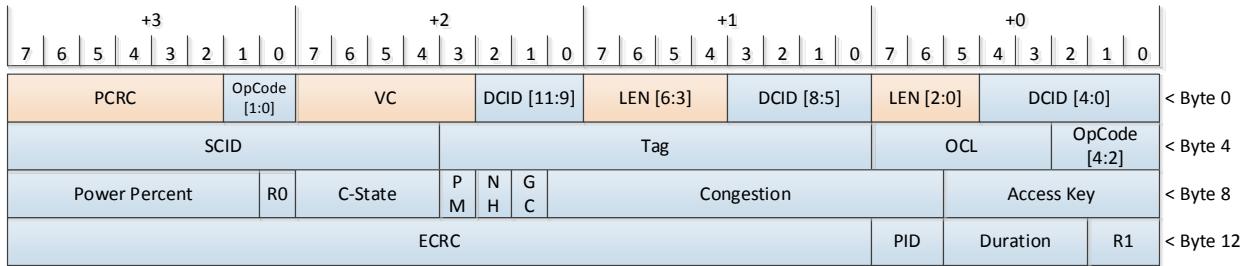


Figure 6-51: C-State Power Control Packet Format

Table 6-42: C-State Power Control Packet Fields

Field Name	Size (bits)	Description
C-State	4	<p>C-State represents either the C-State that the component is being requested to transition to or a notification of the C-State that the component is about to transition.</p> <p>0x0—No state change  0x1—Transition to C-Up  0x2—Transition to C-LP  0x3—Transition C-DLP  0x4—C-Up Transition Notification  0x5—C-LP Transition Notification  0x6—C-DLP Transition Notification  0x7-0xF—Reserved</p>
Duration	4	<p>A component may transition its C-State or its power consumption for an indeterminate or a short-period of time. This field indicates how long the transition will be before the component returns to its prior C-State or power consumption level.</p> <p>0x0—Unlimited time until informed otherwise  0x1—1 <math>\mu</math>s  0x2—2 <math>\mu</math>s  0x3—5 <math>\mu</math>s  0x4—10 <math>\mu</math>s  0x5—100 <math>\mu</math>s  0x6—1 ms  0x7—10 ms  0x8—100 ms  0x9—500 ms  0xA—1 second  0xB—5 seconds  0xC-0xF—Reserved</p>
PID	2	<p>Requests the component to adjust its power to the indicated percentage (Power Percent field) of its maximum power consumption.</p> <p>0x0—No change  0x1—Adjust Power  0x2-0x3—Reserved</p>

Field Name	Size (bits)	Description
<b>Power Percent</b>	7	Percentage of the component's maximum power that the component is requested to either increase or decrease its current power consumption. 0-100—Integer values representing 0% to 100% 101-127—Reserved
<b>R0</b>	1	Reserved
<b>R1</b>	2	Reserved

### 6.10.6. R-Key Update

In a switch topology where a Responder component is accessed by multiple Requesters, a *Region Key (R-Key)* may be frequently updated, e.g., to minimize memory exposure times. Requiring all changes to flow through the management component adds latency and complexity. To avoid this, management 5 may assign R-Key management control to a third-party Requester. Since a third-party does not have direct-access to modify a given R-Key, the third-party uses an R-Key Update packet to indirectly update the R-Keys associated with a specific memory address.

For example, a hypervisor executing on a SoC could be assigned R-Key management to perform R-Key 10 updates for multiple memory components on behalf of its virtual machines. Similarly, a distributed file system agent could be assigned R-Key management for multiple memory components being accessed by multiple SoCs.

The following are the features and requirements associated with R-Key Update packets:

- Any component type may support the R-Key Update packet.
- The R-Key Update packet shall use the *R-Key Update Packet Format*.
- An R-Key Update packet shall be acknowledged by a *Control Standalone Acknowledgment*.
- The Address is used to locate the target memory address (e.g., a page address within a Responder ZMMU) and the associated Read-only and Read-Write R-Keys.
- If the packet is successfully validated (this includes validating that the packet's SCID / SSID fields match the CID / SID R-Key Manager field entry), then the Responder shall use the packet's RO R-Key and the RW R-Key fields to overwrite the respective R-Key fields associated with the memory address. The R-Key fields are overwritten independent of whether a given R-Key is modified.

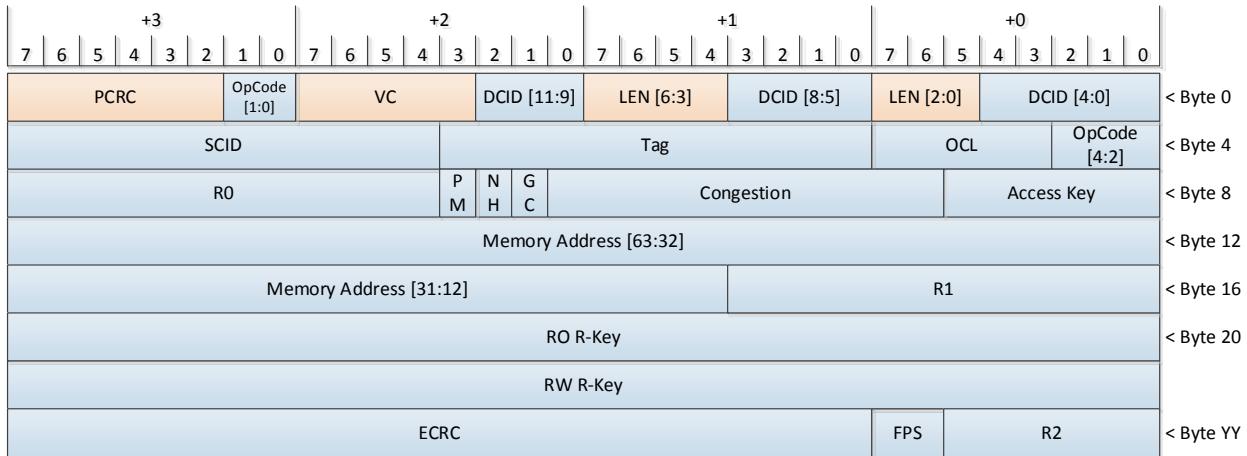


Figure 6-52: R-Key Update Packet Format

Table 6-43: R-Key Update Packet Fields

Field Name	Size (bits)	Description
<b>Address</b>	52	Page address targeted by this request for R-Key update. Address shall be an integer 4096-byte multiple.
<b>RO R-Key</b>	32	Read-only R-Key that overwrites the existing RO R-Key
<b>RW R-Key</b>	32	Read-Write R-Key that overwrites the existing RW R-Key
<b>R0</b>	12	Reserved
<b>R1</b>	12	Reserved
<b>R2</b>	6	Reserved

## 6.10.7. Control Write MSG

A Control Write MSG is used by components that support *In-band Management* to exchange messages (e.g., a message between a fabric manager and a resource manager, or a message between two fabric managers).

The following are the features and requirements associated with Control Write MSG packets:

- Any component type may support the Control Write MSG packet.
- The Control Write MSG packet shall use the *Control Write MSG Packet Format*.
- Unless explicitly stated otherwise by this specification, the functionality, protocol fields, and semantics of a Control Write MSG shall be the same as those specified for a *Write MSG*.
- Setting SDR to 1b shall be done only when performing *Indirect Manager Communication*.
- If DR == 1b, then shall set MSGSZ = 0x1.

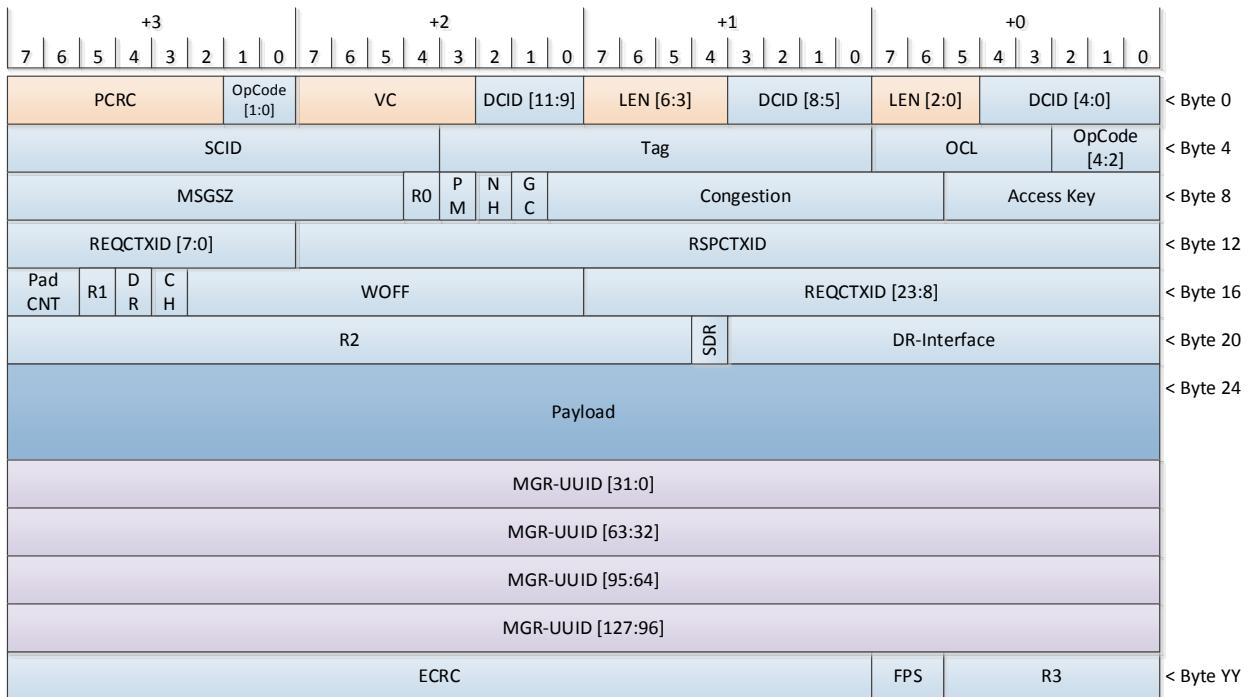


Figure 6-53: Control Write MSG Packet Format

Table 6-44: Control Write MSG Packet Fields

Field Name	Size (bits)	Description
SDR	1	Source Directed Route—Indicates if the DR-Interface field contains the egress interface that management should target in a subsequent directed Control Write MSG packet. If DR == 1b, then the SDR bit shall Reserved. 0b—DR-Interface Reserved 1b—DR-Interface Valid
R0	1	Reserved
R1	1	Reserved
R2	19	Reserved
R3	6	Reserved

## 7. Advanced End-to-end Operations

This section covers advanced request and response packets. These operations are optional, and support will vary by solution and application need.

Unless explicitly stated otherwise by this specification, a Requester shall await receipt of a response packet (a request-specific response or a *Standalone Acknowledgment*) before making ordering decisions that depend upon the successful execution of a given request packet.

### 7.1. Interrupts

10 Interrupt request packets are used to initiate interrupt service event processing at a Responder on behalf of a Requester. *Example Interrupt Set-up and Request Packet Exchange* illustrates an example of how interrupts are configured and interrupt request packets are exchanged. To avoid confusion, the component that originates an Interrupt request packet is called the Interrupt Source and the component targeted by the Interrupt request packet is called the Interrupt Target.

15 Gen-Z supports two types of interrupts, “native interrupts” and “LPD Interrupts”. LPD interrupts are generated by a LPD and carry information not needed by native interrupts. Instances of “interrupt” within this specification may refer to native interrupts, LPD interrupts, or both types of interrupts, depending upon the context.

1. Software such as an operating system allocates one or more interrupt vectors using methods outside of this specification’s scope. The interrupt vector is encoded based on Interrupt Target-specific interrupt execution needs. The interrupt vector for a native interrupt is carried within the interrupt request packet’s Address field. The interrupt vector for an LPD interrupt is carried within the interrupt request packet’s Address and Data fields.
2. For native interrupts, the Interrupt Target communicates the interrupt vector to the Interrupt Source, e.g., using a write request packet to a component-specific memory location. For LPD interrupts, the Interrupt Target (host) configures an MSI or MSI-X Capability structure within the Interrupt Source (LPD) with the interrupt vector.
3. Once configured, the Interrupt Source waits until a component-specific event requires interrupt processing by the Interrupt Target. The Interrupt Source generates an interrupt request packet using the configured interrupt vector.
4. The Interrupt Target receives, validates, and executes the interrupt request packet. If successful, then the Interrupt Target executes component-specific interrupt service processing.
5. Without waiting for interrupt service processing to complete, the Interrupt Target transmits a *Standalone Acknowledgment* to the Requester.

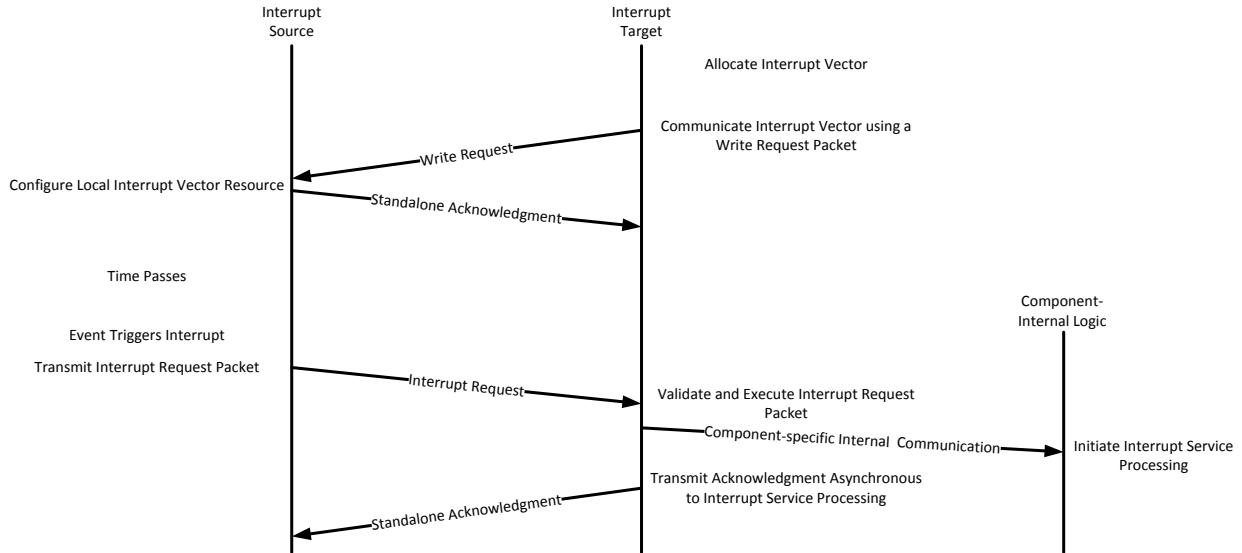


Figure 7-1: Example Interrupt Set-up and Request Packet Exchange

*Example Interrupt Processing Using Requester and Responder ZMMUs* illustrates an example of Interrupt Source and Interrupt Target native interrupt processing if the components support a Requester ZMMU, a Responder ZMMU, or both.

1. The Interrupt Target allocates an interrupt vector that corresponds to a Responder ZMMU page table entry (PTE). The PTE is configured to indicate this entry is used for interrupt processing. The PTE is configured to translate a received interrupt request packet and initiate component-specific interrupt processing.
2. The Interrupt Target communicates the interrupt vector to the Interrupt Source, e.g., using a write request packet to a component-specific memory location.
3. Using the interrupt vector, the Interrupt Source allocates a Requester ZMMU PTE. The PTE is configured to translate any component-local write operation to this page into an interrupt request packet. Minimally, the PTE entry includes the following: the interrupt vector to be encoded within the packet's Address field, the Interrupt Target's CID / SID, and the Interrupt Target-provided R-Key.
4. When an Interrupt Source-specific event requires interrupt processing, a component-local entity performs a write to the page address associated with the interrupt vector. The Interrupt Source walks the Requester ZMMU to locate the PTE, and generates an interrupt request packet.
5. Upon receipt, the Interrupt Target validates and executes the Interrupt request packet. The Interrupt Target uses the packet's Address field to walk the Responder ZMMU to locate the PTE, and uses the packet's R-Key field to validate the Interrupt Source has interrupt access permission. If a valid PTE entry is located, then the Interrupt Target generates a *Standalone Acknowledgment* and executes component-specific interrupt service processing. The Interrupt Target generates the acknowledgment packet without waiting for interrupt service processing to complete.

**Developer Note:** Though a Requester ZMMU can be used in any component type, greater benefit will be realized in more advanced component types such as a SoC, e.g., in SoC-to-SoC messaging solutions. Such solutions benefit from the inherent scalability and ability to efficiently initiate interrupt request packet transmission from application space. Simpler components such as an I/O component can use component-specific means to generate an interrupt request packet. This specification does not architect a Requester ZMMU PTE format for generating LPD interrupts since LPDs are configured for interrupt generation using MSI or MSI-X Capability structures.

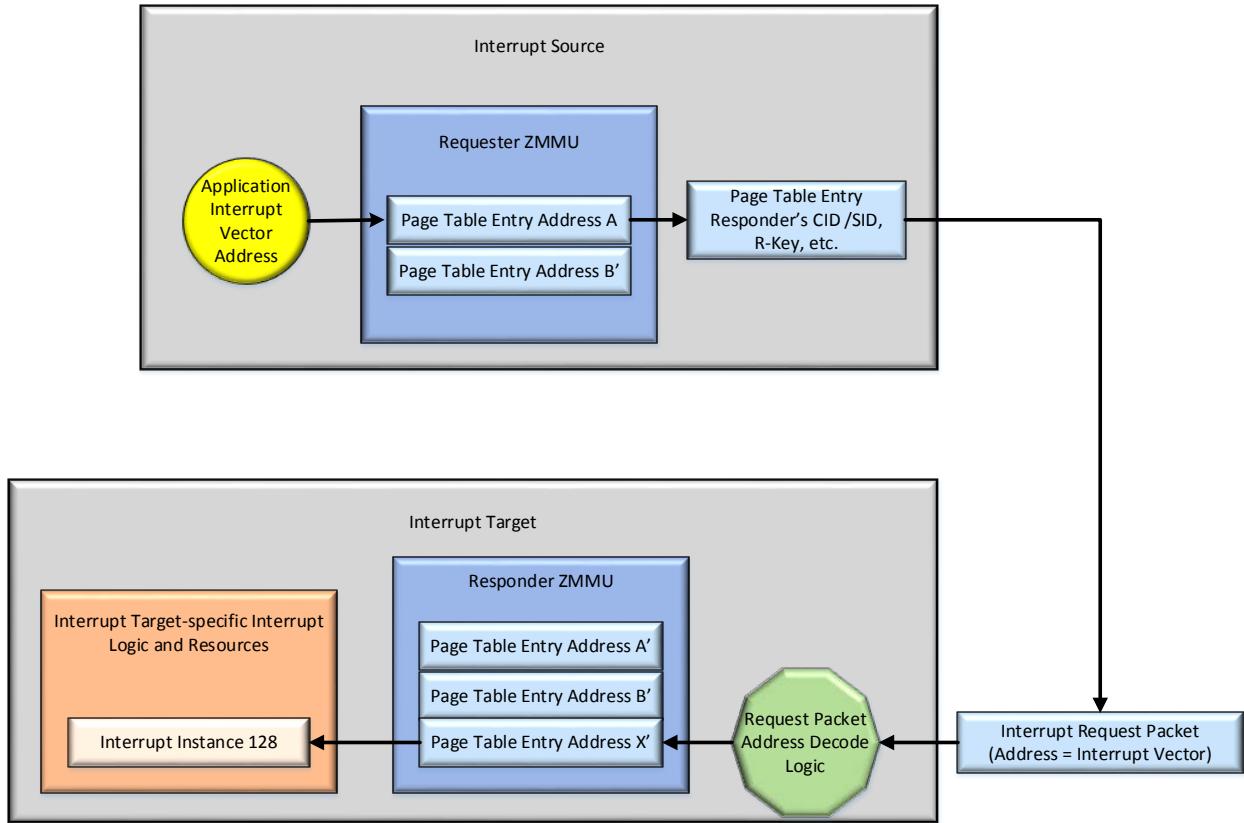


Figure 7-2: Example Interrupt Processing Using Requester and Responder ZMMUs

The following are the features and requirements associated with interrupts:

- Any component type may support interrupt request packets as a Requester (an Interrupt Source), a Responder (an Interrupt Target), or as both a Requester and a Responder (a component that acts as an Interrupt Source and an Interrupt Target, e.g., components that exchange network messages).
- A P2P-Coherency Interrupt request packet shall use the *P2P-Coherency Interrupt Packet Format*.
- A Core 64 Interrupt request packet shall use the *Core 64 Interrupt Packet Format*.
- A Control OpClass Interrupt request packet shall use the *Core 64 Interrupt Packet Format* with LP = 0b.
- Interrupt request packets shall be unicast packets.
- Interrupt request packets shall be validated and executed in less than or equal to the worst-case write request packet latency.
- An interrupt may be transmitted in Write Multi-Op packets (see *Multi-Op Requests*).
- Core 64 Interrupt acknowledgment:
  - If UR == 0b, then a Core 64 interrupt request packet shall be acknowledged by a Core 64 *Standalone Acknowledgment*.
  - If UR == 1b and LP == 1b, then a Core 64 interrupt request packet shall be not be acknowledged. If a packet validation or execution error occurs, then the packet shall be discarded and the error shall be surfaced through an *Unsolicited Event (UE) Packet* based on configured error handling within the *Component Error and Signal Event Structure*.
- A Control OpClass Interrupt request packet shall be acknowledged by a Control OpClass *Standalone Acknowledgment*.

- Unless explicitly stated otherwise by this specification, a Responder shall transmit a *Standalone Acknowledgment* for each received P2P-Coherency interrupt request packet.
  - If *Interface I-CAP 1 Control Omit P2P Standalone Acknowledgment* == 1b, then the Responder shall not transmit a *Standalone Acknowledgment* for a successfully validated and executed P2P-Coherency interrupt request packet.
    - If a P2P-Coherency Write interrupt request packet suffers a non-transient error and *Component Containment* has not been enabled, then the Responder shall return a *Standalone Acknowledgment* (Reason = error detected).
    - Upon detecting a non-transient error, the Responder should stop executing new request packets that modify Data Space resources. If *In-band Management* is supported, then the Responder shall execute new P2P-Coherency *CTL-Read and CTL-Write Packets*.
- The interrupt vector shall be encoded within the interrupt request packet's Address field for native interrupts. The Interrupt Target shall decode the interrupt vector to initiate the component-specific interrupt mechanism. If the maximum address size supported by Data Space or Control Space is less than 64-bits, then the unused upper Address field bits shall be set to zero.
- For LPD interrupts (interrupt request packets with LP = 1b), the interrupt vector shall be encoded within the interrupt request packet's Address and Data fields.
- The time latency between when an interrupt request packet is scheduled and the corresponding interrupt is serviced is solution dependent and will vary as a function of solution attributes and operational characteristics.
- If the Interrupt Target supports the Core 64 OpClass and uses a *Region Key (R-Key)* to isolate interrupts, then the Interrupt Source shall set RK = 1b within the interrupt request packet and the Interrupt Target shall communicate the RW R-Key to the Interrupt Source for subsequent inclusion in an interrupt request packet.

**Developer Note:** When executing LPD interrupts, some Interrupt Targets use a combination of the Address field, LPD BDF<sup>1</sup> field, and / or Data field to enforce implementation-specific access controls in place of or in conjunction with R-Keys.

- Individual interrupt enablement / disablement and polling for native interrupt events are outside of this specification's scope. For example, many operating systems provide APIs to invoke Interrupt Source-specific software drivers to enable or disable interrupt generation or to poll for Requester interrupt events. Given the diversity of component types and implementation options, the mechanisms to control these features are outside of this specification's scope.
- LPD interrupts are managed via an MSI or MSI-X Capability structure. MSI-X and optionally MSI support per-vector masking and interrupt polling mechanisms as specified in *PCI Express Base Specification*.

**Developer Note:** For LPD interrupts, the host configures the MSI/MSI-X Capability structure in the LPD, specifying up to 64 bits of Address and up to 32 bits of Data for each interrupt vector. When system firmware or other Gen-Z-aware software configures the LPD, it configures the Component LPD Structure with the LPD BDF. Thus, the LPD BDF serves as the Device Identifier and the [Address, Data] serve as the Interrupt Identifier.

---

<sup>1</sup> BDF: bus / device / function.

**Developer Note:** There are multiple ways to translate an interrupt request packet's Address field to initiate interrupt processing, especially for native interrupts. For example, consider a generic interrupt controller that requires two inputs: a Device Identifier and an Interrupt Identifier.

- The Device Identifier can be constructed using specific protocol fields, e.g., the SCID and the SSID (if present) can be concatenated to create a unique Device Identifier.
- The Interrupt Identifier can be the interrupt vector encoded within the Address field, e.g., if the Responder uses a 16-bit interrupt vector and a 64 KiB page per interrupt, then the interrupt vector is located at Address[31:16].

There are multiple ways to construct a Requester ZMMU PTE, e.g., consider a Responder that isolates 10 interrupts to 64 KiB pages and uses a Responder-supplied 16-bit interrupt vector. The encoded address could look as follows:

- Address[15:0] = 0x0
- Address[31:16] = interrupt vector
- Address[63:32] = Requester-supplied bits to delineate this page

**Developer Note:** Interrupt service event processing implementations are strongly encouraged to tolerate spurious interrupts. Spurious interrupts can occur for a variety of reasons including due to interrupt request packet retransmission as a result of a transient error.

**Developer Note:** Though the interrupt packet formats do not illustrate the LPD Field, if LP = 1, then the LPD Field will be present in addition to the Data field.

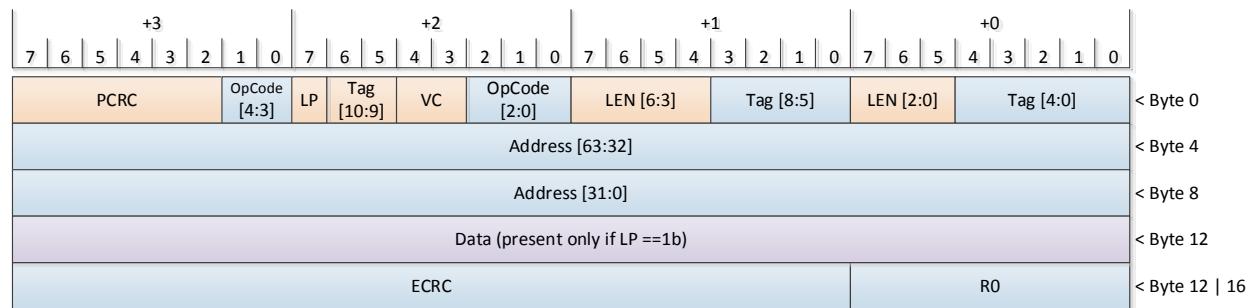


Figure 7-3: P2P-Coherency Interrupt Packet Format

Table 7-1: P2P-Coherency Interrupt Packet Fields

Field Name	Size (bits)	Description
Data	32	If LP == 1b, then the LPD Interrupt Data field shall be present. If LP == 0b, then the LPD Interrupt Data field shall not be present.
R0	8	Reserved

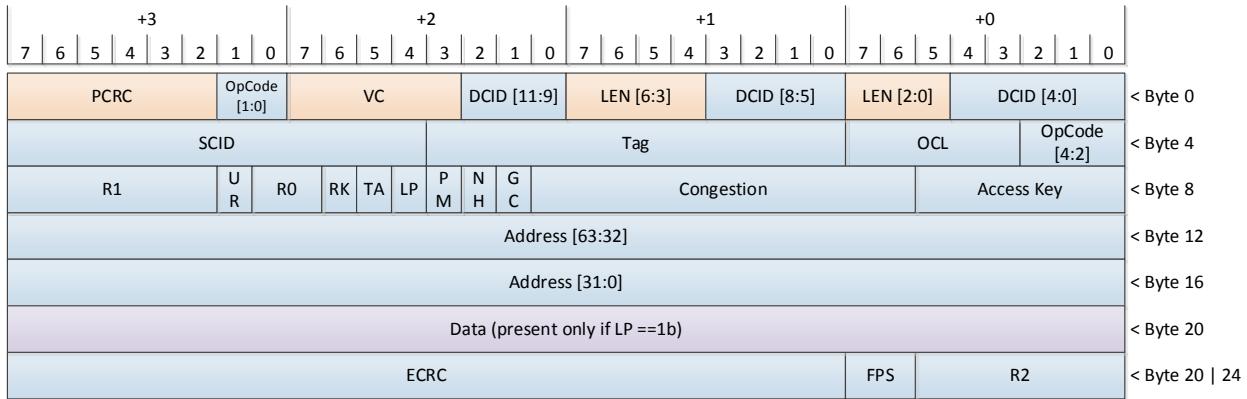


Figure 7-4: Core 64 Interrupt Packet Format

Table 7-2: Core 64 Interrupt Packet Fields

Field Name	Size (bits)	Description
<b>Data</b>	32	If LP == 1b, then the LPD Interrupt Data field shall be present. If LP == 0b, then the LPD Interrupt Data field shall not be present.
<b>R0</b>	2	Reserved
<b>R1</b>	6	Reserved
<b>R2</b>	6	Reserved

## 7.2. Atomic Request and Response Packets

Atomic request packets are indivisible and irreducible requests that allow addressed memory locations to be accessed and optionally modified without interference from other in-flight request packets. Atomic request packets are used to construct advanced synchronization mechanisms when there are multiple producers and / or consumers, to enable lock-free statistics and counter updates, to enable lock-free queue management operations, and many other services.

The following are the features and requirements associated with Atomic request and response packets:

- Any component type may support Atomic request packets as a Requester, a Responder, or as both a Requester and a Responder.
- Explicit Atomic 1 request packets are not idempotent. A Responder shall take the actions as described in *Non-idempotent Request (NIR)* to prevent non-deterministic results and behaviors.
- A component may support all or a subset of the Atomic request types (see *Atomic Request Types*).
- Atomic requests come in 8-bit, 16-bit, 32-bit, 64-bit, 128-bit, 256-bit, and 512-bit sizes.
  - A given atomic request shall use a common size for the operand, accessed memory, and returned data. For example, a 32-bit Add uses a 32-bit add operand, operates on a 32-bit memory location, and returns a 32-bit summed result.
- The address of the data shall be an integer multiple of the data size, i.e.,
  - If an 8-bit data value, then the address shall be a 1-byte multiple
  - If an 16-bit data value, then the address shall be a 2-byte multiple
  - If an 32-bit data value, then the address shall be a 4-byte multiple

- If an 64-bit data value, then the address shall be an 8-byte multiple
- If an 128-bit data value, then the address shall be a 16-byte multiple
- If an 256-bit data value, then the address shall be a 32-byte multiple
- If an 512-bit data value, then the address shall be a 64-byte multiple
- The data payload—operands and results—shall be formatted such that the first byte of the payload is the least significant byte of the first data value and subsequent bytes are strictly increasing in significance.
- The endian format used by Responders to read and write data at the target location shall be implementation specific, and may be whatever the Responder determines to be appropriate for the target memory—little endian or big endian. A component indicates supported endian type via the *OpCode Set Structure*.
- Atomic request packets should execute quickly.
  - If an error is detected, a *Standalone Acknowledgment* (Reason = detected error) shall be returned.
  - If successful and FR == 0b, then an Atomic Response packet shall be returned. If successful and FR == 1b, then an Atomic Results packets shall be returned. An Atomic response packet shall contain the Atomic request packet's Tag.
  - If a Responder supports the P2P-Core or the P2P-Coherency OpClass, then:
    - The Responder shall validate and execute an Atomic request packet and transmit an Atomic response packet within the time indicated by the Responder's *OpCode Set Structure* Atomic LAT field. If it fails to do so, then this shall be treated as an unrecoverable error and shall be handled as configured in the *Component Error and Signal Event Structure* (if supported).
    - Since P2P-Core and P2P-Coherency interfaces are required to support and have *Link-level Reliability (LLR)* enabled, only non-transient events can cause response packets to be lost. As such, a Requester shall not retransmit an Atomic request packet and a Responder shall not maintain the Atomic request packet results. If a non-transient error is detected, then the Responder shall handle it as configured in the *Component Error and Signal Event Structure* (if supported).
  - If a Responder supports the Atomic 1 OpClass, then:
    - If an Atomic request packet's Deadline expires or if an Atomic request packet cannot be validated and executed before the Responder's Deadline expires (see *Deadline Semantics*), then the request packet shall not be executed and shall be silently discarded. Management uses the Responder's *OpCode Set Structure* Atomic LAT field as an input to configure the Responder's Deadline value.
    - If an Atomic request packet was successful executed, then the Responder shall maintain the execution results until these are explicitly released via a *Non-Idempotent Request Release (NIRR)* or the failsafe timer expires. This enables the same results to be returned if the request packet is retransmitted.
    - If an Atomic request packet was successfully executed but the corresponding Atomic response packet cannot be transmitted before the Responder's Deadline expires (see *Deadline Semantics*), then the Atomic Response packet shall be silently discarded. The results of executing the Atomic request packet shall be retained until the Requester retransmits the original request packet and completes all Atomic-related processing.
  - The Responder shall have exclusive control of the underlying resource until the result of the Atomic operation has been successfully written to the underlying resource. For example, if multiple Atomic request packets attempt to modify a resource (in whole or

part, e.g., a shared cache line), then the Responder shall be responsible for ensuring that only one Atomic request packet accesses and updates the resource at a given time.

- If an Atomic request packet indicates persistence is required, then the Responder shall not schedule an Atomic response packet until the request packet has been successfully executed and the result has reached a processing state such that the update to the underlying media is guaranteed to succeed, or an error has been detected.
- If an Atomic request packet fails validation or execution, then the state of the target location shall remain unchanged. If the update to the target media fails for any reason, then the media may be in a non-deterministic state.
- If an Atomic request packet attempts to access poisoned data, then the request packet shall not be executed and the state of the memory location shall remain unchanged. A *Standalone Acknowledgment* (Reason = Poison Data (PD) Failure) shall be returned.
- An Atomic Vector request packet enables multiple memory locations to be atomically updated in a single request packet, e.g., instead of issuing 32 Atomic Add request packets, a single Atomic Sum Vector request containing 32 Add operands can be issued in a single packet.
  - A Responder shall complete all vector operations or none; partial execution shall not be performed.
- If a Requester issues more request packets than supported by the Responder, then the Responder may return a *Responder Not Ready (RNR) NAK* or silently discard the request packet and rely upon the Requester to retransmit.
- If the addressed media targeted by an Atomic request has entered Fatal Media Error Containment, then the request packet shall not be executed, and a *Standalone Acknowledgment* (Reason = Fatal Media Error Containment Triggered) shall be returned.
- If a Responder supports a cache resource, then upon receipt of an Atomic request packet with TC == 1b, the Responder shall place the packet's payload in the cache. Placement may require the cache to evict existing cache lines to free up space and, if supported, then the Responder performing cache coherency actions.
  - If a Responder's cache agent determines that the Requester is not permitted to modify the addressed cache line (e.g., is not the current owner), then the Responder shall handle this as an "Access Error (AE)—Invalid Access Permission".

The following examples illustrate how little endian and big endian Atomic requests are handled:

- Little endian example: For a 64-bit Swap Request targeting location 0x100 with the target memory in little endian format, the first byte of the data payload (Swap Operand) is written to location 0x100, the second byte to 0x101, and so on until the final byte is written to location 0x107. Before performing the writes, the Responder first reads the target memory location so it can return the original value in the Atomic Response packet. The byte address correspondence to the data in the Response is identical to that in the Request.
- Big endian example: For a 64-bit Swap Request targeting location 0x100 with the target memory in big endian format, the first byte of the data payload (Swap Operand) is written to location 0x107, the second byte to 0x106, and so on until the final byte is written to location 0x100. Before performing the writes, the Responder first reads the target memory location so it can return the original value in the Atomic Response packet. The byte address corresponding to the data in the response packet is identical to that in the request packet.
- *Example of Responder Target Memory Access for Add* illustrates how this may be applied to a 64-bit Add request. The bytes in the Add-Op operand and the results are numbered 0—7 with byte 0 being the least significant and byte 7 being the most significant. In each case, the Responder

fetches the target memory operand using the appropriate endian format. Next, the Atomic Compute Logic in the Responder performs the Add request using the original target memory value and the Add-Op operand from the Add Request. The Responder stores the Add result back to the target memory using the same endian format used for the fetch.

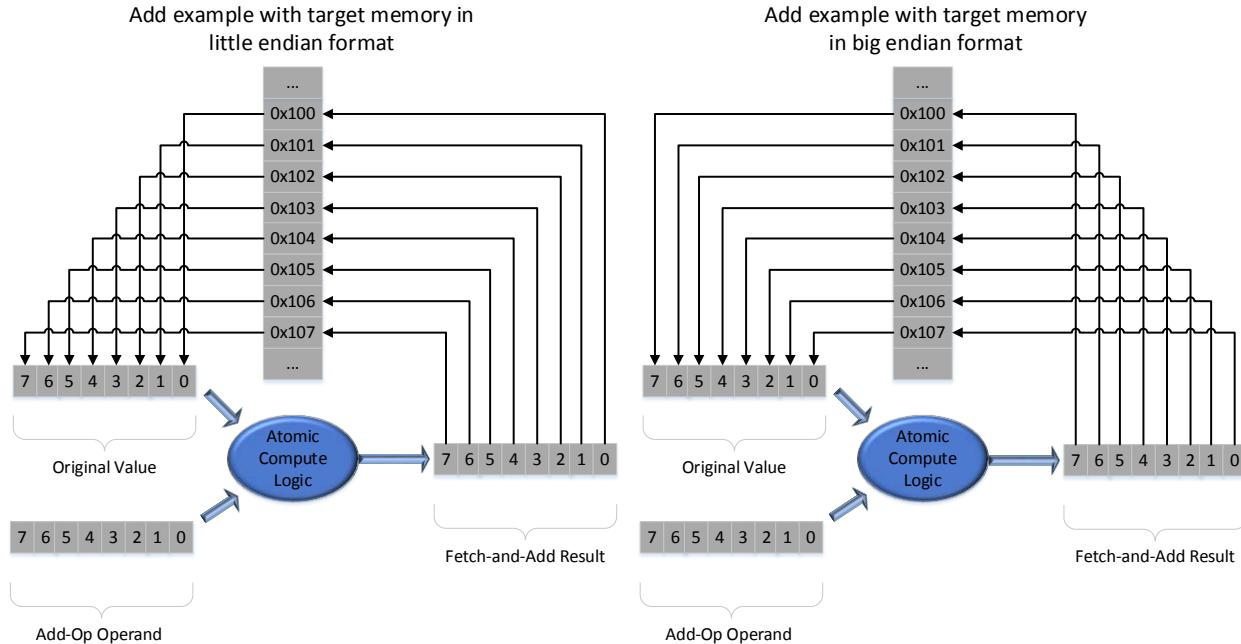


Figure 7-5: Example of Responder Target Memory Access for Add

### 7.2.1. Atomic Request Types

The following atomic request types and associated operational semantics are as specified:

- Arithmetic operations may use signed or unsigned Data and operands.
- Logical operations use unsigned Data and operands.
- Unless explicitly stated otherwise by this specification, if the Atomic request packet's FR == 1b, then the Responder transmits an Atomic Results response packet to return the operation's results, else the Responder transmits an Atomic Response packet.
- Add
  - Use Atomic Single Address Single Operand packet format
  - Data = Read(Address)
  - Sum = Data + Operand
  - Write(Address, Sum)
  - If the packet's FR == 1b, then the Data value shall be placed in the Result field.
- Sum Memory
  - Use Atomic Dual-Address No Operand packet format
  - Data 1 = Read(Address 1)
  - Data 2 = Read (Address 2)
    - Data 1 and Data 2 shall be the same size, else a return a MP error.
  - Sum = Data 1 + Data 2
  - Write(Address 1, Sum)
  - If packet's FR == 1b, then the Sum value shall be placed in the Result field.

- Swap
  - Use Atomic Single Address Single Operand packet format
  - Data = Read(Address)
  - Write(Address, Operand)
  - If packet's FR == 1b, then the Data value shall be placed in the Result field.
- Compare and Swap (CAS)
  - Use Atomic Single Address Dual-Operand packet format
  - Data = Read(Address)
  - If (Data == Operand 1) then Write(Address, Operand 2)
  - If packet's FR == 1b, then the Data value shall be placed in the Result field.
- Compare and Swap (CAS) Not Equal
  - Use Atomic Single Address Dual-Operand packet format
  - Data = Read(Address)
  - If (Data ≠ Operand 1) then Write(Address, Operand 2)
  - If packet's FR == 1b, then the Data value shall be placed in the Result field.
- Compare Store Twin
  - Use Atomic Single Address Single Operand packet format
  - Data1 = Read(Address)
  - Data2 = Read (Address + 1) //Load Data2 from Address + operand size
  - If (Data1 == Data2) then {
    - Write(Address, Operand)
    - Write(Address + 1, Operand)
  - }
  - The packet's FR field shall be set to 0b.
- Increment Bounded
  - Use Atomic Single Address No Operand packet format
  - Data1 = Read (Address)
  - Data2 = Read (Address + 1) // Load Data2 from Address + operand size
  - If (Data1 ≠ Data2) then Write (Address, Data1++)
  - If packet's FR == 1b, then the Data1 value shall be placed in the Result field.
- Increment Equal
  - Use Atomic Single Address No Operand packet format
  - Data1 = Read (Address)
  - Data2 = Read (Address + 1) // Load Data2 from Address + operand size
  - If (Data1 == Data2) then Write (Address, Data1++)
  - If packet's FR == 1b, then the Data1 value shall be placed in the Result field.
- Decrement Bounded
  - Use Atomic Single Address No Operand packet format
  - Data1 = Read (Address)
  - Data2 = Read (Address + 1) // Load Data2 from Address + operand size
  - If (Data1 ≠ Data2) then Write (Address, Data1--)
  - If packet's FR == 1b, then the Data1 value shall be placed in the Result field.
- Test-Zero-and-Modify
  - Use Atomic Single Address Dual Operand packet format
  - Data = Read(Address)
  - If (Data == 0) then Write (Address, (Data | Operand 1) & Operand 2)
  - If packet's FR == 1b, then the Data value shall be placed in the Result field.

- Load Min [Signed | Unsigned]
  - Use Atomic Single Address Single Operand packet format
  - Data = Read (Address)
  - If (US == Unsigned) then {
    - if (Operand < Unsigned (Data)) then Write (Address, Operand)
  - } Else {
    - if (Operand < Signed (Data)) then Write (Address, Operand)
  - }
  - If packet's FR == 1b, then the Data value shall be placed in the Result field.
- Load Max [Signed | Unsigned]
  - Use Atomic Single Address Single Operand packet format
  - Data = Read (Address)
  - If (US == Unsigned) then {
    - if (Operand > Unsigned (Data)) then Write (Address, Operand)
  - } Else {
    - if (Operand > Signed (Data)) then Write (Address, Operand)
  - }
  - If packet's FR == 1b, then the Data value shall be placed in the Result field.
- Logical AND / OR / XOR
  - Use Atomic Single Address Single Operand packet format
  - Data = Read (Address)
    - Write (Address, Data [Logical AND | OR | XOR] Operand)
  - If packet's FR == 1b, then the Data value shall be placed in the Result field.
- Atomic Vector Sum
  - Use Atomic Vector packet format
  - TA<sub>0</sub> = Address
  - For each vector operand [K = 0 to NV],
    - Data<sub>K</sub> = Read (TA<sub>K</sub>)
    - Result<sub>K</sub> = Data<sub>K</sub> + V<sub>K</sub>
    - Write(TA<sub>K</sub>, Result<sub>K</sub>)
    - TA<sub>K+1</sub> = TA<sub>K</sub> + (sizeof(Vector Operand) bytes)
  - If packet's FR == 1b, then the Results vector shall be placed in the Result field.
- Atomic Vector Logical
  - Use Atomic Vector packet format
  - TA<sub>0</sub> = Address
  - For each vector operand [K = 0 to NV],
    - Data<sub>K</sub> = Read (TA<sub>K</sub>)
    - Result<sub>K</sub> = Data<sub>K</sub> [Logical AND | OR | XOR] Operand<sub>K</sub>
    - Write(TA<sub>K</sub>, Result<sub>K</sub>)
    - TA<sub>K+1</sub> = TA<sub>K</sub> + (sizeof(Vector Operand) bytes)
  - If packet's FR == 1b, then the Results vector shall be placed in the Result field.
- Atomic Fetch
  - Use Atomic Single Address No Operand packet format with FR = 1b
  - Data = Read (Address)
  - If supported, then the Responder shall use the *Core Structure* ENIRT instead of the *Core Structure* NIRT to ensure the results resource is eventually released.
  - Return Data in Result field

## 7.2.2. Atomic Packet Formats

The following packet formats are used for Atomic request and Atomic response packets. *Common Atomic Packet-specific Fields* specifies a set of fields that are common to multiple Atomic packet formats (see *Base Formats for End-to-end Packets* for additional fields that are common to Atomic request and Atomic response packets).

5

Table 7-3: Common Atomic Packet-specific Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>Size</b>	SZ	3	<p>Indicates operand size</p> <p>0x0—8-bit operand 0x1—16-bit operand 0x2—32-bit operand 0x3—64-bit operand 0x4—128-bit operand 0x5—256-bit operand 0x6—512-bit operand 0x7—Reserved</p> <p>If the operand size is less than 64 bits, then the payload field shall contain 32-bits and [0-3] Pad bytes.</p>
<b>Fetch Results</b>	FR	1	<p>Indicates if results are to be returned, i.e., the type of Atomic response packet.</p> <p>0b—Atomic Response (no results) 1b—Atomic Results Response (results)</p>
<b>Floating</b>	FL	1	<p>Floating Point Data and Operands</p> <p>0b—Indicates integer data and operands. Use two's complement arithmetic ignoring any carry or overflow. If a non-arithmetic operation, then this field shall be Reserved.</p> <p>1b—Indicates floating point data and operands. Use single or double precision floating point arithmetic as appropriate.</p> <p>This field is applicable only to arithmetic Atomic request packets, else treated as Reserved.</p>
<b>Unsigned</b>	US	1	<p>Unsigned Operation—Indicates if the operands are treated as signed or unsigned data</p> <p>In a dual-Address request packet, the US bit applies to the data targeted by the Address 1 and Address 2 fields.</p> <p>In a vector operand request packet, the US bit applies to each Vector Operand.</p> <p>0b—Signed 1b—Unsigned</p>

Field Name	Field Abbreviation	Size (bits)	Description
<b>ARSP Reason</b>	-	4	<p>This Atomic Response Reason communicates additional information relative to the associated request packet's execution. Only the following Atomic Response Reasons may be returned in an Atomic response packet; all other conditions shall be returned using a <i>Standalone Acknowledgment</i>.</p> <p>0x0—No Error 0x1-0xB—Reserved 0xC—Data Correctable Error Warning 0xD—Data Uncorrectable Error Detected 0xE—Poison Data (PD) Failure 0xF—Reserved</p>
<b>Result</b>	-	-	<p>Result data shall be contiguously placed starting at byte 0. Result length shall be an integer 4-byte multiple.</p> <p>The Responder shall append 0-3 pad bytes to the Result field to ensure the Result length is an integer 4-byte multiple.</p>
<b>Operand</b>	-	Varies	<p>This field contains the Operand used in the Atomic operation. This size of the field depends upon the SZ field.</p> <ul style="list-style-type: none"> <li>• An 8-bit operand is placed at Operand[7:0]</li> <li>• A 16-bit operand is placed at Operand[15:0]</li> <li>• A 32-bit operand is placed at Operand[31:0]</li> <li>• A 64-bit operand is placed at Operand[63:0]</li> <li>• A 128-bit operand is placed in Operand[127:0]</li> <li>• A 256-bit operand is placed in Operand [255:0]</li> <li>• A 512-bit operand is placed in Operand [511:0]</li> </ul> <p>If the size of an operand is less than or equal to 32 bits, then the Operand length field shall be 32 bits.</p> <p>If the size of the an operand is less than 32 bits, then the Pad CNT shall be non-zero and 1-3 pad bytes shall be appended to the operand to ensure integer 4-byte alignment.</p> <p>If a packet contains multiple Operands, then all Operands shall be the same size.</p>
<b>NV</b>	-	4	<p>Number of Vector Operands within Atomic Vector Request. Operands are packed such that the first operand is placed at byte 0 of the request payload (Vector Operand), the second operand immediately following the first, and so forth.</p> <p>0x0—2 Operands 0x1—4 Operands 0x2—8 Operands 0x3—16 Operands 0x4—32 Operands</p>

Field Name	Field Abbreviation	Size (bits)	Description
Vector Operand			0x5—64 Operands 0x6—128 Operands 0x7—256 Operands 0x8-0xF—Reserved Vectors containing 64 or more operands shall apply only to 32-bit or smaller sized operands
	-	-	This field contains the operands used in an atomic vector request. All individual operands shall be of the same size as indicated by SZ.
ByteEndian	BE	1	Indicates if the operands use little or big endian. This bit is compared to <i>OpCode Set Structure</i> CAP 1 Atomic Data Endian Type. If BE does not correspond to a supported Atomic Data Endian Type, then the request packet shall be handled as a MP error. If the Atomic operation does not take endianness into account, then the BE bit shall be ignored. 0b—Little Endian 1b—Big Endian

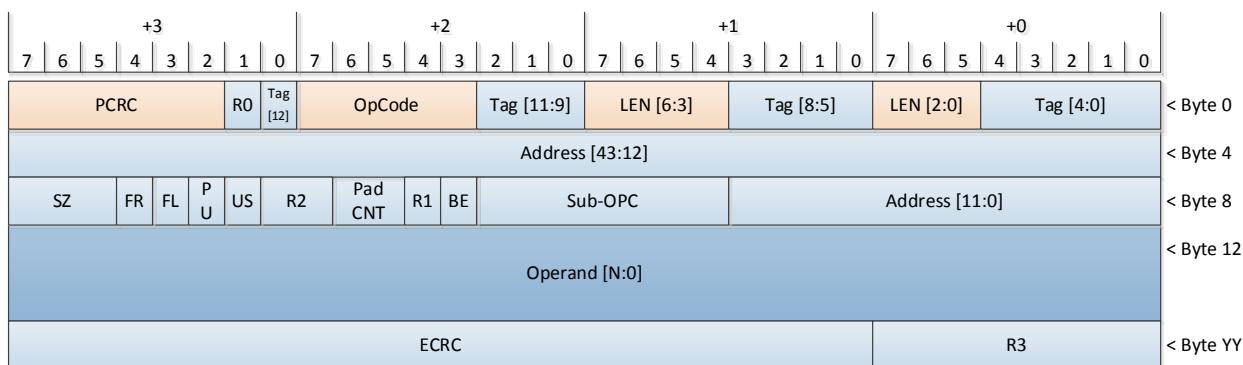


Figure 7-6: P2P-Core Atomic Single Address Single Operand Packet Format

Table 7-4: P2P-Core Atomic Single Address Single Operand Packet-specific Fields

Field Name	Size (bits)	Description
R0	1	Reserved
	1	Reserved
	2	Reserved
	8	Reserved

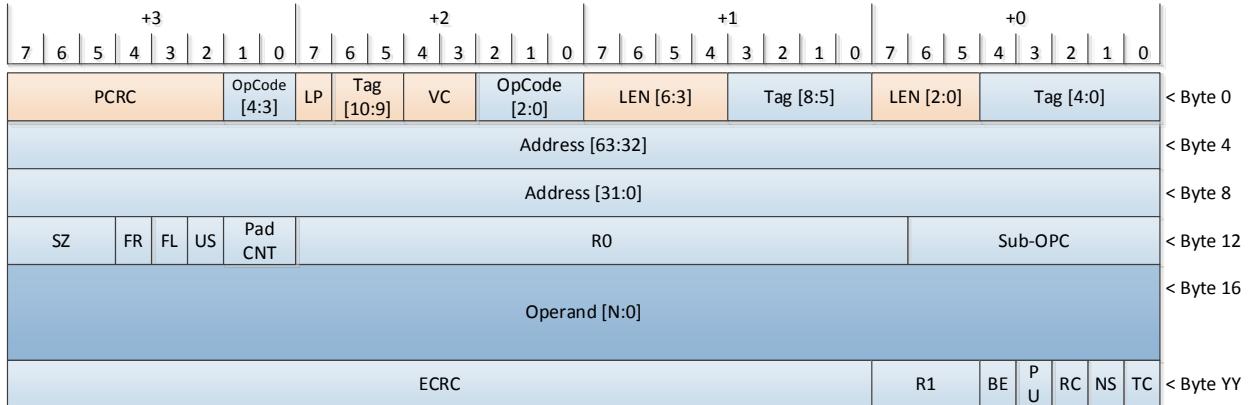


Figure 7-7: P2P-Coherency Atomic Single Address Single Operand Packet Format

Table 7-5: P2P-Coherency Atomic Single Address Single Operand Packet-specific Fields

Field Name	Size (bits)	Description	
R0	17	Reserved	
	3	Reserved	

Figure 7-8: Explicit Atomic 1 Single Address Single Operand Packet Format

Table 7-6: Explicit Atomic 1 Single Address Single Operand Packet-specific Fields

Field Name	Size (bits)	Description	
R0	2	Reserved	

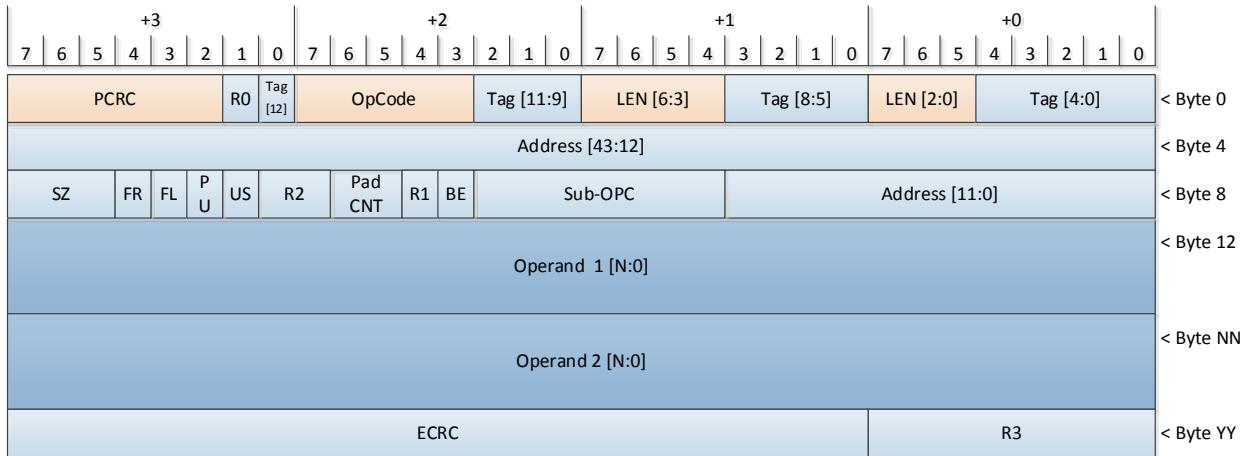


Table 7-7: P2P-Core Atomic Single Address Dual-Operand Request Packet-specific Fields

Field Name	Size (bits)	Description
R0	1	Reserved
	1	Reserved
	2	Reserved
	8	Reserved

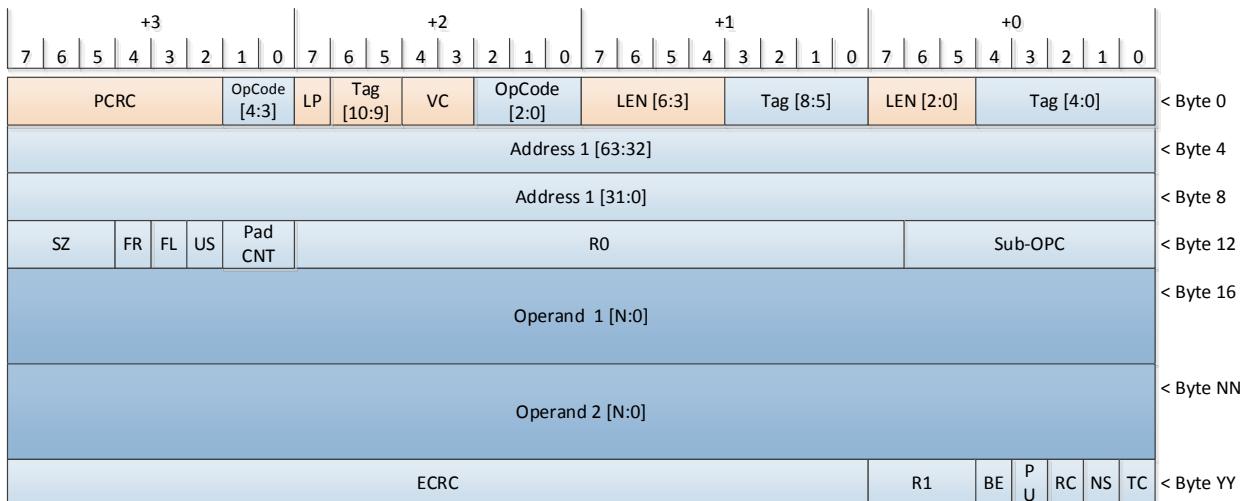


Table 7-8: P2P-Coherency Atomic Single Address Dual-Operand Request Packet-specific Fields

Field Name	Size (bits)	Description
R0	17	Reserved
	3	Reserved

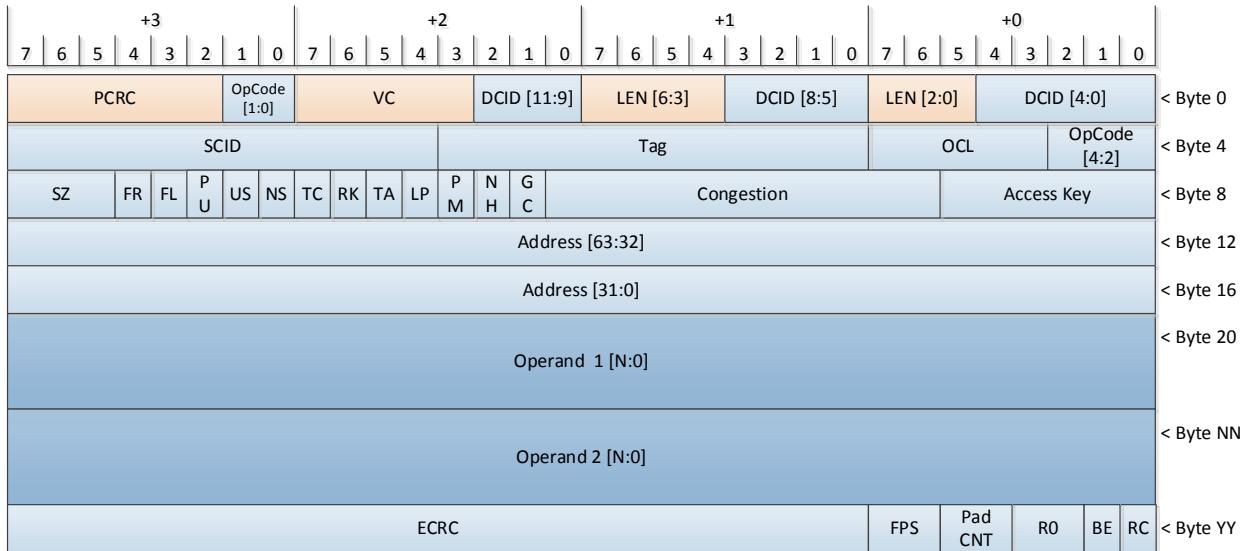


Figure 7-11: Explicit Atomic 1 Single Address Dual-Operand Request Packet Format

Table 7-9: Explicit Atomic 1 Single Address Dual-Operand Request Packet-specific Fields

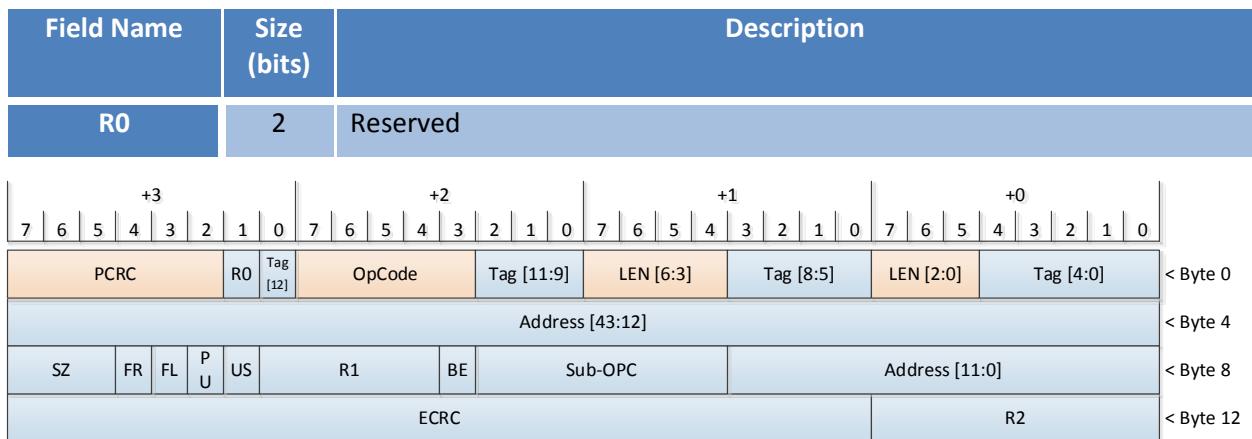


Figure 7-12: P2P-Core Atomic Single Address No Operand Packet Format

Table 7-10: P2P-Core Atomic Single Address No Operand Packet-specific Fields

Field Name	Size (bits)	Description
<b>R0</b>	1	Reserved
<b>R1</b>	5	Reserved
<b>R2</b>	8	Reserved

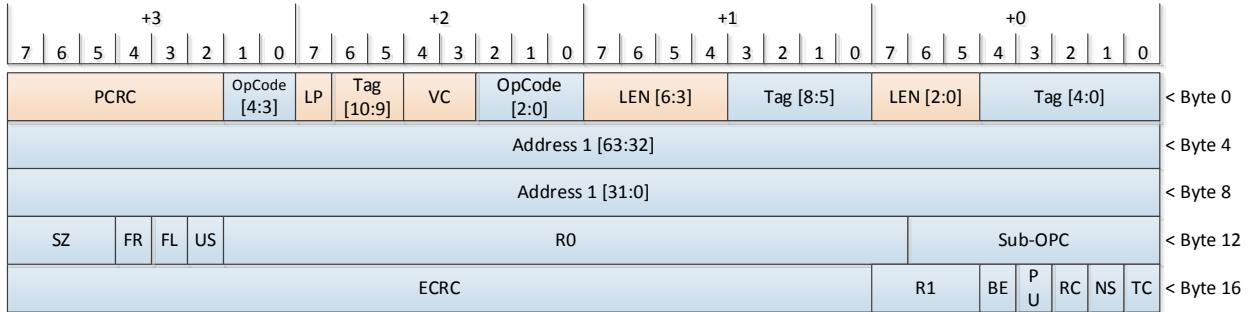


Figure 7-13: P2P-Coherency Atomic Single Address No Operand Packet Format

Table 7-11: P2P-Coherency Atomic Single Address No Operand Packet-specific Fields

Field Name	Size (bits)	Description	
<b>R0</b>	19	Reserved	
<b>R1</b>	3	Reserved	

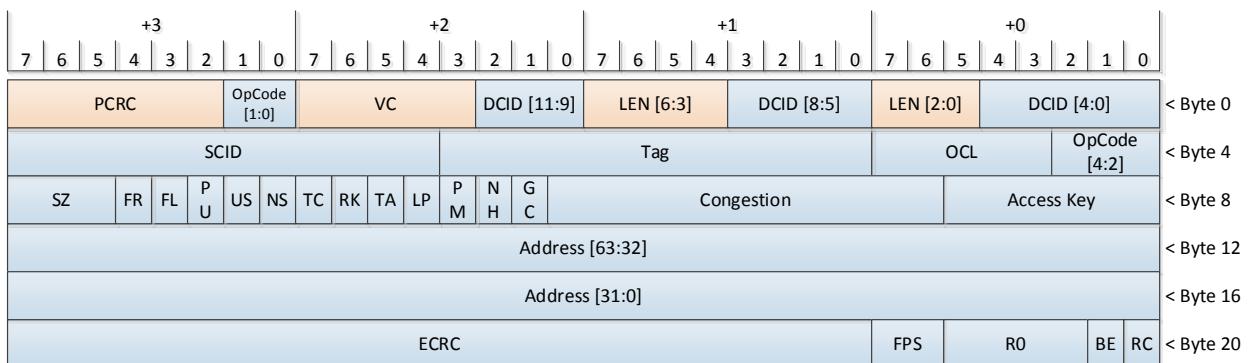


Figure 7-14: Explicit Atomic 1 Single Address No Operand Packet Format

Table 7-12: Explicit Atomic 1 Single Address No Operand Packet-specific Fields

Field Name	Size (bits)	Description	
<b>R0</b>	4	Reserved	
PCRC	3	RO	Tag [12]
Address 1 [43:12]			
Address 2 [43:40]			
Sub-OPC			
Address 1 [11:0]			
Address 2 [39:8]			
ECRC			
Address 2 [7:0]			

Figure 7-15: P2P-Core Atomic Dual-Address No Operand Packet Format

Table 7-13: P2P-Core Atomic Dual-Address No Operand Packet-specific Fields

Field Name	Size (bits)	Description																	
<b>R0</b>	1	Reserved																	
<b>R1</b>	1	Reserved																	
Address 1 [63:32]																			
Address 1 [31:0]																			
SZ	FR	FL	US	R0				Sub-OPC											
Address 2 [63:32]																			
Address 2 [31:0]																			
ECRC								R1	BE	P U	RC	NS	TC						
< Byte 0																			
< Byte 4																			
< Byte 8																			
< Byte 12																			
< Byte 16																			
< Byte 20																			
< Byte 24																			

Figure 7-16: P2P-Coherency Atomic Dual-Address No Operand Packet Format

Table 7-14: P2P-Coherency Atomic Dual-Address No Operand Packet-specific Fields

Field Name	Size (bits)	Description										
<b>R0</b>	19	Reserved										
<b>R1</b>	3	Reserved										
Address 1 [63:32]												
Address 1 [31:0]												
Address 2 [63:32]												
Address 2 [31:0]												
ECRC								FPS	R0	BE	RC	
< Byte 0												
< Byte 4												
< Byte 8												
< Byte 12												
< Byte 16												
< Byte 20												
< Byte 24												
< Byte 28												

Figure 7-17: Explicit Atomic 1 Dual-Address No Operand Packet Format

Table 7-15: Explicit Atomic 1 Dual-Address No Operand Packet-specific Fields

Field Name	Size (bits)	Description																			
<b>R0</b>	4	Reserved																			
																					
																					
Address [43:12]																					
SZ	FR	FL	P U	US	R1	NV	BE	Sub-OPC		Address [11:0]											
Vector Operands																					
ECRC								R2													
< Byte 0																					
< Byte 4																					
< Byte 8																					
< Byte 12																					
< Byte YY																					

Figure 7-18: P2P-Core Atomic Vector Packet Format

Table 7-16: P2P-Core Atomic Vector Packet-specific Fields

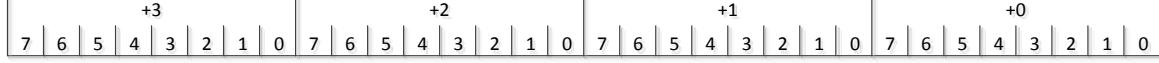
Field Name	Size (bits)	Description																			
<b>R0</b>	1	Reserved																			
<b>R1</b>	1	Reserved																			
<b>R2</b>	8	Reserved																			
																					
																					
Address 1 [63:32]																					
Address 1 [31:0]																					
SZ	FR	FL	US	NV		R0				Sub-OPC											
Vector Operands																					
ECRC								R1	BE	P U	RC	NS	TC								
< Byte 0																					
< Byte 4																					
< Byte 8																					
< Byte 12																					
< Byte 16																					
< Byte YY																					

Figure 7-19: P2P-Coherency Atomic Vector Packet Format

Table 7-17: P2P-Coherency Atomic Vector Packet-specific Fields

Field Name	Size (bits)	Description									
<b>R0</b>	15	Reserved									
<b>R1</b>	3	Reserved									

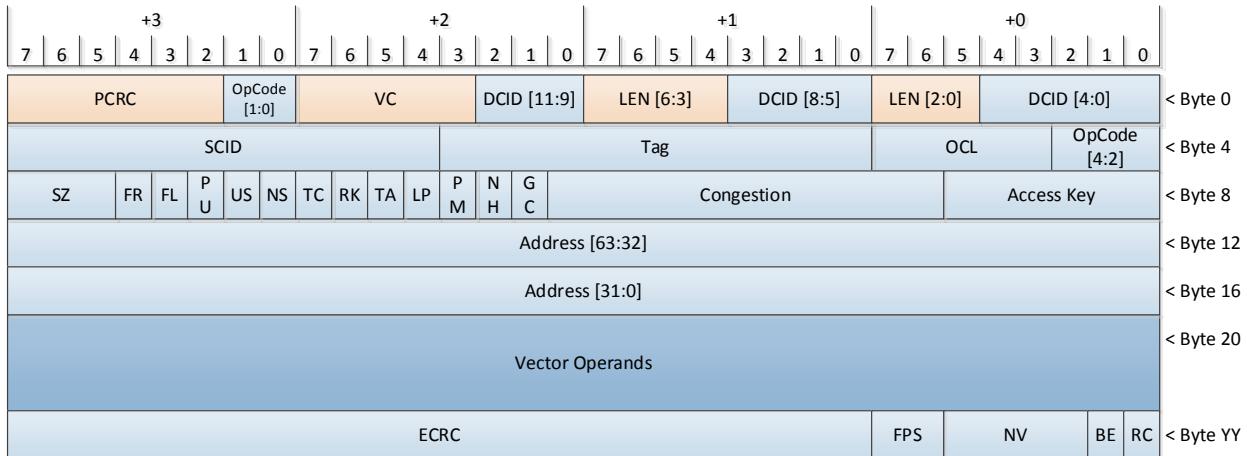


Figure 7-20: Explicit Atomic 1 Vector Packet Format

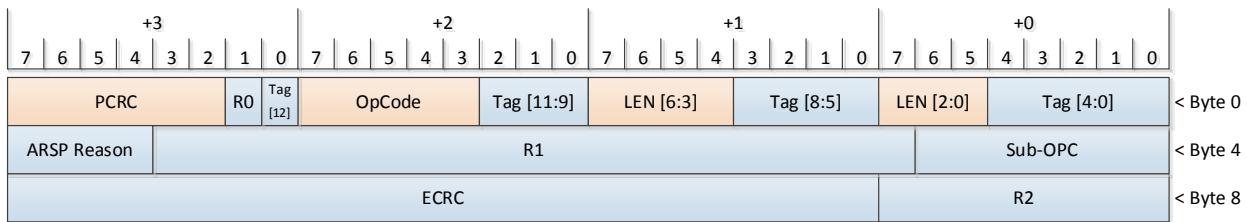


Figure 7-21: P2P-Core Atomic Response Packet Format

Table 7-18: P2P-Core Atomic Response Packet-specific Fields

Field Name	Size (bits)	Description
R0	1	Reserved
R1	21	Reserved
R2	8	Reserved

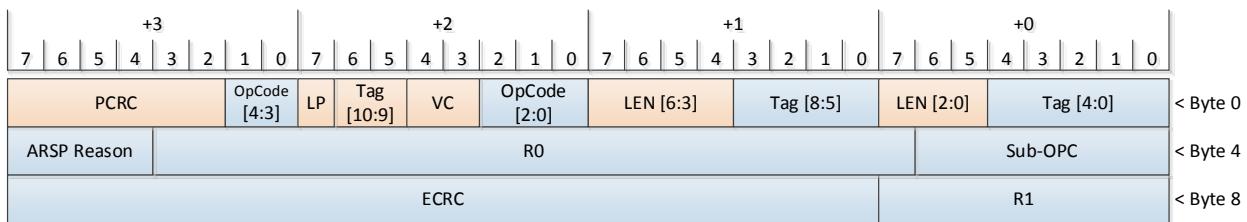


Figure 7-22: P2P-Coherency Atomic Response Packet Format

Table 7-19: P2P-Coherency Atomic Response Packet-specific Fields

Field Name	Size (bits)	Description
R0	21	Reserved
R1	8	Reserved

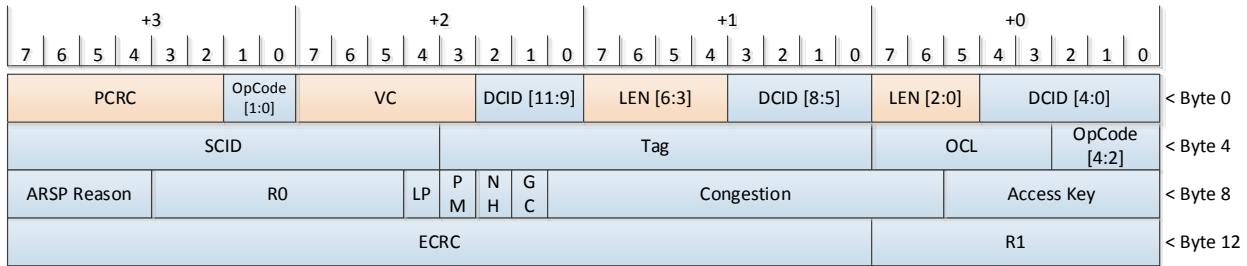


Figure 7-23: Explicit Atomic 1 Response Packet Format

Table 7-20: Explicit Atomic 1 Response Packet-specific Fields

Field Name	Size (bits)	Description
R0	7	Reserved
R1	8	Reserved

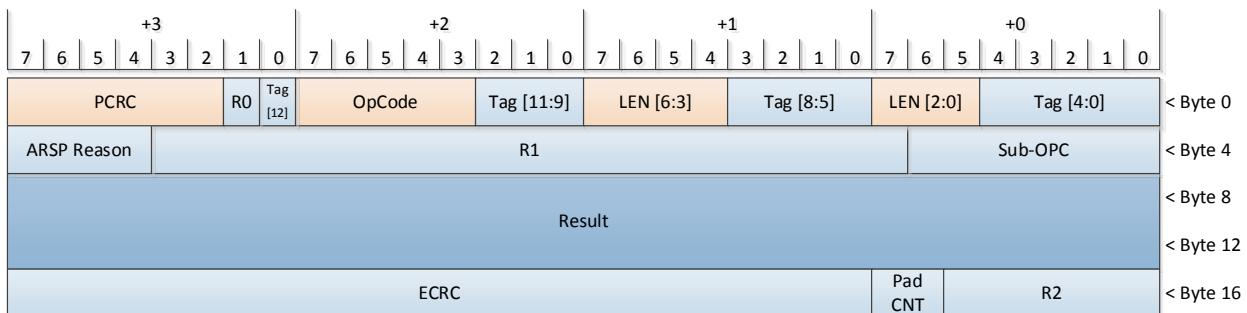


Figure 7-24: P2P-Core Atomic Results Response Packet Format

Table 7-21: P2P-Core Atomic Results Response Packet-specific Fields

Field Name	Size (bits)	Description
R0	1	Reserved
R1	21	Reserved
R2	6	Reserved

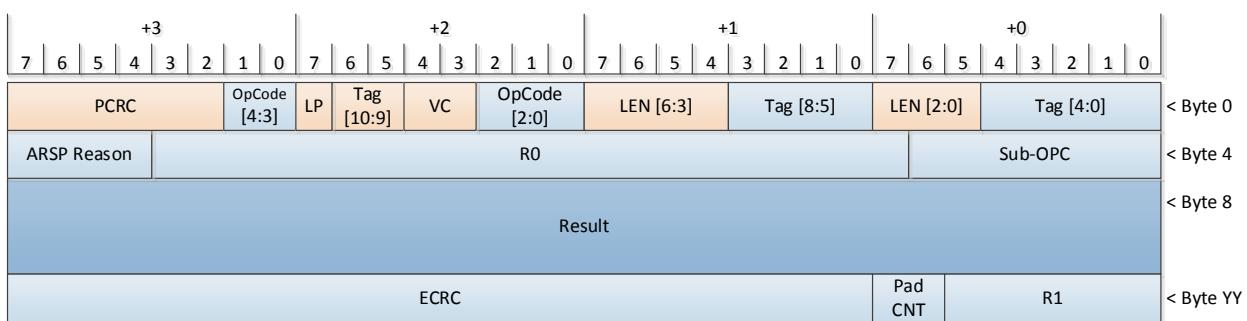


Figure 7-25: P2P-Coherency Atomic Results Response Packet Format

Table 7-22: P2P-Coherency Atomic Results Response Packet-specific Fields

Field Name	Size (bits)	Description																							
R0	21	Reserved																							
R1	6	Reserved																							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		
PCRC			OpCode [1:0]		VC			DCID [11:9]			LEN [6:3]		DCID [8:5]			LEN [2:0]		DCID [4:0]				< Byte 0			
SCID						Tag						OCL			OpCode [4:2]		< Byte 4								
ARSP Reason		R0			LP	P M	N H	G C	Congestion						Access Key				< Byte 8						
Result												R2		Pad CNT	R1		< Byte 12								
ECRC												R2		Pad CNT	R1		< Byte 16								
														Pad CNT			< Byte 20								

Figure 7-26: Explicit Atomic 1 Results Response Packet Format

Table 7-23: Explicit Atomic Results 1 Response Packet-specific Fields

Field Name	Size (bits)	Description
R0	7	Reserved
R1	4	Reserved
R2	2	Reserved

### 7.3. Buffer Requests

5 Buffer requests enable large data movement between two buffers—Buffer A and Buffer B—or between two buffer vectors—vector A and vector B. Buffer data movements are referred to as *Put* and *Get*. A *Put* request writes Buffer A to Buffer B, and a *Get* request reads from buffer B to Buffer A. Buffers may be located in separate components, Requester co-located, or Responder co-located as illustrated in the following figures.

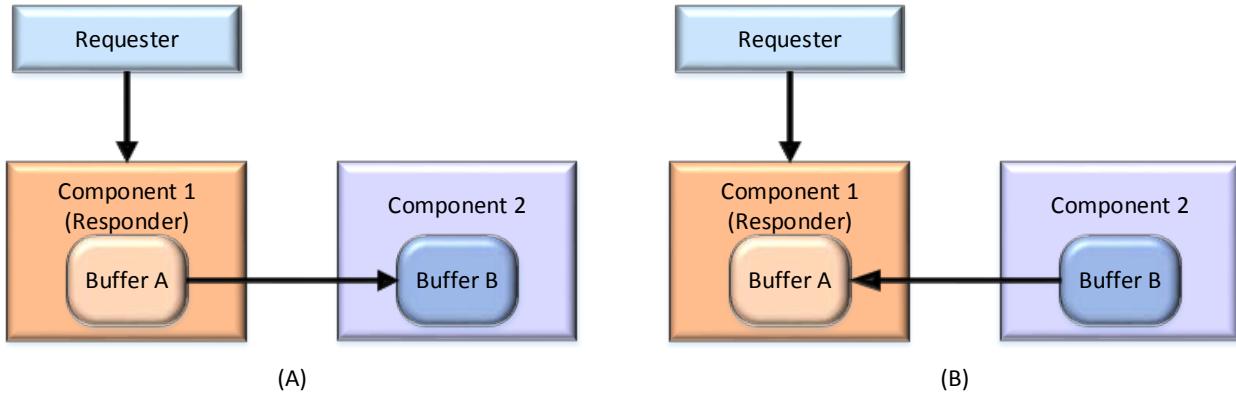


Figure 7-27: Buffer Request between Two Components

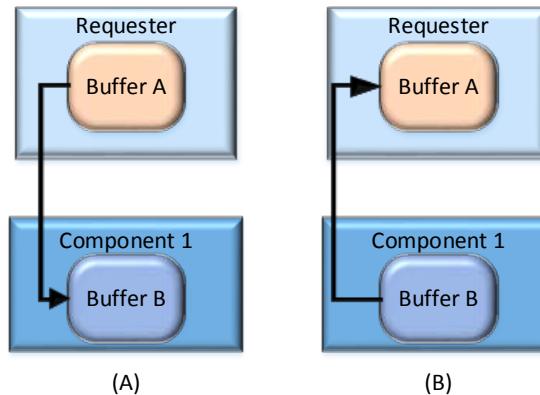
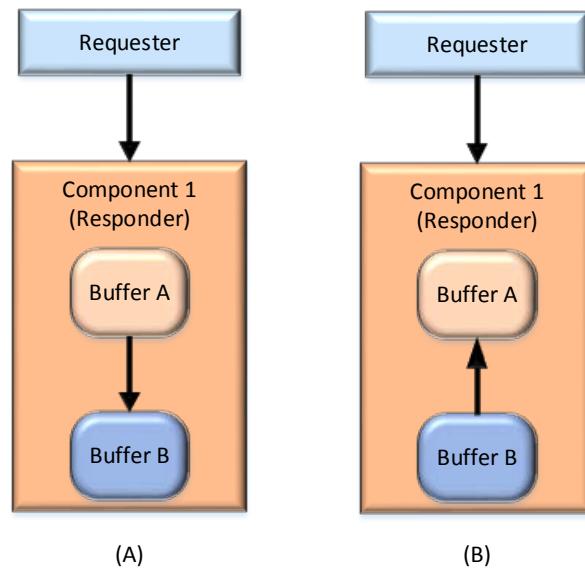


Figure 7-28: Buffer Request between Requester-Local and Remote Buffers



5

Figure 7-29: Buffer Request within a Single Component

Vector requests enable a single Buffer request to *Put* or *Get* multiple buffers between two components as illustrated in *Buffer Vector Request between Two Components*. As with the single Buffer *Put* or *Get*, buffers may be Buffers may be located in separate components, Requester co-located, Responder co-located. Vector requests may also be used to scatter the contents of one

10

buffer across multiple buffers or to gather multiple buffers into one buffer as illustrated in *Buffer Vector Operation Scatter / Gather between Two Components*.

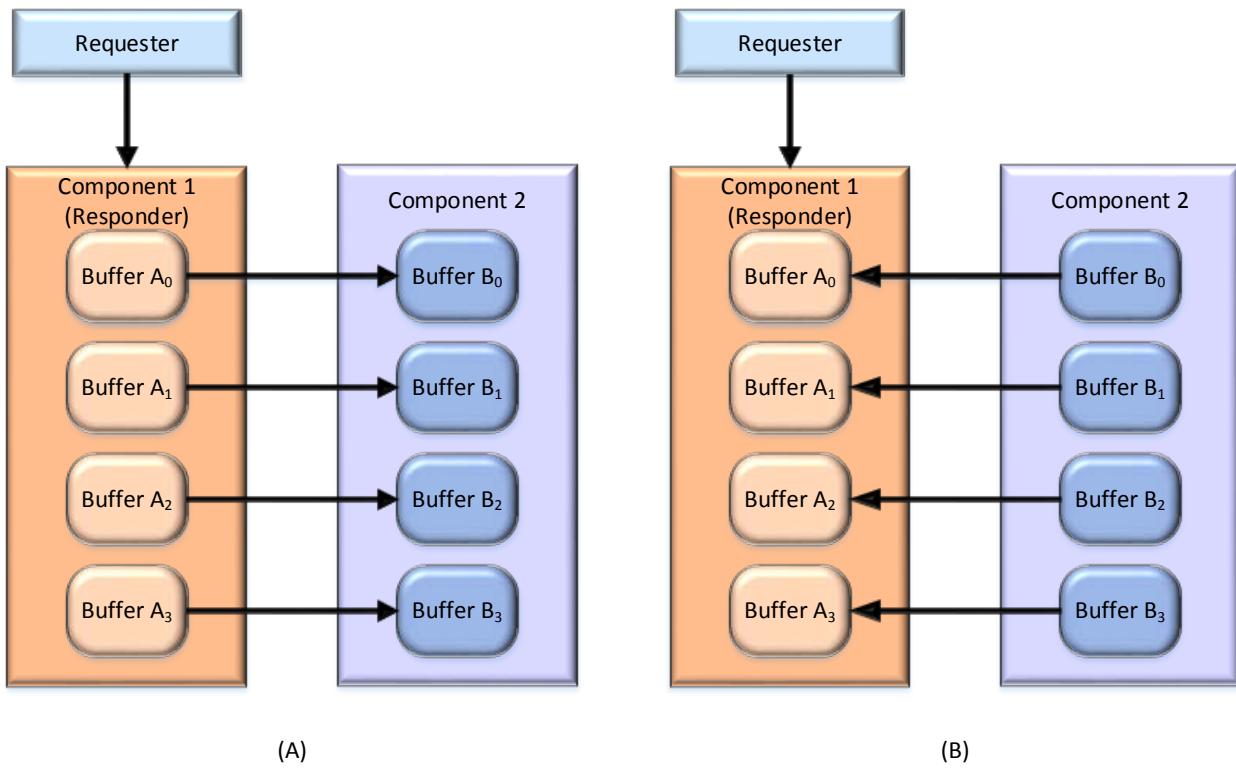
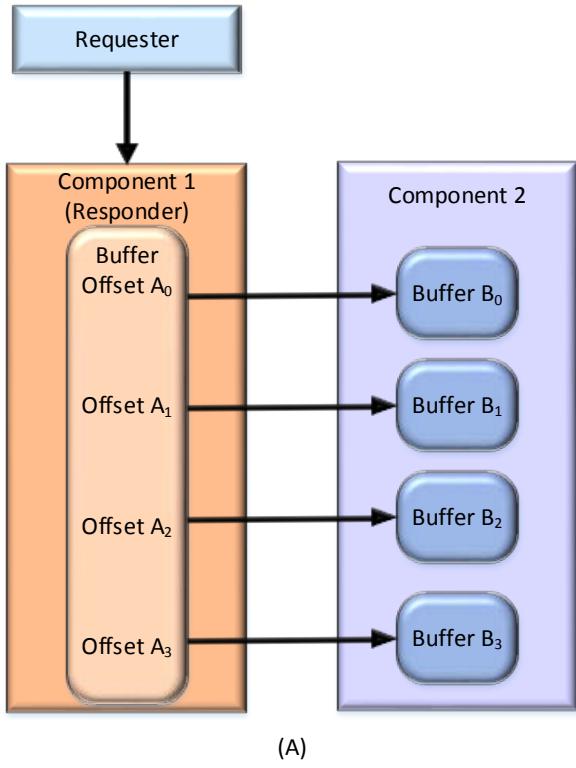
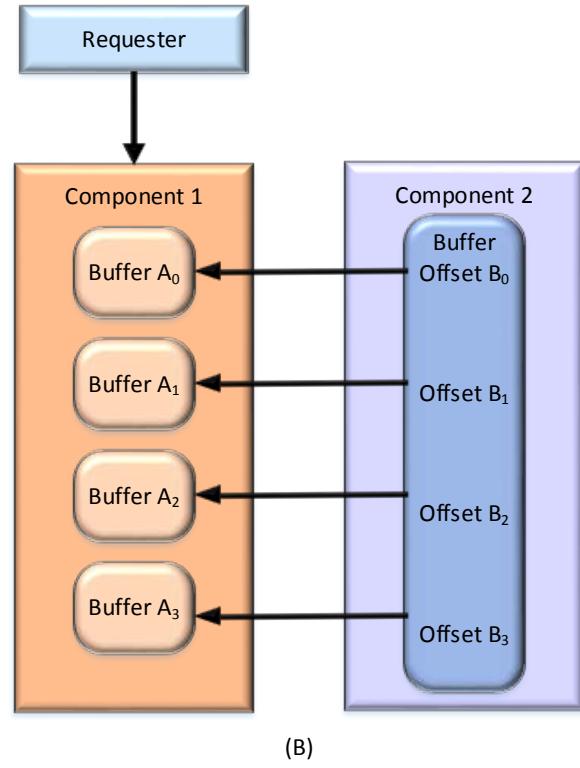


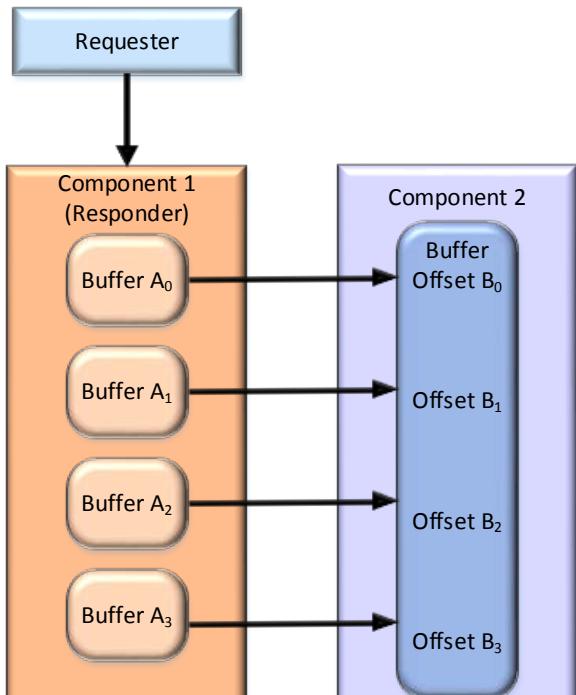
Figure 7-30: Buffer Vector Request between Two Components



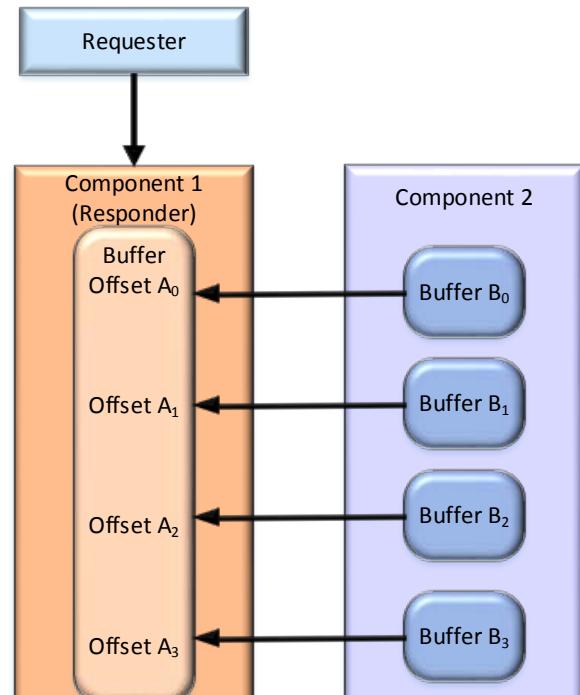
(A)



(B)



(C)



(D)

Figure 7-31: Buffer Vector Operation Scatter / Gather between Two Components

A Buffer request between multiple components is translated into a series of read or write request packets. *Example Buffer Put Request as a Series of Writes* illustrates how a Buffer Put is translated into N write request packets targeting a contiguous buffer, and *Example Buffer Get Request as a Series of Reads* illustrates how a Buffer Get is translated into X read request packets that are completed by Y Read Response packets. Buffer Putv and Getv operate similarly but target multiple buffers. Read and write request packets executed on behalf of a Buffer request are indistinguishable on the wire from standard read and write request packets. As a result, only the Buffer Requester and the Buffer Responder need to comprehend the buffer request; Component B need only comprehend the underlying read and write request packets.

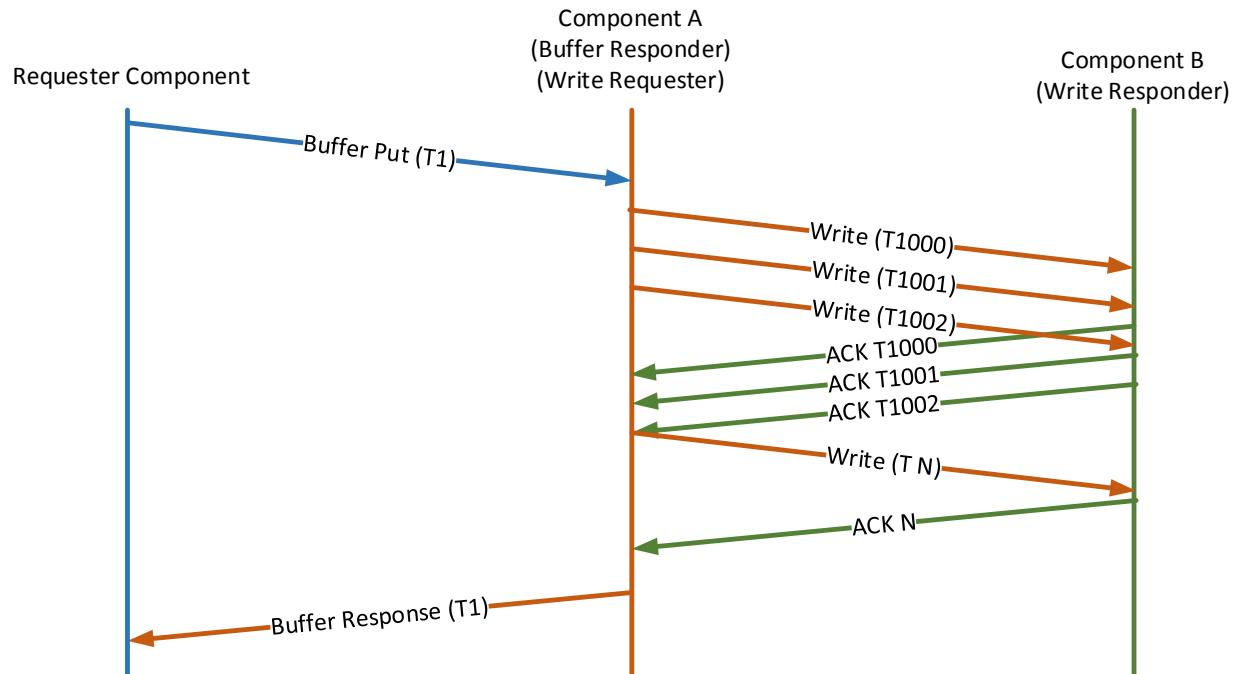


Figure 7-32: Example Buffer Put Request as a Series of Writes

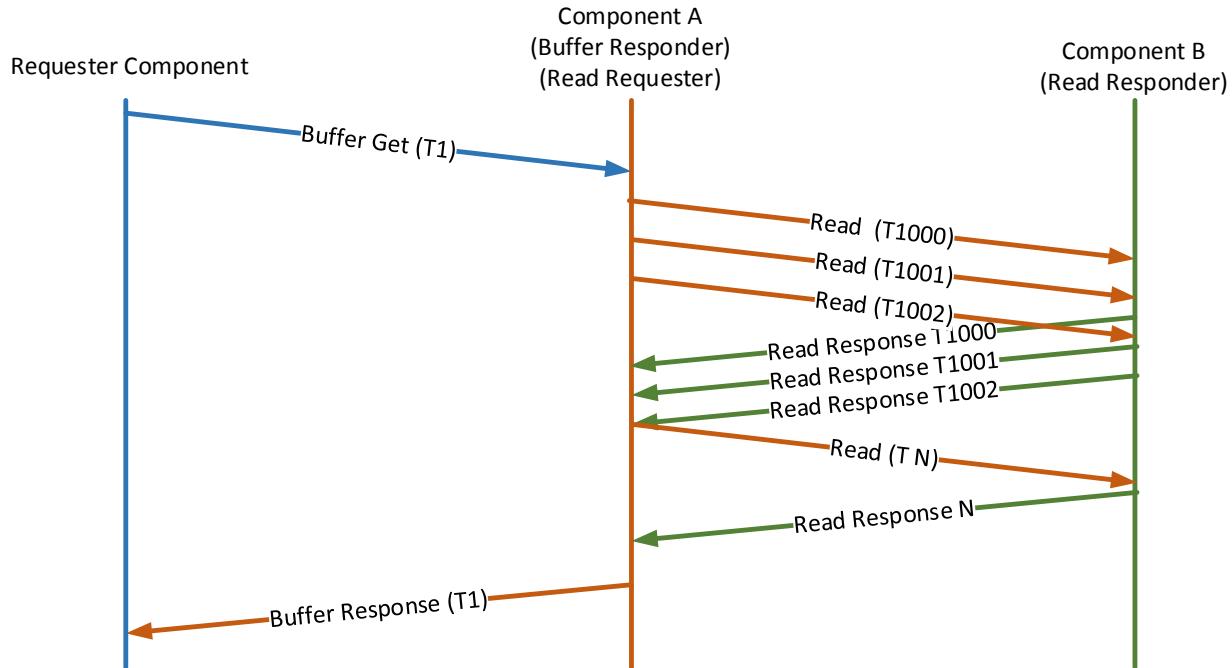


Figure 7-33: Example Buffer Get Request as a Series of Reads

To simplify coordination between cooperating applications, components may support Signaled Buffer Put and Get Requests. For example, instead of following the successful execution of a Buffer Request by a series of application-specific completion messages to cooperating applications, a Signaled Buffer Request atomically increments by one a specified memory location (counter) co-located with the destination buffer (Buffer B if a Put and Buffer A if a Get). Cooperating applications read the atomic counter to detect buffer movement completion. *Example Signaled Buffer Put (A to B) with a Signaled Write* illustrates an example Signaled Buffer Put ladder diagram. The exchange is nearly identical to a Buffer Put with one exception. The last Write uses a Multi-Op Signaled Write request packet to write the last unit of data and increment the specified counter location. *Example Signaled Buffer Get (B to A)* illustrates an example Signaled Buffer Get ladder diagram. The exchange is nearly identical to a Buffer Get with one exception. Once the last Read is successfully completed, component A atomically increments the value (CNT) at the component-local address.

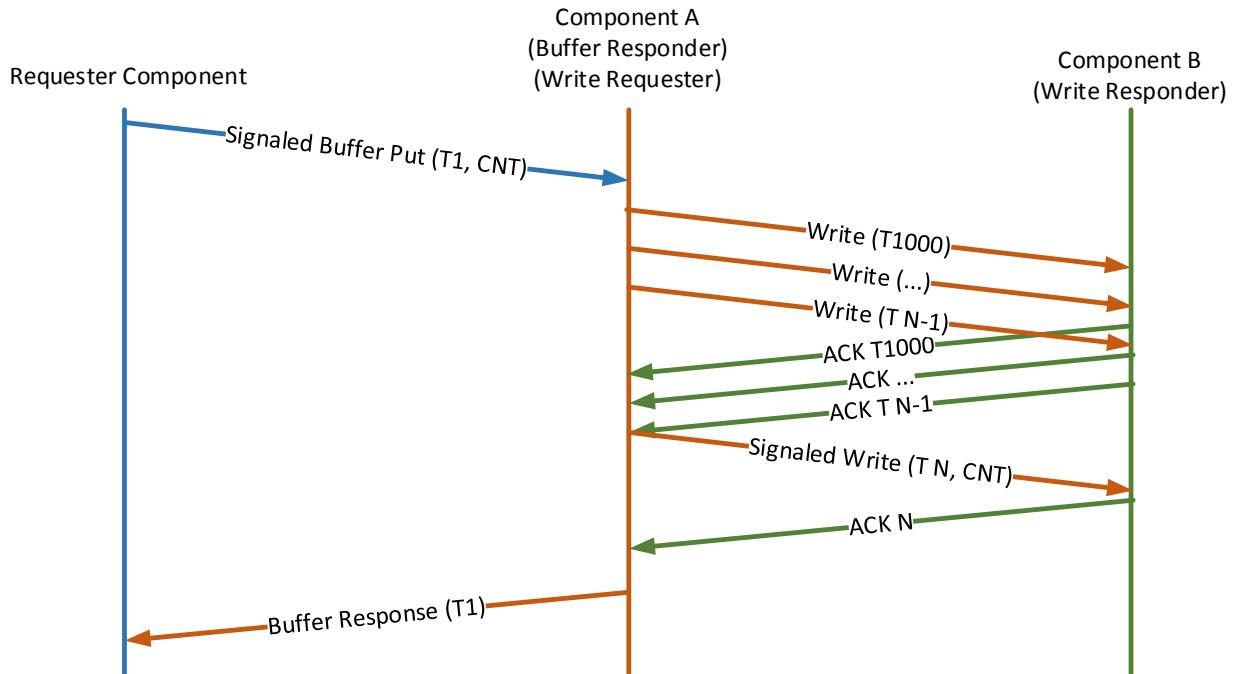


Figure 7-34: Example Signaled Buffer Put (A to B) with a Signaled Write

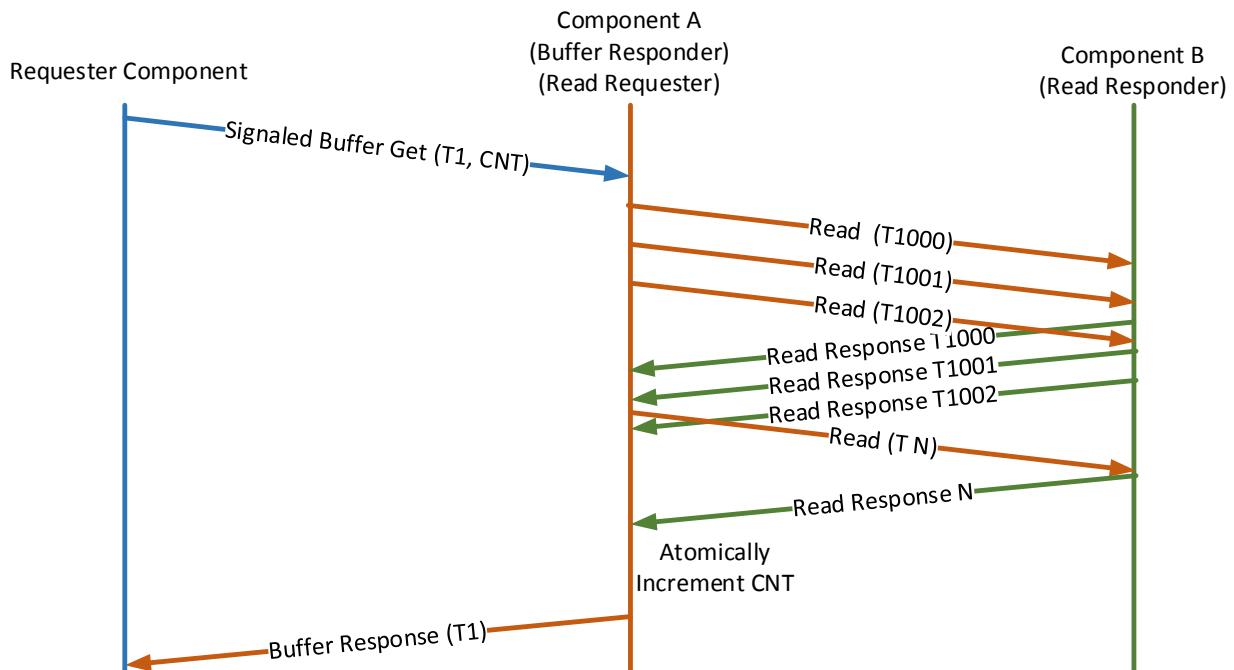


Figure 7-35: Example Signaled Buffer Get (B to A)

- 5 Buffer Put and Get data movement requires cooperating components to allocate and advertise buffers as a prerequisite. For applications that allocate and advertise buffers at a relatively low-rate, the start-up and coordination costs are amortized over multiple data access requests. However, for those applications that operate at a very high buffer allocation rate with limited subsequent data access, these costs can be untenable. To address these costs, the architecture supports Dynamic Buffer Allocation, (Signaled) Dynamic Buffer Put, and (Signaled) Dynamic Buffer Get.
- 10

5 A Dynamic Buffer Allocation Request targets a destination component. The destination component may be a single, physical or logical (Transparent Router) Responder component. Through means outside of this specification's scope, the Responder allocates a buffer of the requested length and returns the component-relative address (zero-based or virtual address). Conceptually, the Responder takes the steps illustrated in *Example Dynamic Buffer Allocation Exchange*.

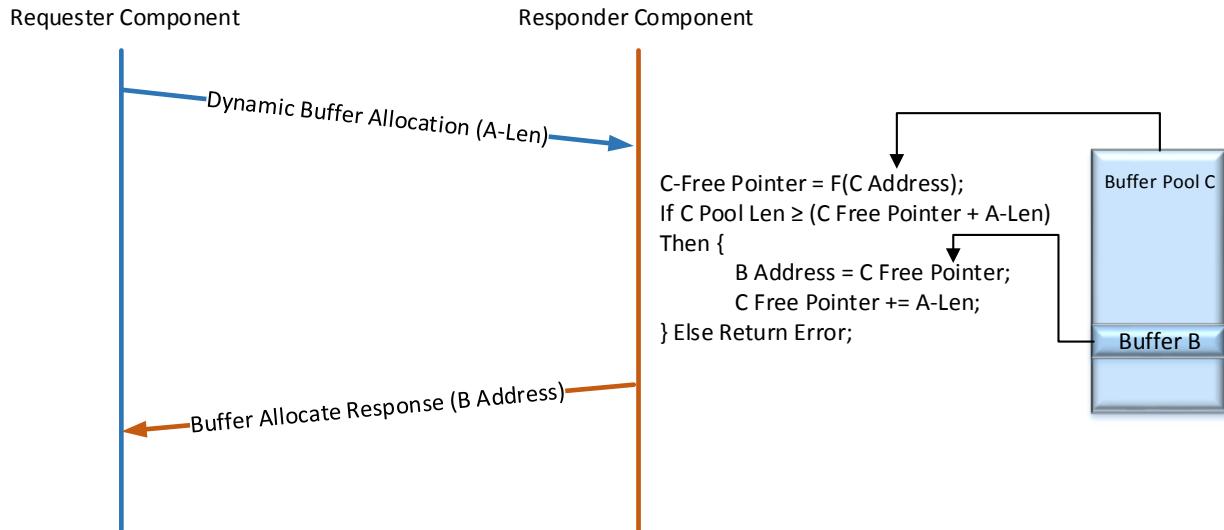


Figure 7-36: Example Dynamic Buffer Allocation Exchange

A Dynamic Buffer Release Request returns a previously dynamically-allocated buffer.

10 A Dynamic Buffer Put Request is a combination of a Dynamic Buffer Allocation exchange and a series of 1-N write request packets to copy the data to the dynamically allocated buffer B as illustrated in *Example Dynamic Put Exchange*. A Signaled Dynamic Buffer Put Request performs the same steps with one exception; the last write N is a Signaled Write (N, Counter).

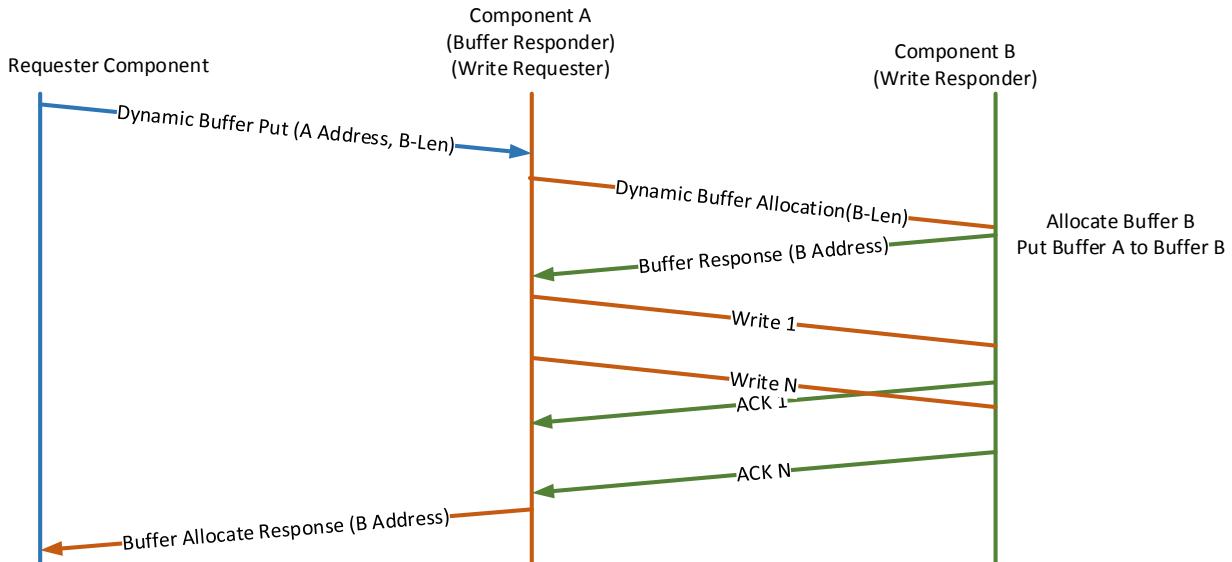


Figure 7-37: Example Dynamic Put Exchange

15

A Dynamic Buffer Get Request also performs buffer allocation, but does not use the Dynamic Buffer Allocation Request as the data destination performs a series of Reads instead of Writes to copy the data from buffer A to the allocated buffer B (see *Example Dynamic Buffer Get Exchange*). The Signaled Dynamic Buffer Get Request performs the same steps except in increments the specified co-located counter upon completion of the last Read Request.

5

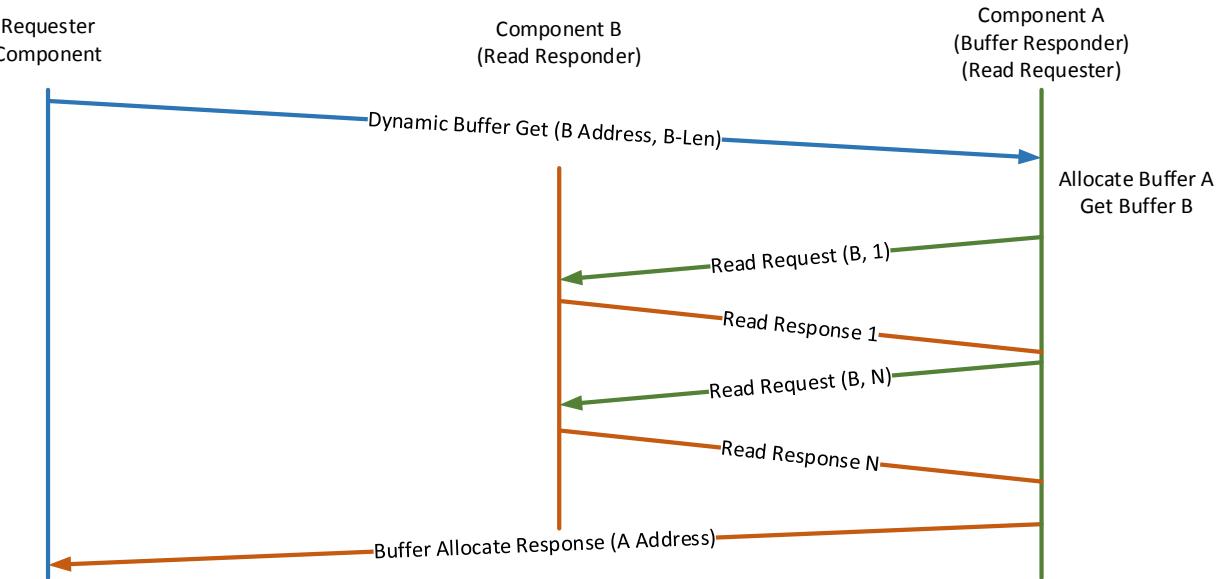


Figure 7-38: Example Dynamic Buffer Get Exchange

### 7.3.1. Buffer Features & Requirements

The following are the features and requirements associated with Buffer requests:

- 10
- Any component type may support Buffer request packets as a Requester, a Responder, or as both a Requester and a Responder.
  - A component may support all or a subset of the *Buffer Request Types*.
  - Buffer request and response packets shall use the following packet formats:
    - *P2P-Core Buffer Put Local Request Packet Format*
    - *(Dynamic) Buffer Put / Get Request Packet Format*
    - *Signaled (Dynamic) Buffer Put / Get Request Packet Format*
    - *Dynamic Buffer Allocate Request Packet Format*
    - *Dynamic Buffer Release Request Packet Format*
    - *Buffer Putv / Getv Request Packet Formats*
    - *Buffer Allocate Response Packet Format*
  - LDM 1 Buffer request-specific requirements:
    - The address of Buffer A shall be the location of the data buffer on component A. If component A supports zero-based addressing (*Core Structure Component CAP 1 Address Field Interpretation == 0b*), then the address represents the offset from byte 0 of component A's Data Space. If component A supports non-zero-based addressing (*Core Structure Component CAP 1 Address Field Interpretation == 1b*), then the address is one comprehended by component A.
    - The address of Buffer B shall be the location of the data buffer on component B. If component B supports zero-based addressing, then the address represents the offset

15

20

25

from byte 0 of component B's Data Space. If component B supports non-zero-based addressing, then the address is one comprehended by component B.

- A (Signaled) Buffer Put / Get operates on a single source buffer and a single destination buffer.
- A LDM 1 Buffer Putv / Getv Request operates on up to four source buffers and up to four destination buffers. Source and destination buffers may each represent a single contiguous buffer or up to four distinct buffers. This format enables scatter and gather buffer movements.
- All versions of LDM 1 Buffer Put / Putv / Get / Getv request packets may involve three components—the Requester, the Responder which contains Buffer A / Buffer Vector A and processes the Buffer Request and returns the corresponding acknowledgment, and the component that contains Buffer B / Buffer Vector B.
  - The Requester and the Responder shall support the LDM 1 OpClass and a common set of Buffer OpCodes.
  - Component B, the component containing Buffer B, may support the LDM 1 OpClass. If component A supports and uses the Signaled Buffer Put request packet, then Components A and B shall support the Signaled Write Request (see *Multi-Op Requests*).
  - The Responder shall translate any version of Buffer Get request packets into a series of Core 64 Read or LDM 1 Read request packets.
  - The Responder shall translate any version of Buffer Put request packets into a series of Core 64 Write request packets.
  - The Responder shall translate the Buffer request's No Snoop and Target Cache (if applicable) fields into the appropriate Core 64 Read, LDM 1 Read, or Core 64 Write request packet respective fields.
- If a LDM 1 Buffer request packet fails validation or execution at the Responder (the component that received the Buffer request packet), then the Responder shall transmit a Core 64 OpClass *Standalone Acknowledgment* with the indicated error.
- A Core 64 *Standalone Acknowledgment* shall be transmitted in response to any of the following LDM 1 Buffer request packets that was successfully executed:
  - Buffer Put
  - Buffer Get
  - Buffer Putv
  - Buffer Getv
  - Signaled Buffer Put
  - Signaled Buffer Get
  - Dynamic Buffer Release
- A Buffer Allocate Response shall be transmitted in response to any of the following LDM 1 Buffer request packets that was successfully executed:
  - Dynamic Buffer Put
  - Dynamic Buffer Get
  - Signaled Dynamic Buffer Put
  - Signaled Dynamic Buffer Get
  - Dynamic Buffer Allocate
- The Buffer request packet's Access Key field shall be copied to the read and write packets used to move the data between buffers.
- The maximum number of outstanding Buffer Requests supported by the Responder is indicated via the *Max\_Buffer\_Ops* field within the *OpCode Set Structure*. If a Responder

receives more Buffer requests (of any type) than supported by the Responder, then the Responder may silently discard the new Buffer request packet or return a *Responder Not Ready (RNR)* NAK.

- Buffer request packets are not idempotent. If a Buffer request is retransmitted, then the Responder shall take the actions specified in *Non-idempotent Request (NIR)*.
- Buffer request packets have non-deterministic execution times due to their inherent variable nature. To ensure Reliable Delivery, Buffer request-response packet exchange shall comply with *Non-Deterministic Request Execution*.
- If Buffer request packet execution fails and the request involves three components, then the Responder shall return a *Standalone Acknowledgment* with the highest precedence error detected recorded in the Reason field.
- Components A and B may support *Region Key (R-Key)* configuration and validation.
  - If R-Key validation is enabled, then the Buffer request packet shall contain the BA R-Key and BB R-Key fields as illustrated in *Buffer Request with R-Key and / or Global DCID B Field*.
  - If the Buffer request is a Put, then the BA R-Key field shall contain the RO R-Key or the RW R-Key associated with buffer A, and the BB R-Key field shall contain the RW R-Key associated with buffer B.
  - If the Buffer request is a Get, then the BA R-Key field shall contain the RW R-Key associated with buffer A, and the BB R-Key field shall contain the RO R-Key or the RW R-Key associated with buffer B.
  - If the BA R-Key and BB R-Key fields contain the Default R-Key, then the read and write request packets used to exchange data shall not include an R-Key field.
  - Prior to transmitting a buffer request packet, using the Requester's *Component RKD Structure*, the Requester shall validate the BA R-Key, BB R-Key, and the SA R-Key (if applicable)

### 7.3.2. Buffer Request Types

The following buffer request types and associated operational semantics are specified as follows.

- Buffer Put Local copies Buffer Length bytes from Buffer A to Buffer B. This request is executed within a single component using the P2P-Core OpClass.
  - The underlying execution of a Buffer Put Local is outside of this specification's scope.
  - A Buffer Put Local shall be acknowledged by a P2P-Core *Standalone Acknowledgment*.
    - Since P2P-Core links use *Link-level Reliability (LLR)*, the Responder shall not transmit a separate non-deterministic *Standalone Acknowledgment*. A single *Standalone Acknowledgment* shall be transmitted to indicate a success or the highest precedence error.
    - The Requester shall use an implementation-specific retransmission timer to ensure operation reliability.
- Buffer Put copies Buffer Length bytes from Buffer A to Buffer B.
  - The underlying execution of a component-local Buffer Put is outside of this specification's scope.
  - The execution of a non-component-local Buffer Put is as follows:
    - A Buffer Put request is translated into a set of N Core 64 Write request packets.
      - Write request packets may be any size supported by both components.

- Each write request packet shall set the DCID = Buffer Request DCID B field. If the Buffer request packet included a DSID field, then the Write request packet shall set GC = 1b and the DSID = Buffer Request DSID B.
- Component A assigns a unique Tag to each write request packet since it acts as the Requester for these request packets.
- The contents of Buffer A may be written to Buffer B in any order.
- The Buffer Responder shall be responsible for tracking when all write request packets have been successfully completed and for surfacing any errors encountered.
- A Signaled Buffer Put copies Buffer Length bytes from Buffer A to Buffer B, and, if successful, then atomically increments the data value at the Signaled Address (co-located with Buffer B).
  - The underlying execution of a component-local Buffer Put and the atomic increment of the data value are outside of this specification's scope.
  - The execution of a non-component-local Signaled Buffer Put is as follows:
    - A Buffer Put request is translated into a set of N-1 Core 64 Write request packets. These write request packets shall follow the same rules as the write request packets for a Buffer Put request packet.
    - The Buffer Responder shall be responsible for tracking when all N-1 Write request packets have successfully completed and for surfacing any encountered errors.
    - Once all N-1 Core Write request packets are successfully completed, the Buffer Responder shall issue a Core 64 Multi-Op Signaled Write request packet containing the last bytes of data. It shall set the Address 2 field of the Multi-Op Signaled Write request packet to the Buffer Put Request's Signaled Address field.
    - The Buffer Responder shall return a *Standalone Acknowledgment* indicating the success or failure of the Signaled Buffer Put request packet.
- Buffer Get copies Buffer Length bytes from Buffer B to Buffer A.
  - The underlying execution of a component-local Buffer Get is outside of this specification's scope.
  - The execution of a non-component-local Buffer Get is as follows:
    - A Buffer Get request packet is translated into a set of X Core 64 Read request packets or Y LDM 1 Read request packets.
      - Read request packets may be any size supported by both components.
      - Each read request packet sets the DCID = Buffer Request DCID B field. If the Buffer request packet included a DSID field, then the read request packet shall set GC = 1b and the DSID = Buffer Request DSID B.
      - Component A assigns a unique Tag to each read request packet since it acts as the Requester.
    - The contents of Buffer B may be read in any order.
    - The Buffer Responder shall be responsible for tracking when all Read Response packets have been successfully received and for surfacing any encountered errors.
- Signaled Buffer Get copies Buffer Length bytes from Buffer B to Buffer A, and, if successful, then the Buffer Responder atomically increments the data value at the Signaled Address (co-located with Buffer A) by one.
  - The underlying execution of a component-local Buffer Get and the atomic increment of the data value are outside of this specification's scope.

- The execution of a non-component-local Signaled Buffer Get request packet is identical to a Buffer Get with one additional step.
  - Once all Read Response packets have been successfully executed, the Buffer Responder shall atomically increment the data value located at Signaled Address and shall store the result at Signaled Address.
- The data is an unsigned 64-bit integer located either at Signaled Address.
- The Buffer Responder shall return a *Standalone Acknowledgment* indicating the success or failure of the Signaled Buffer Get request.
- Buffer Putv copies up to four buffers of Buffer Length<sub>0-3</sub> from A<sub>0-3</sub> to B<sub>0-3</sub>.
  - A Putv is used to move up to four separate buffers from vector A to the four separate buffers represented by vector B. Vector A may also represent up to four addresses within a single contiguous buffer enabling a single buffer to be scattered to vector B. Similarly, Vector A may represent up to four buffers that target a single contiguous buffer enabling the buffers to be gathered into vector B.
  - The underlying execution of a component-local Buffer Putv is outside of this specification's scope.
  - The execution of a non-component-local Buffer Putv is as follows:
    - A Buffer Putv request packet is translated into a set of N Core 64 Write request packets.
      - Write request packets may be any size supported by both components.
      - Each write request packet sets the DCID = Buffer Request DCID B field. If the Buffer request packet included a DSID field, then the Write request packet shall set GC = 1b and the DSID = Buffer Request DSID B.
      - Component A assigns a unique Tag to each write request packet since it acts as the Requester.
    - The contents of Buffers A<sub>0-3</sub> may be respectively written to Buffer B<sub>0-3</sub> in any order. Buffer Length<sub>0</sub> bytes from A<sub>0</sub> are written to B<sub>0</sub>; Buffer Length<sub>1</sub> bytes from A<sub>1</sub> to B<sub>1</sub>; and so forth.
    - Vectors shall be packed, e.g., if there are only two vectors then
      - Buffer Length<sub>0</sub>, A<sub>0</sub>, and B<sub>0</sub> contain non-zero / non-Null values.
      - Buffer Length<sub>1</sub>, A<sub>1</sub>, and B<sub>1</sub> contain non-zero / non-Null values.
      - Buffer Length<sub>2</sub> = Buffer Length<sub>3</sub> = 0.
    - Unused vectors shall have the corresponding Buffer Length field set to zero.
    - The Responder shall be responsible for tracking when all write request packets have successfully completed and for surfacing any errors encountered.
- Buffer Getv copies Buffer Length bytes from Buffer B to Buffer A.
  - A Getv is used to move up to four separate buffers from vector B to the four separate buffers represented by vector A. Vector A may also represent four addresses within a single contiguous buffer enabling up to four buffers represented by vector B to be gathered into a single contiguous buffer. Similarly, Vector B may represent four addresses within a single contiguous buffer enabling the buffer to be scattered across vector A.
  - The underlying execution of a component-local Buffer Getv is outside of this specification's scope.
  - The execution of a non-component-local Buffer Getv is as follows:
    - A Buffer Getv request packet is translated into a set of X Core 64 Read or Y LDM 1 Read request packets.

- Each read request packet sets the DCID = Buffer Request DCID B field. If the Buffer request packet included a DSID field, then the read request packet shall set GC = 1b and the DSID = Buffer Request DSID B.
  - Component A assigns a unique Tag to each read request packet since it acts as the Requester for these requests.
  - The contents of Buffers  $B_{0-3}$  may be read in any order. Buffer Length<sub>0</sub> bytes from  $B_0$  are placed in  $A_0$ ;  $B_1$  placed in  $A_1$ ; and so forth.
  - Vectors shall be packed, e.g., if there are only two vectors then
    - Buffer Length<sub>0</sub>,  $A_0$ , and  $B_0$  contain non-zero / non-Null values.
    - Buffer Length<sub>1</sub>,  $A_1$ , and  $B_1$  contain non-zero / non-Null values.
    - Buffer Length<sub>2</sub> = Buffer Length<sub>3</sub> = 0.
  - Unused vectors shall have the corresponding Buffer Length field set to zero.
  - The Responder shall be responsible for tracking when all Read Response packets have been successfully received and for surfacing any encountered errors.
- 5
- Dynamic Buffer Allocate allocates a buffer from the destination component and returns the address of the allocated buffer to the Requester.
    - Through component-specific methods, the Responder locates available buffer space of at least Allocation Length and allocates Allocation Length bytes. The allocation shall be an integer 16-byte multiple.
      - A Responder may allocate a buffer that is larger than requested, e.g., rounded up to the next closest integer page-size multiple. The actual length is returned in the response packet.
      - A Requester shall not make any assumptions about a Responder's present available space based on its prior allocations.
    - The Responder shall return the allocated address of byte zero of the newly allocated buffer using a Buffer Allocate Response packet.
    - If an allocation fails due to insufficient resources, then Response shall return a *Standalone Acknowledgment* (Reason = Insufficient Space).
    - If a component supports Dynamic Buffer Allocate, then it should support Dynamic Buffer Release.
- 10
- 15
- Dynamic Buffer Release returns a buffer to the destination component.
    - If a component supports Dynamic Buffer Release, then it shall support Dynamic Buffer Allocate.
    - The buffer address shall be the Allocated Address returned in the corresponding Dynamic Buffer Allocate.
      - The Allocated Length shall be the length returned from buffer allocation.
    - Through component-specific methods, the Responder returns the buffer to its free space for subsequent allocation.
- 20
- 25
- 30
- 35
- Dynamic Buffer Put copies Buffer A to a dynamically allocated buffer from a designated buffer pool using *Put* semantics.
    - The Buffer Responder generates a Dynamic Buffer Allocate request packet to the destination component containing the designated buffer pool.
    - Upon successful completion, the Buffer Responder uses the newly allocated buffer address as the destination for a series of write request packets to copy the data from Buffer A.
- 40
- 45

- A Signaled Dynamic Buffer Put transmits N-1 write request packets and then issues a Signaled Write Request N to place the last unit of data and then update the designated counter.
- Dynamic Buffer Get copies Buffer A to a dynamically allocated buffer from a designated buffer pool using *Get* semantics.
  - The Buffer Responder allocates a buffer from the designated component-local buffer pool.
  - Upon successful completion, the Buffer Responder issues a series of read request packets to copy the data from Buffer B to the newly allocated Buffer A.
    - Once all Read Response packets have been successfully executed, if a Signaled Dynamic Buffer Get, the Buffer Responder shall atomically increment the data value located at Signaled Address and shall store the result at Signaled Address.

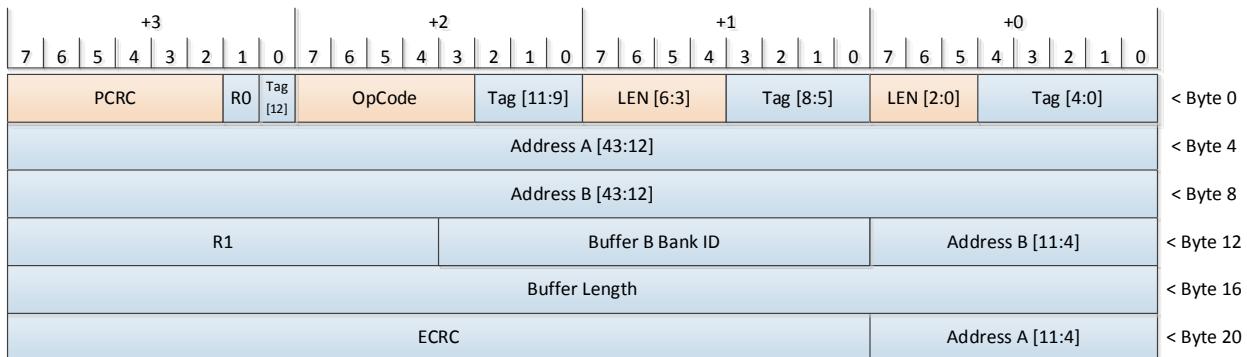


Figure 7-39: P2P-Core Buffer Put Local Request Packet Format

Table 7-24: P2P-Core Buffer Put Local Packet-specific Fields

Field Name	Size (bits)	Description
Buffer A Address	44	Address of buffer A—This is an offset within the logical bank encoded within the Tag field.
Buffer B Address	44	Address of buffer B—This is an offset within the logical bank identified by the Buffer B Bank ID field.
Buffer B Bank ID	12	Logical Bank identifier associated with buffer B.
Buffer Length	32	Length of Buffer A and Buffer B in bytes. If zero, then the actual length is $2^{32}$ bytes.
R0	1	Reserved
R1	12	Reserved

15

If a Buffer request packet has the GC == 1b, then the following additional fields shall be present (DSID B, and Reserved) in the following Buffer request packet formats as illustrated in *Buffer Request with R-Key and / or Global DCID B Field*:

- *(Dynamic) Buffer Put / Get Request Packet Format*

- *Signaled (Dynamic) Buffer Put / Get Request Packet Format*
- *Buffer Putv / Getv Request Packet Formats*

If RK == 0b, then the fields shall be placed starting at byte 16, else they shall be placed starting at byte 20.

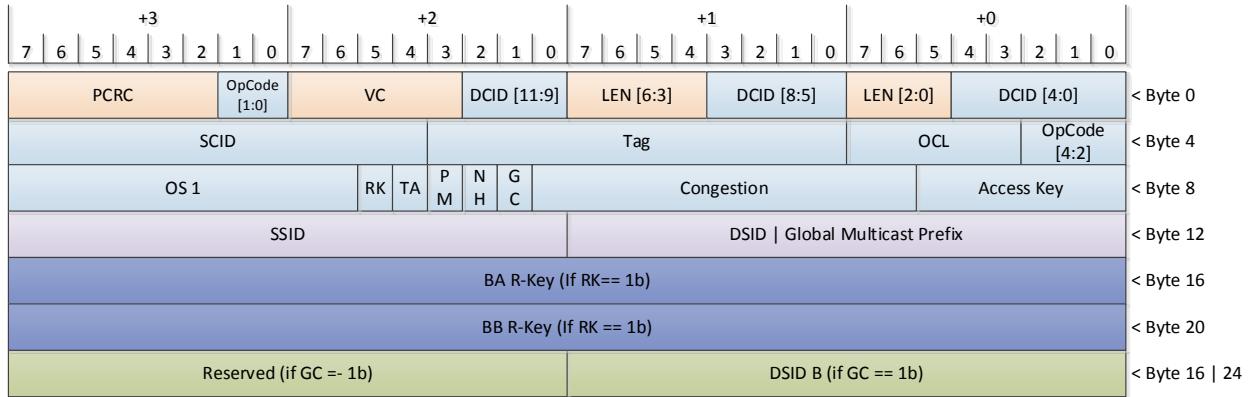


Figure 7-40: Buffer Request with R-Key and / or Global DCID B Field

Table 7-25: Buffer Request with Global DCID B Fields

Field Name	Size (bits)	Description
BA R-Key	32	<i>Region Key (R-Key) associated with buffer A</i>
BB R-Key	32	<i>Region Key (R-Key) associated with buffer B</i>
DSID B	16	This field identifies the SID of component B
Reserved	16	Reserved field

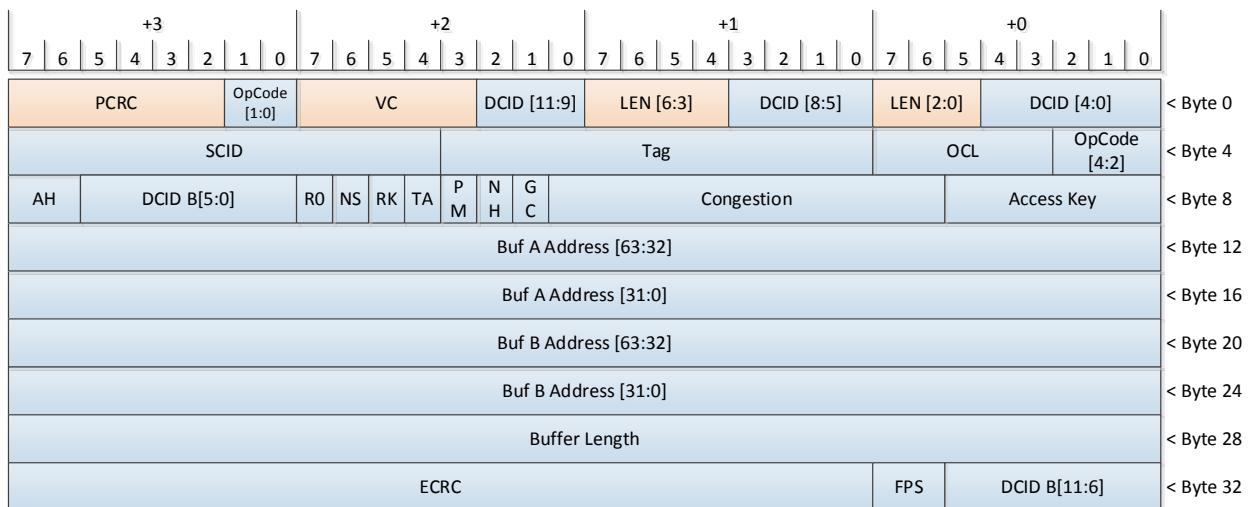


Figure 7-41: (Dynamic) Buffer Put / Get Request Packet Format

Table 7-26: Buffer Put / Get Packet-specific Fields

Field Name	Field Abbreviation	Size (bits)	Description
Allocation Hints	AH	2	<p>If a Dynamic Buffer Put or a Dynamic Buffer Get, then the Responder may use this field to optimize buffer allocation for subsequent access.</p> <p>No optimization indicates the Responder may allocate the buffer using policies outside of this specification's scope.</p> <p>Throughput optimization indicates the Responder should allocate the buffer to maximize high-bandwidth data movement, e.g., large data movement.</p> <p>Latency optimization indicates the Responder should allocate the buffer to minimize subsequent read or write request latency.</p> <p>Throughput and latency optimization indicates the Responder should allocate the buffer such that it equally considers both subsequent throughput and low-latency access.</p> <p>If the request packet is not a Dynamic Buffer Put or a Dynamic Buffer Get request packet, then this field shall be Reserved.</p> <p>0x0—No Optimization 0x1—Throughput optimization 0x2—Latency optimization 0x3—Through and Latency optimization</p>
Buffer A Address	Buffer A	64	<p>Address of buffer A</p> <p>If a Dynamic Buffer Get request packet, then this field shall be set to 0x0. The allocated address is returned within the Buffer Allocate Response packet.</p>
Buffer B Address	Buffer B	64	<p>Address of buffer B</p> <p>If a Dynamic Buffer Put request packet, then this field shall be set to 0x0. The allocated address is returned within the Buffer Allocate Response packet.</p>
DCID of Buffer B	DCID B	12	<p>Destination CID of the component containing Buffer B</p> <p>If DCID == DCIDB and, if indicated, DSID = DSID B, then this is a component-local request.</p>
Buffer Length	-	32	<p>Length of Buffer A and Buffer B in bytes. If zero, then the actual length is <math>2^{32}</math> bytes. If a Dynamic Buffer Put or Dynamic Buffer Get request packet, then this field represents the buffer size to dynamically allocate.</p>
R0	-	1	Reserved

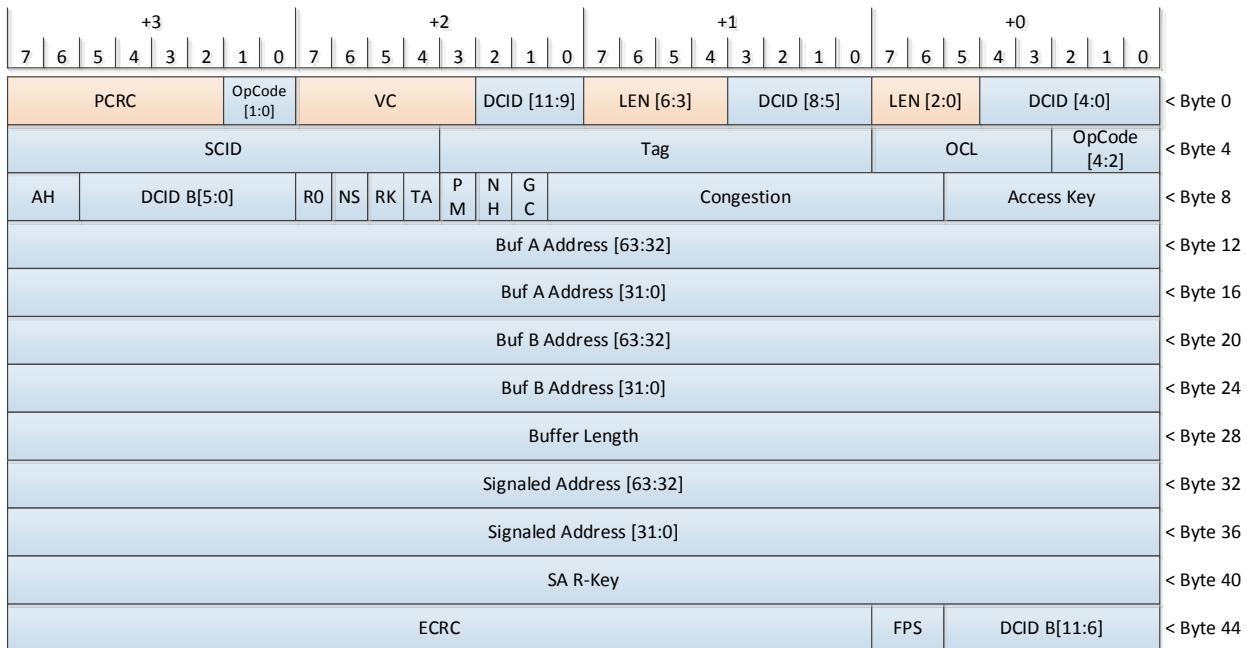


Figure 7-42: Signaled (Dynamic) Buffer Put / Get Request Packet Format

Table 7-27: Signaled (Dynamic) Buffer Put / Get Packet-specific Fields

Field Name	Field Abbreviation	Size (bits)	Description
Allocation Hints	AH	2	<p>If a Signaled Dynamic Buffer Put or a Signaled Dynamic Buffer Get, then the Responder may use this field to optimize buffer allocation for subsequent access.</p> <p>No optimization indicates the Responder may allocate the buffer using policies outside of this specification's scope.</p> <p>Throughput optimization indicates the Responder should allocate the buffer to maximize high-bandwidth data movement, e.g., large data movement.</p> <p>Latency optimization indicates the Responder should allocate the buffer to minimize subsequent read or write request latency.</p> <p>Throughput and latency optimization indicates the Responder should allocate the buffer such that it equally considers both subsequent throughput and low-latency access.</p> <p>If the request is not a Signaled Dynamic Buffer Put or a Signaled Dynamic Buffer Get request packet, then this field shall be Reserved.</p> <p>0x0—No Optimization 0x1—Throughput optimization 0x2—Latency optimization 0x3—Through and Latency optimization</p>

Field Name	Field Abbreviation	Size (bits)	Description
Buffer A Address	Buffer A	64	Address of buffer A If a Signaled Dynamic Buffer Get request packet, then this field shall be set to 0x0. The allocated address is returned within the Buffer Allocate Response packet.
Buffer B Address	Buffer B	64	Address of buffer B If a Signaled Dynamic Buffer Put request packet, then this field shall be set to 0x0. The allocated address is returned within the Buffer Allocate Response packet.
DCID of Buffer B	DCIDB	12	Destination CID of the component containing Buffer B. If DCID == DCIDB then this is a component-local request.
Buffer Length	-	32	Length of Buffer A and Buffer B in bytes. If zero, then the actual length is $2^{32}$ bytes. If a Signaled Dynamic Buffer Put or Signaled Dynamic Buffer Get request packet, then this field represents the buffer size to dynamically allocate.
Signaled Address	-	64	Address of data value to be atomically incremented.
SA R-Key	-	32	Region Key (R-Key) associated with Signaled Address
RO	-	1	Reserved

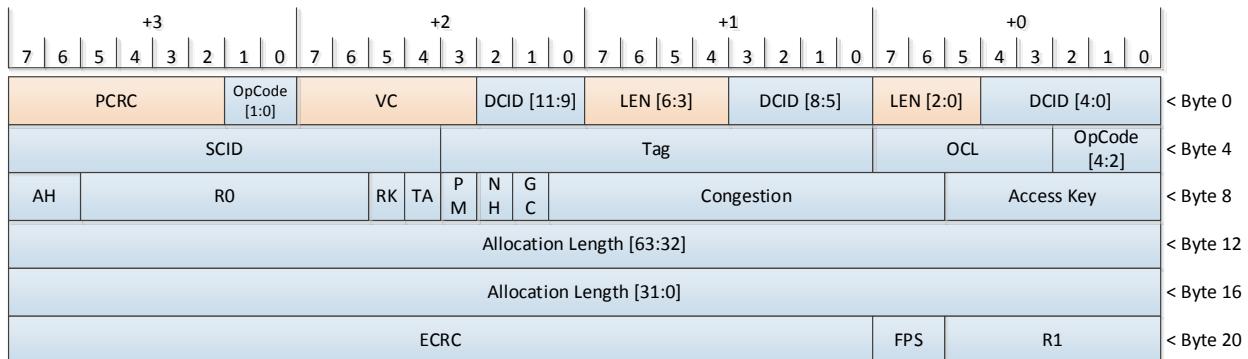


Figure 7-43: Dynamic Buffer Allocate Request Packet Format

Table 7-28: Dynamic Buffer Allocate Packet-specific Fields

Field Name	Size (bits)	Description
Allocation Hints (AH)	2	The Responder may use this field to optimize buffer allocation for subsequent access. No optimization indicates the Responder may allocate the buffer using policies outside of this specification's scope.

Field Name	Size (bits)	Description
Allocation Length	64	Throughput optimization indicates the Responder should allocate the buffer to maximize high-bandwidth data movement, e.g., large data movement.
		Latency optimization indicates the Responder should allocate the buffer to minimize subsequent read or write request latency.
		Throughput and latency optimization indicates the Responder should allocate the buffer such that it equally considers both subsequent throughput and low-latency access.
R0	8	0x0—No Optimization 0x1—Throughput optimization 0x2—Latency optimization 0x3—Through and Latency optimization
		Allocation buffer length  If a Dynamic Buffer Allocate is executed as an independent operation, then the allocated buffer may be up to $2^{64}$ bytes in length.
		If the operation is executed in conjunction with a Put or Get operation, then the allocated buffer shall be limited to the length of the respective request.
R1	6	Reserved

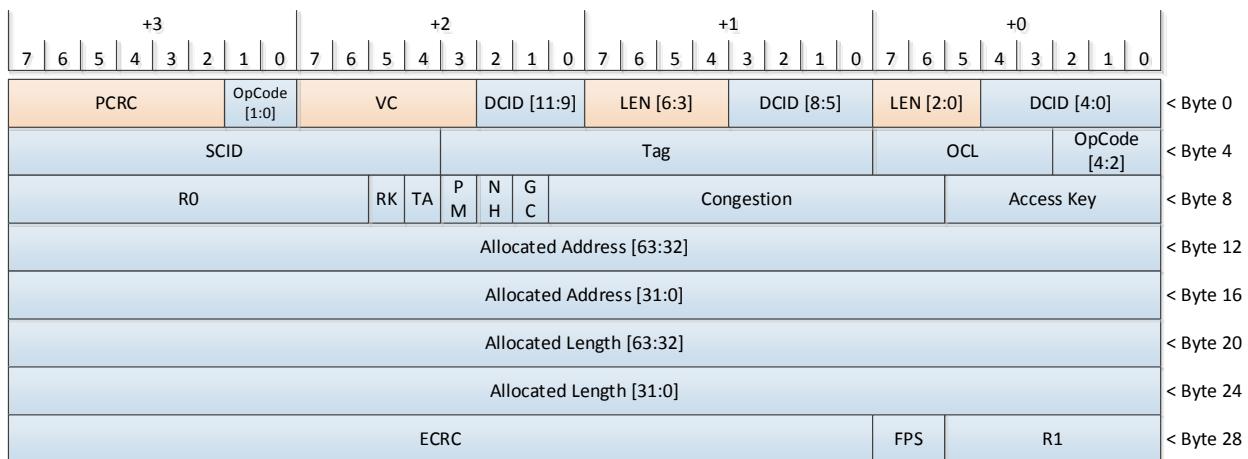


Figure 7-44: Dynamic Buffer Release Request Packet Format

Table 7-29: Dynamic Buffer Release Packet-specific Fields

Field Name	Size (bits)	Description							
<b>Allocated Address</b>	64	The address returned from a previously executed Dynamic Buffer Allocate, Dynamic Buffer Put / Get, or a Signaled Dynamic Buffer Put / Get.							
<b>Allocated Length</b>	64	Length of the previously dynamically allocated buffer							
<b>R0</b>	10	Reserved							
<b>R1</b>	6	Reserved							
PCRC	OpCode [1:0]	VC	DCID [11:9]	LEN [6:3]	DCID [8:5]	LEN [2:0]	DCID [4:0]	< Byte 0	
SCID				Tag				OCL	OpCode [4:2]
AH	DCID B[5:0]	RO	NS	RK	TA	P M	N H	G C	Congestion
Buf A0 Address [63:32]									
Buf A0 Address [31:0]									
Buf B0 Address [63:32]									
Buf B0 Address [31:0]									
Buffer Length 0									
Buf A1 Address [63:32]									
Buf A1 Address [31:0]									
Buf B1 Address [63:32]									
Buf B1 Address [31:0]									
Buffer Length 1									
Buf A2 Address [63:32]									
Buf A2 Address [31:0]									
Buf B2 Address [63:32]									
Buf B2 Address [31:0]									
Buffer Length 2									
Buf A3 Address [63:32]									
Buf A3 Address [31:0]									
Buf B3 Address [63:32]									
Buf B3 Address [31:0]									
Buffer Length 3									
ECRC							FPS	DCID B[11:6]	
< Byte 96									

Figure 7-45: Buffer Putv / Getv Request Packet Formats

Table 7-30: Buffer Putv / Getv Packet-specific Fields

Field Name	Size (bits)	Description
<b>DCID of Buffer B</b>	12	Destination CID of the component containing Buffer B. If (DCID == DCIDB && DSID == DSID B) then this is a component-local request, i.e., the Responder contains both Buffer A and Buffer B.
<b>Buffer A<sub>k</sub></b>	64	One of the Buffer A vector fields. A component may support zero-based or non-zero based addressing (see <i>Core Structure Component CAP 1 Address Field Interpretation</i> ).
<b>Buffer B<sub>k</sub></b>	64	One of the Buffer B vector fields. A component may support zero-based or non-zero based addressing (see <i>Core Structure Component CAP 1 Address Field Interpretation</i> ).
<b>Buffer Length<sub>k</sub></b>	32	One of the Buffer Length vector fields. Buffer Length vectors shall be populated in order they appear in the packet, i.e., Buffer Length <sub>0-3</sub> . If a request does not require all vectors, then the unused vectors shall be set to Null with the corresponding Buffer Length = 0. Null vectors shall be set in reverse order, i.e., Buffer Length <sub>3-1</sub> . Length is in bytes. If zero, then the actual length is $2^{32}$ bytes.
<b>R0</b>	1	Reserved

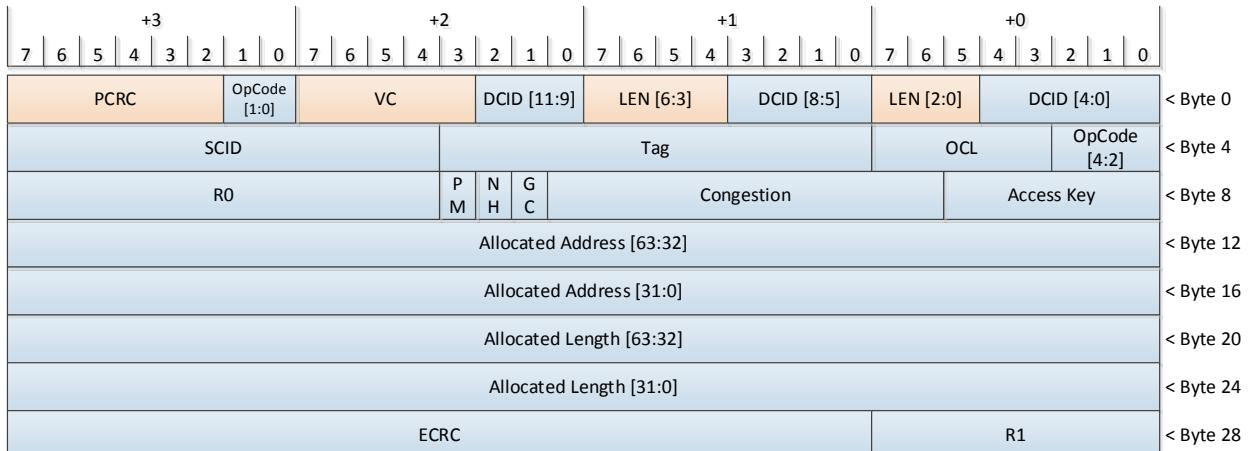


Figure 7-46: Buffer Allocate Response Packet Format

Table 7-31: Buffer Allocate Response Packet-specific Fields

Field Name	Size (bits)	Description
<b>Allocated Address</b>	64	This field contains the address of the start of the integer 16-byte aligned, dynamically-allocated buffer.
<b>Allocated Length</b>	64	This field contains the length (16-byte multiples) of the allocated buffer.

Field Name	Size (bits)	Description
R0		<p>A Responder may return a length that is equal to or greater than the requested length, e.g., a Responder allocates memory in integer 4096-byte multiples and rounds up the allocated length.</p> <p>The Requester shall be responsible for tracking the actual allocated length, and supplying this length if the buffer is released through a Dynamic Buffer Release request packet.</p>
R0	12	Reserved
R1	8	Reserved

## 7.4. Pattern Requests

Pattern requests enable a pattern to be applied to or against the addressed buffer within a Responder.

Pattern requests are divided into two categories:

- Pattern Set—repeatedly copy a pattern to a target buffer
- Pattern Comparison—compare a pattern against a target buffer and return the requested results, e.g., the first address containing the pattern or the number of pattern occurrences within the buffer. Comparison patterns may be either a set of data bytes, e.g., *String S*, or a regular expression comprehended by the Responder. *Pattern Applied to Buffer A* illustrates an example of a pattern executed by a pattern engine against Buffer A with a corresponding Pattern Response returned.

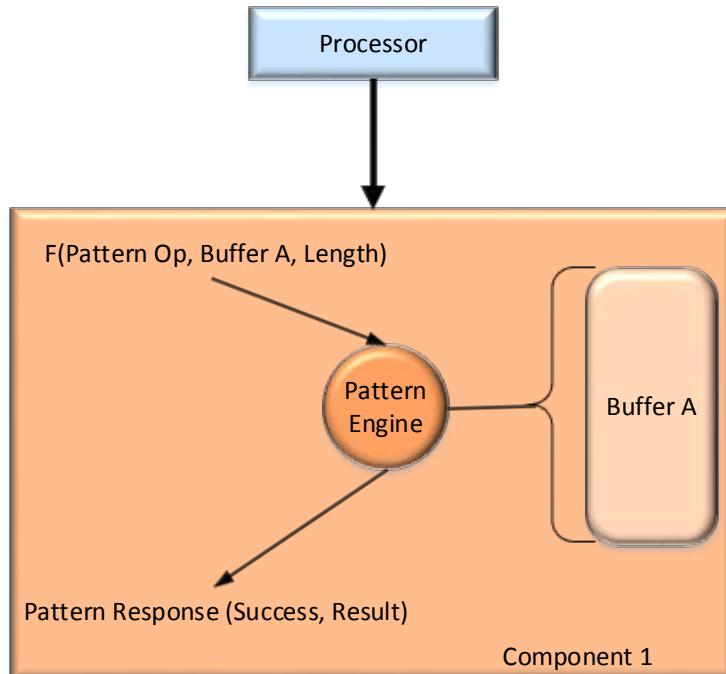


Figure 7-47: Pattern Applied to Buffer A

### 7.4.1. Pattern Features & Requirements

The following are the features and requirements associated with Pattern requests:

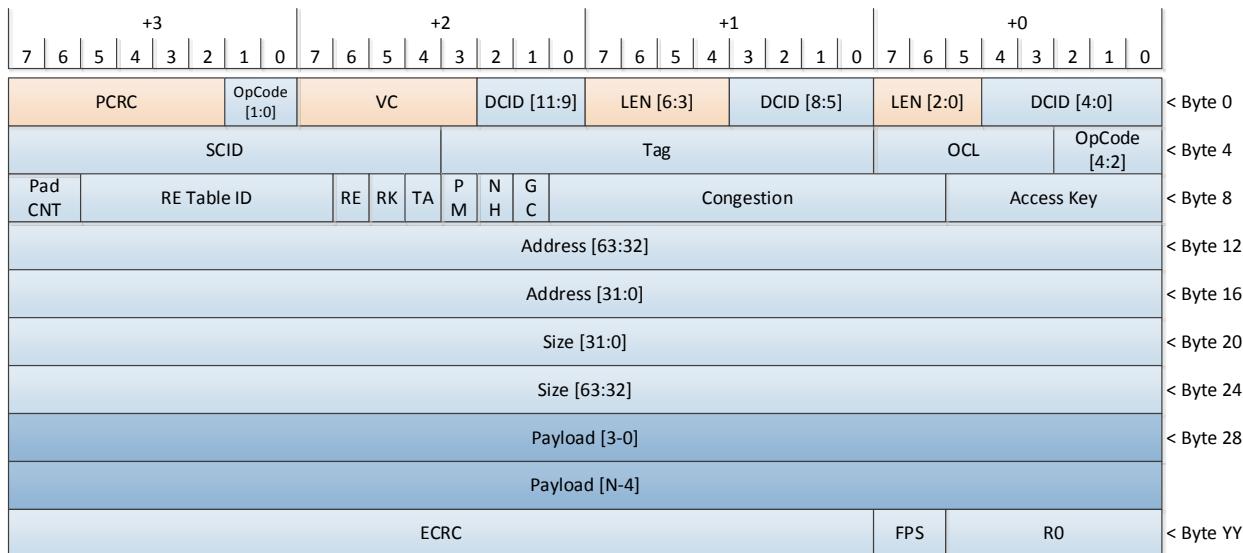
- Any component type may support Pattern operations.
- A component may support all or only a subset of the Pattern request types (*Advanced 1 OpClass OpCodes*).
- A Pattern request packet shall use the *Pattern Request Packet Format*.
- A Pattern response packet shall use the *Pattern Response Packet Format*.
- A Pattern Comparison operation is either a byte-for-byte comparison of the Pattern to the addressed buffer at pattern-size intervals or a regular expression applied to the addressed buffer. For example, an 8-byte Pattern is compared to the addressed buffer at integer 8-byte intervals.
- The payload field may carry 1 to Max\_Packet\_Payload bytes that contain the pattern or is an integer index into the indicated *Component RE Table Structure* that contains the regular expression to apply to the buffer. The payload field shall be rounded up to an integer 4-byte multiple. The Pad CNT field shall reflect the number of pad bytes (0-3). The payload length is calculated as follows:
  - Payload Length = Packet Length – (total protocol bytes) – Pad CNT
- A Pattern request packet shall be treated as non-idempotent. The Requester and the Responder shall take the actions as described in *Non-idempotent Request (NIR)*.
- A Pattern request packet has a non-deterministic execution time. To ensure Reliable Delivery, a Pattern request-Pattern Response packet exchange shall comply with *Non-Deterministic Request Execution*.
- If Pattern request packet validation or execution fails, then the Responder shall return a Core 64 OpClass *Standalone Acknowledgment* to the Requester with the highest precedence error indicated. If Pattern request packet validation and execution succeeds, then the Responder shall return a Pattern Response packet.
- If a Responder receives more Pattern request packets (of any type) than supported by the Responder, then the Responder may silently discard the new Pattern request packet or return a *Responder Not Ready (RNR) NAK*.
- If the solution requires the addressed buffer to remain unmodified during the execution of a Pattern operation, then it is up to the solution to ensure that other request packets do not modify the addressed buffer, e.g., the use of Atomic operations to control buffer access.

### 7.4.2. Pattern Request Types

In the following, Pattern refers to the contents of the packet's Pattern field.

- Pattern Set
  - Request iteratively copies the Pattern at Pattern size strides starting at Address. For example, an 8-byte Pattern of ABCDEFGH is repeatedly copied as ABCDEFGH... ABCDEFGH until it reaches the end of the buffer. If the Buffer Length is not a multiple of the Pattern's size, then a partial copy of the pattern shall occupy the remainder of the Buffer. For example, ABCDEFGH... ABCDEFGH ABCD.
- Pattern Count
  - Request iteratively compares the Pattern (byte sequence or regular expression) with memory starting at Address and returns the total number of successful comparison.

- The PRSLT field shall be set to indicate the PRSLT-dependent field is a 64-bit unsigned integer.
- The total number shall be returned within the PRSLT-dependent field within a Pattern Response packet.
- The Pattern's size shall be a minimum of 4 bytes (payload).
- Pattern Match
  - Request compares the Pattern (byte sequence or regular expression) with the memory starting at address, and returns zero or more 64-bit addresses corresponding to successful comparisons.
    - Addresses are contiguously placed starting at byte 0 of the PRSLT field up through byte 255.
  - If the request is successfully validated and executed, a Pattern Response shall be returned after the PRSLT is filled (the last address indicates the last location successfully compared) or upon reaching the end of the buffer.
- Pattern Response
  - Pattern Response returns the results of the Pattern Request. If successful, then the Requester examines the PRSLT field to interpret the results. If unsuccessful, then the Requester examines the Reason field to interpret the failure cause.



Field Name	Field Abbreviation	Size (bits)	Description							
RE Table ID	-	7	0b—Sequence of bytes 1b— <i>Component RE Table Structure</i> Index							
			If RE == 1b, then this indicates the <i>Component RE Table Structure</i> to use to execute this request packet, else this field shall be Reserved							
R0	-	6	Reserved							

Figure 7-49: Pattern Response Packet Format

Table 7-33: Pattern Response Packet Fields

Field Name	Field Abbreviation	Size (bits)	Value	Description	
Pattern Result	PRSLT	2	-	Indicates if the Result field is present and how to interpret it if so.	
			0x0	No Result field. Typically associated with a successful Pattern Set or a pattern request failed.	
			0x1	64-bit counter. Typically associated with a Pattern Count request.	
			0x2	Effective 64-bit zero-based address with integer 16-byte resolution (60 bit address value with lower 4 bits implicit and treated as zero; unused upper address bits shall be set to zero).	
			0x3	Reserved	
Num Addresses	Num Addresses	7	-	If the response is for a Pattern Match request packet, then this field equals the number of addresses within the payload.	
				For Max_Payload_Size = 256 bytes, the maximum Num Addresses is 64.	
				If the response is not for a Pattern Match request packet, then this field shall be Reserved.	

Field Name	Field Abbreviation	Size (bits)	Value	Description
Payload Result	Payload Result	-	-	If present, content type is indicated by the Result field.
				If the Result is a pointer or a count, then this field is 64-bits in length.
				If the Result is for a Pattern Match, then this field may contain up to 256 bytes. The payload result shall be an integer 4-byte multiple. Unused bytes shall be Reserved.
R0	-	2	-	Reserved
R1	-	8	-	Reserved

## 7.5. Multi-Op Requests

Multi-Op Requests allow multiple requests to be coalesced into a single packet to improve protocol efficiency, reduce request service latency, and to simplify the solution programming model (e.g., a write operation combined with an interrupt). Requests may be Semantically Dependent ensuring that all requests are received at the same effective time. For example, coalescing three Core 64 Read Requests into a single packet provides a significant reduction in protocol overheads and associated processing and enables all three packets to be pipelined, effectively reducing aggregate latency.

Multi-Op Requests can also provide a simple buffer scatter-gather service. Multi-Op Requests also enable an interrupt to be signaled in conjunction with one or more Write sub-requests or a memory location to be atomically incremented after the associated Write sub-request is successfully completed.

The following are the features and requirements associated with Multi-Op request packets:

- Any component type may support Multi-Op request packets as a Requester, a Responder, or as both a Requester and a Responder.
- Multi-Op request packets shall use the Multi-Op Request Packet formats as illustrated in this sub-section.
  - Each Multi-Op request packet contains a sub-OpCode (Sub-OPC). The sub-OpCode specifies the Multi-Op operation type. The *OpCode Set Structure* is used to indicate what sub-OpCodes are supported and enabled.
  - A Core 64 Multi-Op operation shall use a Core 64 Multi-Op packet format of the corresponding sub-OpCode type.
- Unless explicitly stated otherwise by this specification, Multi-Op requests are idempotent.
  - A retransmitted signaled Multi-Op request may result in the addressed location used to signal other components being incremented multiple times. If a solution cannot tolerate such behavior, then it should use *Atomic Request and Response Packets*.
- Multi-Op request packets shall be successfully validated prior to executing the first sub-request.
- Read Multi-Op request requirements:
  - Read Multi-Op Requests shall use only the OpCodes as specified in *Multi-Op Read Request Packet OpCodes*.
  - Read Multi-Op Request OpCodes may be used in any combination.

- Sub-requests within a Read Multi-Op request packet may be executed in any order.
- An independent response packet or *Standalone Acknowledgment* shall be returned for each sub-request.
  - If the sub-request is less than or equal to Max\_Packet\_Payload, then the response to a successfully executed Read Multi-Op request shall be a Read Response packet. If the sub-request is greater than Max\_Packet\_Payload, then the response to a successful executed request shall be a sequence of LDM 1 Read Response packets. Further, the Requester and Responder shall perform the steps specified in *Non-Deterministic Request Execution* using Tag 1 to exchange the additional request and response packets.
  - The response packet or *Standalone Acknowledgment* corresponding to a given sub-request shall reflect the Tag associated with the sub-request.
  - If an error is detected during packet validation, then a separate *Standalone Acknowledgment* that reflects the highest-precedence error shall be returned for each sub-request.
  - If an error is detected during sub-request execution, then a separate *Standalone Acknowledgment* that reflects the highest-precedence error shall be returned for each erroneous sub-request.
- Write Multi-Op request requirements:
  - Write Multi-Op request packets with one or more payload fields shall be sequentially executed, i.e., the sub-request corresponding to Address 1 shall be successfully executed before the sub-request corresponding to Address 2 which shall be successfully executed before the sub-request corresponding to Address 3 (if present) which shall be successfully executed before the sub-request corresponding to Address 4 (if present).
  - A single *Standalone Acknowledgment* shall be returned for each Write Multi-Op request packet. If an error is detected, then the *Standalone Acknowledgment* shall reflect the highest precedence error.
    - A single Core 64 *Standalone Acknowledgment* shall be returned for a Write Multi-Op request packet.
    - If an error is detected during packet validation, no sub-requests shall be executed, and the corresponding underlying resources shall not be modified.
    - If an error is detected during sub-request execution, then the contents of the underlying resources may be non-deterministic.
  - The aggregate size of all Payload fields shall be less than or equal to Max\_Packet\_Payload.
  - Write Multi-Op Requests support the following types of signaling:
    - Interrupt—once the payload associated with Address 1 is successfully placed, the Responder shall generate an interrupt.
      - The packet format shall use the Signaled Write packet format.
      - The interrupt shall target Address 2 in a Signaled Write Multi-Op request. Address 2 may be encoded as a native interrupt (see *Interrupts*).
      - The interrupt may be a system interrupt (interrupts operating system or application execution), or may be a component-local (e.g., one that causes a component to take action based on the contents of the associated data payload).
    - LPD Interrupt—once the payload associated with Address 1 is successfully placed, the Responder shall generate a LPD interrupt.

- Signaled Write—once the payload associated with Address 1 is successfully placed, the Responder shall atomically increment by one the data value located relative to Address 2, and shall store the result relative to Address 2.
  - The packet format shall be the Signaled Write packet format.
  - The data value located at Address 2 shall be treated as an unsigned 64-bit integer.
  - A Signaled Write is a non-idempotent request. The Responder shall take the actions as described in *Non-idempotent Request (NIR)* to prevent non-deterministic results and behaviors. Since the result of the atomic increment is not returned to the Requester, the tracking logic should note the request's success, but should not record the result itself. If a Signaled Write is retransmitted and the request's success was recorded, then neither the Write nor the atomic increment sub-request shall be re-executed; only a positive acknowledgment shall be returned.
  - Atomic and Signaled Write request packets share the same underlying non-idempotent Responder resources and tracking logic.
- Write-Wake—once the payload associated with Address 1 is successfully placed, the Responder shall issue a wake operation of the software process or thread identified by the Thread ID field.
  - The wake operation is Responder-specific, e.g., it may be an instruction specified in an instruction set architecture.
  - The Write sub-request shall be successfully executed prior to executing the Wake operation.
  - A *Standalone Acknowledgment* shall be returned indicating success or the highest-precedence error.
  - If the write sub-request succeeds but the wake fails, then the wake failure shall be handled as specified in *Wake Thread*.
    - If the addressed media targeted by a Multi-Op request has entered Fatal Media Error Containment, then this request shall not be executed, and a *Standalone Acknowledgment* (Reason = Fatal Media Error Containment Triggered) shall be returned.
- Write-Read Multi-Op request requirements:
  - The Write sub-request shall be successfully executed prior to executing the Read operation.
  - The Read operation size shall be less than or equal to Max\_Packet\_Payload.
  - If a Core 64 Write-Read Multi-Op request is successfully validated and executed, then a Core 64 Read Response packet shall be returned. This packet shall acknowledge the Write and Read sub-requests.
  - If a Write-Read Multi-Op request fails packet validation or execution, then a *Standalone Acknowledgment* shall be returned indicating the highest-precedence error.
- If present, then the R-Key within Core 64 Multi-Op request packet shall apply to all sub-requests.
- Unless explicitly stated otherwise by this specification, all sub-requests shall be governed by their respective analog normative specification text, e.g., a Read 32 sub-request is governed by the same requirements and rules as specified in *Read and Read Response*.

Table 7-34: Multi-Op Request Sub-OpCodes

OpCode Name	OpCode Encoding	Description
Read Dual-Op	0x0	Two independent Read operations
Read Triple-Op	0x1	Three independent Read operations
Read Quad-Op	0x2	Four independent Read operations
Write Dual-Op	0x3	Two independent Write operations
Write Triple-Op	0x4	Three independent Write operations
Write Quad-Op	0x5	Four independent Write operations
Signaled Write	0x6	Write then Signaled update operation
Write-Interrupt	0x7	Write then Interrupt operation
Write-Read	0x8	Write then Read operation
Write-Wake	0x9	Write then Wake operation
Reserved	0xA-0x1F	Reserved Sub-OpCodes

For Multi-Op read request packet OpCodes larger than 256 byte, the Requester and Responder shall support and enable the LDM 1 Read Response operation.

Table 7-35: Multi-Op Read Request Packet OpCodes

OpCode Name	OpCode Encoding	Read Response Packet Type	Description
Read 16	0x0	Core 64 Read Response	Read 16 bytes
Read 32	0x1	Core 64 Read Response	Read 32 bytes
Read 64	0x2	Core 64 Read Response	Read 64 bytes
Read 128	0x3	Core 64 Read Response	Read 128 bytes
Read 256	0x4	Core 64 Read Response	Read 256 bytes
Read 1 KiB	0x5	LDM 1 Read Response	Read 1024 bytes
Read 2 KiB	0x6	LDM 1 Read Response	Read 2048 bytes
Read 4 KiB	0x7	LDM 1 Read Response	Read 4096 bytes

OpCode Name	OpCode Encoding	Read Response Packet Type	Description
Read 8 KiB	0x8	LDM 1 Read Response	Read 8192 bytes
Read 32 KiB	0x9	LDM 1 Read Response	Read 32 KiB
Read 64 KiB	0xA	LDM 1 Read Response	Read 64 KiB
Read 128 KiB	0xB	LDM 1 Read Response	Read 128 KiB
Read 256 KiB	0xC	LDM 1 Read Response	Read 256 KiB
Read 512 KiB	0xD	LDM 1 Read Response	Read 512 KiB
Read 1 MiB	0xE	LDM 1 Read Response	Read 1 MiB
Reserved	0xF	-	Reserved

*Common Multi-Op Packet Fields* lists fields that are common to multiple Multi-Op request packets. Some fields may not apply to a given packet format.

Table 7-36: Common Multi-Op Packet Fields

Field Name	Size (bits)	Description
Sub-OPC	5	Sub-OpCode—see <i>Multi-Op Request Sub-OpCodes</i>
Address 1-4	-	Each Address field corresponds to the respective Op field, e.g., Address 1 corresponds to Op 1. A 40-bit Address field is used in Core Multi-Op request packets. A 60-bit Address field is used in Core 64 Multi-Op request packets.
OP 1-4	4	If present, these fields indicate the sub-request OpCodes. Applicable only to Read Multi-Op Requests.
Tag 1-4	12	If a Read Multi-Op, then each Tag corresponds to the respective OP field, e.g., Tag 1 corresponds to OP 1. If a Write Multi-Op, then there is a single Tag field that corresponds to the entire request packet.
Payload 1-4	-	If present, then the data payload associated with a given sub-request, e.g., Payload 1 corresponds to the first sub-request. Each payload shall be an integer 4-byte multiple. If the packet format supports Pad CNT fields and the payload is not an integer 4-byte multiple, then Pad bytes shall be appended to the payload.

Field Name	Size (bits)	Description
W-LEN 1-4	7	If multiple payload fields are present, then the payload field shall be contiguously placed starting with Payload 1, followed by Payload 2, followed by Payload 3 (if present).
		The length of the corresponding payload in integer 4-byte multiples, e.g., W-LEN 1 corresponds to Payload 1. Pad CNT 1-4 are used to respectively calculate the actual data payload. W-LEN n == 0x0 indicates a 128-byte payload.

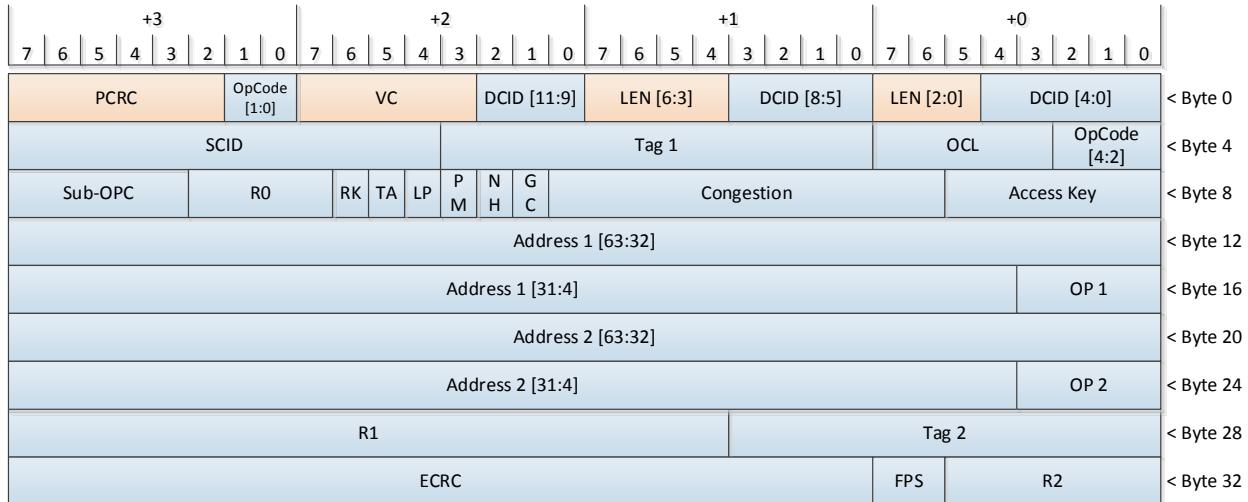


Figure 7-50: Core 64 Read Dual-Op Request Packet Format

Table 7-37: Core 64 Read Dual-Op Request-specific Packet Fields

Field Name	Size (bits)	Description
R0	4	Reserved
R1	20	Reserved
R2	6	Reserved

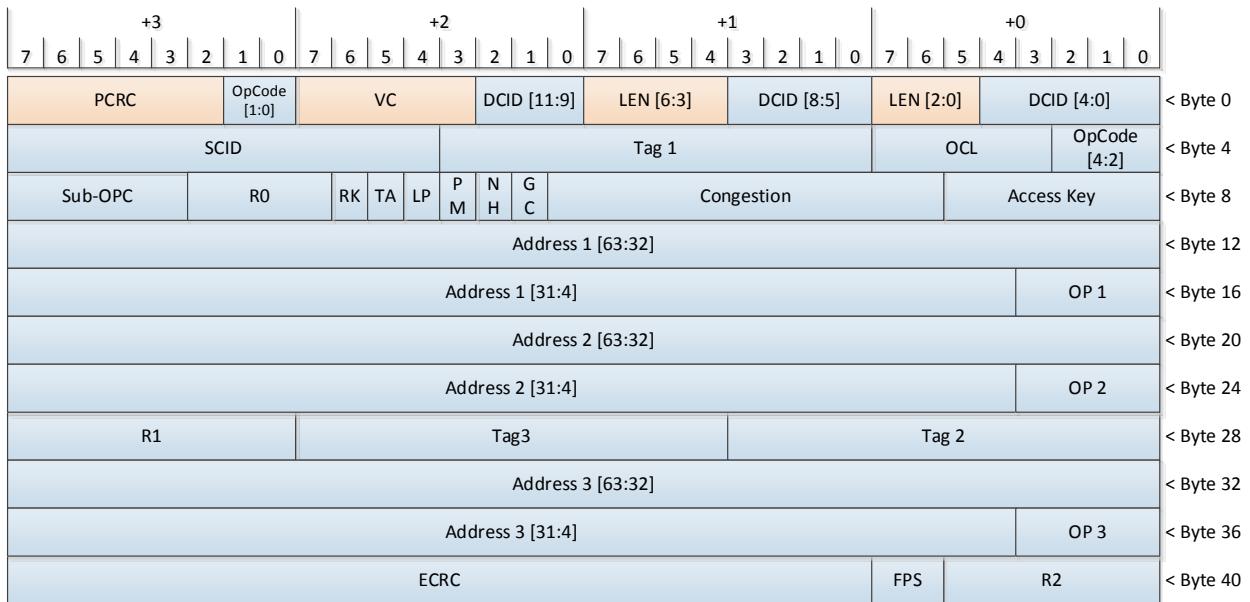


Figure 7-51: Core 64 Read Triple-Op Request Packet Format

Table 7-38: Core 64 Read Triple-Op Request-specific Packet Fields

Field Name	Size (bits)	Description
R0	4	Reserved
R1	8	Reserved
R2	6	Reserved

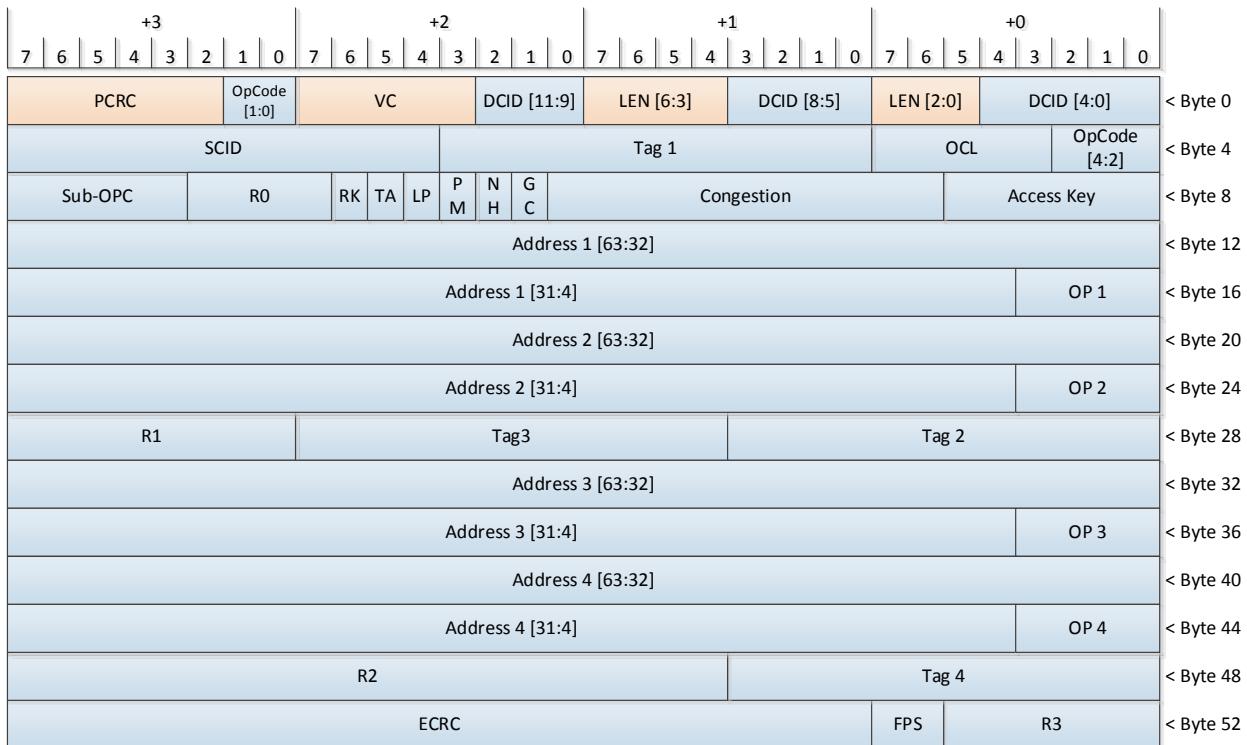


Figure 7-52: Core 64 Read Quad-Op Request Packet Format

Table 7-39: Core 64 Read Quad-Op Request-specific Packet Fields

Field Name	Size (bits)	Description
R0	4	Reserved
R1	8	Reserved
R2	20	Reserved
R3	6	Reserved

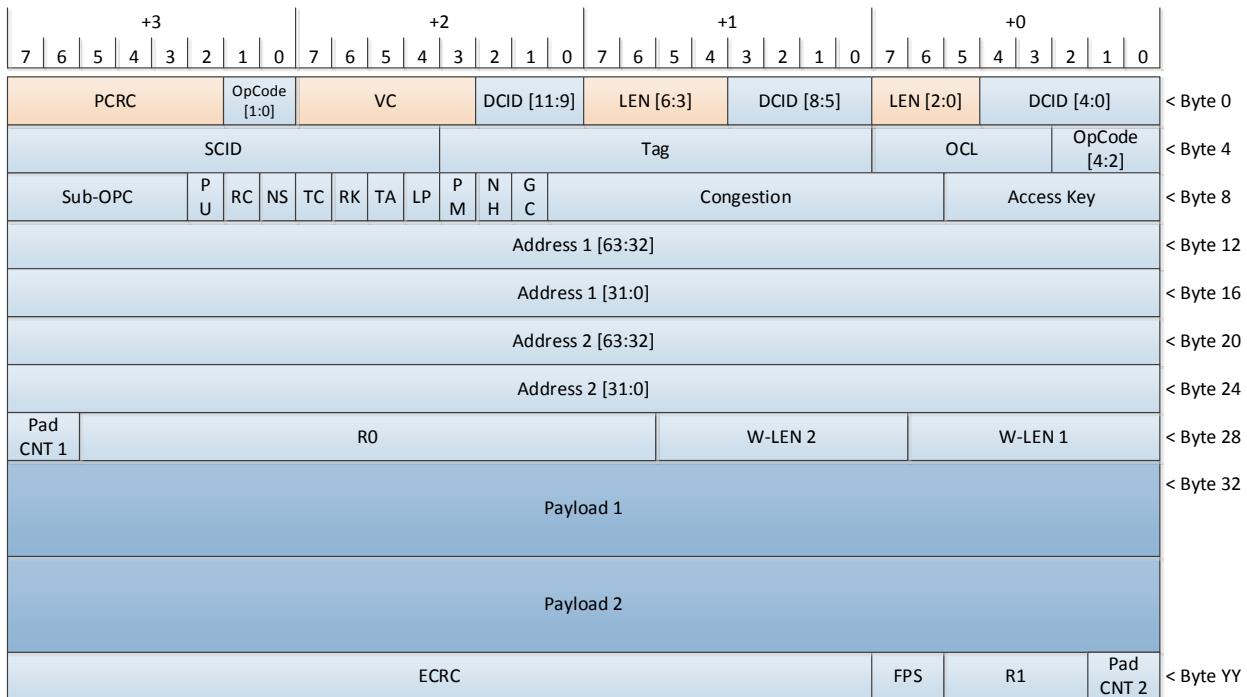


Figure 7-53: Core 64 Write Dual-Op Request Packet Format

Table 7-40: Core 64 Write Dual-Op Request-specific Packet Fields

Field Name	Size (bits)	Description
<b>R0</b>	16	Reserved
<b>R1</b>	4	Reserved

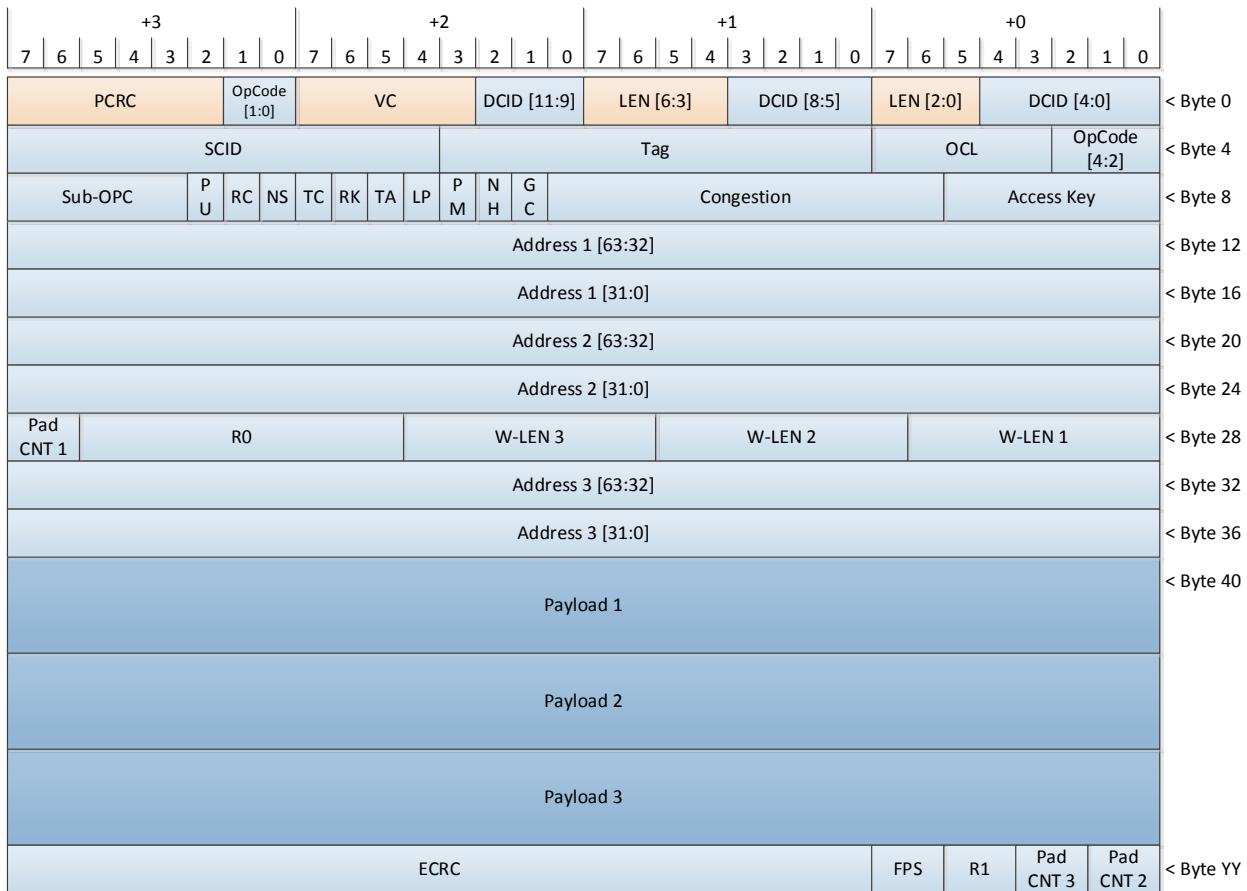


Figure 7-54: Core 64 Write Triple-Op Request Packet Format

Table 7-41: Core 64 Write Triple-Op Request-specific Packet Fields

Field Name	Size (bits)	Description
<b>R0</b>	11	Reserved
<b>R1</b>	2	Reserved

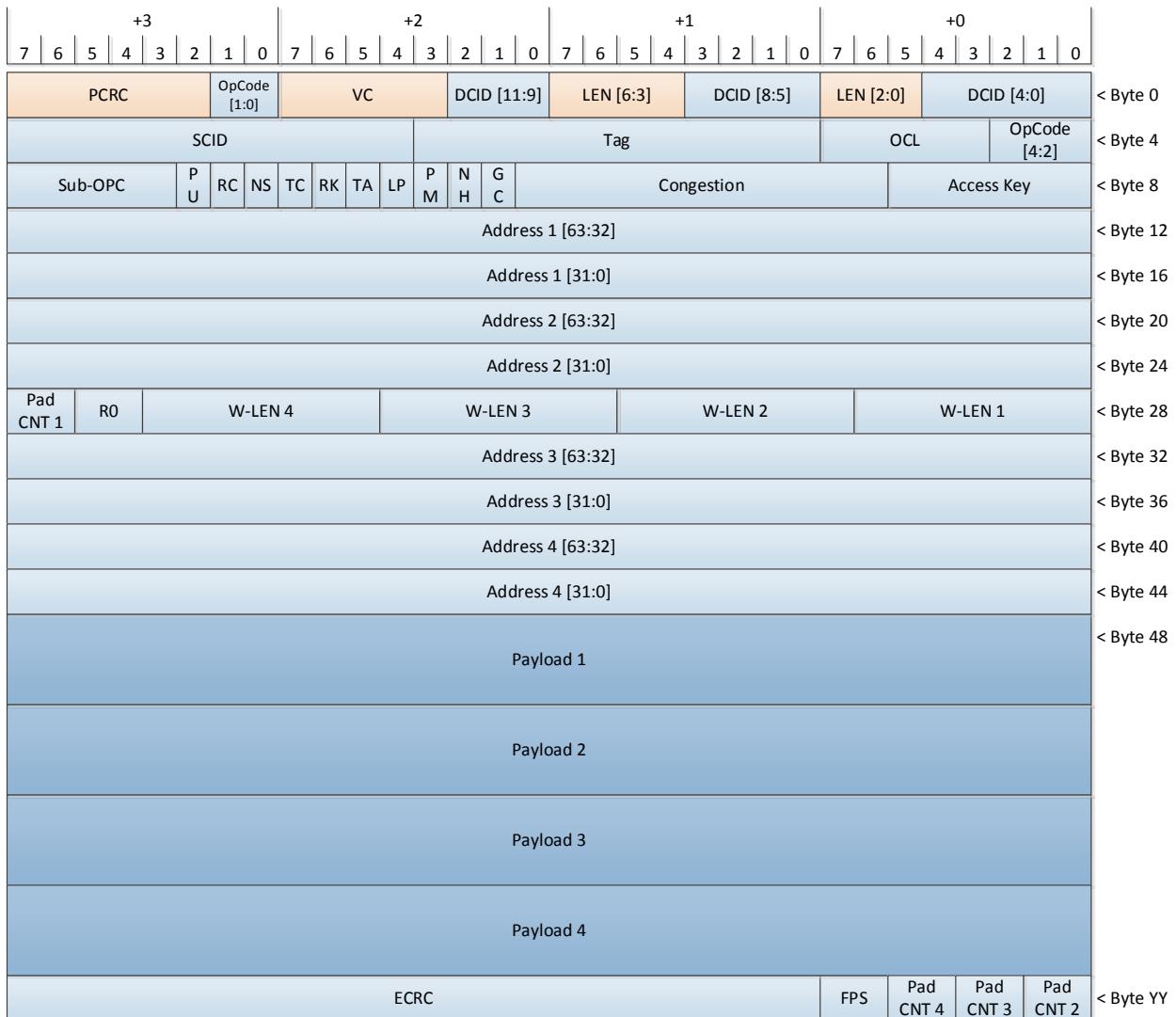


Figure 7-55: Core 64 Write Quad-Op Request Packet Format

Table 7-42: Core 64 Write Quad-Op Request-specific Packet Fields

Field Name	Size (bits)	Description
R0	2	Reserved

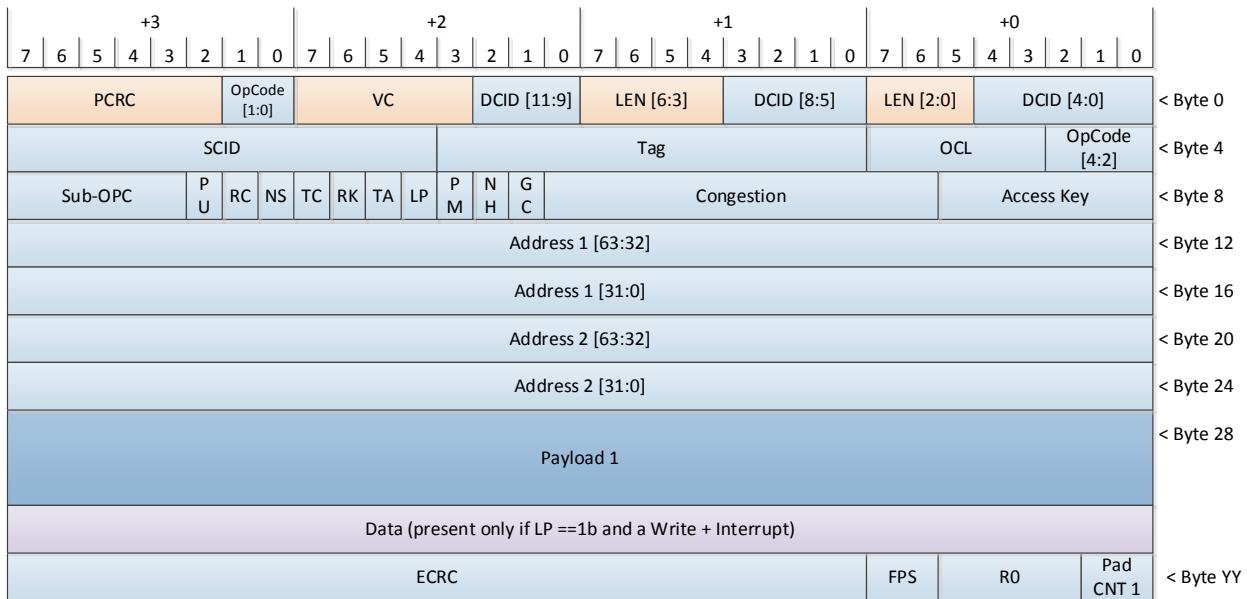


Figure 7-56: Core 64 Signaled Write / Write + Interrupt Request Packet Format

Table 7-43: Core 64 Signaled Write Request Packet-specific Fields

Field Name	Size (bits)	Description
<b>R0</b>	4	Reserved

A Write-Read request is used to write the payload to Address 1, and then read a quantity of data starting at Address 2. Typically, this operation is used to write a portion of a memory range and then to read back the entire memory range.

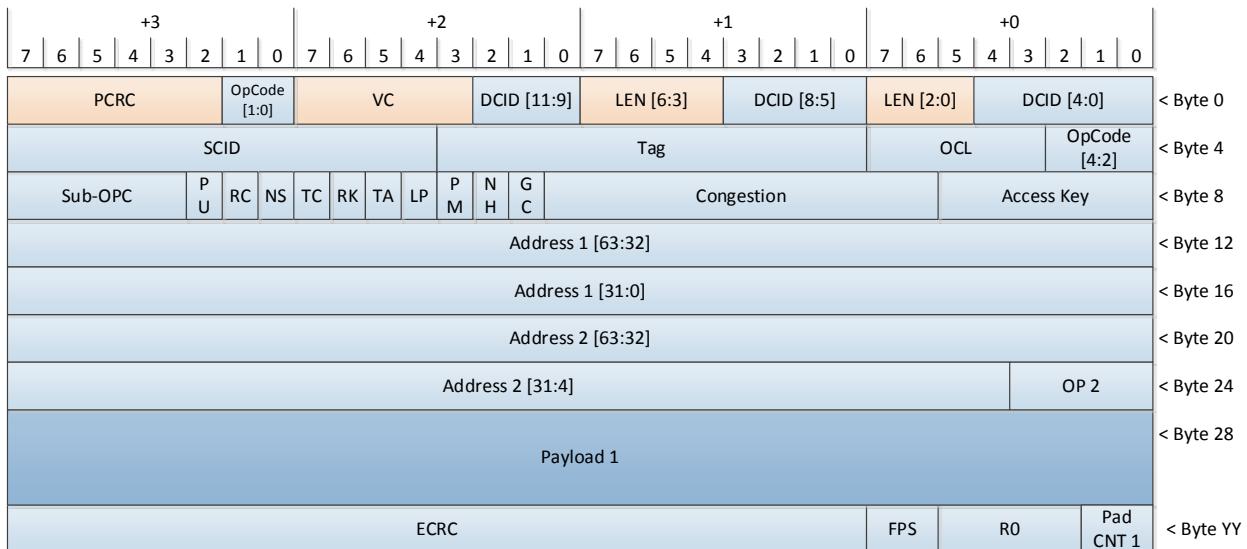


Figure 7-57: Core 64 Write-Read Request Packet Format

Table 7-44: Core 64 Write-Read Request Packet-specific Fields

Field Name	Size (bits)	Description
<b>R0</b>	4	Reserved

A Write-Wake request is used to write the payload to Address 1, and then awaken a process or thread. Awakening a process or thread avoids the cost and complexity associated with interrupt execution, and enables the process or thread to be scheduled based on software-based policies. The efficacy of the wake operation will depend upon the Responder's capabilities and resource robustness.

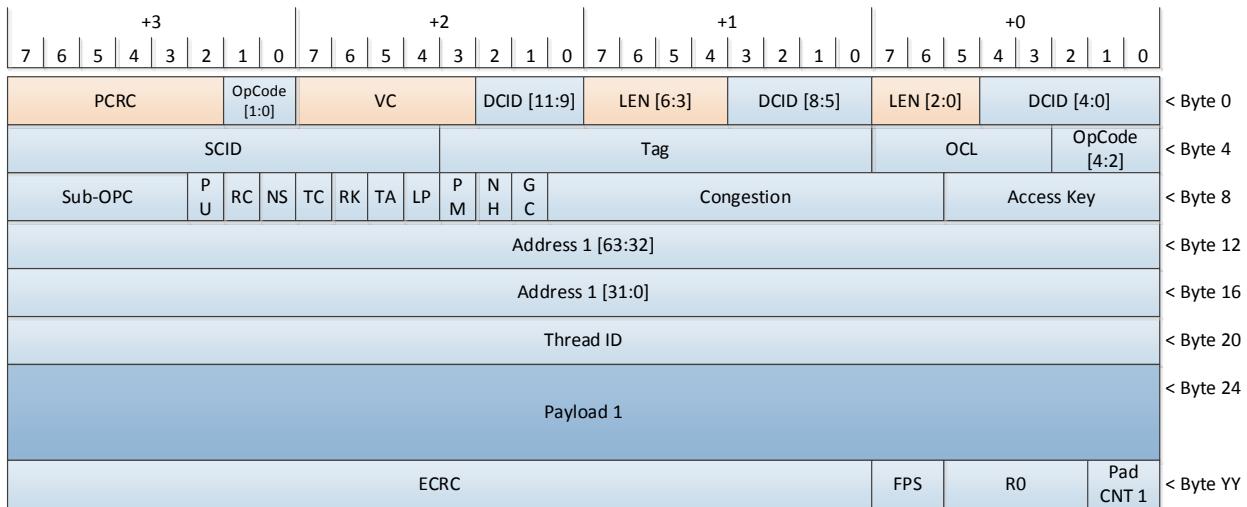


Figure 7-58: Core 64 Write-Wake Request Packet Format

Table 7-45: Core 64 Write-Wake Request Packet-specific Fields

Field Name	Size (bits)	Description
<b>Thread ID</b>	32	The identifier associated with the process or thread of execution that is to be awakened. If the identifier is less than 32 bits, then it shall be placed in the lower bits starting with bit 0 at bit 0. Unused Thread ID upper bits shall be set to zero.
<b>R0</b>	4	Reserved

## 7.6. Vendor-Defined Packets

Vendor-defined packets enable components to customize communication while ensuring basic interoperability at the component and, if applicable, packet relay levels. The P2P-Core OpClass and P2P-Coherency OpClass support up to 32 vendor-defined request operations and 32 vendor-defined response operations. Each explicit vendor-defined OpClass supports up to a total of 32 vendor-defined request and response operations.

The following are the features and requirements associated with explicit vendor-defined OpClass packets:

- Any component type may support vendor-defined packets as a Requester, a Responder, or as both a Requester and a Responder.
- Any component type may support zero or more explicit vendor-defined OpClasses.
  - Each explicit vendor-defined OpClass represents a unique set of vendor-defined OpCodes. All OpCodes within a given vendor-defined OpClass shall be associated with a single UUID. Cooperating components may exchange vendor-defined packets as long as they share a common OpClass and associated UUID.
- Any component type may support P2P-Core or P2P-Coherency OpClass vendor-defined operations or the P2P-Vendor-defined OpClass. All vendor-defined OpCodes associated with any of these OpClass shall be associated with the *OpCode Set Structure* PM-UUID field.
- P2P-Core vendor-defined request packets shall use the *P2P-Core Vendor-Defined Request Packet Format*.
- P2P-Core vendor-defined response packets shall use the *P2P-Core Vendor-Defined Response Packet Format*.
- P2P-Coherency vendor-defined request packets shall use the *P2P-Coherency Vendor-Defined Request Packet Format*.
- P2P-Coherency vendor-defined response packets shall use the *P2P-Coherency Vendor-Defined Response Packet Format*.
- Explicit OpClass Vendor-Defined Packets shall use the *Explicit Vendor-Defined OpClass Packet Format*.
- The implicit P2P-Vendor-defined OpClass (see *End-to-end Operation Classes*) shall use vendor-defined packet formats that are outside of this specification's scope.
- Unless explicitly stated otherwise by this specification, only fields labeled with 'Vendor-Defined' shall be treated as vendor-defined. All other fields shall be governed as specified within this document.
- Explicit vendor-defined OpClasses are enabled through the *OpCode Set Structure*. Software may use the UUID associated with the specific Vendor-defined OpClass in conjunction with a *Vendor-Defined Structure* or mechanisms outside of this specification's scope to interpret the *OpCode Set Structure* and enable the common OpCodes and OpClasses among communicating components.
- Vendor-defined packets that need to traverse a transparent router should contain an appropriate address field in the same location as the respective OpClass packet, or the transparent router shall comprehend the vendor-defined packet formats in order to relay packets.
- Vendor-defined request packets may be acknowledged by a *Standalone Acknowledgment* or by a vendor-defined response packet.
  - Non-vendor-defined fields common to a given OpClass packets shall be validated as such. Any detected errors shall be handled per *Error Detection and Recovery*. A *Standalone Acknowledgment* (if required) shall reflect the highest precedence error.
  - A Requester may use the request packet retransmission algorithms specified within this document or use a vendor-defined mechanism.
- Where applicable, vendor-defined request and response packets may support any of the mechanisms and semantics associated with *Non-idempotent Request (NIR)*, *Responder Not Ready (RNR) NAK*, *Non-Deterministic Request Execution*, etc. If a given mechanism is supported, then the request and response packets shall comply with the corresponding specification sections.
  - If non-idempotent request packets are supported, then the components should support a *Vendor-Defined Structure* to specify the maximum number of outstanding requests per vendor-defined OpClass or set of vendor-defined OpCodes.

- If the addressed media targeted by this request has entered Fatal Media Error Containment, then this request packet shall not be executed, and a *Standalone Acknowledgment* (Reason = Fatal Media Error Containment Triggered) shall be returned.
- If an explicit Vendor-defined OpClass packet uses Gen-Z protocol fields that are located in vendor-defined fields, then the Gen-Z protocol fields shall be located at the specified locations, and shall comply with the specified semantics. For example, if an explicit OpClass vendor-defined request packet supports *Region Key (R-Key)*, then the RK bit and the R-Key field shall be located as illustrated in *Conditional Field Locations within an Explicit OpClass Packet*.

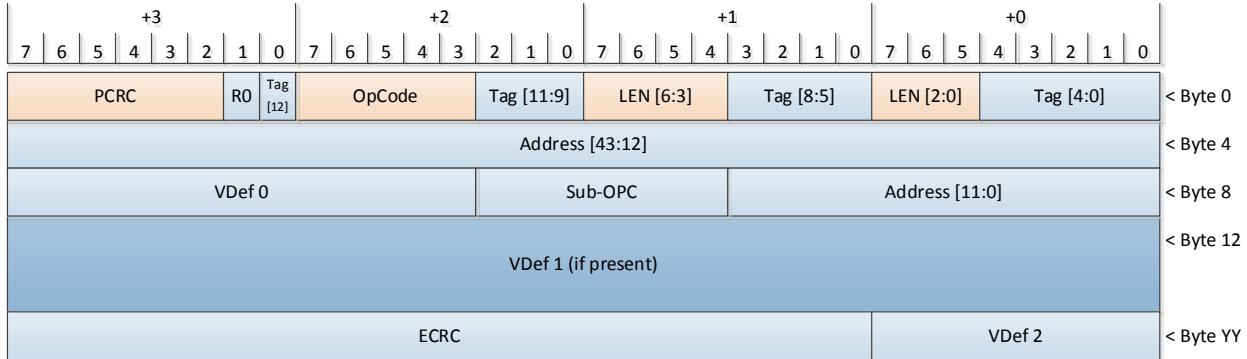


Table 7-46: P2P-Core Vendor-defined Request Packet Fields

Field Name	Size (bits)	Description
VDef 0	13	Vendor-defined field
VDef 1	Variable	Vendor-defined field (if present)
VDef 2	8	Vendor-defined field

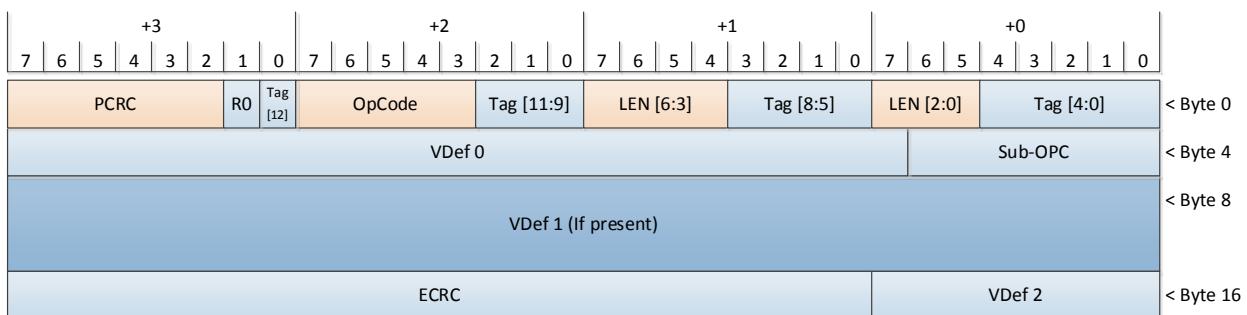


Table 7-47: P2P-Core Vendor-defined Response Packet Fields

Field Name	Size (bits)	Description
VDef 0	25	Vendor-defined field
VDef 1	Variable	Vendor-defined field (if present)
VDef 2	8	Vendor-defined field

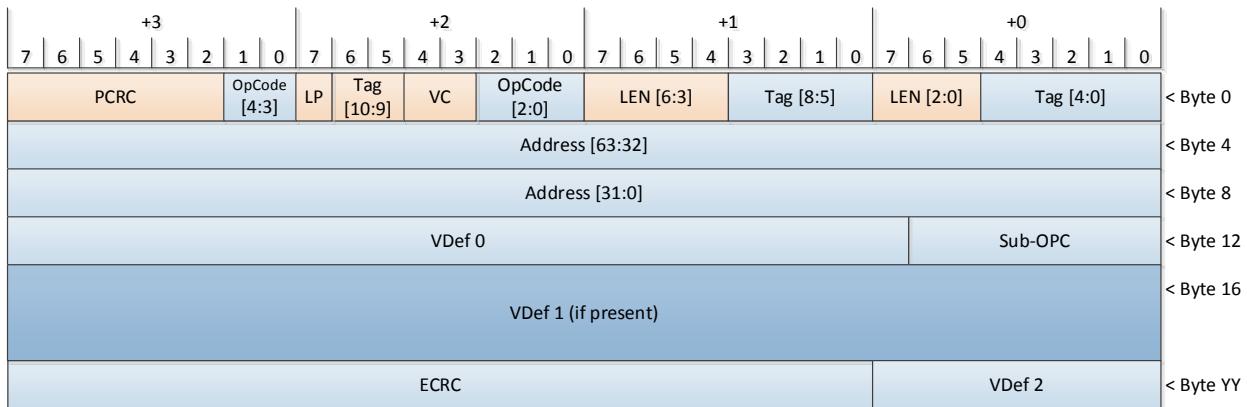


Figure 7-61: P2P-Coherency Vendor-Defined Request Packet Format

Table 7-48: P2P-Coherency Vendor-defined Request Packet Fields

Field Name	Size (bits)	Description
VDef 0	25	Vendor-defined field
VDef 1	Variable	Vendor-defined field (if present)
VDef 2	8	Vendor-defined field

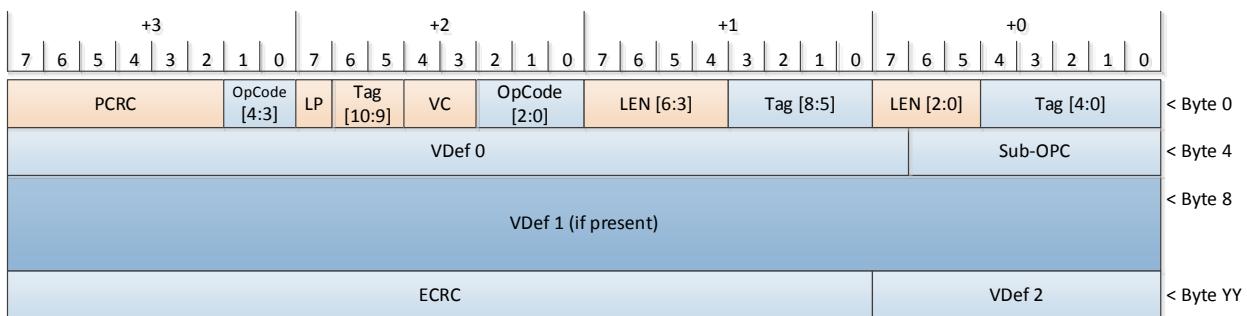


Figure 7-62: P2P-Coherency Vendor-Defined Response Packet Format

Table 7-49: P2P-Coherency Vendor-defined Response Packet Fields

Field Name	Size (bits)	Description
VDef 0	25	Vendor-defined field
VDef 1	Variable	Vendor-defined field (if present)
VDef 2	8	Vendor-defined field

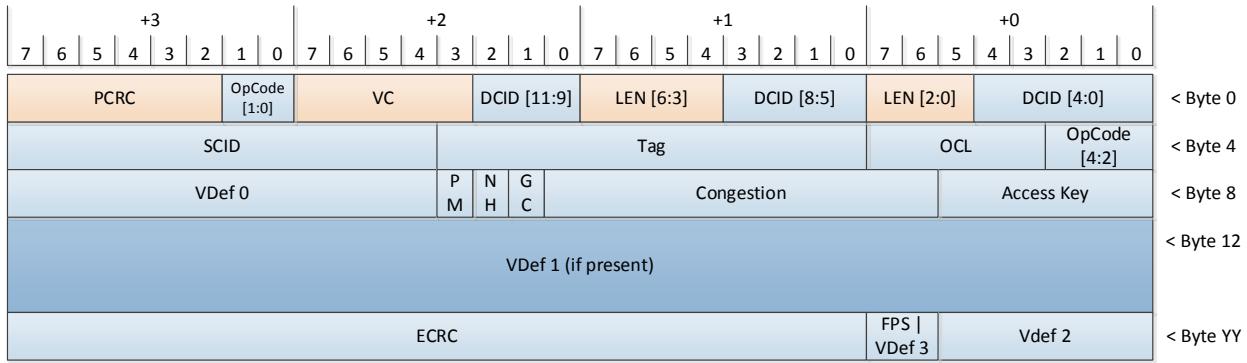


Figure 7-63: Explicit Vendor-Defined OpClass Packet Format

Table 7-50: Explicit OpClass Vendor-defined Packet Fields

Field Name	Size (bits)	Description
VDef 0	12	Vendor-defined field
VDef 1	Variable	Vendor-defined field (if present)
VDef 2	6	Vendor-defined field
FPS   VDef 3	2	If this is a request packet, then this field shall contain the FPS associated with this request packet. If this is a response packet, then this field shall be vendor-defined.

## 7.7. Non-Idempotent Request Release (NIRR)

The Non-idempotent Request Release is used to release resources required to support the retransmission of non-idempotent request packets, e.g., Atomics, Buffers, Pattern, and reliable Write MSG requests packet. See *Non-idempotent Request (NIR)*.

The following are the features and requirements associated with NIRR:

- Any component type may support the NIRR packet.
- Core 64 NIRR packets and CTXID NIRR packets shall be acknowledged by Core 64 *Standalone Acknowledgments*.
- A Core 64 NIRR packet shall use the *Core 64 NIRR Packet Format*.
- A Core 64 NIRR packet shall be used only to release resources associated with operations whose request packet format does not contain a RSPCTXID field, e.g., an Atomic or buffer request packet. In contrast, a CTXID NIRR packet shall be used only to release resources associated with operations whose request format contains a RSPCTXID field, e.g., a Write MSG packet.
- A CTXID NIRR packet shall use the *CTXID NIRR Packet Format*.
- Core 64 NIRR implementations may use the NIR Type field and CTXID NIRR implementations may use the RSPCTXID to facilitate location of the corresponding non-idempotent resources to be released.

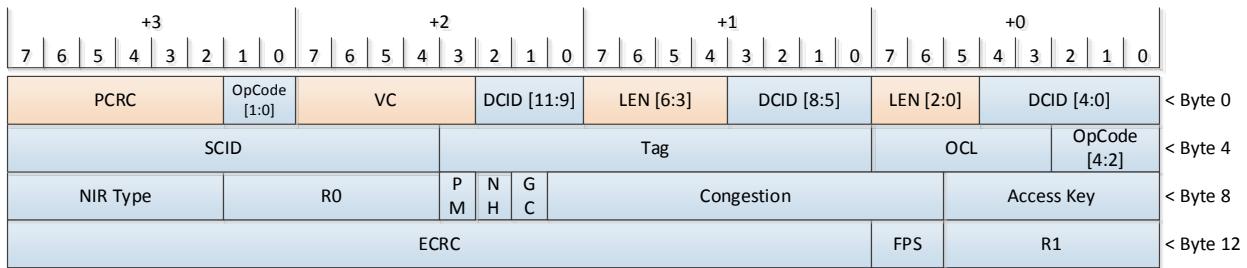


Figure 7-64: Core 64 NIRR Packet Format

Table 7-51: Core 64 NIRR Packet-specific Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>NIR Type</b>	NIR Type	6	<p>Non-idempotent type used to identify the resource type to be released.</p> <p>0x0—Atomics</p> <p>0x1—Buffer</p> <p>0x2—Pattern</p> <p>0x3-0x37 Reserved</p> <p>0x38—Vendor-defined Release corresponding to Vendor-defined OpClass 1</p> <p>0x39—Vendor-defined Release corresponding to Vendor-defined OpClass 2</p> <p>0x3A—Vendor-defined Release corresponding to Vendor-defined OpClass 3</p> <p>0x3B—Vendor-defined Release corresponding to Vendor-defined OpClass 4</p> <p>0x3C—Vendor-defined Release corresponding to Vendor-defined OpClass 5</p> <p>0x3D—Vendor-defined Release corresponding to Vendor-defined OpClass 6</p> <p>0x3E—Vendor-defined Release corresponding to Vendor-defined OpClass 7</p> <p>0x3F—Vendor-defined Release corresponding to Vendor-defined OpClass 8</p>
<b>R0</b>	-	6	Reserved
<b>R1</b>	-	6	Reserved

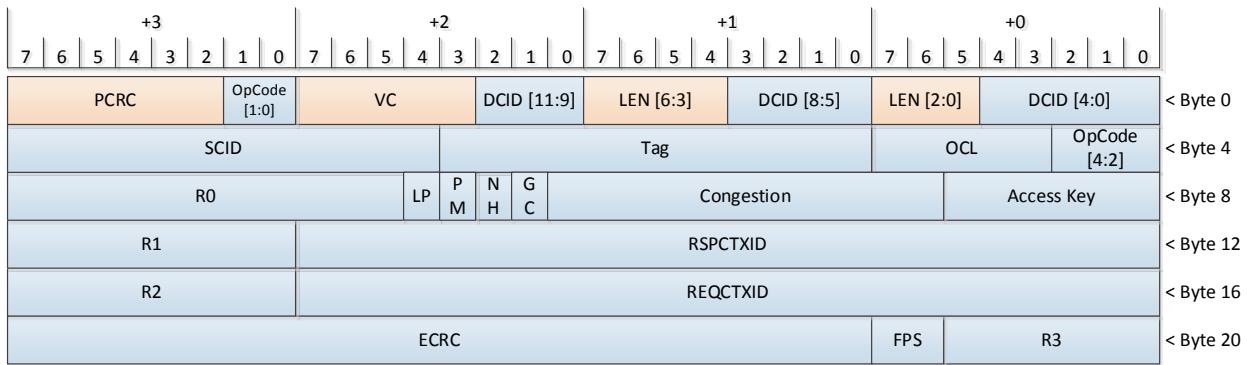


Figure 7-65: CTXID NIRR Packet Format

Table 7-52: CTXID NIRR Packet-specific Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>RSPCTXID</b>	-	24	Identifies the Responder context used to locate an anonymous destination receive buffer at the Responder.
<b>R0</b>	-	11	Reserved
<b>R1</b>	-	8	Reserved
<b>R2</b>	-	8	Reserved
<b>R3</b>	-	6	Reserved

## 7.8. Precision Time

Precision Time is a mechanism (based on IEEE 1588) to distribute a common Master Time value among a set of components; the component set is referred to as a Precision Time Domain (PTD). There are three Precision Time component types:

- Grandmaster Time Component (GTC)—a GTC acts as the primary reference for Master Time within a given PTD. A GTC may be initialized at power-up or be synchronized with an external time mechanism (e.g., *International Atomic Time*); the mechanism to do so is outside of this specification’s scope. Multiple GTC may be synchronized using a common external time mechanism or the set of GTCs may be configured as a separate PTD. Within a given PTD, a GTC acts as a Precision Time Responder.
- Slave Time Component (STC)—a STC acts only as a Precision Time Requester, i.e., it does not distribute Master Time to other components. In general, a STC is a leaf component such as a processor, I/O device, or memory component.
- Boundary Time Component (BTC)—a BTC acts as a Precision Time Requester and a Precision Time Responder in that it distributes Master Time. A BTC is connected to any mix of GTC, BTC, and STCs. In general, a BTC is an intermediate component such as a switch or TR.

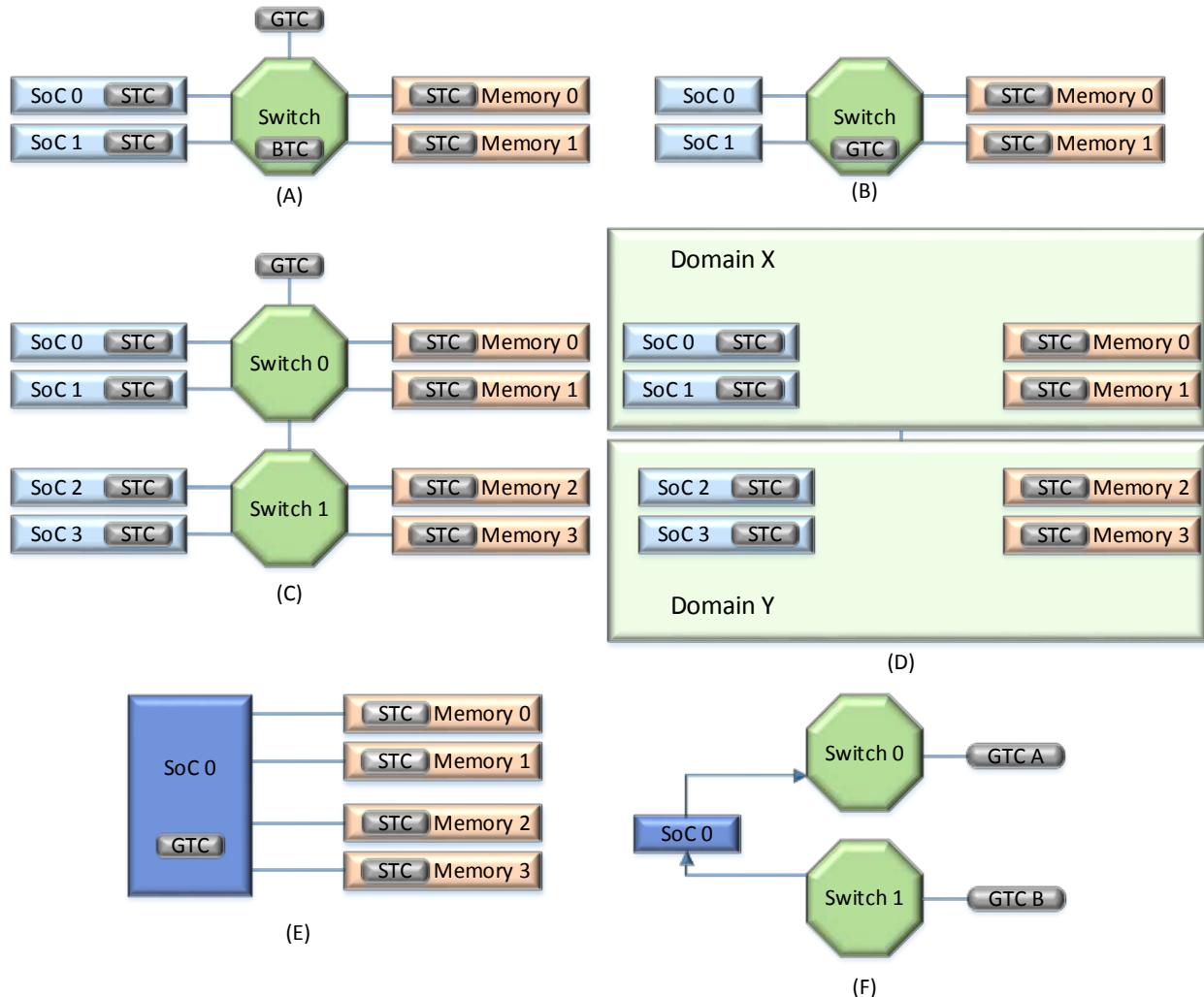


Figure 7-66: Example Precision Time Distribution Topologies

*Example Precision Time Distribution Topologies* illustrates several topologies that support Precision Time. Topologies A-C illustrate single PTDs with discrete and switch-integrated GTC, and topology D illustrates two PTDs operating over a shared physical topology. Topology E illustrates a single SoC that acts as a GTC for a set of memory modules, and topology F illustrates a SoC attached to two independent PTDs.

Precision Time distribution is stepwise, i.e., the GTC and each directly-attached BTC and STC components are synchronized and then each BTC is synchronized with the next set of directly-attached BTC and STC components. *Example Precision Time Distribution (A)* illustrates synchronization is first achieved between the GTC and the directly-attached BTC switch. Upon synchronization, the BTC switch is synchronized with each directly-attached STC SoC and memory component (see (B)). Within a multi-interface component, Master Time distribution to each interface is outside of this specification's scope.

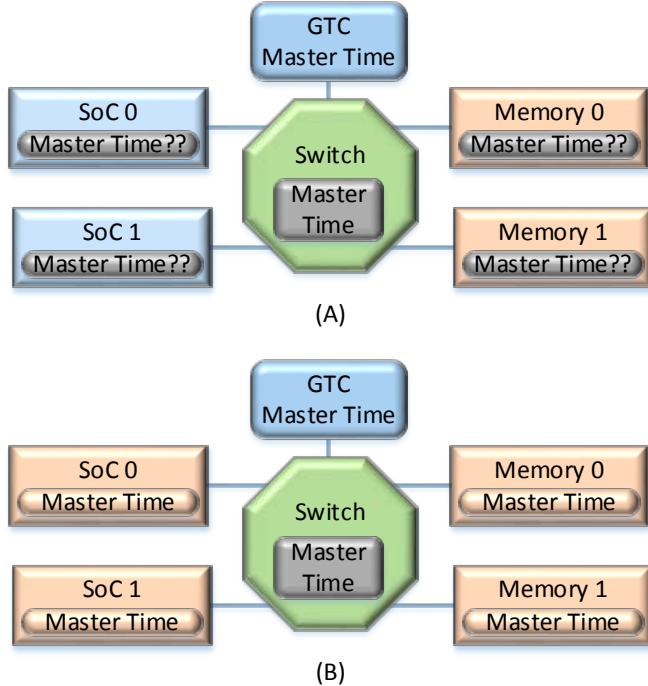


Figure 7-67: Example Precision Time Distribution

*Example Precision Time Request-Response Exchange* illustrates how Precision Time packets are exchanged. As these packets are exchanged, timestamps ( $T_1-T_4$  and  $T_1'-T_4'$ ) are captured (see *Timestamp Measurement between  $T_2$  and  $T_3$* ). Timestamp accuracy will vary due to component design and operating conditions; as a result, Precision Time accuracy will vary. Ideally, timestamps  $[T_2, T_3, T_2', T_3']$  are captured per interface as close to the receipt of bit 0 of byte 0 of a Precision Time Request packet and as close to the transmission of bit 0 of byte 0 of a Precision Time Response packet. Similarly, timestamps  $[T_1, T_4, T_1', T_4']$  are captured per interface as close to the transmission of bit 0 of byte 0 of a Precision Time Request packet and as close to the receipt of bit 0 of byte 0 of a Precision Time Response packet.

It should be noted that synchronized time accuracy is dependent upon the accuracy of the GTC and BTC that distribute Master Time as well as timestamp accuracy in GTC, BTC, and STCs.

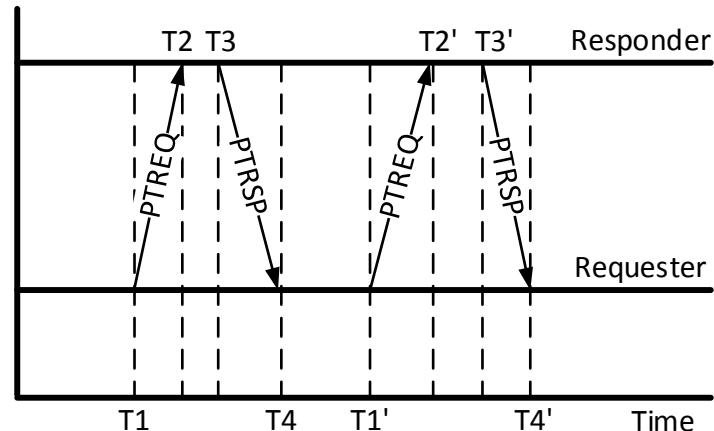


Figure 7-68: Example Precision Time Request-Response Exchange

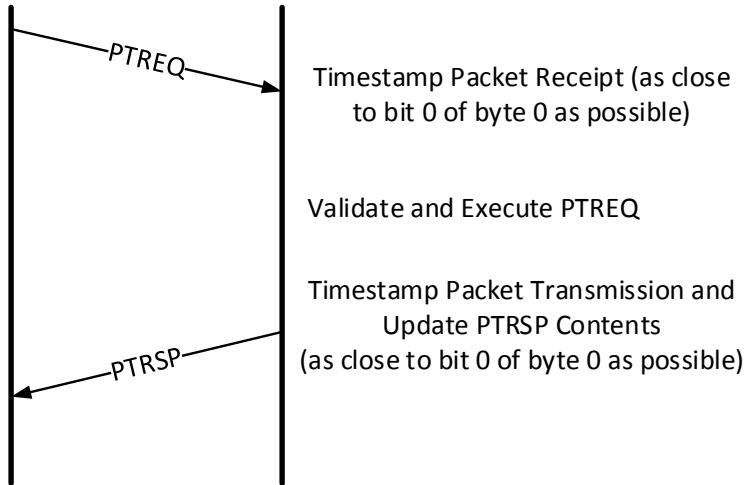


Figure 7-69: Timestamp Measurement between T2 and T3

To calculate the Master Time at time  $T1'$  (i.e., to synchronize Master Time between the two components), timestamps  $[T1-T4, T2']$  from two request-response exchanges are required. The following formula is used:

$$\text{Master Time at } T1' = T2' - ((T4 - T1) - (T3 - T2)) / 2$$

The following are the features and requirements associated with Precision Time:

- Any component type may support Precision Time.
- A component may act as a Precision Time Requester, a Responder, or as a Requester-Responder.
- All components participating in a given PTD shall support Precision Time Request (PTREQ) and Precision Time Response (PTRSP). Precision Time distribution within a given component (e.g., among component services or interfaces) is outside of this specification's scope.
- PTREQ packet shall use the *Precision Time Request Format*.
- PTRSP packet shall use the *Precision Time Response Format*.
- A given PTD shall contain at least one active GTC. All participating components shall be configured with the CID | Global CID of the GTC.
  - PTREQ and PTRSP packets shall contain the CID | Global CID of the GTC used as the Precision Time source. This enables multiple GTC to co-exist within a single topology or for a Requester to detect that a PTD has migrated (planned or unplanned) to a new GTC.
- Once Precision Time is enabled, a Requester shall transmit a PTREQ at least once every 1000 ms and no more than once every 100 us. The one exception is whenever Precision Time is being established, a Requester shall transmit a new PTREQ at least 1  $\mu$ s after receiving a PTRSP with the associated NP bit set to 1b.
  - If a component transitions to C-Down or C-LP or if the interface used to exchange Precision Time transitions to I-Down or I-LP, then the component shall reset the stored timestamps of all impacted interfaces and re-establish Precision Time upon transitioning to C-Up or I-Up.
  - If a component detects through implementation-specific means that time has drifted beyond an acceptable range (e.g. accumulated clock PPM drift), then the component shall invalidate its current Precision Time, reset all stored timestamps, and re-establish Precision time.
  - If Requester detects a new GTC CID | Global CID, then the Requester shall invalidate its current Precision Time, reset all stored timestamps, and re-establish Precision Time.

- A Requester shall have only a single outstanding PTREQ Request at any given time.
- A PTREQ is positively acknowledged by a PTRSP or negatively acknowledged by a Standalone Acknowledgement indicating the associated error.
- PTREQ and PTRSP packets shall use VC0. Given PTREQ and PTRSP packets are only exchanged between directly-attached components, the Access Key field shall be set to zero.
- Timestamps shall be based on the transmission or receipt of bit 0 of byte 0 of a PTREQ or PTRSP packet. If direct measurement is not possible, a component may use an implementation-specific method to predictably adjust each timestamp.
  - Timestamps shall be stored per Requester-Responder pair, e.g., the switch in *Example Precision Time Distribution Topologies* (A) shall store 5 sets of timestamps—one per egress interface. The timestamps stored depends upon the component's role—Requester, Responder, or Requester-Responder.
  - A Requester shall update its stored T1 timestamp whenever a PTREQ packet is transmitted (new or retransmitted request) and shall update its stored T4 timestamp upon receipt of a PTRSP packet.
  - A Responder shall update its stored T2 timestamp upon receipt of a PTREQ packet and shall update its T3 timestamp upon transmission of a PTRSP packet. If the Responder's stored timestamps T2 and T3 are uninitialized or if the Responder needs to update its own Master Time, then shall set PTRSP Transmit PTREQ (TP) = 1b.
- A GTC shall initialize its Master Time (a 64-bit unsigned integer) to zero or be synchronized to an external time mechanism prior to executing the first PTREQ Request. Once initialized, the GTC shall increment Master Time by one (modulo 64) per its local clock period; the clock period is referred to as the Precision Time granularity. For example, if the GTC provides a 10 ns Precision Time granularity, then Master Time is incremented by one every 10 ns.
  - Each component communicates its Precision Time granularity through the *Component Precision Time Structure*.
  - Each component is responsible for reconciling its Precision Time granularity with the PTD's granularity to calculate Master Time. For example, if the GTC provides a 1 ns Precision Time granularity for the PTD and the component supports a 10 ns Precision Time granularity then the component needs to understand how this impacts the accuracy of its timestamps as well as its ability to accurately calculate Master Time.
  - GTC Master Time will wrap (see *Time to Master Time Wrap*). Solutions should be cognizant if this can occur and what implications this might have to a given solution or service. For example, if using Master Time to timestamp packets to ensure uniqueness, Master Time could wrap in as little as 213 days with a 1 picosecond GTC Granularity.
- The Propagation Delay is a 32-bit unsigned integer that represents the time between when a PTREQ is received and when a PTRSP is transmitted, e.g., in *Example Precision Time Request-Response Exchange* the Propagation Delay is  $(T3 - T2)$ . The Propagation Delay uses the PTD's granularity, e.g. if  $(T3 - T2) = 10$  ns and the granularity is 10 ns then the Propagation Delay is 1.
- The granularity used for timestamps, Master Time, and Propagation Delay is component-specific and is set to a value between 1-1023 ns or 1-1023 ps. Precision Time including granularity is managed via the *Component Precision Time Structure*.
- A component may timestamp a packet by placing its understanding of Master Time within the Next Header field (applicable to all explicit OpClass packet formats). If present, then the Master Time[63:0] shall be placed in Next Header[63:0] starting with Master Time[0] to Next Header[0] through Master Time[63] to Next Header[63] (inclusive). Unused Next Header bits shall be Reserved.

- A component may include a HMAC value within the Next Header[127:64] field to authenticate Precision Time packets. This ensures only authenticated components can update Master Time. See *Security*.
- If a packet is (re-) transmitted and contains Master Time (regardless of its location within the packet header or payload), then the Master Time value shall be updated just prior to packet transmission to ensure it reflects the most current value.

Table 7-53: Time to Master Time Wrap

GTC Granularity		Time to Master Time Wrap	
<b>1 picosecond</b>		213.5 days	
<b>1 nanosecond</b>		584.6 years	
<b>10 nanoseconds</b>		5846 years	
<b>20 nanoseconds</b>		11691 years	

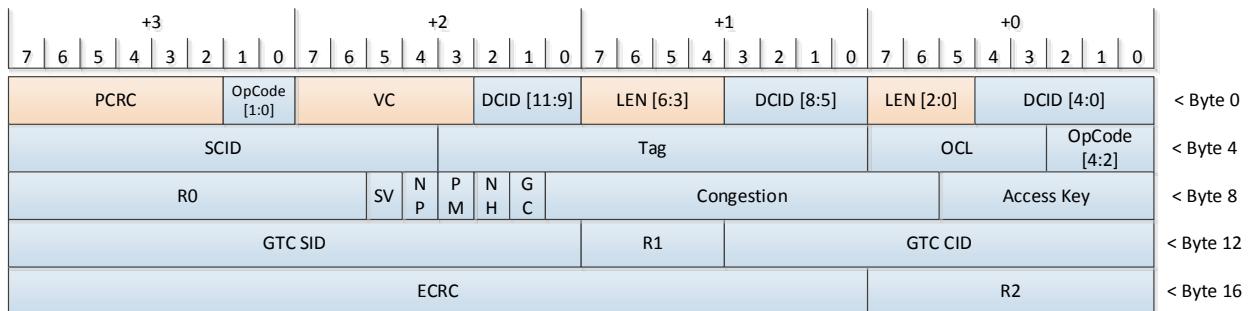


Figure 7-70: Precision Time Request Format

Table 7-54: Precision Time Request Packet-specific Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>GTC CID</b>	-	12	CID of the GTC component associated with this PTD.
<b>GTC SID</b>	-	16	SID of the GTC component associated with this PTD.
<b>NP</b>	NP	1	NP—New PTREQ Sequence Ob—Not a new PTREQ sequence. This PTREQ packet either completes the Precision Time establishment process or is a normal Precision Time refresh. 1b—New PTREQ sequence. This bit shall be set whenever the Requester is (re-) establishing Precision Time. The Requester and the Responder shall reset all relevant timestamps.
<b>SV</b>	-	1	SID Valid—Indicates if the GTC SID field contains a valid SID or is Reserved. Ob—Reserved 1b—Valid SID
<b>RO</b>	-	10	Reserved

Field Name	Field Abbreviation	Size (bits)	Description				
R1	-	4	Reserved				
R2	-	8	Reserved				
< Byte 0 < Byte 4 < Byte 8 < Byte 12 < Byte 16 < Byte 20 < Byte 24 < Byte 28							

Figure 7-71: Precision Time Response Format

Table 7-55: Precision Time Response Packet-specific Fields

Field Name	Field Abbreviation	Size (bits)	Description	
<b>GTC CID</b>	-	11	If the GTC is located within the same subnet as this component, then this field contains the CID of the GTC responsible for Master Time within this PTD. If not co-located, then this field shall be Reserved.	
<b>GTC Global CID</b>	-	32	If the GTC is not located within the same subnet as this component, then this field contains the Global CID of the GTC responsible for Master Time within this PTD. If co-located, then this field shall be Reserved.	
<b>Master Time</b>	-	64	Master Time—64-bit unsigned integer (modulo 64)	
<b>Propagation Delay</b>	-	32	Propagation Delay—32-bit unsigned integer	
<b>TP</b>	-	1	Transmit PTREQ 0b—No PTREQ transmission requested. 1b—New PTREQ transmission requested. Requester shall transmit a new PTREQ at least 1 $\mu$ s after the successful execution of this PTRSP packet in order to complete Precision Time establishment	
<b>GU</b>	-	1	GTC CID Updated	

Field Name	Field Abbreviation	Size (bits)	Description
			0b—GTC CID is unchanged 1b—GTC CID updated
R0	-	9	Reserved
R1	-	4	Reserved
R2	-	8	Reserved

## 7.9. Lightweight Notification (LN)

Lightweight Notification protocol is used by a component to register interest in one or more fixed-sized data blocks (LN Blocks) in another component, and later be notified when the data a LN Block is updated. Notifications provide a lightweight signaling mechanism to reduce software complexity and overhead, synchronization costs, etc.

LN packets are exchanged between an LN Responder (LNRSP) and an LN Requester (LNREQ) as illustrated in *LNREQ-LNRSP Exchange using LN Read Request*:

1. The LNREQ transmits an LN Read Request.
2. If the Address is in a page that supports LN, then the LNRSP registers the LN Block, and returns an LN Read Response packet. Though the registration process is component-specific, conceptually, the LNRSP records the LN Block Address and the [SCID, LN Handle] of each LN Read or LN Write request packet, where the LN Handle identifies associated state at the source component using LN protocol.
  - a. The LN Handle is an opaque field that is returned in future LN Notifications.
  - b. If the Address is not in a memory range supporting LN (not illustrated), then the LNRSP shall transmit a *Standalone Acknowledgment* (Reason = Unsupported Service for Addressed Resource).
3. After some time period, the LN Block is modified. The LNRSP transmits an LN Notification packet to each LNREQ [SCID, LN Handle] that registered interest.
4. The LNREQ transmits an acknowledgment.

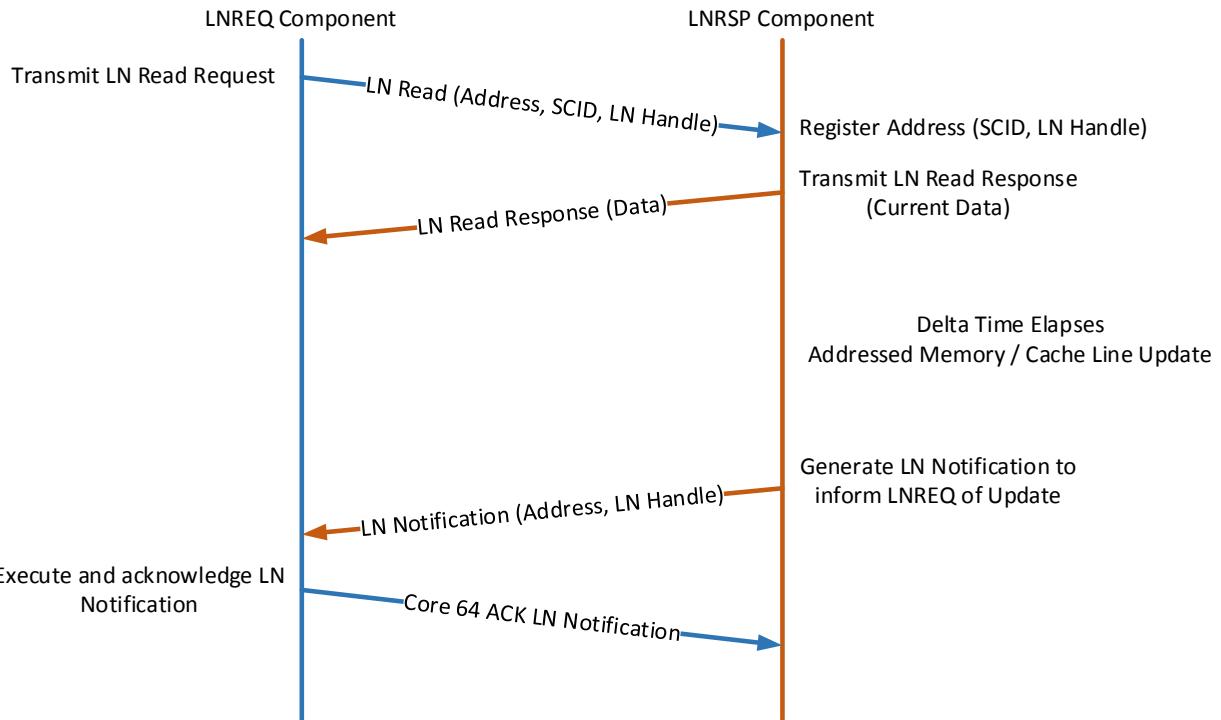


Figure 7-72: LNREQ-LNRSP Exchange using LN Read Request

*LNREQ-LNRSP Exchange using LN Write Request* illustrates a similar exchange using an LN Write:

1. The LNREQ transmits an LN Write Request. The LN Write request contains a zero-length payload, i.e., no payload field, or a fixed-length payload supported by the LNRSP. If zero-length, then this is a request to deregister the LN Block. If non-zero length, then the payload is copied to the LN Block at the location indicated by the packet's Address field.
2. If the Address is associated with a page that supports LN, then the LNRSP registers the Address, and returns a *Standalone Acknowledgment* (Reason = No Error).
  - a. If the Address is not in a page supporting LN (not illustrated), then the LNRSP returns a *Standalone Acknowledgment* (Reason = Unsupported Service for Addressed Resource).
3. After some time period, the LN Block is modified (any portion). The LNRSP transmits an LN Notification packet to each LNREQ that registered interest.
4. The LNREQ transmits an acknowledgment.

5

10

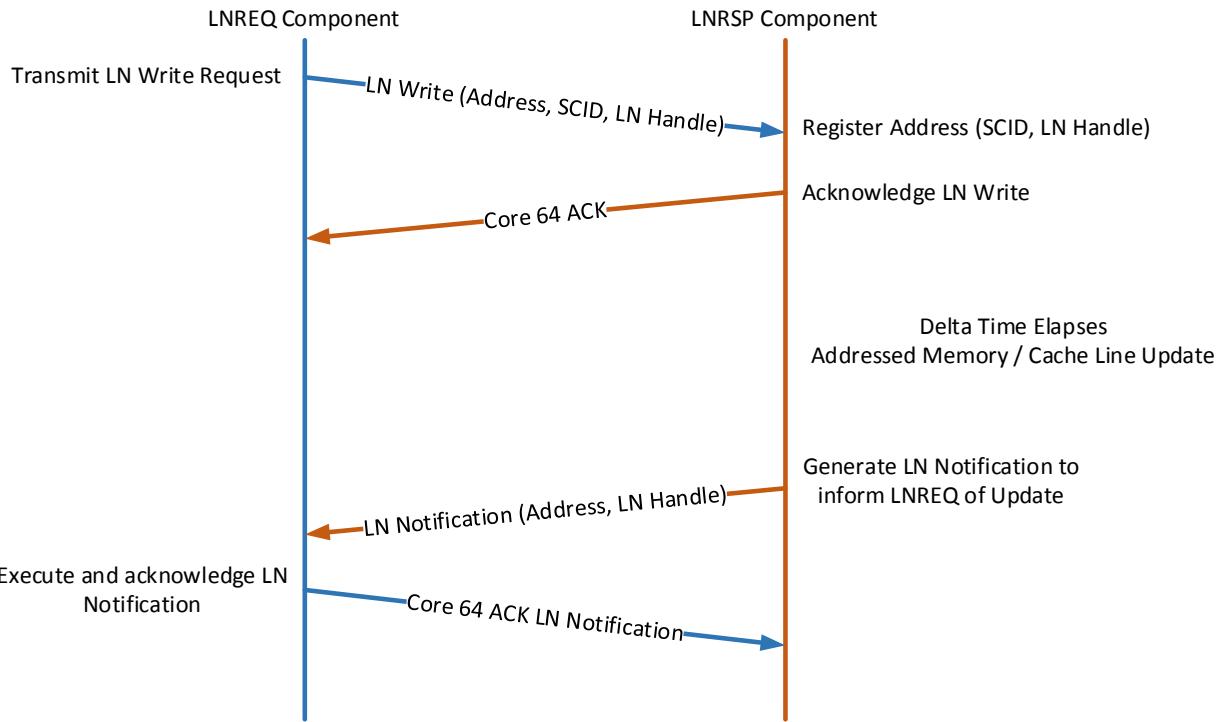


Figure 7-73: LNREQ-LNRSP Exchange using LN Write Request

The following are the features and requirements associated with Lightweight Notification (LN):

- Any component type may support LN.
- A component may be an LNREQ, an LNRSP, or both.
- LN packet formats are as follows:
  - An LN Read packet shall use the *LN Read Request Packet Format*. An LN Read shall use the Lightweight Notification Request OpCode.
  - An LN Read Response packet shall use the *LN Read Response Packet Format*. An LN Read Response shall use the Lightweight Notification Response OpCode.
  - An LN Write packet shall use the *LN Write Packet Format*. An LN Write shall use the Lightweight Notification Request OpCode.
  - An LN Notification packet shall use the *LN Notification Packet Format*. An LN Notification shall use the Lightweight Notification Request OpCode.
- Acknowledgments are as follows:
  - An LN Read 32, LN Read 64, LN Read 128, or LN Read 256 request shall be acknowledged by an LN Read Response packet, or a Core 64 *Standalone Acknowledgment* if an error is detected.
  - Each LN Read Zero, LN Write, LN Write Zero, and LN Notification request packet shall be acknowledged by a Core 64 *Standalone Acknowledgment*.
- LN packets shall set the OCL field to the Advanced 1 OpClass.
- LN services are managed through the *Core Structure Component CAP 3* and *Core Structure Component CAP 3 Control* fields.
- The LN Block address shall correspond to a Data Space address.
- LN Blocks are fixed-sized LNRSP data blocks. The supported LN Block sizes are 32 bytes, 64 bytes, 128 bytes, or 256 bytes.

- An LN Read Zero request is a zero-length read used to probe a page to determine if it supports LN.
  - If supported, then the *Standalone Acknowledgment Request*-specific field shall contain an integer corresponding to the LN Block size supported by the LNRSP. The valid integer values shall be 32, 64, 128, or 256.
  - An LNREQ may issue an LN Read Zero request packet to determine the LNRSP's LN Block size, or use methods outside of this specification's scope to determine this size. If an LNREQ does not support a given LNRSP's LN Block size, then the LNREQ shall not use the LN protocol with this LNRSP.
  - Each page shall support a single LN Block size.
    - If pages overlap, then the overlapping pages shall support the same LN Block size.
    - A component that supports multiple pages may support the same or different LN Block sizes.
- An LN Write request shall be zero-length (indicated by LN OPC = LN Write Zero), or shall contain a payload field whose size is either 32 bytes, 64 bytes, 128 bytes, or 256 bytes.
  - A non-zero payload size is calculated using the following:
    - Payload size = packet length – (protocol bytes)
  - A non-zero payload size shall be equal to the maximum LNRSP LN Block size.
  - A LN Write Zero request shall not have a payload field.
  - A LN Write Zero request shall deregister the LN Block, avoiding a future LN Notification to this LNREQ and freeing up LNRSP resources to handle other registrations.
- The LNRSP and LNREQ shall provision sufficient resources to successfully receive an LN Read Response or an LN Write request based on the LNRSP's LN Block size.
- Pages that support LN registration shall be of no less than 4096 bytes in size. Any LN Block address within this page may be registered.
- The Address returned in the LN Notification packet shall be the LN Block address registered by the LNREQ [LN Handle].
- LN Block address registration:
  - Registration may fail due to:
    - Packet validation errors
    - The addressed page does not support LN.
    - The number of registration requests has exceeded the maximum supported by the LNRSP.
  - An LN Read Zero request shall not register the targeted LN Block.
  - A given LN Block may be registered by one or more LNREQs. For each registration, the LNRSP shall record the minimum to uniquely identify the LNREQ, e.g., the [SCID, LN Handle]. An LNRSP may use implementation-specific means and resources to record this information.
  - An LN notification shall be generated at most once per LN Block Registration (the LN Block might be deregistered before the Notification is triggered).
    - If an LNREQ wants to be informed of future updates to the LN Block, then it shall transmit a new non-zero LN Read or LN Write request packet to register interest. A 1:1 registration-to-LN Notification serves three purposes:
      - It enables an LNRSP to statistically provision registration resources to meet a given application load.

- It enables a given LNREQ to dynamically vary its number of registered LN Blocks without requiring the LNRSP to provision a large number of resources.
- It simplifies registration management—registration failsafe and aging algorithms are not required as the LNRSP can evict (see discussion below) any registration at its discretion.
- LN Notification:
  - The registered LN Block may be modified at any time.
  - Upon modification, for each registered [LNREQ, LN Handle], the LNRSP shall generate an LN Notification packet.
    - The LN Notification packet shall not be transmitted until the LNRSP has completely transmitted all outstanding acknowledgments associated with registrations to this LN Block.
    - Given the potential packet loss, an LNREQ may receive an LN Notification prior to receiving the expected registration acknowledgment. If this occurs,
      - The LNREQ shall release any retransmission resources of the associated LN request packet—retransmission buffer resources, timer resources, etc.
        - If registration acknowledgment arrives during or post release, then the LNREQ shall silently discard the acknowledgment packet.
      - The LNREQ shall wait a minimum of one retransmission cycle before re-using the Tag associated with this specific LN request packet in a new LN request packet.
    - If multiple near-simultaneous modifications are detected to multiple LN Blocks or if multiple registrations are associated with a given LN Block, the LNRSP may transmit corresponding LN Notifications in any order.
    - An LNRSP may evict an LN Block registration at any time. Eviction may be for a single registration (the Address field identifies the evicted registration), or may be for all registrations associated with an LNREQ (the Address field is Reserved).
      - An LNREQ may receive multiple LN Notifications (of one or more types) associated with the same registered LN Block. If this occurs, then the LNREQ shall transmit a *Standalone Acknowledgment* without signaling an error.
      - If evictions are required due to resource shortages, then the component should use a least recently used (LRU) algorithm to remove the oldest registrations.
  - Upon executing an LN Read request packet, the LNRSP shall copy the present contents of the LN Block into the payload of an LN Read Response packet, and transmit the packet to the LNREQ.
    - An LN Read Zero is used to probe a page for LN support without causing a memory registration to occur.
  - Upon executing an LN Write request packet, if non-zero, then the LNRSP shall copy the contents of the packet's payload to the LN Block.

Table 7-56: Common LN Packet Fields

Field Name	Field Abbreviation	Size (bits)	Value	Description	
LN OpCode	LN OPC	4	-	This field indicates LN protocol operation type.	
			0x0	LN Read Zero—Reads 0 bytes	
			0x1	LN Read 32—Reads 32 bytes	
			0x2	LN Read 64—Reads 64 bytes	
			0x3	LN Read 128—Reads 128 bytes	
			0x4	LN Read 256—Reads 256 bytes	
			0x5	LN Read Response	
			0x6	LN Write Zero	
			0x7	LN Write	
			0x8	LN Notification	
			0x9-0xF	Reserved	
Notification Reason	NR	2	-	This field indicates the specific reason for the notification	
				0x0—Update: LN Block was updated 0x1—Eviction: LN Block was evicted 0x2—Eviction: All registered LN Blocks associated with LNREQ were evicted 0x3—Reserved	
LN Handle	-	64	-	Opaque handle sent by the LNREQ packets, and returned in LN Notification packets.	

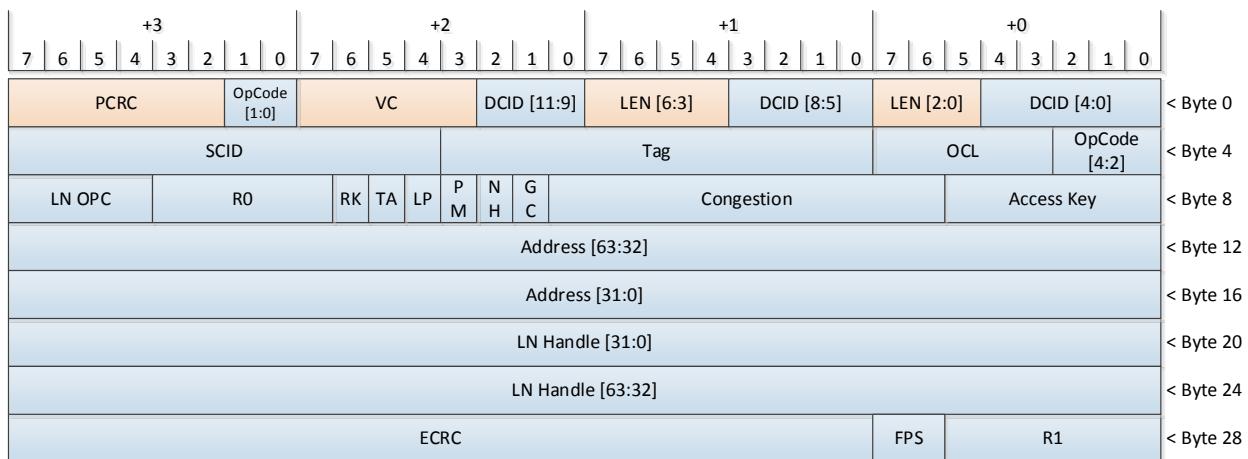


Figure 7-74: LN Read Request Packet Format

Table 7-57: LN Read Request-specific Packet Fields

Field Name	Field Abbreviation	Size (bits)	Value	Description				
R0	5	-	-	Reserved				
	6	-	-					
PCRC   OpCode [1:0]   VC   DCID [11:9]   LEN [6:3]   DCID [8:5]   LEN [2:0]   DCID [4:0]								
SCID			Tag			OCL   OpCode [4:2]		
LN OPC	R0	LP	P M	N H	G C	Congestion   Access Key		
LN Handle [31:0]						< Byte 12		
LN Handle [63:32]						< Byte 16		
Payload (if present)						< Byte 20		
ECRC						R1		
						< Byte YY		

Figure 7-75: LN Read Response Packet Format

Table 7-58: LN Read Response-specific Packet Fields

Field Name	Field Abbreviation	Size (bits)	Value	Description				
R0	-	7	-	Reserved				
	-	8	-					
PCRC   OpCode [1:0]   VC   DCID [11:9]   LEN [6:3]   DCID [8:5]   LEN [2:0]   DCID [4:0]								
SCID			Tag			OCL   OpCode [4:2]		
LN OPC	R0	RK	TA	LP	P M	N H	G C	Congestion   Access Key
Address [63:32]						< Byte 12		
Address [31:0]						< Byte 16		
LN Handle [31:0]						< Byte 20		
LN Handle [63:32]						< Byte 24		
Payload						< Byte 28		
ECRC						FPS   R1		
						< Byte YY		

Figure 7-76: LN Write Packet Format

Table 7-59: LN Write Request-specific Packet Fields

Field Name	Field Abbreviation	Size (bits)	Value	Description	
R0	-	5	-	Reserved	
	-	6	-		
PCRC	OpCode [1:0]	VC	DCID [11:9]	LEN [6:3]	DCID [8:5]
SCID			Tag		
LN OPC	NR	R0	RK	TA	LP
P	M	N	H	G	C
Congestion					
Address [63:32]					
Address [31:0]					
LN Handle [31:0]					
LN Handle [63:32]					
ECRC					
FPS					R1

Figure 7-77: LN Notification Packet Format

Table 7-60: LN Notification Request-specific Packet Fields

Field Name	Field Abbreviation	Size (bits)	Value	Description	
R0	-	3	-	Reserved	
	-	6	-		

## 7.10. Wake Thread

A Wake Thread requests the Responder to awaken the indicated process or thread. If a Responder does not support process or thread services, then the indicated process or thread identifier may represent a Responder-specific resource, and upon receipt, the Responder causes the logic associated with that resource to take action (e.g., to cause a logic block to power-up or initiate execution upon the indicated resource).

If a Responder supports an ISA, then the Responder invokes the corresponding ISA-specific Wake instruction. Invocation may be through execution of a light-weight interrupt service routine to execute the instruction. Alternatively, if a Responder does not support an ISA-specific Wake instruction, then the interrupt service routine may be used to modify the operating system scheduler queue to awaken the process or thread. Multiple implementation options are possible.

The following are the features and requirements associated with a Wake Thread Request:

- Core 64 Wake Thread request packets shall use the *Core 64 Wake Thread Request Packet Format*.
- A Core 64 Wake Thread request packet shall be acknowledged by a *Core 64 Standalone Acknowledgment*.

- If the packet was successfully validated, but the wake operation failed, e.g., the Responder was unable to awaken the corresponding process or thread or the processor or thread does not exist, then the Responder shall return *Standalone Acknowledgment* (Reason = Wake Failure). Upon receipt of a *Standalone Acknowledgment* (Reason = Wake Failure), the Requester should use *Interrupts* to cause the Responder to take action.
- Process or thread identifier communication is outside of this specification's scope.

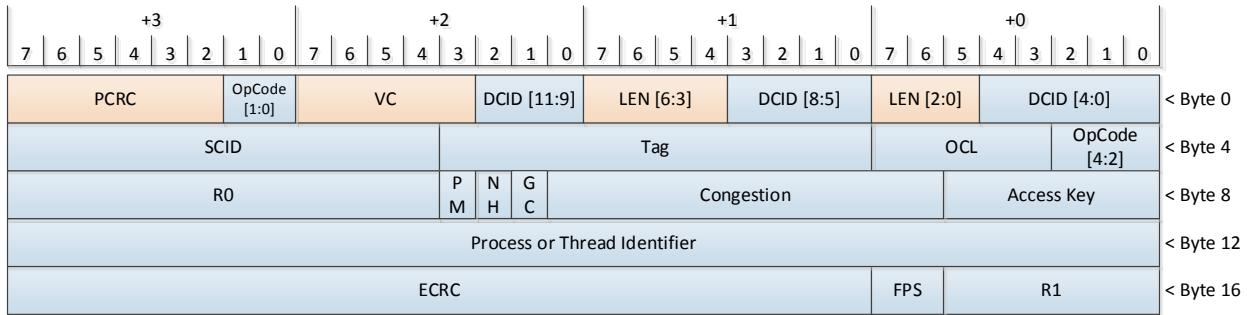


Figure 7-78: Core 64 Wake Thread Request Packet Format

Table 7-61: Core 64 Wake Thread Request Packet Fields

Field Name	Size (bits)	Description
Thread ID	32	The identifier associated with the process or thread of execution that is to be awakened. If the identifier is less than 32 bits, then it shall be placed in the lower bits starting with bit 0 at bit 0. Unused Thread ID upper bits shall be set to zero.
R0	12	Reserved
R1	6	Reserved

## 7.11. Capabilities Read

10 A Capabilities Read request packet enables a Responder to apply additional protection mechanisms to the Read request beyond what would be applied to a normal Read request.

The following are the features and requirements associated with a Capabilities Read Request:

- P2P-Core Capabilities Read request packets shall use the *P2P-Core Capabilities Read Request Packet Format*.
- Core 64 Capabilities Read request packets shall use the *Core 64 Read Request Packet Format*.
- A P2P-Core Capabilities Read request packet shall be acknowledged by a P2P-Core Read Response packet or a *Standalone Acknowledgment* in case of error.
- A Core 64 Capabilities Read request packet shall be acknowledged by a Core 64 Read response packet or a *Standalone Acknowledgment* in case of error.

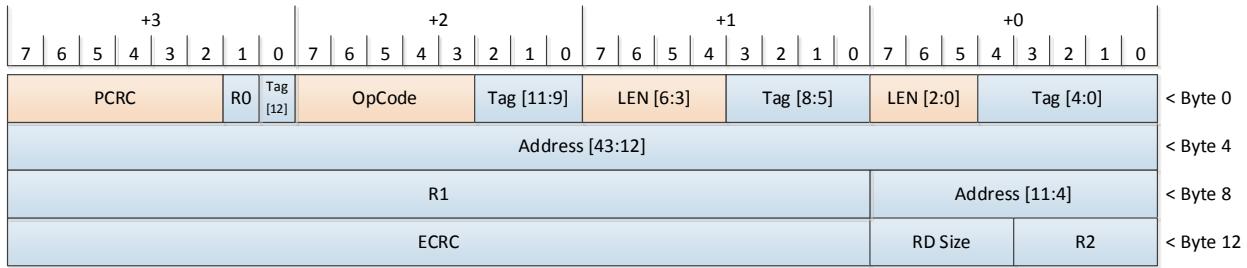


Figure 7-79: P2P-Core Capabilities Read Request Packet Format

Table 7-62: P2P-Core Capabilities Read Request Packet Fields

Field Name	Size (bits)	Description
<b>R0</b>	1	Reserved
<b>R1</b>	24	Reserved
<b>R2</b>	4	Reserved

## 7.12. Capabilities Write

A Capabilities Write request packet enables a Responder to apply additional protection mechanisms to the Write request beyond what would be applied to a normal Write request.

The following are the features and requirements associated with a Capabilities Write Request:

- P2P-Core Capabilities Write request packets shall use the *P2P-Core Meta Write Packet Format*, and the packet's 32-bit Meta field shall be Reserved.
- Core 64 Capabilities Write request packets shall use the *Core 64 Write Request Packet Format*.
- If persistent, then a P2P-Core Capabilities Write request packet shall be acknowledged by a *Standalone Acknowledgment*.
- If UR == 0b or PU = 1b, then a Core 64 Capabilities Write request packet shall be acknowledged by a *Standalone Acknowledgment*.

## 7.13. Unicast Encapsulation Packets

End-to-end packet encapsulation prepends a unicast encapsulation header to an end-to-end packet. The unicast encapsulation header uses the same fields used to transmit and relay a packet to the destination component as a non-encapsulated end-to-end packet.

The following are the features and requirements of the Encapsulation packet:

- Any component type may support Encapsulation packets.
- Unicast Encapsulation packets shall use the *Unicast Encapsulation Packet Format*.
- A Unicast Encapsulation packet shall not encapsulate a Unicast Encapsulation packet, SOD OpClass packet, or Multicast OpClass packet.
- The length of a Unicast Encapsulation packet shall be less than or equal to the maximum supported packet length.
- If applicable, then the *Forward Progress Screen (FPS)* Epoch shall be indicated using the encapsulation packet's FPS field.

- Packet validation is performed in two steps:
    - The first step validates the Unicast Encapsulation packet's protocol fields—the header and tail fields that surround the encapsulated packet. These fields shall be validated as specified in *Destination Component Packet Processing*.
      - The Unicast Encapsulation packet's ECRC shall protect the entire encapsulated packet.
    - The second step validates the encapsulated packet. The encapsulated packet shall be independently validated using encapsulated packet-specific protocol validation.
      - If present, then the encapsulated packet's VC, DCID, PCRC, SCID, Congestion, Access Key, and ECRC fields shall be Reserved.

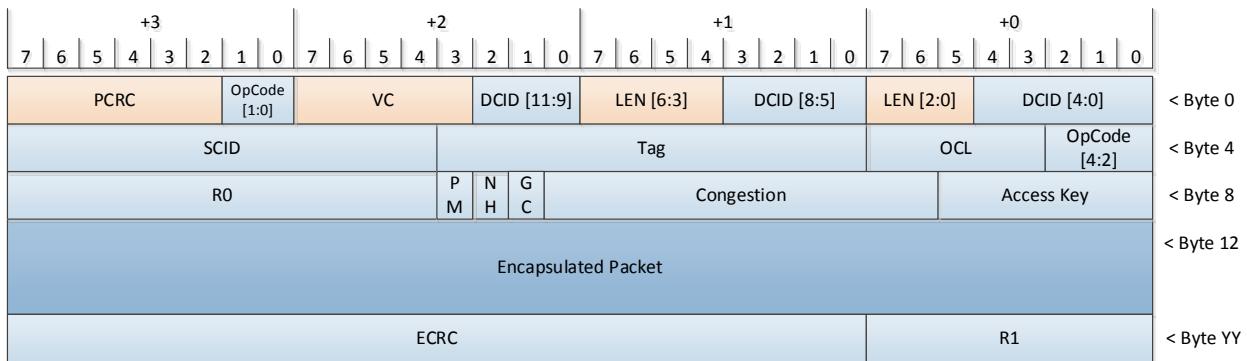


Figure 7-80: Unicast Encapsulation Packet Format  
Table 7-63: Unicast Encapsulation Packet Protocol Fields

Field Name	Size (bits)	Description
R0	12	Reserved
R1	8	Reserved

## 7.14. Coherency-specific Operations

The following are the features and requirements to support coherent communications:

- A Responder that manages a portion of the coherent address space shall support at least one Home Agent.
    - Any component type may support a Home Agent.
    - For each page within the coherent address space, software shall set the corresponding *Responder PTE*'s Page Attributes Cache Coherency Enabled = Enabled.
  - A Requester that initiates coherency request packets and is capable of maintaining copies of data within its own cache structure shall support at least one Caching Agent.
    - Any component type may support a Caching Agent.
    - For each page within the coherent address space, software shall set the corresponding *Requester PTE*'s Page Attributes Target Cache = 1b.
  - The type and number of supported coherency agents is indicated by *Core Structure Component CAP 3 Max Supported Home Agents* and *Max Supported Caching Agents*.

### 7.14.1. Read Exclusive

Read Exclusive is used by components that need to maintain consistent cache state. A Read Exclusive requests a Responder to return an exclusive copy of a single cache line size quantity of data in a Read Response packet. Until released, the cache line is treated as “owned” by the Requester.

5 The following are the features and requirements associated with a Read Exclusive Request:

- Any component type may support Read Exclusive requests. If supported, then the component shall support all OpClass-specific coherency operations.
- A Read Exclusive request packet shall target a Cache Line Size-byte aligned address, where Cache Line Size is the value enabled within the *OpCode Set Structure*.
- 10 • P2P-Coherency Read Exclusive request packets shall use the *P2P-Coherency Read Exclusive / Exclusive / Release / Invalidate Request Packet Format*.
- Core 64 Read Exclusive request packets shall use the *Core 64 Read Exclusive / Exclusive / Release Request Packet Format*.
- 15 • A P2P-Coherency Read Exclusive request packet shall be acknowledged by a P2P-Coherency Read Response packet, or by a *Standalone Acknowledgment* if unable to grant exclusive access or an error is detected.
- A Core 64 Read Exclusive request packet shall be acknowledged by a Core 64 Read Response packet, or by a *Standalone Acknowledgment* if unable to grant exclusive access or an error is detected.
- 20 • Unless explicitly stated otherwise by this specification, Read Response packets shall be as specified in *Read Response Features & Requirements*.
- A Read Exclusive request packet shall be executed only by a Responder that contains the Home Agent for the addressed cache line, else the request packet shall be handled as a MP.
- 25 • If the Responder’s Home Agent is unable to assign exclusive ownership of the data to the Requester, then the Responder shall return a *Standalone Acknowledgment* (Reason = Unable to Grant Exclusive / Shared Access).
- A Requester may give up exclusive control of a cache line using a *Release* or a *Cache Line Attribute* request packet, or by setting RC = 1b in applicable request packets. A Requester may be required to give up exclusive control upon receipt of an *Invalidate and Writeback* or a *Cache Line Attribute* request packet.

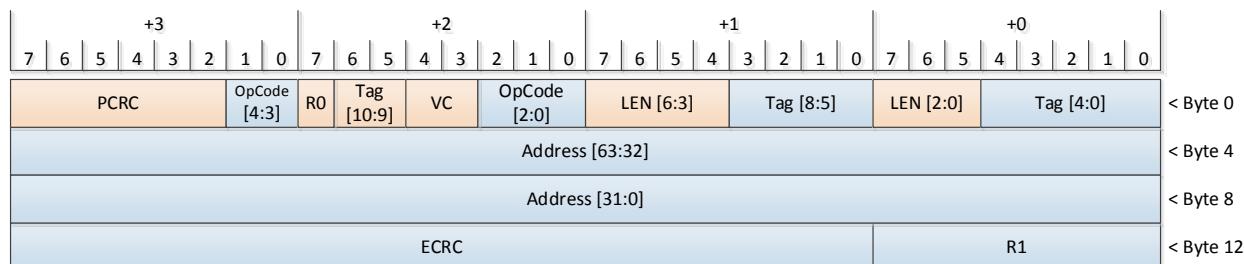


Figure 7-81: P2P-Coherency Read Exclusive / Exclusive / Release / Invalidate Request Packet Format

Table 7-64: P2P-Coherency Read Exclusive / Exclusive / Release / Invalidate Request Packet Fields

Field Name	Size (bits)	Description
Address	64	Target Read Address (integer cache-line size multiple)

Field Name	Size (bits)	Description																			
<b>R0</b>	8	Reserved																			
< Byte 0 < Byte 4 < Byte 8 < Byte 12 < Byte 16 < Byte 20																					

Figure 7-82: Core 64 Read Exclusive / Exclusive / Release Request Packet Format

Table 7-65: Core 64 Read Exclusive / Exclusive Release Request Packet Fields

Field Name	Size (bits)	Description									
<b>R0</b>	1	Reserved									
<b>R1</b>	9	Reserved									
<b>R2</b>	6	Reserved									

### 7.14.2. Read Shared

Read Shared is used by components that need to maintain consistent cache state. A Read Shared requests a Responder to return a non-exclusive copy of a single cache line size quantity of data in a Read Response packet. A Read Shared can also be used to request a copy without requiring cache agents tracking the cache line to update their tracking state.

The following are the features and requirements associated with a Read Shared Request:

- Any component type may support Read Shared requests. If supported, then the component shall support all OpClass-specific coherency operations.
- A Read Shared request packet shall target a Cache Line Size-byte aligned address, where Cache Line Size is the value enabled within the *OpCode Set Structure*.
- P2P-Coherency Read Shared request packets shall use the *P2P-Coherency Read Shared Request Packet Format*.
- Core 64 Read Shared request packets shall use the *Core 64 Read Shared Request Packet Format*.
- A P2P-Coherency Read Shared request packet shall be acknowledged by a P2P-Coherency Read Response packet, or by a *Standalone Acknowledgment* in case of error.
- A Core 64 Read Shared request packet shall be acknowledged by a Core 64 Read Response packet, or by a *Standalone Acknowledgment* in case of error.
- If the Responder's Home Agent is unable to provide a shared copy of the data to the Requester, then the Responder shall return a *Standalone Acknowledgment* (Reason = Unable to Grant Exclusive / Shared Access).

- Unless explicitly stated otherwise by this specification, Read Response packets shall be as specified in *Read Response Features & Requirements*.
- If a Responder simultaneously receives an *Invalidate or Writeback* request packet that targets the same address as a Read Shared request packet, then the Responder shall execute the Invalidate or Writeback request packet prior to executing the Read Shared request packet.
- A shared copy may be flushed / discarded when the component receives an Invalidate request (see *Invalidate and Writeback*).
- A component may inform the Home Agent that it has discarded its shared copy using a *Release*.
- If a Requester needs a clean copy of the cache line (C field), then the Responder shall ensure the cache line is in the clean state prior to returning the corresponding Read Response packet.
- If a Requester needs a cache line that is in any state except the shared dirty state (ND field), then the Responder shall ensure the cache line is not dirty prior to returning the corresponding Read Response packet.
- The C field and the ND field shall not be set to 1b at the same time within a Read Shared request packet.
- If a Requester will not cache the cache line (response data), then the Requester sets SU = 0b (signals that Home Agents and Caching Agents do not need to update their cache tracking logic).
- If the cache line state is invalid, then the Responder may grant exclusive access to the Requester. If granted, then Responder shall transmit a Read Response (Reason = Exclusive Granted).

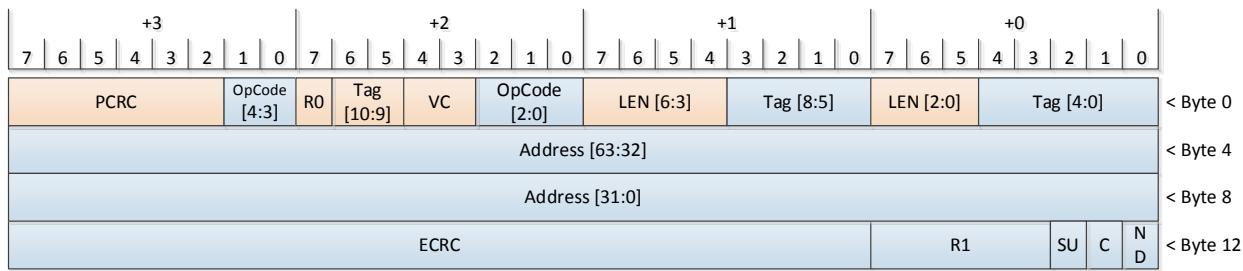


Figure 7-83: P2P-Coherency Read Shared Request Packet Format

Table 7-66: P2P-Coherency Read Shared Request Packet Fields

Field Name	Size (bits)	Description
<b>Address</b>	64	This field provides integer 32-byte address granularity.
<b>C</b>	1	Indicates requested cache line state 0b—Any cache line state 1b—Clean cache line state
<b>ND</b>	1	Indicates requested cache line may be in any state except shared dirty 0b—Any cache line state 1b—Any cache line state except shared dirty
<b>SU</b>	1	Shared Update—Determines if cache agents tracking the requested cache line need to update their shared state. 0b—No Update 1b—Update
<b>R0</b>	1	Reserved

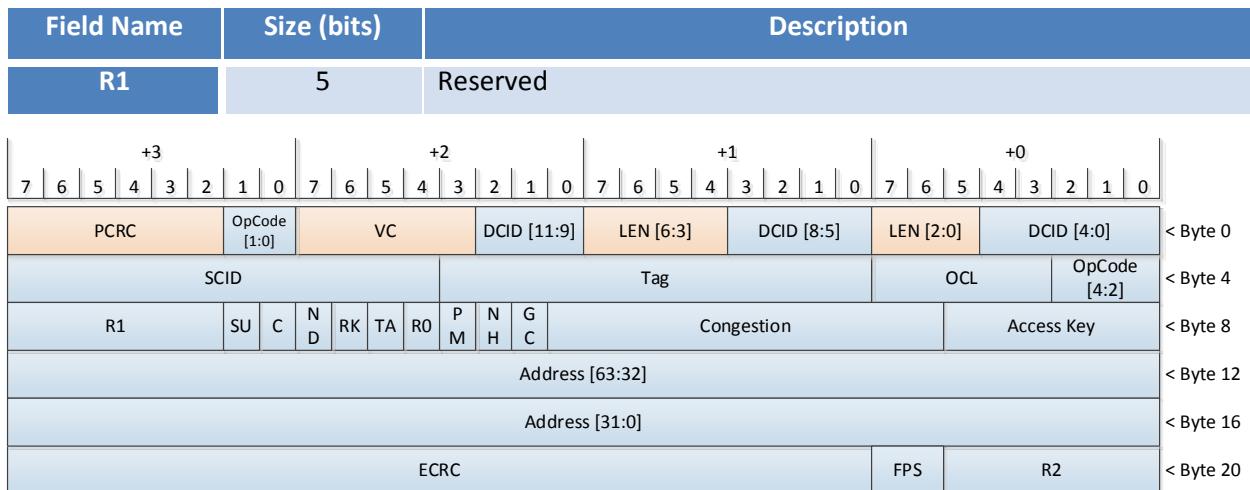


Figure 7-84: Core 64 Read Shared Request Packet Format

Table 7-67: Core 64 Read Shared Request Packet Fields

Field Name	Size (bits)	Description
C	1	Indicates requested cache line state 0b—Any cache line state 1b—Clean cache line state
ND	1	Indicates requested cache line is any state except shared dirty 0b—Any cache line state 1b—Any cache line state except shared dirty
SU	1	Shared Update—Indicates if cache agents tracking the requested cache line need to update their shared state. 0b—No Update 1b—Update
RO	1	Reserved
R1	6	Reserved
R2	6	Reserved

### 7.14.3. Release

Release is used by components that need to maintain consistent cache state. A Release informs a Responder that the Requester no longer has a copy (Exclusive or Shared) of the addressed cache line.

The following are the features and requirements associated with a Release Request:

- Any component type may support Release requests. If supported, then the component shall support all OpClass-specific coherency operations.
- A Release request packet shall target a Cache Line Size-byte aligned address, where Cache Line Size is the value enabled within the *OpCode Set Structure*. A non-zero Cache Line Size shall be configured, else the request packet shall be treated as a MP.

- P2P-Coherency Release request packets shall use the *P2P-Coherency Read Exclusive / Exclusive / Release / Invalidate Request Packet Format*.
- Core 64 Release request packets shall use the *Core 64 Read Exclusive / Exclusive / Release Request Packet Format*.
- P2P-Coherency release packet shall be acknowledged by a *Standalone Acknowledgment*.
- Core 64 Release request packet shall be acknowledged by a *Standalone Acknowledgment*.
- If the Responder was unaware of the Requester's cache line copy, then the Responder shall treat the request packet as a no-op, and shall return a *Standalone Acknowledgment* (Reason = No Error).

#### 10 7.14.4. Exclusive

Exclusive is used by components that need to maintain consistent cache state. An Exclusive requests a Responder to return exclusive ownership of a single cache line size quantity of data to the Requester, but does not return the cache line data itself as would a *Read Exclusive*.

The following are the features and requirements associated with an Exclusive Request:

- Any component type may support Exclusive requests. If supported, then the component shall support all OpClass-specific coherency operations.
- An Exclusive request packet shall target a Cache Line Size-byte aligned address, where Cache Line Size is the value enabled within the *OpCode Set Structure*.
- P2P-Coherency Exclusive request packets shall use the *P2P-Coherency Read Exclusive / Exclusive / Release / Invalidate Request Packet Format*.
- Core 64 Exclusive request packets shall use the *Core 64 Read Exclusive / Exclusive / Release Request Packet Format*.
- A P2P-Coherency Exclusive request packet shall be acknowledged by a *Standalone Acknowledgment*.
- A Core 64 Exclusive request packet shall be acknowledged by a *Standalone Acknowledgment*.
- An Exclusive request packet shall be executed only by a Responder that contains the Home Agent for the addressed cache line, else the request packet shall be handled as a MP.
- If the Responder's Home Agent is unable to assign exclusive ownership of the data to the Requester, then the Responder shall return a *Standalone Acknowledgment* (Reason = Unable to Grant Exclusive /Shared Access).
- A Requester may give up exclusive control of a cache line using a *Release* or a *Cache Line Attribute* request packet, or by setting RC = 1b in applicable request packets. A Requester may be required to give up exclusive control upon receipt of an *Invalidate* or *Writeback* request packet.

#### 7.14.5. Cache Line Attribute

35 Cache Line Attribute is used by components that need to maintain consistent cache state. It may be used to:

- Request the current cache line state as understood by the Responder, and optionally obtain a Shared copy of the cache line.
- Inform the Responder that the Requester has demoted the cache line state from Exclusive to Shared.

The following are the features and requirements associated with Cache Line Attribute requests and responses:

- Any component type may support Cache Line Attribute request packets. If supported, then the component shall support all OpClass-specific coherency operations.
- A Cache Line Attribute request packet shall target a Cache Line Size-byte aligned address, where Cache Line Size is the value enabled within the *OpCode Set Structure*. A non-zero Cache Line Size shall be configured, else the request packet shall be treated as a MP.
- P2P-Coherency Cache Line Attribute request packets shall use the *P2P-Coherency Cache Line Attribute Request Packet Format*.
- Core 64 Cache Line Attribute request packets shall use the *Core 64 Cache Line Attribute Request Packet Format*.
- P2P-Coherency Cache Line Attribute Response packets shall use the *P2P-Coherency Cache Line Attribute Response Packet Format*.
- Core 64 Cache Line Attribute Response packets shall use the *Core 64 Cache Line Attribute Response Packet Format*.
- A P2P-Coherency Cache Line Attribute request packet shall be acknowledged by a P2P-Coherency Cache Line Attribute Response packet, or by a *Standalone Acknowledgment* in case of error.
- A Core 64 Cache Line Attribute request packet shall be acknowledged by a Core 64 Cache Line Attribute Response packet, or by a *Standalone Acknowledgment* in case of error.
- A P2P-Coherency Cache Line Attribute requests the Responder to return the state of the cache line, and may request a copy of the cache line if it is in the Exclusive or Modified state.
- A Core 64 Cache Line Attribute requests the Responder to return the [CID | GCID] of the current owner of the cache line (if exclusively owned), the [CID | GCID] of the Home Agent, and the state of the cache line, and may request a copy of the cache line if it is in the Exclusive or Modified state.
- If the Requester has demoted the cache line state from Exclusive to Shared, then the Requester shall set DS = 1b. If the Requester did not have Exclusive access, then the Responder shall not treat this as an error.
- If HC = 0b and OC = 0b in a Core 64 Cache Line Attribute Response packet, then the Responder does not know the attributes of the requested cache line, and shall set CL State = Invalid.
- If HC = 1b and OC = 0b in a Core 64 Cache Line Attribute Response packet, then the Responder does not know the current owner of the requested cache line, and shall set CL State = Invalid.
- If OC = 1b in a Core 64 Cache Line Attribute Response packet, then the Requester may directly communicate with the current cache line owner for subsequent cache line actions.

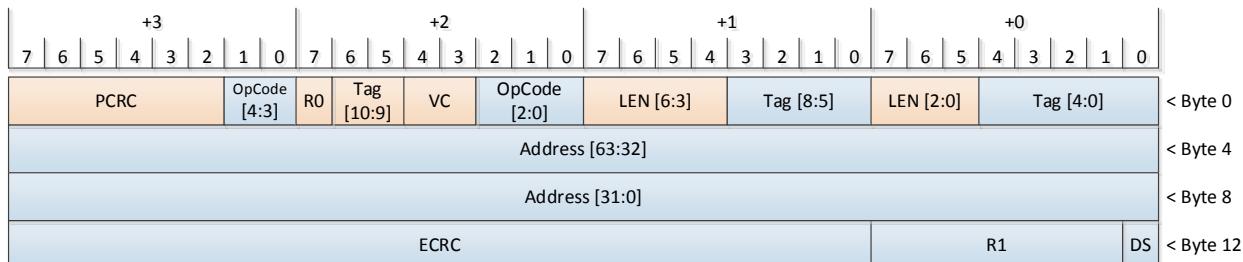


Figure 7-85: P2P-Coherency Cache Line Attribute Request Packet Format

Table 7-68: P2P-Coherency Cache Line Attribute Request Packet Fields

Field Name	Size (bits)	Description
<b>Address</b>	64	This field provides integer 32-byte address granularity.
<b>DS</b>	1	Indicates the Requester has demoted the addressed cache line from Exclusive to Shared or has obtained a Shared copy of the cache line from a different cache agent. 0b—No Cache Line State Change 1b—Shared Cache Line State
<b>R0</b>	1	Reserved
<b>R1</b>	7	Reserved

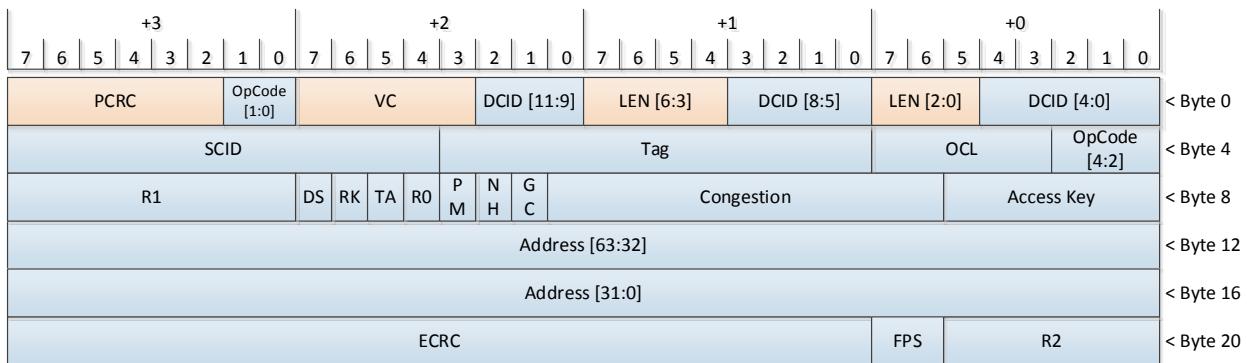


Figure 7-86: Core 64 Cache Line Attribute Request Packet Format

Table 7-69: Core 64 Cache Line Attribute Request Packet Fields

Field Name	Size (bits)	Description
<b>DS</b>	1	Indicates the Requester has demoted the addressed cache line from Exclusive to Shared or has obtained a Shared copy of the cache line. 0b—No Cache Line State Change 1b—Shared Cache Line State
<b>R0</b>	1	Reserved
<b>R1</b>	8	Reserved
<b>R2</b>	7	Reserved

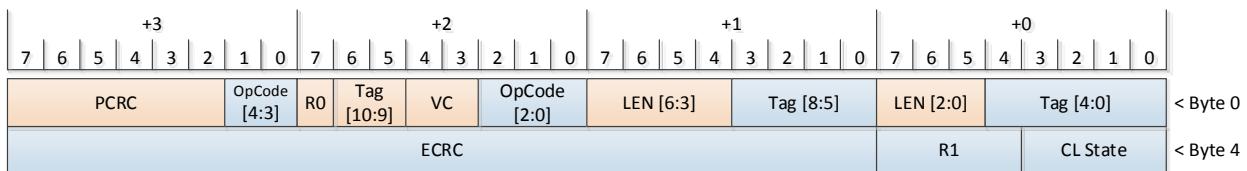


Figure 7-87: P2P-Coherency Cache Line Attribute Response Packet Format

Table 7-70: P2P-Coherency Cache Line Attribute Response Packet Fields

Field Name	Size (bits)	Description
CL State	4	Indicates the Cache Line State 0x0—Modified (Not shared, dirty, must be written back) 0x1—Owned (Shared, dirty, must be written back) 0x2—Exclusive (Not shared, clean) 0x3—Shared (may be clean or dirty, not required to be written back) 0x4—Invalid 0x5-0xF—Reserved
R0	1	Reserved
R1	4	Reserved

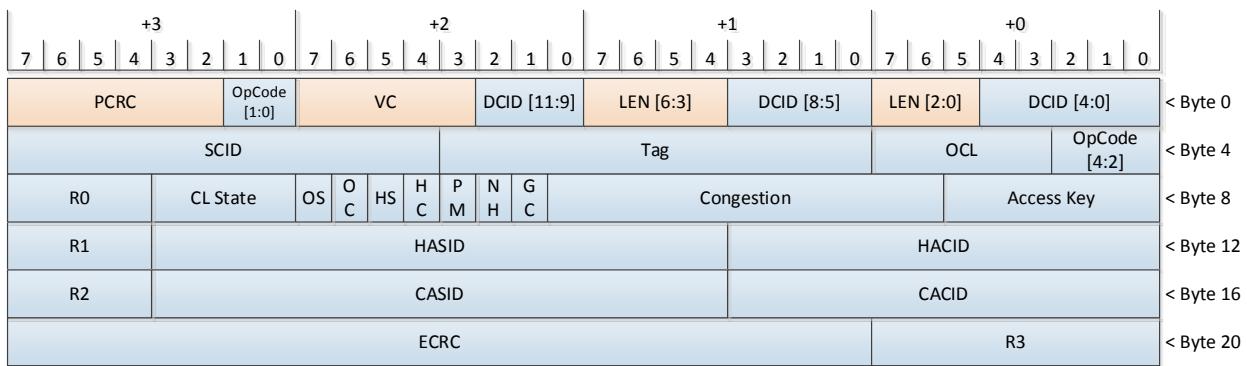


Figure 7-88: Core 64 Cache Line Attribute Response Packet Format

Table 7-71: Core 64 Cache Line Attribute Response Packet Fields

Field Name	Size (bits)	Description
HC	1	Indicates if the HACID field contains a valid CID 0b—Invalid 1b—Valid
HS	1	Indicates if the HASID field contains a valid SID 0b—Invalid 1b—Valid
OC	1	Indicates if the CACID field contains a valid CID 0b—Invalid 1b—Valid
OS	1	Indicates if the CASID field contains a valid SID 0b—Invalid 1b—Valid
CL State	4	Indicates the Cache Line State 0x0—Modified (Not shared, dirty, must be written back) 0x1—Owned (Shared, dirty, must be written back)

Field Name	Size (bits)	Description
		0x2—Exclusive (Not shared, clean) 0x3—Shared (may be clean or dirty, not required to be written back) 0x4—Invalid 0x5-0xF—Reserved
	12	CID of the component (Home Agent) of the target cache line
	16	SID of the component (Home Agent) of the target cache line
	12	CID of the component (Caching Agent / Home Agent) that currently owns the target cache line
	16	SID of the component (Caching Agent / Home Agent) that currently owns the target cache line
	4	Reserved
	4	Reserved
	4	Reserved
	8	Reserved

#### 7.14.6. Invalidate and Writeback

An Invalidate requests a component to invalidate any stored data associated with the specified address or address range independent of the current data state. A Writeback requests a component to write data associated with the specified address or address range to the component(s) where the data's media resides (i.e., its home location) to ensure the memory has the most current values.

The following are the features and requirements associated with Invalidate and Writeback Requests:

- Any component type may support Invalidate and Writeback requests as a Requester, a Responder, or as both a Requester and a Responder. If supported, then the component shall support all OpClass-specific coherency operations.
- P2P-Coherency Invalidate and Writeback request packets target a Cache Line Size-byte aligned address and Cache Line Size multiple address range, where Cache Line Size is the value enabled within the *OpCode Set Structure*.
- Core 64 OpClass Invalidate and Writeback request packets target a Cache Line Size-byte aligned address and Cache Line Size multiple address range, where Cache Line Size is the value enabled within the *OpCode Set Structure*, or an integer 4096-byte aligned address range.
- P2P-Coherency Invalidate request packets shall use the *P2P-Coherency Read Exclusive / Exclusive / Release / Invalidate Request Packet Format*.
- P2P-Coherency Writeback request packets shall use the *P2P-Coherency Writeback Request Packet Format*.
- Core 64 Invalidate request packets shall use the *Core 64 Invalidate Request Packet Format*.
- Core 64 Writeback request packets shall use the *Core 64 Writeback Request Packet Format*.
- Invalidate and Writeback request packets may require a single packet exchange as illustrated in *Example Single Request-Response Packet Exchange*.

- Larger Invalidate and Writeback request packets may require multiple packet exchanges to complete as illustrated in *Example Writeback Request Packet Exchange*.
- Core 64 Invalidate request packets shall be acknowledged by a *Standalone Acknowledgment*.
- If a Responder did not cache the targeted cache line(s) in an Invalidate or Writeback request packet, then the Responder shall treat the request packet as a no-op, and shall return a *Standalone Acknowledgment* (Reason = No Error).
- Core 64 Writeback request packets may be acknowledged by a *Standalone Acknowledgment* or a Read Response packet.
  - If RR = 0b, then a Core 64 Writeback request packet shall be acknowledged by a Core 64 *Standalone Acknowledgment* (Reason = No Error, Contact Home Agent for Cache Line).
  - To avoid a subsequent Read request / Read Response packet exchange with the cache line's Home Agent, the Requester may set the RR = 1b.
    - The Responder shall generate a Read Response packet only for Writeback requests that target a Cache Line Size of data. If the request targets a non-Cache Line Size of data, then the Responder shall execute the Writeback request packet and transmit a Core 64 *Standalone Acknowledgment* (Reason = No Error, Contact Home Agent for Cache Line) in place of a Read Response packet.
    - If successfully executed and the written back data was entirely present within the Responder, then the Responder should transmit a Core 64 Read Response packet. If the written back data is not entirely present within the Responder, then the Responder shall write back any dirty data and transmit a *Standalone Acknowledgment* (Reason = No Error, Contact Home Agent for Cache Line).
    - If the IW == 1b and RR == 1b, then the Read Response shall be generated prior to invalidating the data.
    - If not successfully executed, then the Responder shall transmit a *Standalone Acknowledgment* that indicates the highest precedence error.
- If the total time to execute a Core 64 Invalidate or Writeback request packet and transmit a response packet is longer than *Core Structure COHLAT*, then the Responder shall take the steps specified in *Non-Deterministic Request Execution*
- If a Core 64 Invalidate or Core 64 Writeback request packet includes the R-Key field, then the R-Key associated with the Home Agent's page shall be configured in each Responder ZMMU (or equivalent functionality). The same R-Key shall be used in all request packets required to complete the invalidate operation or writeback operation.

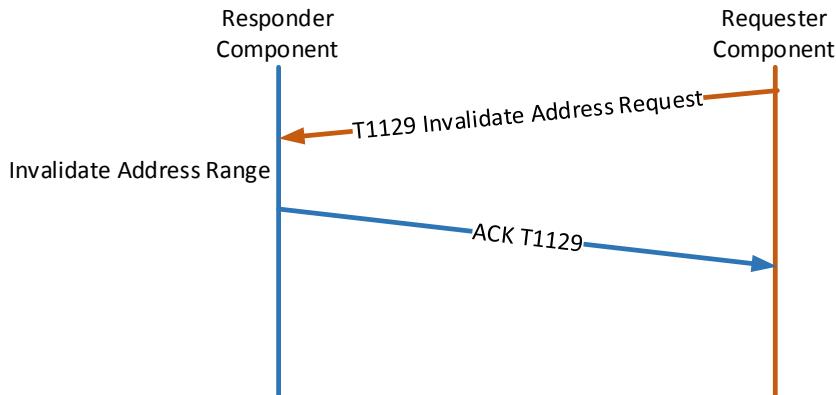


Figure 7-89: Example Single Request-Response Packet Exchange

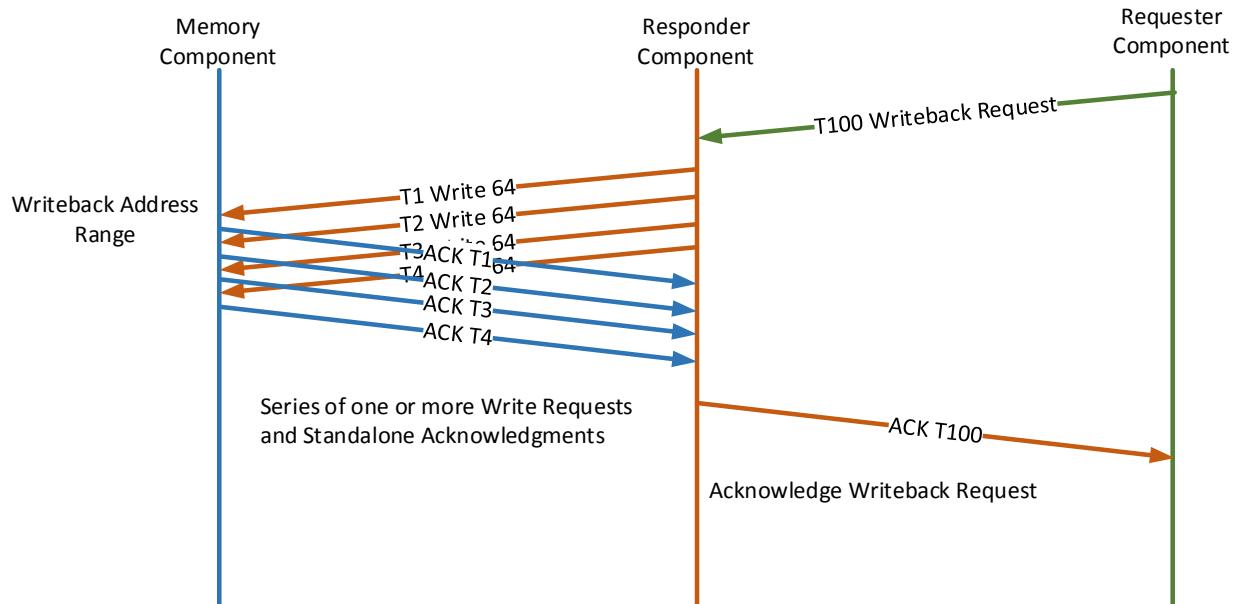


Figure 7-90: Example Writeback Request Packet Exchange

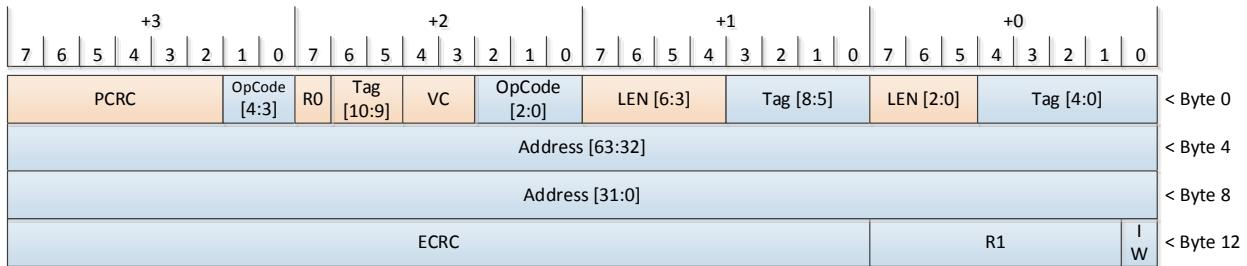


Figure 7-91: P2P-Coherency Writeback Request Packet Format

Table 7-72: P2P-Coherency Writeback Request Packet Fields

Field Name	Size (bits)	Description
<b>Address</b>	39	This field provides integer 32-byte address granularity.
<b>IW</b>	1	Invalidate Post Writeback 0b—The Responder may maintain a local copy of the data upon successful Writeback completion (cache line is in shared state). 1b—The Responder shall invalidate its local copy of the data upon successful Writeback completion.
<b>R0</b>	1	Reserved
<b>R1</b>	7	Reserved

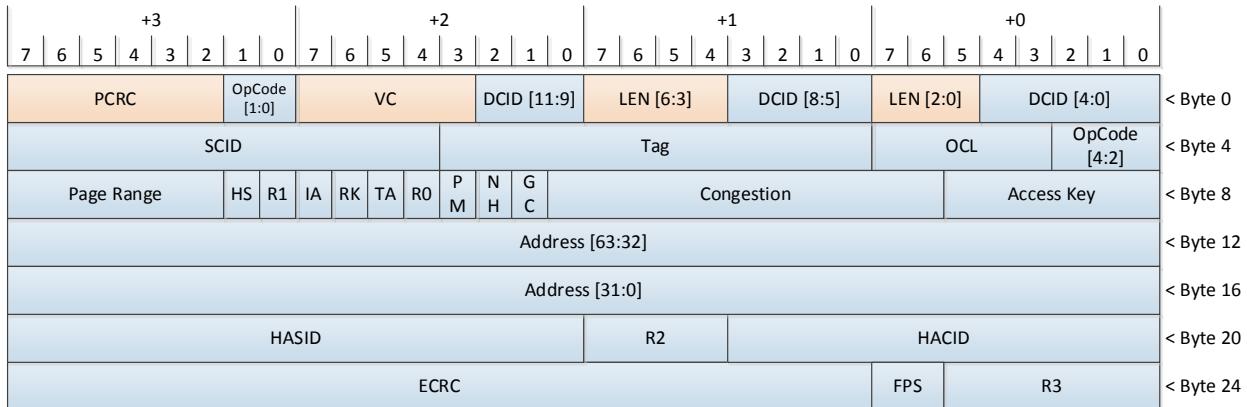


Figure 7-92: Core 64 Invalidate Request Packet Format

Table 7-73: Core 64 Invalidate Request Packet Fields

Field Name	Size (bits)	Description
HS	1	Indicates if the HSID field contains a valid value 0b—Invalid 1b—Valid
HACID	12	CID of the Home Agent Component of the addressed resource, e.g., a cache line
HASID	16	SID of the Home Agent Component of the addressed resource
Page Range	6	Page range to be invalidated or written back. The actual page range is calculated as follows:  Range = Address to (Address + (Page Range * 4096 bytes))  The Address shall be an integer 4096-byte multiple.  If Page Range equals zero, then the request covers a single Cache Line Size of data.
RR	1	Generate Read Response 0b—Responder generates a <i>Standalone Acknowledgment</i> 1b—Responder generates a Read Response with a maximum payload of Max_Packet_Payload bytes.  The payload contains the current copy of the data after written back and prior to any other modifications.  RR shall be Reserved in Invalidate request packets.
IA	1	Independent Acknowledgment 0b—Responder shall generate single <i>Standalone Acknowledgment</i> or a Read Response packet. 1b—Responder shall generate a ND ACK. See <i>Non-Deterministic Request Execution</i> . A separate <i>Standalone Acknowledgment</i> shall be generated once request packet execution has completed. This

Field Name	Size (bits)	Description
R0		
	1	Reserved
	1	Reserved
	4	Reserved
	6	Reserved

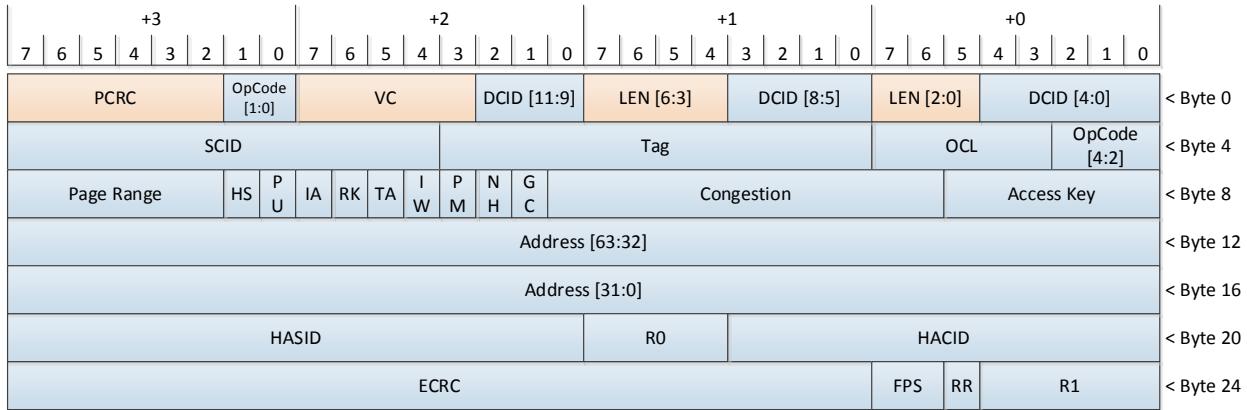


Figure 7-93: Core 64 Writeback Request Packet Format

Table 7-74: Core 64 Writeback Request Packet Fields

Field Name	Size (bits)	Description
<b>HS</b>	1	Indicates if the HSID field contains a valid value 0b—Invalid 1b—Valid
<b>HASID</b>	16	SID of the Home Agent Component of the addressed resource
	12	CID of the Home Agent Component of the addressed resource, e.g., a cache line
<b>Page Range</b>	6	Page range to be invalidated or written back. The actual page range is calculated as follows: Range = Address to (Address + (Page Range * 4096 bytes)) The Address shall be an integer 4096-byte multiple. If Page Range equals zero, then the request covers a single Cache Line Size of data.
<b>RR</b>	1	Generate Read Response 0b—Responder generates a <i>Standalone Acknowledgment</i> 1b—Responder generates a Read Response with a maximum payload of Max_Packet_Payload bytes.

Field Name	Size (bits)	Description
IA		The payload contains the current copy of the data after written back and prior to any other modifications. RR shall be Reserved in Invalidate request packets.
IA	1	Independent Acknowledgment 0b—Responder shall generate single <i>Standalone Acknowledgment</i> or a Read Response packet. 1b—Responder shall generate a ND ACK. See <i>Non-Deterministic Request Execution</i> . A separate <i>Standalone Acknowledgment</i> shall be generated once the request has completed. This is used when the request size is large.
IW	1	Invalidate Post Writeback 0b—The Responder may maintain a local copy of the data upon successful Writeback completion (cache line is in shared state). 1b—The Responder shall invalidate its local copy of the data upon successful Writeback completion.
R0	4	Reserved
R1	5	Reserved

## 7.15. Collective Operations

Collectives are used by applications to coordinate computations or activities across all application components (e.g., a set of SoC or compute engines) participating in a collective group. A collective group can span a subset or all of the application components within a given topology.

5 Gen-Z specifies a set of operations that enable applications to construct a variety of collectives. Further, these operations can be used by collective accelerators within a switch topology to simplify communication and improve solution efficiency. *Example Switch Topology with Collective Accelerators* illustrates an example switch topology. In this example, each switch is directly-attached to a set of computational components. Further, each switch is directly-attached to an optional collective accelerator. To coordinate the compute components, an application on a compute node attached to

10 switch A initiates a collective operation such as a broadcast. In a naïve broadcast collective, the application transmits a Broadcast collective unicast request packet directly to each of the other N-1 compute components participating in the collective. Though simple, this approach is inefficient as communication is limited to the initiating application component's link(s) consuming excessive amount of link bandwidth, and communication consumes an excessive amount of inter-switch link bandwidth.

15

One way to improve efficiency is to implement the broadcast using tree-based communication, i.e., the application that initiated the broadcast collective “tells” two peer components, and in turn, they “tell” two peer components, and so forth. Though this approach increases the effective number of links used to transmit the collective request packets, response efficiency suffers as response communication is limited to the initiating application component's links or additional response packets are required to aggregate peer response packets.

20

An alternative approach is to use a small number of collective accelerators and tiered communications. A collective accelerator is a processing engine that offloads a portion of the collective processing and communications from the application component. In this example, the initiating application transmits a broadcast collective request packet to collective accelerator 0.

- 5 1. The collective request packet contains a collective group identifier and a collective instance identifier. This enables a collective accelerator to simultaneously support multiple collective groups and to delineate collectives.
- 10 2. Collective accelerator 0 acts as the initiating application to the other collective accelerators, and transmits a broadcast collective request packet to collective accelerators 1 and 2. By using tiered communications, the number of request and response packets exchanged across the inter-switch links can be reduced to just 2 request and 2 response packets.
- 15 3. Each collective accelerator issues a broadcast collective request packet to each directly-attached switch leaf component, and then sums the responses. Collective accelerators 1 and 2 return their respective response sums to collective accelerator 0.
- 20 4. Collective accelerator 0 adds these to its own, and returns the total sum to the initiating application component. By using tiered communications, the number of request and response packets exchanged across the application component's link(s) can be reduced to 1 request and 1 response packet. Further, aggregate communication load is reduced since each collective accelerator performs a series of single-hop switch-local packet exchanges instead of the initiating application performing a series of multi-hop packet exchanges.

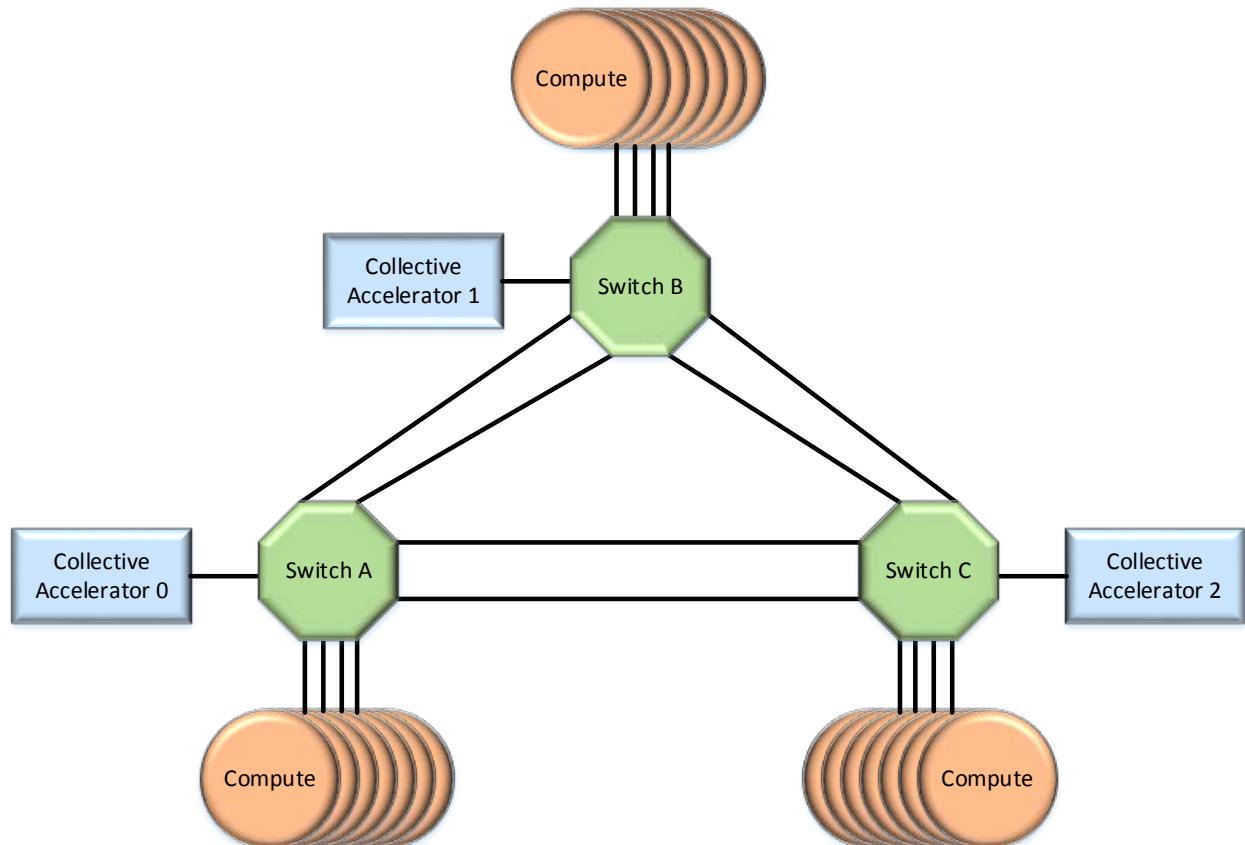


Figure 7-94: Example Switch Topology with Collective Accelerators

The following are the features and requirements associated with Collective operations:

- Each collective group – the set of application instances participating in the collective – shall be identified by a Collective Group value that is included within all collective request and response packets. Collective group management is outside of this specification’s scope (typically handled out-of-band via application-specific logic).
- Any component type may support Collective operations. Many collectives require components to act as Requester-Responders.
- Collective request and response packets shall be unicast packets (see *CTXID OpClass Operation Codes*).
- Applications or middleware that can identify a collective accelerator, should transmit collective request and response packets directly to the accelerator.
- To support collective accelerators where an application cannot identify a collective accelerator, a collective request packet shall be encapsulated within an *Unreliable Multicast Encapsulation Packet*.
  - Switches that do not support multicast packet relay, shall support the Default Collective Egress Interface to relay the multicast packet towards the collective accelerator (see *Switches*).
  - Collective operations that target a multicast group may use the multicast group identifier reserved for collectives—see *Multicast*, or, if the switches support multicast packet relay, may use any supported multicast group identifier.
- Each individual collective operation is identified by a Collective ID that is included within all collective request and response packets. The combination of Collective Group and Collective ID enables applications and collective accelerators to uniquely identify a collective operation within a given topology.
- Collective operations use the CTXID OpClass, and a single OpCode for requests and a single OpCode for responses (see *CTXID OpClass Operation Codes*). Each request or response packet contains a Sub-OPC field that is used to delineate different requests and responses and to indicate the packet format.
- If a topology supports collective accelerators, then:
  - Each collective accelerator shall act as a Responder to the Requester that transmitted the collective request packet. The Requester may be the initiating application component or, if a multi-tier collective accelerator topology, another collective accelerator.
  - Upon receipt of a collective request packet, the collective accelerator shall transmit one collective request packet to each collective Responder assigned to it that is a member of the collective group. Members may be other collective accelerators or leaf Responders.
  - Each collective accelerator shall ensure Reliable Delivery of request packets between itself and each participating Responder.
  - Each collective accelerator shall aggregate the collective-specific results from its assigned set of Responders, and return those to the Requester indicated in the original received collective request packet. For reductions, the accelerator shall perform the reduction on the subset of responses it receives, and send the reduced output to the Requester.
- Applications shall be responsible for synchronizing between invocations of collective operations. If multiple concurrent collective operations access overlapping memory regions, then the behavior is undefined.
- If the time to successfully execute a collective request packet and transmit the corresponding collective response packet exceeds the *Core Structure Responder Deadline*, then the Responder shall handle the collective request packet as specified in *Non-Deterministic Request Execution*.

**Developer Note:** To avoid the additional packet load associated with Non-Deterministic Request Execution, consider configuring all components participating in a collective group as operating in a non-low-latency domain (see Component PA (Peer Attribute) Structure Peer ATTR).

- If a collective needs to be aborted, e.g., due to an application or failure condition, then any participating component may transmit a *Collective Abort* request packet.

5

Common Collective Packet Fields specifies a set of collective protocol fields that are applicable to multiple collective request and response packets.

Table 7-75: Common Collective Packet Fields

Field Name	Size (bits)	Description
Collective Group	32	An integer value that identifies a collective group.
Collective ID	16	An integer value that identifies a specific collective operation.
Sub-OPC	5	<p>OpCode that identifies the operation sub-type</p> <p>0x0—Barrier 0x1—Broadcast 0x2—Scatter 0x3—Gather 0x4—All-Gather 0x5—Reduce 0x6—All-Reduce 0x7—Reduce-Scatter 0x8—Map-Reduce 0x9—All-Map-Reduce 0xA—Collective Abort 0xB—Collective Abort All 0xC—Collective Result 0xD-0x1F—Reserved</p> <p><b>Developer Note:</b> An All-to-All collective is the equivalent of a Scatter collective followed by an All-Gather collective, hence, a separate collective is not required.</p>
Data Type	5	<p>Type of data upon which the collective operates</p> <p>0x0—8-bit signed integer 0x1—8-bit unsigned integer 0x2—16-bit signed integer 0x3—16-bit unsigned integer 0x4—32-bit signed integer 0x5—32-bit unsigned integer 0x6—64-bit signed integer 0x7—64-bit unsigned integer 0x8—128-bit signed integer 0x9—128-bit unsigned integer 0xA—256-bit signed integer</p>

Field Name	Size (bits)	Description
		<p>0x10—256-bit unsigned integer      0x11—16-bit floating point      0x12—32-bit floating point      0x13—64-bit floating point      0x14—128-bit floating point      0x15—256-bit floating point      0x16-0x1D—Reserved      0x1E—Vendor-defined 0      0x1F—Vendor-defined 1</p> <p>Vendor-defined Data Types are identified by the application and are outside of this specification's scope.</p> <p>Vendor-defined Data Types may carry up to Max_Packet_Payload bytes of data.</p>
<b>Collective CID</b>	12	The CID of the component where the data to be accessed is located.
<b>Collective SID</b>	16	The SID of the component where the data to be accessed is located.
<b>RSPCTXID</b>	24	<p>Identifies the Responder context used to locate collective-specific resources at the Responder.</p> <p>RSPCTXID shall be a non-zero value.</p>
<b>REQCTXID</b>	24	<p>Identifies the Requester context used to generate the collective request packet at the Requester.</p> <p>REQCTXID shall be a non-zero value.</p>
<b>CSV</b>	2	<p>If the Collective CID and Collective SID fields are present within the packet, then CSV indicates their contents.</p> <p>0x0—Collective CID is valid; Collective SID is Reserved      0x1—Collective CID and Collective SID are valid      0x2-0x3—Reserved</p>
<b>CH</b>	1	<p>CH indicates if the completion handler associated with the RSPCTXID shall be invoked upon receipt of this packet. If a completion handler is not associated with the RSPCTXID or cannot be invoked for any reason, then this field shall be ignored.</p> <p>A completion handler is a component-local mechanism, e.g., a local interrupt that is associated with the RSPCTXID through a mechanism that is outside of this specification's scope.</p> <p>0b—Do Not Invoke      1b—Invoke</p>
<b>Filter</b>	5	<p>Filter type to apply to a reduction</p> <p>0x0—None      0x1—Less-than      0x2—Less-than-or-equal</p>

Field Name	Size (bits)	Description
Reduce Operator	5	<p>0x3—Equal 0x4—Not-Equal 0x5—Greater-than-or-equal 0x6—Greater-than 0x7-0x1F—Reserved</p> <p>Reduction operation type to perform</p> <p>0x0—MAX (Maximum value) 0x1—MIN (Minimum value) 0x2—SUM (Arithmetic sum, i.e., <math>A = B + C</math>) 0x3—MEAN (Arithmetic mean—sum DIV number of participants) 0x4—PRODUCT (Arithmetic product, i.e., <math>A = B * C</math>) 0x5—LOG-AND (Logical AND; integer operands only) 0x6—BIT-AND (Bitwise AND; integer operands only) 0x7—LOG-OR (Logical OR; integer operands only) 0x8—BIT-OR (Bitwise OR; integer operands only) 0x9—LOG-XOR (Logical XOR; integer operands only) 0xA—BIT-XOR (Bitwise XOR; integer operands only) 0xB—MAXLOC (Location of the participant with maximum value) 0xC—MINLOC (Location of the participant with minimum value) 0xD—COUNT (Count (number of participants that match filter) 0xE-0x1F—Reserved</p> <p>If arithmetic operations overflow, then the results can be non-deterministic. Similarly, if sums of floating point numbers of wildly different magnitude are performed, the results can be non-deterministic: <math>((1E+20 + 1E-20) - 1E+20)</math> will generally resolve to zero, but <math>((1E+20 - 1E+20) + 1E-20)</math> will not. As is normal for collective operations, full commutativity and associativity of operands as well as operations is assumed; deviation from this can lead to non-deterministic results.</p>

### 7.15.1. Barrier Collective

A Barrier collective serves as an application synchronization checkpoint. All application instances participating in the collective group are instructed by a collective initiator to reach a barrier before any proceed beyond it.

- 5 The following steps are taken to perform a Barrier collective:
1. A Requester (collective initiator) transmits a Barrier Collective request packet to each participating Responder (application instance) identified by the Collective Group ID.
  2. Each Responder transmits a *Collective Responder Count Response* packet when it reaches the barrier.

3. Upon receipt of a *Collective Responder Count Response* packet from each Responder, the Requester transmits a *Collective Result* request packet to each Responder. This indicates that all participating application instances have reached the barrier.
4. Each Responder acknowledges the *Collective Result* packet, and the corresponding application instances continue execution.

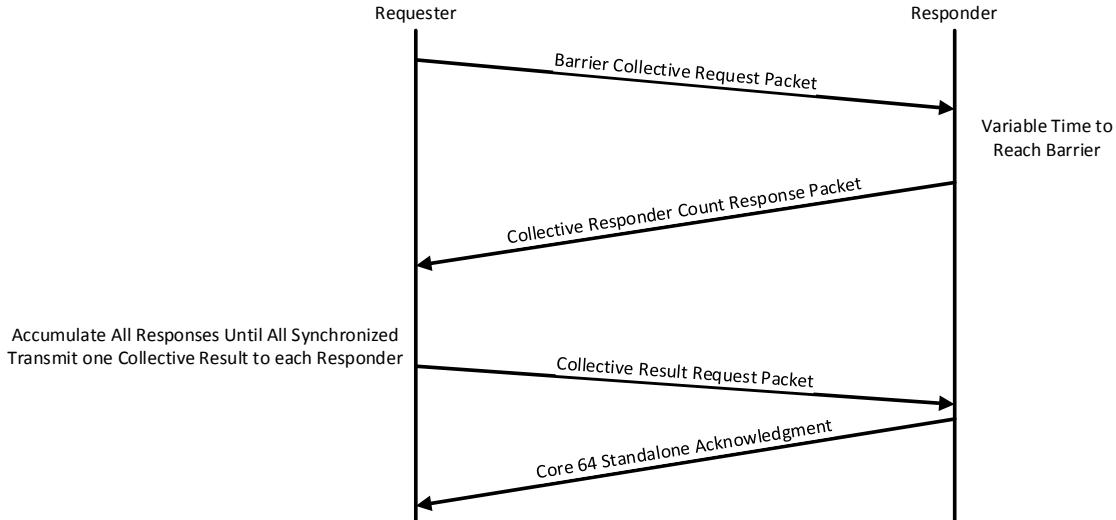


Figure 7-95: Barrier Collective Ladder Diagram

**Developer Note:** If provisioned, collective accelerators can improve Barrier Collective performance by:

- Reducing application synchronization and synchronization release latency (lowers application stall time)
- Reducing the number of packets transmitted between the Requester and all Responders (e.g., an accelerator can aggregate *Collective Responder Count Response* packets and exchange *Collective Result*-acknowledgments for a subset of the Responders). This can also reduce the probability of congestion and jitter.

The following are the features and requirements associated with Barrier Collective and Collective Result request packets:

- Barrier Collective request packets shall use the *Barrier and Abort Collective Request Packet Format*.
- A Barrier Collective request packet shall be acknowledged by a *Collective Responder Count Response* packet (PR shall be set to 1b) or by a *Core 64 Standalone Acknowledgment* in case of error.
- To correlate a *Collective Result* with a Barrier Collective, the *Collective Result* request packet shall use the same Collective ID and Collective Group ID transmitted in the corresponding Barrier Collective request packet. The payload in the *Collective Result* request packet shall not be present, and the Data Type field shall be Reserved.
- If the Responder receives a *Collective Result* request packet and has not reached a barrier, then it shall return may silently discard the packet or transmit a *Responder Not Ready (RNR) NAK*. If the Responder is not participating in an application barrier synchronization checkpoint, then it shall return a *Core 64 Standalone Acknowledgment* (Reason = No Error).

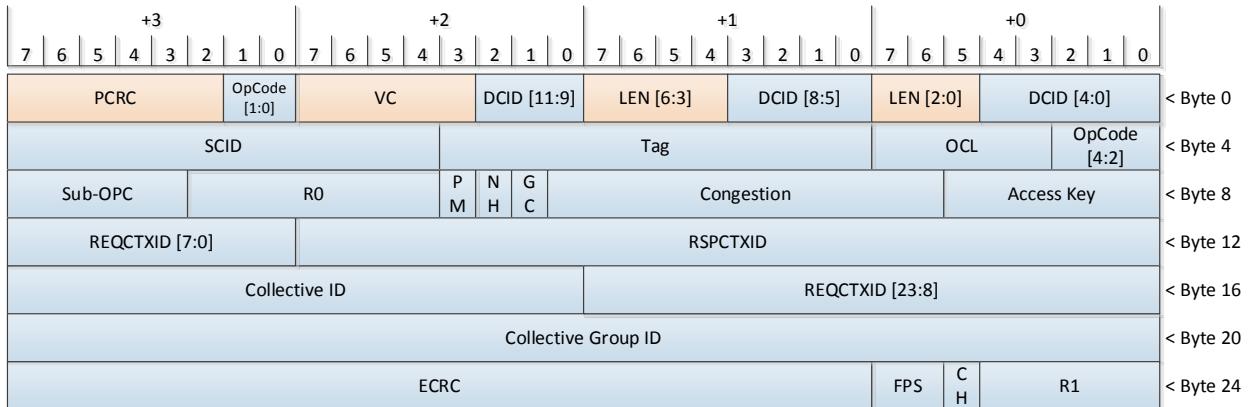


Figure 7-96: Barrier and Abort Collective Request Packet Format

Table 7-76: Barrier and Abort Collective Request Packet Fields

Field Name	Size (bits)	Description
R0	7	Reserved
R1	5	Reserved

## 7.15.2. Broadcast Collective

A Broadcast collective serves to disseminate data from an application instance to all other application instances participating in the collective group.

The following steps are taken to perform a Broadcast collective:

1. A Requester (collective initiator) transmits a Broadcast Collective request packet to each participating Responder (application instance) identified by the Collective Group ID. The Requester copies the data to the request packet's payload field, and sets the corresponding Data Type.
2. Each Responder transmits a *Collective Responder Count Response* packet when it receives the Broadcast Collective request packet.

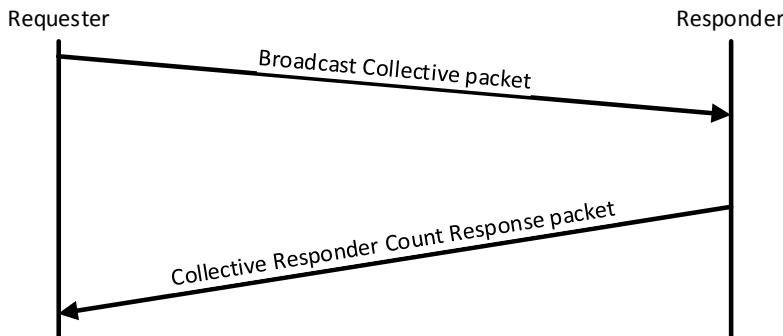


Figure 7-97: Broadcast Collective Ladder Diagram

**Developer Note:** If provisioned, collective accelerators can improve Broadcast Collective performance by:

- *Eliminating serial Broadcast Collective request packet transmission, thus reducing the latency to disseminate the data, which in turn, reduces the latency to receive the response packets and complete the collective.*
- *Reducing the number of packets transmitted between the Requester and all Responders e.g., an accelerator can aggregate Collective Responder Count Response packets for a subset of the Responders). This can also reduce the probability of congestion and jitter.*

5 The following are the features and requirements associated with Broadcast Collective request packets:

- Broadcast Collective request packets shall use the *Broadcast Collective Request Packet Format*.
- A Broadcast Collective request packet shall be acknowledged by a *Collective Responder Count Response packet* (PR shall be set to 1b) or by a *Core 64 Standalone Acknowledgment* in case of error.

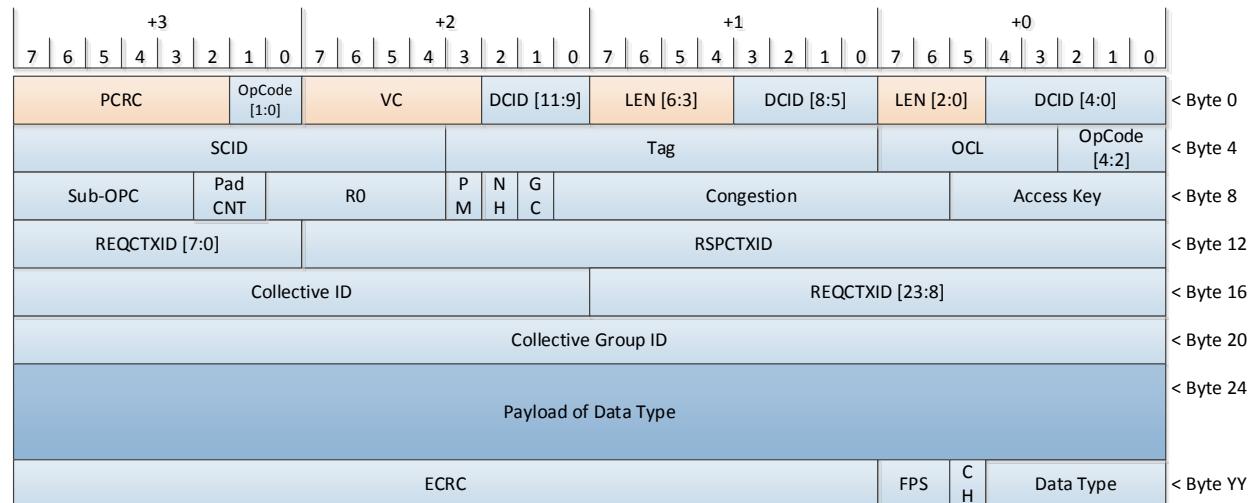


Figure 7-98: Broadcast Collective Request Packet Format

Table 7-77: Broadcast Collective Request Packet Fields

Field Name	Size (bits)	Description
<b>Payload</b>	Variable	The payload shall contain the number of bits indicated by the Data Type. If the payload is not an integer 4-byte multiple, then the payload shall be padded with the number of pad bytes indicated by Pad CNT.
<b>R0</b>	5	Reserved

### 7.15.3. Scatter Collective

15 A Scatter collective enables an application instance to distribute a data set to all other application instances participating in the collective group. Each application instance operates on a subset of the data set.

The following steps are taken to perform a Scatter collective:

- 5
1. A Requester (collective initiator) transmits a Scatter Collective request packet to each participating Responder (application instance) identified by the Collective Group ID. The Requester copies the data set address to the packet's Address field, copies the data set's [CID, SID] to the respective Collective CID and Collective SID fields, and sets the corresponding Data Type.
  2. Each application instance multiplies its rank (e.g., its MPI Rank) by the size indicated by the Data Type to calculate an offset from the data set address to locate its datum.
  3. Using the datum's location, each application instance transmits a Core 64 Read request packet.
  4. Upon receipt of a Core 64 Read Response packet, the application instance transmits a *Collective Responder Count Response* packet to indicate it has received its datum.
- 10

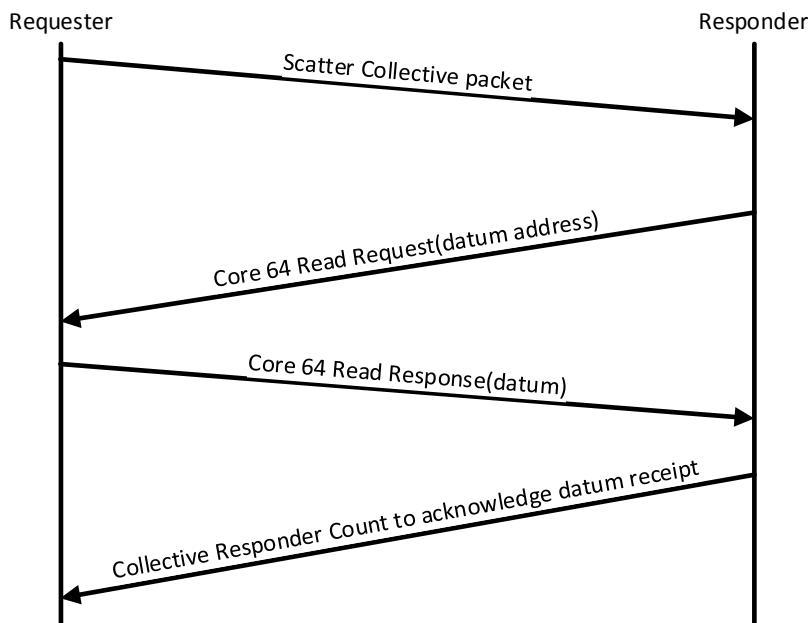


Figure 7-99: Scatter Collective Ladder Diagram

**Developer Note:** If provisioned, collective accelerators can improve Scatter Collective performance by:

- 15
- Eliminating serial Scatter Collective request packet transmission, thus reducing the latency to disseminate the datum location.
  - By reading all of the data from the Requester on behalf the collective accelerators and Responders it services, the accelerator can reduce Requester read load and subsequent read completion latency.
  - Reducing the number of packets transmitted between the Requester and all Responders. This can also reduce the probability of congestion and jitter.
- 20

The following are the features and requirements associated with Scatter Collective request packets:

- 25
- Scatter Collective request packets shall use the *Scatter and Gather Collective Request Packet Format*.
  - A Scatter Collective request packet shall be acknowledged by a *Collective Responder Count Response* packet (PR shall be set to 1b) or by a *Core 64 Standalone Acknowledgment* in case of error.

- If a non-default R-Key is associated with the page where the data set is located, then the Requester shall set RK = 1b and copy the RO R-Key or the RW R-Key associated with the page to the R-Key field.

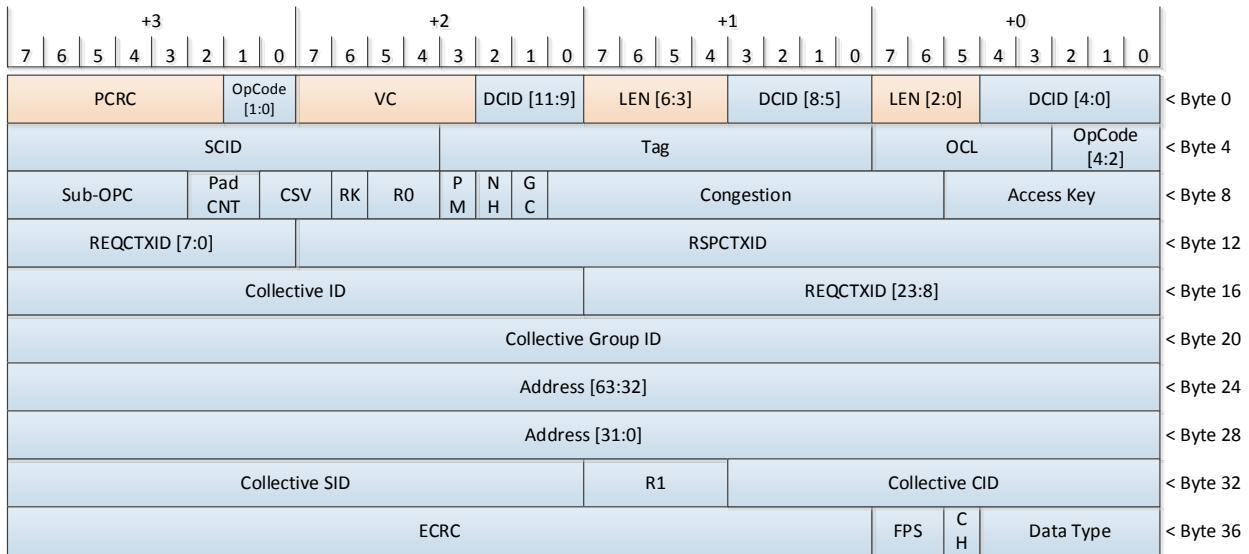


Figure 7-100: Scatter and Gather Collective Request Packet Format

Table 7-78: Scatter and Gather Collective Request Packet Fields

Field Name	Size (bits)	Description
<b>Address</b>	64	This is the address to use in the subsequent read request packet.
<b>R0</b>	2	Reserved
<b>R1</b>	4	Reserved

#### 7.15.4. Gather Collective

A Gather collective enables an application instance to collect datum from multiple application instances and consolidate the datum into a single contiguous data set. A Gather collective is the inverse of the Scatter collective.

The following steps are taken to perform a Gather collective:

1. A Requester (collective initiator) transmits a Gather Collective request packet to each participating Responder (application instance) identified by the Collective Group ID. The Requester copies the data set address to the packet's Address field, copies the data set's [CID, SID] to the respective Collective CID and Collective SID fields, and sets the corresponding Data Type.
2. Each application instance multiplies its rank (e.g., its MPI Rank) by the size indicated by the Data Type to calculate an offset from the data set address to locate where to write its datum.
3. Using the datum's location, each application instance transmits a Core 64 Write request packet.
4. Upon receipt of a Core 64 *Standalone Acknowledgment* packet, the application instance transmits a *Collective Responder Count Response* packet to indicate it has written its datum.

5. The collective initiator writes its datum to the data set (if the data set is co-located with the collective initiator, then a component-local write is performed, else a Core 64 Write request packet is used to write the datum). The collective initiator may write its datum at any time once the collective is initiated.

5. a. If multiple initiators are semantically dependent upon the results of this collective, then they shall use an implementation-specific mechanism to ensure sequential consistency, e.g., collective 0 is successfully completed prior to initiating collective 1, and so forth.

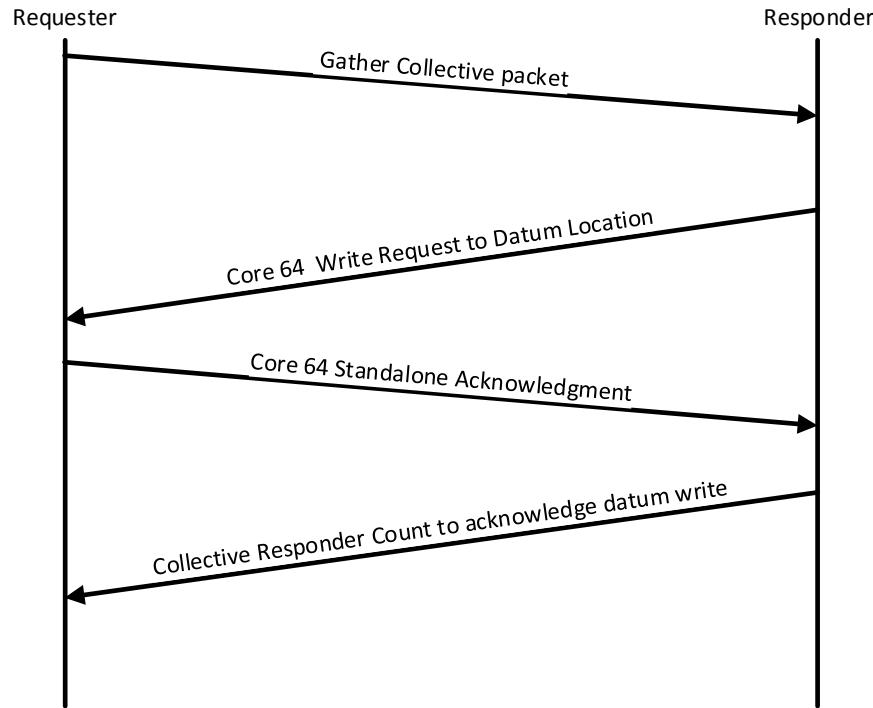


Figure 7-101: Gather Collective Ladder Diagram

**Developer Note:** If provisioned, collective accelerators can improve Gather Collective performance by:

- Eliminating serial Gather Collective request packet transmission, thus reducing the latency to disseminate the data set location.
- Aggregating multiple Responder writes, the collective accelerator can reduce the number of write requests required to update the data set.
- Reducing the number of packets transmitted between the Requester and all Responders. This can also reduce the probability of congestion and jitter.

The following are the features and requirements associated with Gather Collective request packets:

- Gather Collective request packets shall use the *Scatter and Gather Collective Request Packet Format*.
- A Gather Collective request packet shall be acknowledged by a *Collective Responder Count Response* packet (PR shall be set to 1b) or by a *Core 64 Standalone Acknowledgment* in case of error.

- If a non-default R-Key is associated with the page where the data set is located, then the Requester shall set RK = 1b and copy the RO R-Key or the RW R-Key associated with the page to the R-Key field.

### 7.15.5. All-Gather Collective

- 5 An All-Gather collective enables a group of processes, each of which owns a piece of information, to mutually join the information together into a single block that each receives.

The following steps are taken to perform an All-Gather collective:

1. A Requester (collective initiator) transmits an All-Gather Collective request packet to each participating Responder (application instance) identified by the Collective Group ID. The Requester copies the data set address to the packet's Address field, copies the data set's [CID, SID] to the respective Collective CID and Collective SID fields, and sets the corresponding Data Type.
2. Each application instance multiplies its rank (e.g., its MPI Rank) by the size indicated by the Data Type to calculate an offset from the data set address to locate where to write its datum.
3. Using the datum's location, each application instance transmits a Core 64 Write request packet.
4. Upon receipt of a Core 64 *Standalone Acknowledgment* packet, the application instance transmits a *Collective Responder Count Response* packet to indicate it has written its datum.
5. The collective initiator writes its datum to the data set (if the data set is co-located with the collective initiator, then a component-local write is performed, else a Core 64 Write request packet is used to write the datum). The collective initiator may write its datum at any time once the collective is initiated.
6. Upon writing its datum and receipt of *Collective Responder Count Response* packets that confirm all participating application instances have written their respective datums, the collective initiator transmits a *Collective Result* request packet to each participating Responder (application instance) identified by the Collective Group ID. The collective initiator sets the *Collective Result* request packet's ER bit, payload, Collective CID, and Collective SID fields to reflect the data's location.
7. Upon receipt of a *Collective Result* request packet, each application instance reads the gathered data set using a one or more Core 64 Read or LDM 1 Read request packets or a Buffer Get operation.
8. Upon receiving the data set, each application instance transmits a *Collective Responder Count Response* packet to indicate it has successfully read the data set.

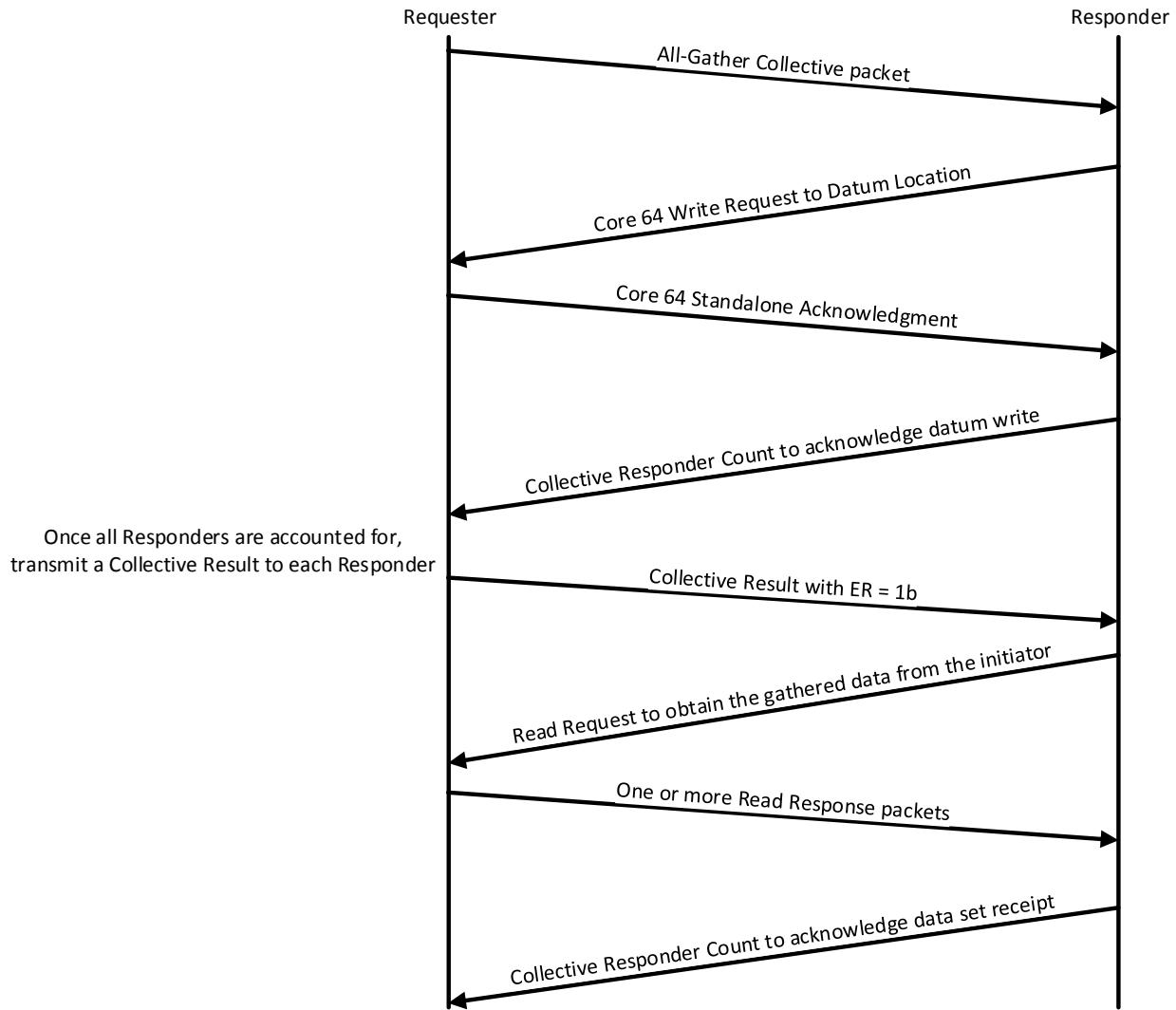


Figure 7-102: All-Gather Collective Ladder Diagram

**Developer Note:** If provisioned, collective accelerators can improve All-Gather Collective performance by:

- Eliminating serial All-Gather Collective request packet transmission, thus reducing the latency to disseminate the data set location.
- Aggregating multiple Responder writes, the collective accelerator can reduce the number of write requests required to update the data set.
- Caching the data set to reduce Requester read load and read latency.
- Reducing the number of packets transmitted between the Requester and all Responders. This can also reduce the probability of congestion and jitter.

The following are the features and requirements associated with All-Gather Collective request packets:

- All-Gather Collective request packets shall use the *Scatter and Gather Collective Request Packet Format*.

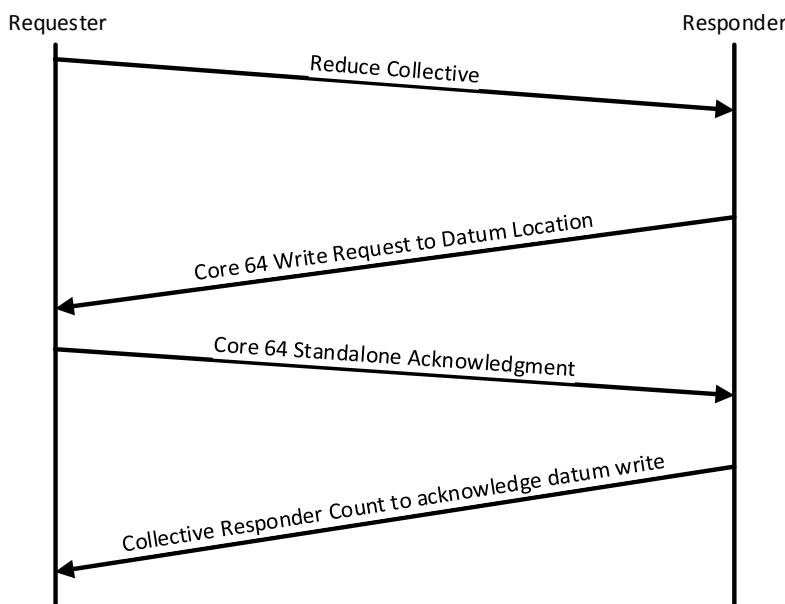
- An All-Gather Collective request packet shall be acknowledged by a *Collective Responder Count Response* packet (PR shall be set to 1b) or by a Core 64 *Standalone Acknowledgment* in case of error.

### 7.15.6. Reduce Collective

- 5 A Reduce collective enables an application instance to receive the result of a calculation performed across datum sets owned by a group of application instances.

The following steps are taken to perform a Reduce collective:

1. A Requester (collective initiator) transmits a Reduce Collective request packet to each participating Responder (application instance) identified by the Collective Group ID. The Requester copies the receive data set's address to the packet's Address field, copies the data set's [CID, SID] to the respective Collective CID and Collective SID fields, and sets the corresponding Data Type. For a simple Reduce collective, Filter = 0x0.
2. Each application instance multiplies its rank (e.g., its MPI Rank) by the size indicated by the Data Type to calculate an offset from the data set address to locate where to write its datum.
3. Using the datum's location, each application instance transmits a Core 64 Write request packet.
4. Upon receipt of a Core 64 *Standalone Acknowledgment* packet, the application instance transmits a *Collective Responder Count Response* packet to indicate it has written its datum.
5. The collective initiator writes its datum to the data set (if the data set is co-located with the collective initiator, then a component-local write is performed, else a Core 64 Write request packet is used to write the datum).
6. Upon writing its datum and receipt of *Collective Responder Count Response* packets that confirm all participating application instances have written their respective datums, the collective initiator performs the selected reduction operation across the data set.



25

Figure 7-103: Reduce Collective Ladder Diagram

**Developer Note:** If provisioned, collective accelerators can improve Reduce Collective performance by:

- Eliminating serial Reduce Collective request packet transmission, thus reducing the latency to disseminate the data set location.
- Aggregating multiple Responder writes, the collective accelerator can reduce the number of write requests required to update the data set.
- Reducing the number of packets transmitted between the Requester and all Responders. This can also reduce the probability of congestion and jitter.

The following are the features and requirements associated with Reduce Collective request packets:

- Reduce Collective request packets shall use the *Reduce Collective Request Packet Format*.
- A Reduce Collective request packet shall be acknowledged by a *Collective Responder Count Response* packet (PR shall be set to 1b) or by a *Core 64 Standalone Acknowledgment* in case of error.
- Pad CNT indicates the number of pad bytes included in the Filter Reference Data Value.
- If a non-default R-Key is associated with the page where the data set is located, then the Requester shall set RK = 1b and copy the RO R-Key or the RW R-Key associated with the page to the R-Key field.

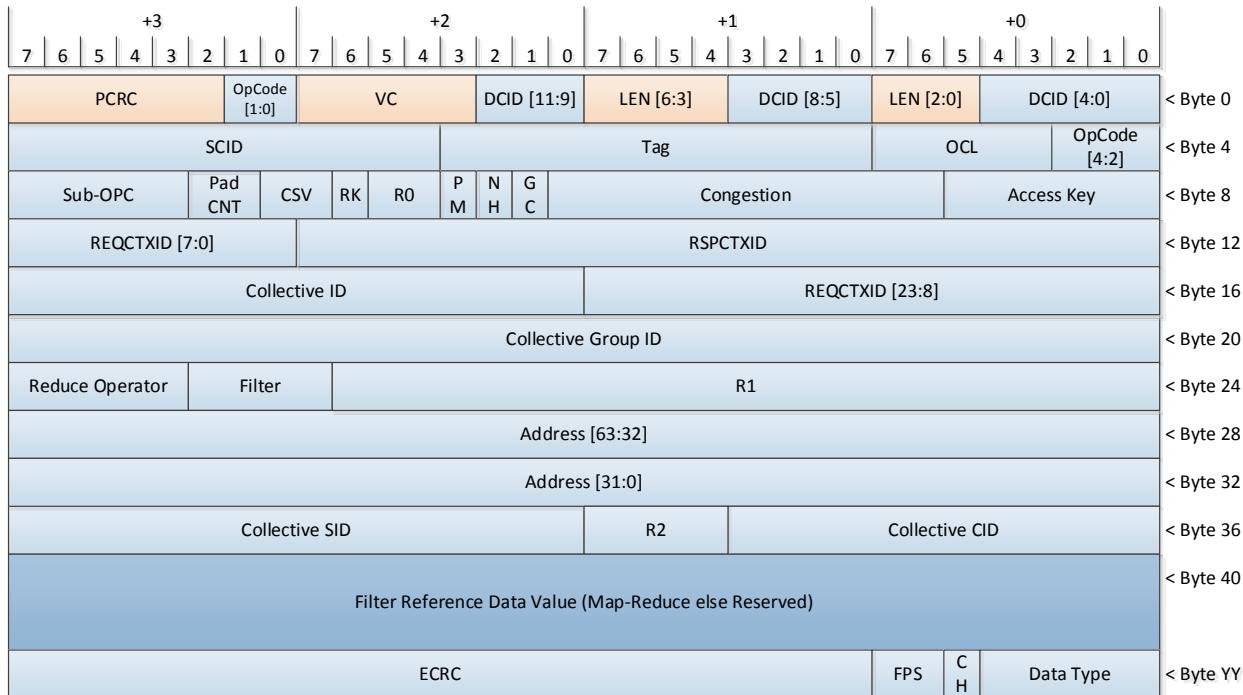


Figure 7-104: Reduce Collective Request Packet Format

Table 7-79: Reduce Collective Request Packet Fields

Field Name	Size (bits)	Description
R0	2	Reserved
R1	25	Reserved

Field Name	Size (bits)	Description
R2	4	Reserved

### 7.15.7. All-Reduce Collective

An All-Reduce operation enables a group of processes to perform a calculation across pieces of information owned by a group of processes and to each receive a copy of the result.

The following steps are taken to perform an All-Reduce collective:

5. A Requester (collective initiator) transmits a Reduce Collective request packet to each participating Responder (application instance) identified by the Collective Group ID. The Requester copies the receive data set's address to the packet's Address field, copies the data set's [CID, SID] to the respective Collective CID and Collective SID fields, and sets the corresponding Data Type. For a simple Reduce collective, Filter = 0x0.
10. Each application instance multiplies its rank (e.g., its MPI Rank) by the size indicated by the Data Type to calculate an offset from the data set address to locate where to write its datum.
15. Using the datum's location, each application instance transmits a Core 64 Write request packet.
20. Upon receipt of a Core 64 *Standalone Acknowledgment* packet, the application instance transmits a *Collective Responder Count Response* packet to indicate it has written its datum.
25. The collective initiator writes its datum to the data set (if the data set is co-located with the collective initiator, then a component-local write is performed, else a Core 64 Write request packet is used to write the datum).
30. Upon writing its datum and receipt of *Collective Responder Count Response* packets that confirm all participating application instances have written their respective datums, the collective initiator performs the selected reduction operation across the data set.
35. Upon completing the reduction, the collective initiator transmits a *Collective Result* request packet to each participating Responder (application instance) identified by the Collective Group ID. The reduced data is copied to the *Collective Result* request packet's payload field, or if it cannot fit, then the collective initiator writes the reduced data to a memory location, and sets the ER bit, payload, Collective CID, and Collective SID fields to reflect the reduced data's location.
40. Upon receiving the data set, each application instance transmits a *Collective Responder Count Response* packet to indicate it has successfully read the data set.

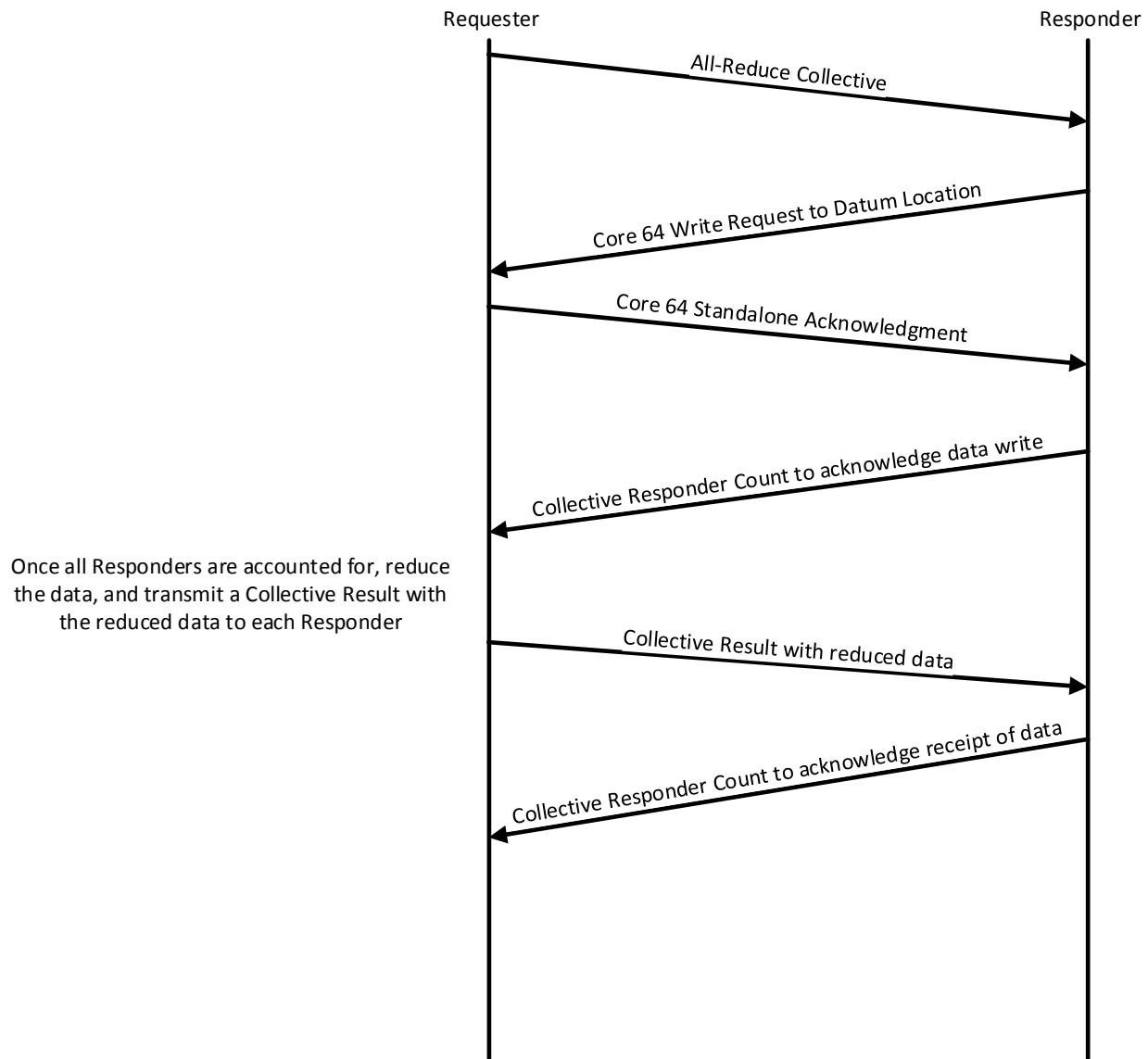


Figure 7-105: All-Reduce Collective Ladder Diagram

**Developer Note:** If provisioned, collective accelerators can improve Reduce Collective performance by:

- Eliminating serial Reduce Collective request packet transmission, thus reducing the latency to disseminate the data set location.
- Aggregating multiple Responder writes, the collective accelerator can reduce the number of write requests required to update the data set.
- Caching the data set to reduce Requester read load and read latency.
- Reducing the number of packets transmitted between the Requester and all Responders. This can also reduce the probability of congestion and jitter.

The following are the features and requirements associated with All-Reduce Collective request packets:

- All-Reduce Collective request packets shall use the *Reduce Collective Request Packet Format*.

- An All-Reduce Collective request packet shall be acknowledged by a *Collective Responder Count Response* packet (PR shall be set to 1b) or by a Core 64 *Standalone Acknowledgment* in case of error.

### 7.15.8. Map-Reduce Collective

- 5 A Map-Reduce operation enables a process to receive the result of a calculation performed across pieces of information owned by a group of processes, where a filter restricts the portions of data that participate in the calculation.

The following steps are taken to perform a Map-Reduce collective:

1. A Requester (collective initiator) transmits a Map-Reduce Collective request packet to each participating Responder (application instance) identified by the Collective Group ID. The Requester copies the receive data set's address to the packet's Address field, copies the data set's [CID, SID] to the respective Collective CID and Collective SID fields, and sets the corresponding Data Type. For a simple Reduce collective, Filter = 0x0.
- 10 2. Each application instance multiplies its rank (e.g., its MPI Rank) by the size indicated by the Data Type to calculate an offset from the data set address to locate where to write its datum.
- 15 3. Using the datum's location, each application instance transmits a Core 64 Write request packet.
4. Upon receipt of a Core 64 *Standalone Acknowledgment* packet, the application instance transmits a *Collective Responder Count Response* packet to indicate it has written its datum.
- 20 5. The collective initiator writes its datum to the data set (if the data set is co-located with the collective initiator, then a component-local write is performed, else a Core 64 Write request packet is used to write the datum).
6. Upon writing its datum and receipt of *Collective Responder Count Response* packets that confirm all participating application instances have written their respective datums, the collective initiator performs the selected reduction operation across the data set.

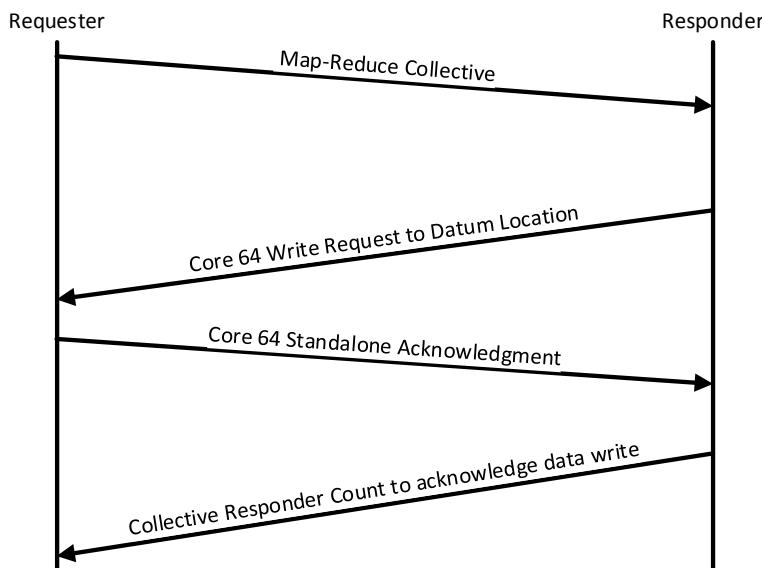


Figure 7-106: Map-Reduce Collective Ladder Diagram

25

**Developer Note:** If provisioned, collective accelerators can improve Reduce Collective performance by:

- *Eliminating serial Reduce Collective request packet transmission, thus reducing the latency to disseminate the data set location.*
- *Aggregating multiple Responder writes, the collective accelerator can reduce the number of write requests required to update the data set.*
- *Reducing the number of packets transmitted between the Requester and all Responders. This can also reduce the probability of congestion and jitter.*

The following are the features and requirements associated with Map-Reduce Collective request packets:

- Map-Reduce Collective request packets shall use the *Reduce Collective Request Packet Format*.
- A Map-Reduce Collective request packet shall be acknowledged by a *Collective Responder Count Response* packet or a *Core 64 Standalone Acknowledgment* in case of error.
  - If a *Collective Responder Count Response* packet is transmitted, then shall set PR = 1b if the Filter is true when applied to the address data and the Filter Reference Data Value, and shall set PR = 0b if not Filter is not true.

### 7.15.9. All-Map-Reduce Collective

An All-Reduce operation enables a group of processes to perform a calculation across pieces of information owned by a group of processes and to each receive a copy of the result, where a filter restricts the portions of data that participate in the calculation.

The following steps are taken to perform an All-Map-Reduce collective:

1. A Requester (collective initiator) transmits an All-Map-Reduce Collective request packet to each participating Responder (application instance) identified by the Collective Group ID. The Requester copies the receive data set's address to the packet's Address field, copies the data set's [CID, SID] to the respective Collective CID and Collective SID fields, and sets the corresponding Data Type. For a simple Reduce collective, Filter = 0x0.
2. Each application instance multiplies its rank (e.g., its MPI Rank) by the size indicated by the Data Type to calculate an offset from the data set address to locate where to write its datum.
3. Using the datum's location, each application instance transmits a Core 64 Write request packet.
4. Upon receipt of a *Core 64 Standalone Acknowledgment* packet, the application instance transmits a *Collective Responder Count Response* packet to indicate it has written its datum.
5. The collective initiator writes its datum to the data set (if the data set is co-located with the collective initiator, then a component-local write is performed, else a Core 64 Write request packet is used to write the datum).
6. Upon writing its datum and receipt of *Collective Responder Count Response* packets that confirm all participating application instances have written their respective datums, the collective initiator performs the selected reduction operation across the data set.
7. Upon writing its datum and receipt of *Collective Responder Count Response* packets that confirm all participating application instances have written their respective datums, the collective initiator performs the selected reduction operation across the data set.

- 5
8. Upon completing the reduction, the collective initiator transmits a *Collective Result* request packet to each participating Responder (application instance) identified by the Collective Group ID. The reduced data is copied to the *Collective Result* request packet's payload field, or if it cannot fit, then the collective initiator writes the reduced data to a memory location, and sets the ER bit, payload, Collective CID, and Collective SID fields to reflect the reduced data's location.
  9. Upon receiving the data set, each application instance transmits a *Collective Responder Count Response* packet to indicate it has successfully read the data set.

10  
Once all Responders are accounted for, reduce the data, and transmit a Collective Result with the reduced data to each Responder

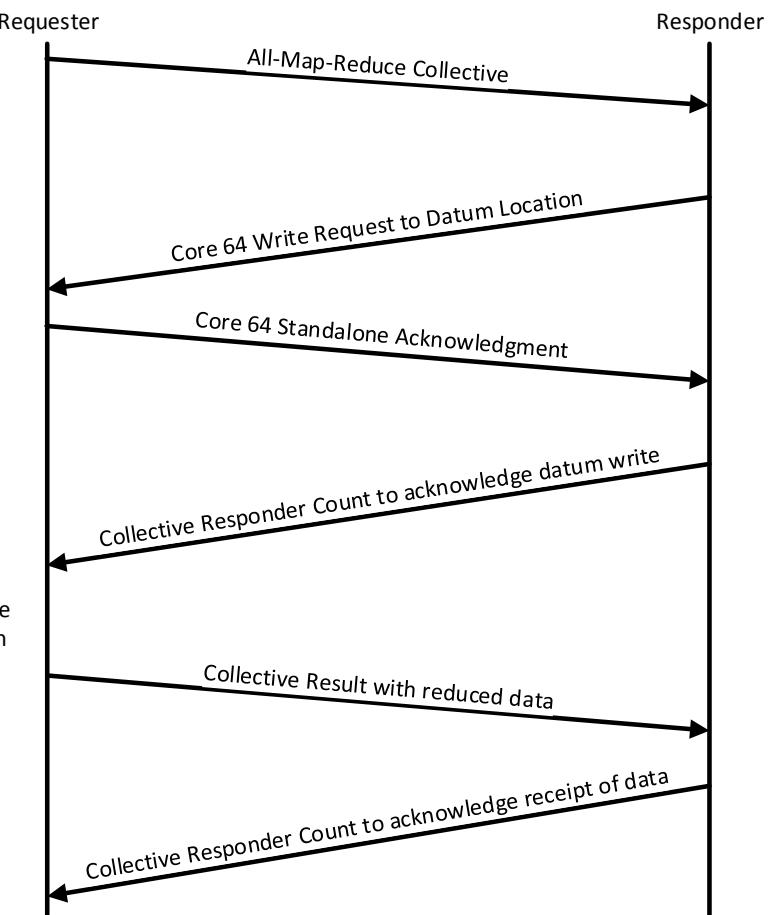


Figure 7-107: All-Map-Reduce Collective Ladder Diagram

**Developer Note:** If provisioned, collective accelerators can improve Reduce Collective performance by:

- 15
- Eliminating serial Reduce Collective request packet transmission, thus reducing the latency to disseminate the data set location.
  - Aggregating multiple Responder writes, the collective accelerator can reduce the number of write requests required to update the data set.
  - Caching the data set to reduce Requester read load and read latency.
  - Reducing the number of packets transmitted between the Requester and all Responders. This can also reduce the probability of congestion and jitter.

The following are the features and requirements associated with All-Map-Reduce Collective request packets:

- All-Map-Reduce Collective request packets shall use the *Reduce Collective Request Packet Format*.
- An All-Map-Reduce Collective request packet shall be acknowledged by a *Collective Responder Count Response* packet or a *Standalone Acknowledgment* in case of error.
  - If a *Collective Responder Count Response* packet is transmitted, then shall set PR = 1b if the Filter is true when applied to the address data and the Filter Reference Data Value, and shall set PR = 0b if not Filter is not true.

### 7.15.10. Collective Result

Multiple collectives use the Collective Result request packet to return the collective request execution results within the payload field. A subset of collectives use the Collective Result request packet to return the address of the location where the results are to be read.

The following are the features and requirements associated with Collective Result request packets:

- Collective Result request packets shall use the *Collective Result Request Packet Format*.
- A Collective Result request packet shall be acknowledged by a Core 64 *Standalone Acknowledgment*.
- If ER == 0b, then the payload shall contain the result data, and Data Type shall be set to the datum type. If there is no result data, then the payload in the *Collective Result* request packet shall not be present, and the Data Type field shall be Reserved.
- If ER == 1b, then:
  - The component where the data to be read is located shall be identified by the Collective CID and Collective SID (if applicable) fields.
  - The Payload field shall contain the length, address, and R-Key associated with the data to be read.
  - Data Type shall be set to the datum type irrespective of the number of datum values to be read.

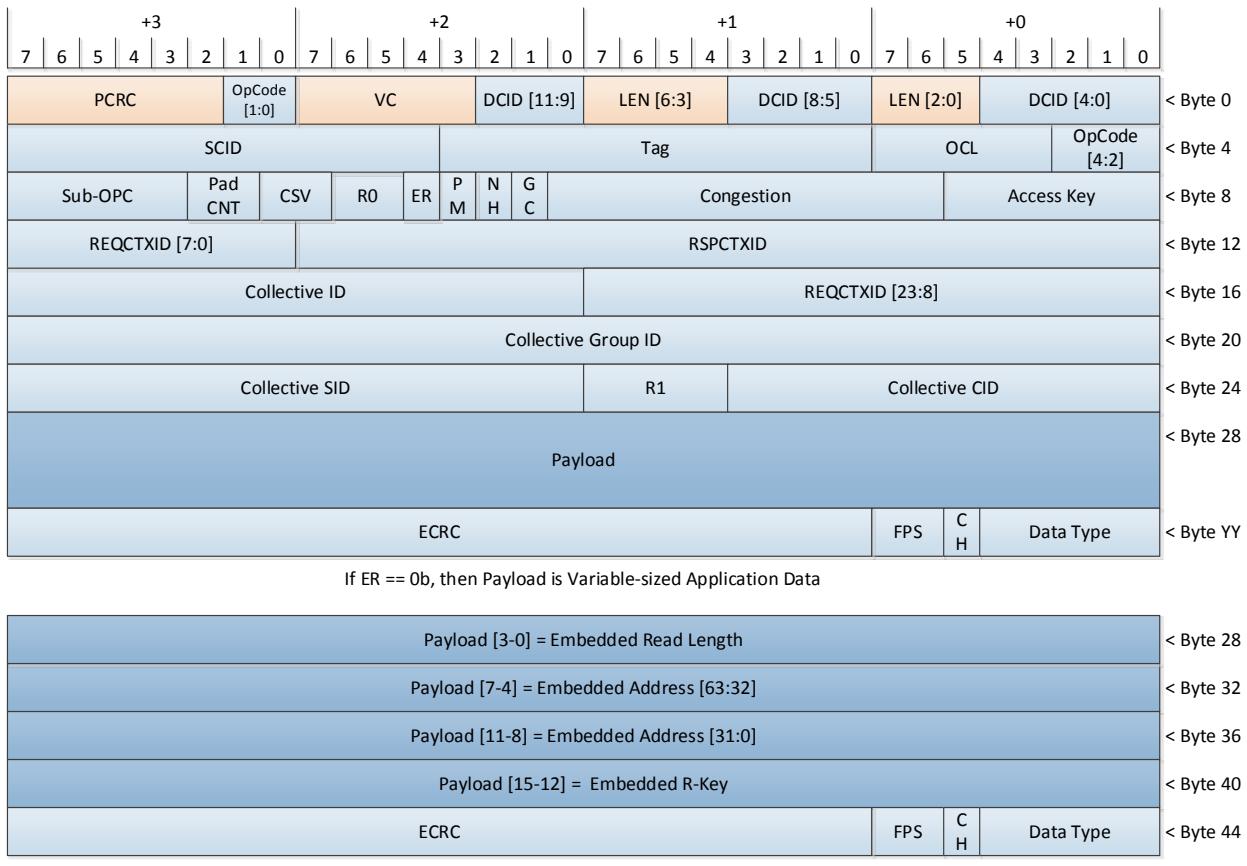


Figure 7-108: Collective Result Request Packet Format

Table 7-80: Collective Result Request Packet Fields

Field Name	Size (bits)	Description
ER	1	<p>Indicates how to interpret the payload field as application data or as an embedded read request.</p> <p>If ER == 0b, then the payload field shall be treated as application data.</p> <p>If ER == 1b, then the payload field shall be 128 bits in length, and shall be interpreted as follows:</p> <ul style="list-style-type: none"> <li>• Payload[3-0] shall be the size of the message to be read by the Responder</li> <li>• Payload[11-4] shall be the address of byte 0 of the data to be read. The address shall be bit-wise placed as follows: <ul style="list-style-type: none"> <li>◦ Address[63:32] in Payload[11-8]</li> <li>◦ Address[31:4] in Payload[15-12]</li> </ul> </li> <li>• Payload[15-12] shall be the R-Key to use in the subsequent Core 64 Read or LDM 1 Read request packets. If the R-Key == Default R-Key, then the subsequent Read or LDM 1 Read request packet may set RK = 0b (if RK is present).</li> </ul>

Field Name	Size (bits)	Description
R0		0b—Application Data 1b—Embedded Read
	2	Reserved
	4	Reserved
	5	Reserved

### 7.15.11. Collective Abort

A Collective Abort is used to abort the indicated collective operation.

The following are the features and requirements associated with Collective Abort request packets:

- Collective Abort request packets shall use the *Barrier and Abort Collective Request Packet Format*.
- A Collective Abort request packet shall be acknowledged by a Core 64 *Standalone Acknowledgment* upon the collective being successfully aborted. If the collective does not exist or is no longer running, then this shall be handled as a successful abort.
- Any component participating in a collective may initiate a Collective Abort.
- A Collective Abort may be transmitted by any application instance participating in the collective.

### 7.15.12. Collective Abort All

A Collective Abort All is used to abort all outstanding collectives associated with indicated Collective Group. A Collective Abort All is typically initiated as part of application shutdown.

The following are the features and requirements associated with Collective Abort All request packets:

- Collective Abort All request packets shall use the *Barrier and Abort Collective Request Packet Format*.
- A Collective Abort request packet shall be immediately acknowledged by a Core 64 *Standalone Acknowledgment* without waiting for the Responder to abort all outstanding collectives.
- Any component participating in a collective may initiate a Collective Abort All.
- A Collective Abort All may be transmitted by any application instance participating in the collective.

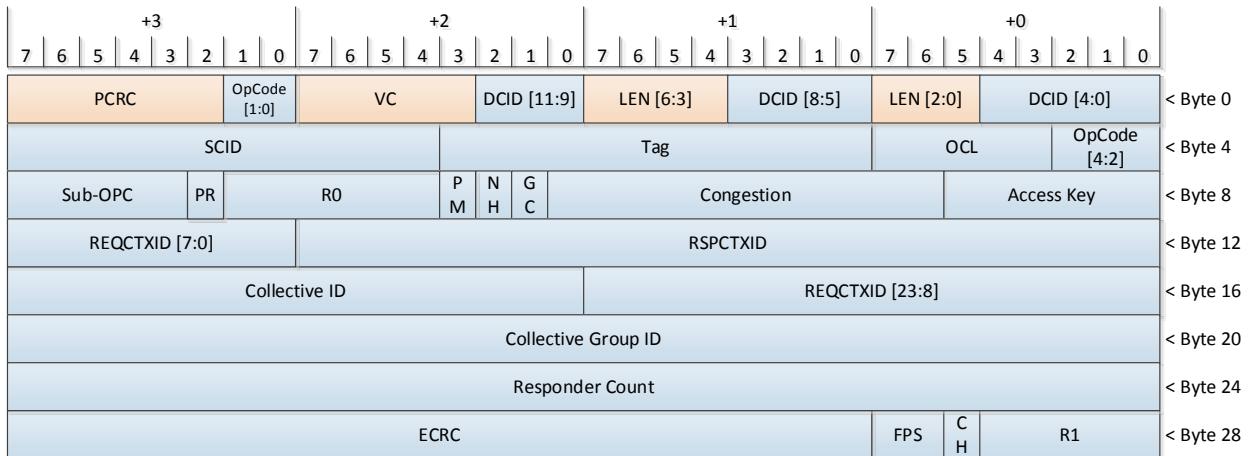
### 7.15.13. Collective Responder Count Response

The Collective Responder Count Response packet is used to inform a Requester that the corresponding collective request packet was received and successfully executed.

The following are the features and requirements associated with Collective Responder Count Response packets:

- Collective Responder Count Response packets shall use the *Collective Responder Count Response Packet Format*.
- If the Collective Responder Count Response packet is from a single leaf Responder, then the Responder Count field shall be set to 0x1.

- If the Collective Responder Count Response packet is from a component such as a collective accelerator that acted as a proxy for multiple leaf Responders or collective accelerators, then the Responder Count field shall be set to indicate the total number of leaf Responders.



5

Figure 7-109: Collective Responder Count Response Packet Format

Table 7-81: Collective Responder Count Response Packet Fields

Field Name	Size (bits)	Description
Responder Count	1	Participating Responder—Indicates if the Responder participates in the specific collective operation (e.g., Map-Reduce Collectives). 0b—Non-participant 1b—Participant
	32	Number of Responders represented by this response packet
	7	Reserved
	5	Reserved

# 8. Configuration and Management

## 8.1. Configuration and Management Responsibilities

Configuration and management is responsible for:

- Component and topology discovery and management
- Component configuration
- Switch configuration
- Dynamic event management
- Power management
- Access control management
- Security management
- Performance monitoring and management

## 8.2. Component Management

Component management may be performed through in-band or out-of-band mechanisms. *In-band Management* uses *In-band Management* read and write request packets to configure Control Space structures. *Out-of-band Management* refers to using an out-of-band interface to perform read and write operations to configure Control Space structures. A component may support only *In-band Management*, *Out-of-band Management*, or a combination thereof.

Independent of the mechanism, configuration starts with the *Core Structure*, which is located at a universally-defined Control Space address. The *Core Structure* contains pointers to other control structures.

## 8.3. Interface States and Descriptions

Table 8-1: Interface States and Descriptions

I-States	Description
I-Down	This state is used to indicate an interface is non-operational. If the transition was caused by an interface, link, or physical layer failure, then the steps specified for a <i>Warm Interface Reset</i> shall be taken. If the transition was not caused by an interface, link, or physical layer failure, then the steps specified for a <i>Full Interface Reset</i> shall be taken.
I-CFG	This state is applicable only if the component supports <i>In-band Management</i> . While in this state: <ul style="list-style-type: none"><li>• All <i>In-band Management</i> read and write request packets shall be allowed.</li><li>• All link-local packets may be exchanged.</li></ul>
I-Up	This state is the normal operation state. While in this state:

I-States	Description
I-LP	<ul style="list-style-type: none"> <li>• All mandatory and enabled packets shall be allowed.</li> <li>• The link may dynamically enter and exit supported low-power operational states. Entry and exit does not impact the interface's state.</li> </ul> <p>This state is used to indicate an interface has entered a low-power state.</p> <p>While in this state:</p> <ul style="list-style-type: none"> <li>• End-to-end and link-local packet generation and execution shall be disabled.</li> <li>• All interface resources and Control Space structure state and contents shall be preserved.</li> </ul>

All component interfaces shall support the *Interface State Machine*.

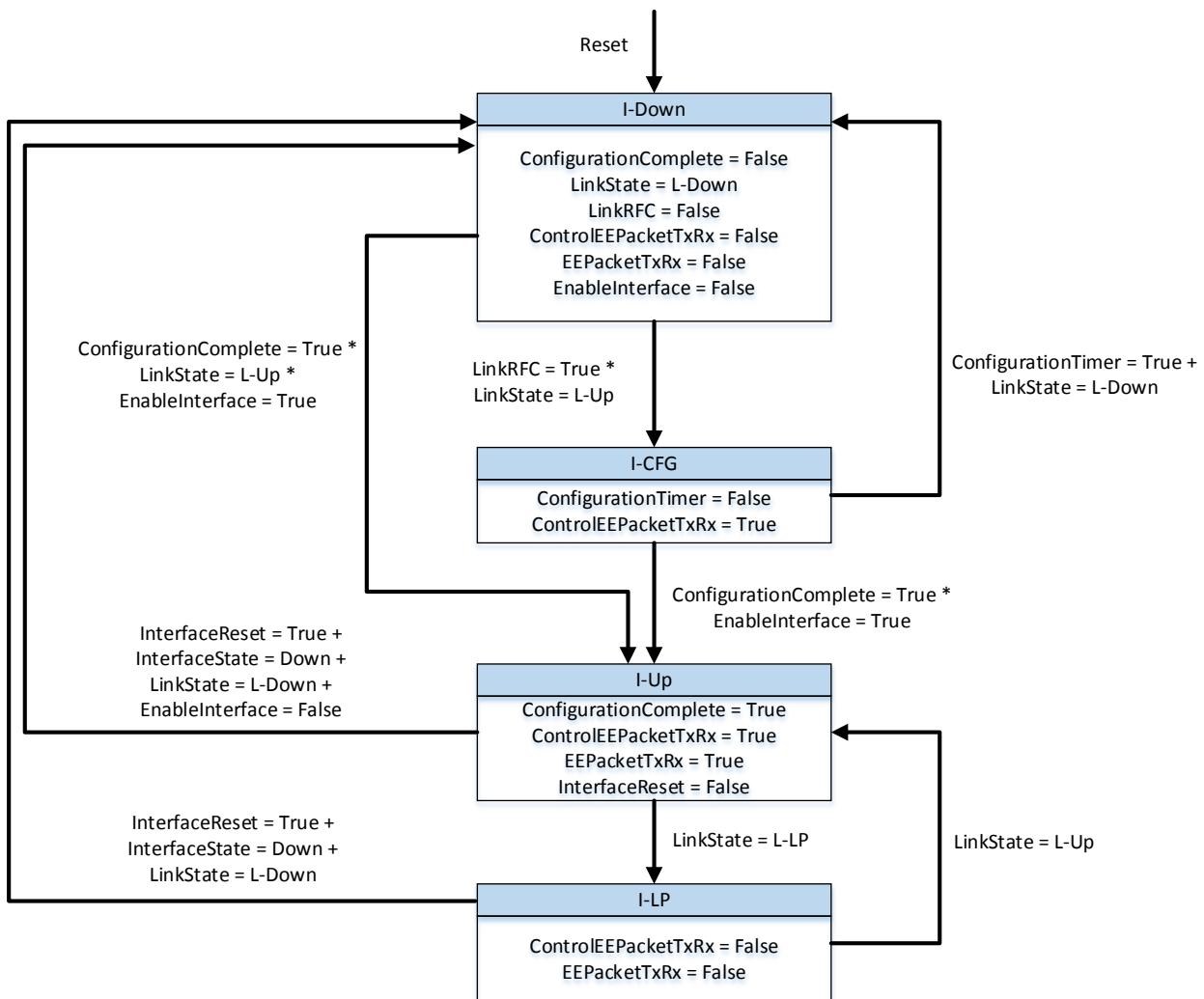


Figure 8-1: Interface State Machine

### 8.3.1. Interface State Machine Terms

Reset—an implementation-specific internal signal to reset the interface. This signal is asserted subsequent to component power-up.

5 ConfigurationComplete—a flag that indicates if the component interface configuration has been completed. Solutions do not require *In-band Management* to configure the interface (e.g., embedded, co-packaged memory, etc.), therefore, may transition directly from I-Down to I-Up.

LinkState—the present link state

LinkRFC—a flag that indicates if the interface is transmitting *Link RFC* packets with  $TTC == 0x0$ .

10 ControlEEPacketTxRx—a flag that determines if *In-band Management* packets may be transmitted or received on this interface.

EEPacketTxRx—a flag that determines if end-to-end packets may be transmitted or received on this interface.

15 EnableInterface—a flag that indicates if management has set *Interface I-Control* Interface Enable == 1b. If the interface does not require management configuration, then the interface shall set *Interface I-Control* Interface Enable = 1b upon completing self-initialization. ConfigurationTimer—a flag that indicates if the Configuration Completion Timer has expired (see Configuration Completion Timer and Release)

InterfaceReset—a flag that indicates if an internal signal or management has initiated an *Interface Reset*.

## 8.4. Component States and Descriptions

Table 8-2: Component States and Descriptions

C-States	Description
C-Down	<p>This state is used to indicate a component is non-operational.</p> <p>Upon transitioning to this state:</p> <ul style="list-style-type: none"><li>Unless explicitly stated otherwise by this specification, then the steps specified in <i>Component Reset</i> for a <i>Full Component Reset</i> shall be taken.</li><li>All interfaces shall transition to the I-Down state.</li></ul> <p>While in this state:</p> <ul style="list-style-type: none"><li>Management may configure the component through <i>In-band Management</i> or <i>Out-of-band Management</i>.</li></ul>
C-CFG	<p>This state is used to configure the component when using in-band configuration. If a component supports only <i>Out-of-band Management</i>, then it shall not enter this state.</p> <p>A component enters this state when at least one component interface is able to transmit and receive <i>Link RFC</i> packets and the component has completed self-initialization such that it can receive and execute <i>In-band Management</i> request packets, and the component has successfully received and executed a Control Write request packet.</p> <p>While in this state:</p>

C-States	Description
C-Up	<ul style="list-style-type: none"> <li>Component power shall be limited to the default maximum power associated with the mechanical package or module.</li> <li>All <i>In-band Management</i> and link-local packets shall be permitted.</li> <li>The configuration completion timer shall be started and operate as described in <i>Configuration Completion Timer and Release</i>.</li> </ul> <p>This state is the normal operation state. A component enters this state once configuration is successfully completed.</p> <p>While in this state:</p> <ul style="list-style-type: none"> <li>Maximum component power shall be limited to the configured maximum.</li> <li>All mandatory and enabled packets shall be allowed.</li> </ul>
C-LP	<p>This state is used to indicate a component has entered a low-power state. The component may automatically enter this state if <i>Core Structure Component CAP 1 Control</i> Automatic C-State Enable == 1b and the component is idle for at least the time as indicated in the <i>Core Structure</i> C-LP Time field while in the C-Up state.</p> <p>While in this state:</p> <ul style="list-style-type: none"> <li>Component power shall be reduced to what is required to maintain configured resources, Data Space media, and Control Space structures and associated media, and to receive link-local and end-to-end packets. <ul style="list-style-type: none"> <li>Management may configure the component to transition to C-LP.</li> <li>A component may manage its state and power to lower internal power consumption based on solution-specific needs.</li> <li>The component should maintain all supported statistics. However, if the power required to maintain these statistics exceeds the maximum power allowed in C-LP, then these statistics may be discarded. If this occurs and the component transitions to C-Up, then the component shall reset these statistics to zero.</li> </ul> </li> <li>If <i>In-band Management</i> is supported: <ul style="list-style-type: none"> <li>Packets that target Control Space shall be received and executed.</li> <li>If so configured and if the component automatically entered this state, then upon receipt of any packet that targets Data Space, the component shall transition to C-Up. If not so configured, then the component shall silently discard any packet that targets Data Space.</li> <li>If the component did not automatically enter this state, then the component shall remain in this state until management directs the component to another state, or a component reset is initiated.</li> </ul> </li> <li>The component shall maintain at least one interface capable in the I-Up state.</li> <li><i>Out-of-band Management</i> may be used to transition a component to the C-Up state or the C-Down state (see <i>Component Reset</i>).</li> </ul>
C-DLP	<p>This state is used to indicate the component has entered a deep low-power state. The component may automatically enter this state if <i>Core Structure Component CAP 1 Control</i> Automatic C-State Enable == 1b and the component is idle for at least the time indicated in the <i>Core Structure</i> C-DLP Time field while in the C-LP state.</p> <p>Prior to entering this state, for each component interface in I-Up and whose status does not indicate the peer component is in C-DLP, the component shall exchange link-</p>

C-States	Description
	<p>local <i>Link CTL</i> packets with each peer component interface to indicate that this component is transitioning to C-DLP.</p> <p>While in this state:</p> <ul style="list-style-type: none"> <li>Component power shall be reduced to what is required to maintain configured resources, Data Space media, and Control Space structures and associated media, and to receive link-local packets. <ul style="list-style-type: none"> <li>Management may configure the component to transition to C-DLP.</li> <li>A component may manage its state and power to lower internal power consumption based on solution-specific needs.</li> <li>The component should maintain all supported statistics. However, if the power required to maintain these statistics exceeds the maximum power allowed in C-DLP, then these statistics may be discarded. If this occurs and the component transitions to C-LP, then the component shall reset these statistics to zero.</li> </ul> </li> <li>The component shall maintain at least one interface capable of receiving link-local packets (I-Up state) given link-local packets (See <i>Link CTL</i>) may be used to transition the component to a new C-State. All other interfaces may be transitioned to I-LP.</li> <li><i>Out-of-band Management</i> may be used to transition a component to the C-Up state (set <i>Core C-Control</i> Transition C-Up = 1b or, if supported, assert the mechanical form factor or component package Power C-Up signal) or the C-Down state (see <i>Component Reset</i>).</li> <li>Any newly received end-to-end OpClass packets shall be silently discarded.</li> </ul> <p>Upon exiting this state to C-Up, the component shall transition all interfaces in the I-LP state to I-Up. For each component interface in I-Up and whose status does not indicate the peer component is in C-DLP, the component shall exchange link-local Link CTL packets to inform the peer interface that this component has transitioned to C-Up.</p>

All components shall support the *Component State Machine*.

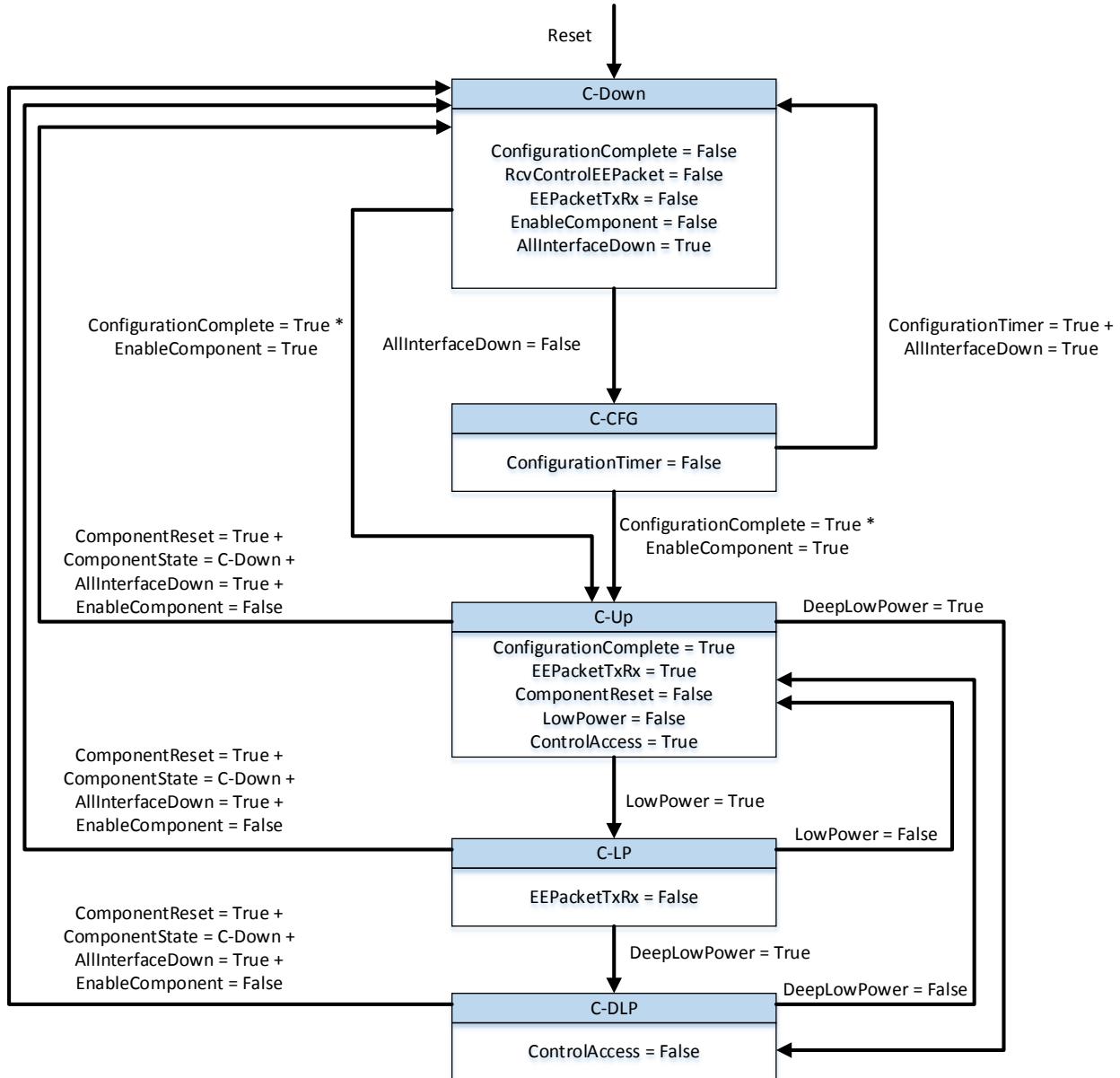


Figure 8-2: Component State Machine

#### 8.4.1. Component State Machine Terms

Reset—an implementation-specific internal signal to reset the component. This signal is asserted subsequent to component power-up.

5

SelfInitialized—component is autonomously initialized to the default state

ConfigurationComplete—a flag that indicates if the component configuration has been completed. Solutions do not require *In-band Management* to configure the component (e.g., embedded, co-packaged memory, etc.), therefore, may transition directly from C-Down to C-Up.

10 EEPacketTxRx—a flag that determines if end-to-end packets may be transmitted or received by this component.

EnableComponent— flag that indicates if management has set *Core C-Control* Component Enable == 1b. If the component does not require management configuration, then the component shall set *Core C-Control* Component Enable = 1b upon completing self-initialization.

AllInterfaceDown—a flag that indicates if all component interfaces have transitioned to I-Down.

5 ConfigurationTimer—a flag that indicates if the Configuration Completion Timer has expired (see Configuration Completion Timer and Release)

ComponentReset—a flag that indicates if an internal signal or management has set *Core C-Control* Component Reset to a non-zero value to initiate a *Component Reset*.

LowPower—a flag that indicates if the component has been configured to enter or exit the C-LP state.

10 ControlAccess—a flag that indicates if *In-band Management* packets may be transmitted and received.

DeepLowPower—a flag that indicates if the component has been configured to enter or exit the C-DLP state.

## 8.5. Out-of-band Management

15 A component maybe managed using an out-of-band management interconnect that is outside the scope of this specification. Upon discovery, management transmits out-of-band management interconnect-specific read requests to the Control Space address of the component's to enumerate the component's type, capabilities, and attributes. The manager proceeds to configure the component. Upon completion, the manager enables the component via the *Core C-Control* Component Enable field, which transitions the component to the C-Up state. Management also enables each operation interface connected to a peer interface by setting *Interface I-Control* Enable Interface = 1b, which transitions the interface to the I-Up state.

20 Out-of-band management shall be performed through an out-of-band management interconnect. The out-of-band management interconnect type is identify in the mechanical form factor specification used to implement the component or, if not embodied in a mechanical form factor, then is identified by the component supplier.

25 The following are the features and requirements associated with out-of-band management:

- All out-of-band management read and write requests shall access only the component's Control Space, i.e., Data Space shall not be accessible.
- An out-of-band management read request to an address that exceeds the maximum Control Space address shall return all 1s.
- An out-of-band management write request to an address that exceeds the maximum Control Space address shall have no impact to the underlying media or component.
- Upon discovery, management transmits out-of-band-specific read requests to the Control Space address of the component's *Core Structure* to enumerate the component's type, capabilities, and attributes. Management configures the component. Upon completion, management enables the component via the *Core C-Control* Component Enable field which transitions the component to the C-Up state. In addition, management enables each operational interface connected to a peer interface setting *Interface I-Control* Enable Interface = 1b, which transitions the interface to the I-Up state.

30 40 *Core Structure Component CAP 1 Out-of-Band Management Support* indicates if the component support in-band management. *Core Structure Component CAP 1 Control Out-of-Band Management Enable*

determines if out-of-band management is enabled. A component may support and have enabled *In-band Management* and out-of-band management.

## 8.6. In-band Management

In-band management use special read and write request packets to access a component's Control Space. Component interfaces that support only the P2P-Core or P2P-Coherency OpClasses use *CTL-Read and CTL-Write Packets*, and components that support the Control OpClass use *Control Read and Control Write Packets*. Further, in switch-based topologies, *Directed Control Space Packet Relay* is used to exchange *Control Read and Control Write Packets* between the management component and the uninitialized component. *Directed Control Space Packet Relay* enables a switch directly attached to an uninitialized component to relay packets between management and the component.

In addition to read and write request packets, in-band management supports notification packets to inform management of component events. Component interfaces that support only the P2P-Core and P2P-Coherency OpClasses use a *CTL-UE Packet*, and components that support the Control OpClass use *Unsolicited Event (UE) Packet*. In addition, components that support the Control OpClass may support packets manage component power, perform high-speed R-Key Domain (RKD) updates, or to enable multiple managers to communicate.

*Core Structure Component CAP 1 In-Band Management Support* indicates if the component support in-band management. *Core Structure Component CAP 1 Control In-Band Management Enable* determines if in-band management is enabled. A component may support and have enabled in-band management and *Out-of-band Management*.

### 8.6.1. In-band Discovery and Initialization

In-band component discovery and initialization is performed using the following steps:

1. The physical layer parameters are either configured through an out-of-band interface, set by default, or the physical layer supports physical-layer-specific in-band negotiation. This is required to enable at least one component link to transition to the L-Up state.
2. For each link that transitions to L-Up, the corresponding interface shall periodically transmit a Link RFC packet.
3. Upon receiving a *Link RFC* packet, if the receiving component is in C-Up and *Component Error and Signal Event Structure* I-Event New Peer Component Detected Event == 1b, then the component informs management as configured (component-local interrupt and / or *Unsolicited Event (UE) Packet*). If management is co-located with the receiving component, then the component may use the *Component Event Structure* or a component-specific method to inform management.
  - a. If the receiving component is not in C-Up or I-Event New Peer Component Detected Event == 0b, then the component shall silently discard the *Link RFC* packet.
4. Once informed, management initiates component configuration through *CTL-Read and CTL-Write Packets* or *Control Read and Control Write Packets*.
  - a. If a switch detected the new component, then management should use *Directed Control Space Packet Relay* to configure the new component.
  - b. If multiple switches detect the new component, then management shall use only one switch to configure the new component.
5. Upon the successful receipt and execution *CTL-Read and CTL-Write Packets* or *Control Read and Control Write Packets*, the component shall transition to C-CFG.

- 5 a. While in C-CFG, a component:
- Shall exchange link-local packets as specified in this document.
  - Shall transmit response packets to *CTL-Read and CTL-Write Packets or Control Read and Control Write Packets*.
  - Shall not transmit any request packets, or any response packets to non-*CTL-Read and CTL-Write Packets or non-Control Read and Control Write Packets*.
  - All *non-CTL-Read and CTL-Write Packets or non-Control Read and Control Write Packets* shall be silently discarded.
- 10 b. *Control Read and Control Write Packets* requirements:
- If GC == 0b in Control Read or Control Write request packets and *Core Structure* PMCID field is not configured and the component is in the C-CFG state, then:
    - The Responder shall not validate the SCID or the DCID packet fields.
    - Upon successful execution of the first Control Write request packet, the component shall copy the packet's SCID to *Core Structure* PMCID.
    - If the Primary Manager is capable of fabric management, then it may configure *Core Structure* PFMCID = PMCID and set *Core Structure Component CAP 1 Control Manager Type* = 1b.
  - If GC == 1b in Control Read or Control Write request packets and *Core Structure* PMCID and PFMCID fields are not configured and the component is in the C-CFG state, then:
    - The Responder shall not validate the SCID, SSID, DCID, or DSID packet fields.
    - Upon successful execution of the first Control Write request packet, the component shall copy the packet's SCID to *Core Structure* PFMCID and the packet's SSID to *Core Structure* PFMSID, and sets *Core Structure Component CAP 1 Control Manager Type* = 1b.

25 **Developer Note:** If a component is not configured and is in the C-CFG state, then the first Primary Manager or Primary Fabric Manager to transmit a Control Write request packet takes management control of the component; the component blindly accepts the first manager. Further, if GC == 0b, then the component defaults to being managed by a Primary Manager, else it defaults to being managed by a Primary Fabric Manager.

- 30 35 iii. If management configured a non-zero *Core Structure* MGR-UUID, then the Responder shall validate the packet's MGR-UUID field.
- The *Core Structure* contains a single MGR-UUID field. If a manager configures the MGR-UUID field, then it is responsible for distributing this value to any other manager authorized to access the component using Control OpClass packets that contain a MGR-UUID field.
- 40 45 6. Once the component is configured, management may enable the component through the *Core C-Control* field.

*Example Discovery Topologies* illustrates how the above steps may be applied to different solution topologies. The figure legend is as follows:

- P0 represents a Requester, e.g., a processor component.
- P0-I0 represents Interface 0 on component P0.
- M0-M3 represent Responders, e.g., memory components.
- M0-I0, M1-I0, M2-I0, and M3-I0 represent Interface 0 on components M0-M3

- M0-I1 represents Interface 1 on component M0
- S0 represents a discrete switch with 5 interfaces—I0-I4.

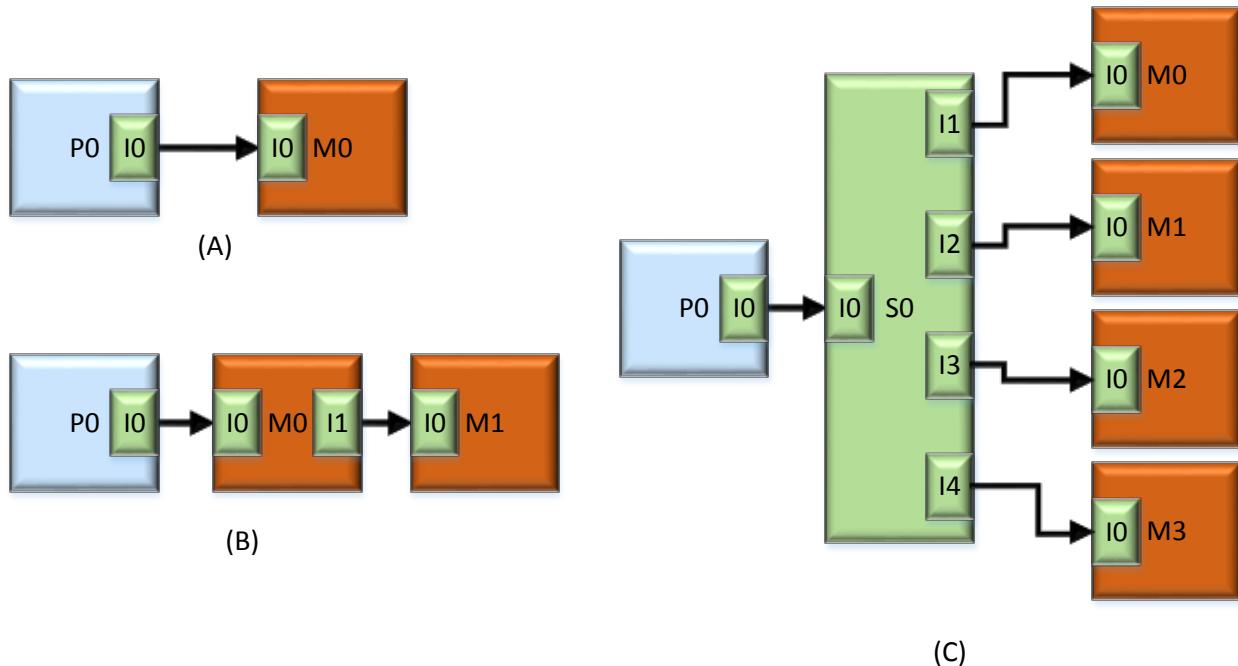


Figure 8-3: Example Discovery Topologies

5 Using *Example Discovery Topologies* (A) topology as the basis, the following steps are taken:

1. At time zero, the following statements are true:
  - a. Component P0 has been configured and is in C-Up. If the component supports *In-band Management*, then management configures the Requester ZMMU to access Control Space, configures *Core Structure* CID 0 to identify P0, and configures the *Component Destination Table Structure* to enable packet routing and transmission through one or more egress interfaces.
  - b. Component M0 is powered-up and in C-Down.
  - c. Interface P0-I0 and M0-I0 are powered-up and in I-Down.
2. P0-I0 and M0-I0 complete physical layer training, and the physical layer transitions to PHY-Up, which causes the link to transition to L-Up.
3. Upon transitioning to L-Up, management configures P0-I0 (transitions to I-Up), and M0-I0 starts periodically transmitting *Link RFC* packets, and transitions to I-CFG.
4. If P0 *Component Error and Signal Event Structure* I-Event New Peer Component Detected Event == 1b, then upon receipt of any *Link RFC* packets, P0 informs management that M0 has been detected.
5. Management initiates a series of *CTL-Read and CTL-Write Packets* or *Control Read and Control Write Packets* through P0-I0 to configure M0.
6. Once management configures M0-I0, it enables the interface, and M0-I0 transitions to I-Up.
7. Once management configures M0, it enables the component, and M0 transitions to C-Up.

25 Using *Example Discovery Topologies* (B) topology as the basis (M0 contains an integrated dual-interface switch), the following steps are taken:

1. At time zero, the following statements are true (with the exception of also configuring M0-I1, the statements a-d are achieved following the same steps as the prior example):
  - a. Component P0 has been configured and is in C-Up.
  - b. Interface P0-I0 has been configured and is in I-Up.
  - c. Component M0 has been configured and is in C-Up.
  - d. Interface M0-I0 has been configured and is in I-Up.
  - e. Interface M0-I1 has not been configured and is in I-Down.
  - f. Component M1 is powered-up but has not been configured (C-Down).
  - g. Interface M1-I0 is powered-up but has not been configured (I-Down).
- 5 2. M0-I1 and M1-I0 initiate and complete physical layer training. M0-I1 link state and M1-I0 link state transition to L-Up.
- 10 3. Upon transitioning to L-Up, management configures M0-I1 (transitions to I-Up), and management configures the integrated switch to enable packet relay between interfaces M0-I0 and M0-I1. Further, M1-I0 starts periodically transmitting *Link RFC* packets, and transitions to I-CFG.
- 15 4. If M0 *Component Error and Signal Event Structure* I-Event New Peer Component Detected Event == 1b, then upon receipt of any *Link RFC* packets, M0 informs management that M1 has been detected.
- 20 5. Management initiates a series of *Control Read and Control Write Packets* through M0-I0 to configure M1.
  - a. Until M1 configuration is complete, *Control Read and Control Write Packets* shall set the following fields as follows:
    - i. Packet's DCID to M0's CID.
    - ii. Packet's SCID to P0's CID.
    - iii. Packet's DR = 1b.
    - iv. DR-Interface field contains the interface ID associated with M0-I1.
  - b. M0 receives *Control Read and Control Write Packets*, detects DR == 1b, and relays the packet to M0-I1.
  - c. M1 receives and executes *Control Read and Control Write Packets*, and returns the corresponding response packets.
- 25 6. Once management configures M1-I0, it enables the interface. M1-I0 transitions to I-Up.
- 30 7. Once management configures M1, it enables the component. M1 transitions to C-Up. Management may directly communicate with M1 with the DR = 0b.

Using *Example Discovery Topologies (C)* topology as the basis, the following steps are taken:

1. At time zero, the following statements are true:
  - a. Component P0 has been configured and is in C-Up.
  - b. Interface P0-I0 has been configured.
  - c. Component S0 is powered-up, but has not been configured (C-Down)
  - d. Interface S0-I0 is powered-up, but has not been configured (I-Down).
  - e. Link S0-I0 is powered-up, but has not begun physical layer training (L-Down).
  - f. Components M0-M3 are powered-up, but have not been configured (C-Down).
  - 40 g. Interfaces M0-I0 through M3-I0 are powered-up, but have not been configured (I-Down).
  - h. Links M0-I0 through M3-I0 are powered-up, but have not begun physical layer training (L-Down).
- 35 2. P0-I0 and S0-I0 initiate and complete physical layer training, and P0-I0 and S0-I0 transition to L-Up.
- 45 3. Upon transitioning to L-Up, management configures P0-I0 (transitions to I-Up), and S0-I0 starts periodically transmitting *Link RFC* packets.

4. If P0 *Component Error and Signal Event Structure* I-Event New Peer Component Detected Event == 1b, then upon receipt of any *Link RFC* packets, P0 informs management that S0 has been detected.
5. Management initiates a series of *Control Read and Control Write Packets* through P0-I0 to configure S0.
6. Once management configures S0-I0, it enables the interface (S0-I0 transitions to I-Up).
7. Once management configures S0, it enables the component (S0 transitions to C-Up).
10. S0-I1 through S0-I4 and M0-I0 through M3-I0, respectively, initiate and complete physical layer training. Links S0-I1 through S0-I4 and M0-I0 through M3-I0 transition to L-Up.
15. Upon transitioning to L-Up, management configures the *Component Switch Structure* in S0 to enable packet relay between S0-I0 and interfaces S0-I1 through S0-I4, and configures interfaces S0-I1 through S0-I4. Further, M1-I0 through M3-I0 0-I0 start periodically transmitting *Link RFC* packets.
10. Upon receiving *Link RFC* packets on S0-I1 through S0-I3, S0 notifies management that a new components have been detected. For each new component detected, the following steps are taken:
15. Management initiates a series of *Control Read and Control Write Packets* through S0-I0 to configure M1-M3.

## 8.7. Configuration Completion Timer and Release

20. If a component supports *In-band Management*, then the component shall maintain a Configuration Completion Timer. This timer ensures that component configuration continues to make forward progress.
  - The Configuration Completion Timer shall operate only while the component is in the C-CFG state or a component interface is in the I-CFG state.
  - The Configuration Completion Timer shall be restarted upon the receipt of each new *In-band Management* request packet.
    - If the Configuration Completion Timer expires, the component shall self-initiate a *Full Component Reset* and transition to the C-Down state. This enables the component to restart the configuration process.
  - 30. The *Core Structure* CCTO indicates the minimum Configuration Completion Timer timeout value. Further, management shall not declare a component as failed until the time indicated by the Configuration Completion Timer has passed since the last non-completed in-band management request packet was transmitted.

**Developer Note:** A component chooses a CCTO value based on the targeted solution's requirements and implementation-specific initialization needs.

## 8.8. Power Management

Power management is controlled through the component and physical layer state machines, as well as through the associated Control Space structures. In general, a component and associated physical layers automatically perform power management within the configured parameters.

## 8.8.1. Physical Layer Power Management

Physical layer power is managed through the *Interface PHY Structure*. Management configures relevant power management boundary fields, and the physical layer automatically operates within these boundaries. Physical layer power also reacts to changes in component, interface, or link states as updated by management, component sensor inputs, or changes in peer component interface and link states.

## 8.8.2. Component Power Management

There are multiple types of component power management.

### 8.8.2.1. Software-directed Power Management

Software-directed power management—software adjusts the component’s C-State to provide gross-level power control.

- Adjustment may be performed using *In-band Management* or *Out-of-band Management*.
- C-State adjustment period shall occur no faster than the sum of the entry and exit latencies associated with the impacted C-States.

### 8.8.2.2. Managed Power Removal

To remove component power, managed power removal requires management and the impacted components perform a sequence of steps to ensure deterministic behavior. Further, these steps should be taken prior to component removal.

The following steps are taken to perform managed power removal from a component:

1. Software shall set *Core C-Control* Transition Component Power-Off = 1b.
  - If the component supports a Logical PCI Device (LPD), then writing to this bit is equivalent to the component receiving a PCIe PME\_Turn\_Off message.
2. The component shall complete processing all received end-to-end request and response packets. Upon completion, for any subsequently received request packet, the component shall return a *Standalone Acknowledgment* (Reason = Component Power-off Transition).
- 25 3. If *Core Structure Component CAP 1 Control* Notify Management on Power Event == 1b, then the component shall notify management. If the component is using *In-band Management*, then it shall transmit an *Unsolicited Event (UE) Packet* with Event = Component Power-off Transition Completed.
  - If the component supports a Logical PCI Device (LPD), then management shall set *Core Structure Component CAP 1 Control* Notify Management on Power Event = 1b. The resulting transmission of an *Unsolicited Event (UE) Packet* with Event = Component Power-off Transition Completed is the equivalent of transmitting a PCIe PME\_TO\_Ack message.
- 30 4. Once management has been notified, the component shall stop transmitting new end-to-end request and response packets.
  - If the component is using *Out-of-band Management*, then the component shall silently discard all newly received end-to-end packets.
  - If the component is using *In-band Management*, then:

- The component shall silently discard all new request packets that target Data Space.
- If the UERT timer expires, then the component may retransmit the *Unsolicited Event (UE) Packet* with Event = Component Power-off Transition Completed.
- Once *Core C-Control* Halt UERT field == 1b, the component shall silently discard all new request packets.

5 5. For each component interface, the component shall immediately take any physical layer-specific actions.

- 10 o If the interface supports the PCI Express physical layer and LTSSM, then the component shall initiate a PM\_Enter\_L23 *Link CTL* packet exchange. Once the exchange is initiated, the interface shall silently discard all packets other than those required to complete the *Link CTL* packet exchange.

15 6. Once all physical layer-specific processing has completed, then the component shall set *Core C-Status* Power-Off Transition Completed = 1b (this enables *Out-of-band Management* to detect when it is safe to remove component power).

### 8.8.2.2.1. Asynchronous Power Removal

To remove component power using the asynchronous power removal model relies on methods outside of this specification's scope to identify the component and ensure that power can be safely removed. Unlike managed power removal, removal may occur without requiring component interaction. As such, asynchronous power removal asynchronously impacts the target component's operation as well as any components that are directly-attached or were communicating with the target component.

20 The following steps shall be taken by the target component:

- 25 1. If the target component supports the *Component Error and Signal Event Structure* and Unexpected Component Power Loss Detect == 1b and there remains sufficient power and time to do so, then the component shall take the configured actions.
2. The component shall transition to the C-Down state.

The following steps shall be taken by all components that are directly attached to the component that lost power:

- 30 1. For each interface that is directly-attached to the target component, if *I-Error Detect* Unexpected Physical Layer Failure—PHY-Down Detect == 1b, then the interface shall take the configured actions.
2. If Interface Containment is triggered and *Interface I-CAP 1 Control* Interface-Component Containment Enable == 1b, then the component shall trigger *Component Containment*.
- 35 3. If component containment is not triggered, then for each interface that is directly-attached to the target component, transition the link to the L-Down state and the interface to I-Down. Upon transitioning to I-Down, the interface shall take the steps specified in *Interface States and Descriptions* (transition to I-Down caused by L-Down transition).

The following steps shall be taken by a peer component that was communicating with the target component:

- 40 1. If a Requester and it supports the *Component Error and Signal Event Structure* and C-Error End-to-end Packet Retry Error Detect = 1b, then the component shall take the configured actions.

### 8.8.2.3. Power Restoration

The following steps are taken to restore power to a component.

1. The component shall set *Core C-Control Transition Component Power-Off* = 0b.
2. For each component interface, the component shall take any physical layer-specific actions required to transition the physical layer to PHY-Up.
3. Once the physical layer has transitioned to PHY-UP, transition the corresponding link to L-Up.
4. If the component was able to maintain interface configuration, then the interface shall transition to I-Up. If interface configuration was not maintained, then the interface shall be in I-Down, and management shall configure the interfaces.
5. If the component was able to maintain component configuration and *Core Structure Component CAP 1 Control Notify Management on Power Event* == 1b, then the component shall notify management. If the component supports *In-band Management*, then it shall transmit an *Unsolicited Event (UE) Packet* with Event = Component Power Restoration Event.
  - o If the component supports *Logical PCI Device (LPD)*, then management shall set *Core Structure Component CAP 1 Control Notify Management on Power Event* = 1b. The resulting management notification (component-local interrupt or an *Unsolicited Event (UE) Packet* with Event = Component Power Restoration Event) is the semantic equivalent of a PCIe PME\_PME message.
  - o If the component was not able to maintain component configuration, then the component shall be in C-Down, and management shall perform configuration accordingly.

### 8.8.2.4. Automatic Power Management

Automatic power management—based on exceeding the minimum idle time, the component exits the present C-State and enters a lower-power C-State.

- Prior to automatically transitioning to C-DLP, the component shall notify each enabled peer component interface of the transition using a *Link CTL* packet exchange. This exchange enables the peer component to take additional steps:
  - o If the component is not an end-to-end packet's destination and an enabled alternative path exists, then the peer component may bypass this component.
  - o Enable the link to transition to a lower-power level (L-Up-LP or L-LP).
- If a component is in C-DLP and a link transitions from L-Down to L-Up, then the component shall notify the peer component interface of the corresponding link that the component has transitioned to C-DLP using a *Link CTL* packet exchange.
- If a component supports the *C-State Power Control* packet, then:
  - o A component may use this packet to inform its peer components of C-State and power consumption level changes.
  - o A component may be requested to transition to a new C-State and / or power consumption level for a duration of time.
- If a component is not in the C-Down, C-Up, or C-CFG state, then upon receipt of an end-to-end packet, the component shall automatically transition to the C-State required to execute the packet, e.g., if the component is in C-LP and it receives a Core 64 OpClass packet, then the component transitions to C-Up.

### 8.8.2.5. Component-specific Power Management

Component-specific power management—a component may provide fine-grain power control within a given C-State using component-specific automatic or software-directed mechanisms.

- 5 If a component contains an integrated switch or a transparent router, the component shall ensure the switch or transparent router remains fully-operational independent of any component-specific power management adjustments.

### 8.8.2.6. Emergency Power Reduction

Under Emergency Power Reduction—the response to unexpected changes in environmental or safety conditions, the component immediately reduces power consumption to less than or equal to the maximum power indicated by *Core Structure* EPWR.

- 10 Emergency Power Reduction shall override all other power management mechanisms except for power down and Power Disable (if supported).
- 15 Emergency Power Reduction shall apply to the entire component package or mechanical module.
- 20 Emergency Power Reduction may be initiated by management. Due to the time-sensitive nature of environmental or safety conditions, management may use a package-specific or mechanical form factor-specific out-of-band Emergency Power Reduction signal to inform the component (other technologies may refer to this signal as PWRBRK). While this signal is asserted, the component shall not exceed the maximum emergency power level.
  - When the Emergency Power Reduction signal is negated, the component may resume normal operation and consume up to its maximum enabled power.
  - 25 If a package or mechanical form factor contains multiple components, then the Emergency Power Reduction signal shall be routed to all components.
    - Each component that supports Emergency Power Reduction shall not exceed its respective EPWR power level.
  - Management may trigger emergency power reduction by setting *Core C-Control* Trigger Emergency Power Reduction = 1b, though this approach may incur additional latency.
- 30 Emergency Power Reduction may be initiated by the component based on component-specific sensor inputs. If these sensors indicate the operating conditions have degraded to an unsafe level, then the component may automatically self-trigger Emergency Power Reduction. The component shall not exceed the EPWR level until the sensors indicate the operating conditions have maintained a safe level for a minimum of 10 seconds, or a *Full Component Reset* or a *Warm Component Reset* is initiated.
- 35 When Emergency Power Reduction is initiated, the component shall set *Core C-Status* Emergency Power Reduction Detected = 1b.
- 40 To exit Emergency Power Reduction, management shall set *Core C-Control* Exit Emergency Power Reduction = 1b, and if applicable, then the component or mechanical form factor Emergency Power Reduction signal shall be negated.
- Until a component exits Emergency Power Reduction,
  - If a Responder lacks sufficient power to execute a request packet, then it shall transmit a *Standalone Acknowledgment* (Reason = Emergency Power Reduction).
  - If a Requester lacks sufficient power to execute a response packet, then it shall initiate Requester-specific error handling.

- A discrete switch, a transparent router, or a component with an integrated switch or transparent router shall continue to relay packets. If it cannot, then the component shall silently discard all end-to-end packets intended for packet relay.
- If a component is unable to maintain its configuration state, volatile media contents, bound addresses, etc., then when the component exits Emergency Power Reduction, management software, service-specific software, and / or component-specific software shall reconfigure the component, restore volatile media contents, bound addresses, etc. *Core Structure Component CAP 1 Configuration Post Emergency Power Reduction* indicates if software configuration is required.
- In order to minimize the number of out-of-band signals, Emergency Power Reduction and Power Disable may share the same out-of-band signal (the mechanical form factor or component specify a shared pin or separate pins).
  - If shared, then Emergency Power Reduction and Power Disable should not be enabled at the same time.
  - If shared and both are enabled, then when the signal is asserted, Emergency Power Reduction shall be triggered first followed by Power Disable if the signal remains asserted for the requisite amount of time.

### 8.8.2.7. Power Disable

Power Disable is an out-of-band signal that may be used to disable power to the component's logic.

Though management may trigger Power Disable for multiple reasons, the most common will be to reset an unresponsive or untrusted component through selective power cycling.

- Power Disable uses a package-specific or mechanical form factor-specific signal.
- If the Power Disable signal is asserted for 5 seconds, then power to the component shall be disabled.
- If the Power Disable signal is negated for 100 ms, then power to the component may be enabled. Upon power restoration, the component shall perform the same steps that follow a power-on reset.
- If a package or mechanical form factor contains multiple components, then the Power Disable signal shall be routed to all components.
  - Power shall be enabled or disabled to each component that supports Power Disable.
- When the Power Disable signal is asserted, the component shall stop drawing power.
- If a component supports media backup services, then when the Power Disable signal is asserted, the component should initiate media backup, e.g., copying the contents of volatile media to non-volatile media.

**Developer Note:** *Multiple industry standards support Power Disable functionality. To the extent possible, Gen-Z Power Disable semantics align with these other industry standards to ensure a consistent customer experience.*

### 8.8.3. C-State Transition and Power Management

Transitioning states requires exiting the current state and entering the new state. The following are the requirements associated with C-State transitions:

- Prior to exiting to a lower-power state, a component may need to:
  - Store Control Space structures, images, etc. to non-volatile media

- Store cached configuration state to non-volatile media
- Transition volatile media to a non-operational, low-power, self-refresh state
- Store volatile media contents to non-volatile media
- As part of entering a lower-power state, a component may need to:
  - If so configured, inform management of the transition.
  - If so configured, inform directly-attached peer component interfaces of the transition using a *Link CTL* packet exchange.
  - Slow or halt computational engines or functional units to operate within the new state's power constraints
  - Transition physical layers of one or more component interfaces to an operational or non-operational low-power state
- Prior to exiting to a higher-power state, a component may need to:
  - Restore Control Space structures, images, etc. from non-volatile media
  - Rebuild cached configuration state from non-volatile media
  - Transition volatile media to an operational state
  - Restore volatile media contents from non-volatile media
- As part of entering a higher-power state, a component may need to:
  - If so configured, inform management of the transition.
  - If so configured, inform directly-attached peer components interfaces of the transition using a *Link CTL* packet exchange.
  - Restart or return computational engines or functional units to operate within the new state's power constraints
  - Transition physical layers of one or more component interfaces to an operational or higher operational state. Once operational, physical layers may automatically adjust their state to meet workload requirements.
- The *Core Structure* provides three fields corresponding to C-Up, C-LP, and C-DLP. These fields describe the entry and exist latencies when transitioning between C-States and the minimum idle latency before a component automatically transitions to a lower power state (if applicable).
  - The C-Up Time field contains five values—the exit latencies from C-LP to C-Up and from C-DLP to Up, the entry latencies from C-Up to C-LP and from C-Up to C-DLP, and the minimum time a component is idle in C-Up before the component automatically transitions to the C-LP state.
  - The C-LP Time field contains four values—the exit latency from C-LP to C-Up, the exit latency from C-LP to C-DLP, the entry latency from C-Up to C-LP, and the minimum time a component is idle in C-LP before the component automatically transitions to the C-DLP state.
  - The C-DLP Time field contains three values—the exit latency from C-DLP to C-Up, the entry latency from C-LP to C-DLP, and the entry latency from C-Up to C-DLP. A component exits the C-DLP state either through an *Out-of-band Management* configuration update, an in-band link-local state change request, or a *Full Component Reset* or a *Warm Component Reset*.
- Management should consider C-State transition latency impacts on Requester retransmission timers.
- A component may be configured to notify management and / or its peers when transitioning between C-States.
  - Peer notification shall be performed using a *Link CTL* exchange.

- Management notification may use a component-specific interrupt or an *Unsolicited Event (UE) Packet*.
- A peer may be configured to notify management when it detects its peer transitioning to C-DLP.

#### 8.8.4. Vendor-defined Power Management

5 Vendor-defined power management is controlled through the vendor-defined structures accessed through applicable control structures, or accessed through means outside of this specification's scope. Vendor-defined power management is outside this specification's scope. The power-on default state of vendor-defined power management shall not usurp or conflict with power management as specified in this document. The solution designer is responsible for every aspect of vendor-defined power  
10 management and, when it is explicitly enabled, any impact it has on features that are controlled by the specification.

### 8.9. Thermal Management

15 Component-specific thermal management is accessed through the *Core Structure Thermal Attributes*, Upper Thermal Limits, and Caution Thermal Limits. Sub-fields within the structure provide status and component-level controls. While a component is in C-Up, the Component Thermal Status indicates one of four thermal operating conditions:

- Nominal operating conditions—the component is operating below its Caution Thermal Limit.
- Caution Thermal Limit—the component is fully operational, but is approaching its Upper Thermal Limit. Performance throttling may be invoked to reduce thermal load. The Caution Thermal Limit shall be set to one of the following values:
  - 90% of the Upper Thermal Limit, measured in degrees Celsius.
  - Any value less than the Upper Thermal Limit at which the component should be treated differently.
- Exceeded Upper Thermal Limit—the component is operational, but potentially unsafe thermal environmental conditions could cause instability in the component's operation or lead to premature failure if this condition persists for a long time period. Performance throttling may be invoked to prevent component damage or failure.
- Thermal shutdown—the component has detected unsafe thermal environmental conditions. Component functional operations shall be reduced to only what is required to inform management of the shutdown event, i.e., the component remains in C-Up but shall execute only *In-band Management* packets (if supported) or *Out-of-band Management* operations. The component shall not generate or execute any non-*In-band Management* request or response packets.

### 8.10. UUIDs

35 A UUID is 128-bit Universally Unique Identifier. A UUID shall be as specified in ITU-T X.667. Gen-Z uses UUIDs for a variety of reasons, e.g., to simplify component and software development, ease solution composition, and increase developer agility and their ability to cooperate without requiring a central authority's approval. Unless a UUID value is explicitly defined by this specification, a UUID is generated through means outside of the scope of this specification, e.g., dynamically generated using an Internet-based UUID generation service.  
40

Gen-Z specifies the following UUIDs (only a small subset is required by all components):

- Z-UUID—The Z-UUID is used to definitively indicate Gen-Z support even if management accesses Control Space through an *Out-of-band Management* interconnect. The Z-UUID is specified by this document and is located within the *Core Structure*.
- C-UUID—The C-UUID is used to delineate component types, e.g., two memory modules from one another. If software needs to be associated with the component and the component supports a single service type, then an OS can use the C-UUID to identify the corresponding software module. This UUID is located within the *Core Structure*.
- FRU-UUID—The FRU-UUID is used to identify a collection of components that are treated as a single field replaceable unit (FRU) for hardware management purposes, e.g., enclosure or a mechanical form factor that contains multiple components. This UUID is located within the *Core Structure*.
- MGR-UUID—The MGR-UUID is used to identify the primary owner of the component, i.e., either the Primary Manager or the Primary Fabric Manager and Secondary Fabric Manager. The owning manager is responsible for generating this UUID. Further, the MGR-UUID is used to validate access permission to Control Space when executing *Control Read* and *Control Write Packets*. This UUID is located within the *Core Structure*. If the owner enables authorized managers to perform *In-band Management* using Control OpClass packets that contain a MGR-UUID field, then the owner is responsible for distributing this value to authorized managers.

**Developer Note:** *It is strongly recommended that management generate a new MGR-UUID value prior to configuring a topology, i.e., not use a static or default MGR-UUID value.*

**Developer Note:** *Use of a MGR-UUID is situational. For example:*

- *Typically, a single-enclosure topology is managed by the Primary Manager, and does not require a MGR-UUID to identify the manager or to validate access permission.*
- *Typically, a multi-enclosure topology is managed by the Primary Fabric Manager and optionally by the Secondary Fabric Manager. To prevent unauthorized fabric managers from being attached to the topology and gaining control of components, developers are strongly encouraged to have the Primary Fabric Manager program the MGR-UUID field of each component that it owns, and for resiliency and / or performance reasons, it share this MGR-UUID with the Secondary Fabric Manager.*
- PM-UUID—The PM-UUID is used to identify vendor-defined operations within the P2P-Vendor-defined and Multicast OpClasses. The PM-UUID is located within the *OpCode Set Structure*.
- Vendor-defined OpClass UUID—A component may support up to 8 Vendor-defined OpClasses. Each Vendor-defined OpClass is identified by a Vendor-defined OpClass UUID located within the *OpCode Set Structure*.
- Vendor-defined UUID—Components may support vendor-defined control structures. A vendor-defined structure may be identified by the C-UUID or if provisioned within the structure, a vendor-defined UUID. The Vendor-defined UUID is located within the *Vendor-Defined Structure*.
- Primary Media UUID—The Primary Media UUID is used to identify addressable primary media within a component. This enables management to delineate between media types or generations of a given media type. The Primary Media UUID may be supplied by the vendor, be a de facto standard UUID, or an industry standard body-defined UUID. The Primary Media UUID is located within the *Component Media Structure*.
- Secondary Media UUID—The Secondary Media UUID is used to identify the secondary media, if present, within a component. This enables management to delineate between media types or

generations of a given media type. The Secondary Media UUID may be supplied by the vendor, be a de facto standard UUID, or an industry standard body-defined UUID. The Secondary Media UUID is located within the *Component Media Structure*.

- Certificate UUID—The Certificate UUID is used to identify security certificate types. The Certificate UUID may be supplied by the vendor, be a de facto standard UUID, or an industry standard body-defined UUID. The Certificate UUID is located within the *Component Security Structure*.
- Image UUID—The Image UUID is used to identify images, e.g., firmware type and version, OS type and version, videos, etc. The Image UUID is supplied by the image creator, by management, etc. The Image UUID is located within the *Component Image Structure*. This structure may contain multiple Image UUID.
- MCTP UUID—The MCTP UUID is used to identify the MCTP protocol when it operates over Gen-Z. The MCTP UUID is specified by this document in *MCTP*.
- Service-UUID—if a component supports multiple services that require unique handling or service-specific software to be associated, then the Service-UUID is used to identify each unique service type. For example, a component that contains a mix of accelerators or a native Gen-Z I/O component that supports storage and network services. The Service-UUID is located within the *Service UUID Structure*.
- MSE-UUID—if a Requester or a Responder supports multi-subnet unicast communications, then the MSE-UUID is used to identify the encoding mechanism used to access the MSAP within the Component PA (Peer Attribute) Structure.
- MCE-UUID—if a component supports multi-subnet multicast communications, then the MCE-UUID is used to identify the encoding mechanism used to access the multi-subnet multicast tables within the *Component Switch Structure* and the *Component Multicast Structure*.
- SFF 8489 UUID—see *Component Mechanical Structure*

## 8.11. MCTP

MCTP is a management protocol specified by the DMTF (Distributed Management Task Force). MCTP is used to communicate management messages between hardware components that compose a platform management subsystem. MCTP will be used by Gen-Z management solutions that support technologies such as the DMTF Redfish.

The following are the features and requirements associated with MCTP:

- If MCTP is transported using a vendor-defined Explicit OpClass, then the corresponding UUID shall be:  
0x8ead46dfe83a4a3d8e8b308994dec439

## 8.12. Control Space structures

Configuration and management is accomplished through a set of Control Space structures. Control Space structures exist in Control Space and are linked together via pointers. *Example Minimalist Control Space Structure Layout* and *Example of a Feature-Rich Control Space Structure Layout* illustrate minimalist to feature-rich configuration and management.

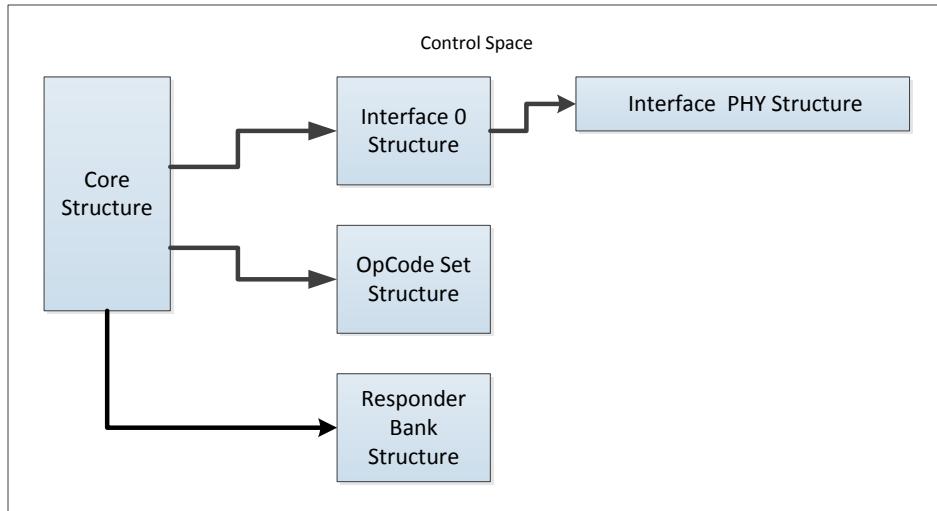


Figure 8-4: Example Minimalist Control Space Structure Layout

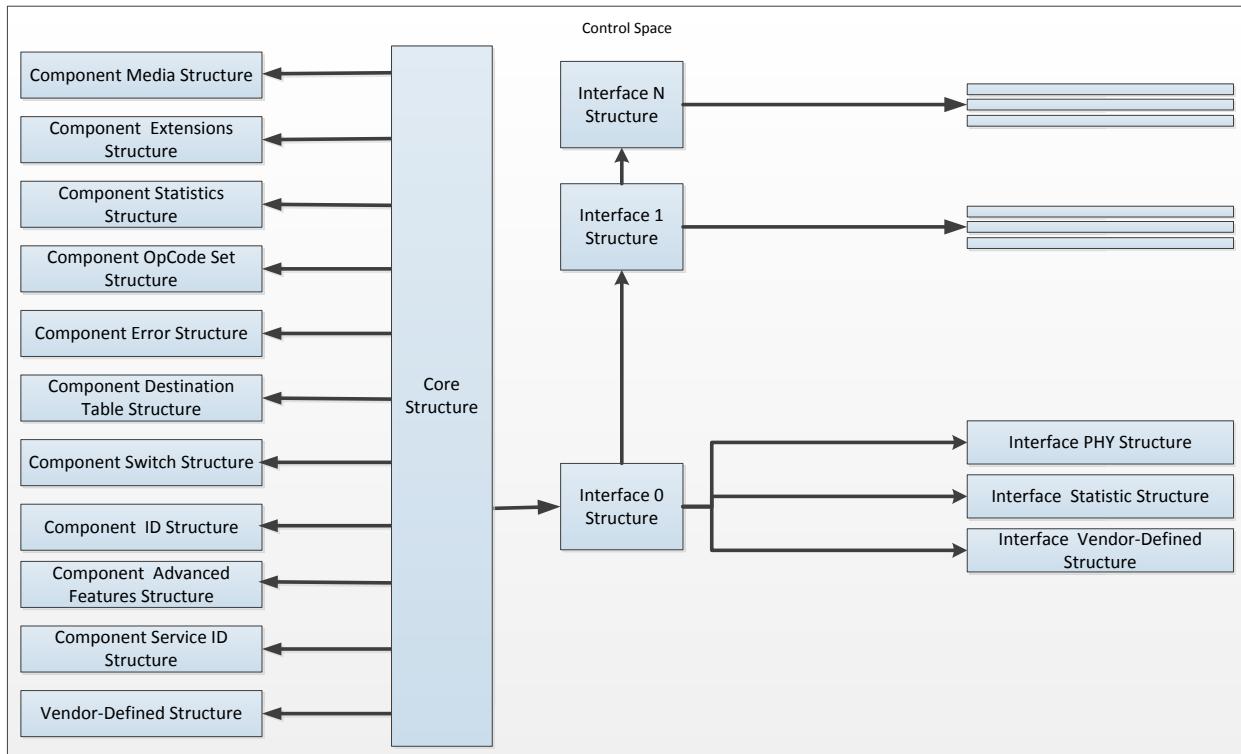


Figure 8-5: Example of a Feature-Rich Control Space Structure Layout

- 5 Control Space structures are configuration and management structures that only exist within Control Space. Control Space structures are self-describing allowing a variety of structure mix and organization tailored to solution-specific needs.

The following are the features and requirements associated with Control Space structures:

- Control Space structures shall be placed within the Control Space. With the exception of the *Core Structure*, a Control Space structure may be placed anywhere within Control Space.
- All Control Space structures shall be accessible through the component's out-of-band interface or through *In-band Management*.

- Control Space structures contain the following three fields:
  - Type—a unique 12-bit unsigned integer value that identifies the structure type; the value is specified in *Appendix A Control Space Structure Encodings*.
  - Version—a unique unsigned integer value that identifies the version of the structure. Unless explicitly stated otherwise by this specification, the version of a Control Space structure is 0x1.
  - Size—a 16-bit unsigned integer that represents the structure size in integer 16-byte multiples, i.e., the actual size is (Size \* 16) bytes. Unless explicitly stated otherwise by this specification, the Size field reflects only the size of the control structure itself; it does not reflect the size of any other structures that are located by pointers contained within the control structure.
- The Control Space address corresponding to byte zero of a Control Space structure shall be an integer 16-byte multiple.
- If a pointer field is used to locate a Control Space structure, then the pointer field shall either be set to zero to indicate a Null pointer (no such structure) or set to Control Space address of byte zero of the Control Space structure.
  - Control Space structure pointers are provisioned in the *Core Structure*, whose pointers are the first step in reaching all other Control Space structures.
  - The *Component Extension Structure* may be used to provision additional Control Space pointers.
  - If a given Control Space structure supports multiple instances, then these may be linked together through structure-specific pointers.
  - Pointer fields may be 32-bit or 48-bits (unused upper pointer field bits shall be set to zero).
    - All control structure fields with a “PTR” in the name are Control Space pointers.
    - Unless explicitly specified otherwise in this specification, a Control Space pointer shall refer to a zero-based Control Space address with integer 16-byte addressability, i.e., the actual location of byte zero of a Control structure is at (Pointer \* 16) bytes. For example, a 32-bit pointer enables a Control Space structure to be located anywhere within the first  $2^{36}$  bytes of Control Space.
  - A Control Space structure may contain pointers to non-Control Space structures. These structures may be vendor-defined or specified by this document.
  - With the exception of the *Core Structure*, control structures may be located anywhere within the supported Control Space, i.e., they do not need to contiguously placed within Control Space. Further, to enable multiple management entities to independently manage Control Space, control structures should be organized onto one or more pages on which hardware-enforced isolation is applied (see *Component C-Access Structure* and *Control Structure Organization*).
- Fields designated as Reserved, RsvdP, or RsvdZ shall be set to zero. Unless explicitly stated otherwise by this specification, Reserved, RsvdP, and RsvdZ fields shall be hardwired to zero.
- If a feature associated with the field is unsupported, then the field shall be hardwired to zero.
- Control Space structure fields are designated as M (mandatory), O (Optional), or MC (Mandatory Conditional). A mandatory field shall be implemented if the control structure is supported. An Optional field may be implemented; if not, it shall be Reserved. A mandatory conditional field shall be implemented if the specific associated capability is supported.
- If supported, a component may use the *Component C-Access Structure* to provide fine-grain protection of Control Space structures.

- A component may translate configuration structure fields into an implementation-specific set of internal resources, effectively caching the configuration structure state. Whenever the *Core C-Control Refresh Component Configuration* == 1b, the component shall refresh its internal resources and operational state to reflect the current configuration fields. A component may cache only configuration fields that explicitly indicate that their contents may be cached.
- Unless specifically stated otherwise by this specification, all vendor-defined structures, vendor-defined statistics, etc. shall be associated with the *Core Structure C-UUID* field.
- Unless specifically stated otherwise by this specification, if management sets a supported non-RO control structure field to an unsupported value, then the component's behavior may be undefined.
- If a reserved timer value is configured, a component should use the maximum permitted value.
- If writing 1b to a read-write control bit causes an action to take place, then writing 0b to the bit shall have no impact on the corresponding action or underlying operation of the component.
- If a capability is unsupported (e.g., a read-only capability field or bit indicates unsupported), then the corresponding control field or bit shall be Reserved.

## 8.13. Control Structure Organization

The organization of structures in Control Space can vary widely as a function of the access mechanisms used as well as the number and type of entities that directly access Control Space.

If a solution supports only *Out-of-band Management*, then the Primary Manager has exclusive access to all structures, and there's no need for hardware to enforce different access permissions for different entities directly accessing Control Space. Structures in Control Space can be placed as needed to optimize hardware logic, and can be closely packed. The Primary Manager proxies any accesses to structures needed by other managers or an Operating System (OS).

If a solution supports *In-band Management* and multiple managers or OS instances need to directly access Control Space, then for security purposes, hardware needs to enforce different access permissions for these different entities, and structures need to be placed in different Control Space regions ("pages") in order to enable this.

Many (if not most) structures will have a single active manager or "owner", which by necessity has RW permission. Most non-owners will have RO permission or no visibility of the structure. The owner can provide proxy mechanisms such as APIs to enable non-owners to determine capabilities and current settings, or request changes. For example, a fabric manager owns the control structures required to configure the fabric to exchange packets, e.g., routing tables, congestion management, etc. The fabric manager proxies requests from a performance manager (non-owner) that tunes the fabric to meet application performance needs. In this example, the fabric manager has RW access and the performance manager has RO access. Proxy mechanisms are outside of this specification's scope.

A *Component C-Access Structure* provides hardware-enforced access control over Control Space structures, with granularity as fine as a per-structure basis. However, this level of granularity requires placing each structure on a dedicated page, and requires a dedicated entry in a C-Access R-Key Table on the Responder as well as a dedicated ZMMU PTE entry in the Requester. To reduce hardware resources, structures that will be owned by a single manager and RO accessed by a single set of non-owners can be co-located on a single page. Structures that will be owned by a single manager and selectively RO accessed by multiple non-owners or independently RW accessed by a resource manager (e.g., an OS) need to be placed on separate pages.

Hardware grants a fabric manager inherent RW access to all Control Space structures, by virtue of its role. Though a fabric manager retains RW access, it can enable RW access of selected structures to other managers. Though policies for granting access permissions to specific Control Space structures is determined by fabric managers, and are outside of this specification's scope, this section is based on the following use cases and assumptions:

- For some environments where managers own most structures, one or more OS instances will have RO access to many of the structures. Notable exceptions are those structures containing private cryptographic keys.
- For some environments, notably ones with simple or trivial fabrics, an OS will own many structures and have RW access to them.
- For some environments, the fabric manager will grant other managers or OSs RO access to certain structures so that they can locate various other structures by following pointers, possibly starting with the *Core Structure*.
- For certain structures, the owner can provide non-owners a pointer to each one, avoiding the need for those non-owners to follow pointers in other structures in order to locate them.
- Some structures require direct access by certain managers for performance reasons, e.g., statistics, performance records, power management, etc.

The following subsections cover various groupings of Control Space structures that are assumed to be owned generally by a single manager, or else have unique characteristics.

### 8.13.1. Grouping: Baseline Structures

These structures should be owned by the Primary Manager, the Primary Fabric Manager, the Secondary Fabric Manager, or a trusted resource manager, and should be co-located on one or more pages independent of all other structures.

- *Core Structure*
- *OpCode Set Structure*
- *Interface Structure*
- *Interface PHY Structure*
- *Component Extension Structure*
- *Component C-Access Structure*
- *Component RKD Structure*

### 8.13.2. Grouping: Routing / Fabric Structures

These structures should be owned by the Primary Fabric Manager, the Secondary Fabric Manager, or a trusted resource manager, and should be located on one or more pages independent of all other structures. This grouping is distinct from the Baseline structures grouping to enable the two groupings to be owned by different resource managers.

- *Component Switch Structure* and related substructures
- *Component Multicast Structure*
- *Component TR Structure*
- *Component Destination Table Structure* and related substructures
- *Component PA (Peer Attribute) Structure* and related substructures
- *Component SOD Structure*

- *Congestion Management Structure*
- *Component Precision Time Structure*

### 8.13.3. Grouping: Component LPD / LPH Structures

5 The *Component LPD Structure* and *Component LPH Structure* are unique in their characteristics. There may be multiple instances within a component, and each instance has an associated region that is usually owned by a different entity.

10 Each *Component LPD Structure* or *Component LPH Structure* is generally owned by the Primary Manager, the Primary Fabric Manager, the Secondary Fabric Manager, or a trusted resource manager. If there are multiple *Component LPD Structure* or *Component LPH Structure* instances, they may be co-located on one or more pages.

15 Each *Component LPD Structure* is associated with one or more 4 KiB regions representing PCI/PCIe Configuration Space and owned by the host OS. All of these 4 KiB regions associated with a single *Component LPD Structure* shall be owned by a single OS. If the page size is greater than 4 KiB, then these 4 KiB regions should be co-located on one or more pages.

20 15 If a component provisions multiple *Component LPD Structure* instances, and some or all can be owned by different OSs, then their associated 4 KiB regions potentially owned by different OSs shall not be co-located on the same page.

25 Each *Component LPH Structure* has an associated LPH ECAM region that maps PCI/PCIe Configuration Space and is owned by the host OS. The entire LPH ECAM region associated with a single *Component LPH Structure* shall be owned by a single OS. The LPH ECAM maps an integer number of PCI bus numbers, each requiring 1 MiB of address space. Thus, large pages should be used for increased efficiency.

30 Each *Component LPH Structure* may have an associated LPH RCRB region presenting a Root Complex Register Block. If present, the LPH RCRB shall be owned by the host OS.

### 25 8.13.4. Grouping: Media Structures

These structures are generally owned by a media resource manager. They may be co-located on one or more pages.

- *Component Media Structure*
- *Responder Bank Structure*
- *Requester Bank Structure*

### 8.13.5. Grouping: Vendor-Defined Structures

35 These structures are generally owned by a manager or OS driver running vendor-specific software. Though their co-location policies are determined by the vendor, these structures should not be co-located on pages containing structures outside of this grouping. Locating these structures on different pages enables the fabric manager to provide independent control over which other managers or OSs can access them.

- *Component Statistics Structure* with ST-Type of a vendor-defined OpClass or statistic structure
- *Vendor-Defined Structure* without a UUID field

- *Vendor-Defined Structure* with UUID structure

### 8.13.6. Structures Requiring Dedicated Pages

Each of these structures may be owned by a different resource manager or OS instance. Each of these structures should be located on a dedicated page. For each structure that supports having multiple instances, the list below addresses them being co-located versus located on dedicated pages (i.e., being isolated).

- *Interface Statistics Structure*—multiple instances should be co-located
- *Component Error and Signal Event Structure*
- *Component Statistics Structure*—multiple instances should be co-located
- *Component Security Structure*
- *Component Image Structure*—multiple instances shall be isolated
- *Component Mechanical Structure*—multiple instances should be co-located
- *Component Event Structure*
- *Component ATP Structure*
- *Component PM Structure*
- *Service UUID Structure*
- *Component RE Table Structure*

## 8.14. Core Structure

The following are the features and requirements associated with the Core Structure:

- A component shall support the Core structure, and the Core structure shall use the *Core Structure Format*.
- The Core Structure shall be located at Control Space address zero.
- Unless explicitly stated otherwise by this specification, unsupported status, capabilities, and control bits shall be hardwired to zero.
- Components that support only the P2P-Core, P2P-Coherency, or P2P-Vendor-defined OpClasses may be managed through an out-of-band interconnect or using in-band packets.
- Components may be managed by one of three potential managers:
  - A component may be managed by a Primary Manager e.g., by system firmware or by a combination of system firmware and an operating system. The component that contains the Primary Manager is identified by the PMCID (components that support only *Out-of-band Management* do not use the PMCID field to identify the Primary Manager).
  - A component may be managed by a Primary Fabric Manager, e.g., a network operating system. The component that contains the Primary Fabric Manager is identified by the PFMCID. If the component is located in a different subnet, then the component is identified by the [PFMCID, PFMSID].
    - A component may be managed by a Secondary Fabric Manager (redundant or federated manager). The component that contains the Secondary Fabric Manager is identified by the SFMCID. If the component is located in a different subnet, then the component is identified by the [SFMCID, SFMSID].

+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	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	7 6 5 4 3 2 1 0																			
R0				Size		Vers	Type	< Byte 0x0																		
C-Status																										
C-Control																										
WRLAT		RDLAT		R-BIST	Max Interface		Base C-Class																			
C-Op-Clock			Max REQ Supported Requests			Max RSP Supported Requests																				
Max Data																										
R1	Max RNR	Max CTL																								
R2	C-LP Time		C-DLP Time		C-Up Time																					
R3	APWR		EPWR		HPWR	NPWR		LPWR																		
Control Structure PTR 1				Control Structure PTR 0																						
Control Structure PTR 3				Control Structure PTR 2																						
Control Structure PTR 5				Control Structure PTR 4																						
Control Structure PTR 7				Control Structure PTR 6																						
Core LPD BDF Table PTR				Control Structure PTR 8																						
Component C-Access PTR				OpCode Set Structure PTR																						
Interface 0 PTR				Component Destination Table PTR																						
Component Error and Signal Event PTR				Component Extension PTR																						
FAILTO		CCTO		CRPTO		MUTO																				
UNRSP		LVETO		SVETO																						
UNREQ		ATSTO		NIRT		UERT																				
PCO FPST		FPST		Deadline Tick		NLL Request Deadline	LL Request Deadline																			
SFMSID		ENIRT		R4	Responder Deadline	NLL Response Deadline	LL Response Deadline																			
SFMCID		PFMSID		PFMCID		PWR MGR CID	PMCID																			
R5	SID 1			CID 1		SCV1	SID 0		CID 0	SCV0																
R6	SID 3			CID 3		SCV3	SID 2		CID 2	SCV2																
MI-MSB	MI-LSB	PWR MGR SID			R7	Min Tag Cycle		Max Requests																		
+15	+14	+13	+12	+11	+10	+9	+8	+7	+6	+5	+4	+3	+2	+1	+0											
TO <sub>7</sub>	TO <sub>6</sub>	TO <sub>5</sub>	TO <sub>4</sub>	TO <sub>3</sub>	TO <sub>2</sub>	TO <sub>1</sub>	TO <sub>0</sub>																			
TO <sub>15</sub>	TO <sub>14</sub>	TO <sub>13</sub>	TO <sub>12</sub>	TO <sub>11</sub>	TO <sub>10</sub>	TO <sub>9</sub>	TO <sub>8</sub>																			
TO <sub>23</sub>	TO <sub>22</sub>	TO <sub>21</sub>	TO <sub>20</sub>	TO <sub>19</sub>	TO <sub>18</sub>	TO <sub>17</sub>	TO <sub>16</sub>																			
TO <sub>11</sub>	TO <sub>30</sub>	TO <sub>29</sub>	TO <sub>28</sub>	TO <sub>27</sub>	TO <sub>26</sub>	TO <sub>25</sub>	TO <sub>24</sub>																			
LTO <sub>7</sub>	LTO <sub>6</sub>	LTO <sub>5</sub>	LTO <sub>4</sub>	LTO <sub>3</sub>	LTO <sub>2</sub>	LTO <sub>1</sub>	LTO <sub>0</sub>																			
LTO <sub>15</sub>	LTO <sub>14</sub>	LTO <sub>13</sub>	LTO <sub>12</sub>	LTO <sub>11</sub>	LTO <sub>10</sub>	LTO <sub>9</sub>	LTO <sub>8</sub>																			
LTO <sub>23</sub>	LTO <sub>22</sub>	LTO <sub>21</sub>	LTO <sub>20</sub>	LTO <sub>19</sub>	LTO <sub>18</sub>	LTO <sub>17</sub>	LTO <sub>16</sub>																			
LTO <sub>31</sub>	LTO <sub>30</sub>	LTO <sub>29</sub>	LTO <sub>28</sub>	LTO <sub>27</sub>	LTO <sub>26</sub>	LTO <sub>25</sub>	LTO <sub>24</sub>																			
Component CAP 1 Control				Component CAP 1																						
Component CAP 2 Control				Component CAP 2																						
Component CAP 3 Control				Component CAP 3																						
Component CAP 4 Control				Component CAP 4																						
ZMMU ATTR																										
ComponentNonce					R8		OOOTO		COHLAT																	
MGR-UUID																										
R9	Caution Thermal Limit	Upper Thermal Limit	Thermal Attributes	Serial Number																						
Z-UUID																										
C-UUID																										
FRU-UUID																										

Figure 8-6: Core Structure Format

Table 8-3: Core Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>Version (Vers)</b>	4	0x1	M	RO	Structure Version
<b>C-Status</b>	64	-	M	-	See <i>Core C-Status</i>
<b>C-Control</b>	64	-	M	-	See <i>Core C-Control</i>
<b>Base C-Class</b>	16	-	M	RO	Base Component Class—Indicates the class of component. Component class is used during power-on / reset initialization to determine how to initially handle a component without comprehending any component UUIDs. See <i>Appendix C Component Class Encodings</i> .
<b>Max Interface</b>	12	-	M	RO	Maximum Number of Supported Interfaces
<b>R-BIST</b>	4	-	O	RW	Component-specific failure results from BIST invocation. Software should set R-BIST = 0x0 prior to invoking BIST.
<b>RDLAT</b>	16	-	M	RO	RDLAT is the maximum time a Responder takes to receive, validate, and execute a Read request packet (Max_Packet_Payload read size), and transmit a Read Response packet (first bit in until last bit transmitted).  If a Responder supports multiple addressable resources, then RDLAT represents the worst-case time when accessing any addressable resource.  If a Responder supports coherency operations, then RDLAT represents the worst-case time required to return a coherent read operation—a coherent read may require additional coherency-specific operations to be performed prior to returning the corresponding response packet.  To calculate latency:  Read Latency = RDLAT * Core Latency Scale
<b>WRLAT</b>	16	-	M	RO	The maximum time a Responder takes to receive, validate, and execute a Write request packet (Max_Packet_Payload size), and transmit a <i>Standalone Acknowledgment</i> (first bit in until last bit transmitted).  If a Responder supports multiple addressable resources, then WRLAT represents the worst-

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					<p>case time when accessing any addressable resource.</p> <p>To calculate latency:</p> $\text{Write Latency} = \text{WRLAT} * \text{Core Latency Scale}$
<b>Max RSP Supported Requests</b>	20	-	M	RO	<p>The maximum number request packets a Responder is capable of executing.</p> <p>If only a Requester, then this field is Reserved.</p>
<b>Max REQ Supported Requests</b>	20	-	M	RO	<p>The maximum number of request packets a Requester or a Requester-Responder is capable of generating.</p> <p>If only a Responder, then this field is Reserved.</p>
<b>C-Op-Clock</b>	24	-	M	RO	<p>Used to calculate the present component operating clock as follows:</p> $\text{Operating Clock} = \text{C-Op-Clock} * 1024 \text{ Hz}$ <p>The component shall update this field at least once every 10 seconds while in C-Up or C-LP or upon any change to C-Up or C-LP.</p>
<b>Max Data</b>	64	-	M	RO	<p>This field indicates the size of the all addressable resources provisioned for Data Space access.</p> <p>Max Data shall be an integer 4096-byte multiple.</p> <p>If a component supports the <i>Component Media Structure</i>, then this structure provides additional details on the specific resources.</p> <p>Max Data may be set to zero in a component that only supports Control Space.</p>
<b>Max CTL</b>	52	-	M	RO	<p>This field indicates the maximum zero-based address required to access the component's Control Space. Control Space may be sparsely populated.</p> <p>Max CTL shall be an integer 4096-byte multiple.</p>
<b>Max RNR</b>	3	-	MC	RO	The maximum <i>Responder Not Ready (RNR)</i> NAK Time Interval Encoding that a Responder

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
C-Up Time					may return as specified in <i>RNR NAK Time Interval Encoding</i> . If the Responder does not support RNR NAK, then this field shall be Reserved.
C-DLP Time	20	-	M	RO	Bits 3:0—Exit latency from C-LP to C-Up Bits 7:4—Exit latency from C-DLP to C-Up Bits 11:8—Entry latency from C-Up to C-LP Bits 15:12—Entry latency from C-Up to C-DLP Bits 18:16—Minimum component idle time before automatically transitioning to C-LP Each 4-bit value is an encoded value selected from <i>Component Maximum Entry-Exit Latency and Idle Time Encodings</i> .
C-LP Time	12	-	M	RO	Bits 3:0—Exit latency from C-DLP to C-Up Bits 7:4—Entry Latency from C-LP to C-DLP Bits 11:8—Entry latency from C-Up to C-DLP Each 4-bit value is an encoded value selected from <i>Component Maximum Entry-Exit Latency and Idle Time Encodings</i> .
LPWR	16	-	M	RO	Bits 3:0—Exit latency from C-LP to C-Up Bits 7:4—Exit latency from C-LP to C-DLP Bits 11:8—Entry latency from C-Up to C-LP Bits 15:12—Minimum component idle time before automatically transitioning to C-DLP Each 4-bit value is an encoded value selected from <i>Component Maximum Entry-Exit Latency and Idle Time Encodings</i> .
NPWR	10	-	M	RO	LPWR is used to calculate the minimum power required for the component to correctly operate. $\text{Min\_Op\_Power} = (\text{LPWR} * \text{PS}) \text{ Watts.}$
	10	-	M	RO	NPWR is used to calculate the maximum power a component is capable of consuming under normal operating conditions. $\text{Max\_Normal\_Power} = (\text{NPWR} * \text{PS}) \text{ Watts.}$

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
HPWR	10	-	M	RO	<p>HPWR is used to calculate the maximum power a component is capable of consuming when configured to operate beyond normal operating conditions, e.g., an overclocked component.</p> $\text{Max_Load_Power} = (\text{HPWR} * \text{PWRS}) \text{ Watts}$
EPWR	10	-	O	RO	<p>EPWR is used to calculate the maximum power a component is capable of consuming when Emergency Power Reduction is triggered.</p> $\text{Max_Emergency_Power} = (\text{EPWR} * \text{PWRS}) \text{ Watts}$
APWR	10	-	O	RO	<p>APWR is used to calculate the maximum power a component is capable of consuming when operating on auxiliary power.</p> $\text{Aux_Power} = (\text{APWR} * \text{APWRS}) \text{ Watts}$
Control Structure PTR n	-	-	M	RO	<p>Pointers to additional Control Space structures.</p> <p>Pointer fields may be sparsely populated.</p> <p>A Null pointer indicates there is no corresponding structure accessible through this field.</p>
Core LPD BDF Table PTR	32	-	O	RO	If supported, then this field points to the provisioned <i>Core LPD BDF Table</i> .
OpCode Set Structure PTR	32	-	M	RO	This field shall point to <i>OpCode Set Structure 0x0</i> .
Component C-Access PTR	32	-	O	RO	If supported, then this points to the first <i>Component C-Access Structure</i> , else shall be Null.
Component Destination Table PTR	32	-	O	RO	If supported, then this points to the <i>Component Destination Table Structure</i> , else shall be Null.
Interface 0 PTR	32	-	M	RO	This field shall point to the <i>Interface Structure</i> associated with Interface 0.
Component Extension PTR	32	-	O	RO	If supported, then this field points to the provisioned <i>Component Extension Structure</i> .

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>Component Error and Signal Event PTR</b>	32	-	O	RO	If supported, then this field points to the provisioned <i>Component Error and Signal Event Structure</i> .
<b>MUTO</b>	16	-	M	RW	<p>If a Requester supports a single component-wide retransmission timer mechanism (see <i>Core Structure Component CAP 1 Retransmission Timer Support</i>), then this field determines the Minimum Unicast Retransmission Timer</p> <p>Timer = MUTO * <i>Core Structure Component CAP 1 Timer Unit</i> (-0%, + 25%).</p> <p>If unsupported, then this field shall be Reserved.</p> <p>If modified, then the new value shall take effect on the next timer expiration.</p> <p>See <i>Unicast Reliable Delivery</i>.</p>
<b>CRPTO</b>	16	-	M	RO	Indicates the maximum Control Request Processing Time (CRPT) in $\mu$ s.
<b>CCTO</b>	16	-	O	HwInit	<p>See <i>Configuration Completion Timer and Release</i>.</p> <p>Timer = CCTO ms (-0%, + 100%)</p>
<b>FAILTO</b>	16	-	O	RO	<p>Indicates the minimum time in <math>\mu</math>s that a Responder shall wait before initiating resource recovery actions. This timer may be associated with long-lived resources.</p> <p>Timer = FAILTO <math>\mu</math>s (-0%, + 100%)</p> <p>If FAILTO = 0, then the Responder does not support a failsafe timer. At least one Requester shall explicitly recover resources by transmitting the appropriate clean up request.</p>
<b>SVETO</b>	16	-	O	RW	<p>Small Variable Execution Timer—used to detect if a small non-deterministic operation failed to complete (see <i>Non-Deterministic Request Execution</i>).</p> <p>Timer = SVETO * <i>Core Structure Component CAP 1 Timer Unit</i> (-5%, +25%).</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
LVETO					If modified, then the new value shall take effect on the next timer expiration.  What constitutes small vs. large is outside of this specification's scope.
	32	-	O	RW	Large Variable Execution Timer—used to detect if a large non-deterministic operation failed to complete (see <i>Non-Deterministic Request Execution</i> ).  Timer = LVETO (-0%, + 50%)  If modified, then the new value shall take effect on the next timer expiration.  What constitutes small vs. large is outside of this specification's scope.
UNREQ					
	16	-	O	RW	Minimum time that a Requester shall wait before transmitting a new unreliable unicast request packet that has the same packet identification protocol field values, i.e., the [SCID   GSCID, Tag, DCID   GDCID], as a previously transmitted unreliable request packet.  Timer = UNREQ * <i>Core Structure Component CAP 1 Timer Unit</i> (-0%, + 50%)  If a Requester receives a <i>Standalone Acknowledgment</i> (Reason = Resource Release) that corresponds to a previously transmitted unreliable request packet, then the Requester may immediately reuse the packet identification protocol field values.  This timer should be larger than the largest UNRSP value configured in all communicating peer Responders.  If modified, then the new value shall take effect on the next timer expiration.
UNRSP					
	16	-	O	RW	Minimum time a Responder shall wait before releasing all resources and tracking logic associated with a multi-packet operation, e.g., a Write MSG. This timer shall be reset upon receipt of a new request packet that is part of a multi-packet operation.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
UERT					<p>Timer = UNRSP <math>\mu</math>s (-0%, + 50%)</p> <p>If modified, then the new value shall take effect on the next timer expiration.</p> <p>This timer shall be applicable only to unicast and unreliable multicast multi-packet operations.</p>
	16	-	O	RW	<p>Minimum Unsolicited Event Retransmission Timer. See <i>Unsolicited Event (UE)</i> Packet.</p> <p>Timer = UERT ms (-0%, + 100%)</p> <p>If modified, then the new value shall take effect on the next timer expiration.</p>
NIRT	16	-	O	RW	<p>Minimum NIR Resource Release Timer—See <i>Non-idempotent Request (NIR)</i></p> <p>Timer = NIRT * <i>Core Structure Component CAP 1 Timer Unit</i> (-0%, +25%).</p> <p>If modified, then the new value shall take effect on the next timer expiration.</p>
ENIRT	16	-	O	RW	<p>Minimum Extended NIR Resource Release Timer. See <i>Non-idempotent Request (NIR)</i></p> <p>Timer = ENIRT <math>\mu</math>s (-0%, + 50%)</p> <p>If modified, then the new value shall take effect on the next timer expiration.</p>
FPST	16	-	MC	RW	<p>Minimum FPS timer that is used by Responders that support <i>Responder Not Ready (RNR) NAK</i>. See <i>Forward Progress Screen (FPS)</i></p> <p>Timer = FPST * <i>Core Structure Component CAP 1 Timer Unit</i>.</p> <p>If modified, then the new value shall take effect on the next timer expiration.</p> <p>This field is used for all FPSs other than a PCO FPS. See PCO FPST field description.</p> <p>If a Responder never transmits an RNR NAK, then this field shall be Reserved.</p>
PCO FPST	16	-	MC	RW	Minimum FPS timer that is used by Responders for a PCO FPS. See <i>Protocol Deadlock Avoidance for PCO</i> .

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					<p>Timer = PCO FPST * <i>Core Structure Component CAP 1 Timer Unit</i> (-0%, + 25%)</p> <p>If modified, then the new value shall take effect on the next timer expiration.</p> <p>If a Responder does not support PCO, then this field shall be Reserved.</p>
LL Request Deadline	10	-	MC	RW	<p>If a Requester supports any explicit OpClass, then this field shall be configured with the Deadline value (see <i>Congestion and Packet Deadline</i>) to use in request packets transmitted within a low-latency domain.</p> <p>Software shall be responsible for setting the LL Request Deadline to a non-zero value.</p> <p>If a Requester does not support explicit OpClass packets, then this field shall be Reserved.</p> <p>If modified, then the new value shall take effect on the next scheduled new end-to-end packet.</p>
NLL Request Deadline	10	-	MC	RW	<p>If a Requester supports any explicit OpClass, then this field shall be configured with the Deadline value (see <i>Congestion and Packet Deadline</i>) to use in request packets transmitted within a non-low-latency domain.</p> <p>Software shall be responsible for setting the NLL Request Deadline to a non-zero value.</p> <p>If a Requester does not support explicit OpClass packets, then this field shall be Reserved.</p> <p>If modified, then the new value shall take effect on the next scheduled new end-to-end packet.</p>
Deadline Tick	12	-	MC	RWS	<p>If the component supports any explicit OpClass, then this field shall be configured with the number of ns per decrement of the Deadline sub-field (see <i>Congestion and Packet Deadline</i>) within an explicit OpClass packet's Congestion field.</p> <p>Software shall be responsible for setting the Deadline Tick to a non-zero value. Software</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
LL Response Deadline					<p>shall be responsible for ensuring all communicating components</p> <p>If the component does not support explicit OpClass packets, then this field shall be Reserved.</p> <p>This field shall be modified only if the component is in a non-C-Up state. If modified while the component is in C-Up, then the component's behavior may be non-deterministic.</p>
NLL Response Deadline	10	-	MC	RW	<p>If a Responder supports any explicit OpClass, then this field shall be configured with the Deadline value (see <i>Congestion and Packet Deadline</i>) to use in response packets transmitted by this Responder to a peer Requester in a low-latency domain (see <i>Component PA (Peer Attribute) Structure</i>).</p> <p>Software shall be responsible for setting the Response Deadline to a non-zero value.</p> <p>If a Responder does not support explicit OpClass packets, then this field shall be Reserved.</p> <p>If modified, then the new value shall take effect on the next scheduled new end-to-end packet.</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Responder Deadline	10	-	MC	RW	<p>If a Responder supports any explicit OpClass, then this field shall be configured with the Deadline value (see <i>Congestion and Packet Deadline</i>) used to determine whether a Responder should generate a response packet.</p> <p>Software shall be responsible for setting the Response Deadline to a non-zero value.</p> <p>If a Responder does not support explicit OpClass packets, then this field shall be Reserved.</p> <p>If modified, then the new value shall take effect on the next transmitted end-to-end packet.</p>
ATSTO	16	-	MC	RW	<p>If the component supports <i>Address Translation and Page Services</i>, then this field indicates the maximum time to generate a <i>Translation Invalidate Response</i> packet or a <i>PRG Response Notification</i> packet.</p> <p>ATSTO shall be greater than 2.</p> <p>Timer = ATSTO ms (+0%)</p> <p>If modified, then the new value shall take effect on the next timer expiration.</p>
PFMSID	16	-	O	RW	<p>If the component supports <i>Global Component Identifiers</i>, then this field may be captured or configured with the SID of the subnet where the primary fabric management component is located.</p> <p>This SID may be configured through <i>In-band Management</i> or <i>Out-of-band Management</i>.</p> <p>If configured and none of the component's SIDs are equal to the PFMSID, then the component shall set GC = 1b in all packets exchanged with the primary fabric management component.</p> <p>If unsupported or if the PFMSID is not configured, then the component shall set GC = 0b in all packets exchanged with the primary fabric management component.</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
PMCID					If a component does not support <i>In-band Management</i> or multi-subnet communications, then this field shall be Reserved.
	12	-	M	RW	If the component is managed by a Primary Manager, then this field is configured with CID of the component that hosts the Primary Manager's.  The PMCID may be captured or configured.  If component management control is transitioned to a different component, then updating PMCID should be the last step taken.  Components that do not support <i>In-band Management</i> shall treat this field as Reserved.
PWR MGR CID	12	-	O	RW	If component power management is performed by a manager other than the component's Primary Manager, Primary Fabric Manager, or Secondary Fabric Manager, then this field is configured with the CID of the component that hosts the power manager.  <i>Core Structure Component CAP 1 Control</i> Power Manager Enable indicates if this field has been configured.  If a component does not support <i>In-band Management</i> , then this field shall be Reserved.
	16	-	O	RW	If component power management is performed by a manager other than the component's Primary Manager, Primary Fabric Manager, or Secondary Fabric Manager, then this field is configured with the SID of the component that hosts the power manager.  <i>Core Structure Component CAP 1 Control</i> Power Manager Enable indicates if this field has been configured.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
PFMCID					<p>If a component does not support <i>In-band Management</i> or multi-subnet communications, then this field shall be Reserved.</p>
	12	-	O	RW	<p>This field is configured with the CID of the Primary Fabric Manager for this component. Fabric management occurs through a fabric switch.</p> <p>The PFMCID may be captured or configured. If component management control is transitioned to a different component, then updating PFMCID should be the last step taken.</p> <p>If the component that hosts the Primary Fabric Manager is located in a different subnet from this component, then the PFMSID field shall be combined with the PFMCID to identify the Primary Fabric Management component.</p> <p>If a component does not support <i>In-band Management</i>, then this field shall be Reserved.</p>
SFMCID	12	-	O	RW	<p>This field is configured with the CID of the Secondary Fabric Manager for this component.</p> <p>If supported, then the SFMCID shall be configured.</p> <p>If the component that hosts the Secondary Fabric Manager is located in a different subnet from this component, then the SFMSID field shall be combined with the SFMCID to identify the Secondary Fabric Management component.</p> <p>If a component does not support <i>In-band Management</i>, then this field shall be Reserved.</p>
SFMSID	16	-	O	RW	If the component supports <i>Global Component Identifiers</i> , then this field shall be configured with the SID of the subnet where the

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					<p>secondary fabric management component is located.</p> <p>If configured and none of the component's SIDs are equal to the SFMSID, then the component shall set GC = 1b in all packets exchanged with the secondary fabric management component.</p> <p>If unsupported or if the SFMSID is not configured, then the component shall set GC = 0b in all packets exchanged with the secondary fabric management component.</p> <p>If a component does not support <i>In-band Management</i> or multi-subnet communications, then this field shall be Reserved.</p>
SCV n	2	-	-	RW	<p>Determines if the corresponding CID and SID fields are configured.</p> <p>If CID 0 and SID 0 (if applicable) are captured, then hardware shall automatically set SCV 0 = 0x1.</p> <p>If a given CID / SID n is unsupported, then SCV n shall be hardwired to 0x0.</p> <p>0x0—Not configured 0x1—CID n Configured 0x2—CID n and SID n Configured 0x3—Reserved</p>
CID n	12	-	O	RW	<p>Component Local Identifier associated with this component</p> <p>If a component supports explicit OpClasses, then it shall support CID 0. CID 0 shall be updated only while the component is in C-CFG. If updated in any other state, then the component's behavior may be non-deterministic.</p> <p>CIDs 1-3 may be configured at any time. If updated subsequent to being enabled, then the component's behavior may be non-deterministic.</p> <p>Unsupported CID fields shall be Reserved.</p> <p>CIDs shall be configured by management.</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>SID n</b>					If a component does not support explicit OpClasses, then these fields shall be Reserved.
	16	-	O	RW	<p>Subnet Identifier associated with this component</p> <p>If a component supports multi-subnet communications, then it shall support SID 0. SID 0 shall be updated only while the component is in C-CFG. If updated in any other state, then the component's behavior may be non-deterministic.</p> <p>IDs 1-3 may be configured at any time. If updated subsequent to being enabled, then the component's behavior may be non-deterministic.</p> <p>If a component does not support multi-subnet communications or explicit OpClasses, then these fields shall be Reserved.</p> <p>SIDs shall be configured by management.</p>
<b>Max Requests</b>	20	-	M	RW	<p>The maximum number of request packets a Requester or Requester-Responder may generate.</p> <p>This field shall be set to a value that is less than or equal to the Max REQ Supported Requests field.</p> <p>If only a Responder, then this field shall be Reserved.</p>
<b>Min Tag Cycle</b>	12	-	O	RO	<p>If a Requester supports unreliable request packets, then this field indicates the minimum number of unique Tag values the Requester will guarantee to use before reusing a Tag in a new unreliable request packet.</p> <p>If a Responder or the Requester does not support unreliable request packets, then this field shall be Reserved.</p>
<b>MI-LSB</b>	7	-	O	RW	Memory Interleave LSB—If a Responder that supports <i>Memory Interleave</i> , then this field is used to configure the least significant bit of the bit range to zero out when packing

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
MI-MSB					<p>regions into contiguous Responder-local addresses.</p> <p>If not a Responder, then this field shall be Reserved.</p> <p>Software shall set <i>Core Structure Component CAP 2 Control</i> Enable Responder Memory Interleave = 0b prior to updating this field.</p> <p><b>Developer Note:</b> If a Responder contains addressable persistent media, then software is responsible for remembering the value of this field and memory interleave group composition.</p>
	7	-	O	RW	<p>Memory Interleave MSB—If a Responder that supports <i>Memory Interleave</i>, then this field is used to configure the most significant bit of the bit range to zero out when packing regions into contiguous Responder-local addresses.</p> <p>If not a Responder, then this field shall be Reserved.</p> <p>Software shall set <i>Core Structure Component CAP 2 Control</i> Enable Responder Memory Interleave = 0b prior to updating this field.</p> <p><b>Developer Note:</b> If a Responder contains addressable persistent media, then software is responsible for remembering the value of this field and memory interleave group composition.</p>
TO <sub>K</sub>	16	-	O	RW	<p>If supported, then each field determines the minimum unicast request retransmission timeout value (see <i>Unicast Reliable Delivery</i>) used for any request packets transmitted by this component on VC<sub>K</sub> within a low-latency domain.</p> <p>Timer = TO<sub>K</sub> * <i>Core Structure Component CAP 1 Timer Unit</i> (-0%, + 10%).</p> <p>If modified, then the new value shall take effect on the next timer expiration.</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>LTO<sub>K</sub></b>					<p>This value shall be applicable to all component interfaces that have VC<sub>K</sub> enabled.</p> <p>If TO<sub>K</sub> == 0x0, then the Requester shall not start the corresponding unicast retransmission timer on VC<sub>K</sub>.</p> <p>TO<sub>K</sub> shall be hardwired to 0x0 for any unsupported VC<sub>K</sub>.</p> <p>TO support is indicated by the <i>Core Structure Component CAP 1 Retransmission Timer Support</i> bits. If unsupported, then this field shall be Reserved.</p> <p>If a Responder, then these fields shall be Reserved.</p>
	16	-	O	RW	<p>If supported, then each field determines the minimum unicast request retransmission time timeout value (see <i>Unicast Reliable Delivery</i>) used for any request packets transmitted by this component on VC<sub>K</sub> within any non-low-latency domain.</p> <p>Timer = LTO<sub>K</sub> * <i>Core Structure Component CAP 1 Timer Unit</i> (-0%, + 25%).</p> <p>If modified, then the new value shall take effect on the next timer expiration.</p> <p>This value shall be applicable to all component interfaces that have VC<sub>K</sub> enabled.</p> <p>If LTO<sub>K</sub> is supported, then the component shall support the <i>Component PA (Peer Attribute) Structure</i>.</p> <p>If LTO<sub>K</sub> == 0x0, then the Requester shall not start the corresponding unicast retransmission timer on VC<sub>K</sub>.</p> <p>LTO<sub>K</sub> shall be hardwired to 0x0 for any unsupported VC<sub>K</sub>.</p> <p>LTO support is indicated by the <i>Core Structure Component CAP 1 Retransmission Timer Support</i> bits. If unsupported, then this field shall be Reserved.</p> <p>If a Responder, then these fields shall be Reserved.</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Component CAP n	64	-	M	RO	See: <i>Core Structure Component CAP 1</i> <i>Core Structure Component CAP 2</i> <i>Core Structure Component CAP 3</i> <i>Core Structure Component CAP 4</i>
Component CAP n Control	64	-	M	RW	See: <i>Core Structure Component CAP 1 Control</i> <i>Core Structure Component CAP 2 Control</i> <i>Core Structure Component CAP 3 Control</i> <i>Core Structure Component CAP 4 Control</i>
COHLAT	16	-	M	RO	COHLAT is the maximum time a Responder takes to validate and execute a coherency operation that could involve multiple components (Home Agents / Caching Agents), e.g., a <i>Read Exclusive</i> or <i>Invalidate and Writeback</i> request packets. COHLAT covers the time from receipt of the coherency operation until the last bit of a response packet is transmitted.  To calculate latency:  $\text{Coherency Latency} = \text{COHLAT} * \text{Core Latency Scale}$
OOOTO	16	-	O	RW	Out-of-order Timer—maximum time that a component will wait for an out-of-order packet to arrive before taking a specific action. The OOOTO is used only for select purposes, e.g., bounding <i>Sequence Number ART</i> exposure.  $\text{Timer} = \text{OOOTO} * \text{Core Structure Component CAP 1 Timer Unit} (-10\%, +0\%).$  If modified, then the new value shall take effect on the next timer expiration.
ComponentNonce	64	-	O	WO	A 64-bit nonce that identifies a component on an interface until an <i>Interface Reset</i> (any type)—See <i>Mitigating In Situ Insertion</i> for additional details.
MGR-UUID	128	-	M	RW	If a component supports <i>In-band Management</i> , then a manager should store a UUID in the MGR-UUID field in components that it manages / owns.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Serial Number					<p>If a Control OpClass request packet contains a MGR-UUID field, then the Responder shall validate this field with the packet's MGR-UUID field, and if not equal, treat this as an AE.</p> <p>A management component (Requester) may use copy this UUID to the MGR-UUID field in applicable in Control OpClass request packets. A management component may support multiple MGR-UUID values (e.g., one per management domain). If supported, then a management component may use a <i>Vendor-Defined Structure</i> or methods outside of the scope of this specification to manage and select a MGR-UUID value to use in a given Control OpClass request packet. See <i>Core Structure Component CAP 1 Control MGR-UUID Enable</i>.</p>
	64	-	O	HwInit	<p>Shall be set to 0x0 or a vendor-assigned number.</p> <p>The Serial Number value may be stored in a separate storage device (e.g., an EEPROM), and then copied into this field using an implementation-specific method during component initialization.</p>
Thermal Attributes	16	-	O	HwInit	<p>If non-zero, then this field describes the thermal attributes of a mechanical form factor. The specific layout of this field is form factor-specific.</p> <p>The Thermal Attributes value may be stored in a separate storage device (e.g., an EEPROM), and then copied into this field using an implementation-specific method during component initialization.</p>
Upper Thermal Limit	10	-	M	RO	Maximum temperature in Celsius that the component remains operational when potentially unsafe thermal environmental conditions occur. The Upper Thermal Limit is used to determine when component-specific performance throttling is invoked.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>Caution Thermal Limit</b>	10	-	M	RO	Maximum temperature in Celsius that the component remains operational when potentially unsafe thermal environmental conditions occur. The Caution Thermal Limit is used to determine when component-specific performance throttling is invoked.
<b>ZMMU ATTR</b>	128	-	O	RO	<p>If a component supports a ZMMU, then this field indicates the attributes.</p> <p>This field applies to a <i>Transparent Router (TR)</i> ZMMU (substitute TR Requester PTE wherever Requester PTE is used).</p>
		Bits 8:0			REQ_PTE_SIZE—If non-zero, then this indicates that the component supports a Requester ZMMU and the size of each <i>Requester PTE</i> in bits.
		Bits 17:9			RSP_PTE_SIZE—If non-zero, then this indicates that the component supports a Responder ZMMU and the size of each <i>Responder PTE</i> in bits.
		Bits 22:18			PTE_GD_SZ—If non-zero, then this indicates that the supported <i>Requester PTE</i> contains a Global Destination field and its size in bits. The maximum size shall be 16 bits.
		Bits 27:23			PTE_RKMGR_SID_SZ—If non-zero, then this indicates that the supported <i>Responder PTE</i> contains a RK_MGRG_SID field and its size in bits. The maximum size shall be 16 bits.
		Bits 32:28			If non-zero, then RKMGR_CID Support shall be set to 1b.
		Bits 37:33			SEC_KEY_ID_SZ—If non-zero, then this indicates that the supported <i>Requester PTE</i> and / or <i>Responder PTE</i> contains a SKE bit and a Security_Key_ID field. SEC_Key_ID_SZ indicates the size of the Security ID field in bits.
					PASID_SZ—If non-zero, then this indicates that the supported <i>Requester PTE</i> and / or <i>Responder PTE</i> contains a PS bit and a PASID field. PASID_SZ indicates the size of the

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					PASID field in bits. The maximum size shall be 20 bits.
		Bit 38			PME Support—Indicates if the PME bit is present in the supported <i>Requester PTE</i> . 0b—Unsupported 1b—Supported
		Bit 39			WPE Support—Indicates if the WPE bit is present in the supported <i>Requester PTE</i> . 0b—Unsupported 1b—Supported
		Bit 40			RKE Support—Indicates if the RKE bit and R-Key field are present in the supported <i>Requester PTE</i> and / or <i>Responder PTE</i> . 0b—Unsupported 1b—Supported
		Bit 41			NSE Support—Indicates if the NSE bit is present in the supported <i>Requester PTE</i> . 0b—Unsupported 1b—Supported
		Bit 42			LPE Support—Indicates if the LPE bit is present in the supported <i>Requester PTE</i> . 0b—Unsupported 1b—Supported
		Bit 43			CE Support—Indicates if the CE bit is present in the supported <i>Requester PTE</i> and / or <i>Responder PTE</i> . 0b—Unsupported 1b—Supported
		Bit 44			PS Support—Indicates if the PS bit is present in the supported <i>Requester PTE</i> and / or <i>Responder PTE</i> . 0b—Unsupported 1b—Supported
		Bit 45			CCE Support—Indicates if the CCE bit is present in the supported <i>Requester PTE</i> and / or <i>Responder PTE</i> . 0b—Unsupported

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					1b—Supported
		Bit 46			PA Support—Indicates if the PA bit is present in the supported <i>Responder PTE</i> . 0b—Unsupported 1b—Supported
		Bit 47			RKMGR_CID Support—Indicates if the Responder ZMMU supports R-Key updates using <i>R-Key Update</i> request packets and if the supported <i>Responder PTE</i> contains a RKMGR-CID field. 0b—Unsupported 1b—Supported
		Bit 48			Write Mode 0 Support—Indicates if the <i>Requester PTE</i> supports Write Mode == 0x0 functionality.
		Bit 49			Write Mode 1 Support—Indicates if the <i>Requester PTE</i> supports Write Mode == 0x1 functionality. 0b—Unsupported 1b—Supported
		Bit 50			Write Mode 2 Support—Indicates if the <i>Requester PTE</i> supports Write Mode == 0x2 functionality. 0b—Unsupported 1b—Supported
		Bit 51			Write Mode 3 Support—Indicates if the <i>Requester PTE</i> supports Write Mode == 0x3 functionality. 0b—Unsupported 1b—Supported
		Bit 52			Write Mode 4 Support—Indicates if the <i>Requester PTE</i> supports Write Mode == 0x4 functionality. 0b—Unsupported 1b—Supported
		Bit 53			Write Mode 5 Support—Indicates if the <i>Requester PTE</i> supports Write Mode == 0x5 functionality.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Z-UUID	128				0b—Unsupported 1b—Supported
					Write Mode 6 Support—Indicates if the <i>Requester PTE</i> supports Write Mode == 0x6 functionality. 0b—Unsupported 1b—Supported
					Write Mode 7 Support—Indicates if the <i>Requester PTE</i> supports Write Mode == 0x7 functionality. 0b—Unsupported 1b—Supported
					LPD Responder Bypass Support—Indicates if the Responder ZMMU supports LPD bypass. 0b—Unsupported 1b—Supported
					PEC Supports—Indicates if the <i>Requester PTE</i> supports the Processor Exception Control bit. 0b—Unsupported 1b—Supported
					RsvdZ
					Traffic Class (TC) Size—Indicates the size of the TC field. 0x0—4 bits 0x1—Reserved
					ZMMU Page Size Support—Bit mask where each Bit <sub>k</sub> indicates if the corresponding page size is supported. Each Bit <sub>k</sub> represents a power-of-2 page size starting at 4 KiB, e.g., Bit <sub>80</sub> indicates 4 KiB page support, Bit <sub>81</sub> indicates 8 KiB page support, and Bit <sub>127</sub> indicates corresponds to 2 <sup>48</sup> byte page support. 0b—Unsupported Page Size 1b—Supported Page Size
					Gen-Z UUID uniquely identifies a component as a Gen-Z component.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>C-UUID</b>					<p>Given the variety of technologies that use UxC to access a component, this UUID enables software to definitively determine if a component is a Gen-Z component.</p> <p>All Gen-Z components shall use the Gen-Z consortium-assigned UUID in this field.</p> <p>The Gen-Z UUID is:</p> <p>0x4813ea5f074e4be2a355a354145c9927</p> <p>This UUID field may be stored in a separate storage device (e.g., an EEPROM).</p>
	128	-	O	HwInit	<p>C-UUID uniquely identifies the component.</p> <p>This is a vendor-assigned UUID. A vendor shall assign the same C-UUID to all instances of a specific component implementation. The C-UUID may be used to associate requisite software with the component.</p> <p>If a component supports multiple services that require distinct software, then it shall support the <i>Service UUID Structure</i>, and use each individual Service-UUID to associate requisite software with the service.</p> <p>The C-UUID value may be stored in a separate storage device (e.g., an EEPROM), and then copied into this field using an implementation-specific method during component initialization.</p>
<b>FRU-UUID</b>	128	-	O	HwInit	<p>FRU-UUID uniquely identifies the set of components that shall be treated as a single field replaceable unit (FRU).</p> <p>This is a vendor-assigned UUID. A vendor shall assign the same FRU-UUID to all instances of a specific FRU implementation.</p> <p>If FRU-UUID == 0x0, then the component shall be treated as the FRU.</p> <p>The FRU-UUID value may be stored in a separate storage device (e.g., an EEPROM), and then copied into this field using an implementation-specific method during component initialization.</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
R0	32	-	-	-	Reserved
	9	-	-	-	RsvdP
	16	-	-	-	RsvdP
	14	-	-	-	RsvdP
	2	-	-	-	RsvdP
	4	-	-	-	RsvdP
	4	-	-	-	RsvdP
	2	-	-	-	RsvdP
	32	-	-	-	RsvdP
	28	-	-	-	Reserved

### 8.14.1. Core C-Status

Table 8-4: Core C-Status

Bit Location	Access	Description
Bits 2:0	RO	C-State—Present component State 0x0—C-Down. Non-operational, uninitialized 0x1—C-CFG. Configuration in progress 0x2—C-Up. Operational 0x3—C-LP. Operational, power-saving state 0x4—C-DLP. Operational, deep power-saving state 0x5-0x7—Reserved
Bit 3	RW1C	<i>Unsolicited Event (UE) Packet</i> Status If 1b, then an <i>Unsolicited Event (UE) Packet</i> exchange is in progress.
Bit 4	RW1C	Non-Fatal Internal Error Detected If 1b, a component-internal non-fatal error has been detected. The component remains operational. The <i>Component Error and Signal Event Structure</i> specifies which errors are mapped to this C-Status bit. This mapping is used independent of <i>Component Error and Signal Event Structure</i> support.
Bit 5	RW1CS	Fatal Internal Error Detected If 1b, a component-internal fatal error has been detected. The component is unable to operate as expected.

Bit Location	Access	Description
Bit 6		<p>The <i>Component Error and Signal Event Structure</i> specifies which errors are mapped to this C-Status bit. This mapping is used independent of <i>Component Error and Signal Event Structure</i> support.</p>
Bit 6	RW1CS	<p>Non-Transient Protocol Error Detected</p> <p>If 1b, a non-transient packet protocol error has been detected.</p> <p>The <i>Component Error and Signal Event Structure</i> specifies which errors are mapped to this C-Status bit. This mapping is used independent of <i>Component Error and Signal Event Structure</i> support.</p>
Bit 7	RW1C	<p>BIST Failure Detected</p> <p>If 1b, then BIST invocation failed. Component-specific results returned in R-BIST field.</p>
Bit 9:8	RO	<p>Component Thermal Status. While in C-Up, status is updated upon changes in thermal operating conditions. This field shall be updated with the present thermal state at least once every second.</p> <p>0x0—Nominal thermal conditions 0x1—Caution Thermal Limit 0x2—Exceeded Upper Thermal Limit 0x3—Thermal shutdown triggered</p>
Bit 10	RW1CS	<p><i>Component Containment</i> Detected</p> <p>If 1b, then <i>Component Containment</i> was triggered, and the component has transitioned to C-Down.</p>
Bit 11	RW1C	<p>Emergency Power Reduction Detected</p> <p>If 1b, then component Emergency Power Reduction was triggered either by the component or by management.</p>
Bit 12	RW1CS	<p>Power-off Transition Completed</p> <p>If 1b, then the component has completed the component power-off steps described in <i>Component Power Management</i> and is ready for power removal and / or component removal.</p>
Bit 13	RW1C	Component Thermal Throttled—the component has throttled performance due to a thermal event
Bit 14	RW1C	Component Thermal Throttled Restoration—the component has restored throttled performance
Bit 15	RW1CS	Cannot Execute <i>Persistent Flush</i> —If a Responder supports addressable persistent resources and the Responder is unable to execute a request packet that modifies the addressable persistent resources (validation or execution error), then the Responder shall set Cannot Execute Persistent Flush = 1b. If Cannot Execute Persistent Flush == 1b upon receipt of a <i>Persistent Flush</i>

Bit Location	Access	Description
Bit 16		request packet, then the Responder shall return a <i>Standalone Acknowledgment</i> (Reason = Persistent Flush Update Failure). Software shall clear Cannot Execute Persistent Flush prior to retransmitting a <i>Persistent Flush</i> request packet.
17	RO	HwInit Valid—indicates that state of all HwInit fields in all Gen-Z control structures and component-specific structures. 0b—In-progress HwInit field initialization. 1b—All HwInit fields have been successfully initialized and may be accessed by software.
Bits 31:18	RW1CS	Refresh Component Configuration Completed If 1b, then indicates the component has completed updating all cached configuration state.
	RsvdZ	

### 8.14.2. Core C-Control

Unless explicitly stated otherwise by this specification, reads of this field shall return zero.

Table 8-5: Core C-Control

Bit Location	Access	Description
Bit 0	RW	Component Enable 0b—Disable Component. The component shall transition to C-Down. 1b—Enable Component. If the component has been successfully configured, then the component shall transition to C-Up. Has no impact if the component is in C-Up.
Bits 3:1	RW	Component Reset—initiate the corresponding component reset 0x0—No impact 0x1— <i>Full Component Reset</i> 0x2— <i>Warm Component Reset</i> 0x3— <i>Warm Non-Switch Component Reset</i> 0x4— <i>Content Component Reset</i> (if supported, else this shall have no impact) 0x5-0x7—Reserved
Bit 4	RW	Halt UERT ( <i>Unsolicited Event (UE) Packet</i> Retransmission Timer) 1b—Halts retransmission of the present <i>Unsolicited Event (UE) Packet</i> . If there are no pending Unsolicited Events to generate, then the C-Status Unsolicited Event Status field shall be set to 0b and the UERT shall be halted.

Bit Location	Access	Description
Bit 5		Reads of this bit shall return 0b.
	RW	<p>Transition C-Up</p> <p>1b—Transitions the component to C-Up from C-LP or C-DLP. Has no impact if not in C-LP or C-DLP.</p>
Bit 6		Reads of this bit shall return 0b.
	RW	<p>Transition C-LP</p> <p>1b—Transitions the component to C-LP from C-Up. Has no impact if not in C-Up.</p>
Bit 7		Reads of this bit shall return 0b.
	RW	<p>Transition C-DLP</p> <p>1b—Transitions the component to C-DLP from C-LP or C-Up. Has no impact if not in C-LP or C-Up.</p>
Bit 8		Reads of this bit shall return 0b.
	RW	<p>Trigger Emergency Power Reduction</p> <p>1b—Trigger Emergency Power Reduction</p>
Bit 9		Reads of this bit shall return 0b.
	RW	<p>Exit Emergency Power Reduction—enables the component to return to its prior configured Maximum Power Level.</p> <p>1b—Exit Emergency Power Reduction</p>
Bit 10		Reads of this bit shall return 0b.
	RW	<p>Transition Component Power-Off—See <i>Component Power Management</i></p> <p>1b—Instructs component to take all steps necessary for power removal / component removal.</p>
Bit 11		Reads of this bit shall return 0b.
	RW	<p>Upper Thermal Limit Performance Throttle—if the component exceeds the Upper Thermal Limit, then this bit enables the component to automatically invoke component-specific performance throttle to reduce thermal load. Upper Thermal Limit performance throttling shall take precedence over Caution Thermal Limit performance throttling.</p> <p>0b—Disable 1b—Enabled</p>
Bit 12		Reads of this bit shall return 0b.
	RW	<p>Caution Thermal Limit Performance Throttle—if the component exceeds the Caution Thermal Limit, then this bit enables the component to automatically invoke component-specific performance throttle to reduce thermal load.</p> <p>0b—Disable 1b—Enabled</p>

Bit Location	Access	Description
Bit 13	RW	LPD Responder ZMMU Bypass Control—If ZMMU ATTR LPD Responder Bypass Support == 1b, then this bit determines if the Responder ZMMU is bypassed when executing received request packets with LP == 1b. 0b—Do Not Bypass Responder ZMMU 1b—Bypass Responder ZMMU
Bit 14	RW	Refresh Component Configuration Control If a component does not cache configuration state ( <i>Core Structure Component CAP 1 Cached Component Control Space Structure Support == 0b</i> ), then, updates to Refresh Component Configuration shall be ignored. 1b—Refresh all cached configuration state. Reads of this bit shall return 0b.
Bits 63:15	RsvdP	

*Component Maximum Entry-Exit Latency and Idle Time Encodings* contains a list of encodings and associated time values. These values are used to communicate the maximum time a component takes to enter or exit a power sub-state, or the maximum time a component is idle before it automatically enters a lower power sub-state.

Table 8-6: Component Maximum Entry-Exit Latency and Idle Time Encodings

Encoding	Time	Encoding	Time
0x0	1 µs	0x8	5 ms
0x1	5 µs	0x9	10 ms
0x2	10 µs	0xA	50 ms
0x3	25 µs	0xB	100 ms
0x4	50 µs	0xC	500 ms
0x5	100 µs	0xD	1 s
0x6	500 µs	0xE	5 s
0x7	1 ms	0xF	10 s

### 5 8.14.3. Core Structure Component CAP 1

Table 8-7: Component CAP 1 Field

Bit Location	Access	Description
2:0	RO	Max_Packet_Payload—Maximum packet payload size supported. 0x0—256 Bytes 0x1-0x7—Reserved

Bit Location	Access	Description
3	RO	<p>No Snoop Support—Indicates if the component supports the No Snoop operation bit in applicable packets.</p> <p>0b—Unsupported 1b—Supported</p>
4	RO	<p>Content Component Reset Support (see <i>Component Reset</i>).</p> <p>0b—Unsupported 1b—Supported</p>
5	RO	<p>Built-in Self-Test (BIST) Support</p> <p>0b—Unsupported 1b—Supported</p>
6	RO	<p>Component Containment Support. Only applicable if the <i>Component Error and Signal Event Structure</i> is supported.</p> <p>0b—Unsupported 1b—Supported</p>
7	RO	<p>Address Field Interpretation</p> <p>0b—The Address field within packets communicates a zero-based address. 1b—The Address field within packets communicates a non-zero-based address.</p> <p>The Responder is responsible for translating the address into component-internal physical resource.</p> <p>If the component supports non-zero-based addressing, then the component shall support a vendor-defined structure to enable software to configure memory management.</p>
8	RO	<p>Next Header Support—Indicates if the component supports all explicit OpClass packets with the <i>Next Header Field</i>.</p> <p>0b—Unsupported 1b—Supported</p>
9	RO	<p>Next Header Control OpClass Support—Indicates if the component supports Control OpClass packets with the <i>Next Header Field</i>. If Next Header Support == 1b, then shall set Next Header Control OpClass Support = 1b.</p> <p>0b—Unsupported 1b—Supported</p>
10	RO	<p><i>Precision Time</i> Support</p> <p>0b—Unsupported 1b—Supported</p>
13:11	RO	<p>Addressable Resource Classification—Indicates if the component contains addressable resources, the semantics used to access the resources, or where to discover additional media-specific details.</p> <p>0x0—No addressable Data Space resources</p>

Bit Location	Access	Description
		0x1—Byte addressable Data Space memory media 0x2—Block addressable Data Space memory media 0x3— <i>Component Media Structure</i> specifies Data Space memory media details 0x4—Non-memory media Data Space resource (e.g., a I/O or message buffer resource) 0x5-0x7—Reserved
14	RO	Cached Component Control Space Structure Support  0b—Unless explicitly stated otherwise by this specification, component control structure state is not cached and updates to the control structures shall take effect prior to the transmission of a <i>Standalone Acknowledgment</i> that acknowledges a <i>Control Space In-Band Operations</i> packet or indicating the success of an <i>Out-of-band Management</i> update.  1b—Component Control Space structure state is cached, i.e., the component requires management to set <i>Core C-Control Refresh Component Configuration Control</i> = 1b to update all cached state.
15	RO	<i>In-band Management</i> Support—Indicates if a component supports <i>In-band Management</i> .  0b—Unsupported 1b—Supported
16	RO	<i>Out-of-band Management</i> Support—Indicates if a component supports <i>Out-of-band Management</i>  0b—Unsupported 1b—Supported
17	RO	Primary Manager Support—Indicates if the component may support a Primary Manager.  0b—Unsupported 1b—Supported
18	RO	Fabric Manager Support—Indicates if the component may support a Primary Fabric Manager or a Secondary Fabric Manager.  0b—Unsupported 1b—Supported
19	RO	Power Manager Supports—Indicates if the component supports a manager other than the Primary Manager, the Primary Fabric Manager, or the Secondary Manager to manage component power.  0b—Unsupported 1b—Supported
20	RO	Automatic C-State Support—A component may automatically change C-State if idle for an extended period of time.  0b—Unsupported 1b—Supported

Bit Location	Access	Description
21	RO	<p>Vendor-defined Power Management Support—A component may support vendor-defined power management.</p> <p>0b—Unsupported 1b—Supported</p>
22	RO	<p>Emergency Power Reduction Support—A component may support Emergency Power Reduction. See <i>Component Power Management</i>.</p> <p>0b—Unsupported 1b—Supported</p>
23	RO	<p>Configuration Post Emergency Power Reduction—if a component supports Emergency Power Reduction, and if the EPWR level is insufficient for the component to maintain its configuration, volatile media, etc., then management, service, and / or component-specific software may be required to re-configure / re-initialize the component once the component exits Emergency Power Reduction.</p> <p>0b—No software is required 1b—Software is required</p>
24	RO	<p>Emergency Power Reduction Packet Relay Support—Determines if the component continues to support packet relay once Emergency Power Reduction is triggered. If unsupported, a component shall silently discard all end-to-end packets until the component exits Emergency Power Reduction.</p> <p>0b—Unsupported 1b—Supported</p>
25	RO	<p>C-State Power Control Support—if the component supports the <i>C-State Power Control</i> packet, then this field indicates if the component supports only transition notification or notification and transition requests.</p> <p>0b—Notification-only 1b—Notification and Transition Requests</p>
26	RO	<p>Power Disable Support—Indicates if the component supports the Power Disable signal (see <i>Component Power Management</i>).</p> <p>If Power Disable Auto Enable is Enabled, then the Power Disable Support shall be Supported.</p> <p>0b—Unsupported 1b—Supported</p>
29:27	RO	<p>Power Scale (PWRS) is used to calculate the various component maximum non-auxiliary power consumption values.</p> <p>0x0—1.0 0x1—0.1 0x2—0.01 0x3—0.001 0x4—0.0001 0x5—0.00001</p>

Bit Location	Access	Description
32:30		0x6—0.000001 0x7—0.0000001
	RO	Auxiliary Power Scale (APWRS) is used to calculate the maximum power consumption a component is capable of consuming when operating on auxiliary power.  0x0—1.0 0x1—0.1 0x2—0.01 0x3—0.001 0x4—0.0001 0x5—0.00001 0x6—0.000001 0x7—0.0000001
33	RO	MCTP over Gen-Z Support—Indicates if the component supports the MCTP protocol transported using the Gen-Z protocol. If supported, then the component shall support at least one Vendor-defined OpClass that is used to transport MCTP. See <i>MCTP</i> .  0b—Unsupported 1b—Supported
35:34	RO	Core Latency Scale—Indicates the latency scale used to calculate Read, Write, and Coherency latencies (See RDLAT, WRLAT, and COHLAT fields)  0x0— $\mu$ s 0x1—ns 0x2—ps 0x3—Reserved
36	RO	Multi-subnet Support—Indicates if the component supports multi-subnet communications  0b—Unsupported 1b—Supported
38:37	RO	Maximum Number CID—Indicates the maximum number of CIDs that may be configured. If the component supports multi-subnet communications, then this also indicates the maximum number of SIDs that may be configured.  0x0—CID 0, SID 0 0x1—CID 0-CID 1, SID 0-SID 1 0x2—CID 0-CID 2, SID 0-SID 2 0x3—CID 0-CID 3, SID 0-SID 3
39	RO	NIRT Support—Indicates if the component supports a failsafe NIR Timer (see <i>Non-idempotent Request (NIR)</i> )  0b—Unsupported 1b—Supported

Bit Location	Access	Description
40	RO	ENIRT Support—Indicates if the component supports an extended failsafe NIR Timer (see <i>Non-idempotent Request (NIR)</i> ) 0b—Unsupported 1b—Supported
41	RO	Meta Read-Write Support—Indicates if the component supports inclusion of the Meta field in applicable read response and write request packet. 0b—Unsupported 1b—Supported
43:42	RO	Retransmission Timer Support—Indicates supported retransmission timers 0x0—Unsupported (Responder-only Component) 0x1—MUTO-only Support 0x2—TO <sub>k</sub> -only Support 0x3—TO <sub>k</sub> and LTO <sub>k</sub> -only Support
47:44	RO	Timer Unit—Indicates the supported unit of time associated with the following timers: MUTO, TO, LTO, NIRT, FPST, SVETO, and LVETO. 0x0—1 ns 0x1—10 ns 0x2—100 ns 0x3—1 $\mu$ s 0x4—10 $\mu$ s 0x5—100 $\mu$ s 0x6—1 ms 0x7—10 ms 0x8—100 ms 0x9—1 s 0xA-0xF—Reserved
63:48	-	RsvdP

#### 8.14.4. Core Structure Component CAP 1 Control

Table 8-8: Component CAP 1 Control Field

Bit Location	Access	Description
2:0	RW	Max_Packet_Payload—Maximum packet payload size enabled by the component on any interface. 0x0—256 Bytes (Default) 0x1-0x7—Reserved
3	RW	No Snoop Control 0b—Disabled: A component shall set No Snoop = 0b in applicable packets.

Bit Location	Access	Description
4		1b—Enabled: A component may set No Snoop = 1b in applicable packets.
4	RW	<p>Built-in Self-Test (BIST) Control</p> <p>0b—Disabled</p> <p>1b—Invoke BIST. While BIST is executing the component shall silently discard all end-to-end packets. BIST results shall be indicated via the R-BIST field within the <i>Core Structure</i>.</p> <p>BIST may be executed while the component is any C-State in order to support both on-line and offline diagnostics. Once invoked, the component remains in BIST mode until this field is updated to disable BIST.</p> <p>Reads of this field shall return 0b.</p>
5	RW	<p>Manager Type—Determines if the component is managed by a Primary Manager or by a Primary or Secondary Fabric Manager.</p> <p>By default, the component shall be managed by a Primary Manager. A Primary Manager may transition management responsibility to a fabric manager by configuring the <i>Core Structure</i> [PFMCID, PFMSID] and, if applicable, the [SFMCID, SFMSID] fields, and then setting Manager Type = 1b.</p> <p>0b—Primary Manager</p> <p>1b—Fabric Manager</p> <p>If a component supports only one type of management, then this bit may be hardwired to that type.</p> <p>If a component supports only <i>Out-of-band Management</i>, then this field shall be hardwired to 0b.</p>
6	RWS	<p>Primary Manager Role—Determines if the Primary Manager is co-located on this component.</p> <p>0b—Not Co-located</p> <p>1b—Co-located</p> <p>To be a candidate to host this manager, the <i>Core Structure Component CAP 1 Primary Manager Support</i> shall be set to 1b. Further, if <i>Core Structure Component CAP 1 In-band Management Support</i> == 1b, then the component shall support the <i>Component Event Structure</i>.</p> <p>If Primary Manager Role == 1b, then:</p> <ul style="list-style-type: none"> <li>▪ This component shall act as the Primary Manager component for components that are accessible by this component through <i>In-band Management</i> or <i>Out-of-band Management</i>.</li> <li>▪ The <i>Core Structure</i> CID 0 shall be copied to the PMCID field prior to setting Primary Manager Role = 1b.</li> <li>▪ Once Primary Manager Role == 1b, then the PMCID field should not be modified; any updates to the PMCID field shall be ignored.</li> </ul>

Bit Location	Access	Description
7		<p>This bit shall be applicable only to components that support <i>In-band Management</i>.</p> <p>If the component supports only <i>Out-of-band Management</i>, then this bit shall be hardwired to 0b.</p> <p>If a component will always host the Primary Manager, then the component shall hardwire this bit to 1b.</p> <p>If the Primary Manager role is transitioned to another Primary Manager or to a fabric manager (primary or secondary) on a different component, then this bit shall be set to 0b. Further, this bit is sticky to permit the new manager to initiate a <i>Warm Component Reset</i> without the component reasserting itself as the Primary Manager.</p>
8	<p>RW</p> <p>Primary Fabric Manager Role—Determines if the Primary Fabric Manager is co-located on this component.</p> <p>0b—Not Co-located 1b—Co-located</p> <p>To be a candidate to host this manager, shall set <i>Core Structure Component CAP 1 Fabric Manager Support</i> = 1b. Further, the component shall support the <i>Component Event Structure</i>.</p> <p>If Primary Fabric Manager Role == 1b, then:</p> <ul style="list-style-type: none"> <li>▪ This component shall act as the Primary Fabric Manager component for fabric-managed components that are accessible by this component through <i>In-band Management</i>.</li> <li>▪ Prior to setting Primary Fabric Manager Role to 1b, the <i>Core Structure CID 0</i> shall be copied to the PFMCID field and, if SID 0 is configured, then it shall be copied to the PFMSID field, and set <i>Core Structure Component CAP 2 Control PFMSID Valid</i> = 1b..</li> <li>▪ Once Primary Fabric Manager Role == 1b, then the PFMCID and PFMSID fields should not be modified; any updates to the PFMCID or PFMSID fields shall be ignored.</li> <li>▪ To ensure fabric management resiliency, a component should not act as both the Primary and Secondary fabric manager.</li> </ul> <p>This field shall be applicable only to components that support <i>In-band Management</i>.</p> <p>If the component supports only <i>Out-of-band Management</i> or may be managed only by a Primary Manager, then this field shall be hardwired to 0b.</p> <p>RW</p> <p>Secondary Fabric Manager Role—Determines if the Secondary Fabric Manager is co-located on this component.</p> <p>0b—Not Co-located 1b—Co-located</p>	

Bit Location	Access	Description
10:9		<p>To be a candidate to host this manager, shall set <i>Core Structure Component CAP 1 Fabric Manager Support</i> = 1b. Further, the component shall support the <i>Component Event Structure</i>.</p> <p>If Secondary Fabric Manager Role == 1b, then:</p> <ul style="list-style-type: none"> <li>▪ This component shall act as the Secondary Fabric Manager component for fabric-managed components that are accessible by this component through <i>In-band Management</i>.</li> <li>▪ Prior to setting Secondary Fabric Manager Role = 1b, the <i>Core Structure</i> CID 0 shall be copied to the SFMCID field and, if SID 0 is configured, then it shall be copied to the SFMSID field, and set <i>Core Structure Component CAP 2 Control SFMSID Valid</i> = 1b.</li> <li>▪ Once Secondary Fabric Manager Role = 1b, then the SFMCID and SFMSID fields should not be modified; any updates to the SFMCID or SFMSID fields shall be ignored.</li> <li>▪ To ensure fabric management resiliency, a component should not act as both the Primary and Secondary Fabric Manager.</li> </ul> <p>This field shall be applicable only to components that support <i>In-band Management</i>.</p> <p>If the component supports only <i>Out-of-band Management</i> or may be managed only by a Primary Manager, then this field shall be hardwired to 0b.</p>
10:9	RW	<p>Power Manager Enable—Determines if a component other than the Primary Manager, the Primary Fabric Manager, or the Secondary Fabric Manager is enabled to perform component power management and the configuration state of the PWG MGR CID and PWR MGR SID fields.</p> <p>0x0—Disabled, PWG MGR CID Not Configured, PWG MGR SID Not Configured      0x1—Enabled, PWR MGR CID Configured, PWR MGR SID Not Configured      0x2—Enabled, PWR MGR CID Configured, PWR MGR SID Configured      0x3—Reserved</p>
11	RW	<p>Primary Manager Transition Enable—if Primary Manager Role == 1b, then this determines if Gen-Z management may be transitioned to a new Primary Manager component or to a fabric manager. Once management has transitioned to the new manager, Primary Manager Role shall be set to 0b.</p> <p>0b—Disabled      1b—Enabled</p>
12	RW	<p>Fabric Manager Transition Enable—if Primary Fabric Manager Role == 1b or Secondary Fabric Manager Role == 1b, then this determines if Gen-Z fabric management may be transitioned to a new Primary or Secondary Fabric Manager component. Once management has transitioned to the new fabric manager, Primary Fabric Manager Role and Secondary Fabric Manager Role shall be set to 0b.</p> <p>0b—Disabled</p>

Bit Location	Access	Description
13	RW	<p>1b—Enabled</p> <p>Next Header Enabled—This field enables the inclusion of the <i>Next Header Field</i> in all explicit OpClass packets at the component level.</p> <p>If the component does not support the <i>Component PA (Peer Attribute) Structure</i>, then when enabled, all explicit OpClass packets shall contain the Next Header field.</p> <p>If the component supports the <i>Component PA (Peer Attribute) Structure</i>, then that structure may be used to selectively enable the inclusion of the Next Header field on a per destination basis. Selective enablement via the <i>Component PA (Peer Attribute) Structure</i> settings shall apply only if the Next Header Enabled == 1b.</p> <p>Shall be enabled only if the component supports explicit OpClass packets and the Next Header field.</p> <p>See the <i>Component Security Structure</i> for additional considerations.</p> <p>0b—Disabled</p> <p>1b—Enabled</p>
14	RW	<p>Next Header Control OpClass Enabled—This field enables the inclusion of the <i>Explicit OpClass Next Header Field</i> in Control OpClass packets at the component level.</p> <p>If the component does not support the <i>Component PA (Peer Attribute) Structure</i>, then when enabled, all Control OpClass packets shall set NH = 1b and shall contain the Next Header field.</p> <p>If the component supports the <i>Component PA (Peer Attribute) Structure</i>, then that structure may be used to selectively enable the inclusion of the Next Header field on Control OpClass packets on a per destination basis. Selective enablement via the <i>Component PA (Peer Attribute) Structure</i> settings shall apply only if Next Header Control OpClass Enabled == 1b.</p> <p>Shall be enabled only if the component supports Control OpClass packets and the <i>Explicit OpClass Next Header Field</i>.</p> <p>See the <i>Component Security Structure</i> for additional considerations.</p> <p>0b—Disabled</p> <p>1b—Enabled</p>
15	RW	<p>Next Header Precision Time Enabled—This field enables the inclusion of a <i>Precision Time</i> value within the <i>Explicit OpClass Next Header Field</i> in explicit OpClass packets at the component level.</p> <p>If the component does not support the <i>Component PA (Peer Attribute) Structure</i>, then when enabled, all explicit OpClass packets shall contain the Next Header field with a <i>Precision Time</i> value.</p> <p>If the component supports the <i>Component PA (Peer Attribute) Structure</i>, then that structure may be used to selectively enable the inclusion of a <i>Precision</i></p>

Bit Location	Access	Description
		<p><i>Time</i> value in the Next Header field on a per destination basis. Selective enablement via the <i>Component PA (Peer Attribute) Structure</i> settings shall apply only if Next Header Precision Time Enabled == 1b.</p> <p>Shall be enabled only if the component supports explicit OpClass packets, the Next Header field, and <i>Precision Time</i>, and Next Header Enabled = 1b.</p> <p>See the <i>Component Security Structure</i> for additional considerations.</p> <p>0b—Disabled 1b—Enabled</p>
16	RW	<p><i>In-band Management</i> Enabled—Determines if <i>In-band Management</i> is used.</p> <p>If a component supports only <i>In-band Management</i>, then this field shall be hardwired to 0b.</p> <p>0b—Enabled (Default) 1b—Disabled</p>
17	RW	<p><i>Out-of-band Management</i> Enabled—Determines if <i>Out-of-band Management</i> is used.</p> <p>If a component supports only <i>Out-of-band Management</i>, then this field shall be hardwired to 0b.</p> <p>0b—Enabled (Default) 1b—Disabled</p>
18	RW	<p>Automatic C-State Enable</p> <p>If the component supports only management-directed C-State changes, then this field shall be hardwired to 0b.</p> <p>0b—Disabled 1b—Enabled</p>
19	RW	<p>Vendor-defined Power Management Support—if the component does not support vendor-defined power management, then this field should be hardwired to 0b. If the vendor-defined power management cannot be disabled, then this field shall be hardwired to 1b.</p> <p>0b—Disabled 1b—Enabled</p>
22:20	RW	<p>Max-Power Control—Maximum power level the component may draw at any time based on the power levels supplied in the <i>Core Structure</i> or the <i>Component Mechanical Structure</i>.</p> <p>0x0—LPWR 0x1—NPWR 0x2—HPWR 0x3—<i>Component Mechanical Structure</i> Max_Mech_Power_LVL 0x4-0x7—Reserved</p>
23	RW	Emergency Power Reduction Enabled

Bit Location	Access	Description
24		<p>If both Emergency Power Reduction and Power Disable are supported and enabled and if they share a mechanical form factor or component package pin, then when the shared out-of-band signal is asserted, the component's behavior may be non-deterministic. If Emergency Power Reduction is triggered by other means or does not share the same pin, then the component shall operate as specified in <i>Component Power Management</i>.</p> <p>0b—Disabled 1b—Enabled</p>
	RW	<p>Notify Requester on Power Event—if the component supports the <i>Component Power Management</i> request and the OpCode is enabled, then this field determines if the component shall notify each peer Requester when it transitions between C-Up and C-LP or C-DLP.</p> <p>0b—Disabled 1b—Enabled</p>
25	RW	<p>Notify Management on Power Event—this field determines if the component shall notify management when:</p> <ul style="list-style-type: none"> <li>• The component transitions between C-Up and C-LP or C-DLP</li> <li>• Emergency Power Reduction is triggered (assumes there is sufficient power for the component to generate the notification)</li> <li>• Any component interface detects a peer component transitioning to or from C-DLP</li> <li>• The component sets C-Status Power-off Transition Completed = 1b.</li> </ul> <p>If a component supports <i>In-band Management</i> and the <i>Component Error and Signal Event Structure</i>, then the component shall notify management as configured.</p> <p>If <i>Core Structure Component CAP 1 Control Out-of-band Management Enable</i> == 1b, then the component shall notify management using a component-specific interrupt.</p> <p>0b—Disabled 1b—Enabled</p>
	RW	<p>C-State Power Control Enable—if the component supports the <i>C-State Power Control packet</i>, then this field determines if the component shall generate and execute only notifications, or may generate and execute notifications and transition requests.</p> <p>0b—Notification-only 1b—Notification and Transition Requests</p>
29:27	RW	<p>Lowest Automatic C-State Level—if <i>Core Structure Component CAP 1 Control Automatic C-State Enable</i> == 1b, then this field indicates the lowest C-State power level that the component may automatically enter.</p> <p>If the component supports only management-directed C-State changes, then this field shall be hardwired to 0x0.</p>

Bit Location	Access	Description
30		<p>0x0—C-Up 0x1—C-LP 0x2—C-DLP 0x3-07—Reserved</p>
	RW	<p>Initiate All Statistics Snapshot—For every provisioned <i>Component Statistics Structure / Interface Statistics Structure / vendor-defined statistics structure / etc.</i> that supports snapshot functionality, perform a snapshot of each structure as specified for that structure.</p> <p>Software shall wait until all individual snapshots have been completed before initiating a new snapshot. Statistics in each individual statistics structure shall be reset to zero as soon as completion of the corresponding snapshot and no later than the completion of all individual snapshots.</p> <p>1b—Perform Component-Wide Snapshot</p> <p>Reads of this bit shall return 0b.</p>
31	RW	<p>Initiate All Interface Statistics Snapshot—For every provisioned <i>Interface Statistics Structure</i> that supports snapshot functionality (see Interface Statistics Snapshot Support), perform a snapshot.</p> <p>Software shall wait until all individual snapshots have been completed before initiating a new snapshot. Statistics in each individual statistics structure shall be reset to zero as soon as completion of the corresponding snapshot and no later than the completion of all individual snapshots.</p> <p>1b—Perform All Interface Snapshot</p> <p>Reads of this bit shall return 0b.</p>
32	RW	<p>Power Disable Enable—if Power Disable is supported and enabled, then the Power Disable out-of-band signal may be used to disable power to the component.</p> <p>If both Emergency Power Reduction and Power Disable are supported and enabled and if they share mechanical form factor or component package pin, then when the shared out-of-band signal is asserted, the component's behavior may be non-deterministic.</p> <p>If Power Disable Auto Enable is Enabled, then this field shall be hardwired to Enabled.</p> <p>0b—Disabled 1b—Enabled</p>
33	RW	<p>Power Disable Auto Enable—if Power Disable is supported, then this field determines if it shall be automatically enabled. This field shall be hardwired to Disabled or Enabled.</p> <p>0b—Disabled 1b—Enabled</p>

Bit Location	Access	Description
34	RW	MCTP Enable—Used to enable MCTP over Gen-Z communications. 0b—Disabled 1b—Enabled
35	RW	Meta Read-Write Header Enabled—This field enables the inclusion of the Meta field in applicable read response and write request packets.  If the component supports explicit OpClass packets but does not support the <i>Component PA (Peer Attribute) Structure</i> , then when enabled, all applicable explicit OpClass packets may include the Meta field.  If the component supports the <i>Component PA (Peer Attribute) Structure</i> , then that structure may be used to selectively enable the inclusion of the Meta field on a per destination basis. Selective enablement via the <i>Component PA (Peer Attribute) Structure</i> settings shall apply only if the Meta Read-Write Header Enabled = 1b.  0b—Disabled 1b—Enabled
37:36	RW	MGR-UUID Enable—if a management component (Requester), then this field determines the UUID field to copy to MGR-UUID field in applicable Control OpClass request packets.  0x0—Copy 0x0 to the packet’s MGR-UUID field 0x1—Copy the <i>Core Structure</i> MGR-UUID to the packet’s MGR-UUID field 0x2—Select UUID using a vendor-defined method to copy to the packet’s MGR-UUID field 0x3—Reserved
55:38	-	RsvdP
56	RWS	Software-defined Management Bit 0—The purpose and use of this bit is outside of this specification’s scope. If a component supports sticky bits, then it shall maintain the setting of this bit across reset events as long as it has sufficient power.
57	RWS	Software-defined Management Bit 1—The purpose and use of this bit is outside of this specification’s scope. If a component supports sticky bits, then it shall maintain the setting of this bit across reset events as long as it has sufficient power.
58	RWS	Software-defined Management Bit 2—The purpose and use of this bit is outside of this specification’s scope. If a component supports sticky bits, then it shall maintain the setting of this bit across reset events as long as it has sufficient power.
59	RWS	Software-defined Management Bit 3—The purpose and use of this bit is outside of this specification’s scope. If a component supports sticky bits, then

Bit Location	Access	Description
60		it shall maintain the setting of this bit across reset events as long as it has sufficient power.
60	RWS	Software-defined Management Bit 4—The purpose and use of this bit is outside of this specification’s scope. If a component supports sticky bits, then it shall maintain the setting of this bit across reset events as long as it has sufficient power.
61	RWS	Software-defined Management Bit 5—The purpose and use of this bit is outside of this specification’s scope. If a component supports sticky bits, then it shall maintain the setting of this bit across reset events as long as it has sufficient power.
62	RWS	Software-defined Management Bit 6—The purpose and use of this bit is outside of this specification’s scope. If a component supports sticky bits, then it shall maintain the setting of this bit across reset events as long as it has sufficient power.
63	RWS	Software-defined Management Bit 7—The purpose and use of this bit is outside of this specification’s scope. If a component supports sticky bits, then it shall maintain the setting of this bit across reset events as long as it has sufficient power.

#### 8.14.5. Core Structure Component CAP 2

Table 8-9: Component CAP 2 Field

Bit Location	Access	Description
1:0	RO	R-Key Support—Indicates if the component supports <i>Region Key (R-Key)</i> . 0x0—Unsupported 0x1—Supported as a Requester 0x2—Supported as a Responder 0x3—Supported as a Requester-Responder
2	RO	Responder Memory Interleave Support—if a Responder, then this indicates if it supports <i>Memory Interleave</i> . If supported, then the Responder shall pack the interleaved regions into contiguous Responder-local addresses by AND-masking the higher-order address bits (MSB:LSB) to zero. 0b—Unsupported 1b—Supported
3	RO	Requester Memory Interleave Support—if a Requester, then this indicates if it supports <i>Memory Interleave</i> . 0b—Unsupported 1b—Supported

4	RO	SOD Support—Indicates if the component supports <i>Strong Ordering Domain (SOD)</i> 0b—Unsupported 1b—Supported
5	RO	Write MSG Embedded Read Support—Indicates if the component supports the <i>Write MSG</i> embedded read capability (Write MSG role is as indicated in the component's <i>OpCode Set Structure</i> ). 0b—Unsupported 1b—Supported
9:6	RO	Poison Granularity Support—If the component supports Write Poison, then this field indicates the supported poison granularity. 0x0—Unsupported 0x1—16 bytes 0x2—32 bytes 0x3—64 bytes 0x4—128 bytes 0x5—256 bytes 0x6—512 bytes 0x7—1024 bytes 0x8—2048 bytes 0x9—4096 bytes 0xA-0xF—Reserved
10	RO	Host LPD Communication Support—Indicates if a host component supports <i>Logical PCI Device (LPD)</i> communications. Non-host components shall hardwired this bit to 0b. 0b—Unsupported 1b—Supported
11	RO	Host LPH Communication Support—Indicates if a host component supports <i>Logical PCI Hierarchy (LPH)</i> communications. Non-host components shall hardwired this bit to 0b. 0b—Unsupported 1b—Supported
14:12	RO	Performance Marker Support—Indicates if the component supports gathering packet-specific data if an explicit OpClass packet contains the PM bit and PM == 1b. If non-zero, indicates the type of information that it is capable of gathering. See <i>Component PM Structure</i> for log record details. 0x0—Unsupported 0x1—Generates Performance Log Record Type 0 0x2—Generates Performance Log Record Type 1 0x3-0x7—Reserved
19:15	RO	Max_Perf_Records—If Performance Marker Support == 1b, then this field indicates the maximum number of Performance Log Records (see <i>Component PM Structure</i> ) of any supported type that may be recorded. If Performance Marker Support == 0b, then this field shall be Reserved.

		Max Perf Log Records = $2^{\text{Max\_Perf\_Records}}$
22:20	RO	<p>Meta Read-Write Support—If the component supports Meta Read and Meta Writes, then this field indicates which UUID is used to identify the contents and semantics of the Meta field within request packets.</p> <p>0x0—Unsupported</p> <p>0x1—<i>Core Structure</i> C-UUID</p> <p>0x2—<i>Vendor-Defined Structure</i> with UUID located using the Vendor-defined PTR within the <i>Component Media Structure</i></p> <p>0x3—A Service-UUID within the <i>Service UUID Structure</i></p> <p>0x4-0x7—Reserved</p>
23	RO	<p><i>LPD Field</i> Type 3 Support—If the component supports <i>Logical PCI Device (LPD)</i> communications, then this bit indicates if the component supports <i>LPD Field</i> Type 3 protocol field format. If the component does not support <i>Logical PCI Device (LPD)</i> communications, then this bit shall be hardwired to 0b.</p> <p>0b—Unsupported</p> <p>1b—Supported</p>
63:24	-	RsvdP

#### 8.14.6. Core Structure Component CAP 2 Control

Table 8-10: Component CAP 2 Control Field

Bit Location	Access	Description
0	RW	<p>R-Key Enable—If a Requester and R-Key Enable == Enabled, then a Requester shall set the RK bit and include the R-Key field in applicable request packets based on the <i>Requester PTE</i> R-Key Enable setting, else the Requester shall set RK = 0b in applicable request packets.</p> <p>If a Responder and R-Key Enable == Enabled and the destination page's <i>Responder PTE</i> R-Key Enabled == Enabled, then the Responder shall perform R-Key validation, else the Responder shall not perform R-Key validation.</p> <p>0b—Disabled</p> <p>1b—Enabled</p>
1	RW	<p>PMCID Valid—Determines if the PMCID field has been configured with a valid CID</p> <p>0b—Invalid</p> <p>1b—Valid</p>
2	RW	<p>PFCMCID Valid—Determines if the PFCMCID field has been configured with a valid CID</p>

Bit Location	Access	Description
		If the component cannot be fabric-managed, then the PFMCID Valid shall be hardwired to 0b. 0b—Invalid 1b—Valid
3	RW	SFMCID Valid—Determines if the SFMCID field has been configured with a valid CID  If the component cannot be fabric-managed, then the SFMCID Valid shall be hardwired to 0b. 0b—Invalid 1b—Valid
4	RW	PFMSID Valid—Determines if the PFMSID field has been configured with a valid SID  If the component cannot be fabric-managed, then the PFMSID Valid shall be hardwired to 0b. 0b—Invalid 1b—Valid
5	RW	SFMSID Valid—Determines if the SFMSID field has been configured with a valid SID  If the component cannot be fabric-managed, then the SFMSID Valid shall be hardwired to 0b. 0b—Invalid 1b—Valid
6	RW	Enable Responder Memory Interleave—If a Responder that supports <i>Memory Interleave</i> , then this field is used to enable or disable memory interleave. 0b—Disabled 1b—Enabled
7	RW	Performance Log Record Enable—If enabled and supported, then the component shall generate a performance log record upon receipt of an explicit OpClass packet with PM == 1b. 0b—Unsupported / Disabled 1b—Enabled
8	RW	Clear Performance Marker Log—If supported, writing to this field shall clear all performance log records. 1b—Clear Performance Log Reads of this bit shall return 0b
9	RW	Logical PCI Communications Enable—If <i>Core Structure Component CAP 2 Host LPD Communications Support</i> == 1b and / or <i>Core Structure Component CAP 2</i>

Bit Location	Access	Description
10		Host LPH Communications Support == 1b, then this determines if LPD and LPH communications are enabled. If both are unsupported, then this bit shall be hardwired to 0b. 0b—Disabled 1b—Enabled
	RO	LPD Field Type 3 Support—if <i>Core Structure Component CAP 2 LPD Field Type 3 Support</i> == 1b, then this bit determines if the component is enabled to use this format. If <i>Core Structure Component CAP 2 LPD Field Type 3 Support</i> == 0b, then this bit shall be hardwired to 0b. 0b—Unsupported 1b—Supported
	-	RsvdP

#### 8.14.7. Core Structure Component CAP 3

Table 8-11: Component CAP 3 Field

Bit Location	Access	Description
1:0	RO	Lightweight Notification Support. See <i>Lightweight Notification (LN)</i> 0x0—Unsupported 0x1—LN Requester 0x2—LN Responder 0x3—LN Requester and LN Responder
	RO	LNREQ Registrations Supported—Exponent used to calculate the maximum number of registered addresses an LNREQ supports. See <i>Lightweight Notification (LN)</i> . The maximum is calculated as $2^{\text{LNREQ Registrations Supported}}$ If unsupported, then this field shall be Reserved.
	RO	LNRSP 32-byte LN Block Size Support If Lightweight Notification is unsupported, then this field shall be Reserved. 0b—Unsupported 1b—Supported
8	RO	LNRSP 64-byte LN Block Size Support If Lightweight Notification is unsupported, then this field shall be Reserved. 0b—Unsupported 1b—Supported
	RO	LNRSP 128-byte LN Block Size Support If Lightweight Notification is unsupported, then this field shall be Reserved.

		0b—Unsupported 1b—Supported
10	RO	LNRSP 256-byte LN Block Size Support  If Lightweight Notification is unsupported, then this field shall be Reserved.  0b—Unsupported 1b—Supported
14:11	RO	Max Supported Home Agents—if the component supports Gen-Z cache coherency protocol operations, then this field indicates the number of supported Home Agents.
18:15	RO	Max Supported Caching Agents—if the component supports Gen-Z cache coherency protocol operations, then this field indicates the number of supported Caching Agents.
63:19	-	Reserved

#### 8.14.8. Core Structure Component CAP 3 Control

Table 8-12: Component CAP 3 Control Field

Bit Location	Access	Description
0	RW	LN Enable—Enable <i>Lightweight Notification (LN)</i>  0b—Disable 1b—Enable
3:1	RW	LNRSP LN Block Size—if non-zero, then this field indicates a fixed-sized, component-wide LN Block.  If <i>Lightweight Notification (LN)</i> is supported and this field is zero, then each page shall indicate its supported LN Block size using the LN protocol.  If <i>Lightweight Notification</i> is unsupported, then this field shall be Reserved.  0x0—Unsupported / Per page LN Block Size 0x1—32-byte LN Block Size 0x2—64-byte LN Block Size 0x3—128-byte LN Block Size 0x4—256-byte LN Block Size 0x5-0x7—Reserved
4	RW	Home Agent Enable—if <i>Core Structure Component CAP 3 Max Supported Home Agents</i> is non-zero, then this bit is used to enable Gen-Z Home Agent operation. This bit applies to all supported Gen-Z Home Agents.  0b—Disable 1b—Enable
5	RW	Caching Agent Enable—if <i>Core Structure Component CAP 3 Max Supported Caching Agents</i> is non-zero, then this bit is used to enable Gen-Z Caching Agent operation. This bit applies to all supported Gen-Z Caching Agents.

Bit Location	Access	Description
63:6	-	0b—Disable 1b—Enable
63:6	-	Reserved

### 8.14.9. Core Structure Component CAP 4

Table 8-13: Component CAP 4 Field

Bit Location	Access	Description
63:0	-	Reserved

### 8.14.10. Core Structure Component CAP 4 Control

Table 8-14: Component CAP 4 Control Field

Bit Location	Access	Description
63:0	-	Reserved

### 8.14.11. Core LPD BDF Table

If the component supports *Logical PCI Device (LPD)* and the *LPD Field Type 0* protocol field format, then it shall support the Core LPD BDF Table. If supported, then:

- The Core LPD BDF Table shall use the *Core LPD BDF Table* Format.
- Each Core LPD BDF Table entry represents a 16-bit BDF value.
- Packets that contain the *LPD Field Type 0* protocol field shall copy the Core LPD BDF Table index that corresponds to the *Logical PCI Device (LPD)* 16-bit BDF.
- The invalid Core LPD BDF Table entry shall be 0xFFFF; all other values indicate a valid table entry. If the *LPD Field Type 0* protocol field is used, then software shall configure the corresponding Core LPD BDF Table entry prior to enabling LPD communications (see *Component LPD Structure Control 1 LPD Communications Enable*).

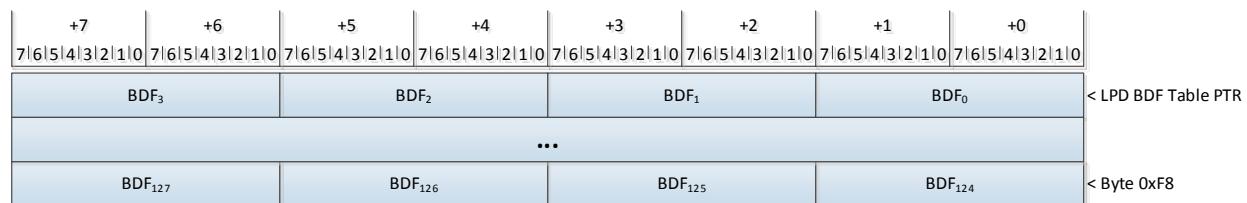


Figure 8-7: Core LPD BDF Table Format

## 8.15. OpCode Set Structure

The following are the features and requirements associated with the OpCode Set Structure:

- If a component acts as a Requester or a Responder, then it shall support the OpCode Set structure.
- The OpCode Set structure shall use the *OpCode Set Structure Format*.
- If the component supports multiple OpCode Sets (e.g., one per major protocol version or one per application service), then:
  - Additional OpCode Sets are located via the Next OpCode Set PTR located in this structure.
  - The maximum number of additional OpCode Set structures a component may support is 8. Each additional OpCode Set structure shall have a unique Set ID.
  - The component may enable multiple OpCode Sets. If only single OpCode Set structure is enabled, then it applies to all communications with all destination components. If multiple OpCode Set structures are enabled, then a component shall use the *Component PA (Peer Attribute) Structure* to associate an OpCode Set structure with a peer component.
- For each OpClass, each OpCode set is represented as two fields—one field representing the supported OpCodes and one field representing enabled OpCodes, i.e., the subset of the supported OpCodes allowed in a given solution.
  - Each field is composed of a set of 2-bit sub-fields.
  - Each sub-field within a given field corresponds to an OpCode, i.e., the encoded OpCode value as listed in *End-to-end Packet Operation Codes* is identified by the corresponding sub-field. For example, the encoded OpCode value 0x0 corresponds to sub-field 0 (located at byte 0, bit 0 of the field); the encoded OpCode value 0x1 corresponds to sub-field 1 (located contiguous to sub-field 0); and so forth.
  - Each sub-field within a Supported OpCode Set field shall be interpreted as follows:
    - 0x0 indicates the corresponding OpCode is unsupported. Unsupported and Reserved OpCodes shall be hardwired to 0x0 in the supported and enabled fields.
    - 0x1 indicates the corresponding OpCode is supported and the component's role shall be as a Requester.
      - If the OpCode is a request operation, then the Requester supports generating the corresponding request packet.
      - If the OpCode is a response operation, then the Requester supports receiving and executing the corresponding response packet.
    - 0x2 indicates the corresponding OpCode is supported and the component's role shall be as a Responder.
      - If the OpCode is a request operation, then the Responder supports receiving and executing the corresponding request packet.
      - If the OpCode is a response operation, then the Responder supports generating the corresponding response packet.
    - 0x3 indicates the corresponding OpCode is supported and the component's role shall be as a Requester-Responder.
      - If the OpCode is a request operation, then the Requester-Responder supports generating as well as receiving and executing the corresponding request packet.

- If the OpCode is a response operation, then the Requester-Responder supports generating as well as receiving and executing the corresponding response packet.
- Each sub-field within an Enabled OpCode Set field shall be interpreted as follows:
  - 0x0 indicates the corresponding OpCode shall not be enabled. Unsupported and Reserved OpCodes shall be hardwired to 0x0 in the supported and enabled fields.
  - 0x1 indicates the corresponding OpCode is enabled and the component's role shall be as a Requester.
    - If the OpCode is a request operation, then the Requester is enabled to generate the corresponding request packet.
    - If the OpCode is a response operation, then the Requester is enabled to receive and execute the corresponding response packet.
  - 0x2 indicates the corresponding OpCode is enabled and the component's role shall be as a Responder.
    - If the OpCode is a request operation, then the Responder is enabled to receive and execute the corresponding request packet.
    - If the OpCode is a response operation, then the Responder is enabled to generate the corresponding response packet.
  - 0x3 indicates the corresponding OpCode is enabled and the component's role shall be as a Requester-Responder.
    - If the OpCode is a request operation, then the Requester-Responder is enabled to generate as well as receive and execute the corresponding request packet.
    - If the OpCode is a response operation, then the Requester-Responder is enabled to generate as well as receive and execute the corresponding response packet.
- Software shall be responsible for determining the common OpCodes among communicating components and setting the corresponding enabled OpCodes to a non-zero value that reflects the component's role within the enabled OpCode Sets.
- A component shall not issue a packet containing an OpCode that has not been enabled.
- If a component supports the P2P-Core OpClass, the P2P-Coherency, or the P2P-Vendor-defined OpClass, then all OpCodes shall be applicable across all component interfaces.

+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 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0					Size	Vers	Type	< Byte 0x0							
OpCode Set CAP 1															
Atomic LAT	Swap-Compare Atomic Sizes	Floating Atomic Sizes	Logical-Fetch Atomic Sizes	Arithmetic Atomic Sizes	Write Poison Sizes	R0	Cache Sizes	< Byte 0x8							
Opt Set PTR					OpCode Set PTR										
VOCL 8	VOCL 7	VOCL 6	VOCL 5	VOCL 4	VOCL 3	VOCL 2	VOCL 1	R1							
R2															
+15	+14	+13	+12	+11	+10	+9	+8	+7 +6 +5 +4 +3 +2 +1 +0							
R4					Next OpCode Set PTR			OpCode Set ID Control 1 R3							
Enabled Core 64 OpCode Set					Supported Core 64 OpCode Set										
Enabled Control OpCode Set					Supported Control OpCode Set										
Enabled P2P-Core OpCode Set					Supported P2P-Core OpCode Set										
Enabled P2P-Coherency OpCode Set					Supported P2P-Coherency OpCode Set										
Enabled Atomic 1 OpCode Set					Supported Atomic 1 OpCode Set										
Enabled LDM 1 OpCode Set					Supported LDM 1 OpCode Set										
Enabled Advanced 1 OpCode Set					Supported Advanced 1 OpCode Set										
Enabled OpClass 0x5 OpCode Set					Supported OpClass 0x5 OpCode Set										
...															
Enabled OpClass 0x14 OpCode Set					Supported OpClass 0x14 OpCode Set										
Enabled Context ID OpCode Set					Supported Context ID OpCode Set										
Enabled Multicast OpCode Set					Supported Multicast OpCode Set										
Enabled SOD OpCode Set					Supported SOD OpCode Set										
Enabled Multi-Op Request Sub-Op Set					Supported Multi-Op Request Sub-Op Set										
R5					Enabled Read Multi-Op Set	Supported Read Multi-Op Set									
...															
Enabled P2P Vendor-defined Set					Supported P2P Vendor-defined Set										
Enabled VDO OpCode Set 1					Supported VDO OpCode Set 1										
...															
Enabled VDO OpCode Set 8					Supported VDO OpCode Set 8										
Supported P2P-Core SubOp Request Set															
Enabled P2P-Core SubOp Request Set															
Supported P2P-Core SubOp Response Set															
Enabled P2P-Core SubOp Response Set															
Supported P2P-Coherency SubOp Request Set															
Enabled P2P-Coherency SubOp Request Set															
Supported P2P-Coherency SubOp Response Set															
Enabled P2P-Coherency SubOp Response Set															
PM-UUID															
VDO-UUID 1															
...															
VDO-UUID 8															

Figure 8-8: OpCode Set Structure Format

Table 8-15: OpCode Set Structure Fields

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description
Version (Vers)  OpCode Set CAP 1	4	0x1	M	RO	Structure Version
	64	-	M	RO	OpCode Set Capability 1
		Bit 0			P2P Vendor-Defined 0b—Unsupported 1b—Supported
		Bit 1			VDO OpClass 1 0b—Unsupported 1b—Supported
		Bit 2			VDO OpClass 2 0b—Unsupported 1b—Supported
		Bit 3			VDO OpClass 3 0b—Unsupported 1b—Supported
		Bit 4			VDO OpClass 4 0b—Unsupported 1b—Supported
		Bit 5			VDO OpClass 5 0b—Unsupported 1b—Supported
		Bit 6			VDO OpClass 6 0b—Unsupported 1b—Supported
		Bit 7			VDO OpClass 7 0b—Unsupported 1b—Supported
		Bit 8			VDO OpClass 8 0b—Unsupported 1b—Supported
		Bits 10:9			Atomic Data Endian Type 0x0—Atomics Unsupported 0x1—Little Endian Data Atomic Support 0x2—Big Endian Data Atomic Support 0x3—Little & Big Endian Data Atomic Support
		Bits 14:11			Protocol version associated with this OpCode set.

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description
OpCode Set CAP 1 Control		Bits 16:15			0x0—Version 1 0x1-0xF—Reserved  Interrupt Role (if <i>Interrupts</i> are supported)  0x0—Unsupported  0x1—Requester role (component generates but does not execute interrupts)  0x2—Responder role (component executes but does not generate interrupts)  0x3—Requester-Responder role (component generates and executes interrupts)  A component's Interrupt Role may differ from the Component Role.
					Single OpCode Set Support—If the component provisions multiple OpCode Set structures, then this bit indicates whether a single or multiple OpCode Set structures may be enabled at a given time.  0b—Single OpCode Set Enabled 1b—Multiple OpCode Sets Enabled
					Per Destination OpCode Set Support—If the component supports multiple OpCode Set structures, then this field indicates if the component supports associating an OpCode Set structure with a destination through the <i>Component Destination Table Structure</i> or uses a method outside of this specification's scope.  0b—Component Destination Table Supported 1b—Vendor-defined Method Supported
					LDM 1 Read Response Meta Support—Indicates if the component supports the Meta field within in LDM 1 Read Response packets (see <i>Meta Read Response and Meta Write</i> ).  0b—Unsupported 1b—Supported
					Reserved
		32	M	RW	OpCode Set Capability Control
					Enabled Cache Line Size—If a component supports any operation that is interlocked to a

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description
Cache Line Sizes					single cache line size, then this field shall indicate the cache-line size.  This field shall apply to all OpCode sets  0x0—Disabled 0x1—32 bytes 0x2—64 bytes 0x3—128 bytes 0x4—256 bytes 0x5-0x7—Reserved
		Bits 31:3			RsvdP
Write Poison Sizes	4	-	O	RO	If a component supports a cache or interacts with components that support a cache, then this field indicates the supported cache line sizes. A component may indicate support for multiple cache line sizes by setting multiple bits. Only a single cache line size shall be enabled at a given time.  If Bit <sub>k</sub> = 1b, then the corresponding cache line size is supported. If Bit <sub>k</sub> == 0b, then the size is unsupported.  Bit 0: 32 bytes Bit 1: 64 bytes Bit 2: 128 bytes Bit 3: 256 bytes
Arithmetic Atomic Sizes	8	-	O	RO	If a component supports <i>Write Poison</i> , then this field indicates the size data that can be poisoned. A component may indicate support for multiple sizes by setting multiple bits.  If Bit <sub>k</sub> = 1b, then the corresponding cache line size is supported. If Bit <sub>k</sub> == 0b, then the size is unsupported.  Bit 0: 32 bytes Bit 1: 64 bytes Bit 2: 128 bytes Bit 3: 256 bytes Bit 4: 4096 bytes Bits 7:5: Reserved
	8	-	O	RO	If a component supports arithmetic Atomic operations, then this bit mask indicates the supported sizes. If Bit <sub>k</sub> == 1b, then the

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description
Logical-Fetch Atomic Sizes					<p>corresponding size is supported for all supported Atomic operations; if Bit<sub>K</sub> == 0b, then the size is unsupported.</p> <p>Bit 0—8-bit Atomic Bit 1—16-bit Atomic Bit 2—32-bit Atomic Bit 3—64-bit Atomic Bit 4—128-bit Atomic Bit 5—256-bit Atomic Bit 6—512-bit Atomic Bit 7—Reserved</p>
	8	-	O	RO	<p>If a component supports logical or fetch Atomic operations, then this bit mask indicates the supported sizes. If Bit<sub>K</sub> == 1b, then the corresponding size is supported for all supported Atomic operations; if Bit<sub>K</sub> == 0b, then the size is unsupported.</p> <p>Bit 0—8-bit Atomic Bit 1—16-bit Atomic Bit 2—32-bit Atomic Bit 3—64-bit Atomic Bit 4—128-bit Atomic Bit 5—256-bit Atomic Bit 6—512-bit Atomic Bit 7—Reserved</p>
Floating Point Atomic Sizes	8	-	O	RO	<p>If a component supports logical or fetch Atomic operations, then this bit mask indicates the supported sizes. If Bit<sub>K</sub> == 1b, then the corresponding size is supported for all supported Atomic operations; if Bit<sub>K</sub> == 0b, then the size is unsupported.</p> <p>Bit 0—8-bit Atomic Bit 1—16-bit Atomic Bit 2—32-bit Atomic Bit 3—64-bit Atomic Bit 4—128-bit Atomic Bit 5—256-bit Atomic Bit 6—512-bit Atomic Bit 7—Reserved</p>

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description
<b>Swap-Compare Atomic Sizes</b>	8	-	O	RO	<p>If a component supports swap or any variation of a compare Atomic operation, then this bit mask indicates the supported sizes. If Bit<sub>K</sub> == 1b, then the corresponding size is supported for all supported Atomic operations; if Bit<sub>K</sub> == 0b, then the size is unsupported.</p> <p>Bit 0—8-bit Atomics          Bit 1—16-bit Atomics          Bit 2—32-bit Atomics          Bit 3—64-bit Atomics          Bit 4—128-bit Atomics          Bit 5—256-bit Atomics          Bit 6—512-bit Atomics          Bit 7—Reserved</p>
<b>Atomic LAT</b>	16	-	O	RO	<p>Worst case latency from the time of the first bit of an Atomic request is received until the last bit of an Atomic Response is transmitted by a Responder. If a component supports Atomic request packets of variable execution times (e.g., variability due to persistence, coherency, multiple operations, etc.), then the Atomic LAT shall represent the worst-case Atomic request packet execution time.</p> <p>Exponent is used to calculate latency in picoseconds as:</p> <p>Latency = Atomic LAT * Core Latency Scale</p>
<b>OpCode Set PTR</b>	32	-	M	RO	Pointer to OpCode Set (Set ID 0)
<b>Opt Set UUID PTR</b>	32	-	O	RO	Points to a single optional set of OpCodes as well as UUIDs associated with vendor-defined operations.
<b>VOCL[1-8]</b>	5	-	O	RW	<p>Vendor-defined OpClass label assigned per Vendor-defined OpCode Set.</p> <p>The VOCL value is used within the explicit OpClasses packet's OpClass field independent of the underlying actual VCO OpCode Set number. For example, if two components agree to use OpClass 0x1C, then component A could set VOCL 1 = 0x1C while component B could set VOCL 4 = 0x1C.</p>

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description
<b>Set ID</b>					<p>This provides a level of indirection to enable components to interoperate without requiring communicating components to have identical VDO OpCode Set layouts.</p> <p>VOCL 1 corresponds to VDO OpCode Set 1; VOCL 2 to VDO OpCode Set 2, and so forth.</p>
	3	-	M	RO	<p>OpCode Set Identifier associated with this set of OpCode Set fields. .</p> <p>Each set of OpCode sets shall have a unique identifier.</p> <p>Identifiers shall be assigned in monotonically increasing order starting at 0x0.</p> <p>Sets shall be linked based on their identifiers, i.e., Set ID 0x0 Next OpCode Set PTR points to Set ID 0x1, and so forth.</p>
<b>OpCode Set ID Control 1</b>	16	-	M	RW	Controls for this specific OpCode Set
		Bit 0			<p>OpCode Set Enabled—Determines if this OpCode Set is enabled.</p> <p>0b—Disabled 1b—Enabled</p>
		Bits 15:1			RsvdP
<b>Next OpCode Set PTR</b>	32	-	O	RO	<p>Pointer to additional OpCode Sets</p> <p>A component may support a maximum of 8 OpCode Sets.</p> <p>Each OpCode Set Structure may be associated with a given destination component through the <i>Component Destination Table Structure</i>.</p> <p>A Null pointer indicates there is no corresponding OpCode Set accessible through this field.</p>
<b>Supported OpCode Set</b>	64	-	M	RO	Supported OpCodes or Sub-OpCodes for a given OpClass. If an OpClass is supported, then all mandatory OpCodes shall be hardwired to “supported” per the component’s role.
<b>Enabled OpCode Set</b>	64	-	M	RW	Enabled OpCodes or Sub-OpCodes for a given OpClass. If an OpClass is supported, then all

Field Name	Size (bits)	Value / Bit Location	M/ O	Access	Description
<b>Supported Read Multi-Op</b>					mandatory OpCodes shall be hardwired to “enabled” per the component’s role.
<b>Enabled Read Multi-Op</b>	32	-	O	RO	Supported Sub-OpCodes for Read <i>Multi-Op Requests</i>
<b>Supported Multi-Op Sub-Ops</b>	32	-	O	RW	Enabled Sub-OpCodes for Read <i>Multi-Op Requests</i>
<b>Enabled Multi-Op Sub-Ops</b>	64	-	O	RO	Supported <i>Multi-Op Requests</i> Sub-Ops
<b>Enabled Multi-Op Sub-Ops</b>	64	-	O	RW	Enabled <i>Multi-Op Requests</i> Sub-Ops
<b>Supported SubOp Sets</b>	256	-	O	RO	Supported P2P-Core and P2P-Coherency OpClass SubOp request and response sets.
<b>Enabled SubOp Sets</b>	256	-	O	RW	Enabled P2P-Core and P2P-Coherency OpClass SubOp request and response sets.
<b>PM-UUID</b>	128	-	O	RO	The PM-UUID is used to identify the vendor-defined OpCodes for the P2P-Core, P2P-Coherency, and P2P-Vendor-defined and Multicast OpClasses.
<b>VDO-UUID[1-8]</b>	128	-	O	RO	VDO-UUID[1-8] respectively identify the corresponding VDO OpClass OpCode Sets[1-8]. Each supported VDO-UUID shall be unique, i.e., the same UUID value shall not be used in multiple VDO-UUID fields. If a VDO-UUID is unsupported, then the corresponding field shall be Reserved.
<b>R0</b>	4	-	-	-	RsvdP
<b>R1</b>	24	-	-	-	RsvdP
<b>R2</b>	64	-	-	-	Reserved
<b>R3</b>	13	-	-	-	RsvdP
<b>R4</b>	64	-	-	-	Reserved
<b>R5</b>	64	-	-	-	Reserved

## 8.16. Interface Structure

The following are the features and requirements associated with the Interface structure:

- Each component interface shall support the Interface structure, and the Interface structure shall use the *Interface Structure Format*.
  - If the interface supports packet relay and / or Access Key validation, then the following fields shall be provisioned as illustrated (illustrated in orange): the VCAT PTR, LPRT PTR, MPRT PTR, Interface Ingress Access Key Mask, and the Interface Egress Access Key Mask. These fields shall be contiguously located as illustrated. Presence of this field is indicated by the *Interface I-CAP 1 Packet Relay-Access Key Field Presence* bit.
- A component may support 1-4096 interfaces (*Core Structure Max Interface*), each with a distinct interface structure.
  - A component may support non-homogeneous interfaces, i.e., the supported signaling rates, the number of transmit lanes, the number of receive lanes, etc. may vary on a per-interface basis. As such, Interface structure composition and capabilities may vary on a per interface basis.
- An interface may support and be enabled to only use P2P-Core, P2P-Coherency, or P2P Vendor-defined packets.
  - If the interface is used to connect components in a daisy-chain topology (P2P-Core OpClass support), then the Interface Structure Peer Daisy Interface ID shall contain the egress interface used by this interface to forward packets to the next hop in the daisy chain.
  - If the component is the last component in the daisy-chain or the component is not in a daisy-chain topology, then Peer Daisy Interface ID shall be Reserved.
  - The P2P Vendor-defined OpClass is uniquely identified by the *OpCode Set Structure*'s PM-UUID.
- If the component does not support *In-band Management*, then all management operations shall be through the component's *Out-of-band Management* interface.
- If supported and enabled, then the interface shall execute the physical link training algorithm whenever the number of transient errors exceeds a given threshold within a given time interval.
  - Each interface shall maintain a Transient Error Counter (TEC). The TEC shall be set to the Transient Error Threshold (TETH).
    - Whenever a Transient Error is detected, TEC shall be decremented by one.
  - The Transient Error Timer (TET) shall be continuously executed while the interface is in the L-Up or the L-Up-LP state.
  - TET shall be restarted and TEC shall be reset to the default or non-zero TETH value whenever the interface transitions to the L-Up or the L-Up-LP state.
  - TET shall be suspended whenever the interface is in the I-Down state or an operational state that suspends packet transmission (e.g., a low-power state), the interface is executing the link training algorithm, or the interface has been reset or disabled.
  - TET shall be set based on the following formulas:
    - If  $TETE = 0$ , then  $TET = 10.0$  seconds,  $+10\% / -0\%$
    - If  $TETE \neq 0$ , then  $TET = 10^{TETE} \text{ ms}$ ,  $+10\% / -0\%$
  - If  $TEC = 0$  before TET expires, then the interface shall invoke physical layering retraining and shall set the I-Status Link Training status bit to 1b.

- If physical layer retraining is invoked more than once every 255 TET expirations, then this shall be classified as an Excessive Physical Layer Retraining Event, and shall be handled as configured in *Interface Error Fields*.
- If TEC is non-zero when TET expires, TEC is reset to [default | non-zero TETH].

5 **Developer Note:** The purpose of the TEC and TET is to only invoke physical layer retraining when operating conditions warrant and not due to a transient burst event. Given an expected BER and signaling rate, the TET may be treated as the time window that at least one transient error will occur under normal operation conditions. For example, assuming a BER of  $10^{-12}$  and four transmit lanes operating at 25 GT/s, a link will likely experience one error every 10 seconds. If the TET is set to 10 seconds, then the TEC needs to be set to a value larger than 1. How much larger is dependent upon the solution and the operating environment. In many cases, transient errors occur in bursts where multiple packets are impacted by the same random event, e.g., transient connector vibration. Developers should consider all sources of errors, and determine a TEC that results in physical layer retraining only when operating conditions warrant.

- If a component supports packet relay and if Access Key validation is supported, then:
  - The component shall support Access Key validation on all interfaces used to relay packets between components.
  - Each component interface used to relay packets shall provision the Interface Ingress Access Key Mask and the Interface Egress Access Key Mask as illustrated in *Interface Structure Format*. The Interface Ingress Access Key Mask is used to perform Access Key validation on received packets, and the Interface Egress Access Key Mask is used to perform Access Key validation on transmitted packets.
    - For each mask, each Bit<sub>K</sub> corresponds to Access Key<sub>K</sub>.
    - If Bit<sub>K</sub> is 1b, then Access Key<sub>K</sub> is valid, and any packet that contains this Access Key shall be permitted.
    - If Bit<sub>K</sub> is 0b, then Access Key<sub>K</sub> is invalid, and packets containing this Access Key shall not be permitted, and any packet containing this Access Key shall be handled as an Access Violation.
    - If Bit<sub>0</sub> is 1b, then all packets without an Access Key field and all packets that contain an Access Key field that is set to the Default Access Key shall be permitted.
  - A permitted packet means:
    - If the packet was received, then the component shall relay the packet from the ingress interface to the egress interface.
    - If the packet is to be transmitted, then the component shall transmit the packet to the directly-attached, peer interface.
    - If Access Key validation is disabled, then all packets shall be permitted.
  - If a packet is not permitted, then:
    - The packet shall be silently discarded, or if transmission has already begun, then the packet's ECRC field shall be stomped.
    - If interface error reporting is enabled, then the Access Violation actions as configured in *Interface Error Fields* shall be taken.
- A component used to relay packets from a directly-attached Requester or Responder may validate a packet's SCID and, if supported, the packet's SSID, to prevent a Requester or Responder from transmitting packets using an unassigned CID or SID. Validation is performed by comparing the packet's SCID and SSID with the Interface structure's Peer CID and Peer SID fields. The Peer CID

5 and Peer SID fields may be populated by performing a Peer-Attribute *Link CTL* request (this request is used to identify the peer component attached to this interface in order to validate the physical topology and to detect cabling issues) and management may enable validation once populated. Alternatively, management may configure these fields with the assigned CID and SID and enable validation. If a mismatch is detected, then the Access Violation actions as configured in *Interface Error Fields* shall be taken.

- Interface Structure organization:
  - The Interface Structure corresponding to Interface 0 shall be located by *Core Structure Interface 0 PTR*.
    - If a component supports multiple interfaces, then the corresponding Interface Structures shall be organized into a sequential list linked together as illustrated in *Example Aggregated Interfaces (A)*. The Next I-PTR is used to link the Interface Structures, i.e., Interface Structure 0 Next I-PTR points to Interface Structure 1, Interface Structure 1 Next I-PTR points to Interface Structure 2, and so forth.
  - A component may organize interfaces into Interface Groups. *Example Interface Groups* illustrates two Interface Groups, each composed of four interfaces. Interface Group composition shall be indicated by the Interface Structure Next IG-PTR.
    - Interface Groups may be supported by any component type.
    - If a component supports an Interface Group, then the corresponding Interface structures shall be organized into a sequential list linked together using the Interface Structure Next IG-PTR. The first interface in an Interface Group shall be the interface with the lowest Interface ID, and shall be either ID 0 or an even-numbered ID value (i.e., 0, 2, 4, ...). The second interface in an Interface Group shall be the interface with the next lowest Interface ID, and so forth. For example, interfaces [0-3] constitute Interface Group 0, and the corresponding Interface Structures are linked together in sequential order [0-3].
  - Interfaces within a single Interface Group may be aggregated to operate as a single interface. For example, if a component supports four interfaces, then these interfaces can be configured to operate as four independent interfaces (*Example Aggregated Interfaces (A)*), as two independent interfaces (*Example Aggregated Interfaces (B)*), or as one interface (*Example Aggregated Interfaces (C)*).
    - To indicate aggregated interface support, each interface within an Interface Group shall set *Interface I-CAP 1 Aggregated Interface Support* = 1b. If supported, then all interfaces within an Interface Group shall support the same configuration capabilities, the same physical layer(s) and associated capabilities, the same resource capacities, and the same operational semantics.
    - The following actions are taken to configure and enable an aggregated interface:
      1. Using policies outside of this specification's scope, management determines which interfaces to aggregate.
      2. The interface with the lowest Interface ID shall be the single, aggregated interface (SAI); all other interfaces that constitute the aggregated interface shall be non-operational, aggregate interfaces (NAI).
        - o The number of interfaces that constitute a SAI shall be a power of 2, i.e., 2, 4, 8, 16, ...
      3. Management shall configure SAI *Interface I-CAP 1 Control Aggregated Interface Control* = 0x2, and for each NAI, *Interface I-CAP 1 Control Aggregated Interface Control* = 0x1.

5

10

15

20

25

30

35

40

4. The interfaces that constitute an aggregated interface are located through the Next AI-PTR field. Management copies the SAI's Next IG-PTR (pointer to the Interface structure corresponding to SAI's Interface ID + 1) to the SAI's Next AI-PTR, and for each additional aggregated interface, management takes similar steps to create a sequential list of structures. For example, Example Interface Group and Aggregated Interface Pointers illustrates 8 interfaces organized into two Interface Groups [0, 1], and management wants to create three aggregated interfaces—ID 0, ID 2, and ID 4. To inform the underlying hardware which interfaces constitute an aggregated interface, management takes the following steps:
  - o For interfaces ID 0, ID 2, and ID 4, sets SAI *Interface I-CAP 1 Control* Aggregated Interface Control = 0x2.
  - o For interfaces ID 1, ID 3, ID 5, ID 6, and ID 7, sets SAI *Interface I-CAP 1 Control* Aggregated Interface Control = 0x1.
  - o Copies ID 0 Next IG-PTR to ID 0 Next IA-PTR, and copies Null to ID 1 Next IA-PTR.
  - o Copies ID 2 Next IG-PTR to ID 2 Next IA-PTR, and copies Null to ID 3 Next IA-PTR.
  - o Copies ID 4 Next IG-PTR to ID 4 Next IA-PTR, copies ID 5 Next IG-PTR to ID 6 Next IA-PTR, copies ID 5 Next IG-PTR to ID 6 Next IA-PTR, and copies Null to ID 7 Next IA-PTR.
5. Management shall configure the *Interface PHY Structure* of each interface that constitutes the aggregate interface to enable the physical layer to operate across all lanes that will compose the aggregated link.
6. Once all SAIs and NAIs are configured to operate as an aggregate interface, management initiates a *Warm Interface Reset* on each SAI and NAI (see *Warm Interface Reset* for additional actions to take).
7. When the SAI comes out of reset, it shall operate as a single aggregated interface over a single link composed of the aggregated lanes. At its discretion, an implementation may aggregate resources (e.g., flow-control buffers) from the NAIs into the SAI. Further, the interface shall be configured and managed only through the SAI's Interface structure, *Interface PHY Structure*, *Interface Statistics Structure*, *Component Mechanical Structure*, and any associated interface-specific *Vendor-Defined Structure*.
8. Until a *Full Interface Reset* is performed on the SAI, all aggregated NAIs shall not exit from an operational perspective. Any access to a NAI's control structures shall not impact SAI configuration or operation, and shall not cause the NAI to take any action. Management shall be responsible for ensuring only SAIs are visible in any control structure that uses an Interface ID. For example, egress interface identifiers within a packet relay tables.

**Developer Note:** The following examples illustrate how aggregated interfaces can be applied:

45

- A mechanical module can contain one or more independent components, yet an enclosure views the module as a single entity. For example, a module might contain a single GPU or contain multiple independent SSDs. If the communicating

5 components support aggregated interfaces, management can detect the number of independent components, and configure the enclosure and module interfaces to meet solution-specific needs. For example, if an enclosure routes four, narrower interfaces to an enclosure slot and a GPU supports only one wide interface, then management can configure the two or more of the four enclosure-supplied interfaces to operate as a SAI that matches the GPU's interface. Without aggregation, the GPU would be limited only one narrow link for communication.

- 10 • A FPGA could be designed to support processor-to-FPGA communications (single, wide interface) and FPGA-to-FPGA communications (four narrower interfaces). When inserted into a processor enclosure, management can configure the processor and the FPGA to use a single SAI (all four interfaces), two SAIs (each SAI aggregates two interfaces), or four independent interfaces based on solution-specific requirements.
- 15 • In general, developers prefer to use narrower interfaces at higher physical layer signaling rates. In order to maximize component interoperability and solution attach (Gen-Z enables any-to-any component attachment), component interfaces can be designed to support multiple signaling rates, e.g., the 802.3 electrical at 25 GT/s, 56 GT/s, and 112 GT/s. Aggregated interface support enables a component (e.g., a memory or I/O component) to be attached to a processor that supports only lower signaling rates and wider interfaces, and the same component to be attached to an alternative processor or a switch that supports high signaling rates and narrower interfaces.

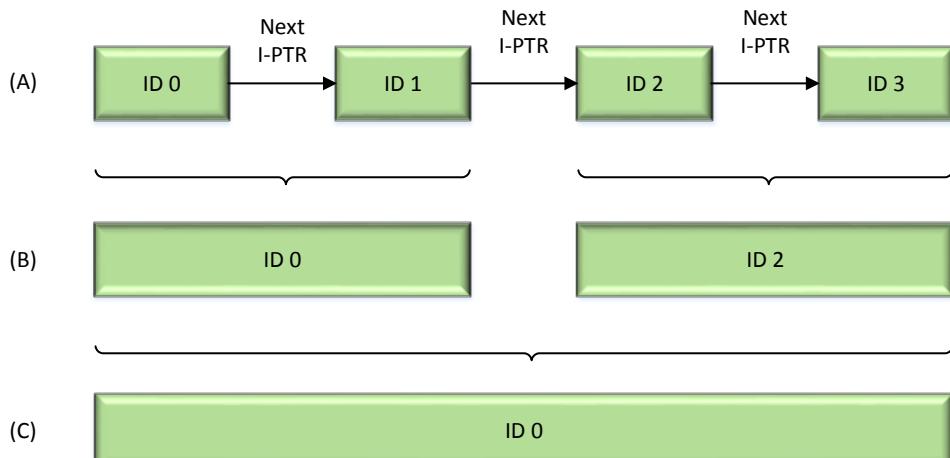


Figure 8-9: Example Aggregated Interfaces

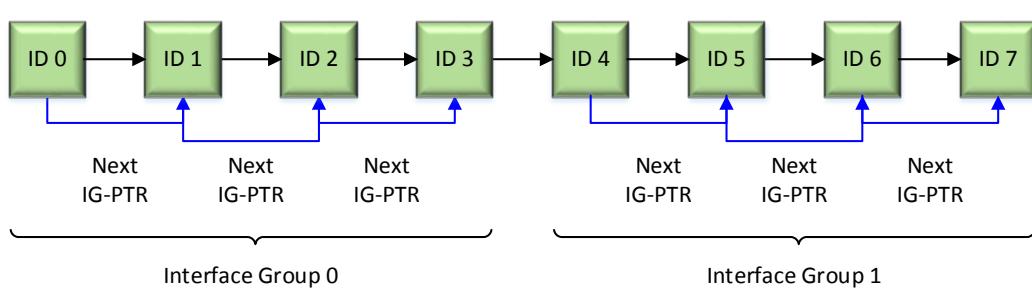


Figure 8-10: Example Interface Groups

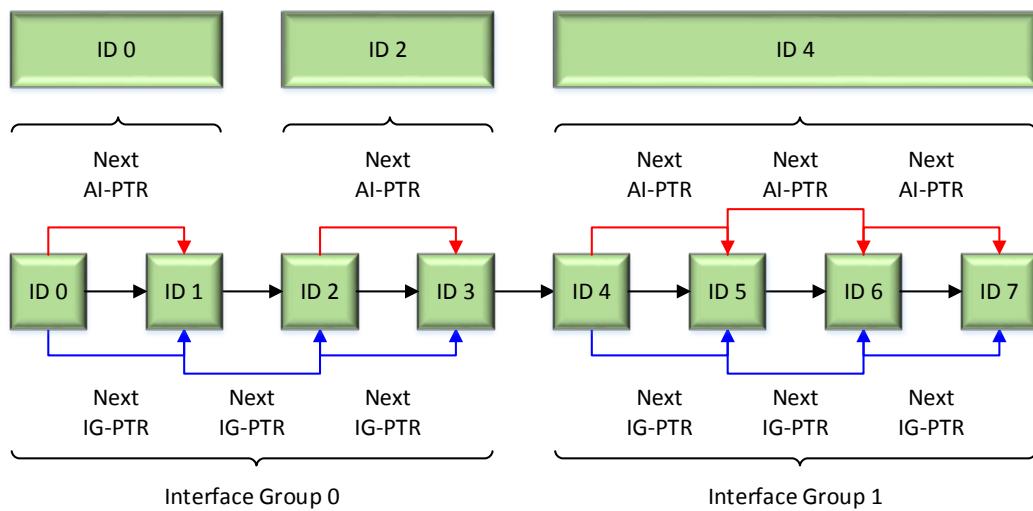


Figure 8-11: Example Interface Group and Aggregated Interface Pointers

+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	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	7 6 5 4 3 2 1 0
Peer Daisy Interface ID	RO	HVS	Interface ID	Size	Vers	Type	< Byte 0x0
I-Control				I-Status			
I-CAP 1 Control				I-CAP 1			
I-CAP 2 Control				I-CAP 2			
I-Error Trigger		I-Error Fault Injection		I-Error Detect		I-Error Status	
FC FWD Progress	TETE	TETH	I-Signal Target				
Peer Base C-Class		R1	Peer Interface ID	Max Implicit FC Credits		LSID	
Peer State				Peer SID		R2	Peer CID
VC PCO Enabled				TR CID	TR Index	Path Propagation Time	
Peer Component TTC		TTC Unit	LSRINT	R3	HVE	Rx Min Packet Start	Tx Min Packet Start
Peer Nonce							
Rx LLRT ACK		Tx LLRT ACK			R4		Aggregation Support
R5				Next I-PTR			
Next IG-PTR				Next IA-PTR			
I-STATS PTR				I-PHY PTR			
VD PTR				Mechanical PTR			

Common Interface Structure Fields (Required)

+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	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	7 6 5 4 3 2 1 0
LPRT PTR				VCAT PTR			
R6				MPRT PTR			
Interface Ingress Access Key Mask							
Interface Egress Access Key Mask							

Packet Relay / Access Key Interface Structure Fields (Optional)

Figure 8-12: Interface Structure Format

Table 8-16: Interface Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Version (Vers)	4	0x1	M	RO	Structure Version
Interface ID	12	-	M	RO	A component-unique, unsigned integer label associated with this interface. Interface IDs shall be sequentially assigned (no gaps) starting at 0, i.e., 0, 1, 2, 3, ...

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>HVS</b>	5	-	M	HwInit	<p>Highest-numbered VC Supported</p> <p>For example:</p> <p>HVS = 0 is VC0. Only VC0 supported</p> <p>HVS = 7 is VC7. VC0-VC7 supported</p> <p>See <i>Virtual Channels</i></p> <p>HVS may equal 0-31 which equates to VC0-VC31</p> <p>Default: HVS = 0</p>
<b>Peer Daisy Interface ID</b>	8	-	O	RW	<p>Interface Identifier of the co-located interface used to forward packets to the next component within a daisy-chain topology.</p> <p>In fixed-topology solutions, a component may hardwire this field to the peer daisy-chain interface identifier.</p> <p>Only applicable to interfaces that support P2P-Core and daisy-chained topologies; else shall be Reserved.</p> <p>Peer Daisy Interface ID shall be a value from 0-255, inclusive.</p>
<b>I-Status</b>	32	-	M	-	See <i>I-Status Structure Field</i>
<b>I-Control</b>	32	-	M	-	See <i>I-Control Space Structure Field</i>
<b>I-CAP 1</b>	32	-	M	-	See <i>I-CAP 1 Field</i>
<b>I-CAP 1 Control</b>	32	-	M	-	See <i>I-Cap 1 Control Field</i>
<b>I-CAP 2</b>	32	-	M	-	See <i>I-Cap 2 Field</i>
<b>I-CAP 2 Control</b>	32	-	M	-	See <i>I-Cap 2 Control Field</i>
<b>I-Error Status</b>	16	-	O	-	See <i>I-Error Status</i>
<b>I-Error Detect</b>	16	-	O	-	See <i>I-Error Detect</i>
<b>I-Error Fault Injection</b>	16	-	O	-	See <i>I-Error Fault Injection</i>
<b>I-Error Trigger</b>	16	-	O	-	See <i>I-Error Trigger</i>
<b>I-Signal Target</b>	48	-	O	RW	Each I-Signal Target 3-bit sub-field indicates how the associated error is to be signaled. If configured, then the component shall generate one or both component-local interrupts, an

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
TETH	4	-	O	RW	<p><i>Unsolicited Event (UE) Packet</i>, or both component local interrupt(s) and an <i>Unsolicited Event (UE) Packet</i>.</p> <p>If this field is supported, then the component shall support the <i>Component Error and Signal Event Structure</i>. The <i>Component Error and Signal Event Structure</i> is used to configure the component-local interrupts and the <i>Unsolicited Event (UE) Packet</i> destination.</p> <p>0x0—No signal is generated 0x1—Component-local interrupt 0 0x2—Component-local interrupt 1 0x3—Component-local interrupts 0 and 1 0x4—<i>Unsolicited Event (UE) Packet</i> 0x5—Component-local interrupt 0 and an <i>Unsolicited Event (UE) Packet</i> 0x6—Component-local interrupt 1 and an <i>Unsolicited Event (UE) Packet</i> 0x7—Component-local interrupts 0 and 1, and an <i>Unsolicited Event (UE) Packet</i></p> <p>Sub-field 0 is located at byte 0, bit 0 of the I-Signal Target field. Sub-field 1 is contiguously located next to sub-field 0. And so forth.</p> <p>Each Sub-field<sub>K</sub> shall correspond respectively to <i>I-Error Detect Bit<sub>K</sub></i>.</p> <p>If an <i>Unsolicited Event (UE) Packet</i> is generated, then the packet's Interface ID field shall contain this interface's identifier and Event-specific Field[3:0] shall be set to K where K corresponds to the bit in the <i>I-Error Detect</i> field of the detected error.</p>
					<p>Transient Error Threshold—the maximum number of transient errors allowed prior to transient error threshold timer expiration.</p> <p>0x0—8 0x1—16 0x2—32 0x3—64 0x4—128 0x5—256 0x6—512</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
TETE					<p>0x7—1024 0x8—2048 0x9-0xF—Reserved</p> <p>If modified, then the new value shall take effect on the next timer expiration.</p>
	4	-	O	RO	<p>Transient Error Timer Exponent—used to calculate the Transient Error Timer (TET).</p> <p>TETE shall be a value [0x0-0x9] inclusive.</p> <p>Values 0xA-0xF shall be Reserved. If configured, a component should default the timeout value to the maximum permitted value.</p>
FC FWD Progress	8	-	M	RW	<p>The maximum number of FCTT expirations (see <i>Explicit Flow Control</i>) without <b>new</b> flow-control credits being returned on any enabled <math>VC_k</math> with fewer than <math>VC_k</math> MIN flow-control credits. If the interface exceeds this number, then it shall take the actions indicated by Auto Stop and the <i>Interface Error Fields</i>.</p> <p>FC FWD Progress shall be a value between 2 and 255 (inclusive).</p> <p>If modified, then the new value shall take effect on the next timer expiration.</p>
LSID	16	-	O	RW	<p>Local Subnet Identifier—if a component supports multi-subnet packet relay based on a packet's DSID, then this field is configured with the SID of the directly-attached subnet.</p> <p>If <i>Component Switch Structure</i> MSS == 1b, then this field shall be configured, else this field shall be Reserved.</p>
Max Implicit FC Credits	16	-	O	HwInit / RWS	<p>If the interface supports implicit flow-control, then this indicates the maximum number of <i>Implicit Flow Control</i> credits that may be consumed without causing buffer overflow.</p> <p>If a Responder interface, then this is a HwInit field.</p> <p>If a Requester interface, then this is a read-write sticky field. If this field is modified, then</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Peer Interface ID					the change will take effect upon the next <i>Warm Interface Reset</i> .
Peer Base C-Class	12	-	MC	RO	This field is updated by hardware upon the successful completion of a Peer-Attribute <i>Link CTL</i> operation.
Peer CID	16	-	M	RO	This field is set by hardware upon completion of a Peer-Attribute <i>Link CTL</i> operation. This contains the Base C-Class of the component presently attached to this interface.  Peer State Valid Peer Base C-Class indicates if this field contains a valid value.
Peer SID	12	-	MC	RW	This field may be configured by software or updated by hardware upon the successful completion of a Peer-Attribute <i>Link CTL</i> operation.  Peer State Valid Peer CID indicates if this field contains a valid value.  If the interface supports only the P2P-Core, P2P-Coherency, or P2P-Vendor-defined OpClass, then this field shall be Reserved.
Peer State	16	-	MC	RW	This field may be configured by software or updated by hardware upon the successful completion of a Peer-Attribute <i>Link CTL</i> operation.  If the interface does not support multi-subnet communications, then this field shall be Reserved.  Peer State Valid Peer SID indicates if this field contains a valid value.  If the interface supports only the P2P-Core, P2P-Coherency, or P2P-Vendor-defined OpClass, then this field shall be Reserved.
	32	-	M	-	Peer State bits are updated automatically by the interface as a result of an <i>Interface Reset</i> , link transition to L-Down, or a Peer-Attribute <i>Link CTL</i> packet exchange.  A subset of Peer State bits may be updated by software.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
		Bits 2:0		RO	<p>Peer Component C-State</p> <p>0x0—Unknown 0x1—C-Down 0x2—C-CFG 0x3—C-Up 0x4—C-LP 0x5—C-DLP 0x6-0x7—Reserved</p>
		Bit 3		RO	<p>Peer Component Manager Type</p> <p>0b—Primary Manager 1b—Fabric Manager</p>
		Bit 4		RW	<p>Valid Peer CID—Indicates if the Peer CID field contains a valid CID. Software may configure the Peer CID or this field may be updated as a result of a Peer-Attribute <i>Link CTL</i> packet exchange.</p> <p>If the interface does not support explicit OpClasses, then this bit shall be hardwired to 0b.</p> <p>0b—Invalid 1b—Valid</p>
		Bit 5		RW	<p>Valid Peer SID—Indicates if the Peer SID field contains a valid SID. Software may configure the Peer SID or this field may be populated as a result of a Peer-Attribute <i>Link CTL</i> packet exchange.</p> <p>If the interface does not support explicit OpClasses, then this bit shall be hardwired to 0b.</p> <p>0b—Invalid 1b—Valid</p>
		Bit 6		RO	<p>Valid Peer Interface ID—Indicates if the Peer Interface ID field contains the peer interface's identifier.</p> <p>0b—Invalid 1b—Valid</p>
		Bit 7		RO	<p>Peer Interface P2P-Core OpClass Support</p> <p>0b—Unsupported 1b—Supported</p>
		Bit 8		RO	<p>Peer Interface P2P-Coherency OpClass Support</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					0b—Unsupported 1b—Supported
		Bit 9		RO	Peer Interface P2P-Vendor-defined OpClass Support 0b—Unsupported 1b—Supported
		Bit 10		RO	Peer Interface explicit OpClasses Support 0b—Unsupported 1b—Supported
		Bit 11		RO	Peer Component Multiple CID Configuration If Valid Peer CID == 0b, then this bit shall be ignored. 0b—Single CID 1b—Multiple CIDs
		Bit 12		RO	Peer Component Multiple SID Configuration If Valid Peer SID == 0b, then this bit shall be ignored. 0b—Single SID 1b—Multiple SIDs
		Bit 13		RO	Peer Component Home Agent Support 0b—Unsupported 1b—Supported
		Bit 14		RO	Peer Component Caching Agent Support 0b—Unsupported 1b—Supported
		Bits 16:15		RO	Peer Interface Flow-control Support 0x0— <i>Implicit Flow Control</i> Support 0x1— <i>Explicit Flow Control</i> Support 0x2— <i>Implicit Flow Control and Explicit Flow Control</i> Support 0x3—Reserved
		Bit 17		RO	Valid Peer Base C-Class—Indicates if the Peer Base C-Class field contains the peer component's Base C-Class. 0b—Invalid 1b—Valid
		Bits 31:18		-	RsvdP

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Path Propagation Time	16	-	O	RW	<p>If the interface is used to transmit explicit OpClass packets, then software may configure this field with the propagation time in ns for a bit to cross the physical path.</p> <p>Alternatively, software may initiate a <i>Link CTL</i> Path Time exchange. Upon completion of this exchange, the interface populates this Path Propagation Time field.</p> <p>Path Propagation Time is used to update the Deadline sub-field (see <i>Congestion and Packet Deadline</i>).</p> <p>This field is primarily intended for variable channel length solutions, e.g., cable applications where the propagation time varies with length.</p>
TR Index	4	-	O	RW	<p>If this is a TR interface, then this field indicates the TR Table entry to use.</p> <p>If this is not a TR interface, then this field shall be Reserved.</p>
TR CID	12	-	O	RW	<p>If this a TR interface, then this field contains the CID that identifies this TR within the attached subnet.</p> <p>If this not a TR interface, then this field shall be Reserved.</p>
VC PCO Enabled	32	-	O	RWS	<p>If a Bit<sub>k</sub> == 1b, then VC<sub>k</sub> shall use PCO Communication semantics.</p> <p>If the interface does not support PCO Communications, then this field shall be Reserved.</p> <p>If this field is modified, then the change will take effect upon the next <i>Warm Interface Reset</i>.</p>
Tx Packet Alignment	8	-	M	RO	(Tx Packet Alignment * 4 bytes) is the packet boundary that Gen-Z Core shall use for sending non- <i>Link Idle</i> packets on CORE_TXDATA on the PLA interface. Tx Packet Alignment shall be set to 0x10 until the interface has successfully received and executed a Tx Packet Alignment <i>Link CTL</i> packet exchange indicating the peer

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Rx Packet Alignment					<p>interface's Rx Packet Alignment value (this value is copied into the Tx Packet Alignment field).</p> <p>The Tx Packet Alignment value shall be a non-zero power of 2, i.e., 1, 2, 4, 8, etc.</p>
	8	-	M	RO	<p>(Rx Packet Alignment * 4 bytes) is the packet boundary supported by Gen-Z Core for receiving non-<i>Link Idle</i> packets on PHY_RXDATA on the PLA interface.</p> <p>The Rx Packet Alignment value shall be a non-zero power of 2, i.e., 1, 2, 4, 8, etc.</p>
Tx Min Packet Start	8	-	M	RO	<p>(Tx Min Packet Start * 4 bytes) is the minimum number of bytes between the start of any two consecutively transmitted non-<i>Link Idle</i> packets on a link, i.e., this is the number of bytes between byte 0 of the first packet and byte 0 of the second packet.</p> <p>If the size of the first packet is less than Tx Min Packet Start, then the transmitter shall transmit <i>Link Idle</i> packets for at least (Tx Min Packet Start – sizeof (first packet)) bytes.</p> <p>Tx Min Packet Start shall be set to 0x10 until the interface has successfully received and executed a Tx Packet Alignment <i>Link CTL</i> packet indicating the peer interface's Rx Min Packet Start value (this value is copied into the Tx Min Packet Start field).</p> <p>The Tx Min Packet Start value shall be a non-zero power of 2, i.e., 1, 2, 4, 8, etc.</p> <p>Tx Min Packet Start shall be greater than or equal to Tx Packet Alignment.</p>
Rx Min Packet Start	8	-	M	RO	<p>(Rx Min Packet Start * 4 bytes) is the minimum number of bytes supported by the Gen-Z Core for receiving non-<i>Link Idle</i> packets on PHY_RXDATA on the PLA interface.</p> <p>The Rx Min Packet Start value shall be a non-zero power of 2, i.e., 1, 2, 4, 8, etc.</p> <p>Rx Min Packet Start shall be greater than or equal to Rx Packet Alignment.</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>HVE</b>	5	-	M	RWS	<p>Highest-numbered VC Enabled (see <i>Virtual Channels</i>)</p> <p>If HVS == 0x0, then this field shall be hardwired to 0x0.</p> <p>Software may configure this field or the field may be automatically updated by a Peer-Attribute <i>Link CTL</i> packet exchange.</p> <p>If this field is modified by software, then the change will take effect upon the next <i>Warm Interface Reset</i>.</p> <p>If this field is modified by a Peer-Attribute <i>Link CTL</i> packet exchange and <i>Explicit Flow Control</i> credits have not been exchanged, then the change will take immediate effect.</p>
<b>LSRINT</b>	5	-	M	RO	<p>Maximum <i>Link Synchronization (Link SR)</i> scheduling interval exponent.</p> <p>LSRINT = 0 indicates link synchronization is unsupported.</p>
<b>TTC Unit</b>	2	-	MC	RO	<p>This field indicates the unit of time associated with a non-zero time units <i>Link RFC</i> packet value.</p> <p>0x0—1 <math>\mu</math>s 0x1—1 ms 0x2—1 s 0x3—Reserved</p> <p>If an interface does not support explicit OpClasses, then this field shall be Reserved.</p>
<b>Peer Component TTC</b>	16	-	MC	RO	<p>If I-Status Peer Link RFC TTC == 1b, then this field shall indicate the number of TTC Units used to calculate the remaining time until the peer component completes self-initialization and is ready for configuration. The interface shall update this field upon receipt of a <i>Link RFC</i> packet with a non-zero time value. The remaining time is calculated as follows:</p> <p>Peer Component TTC * TTC Unit.</p> <p>Upon receipt of a <i>Link RFC</i> packet with a zero time value, this field shall be set to 0x0.</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
PeerNonce					If an interface does not support explicit OpClasses, then this field shall be Reserved.
	64	-	O	WO	<p>A component nonce received over an interface from a peer component—See <i>Mitigating In Situ Insertion</i> for additional details.</p> <p>If this field is modified, then software shall set <i>Interface I-CAP 1 Control PeerNonce Validation Enable</i> = 1b.</p>
Aggregation Support	8	-	O	RO	<p>If an interface is capable of being a SAI, then this bit vector indicates aggregation support (if zero, then the interface does not support interface aggregation). Further, each Bit<sub>k</sub> indicates support for the corresponding number of aggregated interfaces.</p> <p>Bit 0: 2 interfaces      Bit 1: 4 interfaces      Bit 2: 8 interfaces      Bit 3: 16 interfaces      Bit 4: 32 interfaces      Bit 5: 64 interfaces      Bit 6: 128 interfaces      Bit 7: 256 interfaces</p>
Tx LLRT ACK	20	-	O	RO	<p>Transmit <i>Link-level Reliability (LLR) ACK</i> Period—determines the maximum number of UI until an interface is required to transmit a LLR ACK packet.</p> <p>This field shall be updated upon receipt of a Tx Packet Alignment <i>Link CTL</i> packet.</p> <p>If Tx LLRT ACK == 0x0, then Tx ACK period = <math>2^{20}</math></p> <p>If <i>Link-level Reliability (LLR)</i> is unsupported, then this field shall be Reserved.</p>
Rx LLRT ACK	20	-	O	RW	<p>Receive <i>Link-level Reliability (LLR) ACK</i> Period—determines the maximum number of UI this interface can tolerate before it requires receipt of a LLR ACK packet to avoid performance degradation.</p> <p>If Rx LLRT ACK == 0x0, then Rx ACK period = <math>2^{20}</math></p> <p>An interface may hardwire this value.</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					If <i>Link-level Reliability (LLR)</i> is unsupported, then this field shall be Reserved.
Next I-PTR	32	-	M	RO	Pointer to next Interface Structure or Null
Next IG-PTR	32	-	M	RO	Pointer to next Interface structure within this Interface Group. The Interface structure corresponding to the last interface within an interface group shall set this field to Null.
Next AI-PTR	32	-	M	RW	Pointer to next Interface Structure within this Aggregated Interface. Management shall set Next AI-PTR = Null in the Interface structure corresponding to the last interface within an aggregated interface.
I-PHY PTR	32	-	M	RO	This field shall point to the <i>Interface PHY Structure</i> corresponding to this interface.
I-STATS PTR	32	-	O	RO	This field points to an <i>Interface Statistics Structure</i> . If unsupported, then this field shall be Null.
Mechanical PTR	32	-	O	RO	This field points to a <i>Component Mechanical Structure</i> . If unsupported, then this field shall be Null.
VD PTR	32	-	O	RO	This field points to a <i>Vendor-Defined Structure</i> . If unsupported, then this field shall be Null.
VCAT PTR	32	-	O	RO	Pointer to the VCAT (VC Action Table) associated with this interface (see <i>Component Switch Structure</i> ). If supported, then each interface shall support a distinct VCAT.
LPRT PTR	32	-	O	RO	Pointer to the LPRT (Linear Packet Relay Table) associated with this interface (see <i>Component Switch Structure</i> ). If supported, then each interface shall support a distinct LPRT.
MPRT PTR	32	-	O	RO	Pointer to the MPRT (Multi-subnet Packet Relay Table) associated with this interface (see <i>Component Switch Structure</i> ). If supported, then each interface shall support a distinct MPRT and a distinct LPRT.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Interface Ingress Access Key Mask	64	-	O	RW	<p>Per interface bit mask where each Bit<sub>K</sub> determines if the corresponding Access Key is configured on this interface and can be used during receive packet validation.</p> <p>If modified, then the effect shall take place on the next received or transmitted end-to-end packet.</p>
Interface Egress Access Key Mask	64	-	O	RW	<p>Per interface bit mask where each Bit<sub>K</sub> determines if the corresponding Access Key is configured on this interface and can be used to validate a packet prior to its transmission.</p> <p>If modified, then the effect shall take place on the next received or transmitted end-to-end packet.</p>
R0	7	-	-	-	RsvdP
R1	4	-	-	-	RsvdP
R2	4	-	-	-	RsvdP
R3	4	-	-	-	RsvdP
R4	16	-	-	-	RsvdP
R5	32	-	-	-	Reserved
R6	32	-	-	-	Reserved

### 8.16.1. Interface I-Status

Table 8-17: I-Status Structure Field

Bit Location	Access	Description
2:0	RO	<p>Interface State—The interface updates this field upon any interface state change.</p> <p>0x0—I-Down, uninitialized state</p> <p>0x1—I-CFG, interface initialization in progress</p> <p>0x2—I-Up, operational state (packet exchange enabled)</p> <p>0x3—I-LP, low-power state (packet exchange disabled)</p> <p>0x4-0x7 Reserved</p>
3	RW1CS	Full Interface Reset

Bit Location	Access	Description
4		If 1b, then a <i>Full Interface Reset</i> has completed.
	RW1CS	Warm Interface Reset If 1b, then a <i>Warm Interface Reset</i> has completed.
5	RW1C	Link RFC Status If 1b, then the interface is transmitting <i>Link RFC</i> packets.
6	RW1C	Peer Link RFC Ready If 1b, then the interface received at least one <i>Link RFC</i> packet that indicates the peer component is ready for configuration. If Peer Link RFC Ready == 1b, then the interface shall set Peer Link RFC TTC = 0b.
7	RO	Peer Link RFC TTC If 1b, then the interface received at least one <i>Link RFC</i> packet that indicates the peer component is self-initializing. Peer Component TTC and TTC Unit are used to calculate the amount of time until the peer component completes self-initialization. If Peer Link RFC TTC == 1b, then the interface shall set Peer Link RFC Ready = 0b.
8	RW1CS	Exceeded Transient Error Threshold If 1b, then the interface has exceeded the transient error threshold one or more times.
9	RW1CS	Active Link Failure If 1b, then the link has transitioned to L-Down due to a failure event, e.g., hardware failure, physical layer unexpectedly transitioned to PHY-Down, etc. This bit shall be updated independent of how <i>I-Error Detect Non-Transient Link Error Detect</i> is configured. Clearing this field shall have no impact on I-Error Non-Transient Link Error Detect fields.
10	RW1C	<i>Link CTL</i> Completed If 1b, then the present outstanding <i>Link CTL</i> request has been acknowledged.
15:11	RW	<i>Link CTL</i> Completion Status—Results of a <i>Link CTL</i> request 0x0—No outstanding <i>Link CTL</i> request or no <i>Link CTL</i> response received 0x1—Success. Link ACK return 0x2-0x1F—Error. Link NAK Return. Value indicates the highest precedence error detected. See <i>Link NAK Link CTL</i> for the specific error values.

Bit Location	Access	Description
16	RW1C	Interface Containment Detected If 1b, then <i>Interface Containment</i> was triggered, and the interface has transitioned to I-Down.
17	RW1C	Interface-Component Containment Detected If 1b, then <i>Component Containment</i> was triggered when <i>Interface Containment</i> was triggered due to an Unexpected Physical Layer Failure—PHY-Down and Interface-Component Containment Enable == 1b.
18	RW1CS	Peer Interface Incompatibility Detected—one or more peer interface attributes (see Peer State field) are incompatible such that the interfaces are unable to interoperate, e.g., incompatible OpClass, incompatible <i>Link-Local Flow Control</i> , etc.
31:19	-	RsvdZ

### 8.16.2. Interface I-Control

Table 8-18: I-Control Space Structure Field

Bit Location	Access	Description
0	RW	Interface Enable—If not already in I-Up, transition to I-Up and enable normal packet transmit and receive packet processing. If already in I-Up, then no actions shall occur. 0b—Disable (Transition to I-Down) 1b—Enable
1	RW	Initiate Full Interface Reset 1b—Initiate <i>Full Interface Reset</i> , and set <i>Interface I-Status</i> Full Interface Reset = 1b. Reads of this bit shall return 0b.
2	RW	Initiate Warm Interface Reset 1b—Initiate <i>Warm Interface Reset</i> , and set <i>Interface I-Status</i> Warm Interface Reset = 1b. Reads of this bit shall return 0b.
3	RW	<i>Link RFC</i> Packet Disable If <i>Link RFC</i> Packet Disable == 0b, then upon meeting the conditions specified in <i>Link RFC</i> , the interface shall start transmitting <i>Link RFC</i> packets. If <i>Link RFC</i> Packet Disable == 1b, then upon transitioning to L-Up, the interface shall not generate <i>Link RFC</i> packets.

		If <i>Core Structure Component CAP 1 Control In-band Management Support</i> == 0b, then this bit shall be hardwired to 1b.  0b—Enable 1b—Disable
4	RW	Initiate Peer-C-Reset—If supported, then the interface shall transmit a Peer-C-Reset <i>Link CTL</i> packet to the directly-attached component informing it to perform a <i>Full Component Reset</i>  1b—Initiate Link CTL  Reads of this bit shall return 0b.
5	RW	Initiate Peer-C-Up Transition—If supported, then the interface shall transmit a Peer-C-Up <i>Link CTL</i> packet to the directly-attached component informing it to transition to C-Up  1b—Initiate Link CTL  Reads of this bit shall return 0b.
6	RW	Initiate Peer-Attribute Request—The interface shall transmit a Peer-Attribute <i>Link CTL</i> packet to the directly-attached component.  1b—Initiate Peer-Attribute Link CTL  Reads of this bit shall return 0b.
7	RW	Enable Interface Access Key Validation—If Interface Access Key Validation is supported, then this bit is used to enable Access Key validation on all packets received and all packets to be transmitted on this interface respectively using the Interface Ingress Access Key Mask and the Interface Egress Access Key Mask.  0b—Disable 1b—Enable
8	RW	Transition I-LP—If not already in the I-LP state, transition the interface to the I-LP state. If already in I-LP, then no actions shall occur.  1b—Transition to Interface LP  Reads of this bit shall return 0b.
9	RW	Auto Stop—If an interface exceeds FC FWD Progress and Auto Stop == 1b, then the interface shall silently discard all scheduled end-to-end packets and stop transmitting new end-to-end packets. The interface may continue to receive new end-to-end packets. If <i>I-Error Trigger Interface FC FWD Progress Violation Trigger</i> == 1b, then trigger <i>Interface Containment</i> .  If the interface has auto stopped and management sets Auto Stop = 0b, then the interface shall resume transmitting new end-to-end packets.  0b—Do Not Auto Stop 1b—Auto Stop
10	RW	Initiate Path Time Request—The interface shall transmit a Path Time <i>Link CTL</i> packet to the directly-attached peer interface. The interface calculates the

		time between the transmission of the Path Time <i>Link CTL</i> and the receipt of the corresponding Link ACK <i>Link CTL</i> . Using this time, the interface shall set the Path Propagation Time field.  1b—Transmit Path Link CTL  Reads of this bit shall return 0b.
11	RW	Force <i>Physical Layer Abstraction</i> Retraining—Directs the PLA to transition to the PHY-Down-Retrain state. The physical layer shall initiate retraining based on the <i>PHY Control</i> Physical Layer Retraining Level.  1b—Initiate retraining  Reads of this field shall return 0b.
14:12	RW	Initiate L-LP Transition—The interface shall initiate an Enter-Link-LP <i>Link CTL</i> -Link ACK <i>Link CTL</i> packet exchange to transition the physical layer to the indicated PHY-LP [1-4] state. Upon completion, the link shall transition to the indicated L-LP state.  0x0—No impact 0x1-0x4—Requested PHY-LP [1-4] state 0x5-0x7—Reserved  Reads of this field shall return 0x0.
17:15	RW	Initiate L-Up-LP Transition—The interface shall initiate an Enter-Link-Up-LP <i>Link CTL</i> -Link ACK <i>Link CTL</i> packet exchange to transition the physical layer to the indicated PHY-Up-LP [1-4] state. Upon completion, the link shall transition to the indicated L-Up-LP state.  0x0—No Impact 0x1-0x4—Requested PHY-Up-LP [1-4] state 0x5-0x7—Reserved  Reads of this field shall return 0x0.
18	RW	Initiate Peer-Set-Attribute Request—The interface shall transmit a Peer-Set-Attribute <i>Link CTL</i> packet to the directly-attached component.  1b—Initiate Peer-Set-Attribute Link CTL  Reads of this bit shall return 0b.
19	RW	Ingress DR Enabled—if <i>Interface I-CAP 1</i> Packet Relay Access Key Field Provisioned == 1b, then this bit determines if the interface may receive a Control OpClass request packet with DR == 1b.  If <i>Interface I-CAP 1</i> Packet Relay Access Key Field Provisioned == 0b, then this bit shall be hardwired to 0b.  0b—Receipt of a Control OpClass request packet with DR == 1b shall be handled as a UP.  1b—May receive a Control OpClass request packet with DR == 1b.

31:20	RW	Reserved
-------	----	----------

### 8.16.3. Interface I-CAP 1

Table 8-19: I-CAP 1 Field

Bit Location	Access	Description
0	RO	Interface Containment Support 0b—Not Supported 1b—Supported
1	RO	Interface Error Reporting Support 0b—Not Supported 1b—Supported
2	RO	Interface Error Logging Support 0b—Not Supported 1b—Supported
3	RO	Transient Error Threshold Support 0b—Not Supported 1b—Supported
4	RO	<i>I-Error Fault Injection</i> Support—indicates if the interface supports fault injection 0b—Not Supported 1b—Supported
5	RO	LPRT Wildcard Packet Relay Support—See <i>Component Switch Structure</i> Applicable only to switch interfaces. 0b—Not Supported 1b—Supported
6	RO	MPRT Wildcard Packet Relay Support—See <i>Component Switch Structure</i> Applicable to only switch interfaces. 0b—Not Supported 1b—Supported
7	RO	Interface Loopback Support—Indicates if the interface supports packet loopback. If supported, if the packet's DCID and SCID fields are identical, then the packet is not transmitted on the link. Instead, the packet is copied using implementation-specific means from the transmit queue to the receive queue. 0b—Not Supported 1b—Supported
8	RO	Implicit Flow Control Support 0b—Unsupported. Interface shall support <i>Explicit Flow Control</i>

Bit Location	Access	Description
9		1b—Supported. Interface shall support the P2P-Core OpClass, and may support the P2P-Vendor-defined OpClass.
9	RO	Explicit Flow Control Support 0b—Unsupported. Interface shall support <i>Implicit Flow Control</i> . 1b—Supported. Interface may support any OpClass.
10	RO	P2P-Core Support—Indicates if the interface can be configured to support the P2P-Core OpClass. Applicable to only non-switch component types. 0b—Unsupported 1b—Supported
11	RO	P2P-Coherency Support—Indicates if the interface can be configured to support the P2P-Coherency OpClass. Applicable to only non-switch component types. 0b—Unsupported 1b—Supported
12	RO	P2P Vendor-defined Support—Indicates if the interface can be configured to support the P2P Vendor-defined OpClass. Applicable to only non-switch component types. 0b—Unsupported 1b—Supported
13	RO	Daisy-Chain Support—Indicates if an interface can be used to forward P2P-Core OpClass packets between two directly-attached non-switch components 0b—Unsupported 1b—Supported
14	RO	Peer-C-Reset Support—indicates if the interface supports <i>Link CTL</i> packets used to initiate a <i>Full Component Reset</i> 0b—Unsupported 1b—Supported
15	RO	Peer C-Up Transition Support—Indicates if the interface supports <i>Link CTL</i> packets used to transition a component to C-Up 0b—Unsupported 1b—Supported
16	RO	Packet Relay Access Key Field Provisioned—Indicates if the Interface structure contains the optional Packet Relay and Access Key fields. 0b—Not Provisioned 1b—Provisioned
17	RO	Interface Access Key Validation Support—Indicates if the interface supports Access Key validation on all end-to-end packets received and transmitted. If supported, then the Packet Relay and Access Key fields shall be provisioned. 0b—Unsupported

Bit Location	Access	Description
18		1b—Supported
	RO	<i>Link-level Reliability (LLR) Support</i> —Indicates if the interface supports LLR
19		0b—Unsupported 1b—Supported
	RO	TR Interface Support—Indicates if the interface supports <i>Transparent Router (TR)</i> packet relay
20		0b—Unsupported 1b—Supported
	RO	Source CID Packet Validation Support—If the interface supports packet relay, then this field indicates if the interface supports source CID validation, i.e., validates the packet's SCID == Peer CID.
21		0b—Unsupported 1b—Supported
	RO	Source SID Packet Validation Support—If the interface supports multi-subnet packet relay, then this field indicates if the interface supports source SID validation, i.e., validates the packet's SSID == Peer SID.
22		0b—Unsupported 1b—Supported
	RO	Adaptive FC Credit Support—Indicates if the interface supports adaptive flow-control credit management (see <i>Explicit Flow Control</i> ).
23		0b—Unsupported 1b—Supported
	RO	PCO Communications Support—Indicates if the interface supports the semantics required to support <i>PCIe-Compatible Ordering (PCO)</i> solutions.
24		0b—Unsupported 1b—Supported
	RO	Packet Relay-Access Key Field Presence—Indicates if the packet relay and Access Key fields are present within the structure.
25		0b—Not present 1b—Present
	RO	Aggregated Interface Support—Indicates if the interface can operate only as an independent interface or can operate as either an independent or aggregated interface.
26		0b—Non-aggregated Interface 1b—Non-aggregated or Aggregated Interface
	RO	Aggregated Interface Role—If Aggregated Interface Support == 1b, then this bit Indicates if the interface can be configured to operate as a SAI or a NAI.
		0b—Only NAI 1b—SAI or NAI

Bit Location	Access	Description
27	RO	PeerNonceValidationSupport—Indicates if the interface supports PeerNonceValidation as described in <i>Mitigating In Situ Insertion</i> . 0b—Unsupported 1b—Supported
28	RO	RequireP2PStandaloneAcknowledgmentRequired—Indicates if <i>Interface I-CAP 1 Omit P2P Standalone Acknowledgment</i> is hardwired to 0b. Only interfaces that support the P2P-Core OpClass, P2P-Coherency OpClass, or P2P-Vendor-defined OpClass may support this capability, else this bit shall be Reserved. 0b—Configurable 1b—Hardwired
29	RO	InterfaceGroupSupport—Indicates if the interface is a member of an Interface Group. 0b—Unsupported 1b—Supported
30	RO	RequiresInterfaceGroupSingleOpClass—If an interface supports multiple OpClasses, and the interface is a member of an Interface Group, then this bit indicates if the implementation requires that only one OpClass be enabled across all interfaces within the same Interface Group. 0b—Not Required 1b—Required  <b>Developer Note:</b> Due to the imposed operational constraints, developers are strongly encouraged to evaluate the component's target solution operational needs before deciding to support this capability.
31	-	RsvdZ

#### 8.16.4. Interface I-CAP 1 Control

Table 8-20: I-Cap 1 Control Field

Bit Location	Access	Description
0	RW	InterfaceContainmentControl 0b—Exit Interface Containment. Has no impact if not in Interface Containment. 1b—Trigger <i>Interface Containment</i> independent of the <i>I-Error Trigger</i> field settings. Only valid if Interface Containment is supported.
1	RW	TransientErrorThresholdControl 0b—Disable (default).

Bit Location	Access	Description
2	RW	<p>1b—Enable Transient Error Threshold Operation</p> <p><i>I-Error Fault Injection Control</i></p> <p>0b—Disable I-Error Fault Injection</p> <p>1b—Enable I-Error Fault Injection</p>
3	RW	<p>Wildcard Packet Relay Control—See <i>Component Switch Structure</i></p> <p>0b—Disable</p> <p>1b—Enable</p> <p>Shall be applicable only to interfaces that support non-daisy-chain packet relay.</p>
4	RW	<p>Enable Interface Loopback</p> <p>0b—Disable (default)</p> <p>1b—Enable</p>
5	RWS	<p>Flow-Control Type</p> <p>0b—Interface shall use <i>Explicit Flow Control</i></p> <p>1b—Interface shall use <i>Implicit Flow Control</i>. If configured to use implicit flow-control and a Requester egress interface, then the Max Implicit FC Credits field shall be configured to a non-zero value prior to enabling the interface.</p> <p>If the interface supports a single flow-control type, then this field shall be hardwired to that type.</p> <p>If this field is modified, then the change will take effect upon the next <i>Warm Interface Reset</i>.</p>
8:6	RWS	<p>OpClass Select—Determines if explicit OpClasses, P2P-Core, P2P-Coherency, or P2P Vendor-defined end-to-end packets may be exchanged on this interface.</p> <p>0x0—Interface shall be used only to exchange explicit OpClass packets.</p> <p>0x1—Interface shall be used only to exchange P2P-Core packets.</p> <p>0x2—Interface shall be used only to exchange P2P-Coherency packets.</p> <p>0x3—Interface shall be used only to exchange P2P Vendor-defined packets</p> <p>0x4-0x7—Reserved</p> <p>If this field is modified, then the change will take effect upon the next <i>Warm Interface Reset</i>.</p>
9	RW	<p>Control OpClass Packet Filtering Enable—Used to prevent a component from communicating with non-management components, e.g., while the component is in the C-CFG state. While enabled, the interface shall silently discard any non-Control OpClass packet received or scheduled for transmission on this interface.</p> <p>Shall be applicable only to components that support the <i>Component Switch Structure</i>. If the <i>Component Switch Structure</i> is unsupported, then this bit shall be Reserved.</p>

Bit Location	Access	Description
10		<p>0b—Disabled 1b—Enabled</p>
	RW	<p>Control Write MSG Packet Filtering Enable—Used to enable fabric managers of two independent connected switch topologies to communicate with one another using directed <i>Control Write MSG</i> request packets. While enabled:</p> <ul style="list-style-type: none"> <li>• The interface shall silently discard any non-directed <i>Control Write MSG</i> request packet received or scheduled for transmission on this interface.</li> <li>• Upon receipt, a switch shall not validate the packet's SCID, DCID, and, if applicable, SSID and DSID fields.</li> </ul> <p>Shall be applicable only to components that support the <i>Component Switch Structure</i>. If the <i>Component Switch Structure</i> is unsupported, then this bit shall be Reserved.</p> <p>0b—Disabled 1b—Enabled</p>
11	RW	<p>LPRT Enable—if a LPRT is provisioned, then this bit determines if the LPRT has been configured and enables / disables single-subnet unicast packet relay (see <i>Switch Packet Processing</i>). If a LPRT is not provisioned, then this bit shall be Reserved.</p> <p>This bit shall be applicable only to interfaces that support single-subnet unicast packet relay. If the interface does not support single-subnet unicast packet relay, then this bit shall be hardwired to 0b.</p> <p>0b—Disabled 1b—Enabled</p>
12	RW	<p>MPRT Enable—if a MPRT is provisioned, then this bit determines if the MPRT has been configured and enables / disables multi-subnet unicast packet relay (see <i>Switch Packet Processing</i>).</p> <p>This bit shall be applicable only to interfaces that support multi-subnet packet relay. If the interface does not support multi-subnet packet relay, then this bit shall be hardwired to 0b.</p> <p>0b—Disabled 1b—Enabled</p>
13	RW	<p>Daisy-chain Peer Interface Configured—Determines if the Peer Daisy-chain Interface ID field has been configured. If this is the last component in a daisy-chain, then this field shall be set to 0b.</p> <p>0b—Not Configured 1b—Configured</p>
14	RW	<p><i>Link-level Reliability (LLR)</i> CRC Trigger—if the interface supports LLR, then this field indicates whether LLR recovery is initiated only upon detecting a PCRC error or upon detecting a PCRC or ECRC error.</p> <p>If the interface is configured to support the P2P-Core, P2P-Coherency, or the P2P-Vendor-defined OpClass, then software shall set this bit to 0b. If the</p>

Bit Location	Access	Description
		interface cannot be used to exchange explicit OpClass packets, then this bit shall be hardwired to 0b.  0b—PCRC Error or ECRC Error 1b—Only PCRC Error  If LLR is unsupported, then this bit shall be Reserved.
15	RW	Source CID Packet Validation Enable—if supported, then validation shall be enabled only if the peer component is a Requester or a Responder (see Peer Base C-Class field). Further, validation shall be explicitly enabled only after the Peer CID field is configured by software or after a successful Peer-Attribute <i>Link CTL</i> exchange.  0b—Disabled 1b—Enabled
16	RW	Source SID Packet Validation Enable—if supported, then validation shall be explicitly enabled only after the Peer SID field is configured by software or after a successful Peer-Attribute <i>Link CTL</i> -Link ACK <i>Link CTL</i> exchange.  0b—Disabled 1b—Enabled
17	RW	Peer CID Configured—Determines if the Peer CID field contains a software-configured value. If configured, then a successful Peer-Attribute <i>Link CTL</i> -Link ACK <i>Link CTL</i> exchange shall not update the Peer CID field.  0b—Not Configured 1b—Configured
18	RW	Peer SID Configured—Determines if the Peer SID field contains a software-configured value. If configured, then a successful Peer-Attribute <i>Link CTL</i> -Link ACK <i>Link CTL</i> exchange shall not update the Peer SID field.  0b—Not Configured 1b—Configured
19	RWS	Adaptive FC Credit Enable—if supported, then this field is used to enable Adaptive FC Credits. By default, if supported, then this field should be set to Enabled.  0b—Disabled 1b—Enabled  If unsupported, then this field shall be hardwired to 0b.  If this field is modified, then the change will take effect upon the next <i>Warm Interface Reset</i> .
20	RW	Omit P2P <i>Standalone Acknowledgment</i> —If the interface supports and has enabled the P2P-Core, P2P-Coherency, or P2P-Vendor-defined OpClass, then this bit determines if a Responder is to omit, i.e., not transmit a <i>Standalone Acknowledgment</i> in response to specific successfully-executed request packets. Further, this bit determines if the Requester is to await receipt of a

Bit Location	Access	Description
		<p><i>Standalone Acknowledgment</i> corresponding to any Outstanding Request Packet of these specific request packets on this interface. In general, the specific request packets governed by this bit are reliable, non-persistent Data Space writes and interrupt request packets (the corresponding request packet specification text explicitly states Omit P2P <i>Standalone Acknowledgment</i> semantics).</p> <p>If Omit P2P <i>Standalone Acknowledgment</i> == 0b, then:</p> <ul style="list-style-type: none"> <li>• When applicable, a Responder shall always transmit a <i>Standalone Acknowledgment</i>.</li> <li>• A Requester shall await receipt of a <i>Standalone Acknowledgment</i> before making ordering decisions that depend upon the completion of the Outstanding Request Packet.</li> </ul> <p>If Omit P2P <i>Standalone Acknowledgment</i> == 1b, then:</p> <ul style="list-style-type: none"> <li>• A Responder shall execute all request packets in the order that they arrived on the receiving interface. A Responder may execute request packets that arrive on different receiving interfaces in any order.</li> <li>• A Responder shall make modified data visible to all subsequently received any type of read or read-modify-write (e.g., Atomics) request packets based on the order that they arrived on the receiving interface. For example, if a read request packet targets the same addressed data as a previously received a write request packet received on the same interface, then the write request packet's payload shall be visible to the subsequently received read request packet.</li> <li>• A Responder shall transmit a <i>Standalone Acknowledgment</i> for request packets of a type whose behavior is not governed by this bit, but shall not do so for request packets who behavior is governed by this bit.</li> <li>• After transmitting a request packet of a type whose behavior is governed by this bit, a Requester is not required to await receipt of a <i>Standalone Acknowledgment</i> before making ordering decisions that depend upon the completion of the request packet.</li> </ul> <p>The Omit P2P <i>Standalone Acknowledgment</i> bit shall be set to the same value in all interfaces participating in a given daisy-chain topology.</p> <p>If a daisy-chain contains two Requesters (one at each end), then management shall set Omit P2P <i>Standalone Acknowledgment</i> = 0b in all interfaces participating in the daisy-chain topology.</p> <p>If an interface supports only explicit OpClass packets, then this bit shall be hardwired to 0b.</p> <p>If an interface is enabled to support explicit OpClass packets, then this bit shall be set to 0b.</p>

Bit Location	Access	Description
		If <i>Interface I-CAP 1 P2P Standalone Acknowledgment Required</i> == 1b, then this bit shall be hardwired to 0b. 0b—Do not Omit 1b—Omit
21	RW	Interface-Component Containment Enable—Determines if the component is to initiate <i>Component Containment</i> once the interface has triggered <i>Interface Containment</i> due to an Unexpected Physical Layer Failure—PHY-Down trigger. 0b—Disabled 1b—Enabled
22	RW	PeerNonce Validation Enable—Determines if the PeerNonce field has been configured, and causes the interface to initiate a Nonce <i>Link CTL</i> exchange. If the PeerNonce field is modified, then software shall re-enable this bit. See <i>Mitigating In Situ Insertion</i> . 0b—Disabled 1b—Enabled
23	RW	Auto Peer-Attribute <i>Link CTL</i> Exchange—Determines if the interface is to automatically perform a Peer-Attribute <i>Link CTL</i> -Link ACK <i>Link CTL</i> exchange when the link transitions from L-Down to L-Up. See <i>Link States</i> for any steps that need to be completed prior to performing this exchange. If the interface may be used to exchange <i>In-band Management</i> packets or if HVS > 0, then this bit shall be hardwired to 1b. 0b—Disabled 1b—Enabled
24	RWS	<i>Link-Level Reliability (LLR)</i> Enable—Determines if LLR is to be used on the interface. If the interface is configured to use the P2P-Core OpClass or the P2P-Coherency OpClass, then LLR shall be enabled. 0b—Disabled 1b—Enabled If unsupported, then this field shall be hardwired to 0b. If this field is modified, then the change will take effect upon the next <i>Warm Interface Reset</i> .
26:25	RWS	Aggregated Interface Control—Determines if the interface is configured to operate as an independent or as an aggregated interface. 0x0—Independent, non-aggregated Interface 0x1—NAI 0x2—SAI 0x3—Reserved
31:27	-	Reserved

## 8.16.5. Interface I-CAP 2

Table 8-21: I-Cap 2 Field

Bit Location	Access	Description
Bits 31:0	-	Reserved

## 8.16.6. Interface I-CAP 2 Control

Table 8-22: I-Cap 2 Control Field

Bit Location	Access	Description
0	RWS	Software Defined I-Bit 0  Software-defined and managed sticky-bit. The purpose and use of this bit is outside of this specification's scope. If a component supports sticky bits, then it shall maintain the setting of this bit across reset events as long as it has sufficient power.
		Software Defined I-Bit 1  Software-defined and managed sticky-bit. The purpose and use of this bit is outside of this specification's scope. If a component supports sticky bits, then it shall maintain the setting of this bit across reset events as long as it has sufficient power.
Bits 31:2	-	RsvdP

## 8.16.7. Interface Error Fields

An interface may support the I-Error Status, I-Error Detect, I-Error Trigger, and I-Error Fault Injection fields.

- Each error Bit<sub>k</sub> shall be implemented as a set across the I-Error Status, I-Error Detect, and I-Error Trigger fields, i.e., Bit<sub>k</sub> shall be implemented in all fields or not at all. I-Error Fault Injection bits may be implemented.
- Each unsupported Bit<sub>k</sub> shall be hardwired to 0b.
- If Bit<sub>k</sub> is supported, then the default Bit<sub>k</sub> value shall be 0b.
- If a component supports the *Component Error and Signal Event Structure* and the I-Signal Target corresponding to the detected error is non-zero, then the interface shall log the error as specified in *Component Error Log – Component Interface Error Format Fields*. Errors triggered through I-Error Fault Injection shall be logged as an Interface Fault Injection log record type.

### 8.16.7.1. I-Error Status

The I-Error Status field shall be as specified in *I-Error Status*. For each I-Error Status Bit<sub>k</sub> = 1b, the corresponding error condition has been detected. Bit<sub>k</sub> shall be updated only if the corresponding *I-Error Detect* Bit<sub>k</sub> == 1b. Software clears Bit<sub>k</sub> by writing 1b.

Table 8-23: I-Error Status

Bit Location	Access	Description
0	RW1CS	Excessive Physical Layer Retraining Events Recorded
	RW1CS	Non-transient Link Error Recorded
	RW1CS	<i>Interface Containment</i> Recorded
	RW1CS	Interface Access Key Violation Recorded
	RW1CS	Interface FC FWD Progress Violation Recorded
	RW1CS	Unexpected Physical Layer Failure—PHY-Down Recorded
	RW1CS	Unexpected Degraded Link Performance Recorded
	RW1CS	Interface AE Recorded
	RW1CS	Switch Packet Relay Failure Recorded
15:9	-	RsvdZ

### 8.16.7.2. I-Error Detect

The I-Error Detect field shall be as specified in *I-Error Detect*. For each I-Error Detect Bit<sub>k</sub> =1b, when the corresponding error condition is detected, the interface shall set *I-Error Status* Bit<sub>k</sub> = 1b.

Table 8-24: I-Error Detect

Bit Location	Access	Description
0	RW	Excessive Physical Layer Retraining Events Detect
	RW	Non-transient Link Error Detect—This covers non-transient link protocol errors, transition to L-Down due to a failure event, e.g., hardware failure, unexpected transition to PHY-Down, etc.
	RW	<i>Interface Containment</i> Detect
	RW	Interface Access Key Violation Detect
	RW	Interface FC FWD Progress Violation Detect
	RW	Unexpected Physical Layer Failure—PHY-Down Detect—This covers unplanned interface and link failure events.
	RW	Unexpected Degraded Link Performance Detect—This covers unplanned link performance reduction events including, but not limited to: lane failure, physical layer signaling rate reduction, etc.

7	RW	Interface AE Detect—This covers packet filtering violations (e.g., Control OpClass packets, packets with an invalid CID and / or SID, etc.), Peer Nonce mismatch, invalid <i>Directed Control Space Packet Relay</i> manager, etc.
8	RW	Switch Packet Relay Failure Detect—This covers packet relay issues, e.g., detection of a UP error, invalid DR Interface, etc. Applicable only to interfaces that support packet relay.
15:9	-	RsvdP

### 8.16.7.3. I-Error Trigger

The I-Error Trigger field shall be as specified in *I-Error Trigger*. For each I-Error Trigger Bit<sub>k</sub> = 1b, if the corresponding error is detected, the component shall trigger *Interface Containment*.

Table 8-25: I-Error Trigger

Bit Location	Access	Description
0	RW	Excessive Physical Layer Retraining Events Trigger
1	RW	Non-transient Link Error Trigger
2	RW	<p><i>Interface Containment</i> Trigger</p> <p>Writing 0b to this bit shall disarm <i>Interface Containment</i>.</p> <p>Writing 1b to this bit shall clear existing <i>Interface Containment</i>, and shall enable and arm <i>Interface Containment</i>.</p> <p>Reads of this bit shall return 0b.</p>
3	RW	Interface Access Key Violation Trigger
4	RW	Interface FC FWD Progress Violation Trigger
5	RW	Unexpected Physical Layer Failure—PHY-Down Trigger
6	RW	Unexpected Degraded Link Performance Trigger
7	RW	Interface AE Trigger
8	RW	Switch Packet Relay Failure Trigger
15:9	-	RsvdP

### 8.16.7.4. I-Error Fault Injection

- 5 The I-Error Fault Injection field shall be as specified in *I-Error Fault Injection*. If I-Error Fault Injection Bit<sub>k</sub> == 1b and the corresponding *I-Error Detect* Bit<sub>k</sub> == 1b, then the interface shall initiate the corresponding error processing as though an error was detected.

Table 8-26: I-Error Fault Injection

Bit Location	Access	Description
0	WO	Test Excessive Physical Layer Retraining Events Error
	WO	Test Non-transient Link Error
	WO	Test <i>Interface Containment</i>
	WO	Test Interface Access Key Violation Error
	WO	Test Interface FC FWD Progress Violation Error
	WO	Test Unexpected Physical Layer Failure—PHY-Down
	WO	Test Unexpected Degraded Link Performance
	WO	Test Interface AE
	WO	Test Switch Packet Relay Failure
15:9	-	RsvdP

## 8.17. Interface PHY Structure

The Interface PHY Structure contains two parts. The first part contains physical layer-independent fields and the second part contains physical layer-specific fields (specified in the corresponding *Gen-Z Physical Specification*).

- 5 The following are the features and requirements associated with the Interface PHY Structure:
- At least one Interface PHY structure shall be provisioned per component interface.
  - If an interface supports multiple physical layers, then an additional Interface PHY structure shall be provisioned for each additional supported physical layer associated with the interface. These structures shall be located using the Next Interface PHY PTR field.
  - The Interface PHY structure shall use the *Interface PHY Structure Format*.
- 10



Figure 8-13: Interface PHY Structure Format

Table 8-27: Interface PHY Structure Common Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>Version (Vers)</b>	4	0x1	M	RO	Structure Version
<b>PHY Type</b>	8	-	M	RO	Indicates the physical layer type associated with this Interface PHY structure. 0x0—PHY Clause 5: 25G Fabric 0x1—PHY Clause 6: 25G Local 0x2—PHY Clause 7: PCIe through 16 GT/s electrical, logical, and LTSSM 0x3-0xFF—Reserved

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
TxDLY	4	-	M	RW	Tx Request to Valid Delay—Number of core clock cycles of delay between PHY_TXDATAREQ and PHY_TXVALID (See <i>Physical Layer Abstraction</i> )
Next Interface PHY PTR	32	-	O	RO	If an interface supports multiple physical layers, then this points to the next Interface PHY structure that describes that physical layer.
					If the interface does not support multiple physical layers, then this field shall be Null.
VD PTR	32	-	O	RO	If supported, then this points to a vendor-defined structure pointer, else shall be Null.
PHY Status	32	-	M	-	See <i>PHY Status</i>
PHY Control	32	-	M	-	See <i>PHY Control</i>
PHY CAP 1	32	-	M	-	See <i>PHY CAP 1</i>
PHY CAP 1 Control	32	-	M	-	See <i>PHY CAP 1 Control</i>
PHY Events	32	-	M	-	See <i>PHY Events</i>
PHY Lane Status	32	-	M	-	See <i>PHY Lane Status</i>
PHY Lane Control	32	-	M	-	See <i>PHY Lane Control</i>
PHY Lane CAP	32	-	M	-	See <i>PHY Lane CAP</i>
PHY Remote Lane CAP	32	-	M	-	See <i>PHY Remote Lane CAP</i>
PHY LP Control	32	-	M	-	See <i>PHY Low Power Control</i>
PHY LP Timing CAP	32	-	M	-	See <i>PHY Low Power Timing Capability</i>
PHY LP CAP	32	-	M	-	See <i>PHY Low Power CAP</i>
PHY Remote LP CAP	32	-	M	-	See <i>PHY Remote Low Power CAP</i>
PHY UP LP Control	32	-	M	-	See <i>PHY Up Low Power Control</i>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
PHY UP LP Timing CAP	32	-	M	-	See <i>PHY Up Low Power Timing Capability</i>
PHY UP LP CAP	32	-	M	-	See <i>PHY Up Lower Power CAP</i>
PHY Remote UP LP CAP	32	-	M	-	See <i>PHY Up Remote Lower Power CAP</i>
PHY Extended Status	32	-	M	RO	Physical layer extended features status—This field is outside the scope of this specification. See <i>Gen-Z Physical Specification</i>
PHY Extended Control	32	-	M	RW	Physical layer extended features controls—This field is outside the scope of this specification. See <i>Gen-Z Physical Specification</i>
PHY Extended CAP	32	-	M	RO	Physical layer extended features capability—This field is outside the scope of this specification. See <i>Gen-Z Physical Specification</i>
PHY Remote Extended Cap	32	-	M	RW	Peer interface physical layer extended features capability—This field is outside the scope of this specification. See <i>Gen-Z Physical Specification</i>
R0	20	-	-	-	RsvdP
R1	32	-	-	-	Reserved
R2	64	-	-	-	Reserved

Table 8-28: PHY Status

Bit Location	Access	Description
3:0	RO	<p>Physical Layer Operational Status</p> <p>0x0—PHY-Down, uninitialized state</p> <p>0x1—PHY-Up</p> <p>0x2—PHY-Down-Retrain</p> <p>0x3—PHY-Up Low-Power 1</p> <p>0x4—PHY-Up Low-Power 2</p> <p>0x5—PHY-Up Low-Power 3</p> <p>0x6—PHY-Up Low-Power 4</p> <p>0x7—PHY Low-Power 1</p>

Bit Location	Access	Description
		0x8—PHY Low-Power 2 0x9—PHY Low-Power 3 0xA—PHY Low-Power 4 0xB—0xF Reserved
7:4	RO	Previous Physical Layer Operational Status  0x0—PHY-Down, uninitialized state 0x1—PHY-Up 0x2—PHY-Down-Retrain 0x3—PHY-Up Low-Power 1 0x4—PHY-Up Low-Power 2 0x5—PHY-Up Low-Power 3 0x6—PHY-Up Low-Power 4 0x7—PHY Low-Power 1 0x8—PHY Low-Power 2 0x9—PHY Low-Power 3 0xA—PHY Low-Power 4 0xB—0xF Reserved
9:8	RO	Physical Layer Training Status  0x0—Training has not occurred 0x1—Training succeeded 0x2—Training has failed 0x3—Reserved
11:10	RO	Physical Layer Retraining Status  0x0—Retraining has not occurred 0x1—Retraining succeeded 0x2—Retraining failed 0x3—Reserved
12	RO	PHY Tx Link Width Reduced  0b—Nominal Tx Link Width 1b—Reduced Tx Link Width
13	RO	PHY Rx Link Width Reduced  0b—Nominal Rx Link Width 1b—Reduced Rx Link Width
14	RO	PHY Tx Error Detected  0b—No Tx errors detected 1b—Tx error detected
15	RO	PHY Rx Error Detected  0b—No Rx errors detected 1b—Rx error detected
31:16	-	RsvdZ

Table 8-29: PHY Control

Bit Location	Access	Description
0	RW	<p>Enable Physical Layer—If not enabled, then enable the physical layer associated with this structure and initiate physical layer training.</p> <p>If an interface supports multiple Interface PHY structures, then this field shall be enabled only in one Interface PHY structure at any given time. If this field is enabled in multiple Interface PHY structures, then the behavior may be non-deterministic.</p> <p>1b—Enable</p> <p>Reads of this field shall return 0b.</p>
1	RW	<p>Disable Physical Layer</p> <p>1b—Disable and transition to PHY-Down.</p> <p>Reads of this field shall return 0b.</p>
2	RW	<p>Disable Physical Layer Auto-retraining – Prevents physical layer from autonomously retraining based on physical layer errors. Physical layer retraining then becomes the responsibility of the Core.</p> <p>1b—Disable autonomous physical layer retraining</p> <p>Reads of this field shall return 0b.</p>
3	RW	<p>Exit Low-power State—Instructs the physical layer to exit any low-power state it may be in and return to normal operation.</p> <p>1b—Exit Low Power</p> <p>Reads of this field shall return 0b.</p>
4	RW	<p>Clear PHY Events—Instructs the physical layer to reset the PHY Events Interface Structure</p> <p>1b—Reset</p> <p>Reads of this field shall return 0b.</p>
6:5	RW	<p>Physical Layer Retraining Level—Retraining level to initiate whenever an <i>Interface I-Control</i> Force Physical Layer Abstraction Retraining = 1b or <i>PHY Control</i> Force Physical Layer Retraining = 1b.</p> <p>Level 1-4 Retraining bypasses specific physical layer training states to decrease time for non-error-based retraining control transitions. See the <i>Gen-Z Physical Specification</i> for additional details.</p> <p>0x0—PHY Full Retraining (Level 1 – Default)      0x1—PHY Retraining Level 2      0x2—PHY Retraining Level 3      0x3—PHY Retraining Level 4</p>

Bit Location	Access	Description
7	RW	Force Physical Layer Retraining—Instructs the physical layer to initiate retraining based on the <i>PHY Control</i> Physical Layer Retraining Level. 1b—Initiate retraining Reads of this field shall return 0b.
31:8	-	RsvdP

Table 8-30: PHY CAP 1

Bit Location	Access	Description
0	RO	Default PHY—if an interface supports multiple Interface PHY structures, then this field indicates the Interface PHY structure to use absent explicit configuration. Only one Interface PHY structure shall indicate it is the Default PHY. If an interface supports a single Interface PHY structure, then this field shall be hardwired to 1b. 0b—Non-Default 1b—Default
1	RO	Auto-Train Support—Indicates if the physical layer associated with this Interface PHY structure automatically performs physical layer training whenever initialized. 0b—Unsupported 1b—Supported
2	RO	Level 1 Retrain Support—Level 1 Retraining bypasses specific physical layer training states to decrease time for non-error-based retraining control transitions. The <i>Gen-Z Physical Specification</i> further defines the requirements and capabilities of this feature. 0b—Unsupported 1b—Supported
3	RO	Level 2 Retrain Support—Level 2 Retraining bypasses specific physical layer training states to decrease time for non-error-based retraining control transitions. The <i>Gen-Z Physical Specification</i> further defines the requirements and capabilities of this feature. 0b—Unsupported 1b—Supported
4	RO	Level 3 Retrain Support—Level 3 Retraining bypasses specific physical layer training states to decrease time for non-error-based retraining control transitions. The <i>Gen-Z Physical Specification</i> further defines the requirements and capabilities of this feature. 0b—Unsupported 1b—Supported

5	RO	Level 4 Retrain Support—Level 4 Retraining bypasses specific physical layer training states to decrease time for non-error-based retraining control transitions. The <i>Gen-Z Physical Specification</i> further defines the requirements and capabilities of this feature.  0b—Unsupported 1b—Supported
9:6	RO	Retrain Level 1 worst-case, maximum time to execute physical layer retraining See <i>PHY Worst-Case Retraining Time Encodings</i>
13:10	RO	Retrain Level 2 worst-case, maximum time to execute physical layer retraining See <i>PHY Worst-Case Retraining Time Encodings</i>
17:14	RO	Retrain Level 3 worst-case, maximum time to execute physical layer retraining See <i>PHY Worst-Case Retraining Time Encodings</i>
21:18	RO	Retrain Level 4 worst-case, maximum time to execute physical layer retraining See <i>PHY Worst-Case Retraining Time Encodings</i>
31:22	-	Reserved

Table 8-31: PHY CAP 1 Control

Bit Location	Access	Description
0	RW	Flit Encoding Enable—Determines if flit encoding is enabled. See <i>Physical Layer Abstraction</i> .  The default setting for this field is defined within each unique <i>Gen-Z Physical Specification</i> Clause.  1b—Enable Flit CRC  Reads of this bit shall return 0b.
1	RW	Extended Feature Enable—if an interface supports extended features (indicated by a non-zero PHY Extended CAP field), then this field determines if the interface is enabled to use these extended features. Extended features are outside of the scope of the <i>Gen-Z Physical Specification</i> . If enabled, then the PHY Extended Status field shall reflect the status of the extended features.  0b—Disable 1b—Enable
5:2	RWS	Aggregation Interface Number—Determines the number of aggregated interfaces  0x0—0 0x1—2 0x2—4 0x3—8 0x4—16 0x5—32

31:6	0x6—64	
	0x7—128	
	0x8—256	
	0x9-0xF—Reserved	
-	Reserved	

Table 8-32: PHY Entry and Exit Latency Encodings

Encoding	Latency	Encoding	Latency
0x0	10 ns	0x8	5 $\mu$ s
0x1	25 ns	0x9	10 $\mu$ s
0x2	50 ns	0xA	25 $\mu$ s
0x3	100 ns	0xB	50 $\mu$ s
0x4	250 ns	0xC	100 $\mu$ s
0x5	500 ns	0xD	250 $\mu$ s
0x6	1 $\mu$ s	0xE	500 $\mu$ s
0x7	2 $\mu$ s	0xF	1000 $\mu$ s

Table 8-33: PHY Worst-Case Retraining Time Encodings

Encoding	Latency	Encoding	Latency
0x0	50 ns	0x8	50 $\mu$ s
0x1	100 ns	0x9	100 $\mu$ s
0x2	250 ns	0xA	500 $\mu$ s
0x3	500 ns	0xB	1 ms
0x4	1 us	0xC	10 ms
0x5	5 us	0xD	50 ms
0x6	10 $\mu$ s	0xE	100 ms
0x7	25 $\mu$ s	0xF	500 ms

Table 8-34: PHY Events

Bit Location	Access	Description
<b>15:0</b>	RO	PHY Low Power Events—This counter shall be incremented by one (modulo $2^{16}$ ) whenever a physical layer low power state is entered. Event counts are reset using the Clear PHY Events field in the PHY Control Interface Structure.
<b>31:16</b>	RO	PHY Retraining Events—This counter shall be incremented by one (modulo $2^{16}$ ) whenever physical layer retraining is invoked due to an event independent of interface initialization and reset events. For example, whenever the link initiates physical layer retraining due to a failure to transmit flow-control credits, software forces retraining, <i>Link Resynchronization</i> fails, etc. Event counts are reset using the Clear PHY Events field in the PHY Control Interface Structure.

Table 8-35: PHY Lane Status

Bit Location	Access	Description
<b>0</b>	RO	PHY Tx Link Width Reduced 0b—Nominal Tx Link Width 1b—Reduced Tx Link Width
<b>1</b>	RO	PHY Rx Link Width Reduced 0b—Nominal Rx Link Width 1b—Reduced Rx Link Width
<b>3:2</b>	RO	Asymmetric Lane Status—Indicates if the physical layer status for transmit and receive lanes 0x0—Symmetric 0x1—Asymmetric 0x2-0x3—Reserved
<b>4</b>	RO	TX Reversal Status 0b—TX lanes not reversed 1b—TX lanes reversed
<b>5</b>	RO	RX Reversal Status 0b—RX lanes not reversed 1b—RX lanes reversed
<b>7:6</b>	-	Reserved
<b>19:8</b>	RO	If a SAI and in PHY UP, then this indicates the current number of Tx Lanes in the aggregated interface. If a NAI and in PHY UP, then this field shall be set to 0x0. If the interface is not aggregated and in PHY UP, then this indicates the current number of Tx Lanes.

31:20	RO	If a SAI and in PHY UP, then this indicates the current number of Rx Lanes in the aggregated interface. If a NAI and in PHY UP, then this field shall be set to 0x0.  If the interface is not aggregated and in PHY UP, then this indicates the current number of Rx Lanes.
-------	----	---

Table 8-36: PHY Lane Control

Bit Location	Access	Description
1:0	RW	Enable Lane Asymmetry—If the interface does not support an asymmetric number of transmit and receives lanes, then this field shall be hardwired to 0x0.  0x0—Symmetric Enabled (fixed number of transmit and receives lanes) 0x1—Static Asymmetric Enabled (requires physical layer re-initialization) 0x2—Dynamic Asymmetric Enabled (physical layer automatically reduces the number of transmit or receive lanes based on the system connectivity or lane faults) 0x3—Reserved
3:2	RW	Enable TX Reversal – If the interface does not support lane reversal, then this field shall be hardwired to 0x0. Prior to setting this bit to 1b, management needs to comprehend the impact if asymmetry is supported.  0x0—TX reversal disabled 0x1—Dynamic TX reversal enabled 0x2—Static TX reversal enabled 0x3—Reserved
5:4	RW	Enable RX Reversal – If the interface does not support lane reversal, then this field shall be hardwired to 0x0. Prior to setting this bit to 1b, management needs to comprehend the impact if asymmetry is supported.  0x0—RX reversal disabled 0x1—Dynamic RX reversal enabled 0x2—Static RX reversal enabled 0x3—Reserved
7:6	-	RsvdP
19:8	RW	Tx Lane Enable  If a SAI and in PHY UP, then this determines the number of enabled Tx Lanes in the aggregated interface. If a NAI and in PHY UP, then this field shall be set to 0x0.  If the interface is not aggregated and in PHY UP, then this determines the number of enabled Tx Lanes.

<b>31:20</b>	RW	<p><b>Rx Lane Enable</b></p> <p>If a SAI and in PHY UP, then this determines the number of enabled Rx Lanes in the aggregated interface. If a NAI and in PHY UP, then this field shall be set to 0x0.</p> <p>If the interface is not aggregated and in PHY UP, then this determines the number of enabled Rx Lanes.</p>
--------------	----	---

Table 8-37: PHY Lane CAP

Bit Location	Access	Description
1:0	RO	<p>Asymmetric Lane Support—Indicates if the physical layer supports configuring a different number of transmit and receive lanes</p> <p>0x0—Symmetric (fixed number of transmit and receives lanes)</p> <p>0x1—Static Asymmetric (requires physical layer re-initialization)</p> <p>0x2—Dynamic Asymmetric (physical layer automatically reduces the number of transmit and receive lanes based on system connectivity or lane faults)</p> <p>0x3—Static and Dynamic Asymmetric</p>
4:2	RO	<p>Reversal Support—Indicates if the physical layer supports reversal of transmit and receive lanes</p> <p>0x0—Reversal not supported</p> <p>0x1—Static uniform reversal supported (both transmit and receive directions shall be reversed)</p> <p>0x2—Dynamic and static uniform reversal supported (both transmit and receive directions shall be reversed)</p> <p>0x3—Static non-uniform reversal supported (either transmit or receive directions may be reversed)</p> <p>0x4—Dynamic and static non-uniform reversal supported (either transmit or receive directions may be reversed)</p> <p>0x5-0x7—Static and Dynamic Asymmetric</p>
5	RO	<p>Asymmetric Lane with Reversal Support</p> <p>0b—Unsupported</p> <p>1b—Supported</p>
7:6	-	Reserved
19:8	RO	Maximum number of supported Tx lanes in a non-aggregated interface
31:20	RO	Maximum number of supported Rx lanes in a non-aggregated interface

Table 8-38: PHY Remote Lane CAP

Bit Location	Access	Description
1:0	RW	<p>Remote Asymmetric Lane Support—Indicates if the physical layer supports configuring a different number of transmit and receive lanes</p> <p>0x0—Symmetric (fixed number of transmit and receives lanes)</p> <p>0x1—Static Asymmetric (requires physical layer re-initialization)</p> <p>0x2—Dynamic Asymmetric (physical layer automatically reduces the number of transmit and receive lanes based on system connectivity or lane faults)</p> <p>0x3—Static and Dynamic Asymmetric</p>
4:2	RW	<p>Remote Reversal Support—Indicates if the physical layer supports reversal of transmit and receive lanes</p> <p>0x0—Reversal not supported</p> <p>0x1—Static uniform reversal supported (both transmit and receive directions shall be reversed)</p> <p>0x2—Dynamic and static uniform reversal supported (both transmit and receive directions shall be reversed)</p> <p>0x3—Static non-uniform reversal supported (either transmit or receive directions may be reversed)</p> <p>0x4—Dynamic and dynamic non-uniform reversal supported (either transmit or receive directions may be reversed)</p> <p>0x5-0x7—Static and Dynamic Asymmetric</p>
5	RW	<p>Remote Asymmetric Lane with Reversal Support</p> <p>0x0—Unsupported</p> <p>0x1—Supported</p>
7:6	-	Reserved
19:8	RW	<p>Remote Tx Lane Enable</p> <p>If a SAI and in PHY UP, then this determines the number of enabled Tx Lanes in the remote aggregated interface. If a NAI and in PHY UP, then this field shall be set to 0x0.</p> <p>If the remote interface is not aggregated and in PHY UP, then this determines the number of enabled Tx Lanes.</p>
31:20	RW	<p>Remote Rx Lane Enable</p> <p>If a SAI and in PHY UP, then this determines the number of enabled Rx Lanes in the remote aggregated interface. If a NAI and in PHY UP, then this field shall be set to 0x0.</p>

		If the remote interface is not aggregated and in PHY UP, then this determines the number of enabled Rx Lanes.
--	--	---

Table 8-39: PHY Low Power Control

Bit Location	Access	Description
0	RW	PHY-LP 1 State Enable 0b—Entry Disabled 1b—Entry Enabled
1	RW	PHY-LP 2 State Enable 0b—Entry Disabled 1b—Entry Enabled
2	RW	PHY-LP 3 State Enable 0b—Entry Disabled 1b—Entry Enabled
3	RW	PHY-LP 4 State Enable 0b—Entry Disabled 1b—Entry Enabled
31:4	-	RsvdP

Table 8-40: PHY Low Power Timing Capability

Bit Location	Access	Description
3:0	RO	Entry Latency from PHY-Up to PHY-LP1—See <i>PHY Entry and Exit Latency Encodings</i>
7:4	RO	Exit latency from PHY-LP1 to PHY-Up—See <i>PHY Entry and Exit Latency Encodings</i>
11:8	RO	Entry Latency from PHY-Up to PHY-LP2—See <i>PHY Entry and Exit Latency Encodings</i>
15:12	RO	Exit latency from PHY-LP2 to PHY-Up—See <i>PHY Entry and Exit Latency Encodings</i>
19:16	RO	Entry Latency from PHY-Up to PHY-LP3—See <i>PHY Entry and Exit Latency Encodings</i>
23:20	RO	Exit latency from PHY-LP3 to PHY-Up—See <i>PHY Entry and Exit Latency Encodings</i>
27:24	RO	Entry Latency from PHY-Up to PHY-LP4—See <i>PHY Entry and Exit Latency Encodings</i>

<b>31:28</b>	RO	Exit latency from PHY-LP4 to PHY-Up—See <i>PHY Entry and Exit Latency Encodings</i>
--------------	----	---

Table 8-41: PHY Low Power CAP

Bit Location	Access	Description
0	RO	PHY-LP 1 Support 0b—Unsupported 1b—Supported
	RO	PHY-LP 2 Support 0b—Unsupported 1b—Supported
	RO	PHY-LP 3 Support 0b—Unsupported 1b—Supported
	RO	PHY-LP 4 Support 0b—Unsupported 1b—Supported
	-	RsvdP

Table 8-42: PHY Remote Low Power CAP

Bit Location	Access	Description
0	RO	PHY-LP 1 Support 0b—Unsupported 1b—Supported
	RO	PHY-LP 2 Support 0b—Unsupported 1b—Supported
	RO	PHY-LP 3 Support 0b—Unsupported 1b—Supported
	RO	PHY-LP 4 Support 0b—Unsupported 1b—Supported
	-	RsvdP

Table 8-43: PHY Up Low Power Control

Bit Location	Access	Description
0	RW	PHY-Up-LP 1 State Enable 0b—Entry Disabled 1b—Entry Enabled
1	RW	PHY-Up-LP 2 State Enable 0b—Entry Disabled 1b—Entry Enabled
2	RW	PHY-Up-LP 3 State Enable 0b—Entry Disabled 1b—Entry Enabled
3	RW	PHY-Up-LP 4 State Enable 0b—Entry Disabled 1b—Entry Enabled
31:4	-	RsvdP

Table 8-44: PHY Up Low Power Timing Capability

Bit Location	Access	Description
3:0	RO	Entry Latency from PHY-Up to PHY-Up-LP1—See <i>PHY Entry and Exit Latency Encodings</i>
7:4	RO	Exit latency from PHY-Up-LP1 to PHY-Up—See <i>PHY Entry and Exit Latency Encodings</i>
11:8	RO	Entry Latency from PHY-Up to PHY-Up-LP2—See <i>PHY Entry and Exit Latency Encodings</i>
15:12	RO	Exit latency from PHY-Up-LP2 to PHY-Up—See <i>PHY Entry and Exit Latency Encodings</i>
19:16	RO	Entry Latency from PHY-Up to PHY-Up-LP3—See <i>PHY Entry and Exit Latency Encodings</i>
23:20	RO	Exit latency from PHY-Up-LP3 to PHY-Up—See <i>PHY Entry and Exit Latency Encodings</i>
27:24	RO	Entry Latency from PHY-Up to PHY-Up-LP4—See <i>PHY Entry and Exit Latency Encodings</i>
31:28	RO	Exit latency from PHY-Up-LP4 to PHY-Up—See <i>PHY Entry and Exit Latency Encodings</i>

Table 8-45: PHY Up Lower Power CAP

Bit Location	Access	Description
0	RO	PHY-Up-LP 1 Support 0b—Unsupported 1b—Supported
1	RO	PHY-Up-LP 2 Support 0b—Unsupported 1b—Supported
2	RO	PHY-Up-LP 3 Support 0b—Unsupported 1b—Supported
3	RO	PHY-Up-LP 4 Support 0b—Unsupported 1b—Supported
31:4	-	Reserved

Table 8-46: PHY Up Remote Lower Power CAP

Bit Location	Access	Description
0	RO	PHY-Up-LP 1 Support 0b—Unsupported 1b—Supported
1	RO	PHY-Up-LP 2 Support 0b—Unsupported 1b—Supported
2	RO	PHY-Up-LP 3 Support 0b—Unsupported 1b—Supported
3	RO	PHY-Up-LP 4 Support 0b—Unsupported 1b—Supported
31:4	-	Reserved

## 8.18. Interface Statistics Structure

The Interface Statistics Structure is used to control and extract statistics for a given interface.

The following are the features and requirements associated with the Interface Statistics Structure:

- Any interface may support Interface Statistics Structure.
- The Interface Statistic structure shall use the *Interface Statistics Structure Format*.
  - Common Interface Statistics fields (blue) are common for all component interfaces.
  - Requester and Responder Interface Statistics fields (orange) are applicable to Requester and Responder component interfaces.
  - Packet Relay Statistics fields (purple) are unique to packet relay component interfaces. If a Requester or Responder contains an integrated switch or wants to provide additional insight, then the interface may support these fields. If Requester and Responder and packet relay fields are provisioned, then the Packet Relay fields shall be placed contiguous to the Requester and Responder fields.
- All Interface Statistic Structure fields shall be set to zero due to a *Full Component Reset*, a *Warm Component Reset*, a *Full Interface Reset*, or a *Warm Interface Reset* or the interface exits a low-power state that precludes maintaining statistics.
- Total bytes represents the sum of each applicable end-to-end packet's LEN field.
- Vendor-Defined statistics may be accessed through a Vendor-Defined Structure located by the Vendor-Defined Pointer.

**Developer Note:** Management can determine the number of packet and bytes received by an ingress interface in a packet relay component by tracking the number of packets and bytes transmitted by the peer interface attached to the ingress interface.

+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	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	7 6 5 4 3 2 1 0
I-STAT Status	I-STAT Control	I-STAT CAP 1	Size	Vers	Type	< Byte 0x0	
I-Snapshot PTR		Vendor-Defined PTR				< Byte 0x8	
I-Snapshot Interval						< Byte 0x10	
ECRC Errors		PCRC Errors				< Byte 0x18	
Rx Stomped ECRC		Tx Stomped ECRC				< Byte 0x20	
LLR Recovery		Non-CRC Transient Errors				< Byte 0x28	
Marked ECN		Packet Deadline Discards				< Byte 0x30	
AKEY Violations	Link NTE	Received ECN				< Byte 0x38	
R0						< Byte 0x40	
R1						< Byte 0x48	

Common Interface Statistics Fields

Total Transmitted Requests	< Byte 0x50
Total Transmitted Request Bytes	< Byte 0x58
Total Received Requests	< Byte 0x60
Total Received Request Bytes	< Byte 0x68
Total Transmitted Responses	< Byte 0x70
Total Transmitted Response Bytes	< Byte 0x78
Total Received Responses	< Byte 0x80
Total Received Response Bytes	< Byte 0x88

Requester and Responder Interface Statistics Fields

Total Transmitted Packets VC <sub>0</sub>	< Byte 0x50 / 0x90
Total Transmitted Bytes VC <sub>0</sub>	
Total Received Packets VC <sub>0</sub>	
Total Received Bytes VC <sub>0</sub>	
Occupancy VC <sub>0</sub>	
...	
Total Transmitted Packets VC <sub>N</sub>	
Total Transmitted Bytes VC <sub>N</sub>	
Total Received Packets VC <sub>N</sub>	
Total Received Bytes VC <sub>N</sub>	
Occupancy VC <sub>N</sub>	< Byte 0xYY

Packet Relay Interface Statistics Fields

Figure 8-14: Interface Statistics Structure Format

Table 8-47: Interface Statistics Structure Fields

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description
<b>Version (Vers)</b>	4	0x1	M	RO	Structure Version
<b>I-STAT CAP 1</b>	16	-	M	RO	Statistic Capabilities
		Bits 1:0			Provisioned Statistics Fields 0x0—Common Interface Fields 0x1—Common and Requester and Responder Interface Statistics Fields 0x2—Common and Packet Relay Interface Statistics Fields 0x3—Common, Requester and Responder Interface Statistics fields, followed by Packet Relay Interface Statistics Fields
		Bits 3:2			Maximum Snapshot Time—Indicates the maximum amount of time a snapshot takes to complete. If a new snapshot is initiated in less than Maximum Snapshot Time, then the snapshot results may be non-deterministic. 0x0—1 ms 0x1—10 ms 0x2—100 ms 0x3—1 second
		Bits 15:4			RsvdP
<b>I-STAT Control</b>	8	-	M	RW	Control statistics operations on this interface
		Bit 0			Enable Statistics Gathering 0b—Disable 1b—Enable Reads of this bit shall return 0b.
		Bit 1			Reset All Statistics—resets all statistics fields to 0x0. 1b—Reset Reads of this bit shall return 0b.
		Bit 2			Initiate Interface Statistics Snapshot—If supported, copy all interface statistics to the location identified by the I-Snapshot Pointer

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description
					<p>field. Statistics are contiguously copied starting from byte 0x18 of this instance of the Interface Statistics structure.</p> <p>During the copy process, an interface should continue to update all relevant statistics. As soon as the I-Snapshot Interval field is copied and no later than snapshot operation completion, the I-Snapshot Interval field shall be reset to 0x0 and resume operation. Further, once the snapshot operation completes, all interface statistics shall be reset to zero.</p> <p>1b—Perform Snapshot</p> <p>Reads of this bit shall return 0b.</p>
		Bits 7:3			RsvdP
I-STAT Status	8	-	M	RW1C	Statistic Status
		Bit 0			Interface Statistics Reset Status—Indicates if the statistics have been reset
		Bit 1			0b—No Reset Event 1b—Reset Event
		Bits 7:2			Snapshot Status—Indicates if an interface statistics snapshot request has been completed. If unsupported, then this field shall be hardwired to 0b. 0b—Incomplete 1b—Completed
					RsvdZ
Vendor-Defined PTR	32	-	O	RO	This field points to a <i>Vendor-Defined Structure</i> . If unsupported, then this field shall be Null.
I-Snapshot PTR	32	-	O	RO	This field points to the location where the Interface statistics associated with this component interface are copied upon command. If unsupported, then this field shall be Null.
I-Snapshot Interval	64	-	O	RO	If supported, then the interface shall update this field shall to contain the number of component cycles that have occurred since any of the following occurred; all of these events

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description
PCRC Errors					<p>shall cause I-Snapshot Interval to be immediately set to 0x0.</p> <ul style="list-style-type: none"> <li>• Interface Snapshot was taken</li> <li>• <i>Component Reset</i> (any type)</li> <li>• <i>Interface Reset</i> (any type)</li> <li>• L-Down Event</li> </ul> <p>Management may calculate the elapsed time as follows (<i>Core Structure C-Op-Clock</i>):</p> <p>Elapsed time =</p> $\text{I-Snapshot Interval} / (\text{C-Op-Clock} * 1024 \text{ Hz})$ <p>If unsupported, then this field shall be Reserved.</p>
	32	-	M	RO	Total number of PCRC transient errors detected in received packets.
	32	-	M	RO	Total number of ECRC transient errors detected in received packets.
	32	-	M	RO	Total number of packets that this interface stomped the ECRC field
	32	-	M	RO	Total number of packets that this interface received with a stomped ECRC field
	32	-	M	RO	Total number of Transient Errors detected that are unrelated to CRC validation (covers link-local and end-to-end packets)
	32	-	M	RO	Total number of times <i>Link-level Reliability (LLR)</i> recovery has been initiated by this interface (not the number of packets retransmitted due to initiating LLR recovery)
	32	-	M	RO	Number of packets discarded by this interface due to the Congestion Deadline sub-field reaching zero prior to packet transmission
	32	-	M	RO	Number of packets that the component set Congestion ECN = 1b prior to transmission through this interface.
	32	-	M	RO	Number of packets received on this interface with the Congestion ECN = 1b

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description
Link NTE	16	-	M	RO	Total number of link-local non-transient errors detected.
AKEY Violations	16	-	M	RO	Total number of Access Key Violations detected for packets received or transmitted on this interface.
Total Transmitted Requests	64	-	M	RO	Total number of transmitted request packets Shall be applicable only to Requester and Responder components.
Total Transmitted Requests Bytes	64	-	M	RO	Total transmitted request packet bytes (sum of the LEN field in all request packets) Shall be applicable only to Requester and Responder components.
Received Requests	64	-	M	RO	Total number of request packets received Shall be applicable only to Requester and Responder components.
Total Received Requests Bytes	64	-	M	RO	Total received request packet bytes (sum of the LEN field in all request packets) Shall be applicable only to Requester and Responder components.
Total Transmitted Responses	64	-	M	RO	Total number of transmitted response packets Shall be applicable only to Requester and Responder components.
Total Transmitted Response Bytes	64	-	M	RO	Total transmitted response packet bytes (sum of the LEN field in all response packets) Shall be applicable only to Requester and Responder components.
Received Responses	64	-	M	RO	Total number of received response packets Shall be applicable only to Requester and Responder components.
Total Received Response Bytes	64	-	M	RO	Total received response packet bytes (sum of the LEN field in all response packets) Shall be applicable only to Requester and Responder components.

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description
<b>Total Transmitted Packets VC<sub>k</sub></b>	64	-	M	RO	Total number of end-to-end packets transmitted on packet relay interface VC <sub>k</sub>
<b>Total Transmitted Bytes VC<sub>k</sub></b>	64	-	M	RO	Total bytes transmitted on packet relay interface VC <sub>k</sub> (sum of the LEN field in all end-to-end packets)
<b>Total Received Packets VC<sub>k</sub></b>	64	-	M	RO	Total number of end-to-end packets received on packet relay interface VC <sub>k</sub>
<b>Total Received Bytes VC<sub>k</sub></b>	64	-	M	RO	Total bytes received on packet relay interface VC <sub>k</sub> (sum of the LEN field in all end-to-end packets)
<b>Occupancy VC<sub>k</sub></b>	64	-	M	RO	Sum of the number of end-to-end packets present in the input queue of a packet relay interface accumulated each cycle.
<b>R0</b>	64	-	-	-	Reserved
<b>R1</b>	64	-	-	-	Reserved

## 8.19. Component Media Structure

The Component Media structure is used by components that contain addressable memory media. The Component Media structure describes the memory media and in some cases, how the memory media is organized.

- 5 The following are the features and requirements associated with the Component Media structure:
- Any component type with addressable memory media may support the Component Media structure.
    - If a component supports the P2P-OpClass and contains addressable media accessible using P2P-Core OpClass request packets, then:
      - The component shall support the Component Media structure
      - The component shall support the *Responder Bank Structure*
      - The component may support multiple instances of the Component Media structure.
      - A Component Media structure may be associated with a subset of component interfaces, however, only one Component Media structure may be associated with a given component interface.
    - If a component does not support the P2P-OpClass and contains addressable media, then it may support a single Component Media structure.
  - A component may contain multiple media types, e.g., a mix of volatile and non-volatile media. If multiple media types are supported and each media is independently addressable, then the component shall support the Component Media structure.

- The Component Media structure shall use the *Component Media Structure Format*.
- A component shall communicate its worst-case read request packet execution latency for each supported media type. This value is used to calculate Requester retransmission timer values. If both media types contain addressable memory, then the worst-case read latency of the two media shall be reported in the *Core Structure* RDLAT field. Read Latency is calculated using the following formula:
  - $\text{Read Latency} = (\text{Read Latency Base} * \text{Media Latency Scale})$
- A component shall communicate its worst-case write request packet execution latency for each supported media type. This value is used to calculate Requester retransmission timer values. If both media types contain addressable memory, then the worst-case write latency of the two media shall be reported in the *Core Structure* WRLAT field. Write Latency is calculated using the following formula:
  - $\text{Write Latency} = (\text{Write Latency Base} * \text{Media Latency Scale})$
- Media read and write endurances (i.e., the number of Read or Write Request executions before the media is considered unreliable) vary by media type. The following formulas are used to calculate these endurances:
  - $\text{Read Endurance} = (\text{Read Endurance Base} * \text{Endurance Scale}) \text{ cycles}$
  - $\text{Write Endurance} = (\text{Write Endurance Base} * \text{Endurance Scale}) \text{ cycles}$
- The primary media shall be addressable media, i.e., read and write request packets shall be able to read from and write to this media.
- The secondary media may be addressable media.
  - The secondary media may be non-addressable media, i.e., operations such as read and write request packets shall not be able to read from or write to this media. In this case, the secondary media provides a different role, e.g., acts as a cache or non-volatile backup for the primary media.
- The secondary media may act as a backup for the primary media. A backup involves copying the contents from the primary media to the secondary media. A restore copies the contents of the secondary media to the primary media.
  - The secondary media shall be provisioned with memory resources that are equal to or greater than the primary media's memory resources (including data integrity bits such as ECC).
  - The secondary media may be provisioned with memory resources capable of storing multiple versions of the primary media.
    - The resources shall be consumed in FIFO, with newer versions overwriting older versions once all backup resources have been consumed.
- If the secondary media is configured to provide backup service, then a backup shall be performed:
  - On-demand when management sets the Initiate Backup Service field. While the backup is performed, the component should stop executing request packets that modify the primary media sub-ranges being copied (a Responder may return a *Responder Not Ready (RNR) NAK* to inform the Requester to retransmit the request packet at a future time, ideally, after the target sub-range has been copied). Management is responsible for ensuring that the stoppage does not cause applications to experience serious errors or to fail.
  - If the primary media is enabled for emergency backup (referred to as "armed") and when conditions warrant (e.g., power fail), then the component shall automatically copy the contents of the primary media to the secondary media. While an emergency backup is being performed, the component shall stop executing all new request packets other than

5 *In-band Management* request packets (if supported). An emergency backup may cause applications to experience serious errors or to fail. Applications and management are responsible for any recovery actions.

- 10
- If the secondary media is configured to provide restore service, then a restore shall be performed on-demand when management sets the Initiate Restore Service field. While the restoration is performed, the component shall stop executing all new request packets other than *In-band Management* request packets (if supported). Management is responsible for ensuring that the stoppage does not cause applications to experience serious errors or to fail.
  - If the secondary media supports multiple versions, then the latest version shall be restored unless indicated otherwise.
  - 15 • Given media abstraction, the architecture does not mandate a specific error detection and correction code or algorithm. Instead, the component communicates the number of bit errors a component can detect and the number of bit errors a component can correct over a given bit or byte range. Software uses this information in conjunction with the number of available row and media device spares to determine component resiliency, and to initiate solution-level resiliency management actions.
  - 20 • Notification of correctable and uncorrectable errors detected during request packet execution may be returned within response packets or independently communicated to management. Notification is controlled through the *Primary Media CAP 1 Control* and *Secondary Media CAP 1 Control*.
  - 25 • Sanitize and Erase (SE) operations are used to return or set memory media to a specified state, e.g., to zero all or a sub-range media, to overwrite media, to reset cryptographic keys, etc. SE operations can be very light-weight and complete very quickly (e.g., zero media or reset cryptographic keys), or can take significant amounts of time (overwrite media multiple times). Software determines when a SE operation is performed, e.g., immediately, upon the next component reset, or as part of initialization. SE operations and requirements are as follows:

30 **Developer Note:** *Solution developers are responsible for ensuring that SE operations and applications do not simultaneously access the same media ranges. Simultaneous access will result in non-deterministic media state and / or application behavior.*

- 35
- A SE operation shall not impact any resource (media, intermediate cache, etc.) that cannot contain application or user data (e.g., a resource that contains executables used by the media controller, data-centric accelerators, option ROMs, etc.).
  - A SE operation that is performed on a specified range shall impact only the media itself.
    - Prior to initiating the SE operation, software sets the range's address (Primary Region Address or Secondary Region Address) and the length (Primary Region Length or Secondary Region Length).
    - Due to media-specific requirements, an SE operation may impact a larger quantity of media than indicated by the range. For example, if a media supports only block access, then the media controller would execute the SE operation onto an integer number of media blocks.

40

  - Software may terminate a SE operation prior to its completion. Upon termination, the media controller shall set the SE Failed bit in the media's status field. Further, all impacted resources and media contents are non-deterministic.
  - 45 • Upon successful SE operation completion, the media controller shall set the SE Completed bit in the media's status field.

- Upon SE operation failure, the media controller shall set the SE Failed bit in the media's status field.
- A component may support SE Fast Zero Media. A SE Fast Zero Media operation sets the addressed media to the uninitialized state, i.e., each media bit shall be set to 0b. If the media controller provides Error Detection / Error Correction, then these bits shall be set to reflect the media uninitialized state, i.e., no error.
  - The method used to set each bit is outside of this specification's scope.
  - The media controller may mark the media's Meta data to indicate that the media has been zeroed and immediately complete the operation. If so, then the media controller shall zero the media in the background or upon subsequent access.
  - SE Fast Zero Media shall apply only to the primary or secondary media resources currently used to contain application or user data.
- A component may support SE Fast Zero Range Media. The semantics of a SE Fast Zero Range Media operation shall be as specified as for a SE Fast Zero Media with one exception—the operation shall be applied only to the indicated media range.
  - Upon receipt of a request packet that targets Data Space, the media controller shall not execute the request packet. If the remaining time to complete a SE Fast Zero Range Media operation is short, then the media controller should generate a *Responder Not Ready (RNR) NAK*, else it shall return a *Standalone Acknowledgment* (Reason = In-progress Sanitize and Erase (SE) Operation).
- If a component supports any of the following operations: SE Zero Media, SE Cryptographic Key, SE Overwrite Media, or SE Vendor-defined, then:
  - The media controller shall ensure that all application and user data stored or capable of being stored in the targeted media, in any intermediate caches, in media controller buffers, in media log entries, in control plane structures (e.g., work request or completion queues), spared or deallocated media ranges, meta data, application and user data integrity fields, etc. cannot be recovered or accessed by any means subsequent to successful completion of the SE operation.
  - Once a media controller is requested to initiate any of these SE operations, Requesters should not transmit request packets to the media controller that target Data Space. Upon receipt of a request packet that target Data Space, the media controller shall not execute the request packet. Instead, it shall return a *Standalone Acknowledgment* (Reason = In-progress Sanitize and Erase (SE) Operation).
  - Once any of these operations are initiated, the media controller shall set the SE In-Progress bit in the media's status field, and shall update the SE Percentage field to indicate current operation completion progress.
- A component may support SE Zero Media. A SE Zero Media operation returns all volatile and non-volatile addressable media to the uninitialized state, i.e., each media bit shall be set to 0b.
  - The method used to clear each media bit is outside of this specification's scope.
  - If configured, a SE Zero Media operation shall be performed as part of media initialization.
- A component may support SE Cryptographic Key. A SE Cryptographic Key operation scrambles all encryption keys associated with the media.
  - The method used to scramble encryption keys is outside of this specification's scope.

- A component may support SE Overwrite Media. A SE Overwrite Media operation applies the SE Overwrite Pattern.
  - The SE Overwrite Pattern may be applied multiple times.
  - The SE Overwrite Pattern may be inverted with each application. To invert the SE Overwrite Pattern, exclusive OR each bit with 1b.
  - Non-media resources are overwritten with all 1s. If inversion is indicated, then the media controller alternates between all 1s and all 0s with each subsequent overwrite pass.
  - A SE Overwrite Media operation may be performed on a specified media range.
- A component may support SE Vendor-defined. A SE Vendor-defined shall return the media to a state as specified by entities outside of this specification's scope, e.g., a government regulatory body or a national / international standard. SE Vendor-defined operation attributes and semantics shall be communicated through a *Vendor-Defined Structure* with UUID located via the SE Vendor-defined PTR within the Component Media Structure.
  - A SE Vendor-defined operation may be performed on a specified media range.
- A component may support deallocation of a region within the primary or secondary media. To deallocate a range, software sets the region's address (Primary Region Address or Secondary Region Address) and the length (Primary Region Length or Secondary Region Length). Once set, software initiates deallocation (Deallocate Primary Range or Deallocate Secondary Range). Upon completion, the media controller sets the media status bit (Deallocate Primary Range Completed or Deallocate Secondary Range Completed).
- Primary Media Log and Secondary Media Log requirements:
  - If a component supports the Component Media structure, it shall support the Primary Media Log. If the component supports a secondary media, then it shall support the Secondary Media Log.
  - Each media log shall contain at least the minimum number of log entries. The Primary Media Log contains Plog Size entries, and the Secondary Media Log contains Slog Size entries.
  - The component shall consume log entries in monotonically-increasing order (modulo the respective log's size) starting with log entry 0.
    - If there are no available log entries, then the component shall silently discard the new information that was to be logged, and shall update the corresponding log status field to indicate logging failed due to a lack of available log entries.
  - Software shall release log entries in monotonically increasing order (modulo the respective log size).
- The component shall maintain a monotonically-increasing correctable error counter (modulo  $2^{24}$ ).
  - The current value of this counter shall be copied into the Correctable Error Count field whenever a log entry is generated.
  - Whenever the counter wraps to zero, the component shall update the corresponding status field.
  - The counter shall be set to zero whenever the component is initialized or is reset via a *Component Reset* (any type).
- The component shall maintain a monotonically-increasing uncorrectable error counter (modulo  $2^{24}$ ).
  - The current value of this counter shall be copied into the Uncorrectable Error Count field whenever a log entry is generated.

- Whenever the counter wraps to zero, the component shall update the corresponding status field.
- The counter shall be set to zero whenever the component is initialized or is reset via a *Component Reset* (any type).
- If a component supports non-volatile media, then media maintenance, e.g., garbage collection, needs to be periodically performed to ensure optimal performance and correct operation. Media maintenance needs vary as a function of media-specific attributes, static and dynamic wear-leveling, data relocation due to memory row / block sparing, creation of erased or cleansed memory ranges for subsequent write operations, etc.
  - Unless configured otherwise, media maintenance shall be autonomously performed as a media controller background activity. The media controller should minimize maintenance impacts on application performance.
  - To reduce performance jitter, software may disable primary and / or secondary media maintenance.
    - Software should not indefinitely disable primary or secondary media maintenance.
    - If a media controller reaches a state where maintenance cannot be delayed any further, then the media controller may temporarily override its configuration to perform maintenance that can be completed within the minimum maintenance time (Min PRI Maintenance Time or Min SEC Maintenance Time).
    - Management may initiate media maintenance at its discretion. Prior to initiation, management configures the maximum maintenance time (Max PRI Maintenance Time or Max SEC Maintenance Time).
    - Whenever directed by management or an override situation occurs, the media controller should perform media maintenance at its maximum capability even if this temporarily causes significant performance jitter.
    - Based on current media state and media access rate and operation patterns, a media controller shall continuously calculate the approximate delta time until media maintenance will be required. If delta time is less than or equal to the time period required to inform management (Inform PRI Media Maintenance / Inform SEC Media Maintenance), then the media controller shall generate an event to inform management (see *Component Error and Signal Event Structure C-Event*).
      1. If an *Unsolicited Event (UE) Packet* is used, then the Event-specific field shall be set to indicate which media and the amount of time at the current load until maintenance is required.
      2. If a component-local interrupt is used, then the time period required to inform management is treated as the amount of time at the current load until maintenance is required.

+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	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	7 6 5 4 3 2 1 0										
Vendor-Defined PTR		Size		Vers	Type	< Byte 0x0											
Next Media Structure PTR		Component Media Control				< Byte 0x8											
Primary Media Status																	
Secondary Media Status																	
Primary Media CAP 1 [63:0]																	
Primary Media CAP 1 [127:64]																	
Secondary Media CAP 1 [63:0]																	
Secondary Media CAP 1 [127:64]																	
Primary Media CAP 1 Control																	
Secondary Media CAP 1 Control																	
Fault Injection																	
Primary Media Base Address						Plog Size	PRI-MLEN										
Secondary Media Base Address						Slog Size	SEC-MLEN										
C-Media ID	R0			Interface ID Mask													
Primary SIT PTR				Primary Namespace PTR													
Secondary SIT PTR				Secondary Namespace PTR													
Power Status	Uncorrectable Error Count			Correctable Error Count													
R1	SE Percentage	Media Signal	Media MGR SID		Media MGR CID	MGR ID	< Byte 0x88										
Interrupt Address 0																	
Interrupt Address 1																	
Max SEC Maintenance Time	Max PRI Maintenance Time	Min SEC Maintenance Time		Min PRI Maintenance Time		< Byte 0xA0											
SE Overwrite Pattern		Inform SEC Maintenance Time		Inform PRI Maintenance Time		< Byte 0xA8											
R2		SE Vendor-defined PTR															
R3																	
Primary Region Address																	
Primary Region Length																	
Secondary Region Address																	
Secondary Region Length																	
Primary Media UUID [63:0]																	
Primary Media UUID [127:64]																	
Secondary Media UUID [63:0]																	
Secondary Media UUID [127:64]																	
Primary Media Log																	
Secondary Media Log																	
< Byte 0xYYY																	

Figure 8-15: Component Media Structure Format

Table 8-48: Component Media Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Version (Vers)	4	0x1	M	RO	Structure Version
Vendor-Defined PTR	32	-	O	RO	<p>Pointer to a <i>Vendor-Defined Structure</i> associated with the memory media or Null if unsupported.</p> <p><b>Developer Note:</b> Developers are encouraged to use a Vendor-Defined with UUID structure to uniquely identify the capabilities.</p>
Next Media Structure PTR	32	-	O	RO	<p>Pointer to the next supported Component Media Structure or Null if not present</p> <p>A component may support a maximum of three Component Media structures.</p>
Component Media Control	32	-	M	RWS	<p>Controls media and media-related services at the component level</p> <p>Bits 1:0</p> <ul style="list-style-type: none"> <li>Sanitize and Erase (SE) Media.</li> <li>0x0—No impact</li> <li>0x1—Immediately execute the configured SE type on the primary and / or secondary media</li> <li>0x2—On the next component initialization or component reset event, execute the configured SE type on the primary and / or secondary media.</li> <li>0x3—Reserved</li> </ul> <p>Reads of this field shall return 0x0.</p> <p>Bit 2</p> <ul style="list-style-type: none"> <li>Terminate SE Operation</li> <li>1b—Immediately terminate any in-progress SE operation. Upon termination, all media contents shall be treated as non-deterministic.</li> </ul> <p>Reads of this bit shall return 0b.</p> <p>Bits 4:3</p> <ul style="list-style-type: none"> <li>Management Notification Level—Event severity level that causes the component to inform management of the event (see <i>Component Error and Signal Event Structure</i>)</li> <li>0x0—Critical</li> <li>0x1—Critical and Caution</li> <li>0x2—Critical, Caution, and Non-critical</li> <li>0x3—Reserved</li> </ul>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					Reads of this field shall return 0x0.
		Bit 5			<p>Arm Emergency Backup Service</p> <p>0b—Disables emergency backup service</p> <p>1b—If the secondary media supports emergency backup services, then this informs the component to arm the service to enable emergency primary media backup when conditions warrant, e.g., power failure.</p>
		Bit 6			<p>Initiate Backup Service—If the secondary media supports backup services, then this informs the component to copy the contents of the primary media to the backup media.</p> <p>1b—Initiates backup</p> <p>Reads of this bit shall return 0b.</p>
		Bits 9:7			<p>Initiate Restore Service—If the secondary media supports back services, then this informs the component to copy the contents of the backup media to the addressable media.</p> <p>0x0—No impact</p> <p>0x1—Initiates restoration of the latest backup</p> <p>0x2—Initiates restoration of the (latest – 1) backup</p> <p>0x3—Initiates restoration of a vendor-specified backup. Vendor-specified is controlled through a vendor-defined structure.</p> <p>0x4—Aborts in-flight restoration operation</p> <p>0x5-0x7—Reserved</p> <p>Reads of this field shall return 0x0.</p>
		Bit 10			<p>Wait-for-Backup-Power—If the component supports a secondary power source, then this determines if the component shall wait until the secondary power source is fully charged before setting the Component Power Good bit within the Power Status field.</p> <p>0b—Does not Wait</p> <p>1b—Wait</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					Reads of this bit shall return 0b.
		Bit 11			<p>Disable Primary Media Maintenance—if disabled, then all media controller-initiated primary media maintenance (e.g., garbage collection) that could cause inconsistent performance shall be delayed.</p> <p>0b—Enabled 1b—Disabled</p>
		Bit 12			<p>Disable Secondary Media Maintenance—if disabled, then all media controller-initiated secondary media maintenance (e.g., garbage collection) that could cause inconsistent performance shall be delayed.</p> <p>0b—Enabled 1b—Disabled</p>
		Bit 13			<p>Initiate Primary Media Maintenance—Directs the media controller to immediately begin primary media maintenance (irrespective of Disable Primary Media Maintenance configuration), and to update Primary Media Status upon completion. Primary media maintenance may execute up to Max Pri Maintenance time.</p> <p>Component performance may be inconsistent during primary media maintenance.</p> <p>1b—Initiate Primary Media Maintenance</p> <p>Reads of this bit shall return 0b.</p>
		Bit 14			<p>Initiate Secondary Media Maintenance—Directs the media controller to immediately begin secondary media maintenance (irrespective of Disable Secondary Media Maintenance configuration), and to update Secondary Media Status upon completion. Secondary media maintenance may execute up to Max Sec Maintenance time.</p> <p>Component performance may be inconsistent during secondary media maintenance.</p> <p>1b—Initiate Secondary Media Maintenance</p> <p>Reads of this bit shall return 0b.</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Primary Media Status		Bits 32:15			RsvdP
	64	-	M	-	See <i>Primary Media Status</i>
	64	-	O	-	See <i>Secondary Media Status</i>
	128	-	M	-	See <i>Primary Media CAP 1 Fields</i>
	128	-	O	-	See <i>Secondary Media CAP 1 Fields</i>
	64	-	M	RW	See <i>Primary Media CAP 1 Control</i>
	64	-	M	RW	See <i>Secondary Media CAP 1 Control</i>
	64	-	O	RW	Controls which error condition to test. This field is not intended for use in normal operation. See <i>Component Media Structure Fault Injection Field</i>
	8	-	M	RO	This field is used to indicate the primary media length. Primary Media Length = $2^{\text{PRI-MLEN}}$ bytes. PRI-MLEN shall be less than or equal to 64.
	4	-	O	RO	Primary Media Log Size 0x0—16 entries 0x1—64 entries 0x2—128 entries 0x3—256 entries 0x4—512 entries 0x5—1024 entries 0x6—2048 entries 0x7—4096 entries 0x8-0xF—Reserved
Primary Media Base Address	52	-	O	RO	Address associated with byte 0 of the Primary Media. The Address is an integer 4096-byte multiple, i.e., the lower 12 bits are implicit and treated as 0x0.  If a component supports zero-based addressing ( <i>Core Structure Component CAP 1 Address Field</i>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>Secondary Media Base Address</b>					Interpretation == 0b), then this field shall be hardwired to 0x0.
<b>SEC-MLEN</b>	52	-	O	RO	Address associated with byte 0 of the Secondary Media. The Address is an integer 4096-byte multiple, i.e., the lower 12 bits are implicit and treated as 0x0.
<b>Slog Size</b>	8	-	M	RO	<p>This field is used to indicate the secondary media length. Secondary Media Length = <math>2^{\text{SEC-MLEN}}</math> bytes.</p> <p>SEC-MLEN shall be less than or equal to 64.</p>
	4	-	O	RO	<p>Secondary Media Log Size</p> <p>0x0—16 entries</p> <p>0x1—64 entries</p> <p>0x2—128 entries</p> <p>0x3—256 entries</p> <p>0x4—512 entries</p> <p>0x5—1024 entries</p> <p>0x6—2048 entries</p> <p>0x7—4096 entries</p> <p>0x8-0xF—Reserved</p>
<b>Interface ID Mask</b>	16	-	O	RO	<p>If the component supports the P2P-Core OpClass, then this field represents a bit mask where each Bit<sub>K</sub> corresponds to a component interface (bit 0 to interface 0, bit 1 to interface 1, etc.). If Bit<sub>K</sub> is 1b, then the packets exchanged using interface K shall use this instance of the Component Media structure. This enables a component that supports multiple Component Media structures to identify the corresponding interfaces.</p> <p>A given Bit<sub>K</sub> shall be set to 1b only in a single instance among all supported Component Media structures.</p> <p>This field shall be applicable only if the component supports the P2P-Core OpClass, else, it shall be Reserved and the Component Media structure shall apply to the entire component.</p>
<b>C-Media ID</b>	4	-	M	RO	Identifier associated with this instance of the Component Media Structure.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>Primary Namespace PTR</b>					Identifiers shall be unique among all instances of this structure type supported by the component.
	32	-	O	RO	<p>If the primary media is accessed using block storage semantics and supports Namespaces, then this field points to the location of the associated table.</p> <p>To support a variety of table formats, this pointer shall point to a <i>Vendor-Defined Structure</i> with UUID. The vendor-defined UUID is used to identify the specific table type and format (the structure's layout is outside of this specification's scope).</p> <p>If unsupported, then this field shall be hardwired to Null.</p>
<b>Primary SIT PTR</b>	32	-	O	RO	<p>If the primary media is accessed using block storage semantics and supports a Software Interface Table (SIT), then this field points to the location of the associated table.</p> <p>To support a variety of table formats, this pointer shall point to a <i>Vendor-Defined Structure</i> with UUID. The vendor-defined UUID is used to identify the specific table type and format (the structure's layout is outside of this specification's scope).</p> <p>If unsupported, then this field shall be hardwired to Null.</p>
<b>Secondary Namespace PTR</b>	32	-	O	RO	<p>If the secondary media is accessed using block storage semantics and supports Namespaces, then this field points to the location of the associated table.</p> <p>To support a variety of table formats, this pointer shall point to a <i>Vendor-Defined Structure</i> with UUID. The vendor-defined UUID is used to identify the specific table type and format (the structure's layout is outside of this specification's scope, e.g., could be specified by another industry body).</p> <p>If unsupported, then this field shall be hardwired to Null.</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Secondary SIT PTR	32	-	O	RO	<p>If the secondary media is accessed using block storage semantics and supports a Software Interface Table (SIT), then this field points to the location of the associated table.</p> <p>To support a variety of table formats, this pointer shall point to a <i>Vendor-Defined Structure</i> with UUID. The vendor-defined UUID is used to identify the specific table type and format (the structure's layout is outside of this specification's scope).</p> <p>If unsupported, then this field shall be hardwired to Null.</p>
Correctable Error Count	24	-	M	RW	For each correctable error detected in the primary or secondary media associated with this structure, this counter shall be incremented by one (modulo $2^{24}$ ).
Uncorrectable Error Count	24	-	M	RW	For each uncorrectable error detected in the primary or secondary media associated with this structure, this counter shall be incremented by one (modulo $2^{24}$ ).
Power Status	16	-	O	RW1CS	Component Power Status
		Bit 0			Component Power Good—Indicates the primary power source and the secondary power source are operating as expected.
		Bit 1			Unstable / Insufficient Power—component has detected the primary power source isn't operating within expected parameters
		Bit 2			Unexpected Power Loss—component has detected the primary power source has failed
		Bit 3			Battery-Capacitor Present—the component supports a secondary power source such as a battery, capacitor, etc. that is capable of maintaining volatile media should the primary power supply fail
		Bit 4			Battery-Capacitor Charged—if the component supports a secondary power source, then this indicates if it is currently charged
		Bit 5			Battery-Capacitor Not Ready—if the component supports a secondary power source, then this

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
		Bits 15:6			indicates it does not have sufficient power to meet the expected use.
					RsvdZ
<b>MGR ID</b>	2	-	M	RW	<p>Determines if the media is independently managed by a component other than any of the management components identified within the <i>Core Structure</i>, and whether the Media MGR CID and Media MGR SID fields are configured.</p> <p>0x0—Not Independently Managed</p> <p>0x1—Independently Managed, Media MGR CID Valid</p> <p>0x2—Independently Managed, Media MGR CID and Media MGR SID Valid</p> <p>0x3—Reserved</p>
<b>Media MGR CID</b>	12	-	O	RW	If the media is independently managed by a component other than any of the management components identified within the <i>Core Structure</i> , then this field contains the CID of the media manager.
<b>Media MGR SID</b>	16	-	O	RW	If the media is independently managed by a component other than any of the management components identified within the <i>Core Structure</i> , then this field contains the SID of the media manager.
<b>Media Signal</b>	4	-	O	RW	<p>A component may signal management when events of a given severity are logged within the Primary Media log or the Secondary Media log.</p> <p>If configured, then the component shall generate one or both component-local interrupts, an <i>Unsolicited Event (UE) Packet</i>, or both component local interrupt(s) and an <i>Unsolicited Event (UE) Packet</i>.</p> <p>0x0—No signal is generated</p> <p>0x1—Component-local interrupt 0</p> <p>0x2—Component-local interrupt 1</p> <p>0x3—Component-local interrupts 0 and 1</p> <p>0x4—<i>Unsolicited Event (UE) Packet</i></p> <p>0x5—Component-local interrupt 0 and an <i>Unsolicited Event (UE) Packet</i></p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
SE Percentage					<p>0x6—Component-local interrupt 1 and an <i>Unsolicited Event (UE) Packet</i>      0x7—Component-local interrupts 0 and 1, and an <i>Unsolicited Event (UE) Packet</i>      0x8-0xF—Reserved</p>
	7	-	O	RO	<p>When any of the following SE operations are initiated—SE Zero Media, SE Cryptographic Key, SE Overwrite Media, or SE Vendor-defined—the media controller shall set this field to zero to indicate 0% operation completion.</p> <p>As the operation progresses, the media controller shall update this field to indicate current percentage (integer 0-100) of operation completion.</p>
Interrupt Address 0	64	-	O	RW	The component-local address to target component-local interrupt 0
Interrupt Address 1	64	-	O	RW	The component-local address to target component-local interrupt 1
Min PRI Maintenance Time	16	-	O	RO	Minimum amount of time in $\mu$ s required to perform essential primary media maintenance to ensure correct operation when Initiate Primary Media Maintenance == 1b.
Min SEC Maintenance Time	16	-	O	RO	Minimum amount of time in $\mu$ s required to perform essential secondary media maintenance to ensure correct operation when Initiate Secondary Media Maintenance == 1b.
Max PRI Maintenance Time	16	-	O	RW	<p>Maximum amount of time in <math>\mu</math>s that a media controller has to perform primary media maintenance when Initiate Primary Media Maintenance == 1b.</p> <p>If Max PRI Maintenance Time is less than Min PRI Maintenance Time, then Min PRI Maintenance Time shall be used instead.</p>
Max SEC Maintenance Time	16	-	O	RW	Maximum amount of time in $\mu$ s that a media controller has to perform secondary media maintenance when Initiate Secondary Media Maintenance == 1b.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					If Max SEC Maintenance Time is less than Min SEC Maintenance Time, then Min SEC Maintenance Time shall be used instead.
Inform PRI Maintenance Time	16	-	O	RW	<p>Time period (in <math>\mu</math>s) required to inform management prior to primary media maintenance being required.</p> <p>If zero, then the media controller shall not inform management when primary media maintenance is required.</p> <p>This field shall be acted upon if Disable Primary Media Maintenance == 1b.</p>
Inform SEC Maintenance Time	16	-	O	RW	<p>Time period (in <math>\mu</math>s) required to inform management prior to secondary media maintenance being required.</p> <p>If zero, then the media controller shall not inform management when secondary media maintenance is required.</p> <p>This field shall be acted upon if Disable Secondary Media Maintenance == 1b.</p>
SE Overwrite Pattern	64	-	O	RW	If the component support SE Overwrite Media, then the contents of this field are used to perform the SE Overwrite Media operation. Software shall write the pattern to this field prior to initiating an SE Overwrite Media operation.
SE Vendor-defined PTR	32	-	O	RO	If the component supports a SE Vendor-defined operation, then this field contains a pointer to a Vendor-Defined Structure with UUID that identifies the operation's type and attributes.
Primary Region Address	64	-	O	RW	This field contains the address to byte 0 of a primary media memory region on which an operation is performed, e.g., a sanitize and erase operation, a deallocation operation, etc.
Primary Region Length	64	-	O	RW	This field contains the length of the primary media memory region on which an operation is performed.
Secondary Region Address	64	-	O	RW	This field contains the address to byte 0 of a secondary media memory region on which an

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Secondary Region Length					operation is performed, e.g., a sanitize and erase operation, a deallocation operation, etc.
	64	-	O	RW	This field contains the length of the secondary media memory region on which an operation is performed.
	128	-	M	RO	Primary Media UUID uniquely identifies the media type. The UUID may be unique to a given vendor's media, a de facto identifier for a class of media, or an industry standard media.
	128	-	O	RO	Secondary Media UUID uniquely identifies the media type. The UUID may be unique to a given vendor's media, a de facto identifier for a class of media, or an industry standard media.
	-	-	M	-	Each log entry shall use the <i>Primary and Secondary Media Log Entry Format</i> .
	-	-	O	-	Each log entry shall use the <i>Primary and Secondary Media Log Entry Format</i> .
R0	44	-	-	-	RsvdP
	23	-	-	-	RsvdP
	32	-	-	-	Reserved
	64	-	-	-	Reserved

### 8.19.1. Primary Media Status

If a bit sub-field, then a value of 1b indicates the status / event / operating condition is true, and a value of 0b indicates that the status / event / operating condition is false.

Table 8-49: Primary Media Status

Bit Location	Access	Description
0	RW1CS	Uninitialized—the media is in its default state. This may be due to a power-up event, a component reset event, or due to completion of a SE operation.
	RW1CS	Media Initialization / Restoration Failed
	RW1CS	Media Initialization / Restoration Succeeded
	RW1CS	SE In-progress—the media is in a non-deterministic state

Bit Location	Access	Description
4	RW1CS	SE Completed—the media was successfully set to the expected state
5	RW1CS	SE Failed—the media is in a non-deterministic state
6	RW1CS	Patrol Scrubbing Completed—the component has successfully completed at least one patrol scrubbing cycle of the primary media.
7	RW1CS	Fatal Media Error Containment—the component has detected an uncorrectable error, but is unable to isolate to a specific media address.  If this media is volatile, all request packets shall continue to be validated and executed.  If this media is non-volatile, all request packets that modify the impacted media range (e.g., a Write request), shall not be executed, and a <i>Standalone Acknowledgment</i> (Reason = Fatal Media Error Containment Triggered) shall be returned.  A <i>Component Reset</i> (any type) shall clear Fatal Media Error Containment.
8	RW1CS	Logging Failed—one or more attempts to generate a log entry failed due to a lack of free log entries.
9	RW1CS	Correctable Error Counter Wrapped
10	RW1CS	Uncorrectable Error Counter Wrapped
11	RW1CS	ARM Succeeded—Emergency backup functionality is armed
12	RW1CS	ARM Failure—Failed to arm emergency backup functionality
13	RW1CS	Restore Backup Required—the primary media needs to be restored from its backup copy
14	RW1CS	Backup Failure—The media was not successfully copied to the backup media
15	RW1CS	Backup Succeeded—The media was successfully copied to the backup media
16	RW1CS	Restoration Failure—The backup media was not successfully copied to this media
17	RW1CS	Restoration Succeeded—The backup media was successfully copied to this media
19:18	RO	Read Media Endurance Level  0x0—Less than 80% of maximum read media endurance  0x1—Exceeded 80% of maximum read media endurance  0x2—Exceeded 90% of maximum read media endurance. Upon exceeding 90%, solutions should back up or migrate data in order to prevent possible data loss.  0x3—Reached 100% of maximum read media endurance. Upon reaching 100%, the component should be taken offline, and a <i>Standalone</i>

Bit Location	Access	Description
21:20		<i>Acknowledgment</i> (Reason = Media Endurance Exceeded) shall be returned to all new request packets that attempt to read this media.
21:20	RO	<p>Write Media Endurance Level</p> <p>0x0—Less than 80% of maximum write media endurance</p> <p>0x1—Exceeded 80% of maximum write media endurance</p> <p>0x2—Exceeded 90% of maximum write media endurance. Upon exceeding 90%, solutions should back up or migrate data in order to prevent possible data loss.</p> <p>0x3—Exceeded 100% of maximum write media endurance. Upon reaching 100%, the component should be taken offline, and a <i>Standalone Acknowledgment</i> (Reason = Media Endurance Exceeded) shall be returned to all new request packets that attempt to modify this media.</p>
22	RW1CS	Media Device Sparing Succeeded
23	RW1CS	Media Device Sparing Failed
24	RW1CS	Row Remapping Succeeded—the media controller successfully transparently remapped a media page
25	RW1CS	Row Remapping Failed
29:26	RO	Current Number of Spare Memory Devices
30	RW1C	Primary Media Maintenance Completed
31	RW1C	Deallocate Primary Range Completed
32	RO	Patrol Scrubbing In-progress
63:33	-	RsvdZ

### 8.19.2. Secondary Media Status

If a bit sub-field, then a value of 1b indicates the status / event / operating condition is true, and a value of 0b indicates that the status / event / operating condition is false.

Table 8-50: Secondary Media Status

Bit Location	Access	Description
0	RW1CS	Uninitialized—the media is in its default state. This may be due to a power-up event, a component reset event, or due to completion of a SE operation.
1	RW1CS	Media Initialization / Restoration Failed
2	RW1CS	Media Initialization / Restoration Succeeded

Bit Location	Access	Description
3	RW1CS	SE In-progress—the media is in a non-deterministic state
4	RW1CS	SE Completed—the media was successfully set to the expected state
5	RW1CS	SE Failed—the media is in a non-deterministic state
6	RW1CS	Patrol Scrubbing Completed—the component has successfully completed at least one patrol scrubbing cycle of the secondary media.
7	RW1CS	Fatal Media Error Containment—the component has detected an uncorrectable error, but is unable to isolate it to a specific media address.  If this media is volatile, all request packets shall continue to be validated and executed.  If this media is non-volatile, all request packets that modify the impacted media range (e.g., a Write request), shall not be executed, and a <i>Standalone Acknowledgment</i> (Reason = Fatal Media Error Containment Triggered) shall be returned.  A <i>Component Reset</i> (any type) shall clear Fatal Media Error Containment.
8	RW1CS	Logging Failed—one or more attempts to generate a log entry failed due to a lack of free log entries.
9	RW1CS	Correctable Error Counter Wrapped
10	RW1CS	Uncorrectable Error Counter Wrapped
12:11	RO	Read Media Endurance Level  0x0—Less than 80% of maximum read media endurance  0x1—Exceeded 80% of maximum read media endurance  0x2—Exceeded 90% of maximum read media endurance. Upon exceeding 90%, solutions should back up or migrate data in order to prevent possible data loss.  0x3—Exceeded 100% of maximum read media endurance. Upon reaching 100%, the component should be taken offline, and a <i>Standalone Acknowledgment</i> (Reason = Media Endurance Exceeded) shall be returned to all new request packets that attempt to read this media.
14:13	RO	Write Media Endurance Level  0x0—Less than 80% of maximum write media endurance  0x1—Exceeded 80% of maximum write media endurance  0x2—Exceeded 90% of maximum write media endurance. Upon exceeding 90%, solutions should back up or migrate data in order to prevent possible data loss.

Bit Location	Access	Description
		0x3—Exceeded 100% of maximum write media endurance. Upon reaching 100%, the component should be taken offline, and a <i>Standalone Acknowledgment</i> (Reason = Media Endurance Exceeded) shall be returned to all new request packets that attempt to modify this media.
15	RW1CS	Media Device Sparing Succeeded
16	RW1CS	Media Device Sparing Failed
17	RW1CS	Row Remapping Succeeded—the media controller successfully transparently remapped a memory row
18	RW1CS	Row Remapping Failed
22:19	RO	Current Number of Spare Memory Devices
23	RW1C	Secondary Media Maintenance Completed
24	RW1C	Deallocate Secondary Range Completed
25	RO	Patrol Scrubbing In-progress
63:26	-	RsvdZ

### 8.19.3. Primary Media CAP 1

Table 8-51: Primary Media CAP 1 Fields

Bit Location	Access	Description
6:0	RO	Read Latency Base—an unsigned integer used to calculate the worst-case Read Request execution latency associated with this media type
13:7	RO	Write Latency Base—an unsigned integer used to calculate the worst-case Write Request execution latency associated with this media type
17:14	RO	Latency Scale—Read and Write latency shall use the same scale. 0x0— $10^{-3}$ 0x1— $10^{-4}$ 0x2— $10^{-5}$ 0x3— $10^{-6}$ 0x4— $10^{-7}$ 0x5— $10^{-8}$ 0x6— $10^{-9}$ 0x7— $10^{-10}$ 0x8— $10^{-11}$ 0x9— $10^{-12}$ 0xA-0xF—Reserved

Bit Location	Access	Description
25:18	RO	<p>Read Endurance Base—an unsigned integer used to calculate the maximum Read endurance associated with this media type. The maximum read endurance is the maximum number of times the media can be read before it becomes unreliable. If zero, then Maximum Read Endurance shall be treated as infinite.</p> <p>Maximum Read Endurance = Read Endurance Base * Endurance Scale</p>
33:26	RO	<p>Write Endurance Base—an unsigned integer used to calculate the maximum Write endurance associated with this media type. The maximum write endurance is the maximum number of times the media can be written to before it becomes unreliable. If zero, then Maximum Write Endurance shall be treated as infinite.</p> <p>Maximum Write Endurance = Write Endurance Base * Endurance Scale</p>
37:34	RO	<p>Endurance Scale—Read and Write endurance shall use the same scale.</p> <p>0x0—<math>10^5</math>      0x1—<math>10^6</math>      0x2—<math>10^7</math>      0x3—<math>10^8</math>      0x4—<math>10^9</math>      0x5—<math>10^{10}</math>      0x6—<math>10^{11}</math>      0x7—<math>10^{12}</math>      0x8—<math>10^{13}</math>      0x9—<math>10^{14}</math>      0xA—<math>10^{15}</math>      0xB—<math>10^{16}</math>      0xC—<math>10^{17}</math>      0xD—<math>10^{18}</math>      0xE-0xF—Reserved</p>
47:38	RO	<p>Maximum Media power level (Max_Media_Power_Level) shall be a non-zero value.</p> <p>To calculate the maximum media power consumption:</p> <p>Max_Media_Power = (Max_Power_Level * MPWRS) Watts.</p>
50:48	RO	<p>Media Power Scale (MPWRS)</p> <p>0x0—1.0      0x1—0.1      0x2—0.01      0x3—0.001      0x4-0x7—Reserved</p>
54:51	RO	<p>Error Detection / Corrected Memory Range Size (in bytes)</p> <p>0x0—0</p>

Bit Location	Access	Description
58:55		0x1—8 0x2—16 0x3—32 0x4—64 0x5—128 0x6—256 0x7—512 0x8—1024 0x9—2048 0xA—4096 0xB—8192 0xC-0xF—Reserved
58:55	RO	Correctable Error Threshold—Number of correctable errors detected per given media device before the component spares the media device (if supported and a spare is available), and informs management of this condition. 0x0— $2^{10}$ 0x1— $2^{11}$ 0x2— $2^{12}$ 0x3— $2^{13}$ 0x4— $2^{14}$ 0x5— $2^{15}$ 0x6— $2^{16}$ 0x7— $2^{17}$ 0x8— $2^{18}$ 0x9— $2^{19}$ 0xA— $2^{20}$ 0xB-0xF—Reserved
62:59	RO	Uncorrectable Error Threshold—Number of uncorrectable errors detected per given media device before the component spares the media device (if supported and a spare is available), and informs management of this condition. 0x0— $2^3$ 0x1— $2^4$ 0x2— $2^5$ 0x3— $2^6$ 0x4— $2^7$ 0x5— $2^8$ 0x6— $2^9$ 0x7— $2^{10}$ 0x8— $2^{11}$ 0x9— $2^{12}$ 0xA— $2^{13}$ 0xB-0xF—Reserved

Bit Location	Access	Description
63	RO	Media Device Sparing Support 0b—Unsupported 1b—Supported
64	RO	Media Controller Demand Scrubbing Support 0b—Unsupported 1b—Supported
65	RO	Media Controller Patrol Scrubbing Support 0b—Unsupported 1b—Supported
66	RO	Poison Support 0b—Unsupported 1b—Supported
73:67	RO	Error Detection—maximum number of detectable bit errors per Error Detection / Corrected Memory Range Size. If 0x0, then the maximum number shall be 64.
80:74	RO	Error Correction—maximum number of correctable bit errors the component can correct per Error Detection / Corrected Memory Range Size. If 0x0, then the maximum number shall be 64.
81	RO	Dedicated Backup Power Support—Component uses a dedicated backup power source. 0b—Unsupported 1b—Supported
82	RO	Shared Backup Power Support—Component may share the backup power source with other components 0b—Unsupported 1b—Supported
83	RO	Fault Injection Support 0b—Unsupported 1b—Supported
84	RO	SE Zero Media Support 0b—Unsupported 1b—Supported
85	RO	SE Zero Media Range Support 0b—Unsupported 1b—Supported
86	RO	SE Cryptographic Key Support 0b—Unsupported

Bit Location	Access	Description
87		1b—Supported
	RO	SE Overwrite Media Support 0b—Unsupported 1b—Supported
88	RO	SE Vendor-Defined Support 0b—Unsupported 1b—Supported
89	RO	SE Vendor-Defined Range Support 0b—Unsupported 1b—Supported
90	RO	Fatal Media Error Containment Support 0b—Unsupported 1b—Supported
94:91	RO	Number of Spare Memory Devices
98:95	RO	Row Remapping Size—Indicates the number of rows impacted when a component transparently maps a new memory row in place of a bad memory row. 0x0—Unsupported capability 0x1—1 row 0x2—2 rows 0x3—4 rows 0x4—8 rows 0x5-0xF—Reserved
100:99	RO	Media Volatility—Indicates whether the media is volatile or non-volatile. If non-volatile, then it indicates whether a Write Persistent or a <i>Persistent Flush</i> should be used to ensure persistency. Software configures the corresponding <i>Requester Bank Structure</i> PRI CTL field. 0x0—Volatile Media 0x1—Non-Volatile Media. Use Write Persistent 0x2—Non-Volatile Media. Use <i>Persistent Flush</i> 0x3—Reserved
127:100	RO	RsvdZ

## 8.19.4. Secondary Media CAP 1

Table 8-52: Secondary Media CAP 1 Fields

Bit Location	Access	Description
6:0	RO	Read Latency Base—an unsigned integer used to calculate the worst-case Read Request execution latency associated with this media type
13:7	RO	Write Latency Base—an unsigned integer used to calculate the worst-case Write Request execution latency associated with this media type
17:14	RO	Media Latency Scale—Read and Write latency shall use the same scale. 0x0— $10^{-3}$ 0x1— $10^{-4}$ 0x2— $10^{-5}$ 0x3— $10^{-6}$ 0x4— $10^{-7}$ 0x5— $10^{-8}$ 0x6— $10^{-9}$ 0x7— $10^{-10}$ 0x8— $10^{-11}$ 0x9— $10^{-12}$ 0xA-0xF—Reserved
25:18	RO	Read Endurance Base—an unsigned integer used to calculate the maximum Read endurance associated with this media type. The maximum read endurance is the maximum number of times the media can be read before it becomes unreliable. If zero, then Maximum Read Endurance shall be treated as infinite.  Maximum Read Endurance = Read Endurance Base * Endurance Scale
33:26	RO	Write Endurance Base an unsigned integer used to calculate the maximum Write endurance associated with this media type. The maximum write endurance is the maximum number of times the media can be written to before it becomes unreliable. If zero, then Maximum Read Endurance shall be treated as infinite.  Maximum Write Endurance = Write Endurance Base * Endurance Scale
37:34	RO	Endurance Scale—Read and Write endurance shall use the same scale. 0x0— $10^5$ 0x1— $10^6$ 0x2— $10^7$ 0x3— $10^8$ 0x4— $10^9$ 0x5— $10^{10}$ 0x6— $10^{11}$ 0x7— $10^{12}$

Bit Location	Access	Description
		0x8— $10^{13}$ 0x9— $10^{14}$ 0xA— $10^{15}$ 0xB— $10^{16}$ 0xC— $10^{17}$ 0xD— $10^{18}$ 0xE-0xF—Reserved
47:38	RO	Maximum Media power level (Max_Media_Power_Level) shall be a non-zero value.  To calculate the maximum media power consumption, $\text{Max_Media_Power} = (\text{Max_Power_Level} * \text{MPWRS}) \text{ Watts.}$
50:48	RO	Media Power Scale (MPWRS)  0x0—1.0 0x1—0.1 0x2—0.01 0x3—0.001 0x4-0x7—Reserved
54:51	RO	Error Detection / Corrected Memory Range Size (in bytes)  0x0—0 0x1—8 0x2—16 0x3—32 0x4—64 0x5—128 0x6—256 0x7—512 0x8—1024 0x9—2048 0xA—4096 0xB—8192 0xC-0xF—Reserved
58:55	RO	Correctable Error Threshold—Number of correctable errors detected per given media device before the component spares the media device (if supported and a spare is available), and informs management of this condition.  0x0— $2^{10}$ 0x1— $2^{11}$ 0x2— $2^{12}$ 0x3— $2^{13}$ 0x4— $2^{14}$ 0x5— $2^{15}$ 0x6— $2^{16}$ 0x7— $2^{17}$

Bit Location	Access	Description
		0x8— $2^{18}$ 0x9— $2^{19}$ 0xA— $2^{20}$ 0xB—0xF—Reserved
62:59	RO	Uncorrectable Error Threshold—Number of uncorrectable errors detected per given media device before the component spares the media device (if supported and a spare is available), and informs management of this condition.  0x0— $2^3$ 0x1— $2^4$ 0x2— $2^5$ 0x3— $2^6$ 0x4— $2^7$ 0x5— $2^8$ 0x6— $2^9$ 0x7— $2^{10}$ 0x8— $2^{11}$ 0x9— $2^{12}$ 0xA— $2^{13}$ 0xC—0xF—Reserved
63	RO	Media Device Sparing Support  0b—Unsupported 1b—Supported
64	RO	Media Controller Demand Scrubbing Support  0b—Unsupported 1b—Supported
65	RO	Media Controller Patrol Scrubbing Support  0b—Unsupported 1b—Supported
66	RO	Poison Support  0b—Unsupported 1b—Supported
73:67	RO	Error Detection—maximum number of bit errors the component can detect per Error Detection / Corrected Memory Range Size. If 0x0, then the maximum number shall be 64.
80:74	RO	Error Correction—maximum number of correctable bit errors the component can correct per Error Detection / Corrected Memory Range Size. If 0x0, then the maximum number shall be 64.

Bit Location	Access	Description
81	RO	Dedicated Backup Power Support—Component uses a dedicated backup power source. 0b—Unsupported 1b—Supported
82	RO	Shared Backup Power Support—Component may share the backup power source with other components 0b—Unsupported 1b—Supported
83	RO	Fault Injection Support 0b—Unsupported 1b—Supported
84	RO	SE Zero Media Support 0b—Unsupported 1b—Supported
85	RO	SE Zero Media Range Support 0b—Unsupported 1b—Supported
86	RO	SE Cryptographic Key Support 0b—Unsupported 1b—Supported
87	RO	SE Overwrite Media Support 0b—Unsupported 1b—Supported
88	RO	SE Vendor-Defined Support 0b—Unsupported 1b—Supported
89	RO	SE Vendor-Defined Support 0b—Unsupported 1b—Supported
90	RO	Fatal Media Error Containment Support 0b—Unsupported 1b—Supported
91	RO	Emergency Backup Support—if supported, then the secondary media is a non-addressable resource. 0b—Unsupported 1b—Supported
92	RO	Secondary Backup Support—Indicates if the secondary media provides primary media backup services. The secondary may perform on-demand backup

Bit Location	Access	Description
93		<p>services, or perform emergency backup services when conditions warrant. If supported, then the secondary media is a non-addressable resource.</p> <p>0b—Unsupported 1b—Supported</p>
97:94	RO	<p>Secondary Media Caching Support—Indicates if the secondary media is capable of acting as a cache for the primary media. For example, if the secondary media supports a higher-speed, higher-endurance media compared to the primary media, then the secondary media can be configured to operate as a cache. If supported, then the secondary media is a non-addressable resource.</p> <p>0b—Unsupported 1b—Supported</p>
101:98	RO	<p>Number of Spare Memory Devices</p>
104:102	RO	<p>Row Remapping Size—Indicates the number of rows impacted when a component transparently maps a new memory row in place of a bad memory row.</p> <p>0x0—Unsupported capability 0x1—1 row 0x2—2 rows 0x3—4 rows 0x4—8 rows 0x5-0xF—Reserved</p>
127:105	RO	<p>Media Volatility—Indicates whether the media is volatile or non-volatile. If non-volatile, then it indicates whether a Write Persistent or a <i>Persistent Flush</i> should be used to ensure persistency. Software configures the corresponding <i>Requester Bank Structure</i> SEC CTL field.</p> <p>0x0—Volatile Media 0x1—Non-Volatile Media. Use Write Persistent 0x2—Non-Volatile Media. Use <i>Persistent Flush</i> 0x3—Reserved</p>
	RO	Reserved

### 8.19.5. Primary Media CAP 1 Control

Table 8-53: Primary Media CAP 1 Control

Bit Location	Access	Description
0	RW	Correctable Error Reporting Enabled—If enabled, then response packets with a Reason field shall set (Reason = Data Correctable Error Warning) if an uncorrectable error was detected during request packet execution.

Bit Location	Access	Description
1		0b—Disabled 1b—Enabled
1	RW	Uncorrectable Error Reporting Enabled—if enabled, then response packets with a Reason field shall set (Reason = Data Uncorrectable Error Detected) if an uncorrectable error was detected during request packet execution. 0b—Disabled 1b—Enabled
2	RW	Demand Scrubbing Enabled—if enabled, then upon detection of a media error, the media controller shall perform demand scrubbing on the impacted media ranged. 0b—Enabled 1b—Disabled
3	RW	Patrol Scrubbing Enabled 0b—Enabled 1b—Disabled
7:4	RW	Patrol Scrubbing Frequency—maximum time to patrol scrub the primary media in its entirety. 0x0—48 hours 0x1—36 hours 0x2—24 hours 0x3—12 hours 0x4—8 hours 0x5—4 hours 0x6—2 hours 0x7—1 hour 0x8-0xF—Reserved
8	RW	Media Device Sparing Enabled 0b—Disabled 1b—Enabled
9	RW	Poison Forwarding Enabled—if enabled, then response packets with a Reason field shall if poisoned data was detected during request packet execution. 0b—Disabled 1b—Enabled
10	RW	Fault Injection Enabled 0b—Disabled 1b—Enabled
11	RW	Error and Event Notification 0b—Return error reasons in response packets 1b—Do not return error reasons in response packets. Independently inform management of all errors and events.

Bit Location	Access	Description
		This field should be configured the same in the Primary Media CAP 1 Control and Secondary CAP Media CAP 1 Control structures.
13:12	RW	<p>Management Event Notification—Indicates whether to inform management based on the type of event detected. If Management Event Notification Method indicates that an <i>Unsolicited Event (UE) Packet</i> is to be transmitted, then the component shall generate a Component Media Structure Plog Event based on the PLog entry information (see <i>Primary and Secondary Media Log Entry Fields</i>).</p> <p>0x0—No management event notification 0x1—Notify management of new critical events 0x2—Notify management of new critical and caution events. 0x3—Notify management of new critical, caution, and non-critical events.</p>
16:14	RW	<p>Management Event Notification Method—Indicates the method(s) used to signal management of an event. The corresponding interrupt and / or Media MGR CID and Media MGR SID shall be configured prior to setting this field to a non-zero value.</p> <p>0x0—No signal is generated 0x1—Component-local interrupt 0 0x2—Component-local interrupt 1 0x3—Component-local interrupts 0 and 1 0x4—<i>Unsolicited Event (UE) Packet</i> 0x5—Component-local interrupt 0 and an <i>Unsolicited Event (UE) Packet</i> 0x6—Component-local interrupt 1 and an <i>Unsolicited Event (UE) Packet</i> 0x7—Component-local interrupts 0 and 1, and an <i>Unsolicited Event (UE) Packet</i></p>
17	RW	<p>Enable Emergency Backup Functionality—If Enabled, then the primary media shall be backed up to the secondary media when emergency back-up is triggered.</p> <p>0b—Disable 1b—Enable (Armed)</p>
19:18	RW	<p>Event Logging Level—Indicates the level of events to log</p> <p>0x0—Log only Critical events 0x1—Log Critical and Caution events 0x2—Log Critical, Caution, and Non-critical events 0x3—Reserved</p>
23:20	RW	<p>Primary Sanitize and Erase—When invoked (see Component Media Control), this field indicates the type of Sanitize and Erase operation to perform on the primary media.</p> <p>0x0—SE Fast Zero Media 0x1—SE Fast Zero Media Range 0x2—SE Zero Media 0x3—SE Cryptographic Key</p>

Bit Location	Access	Description
26:24		<p>0x4—SE Overwrite Media (no inversion)      0x5—SE Overwrite Media (alternating inversion)      0x6—SE Vendor-defined      0x7—SE Vendor-defined Range      0x8-0xF—Reserved</p> <p>Only one Primary Media Sanitize and Erase operation may be outstanding at any given time.</p> <p>A SE Fast Zero Media Range or a SE Vendor-defined Range operation shall not be invoked while any other operation that uses the Primary Region Length and Primary Region Address fields is outstanding.</p> <p>Reads of this field shall return 0x0.</p>
26:24	RW	<p>SE Overwrite Media Count—indicates the number of times the SE Overwrite Pattern is to be written.</p> <p>0x0—1      0x1—8      0x2—16      0x3—32      0x4-0x7—Reserved</p>
27	RW	<p>Deallocate Primary Range—When set to 1b, the media controller deallocates Primary Region Length bytes starting at Primary Region Address.</p> <p>A Deallocate Primary Range shall not be invoked while any other operation that uses the Primary Region Length and Primary Region Address fields is outstanding.</p> <p>Reads of this field shall return 0b.</p>
63:28	-	RsvdP

### 8.19.6. Secondary Media CAP 1 Control

Table 8-54: Secondary Media CAP 1 Control

Bit Location	Access	Description
0	RW	<p>Correctable Error Reporting Enabled—if enabled, then response packets with a Reason field shall set (Reason = Data Correctable Error Warning) if an uncorrectable error was detected during request packet execution.</p> <p>0b—Disabled      1b—Enabled</p>
1	RW	Uncorrectable Error Reporting Enabled—if enabled, then response packets with a Reason field shall set (Reason = Data Uncorrectable Error Detected) if an uncorrectable error was detected during request packet execution.

Bit Location	Access	Description
2		0b—Disabled 1b—Enabled
2	RW	Demand Scrubbing Enabled—if enabled, then upon detection of a media error, the media controller shall perform demand scrubbing on the impacted media ranged. 0b—Disabled 1b—Enabled
3	RW	Patrol Scrubbing Enabled 0b—Disabled 1b—Enabled
7:4	RW	Patrol Scrubbing Frequency—maximum time to patrol scrub the secondary media in its entirety. 0x0—48 hours 0x1—36 hours 0x2—24 hours 0x3—12 hours 0x4—8 hours 0x5—4 hours 0x6—2 hours 0x7—1 hour 0x8-0xF—Reserved
8	RW	Media Device Sparing Enabled 0b—Disabled 1b—Enabled
9	RW	Poison Forwarding Enabled—if enabled, then response packets with a Reason field shall indicate if poisoned data was detected during request packet execution. 0b—Disabled 1b—Enabled
10	RW	Fault Injection Enabled 0b—Disabled 1b—Enabled
11	RW	Error and Event Notification 0b—Return error reasons in response packets 1b—Do not return error reasons in response packets. Independently inform management of all errors and events. This field should be configured the same in the Primary Media CAP 1 Control and Secondary Media CAP 1 Control structures.
13:12	RW	Management Event Notification—Indicates whether to inform management based on the type of event detected. If Management Event Notification

Bit Location	Access	Description
16:14		<p>Method indicates that an <i>Unsolicited Event (UE) Packet</i> is to be transmitted, then the component shall generate a Component Media Structure Plog Event based on the PLog entry information (see <i>Primary and Secondary Media Log Entry Fields</i>).</p> <p>0x0—No management event notification 0x1—Notify management of new critical events 0x2—Notify management of new critical and caution events. 0x3—Notify management of new critical, caution, and non-critical events.</p>
	RW	<p>Management Event Notification Method—Indicates the method(s) used to signal management of an event. The corresponding interrupt and / or Media MGR CID and Media MGR SID shall be configured prior to setting this field to a non-zero value.</p> <p>0x0—No signal is generated 0x1—Component-local interrupt 0 0x2—Component-local interrupt 1 0x3—Component-local interrupts 0 and 1 0x4—<i>Unsolicited Event (UE) Packet</i> 0x5—Component-local interrupt 0 and an <i>Unsolicited Event (UE) Packet</i> 0x6—Component-local interrupt 1 and an <i>Unsolicited Event (UE) Packet</i> 0x7—Component-local interrupts 0 and 1, and an <i>Unsolicited Event (UE) Packet</i></p>
18:17	RW	<p>Event Logging Level—Indicates the level of events to log</p> <p>0x0—Log only Critical events 0x1—Log Critical and Caution events 0x2—Log Critical, Caution, and Non-critical events 0x3—Reserved</p>
19	RW	<p>Secondary Backup Enable—Places the secondary media into a backup service mode. When enabled, the secondary media shall not be addressable and the media shall not be used as a cache for the primary media.</p> <p>0b—Disabled 1b—Enabled</p>
20	RW	<p>Secondary Media Caching Enable—Places the secondary media into a primary media cache mode. When enabled, the secondary media shall not be addressable and the media shall not be used as a backup for the primary media.</p> <p>0b—Disable 1b—Enabled</p>
24:21	RW	<p>Secondary Sanitize and Erase—When invoked (see Component Media Control), this field indicates the type of Sanitize and Erase operation to perform on the secondary media.</p> <p>0x0—SE Fast Zero Media 0x1—SE Fast Zero Media Range</p>

Bit Location	Access	Description
		<p>0x2—SE Zero Media      0x3—SE Cryptographic Key      0x4—SE Overwrite Media (no inversion)      0x5—SE Overwrite Media (alternating inversion)      0x6—SE Vendor-defined      0x7—SE Vendor-defined Range      0x8-0xF—Reserved</p> <p>Only one Secondary Media Sanitize and Erase operation may be outstanding at any given time.</p> <p>A SE Fast Zero Media Range or a SE Vendor-defined Range operation shall not be invoked while any other operation that uses the Secondary Region Length and Secondary Region Address fields is outstanding.</p> <p>Reads of this field shall return 0x0.</p>
27:25	RW	<p>SE Overwrite Media Count—indicates the number of times the SE Overwrite Pattern is to be written.</p> <p>0x0—1      0x1—8      0x2—16      0x3—32      0x4-0x7—Reserved</p>
28	RW	<p>Deallocate Secondary Range—When set to 1b, the media controller deallocates Secondary Region Length bytes starting at Secondary Region Address.</p> <p>A Deallocate Secondary Range shall not be invoked while any other operation that uses the Secondary Region Length and Secondary Region Address fields is outstanding.</p> <p>Reads of this field shall return 0b.</p>
63:29	-	RsvdP

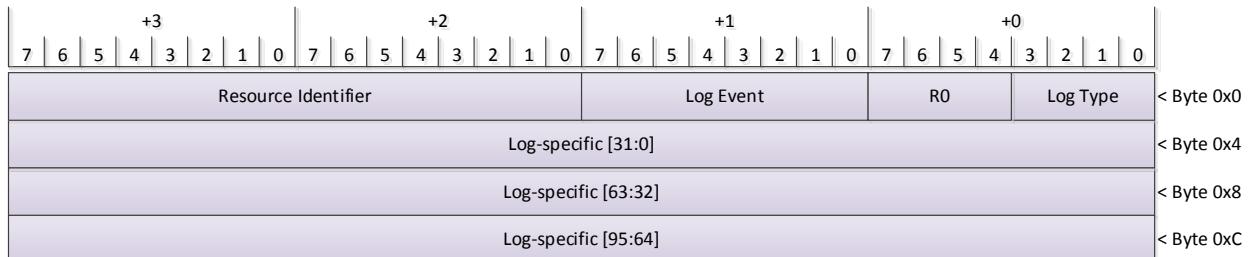


Figure 8-16: Primary and Secondary Media Log Entry Format

The following are the log entry requirements:

- If logging is supported, then all log entry fields shall be mandatory.
- The Log Type field shall have read-write access. All other fields shall have read-only access.

- All log entry fields contain the associated state at the time the log entry was made.
- The Severity field in *Primary and Secondary Media Log Entry Fields* is used to determine if the component should notify management when the corresponding event is detected.
- Unless specifically stated in the event entry's description, the Log-specific field shall be Reserved.

Table 8-55: Primary and Secondary Media Log Entry Fields

Field Name	Size (bits)	Value	Severity	Description
<b>Log Type</b>	4	-	-	<p>Log Type—Unless otherwise indicated by the Log Type encoding, the log entry format shall use the <i>Primary and Secondary Media Log Entry Format</i>.</p> <p>0x0—Free Entry—For each log entry, the component shall set the Log Type to Free Entry upon power-up / <i>Component Reset</i> (any type). The component sets Log Type to a non-zero value when it consumes a log entry. Software shall set the Log Type to Free Entry once it completes processing the log entry.</p> <p>0x1—Component Media Log Entry—Resource Identifier may contain event-specific information; else is Reserved</p> <p>0x2—Bank Entry—Resource Identifier contains the identifier of the impacted bank.</p> <p>0x3—Media Device Entry—Resource Identifier contains the identifier of the impacted media device.</p> <p>0x4-0xE—Reserved</p> <p>0xF—Vendor-defined Entry. If indicated, bytes 4-15 shall be vendor-defined.</p>
<b>Resource Identifier</b>	16	-	-	<p>Identifier of the associated resource.</p> <p>Unless explicitly stated in the event entry, the Resource Identifier shall be Reserved.</p>
<b>Log Event</b>	8	-	-	<p>Indicates what event was logged and how to interpret the Resource Identifier and Log-specific fields.</p>
	0x0	Critical		<p>Unstable / Insufficient Power</p> <p>Log Type = 1</p>
	0x1	Critical		<p>Unexpected Power Loss—e.g., voltage regulator or backup power capacitor failure</p> <p>Log Type = 1</p>
	0x2	Non-critical		<p>Battery-Capacitor Present</p>

Field Name	Size (bits)	Value	Severity	Description
				Log Type = 1
		0x3	Non-critical	Battery-Capacitor Charged Log Type = 1
		0x4	Caution	Battery-Capacitor Not Ready Log Type = 1
		0x5	Caution	Power-up / Reset Initialization Completed Without Backup Power Log Type = 1
		0x6	Caution	Non-critical Media Device Error Resource Identifier identifies the impacted media device, and the Log-specific field contains the row number of the impacted memory range. Log Type = 3
		0x7	Critical	Critical Media Device Error Resource Identifier identifies the impacted media device, and the Log-specific field contains the row number of the impacted memory range. Log Type = 3
		0x8	Critical	Fatal Media Device Error Resource Identifier identifies the impacted media device, and the Log-specific field contains the row number of the impacted memory range. Log Type = 3
		0x9	Caution	Uncorrectable Backup Error Log-specific field byte 0 contains one of the following codes: 0x0—Insufficient non-volatile memory storage to successfully complete backup 0x1—Internal component or execution error prevented backup completion 0x2-0xFF—Reserved Log Type = 1 Log-specific field bytes 1-15 may contain additional information.

Field Name	Size (bits)	Value	Severity	Description
		0xA	Caution	<p>Uncorrectable Restore Error</p> <p>Log-specific field byte 0 contains one of the following codes:</p> <ul style="list-style-type: none"> <li>0x0—No restoration data is available</li> <li>0x1—Unknown or corrupted meta data</li> <li>0x2-0xFF—Reserved</li> </ul> <p>Log Type = 1</p> <p>Log-specific field bytes 1-15 may contain additional information.</p>
		0xB	Caution	<p>Uncorrectable Erase Error</p> <p>Log-specific field may contain additional details.</p> <p>Log Type = 1</p>
		0xC	Caution	<p>Backup energy source error</p> <p>Log-specific field contains the cause of this error:</p> <ul style="list-style-type: none"> <li>0x0—No backup energy source detected</li> <li>0x1—Lost previously detected energy source</li> <li>0x2-0xFF—Reserved</li> </ul> <p>Log Type = 1</p>
		0xD	Caution	<p>Backup Energy Source Not Charged Error</p> <p>Log-specific field contains the cause of this error:</p> <ul style="list-style-type: none"> <li>0x0—Backup energy source component failure</li> <li>0x1—Unable to complete backup energy charging</li> <li>0x3-0xFF—Reserved</li> </ul> <p>Log Type = 1</p>
		0xE	Caution	<p>Reached 80% of Maximum Read Media Endurance</p> <p>Log Type = 1</p> <p>Resource Identifier = 0x1 indicates Primary Media</p> <p>Resource Identifier = 0x2 indicates Secondary Media</p>
		0xF	Critical	<p>Reached 90% of Maximum Read Media Endurance</p> <p>Log Type = 1</p> <p>Resource Identifier = 0x1 indicates Primary Media</p>

Field Name	Size (bits)	Value	Severity	Description
				Resource Identifier = 0x2 indicates Secondary Media
		0x10	Critical	Reached or Exceeded 100% of Maximum Read Media Endurance Log Type = 1 Resource Identifier = 0x1 indicates Primary Media Resource Identifier = 0x2 indicates Secondary Media
		0x11	Caution	Reached 80% of Maximum Write Media Endurance Log Type = 1 Resource Identifier = 0x1 indicates Primary Media Resource Identifier = 0x2 indicates Secondary Media
		0x12	Critical	Reached 90% of Maximum Write Media Endurance Log Type = 1
		0x13	Critical	Reached or Exceeded 100% of Maximum Write Media Endurance. Log Type = 1 Resource Identifier = 0x1 indicates Primary Media Resource Identifier = 0x2 indicates Secondary Media
		0x14	Critical	Media Controller Internal error Log-specific field may contain vendor-specific information. Log Type = 1
		0x15	Non-critical	Uncorrectable Error Detected Resource Identifier identifies the impacted bank, Log-specific[15:0] indicates the number of rows, and Log-specific[95:32] indicates the starting row number of the impacted memory range. This event is generated if the component does not support poison or the uncorrectable error cannot be poisoned. Log Type = 2
		0x16	Non-critical	Poison Event Resource Identifier identifies the impacted bank, Log-specific[15:0] indicates the number of rows, and Log-

Field Name	Size (bits)	Value	Severity	Description
				<p>specific[95:32] indicates the starting row number of the impacted memory range.</p> <p>This event is generated if the component supports poison and the uncorrectable error was successfully poisoned.</p> <p>Log Type = 2</p>
		0x17	Caution	<p>Post-package Repair (PPR) Event</p> <p>Log Type = 3</p> <p>Resource Identifier identifies the impacted media device, Log-specific[15:0] indicates the number of rows, and Log-specific[95:32] indicates the starting row number of the impacted memory range.</p>
		0x18	Critical	<p>Spare Media Device Consumed</p> <p>Log Type = 1</p> <p>Resource Identifier = 0x1 indicates Primary Media</p> <p>Resource Identifier = 0x2 indicates Secondary Media</p> <p>Log-specific contains the consumed spare media device number (0x0-0xF)</p>
		0x19	Critical	<p>Last Spare Media Device Consumed</p> <p>Log Type = 1</p> <p>Resource Identifier = 0x1 indicates Primary Media</p> <p>Resource Identifier = 0x2 indicates Secondary Media</p>
		0x1A	Critical	<p>Fatal Media Error Containment Triggered</p> <p>Log Type = 1</p> <p>Resource Identifier = 0x1 indicates Primary Media</p> <p>Resource Identifier = 0x2 indicates Secondary Media</p>
		0x1B	Critical	<p>Backup Failure Event</p> <p>Log Type = 1</p> <p>Resource Identifier = 0x1 indicates Primary Media</p> <p>Resource Identifier = 0x2 indicates Secondary Media</p> <p>Log-specific field may contain vendor-specific information.</p>

Field Name	Size (bits)	Value	Severity	Description
		0x1C	Non-critical	<p>Backup Succeeded Event</p> <p>Log Type = 1</p> <p>Resource Identifier = 0x1 indicates Primary Media</p> <p>Resource Identifier = 0x2 indicates Secondary Media</p>
		0x1D	Critical	<p>Restoration Failure Event</p> <p>Log Type = 1</p> <p>Resource Identifier = 0x1 indicates Primary Media</p> <p>Resource Identifier = 0x2 indicates Secondary Media</p> <p>Log-specific field may contain vendor-specific information.</p>
		0x1E	Non-critical	<p>Restoration Succeeded Event</p> <p>Log Type = 1</p> <p>Resource Identifier = 0x1 indicates Primary Media</p> <p>Resource Identifier = 0x2 indicates Secondary Media</p>
		0x1F	Non-critical	<p>Row Remapping Event</p> <p>Resource Identifier identifies the impacted bank, and the Log-specific field contains the row number of the impacted memory range.</p> <p>Log Type 1</p>
		0x20	Non-critical	<p>Reached 90% Consumption of Spare Row Remapping Resources Event</p> <p>Log Type = 1</p> <p>Resource Identifier = 0x1 indicates Primary Media</p> <p>Resource Identifier = 0x2 indicates Secondary Media</p>
		0x21	Critical	<p>Reached 100% Consumption of Spare Row Remapping Resources Event</p> <p>Log Type = 1</p> <p>Resource Identifier = 0x1 indicates Primary Media</p> <p>Resource Identifier = 0x2 indicates Secondary Media</p>
		0x22	Non-critical	<p>Media Maintenance Required Event—Indicates the media requires maintenance (e.g., garbage collection) in order to continue to function correctly.</p> <p>This event is applicable only if media maintenance</p>

Field Name	Size (bits)	Value	Severity	Description
Log-specific				<p>had been disabled and media is approaching the maximum time it can safely delay maintenance.</p> <p>Log Type = 1</p> <p>Resource Identifier = 0x1 indicates Primary Media</p> <p>Resource Identifier = 0x2 indicates Secondary Media</p>
				<p>0x23</p> <p>Non-critical</p> <p>Media Maintenance Override Event—Indicates the component has overridden its configuration to perform necessary media maintenance (e.g., garbage collection). This event is applicable only if media maintenance had been disabled and the media controller cannot delay maintenance required to function correctly. The media controller shall perform maintenance only for the corresponding media's minimum maintenance time.</p> <p>Log Type = 1</p> <p>Resource Identifier = 0x1 indicates Primary Media</p> <p>Resource Identifier = 0x2 indicates Secondary Media</p>
		0x24	Caution	<p>Media Spare Range Event—Indicates the component spared a block of memory. Log-specific[63:0] is the starting address of the spared region (address relative to byte 0 of the impacted media), and Log-specific[95:64] indicates the spared region's length in bytes.</p> <p>Log Type 1</p> <p>Resource Identifier = 0x1 indicates Primary Media</p> <p>Resource Identifier = 0x2 indicates Secondary Media</p>
				<p>0x24-0xFF</p> <p>-</p> <p>Reserved</p>
	96	-	-	<p>Log-specific field. For example, this field may contain the row number associated with this entry if the log is for a specific memory range.</p> <p>Unless the Log-specific field's contents are explicitly stated in the Log Event description, then this field shall be Reserved.</p> <p>Log-specific data is contiguously placed starting at byte 0, bit 0 of the Log-specific field. Unused bits shall be Reserved.</p>

Field Name	Size (bits)	Value	Severity	Description
RO	4	-	-	RsvdP

The Fault injection field shall be as specified in *Component Media Structure Fault Injection Field*. If Fault Injection Bit<sub>K</sub> == 1b, then the component shall initiate the corresponding error processing as though the error was detected. Unsupported Fault Injection bits shall be hardwired to 0b. Reads of this field shall return zero. These bits correspond to the Media Status field used in log entries.

5

Table 8-56: Component Media Structure Fault Injection Field

Bit Location	Default	Access	Description
0	0b	RW	Unstable / Insufficient Power
1	0b	RW	Unexpected Power Loss
2	0b	RW	Battery-Capacitor Present
3	0b	RW	Battery-Capacitor Charged
4	0b	RW	Battery-Capacitor Not Ready
5	0b	RW	Power-up / reset Initialization completed without backup power
6	0b	RW	Non-critical media device error. The component may use any valid Bank / Media Device Identifier to emulate the impacted media device, and the component may use any valid corresponding row number to emulate the impacted memory range.  The component shall fill in the corresponding log entry Bank / Media Device Identifier and the Log-specific fields with the corresponding values.
7	0b	RW	Critical media device error. The component may use any valid Bank / Media Device Identifier to emulate the impacted media device, and the component may use any valid corresponding row number to emulate the impacted memory range.  The component shall fill in the corresponding log entry Bank / Media Device Identifier and the Log-specific fields with the corresponding values.
8	0b	RW	Fatal media device error. The component may use any valid Bank / Media Device Identifier to emulate the impacted media device, and the component may use any valid corresponding row number to emulate the impacted memory range.  The component shall fill in the corresponding log entry Bank / Media Device Identifier and the Log-specific fields with the corresponding values.

Bit Location	Default	Access	Description
9	0b	RW	Uncorrectable backup error
10	0b	RW	Uncorrectable restore error
11	0b	RW	Uncorrectable erase error
12	0b	RW	Energy source error
13	0b	RW	Energy source not charged error
14	0b	RW	Reached 80% of maximum read media endurance
15	0b	RW	Reached 90% of maximum read media endurance
16	0b	RW	Reached or exceeded 100% of maximum read media endurance
17	0b	RW	Reached 80% of maximum write media endurance
18	0b	RW	Reached 90% of maximum write media endurance
19	0b	RW	Reached or exceeded 100% of maximum write media endurance
20	0b	RW	Media controller internal error
21	0b	RW	Poison event. The component may use any valid Bank / Media Device Identifier to emulate the impacted media device, and the component may use any valid corresponding row number to emulate the impacted memory range.  The component shall fill in the corresponding log entry Bank / Media Device Identifier and the Log-specific fields with the corresponding values.
22	0b	RW	Last spare media device consumed
23	0b	RW	Fatal Media Error Containment Triggered
24	0b	RW	Backup Failure event
25	0b	RW	Backup Succeeded event
26	0b	RW	Restoration Failure event
27	0b	RW	Restoration Succeeded event
28	0b	RW	Row Remapping event
29	0b	RW	Reached 90% consumption of spare row remapping resources event
30	0b	RW	Reached 100% consumption of spare row remapping resources event
31	0b	RW	Media Maintenance Required Event
32	0b	RW	Media Maintenance Override Event
63:33	0b	RW	Reserved

## 8.20. P2P-Core Encoded Structures

P2P-Core encoded structures are used to configure Requesters (e.g., a SoC) and Responders that support the P2P-Core OpClass in point-to-point and daisy-chain topologies.

- A *Requester Bank Structure* is used to describe Responder memory components. This structure enables a Requester to target request packets to a specific Responder within a point-to-point or daisy-chain topology and to target a given logical bank in a specific Responder.
- A *Responder Bank Structure* is used to describe the number of logical banks and logical rows associated with each supported media type and to configure the encoded Tag subrange used to determine if a request packet corresponds to the Responder and, if so, to target a specific logical bank. The *Component Media Structure* provides additional media details, and complements the *Responder Bank Structure*.

### 8.20.1. Responder Bank Structure

The following are the features and requirements associated with the Responder Bank structure:

- Any Responder that supports the P2P-Core OpClass and contains addressable media may support the Responder Bank structure.
- The Responder Bank structure shall use the *Responder Bank Structure Format*.
- If a Responder memory component supports zero-based addressing (*Core Structure Component CAP 1 Address Field Interpretation == 0b*), then:
  - A Responder memory component may support multiple logical banks.
  - The total number of logical banks shall be  $2^N$  where  $1 \leq N \leq 13$ .
    - Total Logical Banks =  $2^{\text{PRI Max Banks}} + 2^{\text{SEC Max Banks}}$ 
      - Where  $2^{\text{PRI-Max-Banks}}$  is the number of logical banks associated with the primary media, and  $2^{\text{SEC-Max-Banks}}$  is the number of logical banks associated with the secondary media. If a secondary media is unsupported or is not addressable (e.g., the secondary media is used as a back-up for the primary media), then all logical banks shall be associated with the primary media.
    - Logical banks associated with the primary media shall be sequentially numbered 0 to  $(2^{\text{PRI Max Banks}} - 1)$
    - Logical banks associated with the secondary media shall be sequentially numbered  $2^{\text{PRI-Max-Banks}}$  to  $(\text{Total Logical Banks} - 1)$
  - All logical banks associated with a given media shall have the same attributes:
    - All logical banks associated with the primary media shall be the same size.
    - All logical banks associated with the secondary media shall be the same size.
    - The maximum size of a logical bank shall be  $2^{44}$  bytes.
      - The combination of logical bank size and multiple logical banks enables a Responder to support a maximum of  $2^{57}$  bytes of addressable resources.
  - Each logical bank shall contain a set of logical rows.
    - All logical rows associated with the primary media shall be the same size.
    - All logical rows associated with the secondary media shall be the same size.
  - Each logical bank shall be associated with an encoded Tag subrange containing  $2^{\text{Max-BTags}}$  tags, where  $2^{\text{Max-BTags}}$  is the maximum number of request packets per logical bank.

- To determine if a request packet targets a Responder and subsequently, a given logical bank, the following steps are taken:
  - Management calculates the number of Tags that can be associated with each Responder:
    - $\text{Responder}_k \text{ Tags} = \text{Total Logical Banks} * 2^{\text{Max-BTags}}$
  - Management uses the maximum  $\text{Responder}_k \text{ Tags}$  across all Responders within the daisy-chain to determine the CTag Mask field. The CTag Mask field is used to partition the Tag Space into equally-sized ranges, where each partition is associated with a single Responder.
    - For example, if the maximum  $\text{Responder}_k \text{ Tags}$  value is 1024, then the maximum number of Responders per daisy-chain is 8 ( $2^{13} \text{ DIV } 2^{10}$ ). The Responder's CTag Mask field is configured to 3 to indicate the upper 3 bits within the Tag field are used to determine if a request packet targets a given Responder.
    - Each Responder is uniquely identified by configuring the CTag ID field. The upper CTag Mask bits within the CTag ID field are set to the corresponding identifier. In the prior example, the upper 3 bits are set to a value of 0-7.
    - In a daisy-chain topology, if the upper CTag Mask bits of the request packet's Tag do not match the upper CTag Mask bits within the CTag ID field, then the Responder forwards the request packet to the next component in the daisy-chain or returns an error if no such component is provisioned.
    - If a request packet targets Control Space, then the Tag is encoded only to identify the Responder; logical bank encoding shall not be performed (Control Space does not support logical banks).
- To identify a logical bank, the Responder masks off the upper CTag Mask bits and the lower Max-BTags bits, and uses the remaining bits to identify the logical bank. For example, if CTag Mask equals 3 and Max-BTags equals 5, then the logical bank number is a value between 0 to  $(2^5 - 1)$  inclusive, i.e., the middle 5 bits within the encoded Tag.
- Once the logical bank is identified, to locate the target data, the Responder treats the request packet's Address field as an offset within the identified logical bank.
  - If the logical bank size is less than  $2^{44}$  bytes, then the unused upper bits of the request packet's Address field shall be Reserved.
- If a Responder supports non-zero-based addressing then (*Core Structure Component CAP 1 Address Field Interpretation == 1b*):
  - The Responder shall not support an addressable secondary media (SEC-Max-Banks and SEC-LRSZ shall be 0x0).
  - The maximum size of the primary media shall be  $2^{44}$  bytes.
  - The Responder shall be responsible for decoding a request packet's Address field into a Responder-local resources.
  - The Responder shall be responsible for validating the request packet's Tag field to ensure that the request packet is for this Responder.

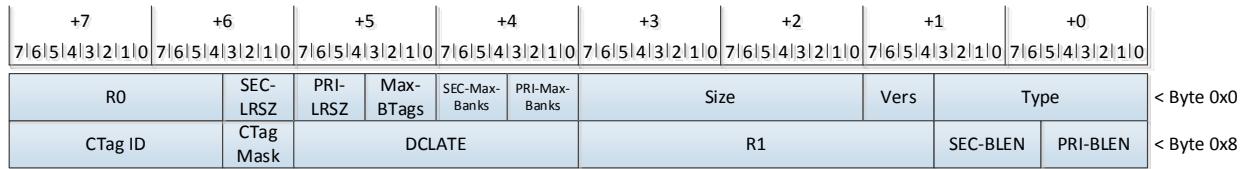


Figure 8-17: Responder Bank Structure Format

Table 8-57: Responder Bank Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description			
<b>Version (Vers)</b>	4	0x1	M	RO	Structure Version			
<b>PRI-Max-Banks</b>	4	-	M	RO	Primary Media Max Banks—used to indicate the number of primary media memory banks provisioned. Number of primary media banks = $2^{\text{PRI-Max-Banks}}$ PRI-Max-Banks shall be value such that the following is true: $0 < \text{PRI-Max-Banks} < 0xD$ PRI-Max-Banks values 0xD-0xF shall be Reserved.			
<b>SEC-Max-Banks</b>	4	-	O	RO	Secondary Media Max Banks—used to indicate the number of secondary media memory banks provisioned. Number of secondary media banks = $2^{\text{SEC-Max-Banks}}$ If an addressable secondary media is supported, then $(\text{PRI-Max-Banks} + \text{SEC-Max-Banks})$ shall be less than or equal to 13. SEC-Max-Banks values 0xD-0xE shall be Reserved. If a secondary media is unsupported or is not addressable, then SEC-Max-Banks shall be hardwired to 0xF.			
<b>Max-BTags</b>	4	-	M	RO	Max Bank Tags—used to indicate the maximum number of request packets supported per logical bank. Max-BTags shall be value such that the following is true: $0 < \text{Max-BTags} < 0xE$ Max-BTags values 0xE-0xF shall be Reserved.			
<b>PRI-LRSZ</b>	4	-	M	RO	Primary Media Logical Row Size—Indicates the number of bytes per logical row of the primary media. The logical row size equals the range of bytes that a component may optimize for spatial locality.			

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
SEC-LRSZ	4	-	O	RO	0x0—Responder does not support logical rows 0x1—128 Bytes 0x2—256 Bytes 0x3—512 Bytes 0x4—1024 Bytes 0x5—2048 Bytes 0x6—4096 Bytes 0x7—8192 Bytes 0x8-0xF—Reserved If a Responder does not support logical rows, then P2P-Core Read Offset and Write Offset operations shall not be enabled.
					Secondary Media Logical Row Size—Indicates the number of bytes per logical row of the secondary media. The logical row size equals the range of bytes that a component may optimize for spatial locality. 0x0—No secondary media support 0x1—128 Bytes 0x2—256 Bytes 0x3—512 Bytes 0x4—1024 Bytes 0x5—2048 Bytes 0x6—4096 Bytes 0x7—8192 Bytes 0x8-0xF—Reserved
DCLATE	16	-	O	RO	Daisy-Chain Latency—Indicates the worst-case latency from the time of the first bit of a Max_Payload_Size request or response packet is received until the last bit of the packet is transmitted to the next component within a daisy-chain topology. A Requester shall account for this latency for each component within a daisy-chain topology when determining request packet retransmission timers. Exponent is used to calculate latency in picoseconds as: Latency = DCLATE ns

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
PRI-BLEN					All internal processing, queuing delays, etc. that contribute to latency need to be included when determining this value.
	6	-	M	RO	This field is used to indicate the bank length associated with the primary media. Primary Bank Length = $2^{\text{PRI-BLEN}}$ bytes. PRI-BLEN shall be less than or equal to 44.
SEC-BLEN	6	-	M	RO	This field is used to indicate the bank length associated with the secondary media. Secondary Bank Length = $2^{\text{SEC-BLEN}}$ bytes.  If the secondary media is supported and addressable, then SEC-BLEN shall be less than or equal to 44, else SEC-BLEN shall be hardwired to 0x3F.
CTag Mask	4	-	M	RW	Component Tag Mask—used to configure the number of upper bits within the Tag field used to uniquely identify a Responder
CTag ID	12	-	M	RW	Component Tag ID—The unique value used to identify a Responder. This value is placed within the upper CTag Mask Bits.
R0	12	-	-	-	Reserved
R1	20	-	-	-	Reserved
R2	16	-	-	-	Reserved

## 8.20.2. Requester Bank Structure

The Requester Bank structure is used to describe each Responder memory component directly or daisy-chain-attached to the Requester.

The following are the features and requirements associated with the Requester Bank structure:

- 5 A Requester that uses P2P-Core OpClass packets to communicate with byte-addressable memory components shall support the Requester Bank structure.
- The Requester Bank structure shall use the *Requester Bank Structure Format*.
- Each Requester Bank structure shall correspond to a single Responder that is directly or daisy-chained-attached to the Requester.
- 10 If a Requester supports multiple Requester Bank structures, then these may be located using the Next Requester Bank Structure PTR.
- To target a given Responder memory component, the Requester takes the following steps:

- Software configures the Requester-local addresses associated with the primary media and the secondary media (if supported). Software also configures the lengths of the primary media and the secondary media (if supported).
- If the Responder supports zero-based addressing (*Core Structure Component CAP 1 Address Field Interpretation == 0b*), then:
  - The Responder may support up to  $2^{56}$  bytes of addressable memory through the use of multiple logical banks where each logical bank may support up to  $2^{44}$  bytes of addressable memory.
  - Using the Requester-local address indicated in the request operation, the Requester identifies the corresponding Requester Bank structure by comparing the Requester-local address with the configured PRI-REQLA and SEC-REQLA and respective corresponding length fields. Once the structure and the target media are identified, the Requester identifies the logical Bank Number and offset into the logical bank.
    - If the target media is the primary media, then to identify the logical Bank Number, the Requester masks off the lower PRI-BLEN bits and the upper  $(64 - \text{PRI-MLEN})$  bits from the Requester-local address. The remaining bits identify the logical Bank Number.
      - The offset is the lower PRI-BLEN bits of the Requester-local address. The offset is copied to the request packet's Address field.
    - If the target media is the secondary media, then to identify the logical Bank Number, the Requester masks off the lower SEC-BLEN bits and the upper  $(64 - \text{SEC-MLEN})$  bits from the Requester-local address. The remaining bits identify the logical Bank Number.
      - The offset is the lower SEC-BLEN bits of the Requester-local address. The offset is copied to the request packet's Address field.
    - The Requester selects an unused Tag from the Tag subrange which is copied to the request packet's Tag field:
      - $((\text{Ctag ID} + (\text{Bank Number} * 2^{\text{Max-Btags}})) \text{ to } (\text{Ctag ID} + ((\text{Bank Number} + 1) * 2^{\text{Max-Btags}} - 1))$ .

**Developer Note:** The primary and secondary media can be mapped into non-contiguous Requester-local address ranges. As a result, software is responsible for ensuring memory pages are completely contained within the Requester-local range associated with either the primary or the secondary media. Developers should also note that the Requester-local addresses are respectively aligned to the size of the primary or the secondary media.

- If the Responder does not support zero-based addressing (*Core Structure Component CAP 1 Address Field Interpretation == 1b*), then:
  - The Responder may support up to  $2^{44}$  bytes of addressable memory.
  - The maximum number of Tags that may be associated with the Responder shall be  $2^{\text{Max-Btags}}$
  - The Requester copies the lower 44 bits of the Requester-local address into the request packet's Address field.
  - To target a specific Responder, the Requester selects an unused Tag from the Tag subrange and copies it to the request packet's Tag field:

- 5
- $(\text{Ctag ID to } (\text{Ctag ID} + 2^{\text{Max-BTags}}) - 1)$
  - If a request packet targets Control Space, then the Requester shall encode the Tag only to identify the Responder; it shall not perform logical bank encoding (Control Space does not support logical banks).
  - If a Requester supports multiple interfaces that can be attached to a given Responder, then management configures the Interface ID fields to indicate which interfaces may be used. The Requester uses a component-specific mechanism to determine which interfaces to use to transmit a given request packet. Further, management shall set *Interface I-CAP 1 Control Omit P2P Standalone Acknowledgment* = 1b on these interfaces.

10

Requester Bank Structure Format							
Requester Bank Structure Fields							
Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description		
Version (Vers)	4	0x1	M	RO	Structure Version		
Max Interface	4	-	M	RO	The maximum number of component interfaces that can be configured to access an attached memory component. If Max Interfaces == 0x0, then the maximum number shall be 16.		
PRI CTL	8		M	RW	Controls Requester's primary media behavior		
		Bits 1:0			Media Volatility—Determines whether the media is volatile or non-volatile. If non-volatile, then it indicates whether a Write Persistent or a <i>Persistent Flush</i> should be used to ensure write persistency. If non-volatile, then set PU = 1b in applicable request packets. 0x0—Volatile Media 0x1—Non-Volatile Media. Use Write Persistent 0x2—Non-Volatile Media. Use <i>Persistent Flush</i> 0x3—Reserved		

Figure 8-18: Requester Bank Structure Format

Table 8-58: Requester Bank Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Version (Vers)	4	0x1	M	RO	Structure Version
Max Interface	4	-	M	RO	The maximum number of component interfaces that can be configured to access an attached memory component. If Max Interfaces == 0x0, then the maximum number shall be 16.
PRI CTL	8		M	RW	Controls Requester's primary media behavior Media Volatility—Determines whether the media is volatile or non-volatile. If non-volatile, then it indicates whether a Write Persistent or a <i>Persistent Flush</i> should be used to ensure write persistency. If non-volatile, then set PU = 1b in applicable request packets. 0x0—Volatile Media 0x1—Non-Volatile Media. Use Write Persistent 0x2—Non-Volatile Media. Use <i>Persistent Flush</i> 0x3—Reserved

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
SEC CTL	8	Bits 7:2	M	RW	RsvdP
					Controls Requester's secondary media behavior
		Bits 1:0			Media Volatility—Determines whether the media is volatile or non-volatile. If non-volatile, then it indicates whether a Write Persistent or a <i>Persistent Flush</i> should be used to ensure write persistency. If non-volatile, then set PU = 1b in applicable request packets. 0x0—Volatile Media 0x1—Non-Volatile Media. Use Write Persistent 0x2—Non-Volatile Media. Use <i>Persistent Flush</i> 0x3—Reserved
		Bits 7:2			RsvdP
PRI-Max-Banks	4	-	M	RW	Primary Media Max Banks is used to calculate the number of primary media memory banks provisioned in the corresponding Responder. Number of primary media banks = $2^{\text{PRI-Max-Banks}}$ PRI-Max-Banks shall be value such that the following is true: $0 \leq \text{PRI-Max-Banks} < 0xD$ PRI-Max-Banks values 0xD-0xF shall be Reserved.
SEC-Max-Banks	4	-	O	RW	Secondary Media Max Banks is used to calculate the number of addressable secondary media memory banks provisioned in the corresponding Responder Number of secondary media banks = $2^{\text{SEC-Max-Banks}}$ If an addressable secondary media is supported, then $(\text{PRI-Max-Banks} + \text{SEC-Max-Banks})$ shall be less than or equal to 13. SEC-Max-Banks values 0xD-0xE shall be Reserved. If a secondary media is unsupported or is not addressable, then SEC-Max-Banks shall be configured as 0xF.
Max-BTags	4	-	M	RW	Max Bank Tags—used to calculate the maximum number of request packets supported per logical bank of the corresponding Responder. Max Bank Tags = $2^{\text{Max-BTags}}$

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
CTag Mask					<p>Max-BTags shall be value such that the following is true: <math>0 &lt; \text{Max-BTags} &lt; 0xE</math></p> <p>Max-BTags values <math>0xE-0xF</math> shall be Reserved.</p>
PRI-LRSZ	4	-	M	RW	<p>Component Tag Mask—used to indicate the number of upper bits within the Tag field used to uniquely identify the corresponding Responder</p>
	4	-	M	RW	<p>Primary Media Logical Row Size—Indicates the number of bytes per logical row of the primary media of the corresponding Responder. The logical row size equals the range of bytes that a component may optimize for spatial locality.</p> <p>0x0—Responder does not support logical rows  0x1—128 Bytes  0x2—256 Bytes  0x3—512 Bytes  0x4—1024 Bytes  0x5—2048 Bytes  0x6—4096 Bytes  0x7—8192 Bytes  0x8-0xF—Reserved</p> <p>If a Responder does not support logical rows, then P2P-Core Read Offset and Write Offset operations shall not be enabled.</p>
SEC-LRSZ	4	-	O	RW	<p>Secondary Media Logical Row Size—Indicates the number of bytes per logical row of the secondary media of the corresponding Responder. The logical row size equals the range of bytes that a component may optimize for spatial locality.</p> <p>0x0—No secondary media support or secondary media is not addressable.  0x1—128 Bytes  0x2—256 Bytes  0x3—512 Bytes  0x4—1024 Bytes  0x5—2048 Bytes  0x6—4096 Bytes  0x7—8192 Bytes  0x8-0xF—Reserved</p>
PRI-MLEN	8	-	M	RW	This field is used to indicate the primary media length. Primary Media Length = $2^{\text{PRI-MLEN}}$ bytes.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
SEC-MLEN					PRI-MLEN shall be less than or equal to 64.
PRI-BLEN	8	-	M	RW	<p>This field is used to indicate the secondary media length. Secondary Media Length = <math>2^{\text{SEC-MLEN}}</math> bytes.</p> <p>If the secondary media is supported and addressable, then SEC-MLEN shall be less than or equal to 64, else this field shall be Reserved.</p>
SEC-BLEN	6	-	M	RW	<p>This field is used to indicate the bank length associated with the primary media. Primary Bank Length = <math>2^{\text{PRI-BLEN}}</math> bytes. PRI-BLEN shall be less than or equal to 44.</p>
CTag ID	6	-	M	RW	<p>This field is used to indicate the bank length associated with the secondary media. Secondary Bank Length = <math>2^{\text{SEC-BLEN}}</math> bytes.</p> <p>If the secondary media is supported and addressable, then SEC-BLEN shall be less than or equal to 44, else SEC-BLEN shall be configured to 0x3F.</p>
Interface ID n	12	-	M	RW	Component Tag ID—The unique value used to identify a Responder. This value is placed within the upper Ctag Mask Bits.
PRI-REQLA	12	-	M	RW	<p>The identifier of the Requester's egress interface used to reach the corresponding Responder.</p> <p>If the Requester supports a fixed association between a Requester Bank structure and a component egress interface, then this field shall be hardwired to the component egress interface identifier.</p>
SEC-REQLA	52	-	M	RW	This field is configured with the Requester-local address that is associated with the Responder's primary media.
Valid Interface ID	52	-	M	RW	<p>This field is configured with the Requester-local address that is associated with the Responder's secondary media.</p> <p>If the secondary media is unsupported or not addressable, then this field shall be Reserved.</p>
	16	-	M	RW	A bit mask field used to indicate if Interface ID K field is configured with a valid interface identifier.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Next Requester Bank Structure PTR					If Bit <sub>K</sub> == 1b, then Interface ID K field contains a configured interface identifier, else Interface ID K field is not configured.
R0	32	-	O	RO	A Requester may provision multiple Requester Bank structures. If this field is non-Null, then this field points to the next provisioned Requester Bank structure.
R1	12	-	-	-	RsvdP
R2	4	-	-	-	RsvdP
R3	4	-	-	-	RsvdP
	32	-	-	-	Reserved

## 8.21. Component Error and Signal Event Structure

The Component Error and Signal Event Structure provides error status, logging, event signaling, and control services.

The following are the features and requirements associated with the Component Error Signal Event structure:

- Any component type may support the *Component Error and Signal Event Structure*.
- The *Component Error and Signal Event Structure* shall use the *Component Error and Signal Event Structure Format*.
- The ECTL (Error Control) field shall be used to disable or enable error detection and handling. Each encoding defines the set of error detection and handling that is enabled.
- Error management uses the following fields:
  - Error Status indicates which errors have been recorded.
  - Error Detect indicates which errors to detect.
  - *Component Containment* Trigger indicates which errors trigger component containment.
  - Signal Target indicates which errors and how to signal management.
  - Fault Injection may be used by software to trigger error handling for a given error.
- Error Status, Error Detect, Error signal, and Trigger Error are bit masks where each Bit<sub>K</sub> shall be implemented as a set across these fields, i.e., be present in all fields or not at all.
  - Each Bit<sub>K</sub> may be present in the Error Test field.
  - For each Bit<sub>K</sub> implemented, the corresponding sub-field<sub>K</sub> within the Signal Target field shall be implemented.
- Multiple error detection:
  - If multiple errors are detected relative to a single action, e.g., validation of a single packet, then only the highest precedence error shall be reflected in the Error Status field.

- If multiple errors are detected relative to multiple actions, e.g., validation of multiple packets, then the Error Status field may have multiple Bit<sub>k</sub> = 1b. For each detected error with a Signal Bit<sub>k</sub> = 1b, the corresponding Signal Target sub-field<sub>k</sub> actions shall be taken.
- If multiple errors are detected that can trigger *Component Containment*, component containment shall be triggered only once. Component containment is indicated in the *Core C-Status* field.
- A component may log error information:
  - If a component supports the *Component Error and Signal Event Structure*, it shall support at least one error log entry.
  - Log entries are linearly accessed starting at entry 0 through ELog Size.
  - The component shall consume log entries in monotonically-increasing order modulo ELog Size.
    - A log entry shall be consumed only if A == 1b. If consumed, then the component shall set A = 1b.
    - If the log does not contain unallocated log entries, then the component shall silently discard the log information, and shall update the corresponding E-Status field to indicate logging failed due to a lack of unallocated log entries.
  - Software shall release log entries in monotonically increasing order modulo ELog Size.
    - Once software is done with a log entry, software shall set A = 0b.
  - Each log entry identifies the type of error information logged. The following log entry formats are specified—Component Error (*Component Error Log Format – Component Error*) and Interface Error (*Component Error Log Format – Component Interface Error*).
  - If a component has previously logged packet-related errors, and it detects a new erroneous packet, it may compare the erroneous packet's header with the error log entries to determine if this erroneous packet is a retransmitted packet, and therefore, should not generate a new error log entry and should not invoke the same configured error (report, trigger, etc.) actions.
  - If multiple errors are detected from different component elements, then the following precedence (highest-to-lowest) shall be used to determine what information is logged:
    1. Component Execution Error
    2. Component Internal Error
    3. Component Packet Validation Error
    4. Interface Error
    5. Vendor-defined Error
  - If multiple packets fail packet validation, then the *Component Error and Signal Event Structure* shall reflect only the results of the first packet for each given error type.
  - Each error detection point—component, interface, and service specific—is responsible for determining what error information is placed within the error log.
    - If a Component Execution Error or Component Internal Error is detected, then the component shall record any relevant component-specific information. The log information will require subsequent interpretation through component-specific or vendor-specified means.
    - If a Component Packet Validation Error is detected, then the component shall copy the first 30 bytes of the packet into the Error-specific field. Bytes are copied with packet byte 0 to Packet Header field byte 0 with subsequent bytes sequentially placed. Unused Error-specific bytes shall be set to zero.

- The C-Error Signal Target, C-Event Signal Target, and I-Event Signal Target fields may be configured to generate one or two component-local interrupts, an *Unsolicited Event (UE) Packet*, or an *Unsolicited Event (UE) Packet* and component-local interrupt(s) for each selected error or event.
- The C-Error Signal Target field identifies if an error is to be signaled and how. If a component supports only *Out-of-band Management*, then software shall configure at least one component-local interrupt to inform management when a configured error is detected.
- The C-Event Detect may be configured to cause the component to inform management of a component-level event.
- The I-Event Detect may be configured to cause the component to inform management of an interface-level event. The I-Event Detect field shall apply to all component interfaces.
- Each MGMT VC K and MGMT Interface K fields identifies a [VC, egress interface] that can be used to transmit an *Unsolicited Event (UE) Packet* to a management component. For each distinct manager that is configured to receive an *Unsolicited Event (UE) Packet*, the corresponding UEP Mask field shall be set to a non-zero value. For each Bit<sub>K</sub> == 1b, the corresponding [MGMT VC K, MGMT Interface K] may be used to transmit the *Unsolicited Event (UE) Packet* to that management component.

5

10

15

+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	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	7 6 5 4 3 2 1 0			
E-Status	E-Control	Size	Vers	Type	< Byte 0x0					
ELog Size	R0	Error MGR SID	Error MGR CID	< Byte 0x8						
R1		Event MGR SID	Event MGR CID	< Byte 0x10						
Interrupt Address 0								< Byte 0x18		
Interrupt Address 1								< Byte 0x20		
Interrupt Data 1		Interrupt Data 0		< Byte 0x28						
C-Error Status								< Byte 0x30		
C-Error Detect								< Byte 0x38		
C-Error Signal								< Byte 0x40		
C-Error Trigger								< Byte 0x48		
C-Error Fault Injection								< Byte 0x50		
C-Error Signal Target [63:0]								< Byte 0x58		
C-Error Signal Target [127:64]								< Byte 0x60		
C-Error Signal Target [191:128]								< Byte 0x70		
C-Event Detect								< Byte 0x78		
C-Event Injection								< Byte 0x80		
C-Event Signal Target [63:0]								< Byte 0x88		
C-Event Signal Target [127:64]								< Byte 0x90		
C-Event Signal Target [191:128]								< Byte 0x98		
I-Event Detect								< Byte 0xA0		
I-Event Injection								< Byte 0xA8		
I-Event Signal Target [63:0]								< Byte 0xB0		
I-Event Signal Target [127:64]								< Byte 0xB8		
I-Event Signal Target [191:128]								< Byte 0xC0		
R2	MGMT Interface ID 2	MGMT VC 2	MV2	MGMT Interface ID 1	MGMT VC 1	MV1	MGMT Interface ID 0	MGMT VC 0	MVO	< Byte 0xC8
R3	MGMT Interface ID 5	MGMT VC 2	MV5	MGMT Interface ID 4	MGMT VC 4	MV4	MGMT Interface ID 3	MGMT VC 3	MVO3	< Byte 0xD0
R4				MGMT Interface ID 7	MGMT VC 7	MV7	MGMT Interface ID 6	MGMT VC 6	MVO6	< Byte 0xD8
MECH MGR UEP Mask	PWR MGR UEP Mask	Media UEP Mask	Event UEP Mask	Error UEP Mask	SFM UEP Mask	PFM UEP Mask	PM UEP Mask		< Byte 0xE0	
R5									< Byte 0xE8	
Error Log <sub>0-N</sub>									< Byte 0xF0	
									< Byte 0xYY	

Figure 8-19: Component Error and Signal Event Structure Format

Table 8-59: Component Error and Signal Event Structure Fields

Field Name	Size (bits)	Value / Bit Location	M O	Access	Description
<b>Version (Vers)</b>	4	0x1	M	RO	Structure Version
<b>E-Control</b>	16	-	M	RWS	Control
		Bits 2:0			Error Enablement 0x0—Disable all error detection and handling 0x1—Enable component-level error detection and handling 0x2—Enable component and interface-level error detection and handling 0x3—Enable component, interface, and service-level error detection and handling 0x4-0x7—Reserved
		Bit 3			Trigger <i>Component Containment</i> .—If set to 1b, then triggers <i>Component Containment</i> . This may be used by software to trigger containment at any time. Reads of this bit shall return 0b.
		Bits 6:4			Event Logging Level—If logging is supported, then this field indicates the error severity levels to create a corresponding log entry. See error severity levels listed in <i>C-Error Detect</i> 0x0—Log only Critical errors 0x1—Log Critical and Caution errors 0x2—Log Critical, Caution, and Non-critical errors 0x3-0x7—Reserved
		Bits 8:7			Error UEP Target—If an <i>Unsolicited Event (UE) Packet</i> is used to inform management of errors, then this is used to identify the management component. 0x0—Target Primary Manager as indicated in the <i>Core Structure</i> 0x1—Target the Primary or Secondary Fabric Manager as indicated in the <i>Core Structure</i>

E-Status	16	M	RW1CS	0x2—Target independent manager—Error MGR CID valid, Error MGR SID invalid			
				0x3—Target independent manager—Error MGR CID valid, Error MGR SID valid			
				Bits 10:9			
				Event UEP Target—If an <i>Unsolicited Event (UE) Packet</i> is used to inform management of events, then this is used to identify the management component.			
				0x0—Target Primary Manager as indicated in the <i>Core Structure</i>			
				0x1—Target the Primary or Secondary Fabric Manager as indicated in the <i>Core Structure</i>			
				0x2—Target independent manager—Event MGR CID valid, Event MGR SID invalid			
				0x3—Target independent manager—Event MGR CID valid, Event MGR SID valid			
				Bits 15:11			
				RsvdP			
ELog Size				-			
				Bit 0			
				Bits 15:1			
Error MGR CID				Status			
				Logging Failed—one or more attempts to generate a log entry failed due to a lack of free log entries.			
Error MGR SID				RsvdZ			
				Number of error log entries supported by the component. If ELog Size == 0x0, then the number of provisioned log entries shall be 256.			
Event MGR CID				12			
				-			
				M			
				RW			
				If errors are independently managed by a component other than any of the management components identified within the <i>Core Structure</i> , then this field contains the CID of the error manager to transmit the <i>Unsolicited Event (UE) Packet</i> .			
				16			
				-			
				M			
				RW			
				If errors are independently managed by a component other than any of the management components identified within the <i>Core Structure</i> , then this field contains the SID of the error manager to transmit the <i>Unsolicited Event (UE) Packet</i> .			
				12			
				-			
				M			
				RW			
				If events are independently managed by a component other than any of the			

					management components identified within the <i>Core Structure</i> , then this field contains the CID of the event manager to transmit the <i>Unsolicited Event (UE) Packet</i> .
<b>Event MGR SID</b>	16	-	M	RW	If events are independently managed by a component other than any of the management components identified within the <i>Core Structure</i> then this field contains the SID of the event manager to transmit the <i>Unsolicited Event (UE) Packet</i> .
<b>Interrupt Address 0</b>	64	-	O	RW	The component-local address to target component-local interrupt 0.
<b>Interrupt Address 1</b>	64	-	O	RW	The component-local address to target component-local interrupt 1
<b>Interrupt Data 0</b>	32	-	O	RW	The interrupt data associated with component-local interrupt 0 Contents of this field are component-specific
<b>Interrupt Data 1</b>	32	-	O	RW	The interrupt data associated with component-local interrupt 1 Contents of this field are component-specific
<b>C-Error Status</b>	64	-	M	-	Indicates which component errors if any have been detected. See <i>C-Error Status</i>
<b>C-Error Detect</b>	64	-	M	-	Controls which component errors if any to detect. If an error is detected, then the component examines the C-Error Signal Target field to determine whether to signal management in addition to updating the Error Status field. See <i>C-Error Detect</i>
<b>C-Error Trigger Error</b>	64	-	M	-	Controls which component errors should trigger <i>Component Containment</i> . See <i>C-Error Trigger</i> .
<b>C-Error Fault Injection</b>	64	-	M	-	Controls which component faults to inject. This field is not intended for use in normal operation. See <i>C-Error Fault Injection</i>
<b>C-Error Signal Target</b>	192	-	M	RW	Each Signal Target 3-bit sub-field indicates how the associated component error is to be signaled. If configured, then the component shall generate one or both component-local interrupts, an <i>Unsolicited Event (UE) Packet</i> (or if the component supports only the P2P-Core or P2P-Coherency OpClass, a <i>CTL-UE</i>

					<p><i>Packet</i>), or both component local interrupt(s) and an <i>Unsolicited Event (UE) Packet</i>.</p> <p>0x0—No signal is generated 0x1—Component-local interrupt 0 0x2—Component-local interrupt 1 0x3—Component-local interrupts 0 and 1 0x4—<i>Unsolicited Event (UE) Packet</i> 0x5—Component-local interrupt 0 and an <i>Unsolicited Event (UE) Packet</i> 0x6—Component-local interrupt 1 and an <i>Unsolicited Event (UE) Packet</i> 0x7—Component-local interrupts 0 and 1, and an <i>Unsolicited Event (UE) Packet</i></p> <p>Sub-field 0 is located in byte 0, bit 0. Sub-field 1 is contiguous to sub-field 0. And so forth.</p>
<b>C-Event Detect</b>	64	-	O	-	Controls which component events to signal to management. See <i>C-Event Detect</i>
<b>C-Event Inject</b>	64	-	O	-	Controls which component events to inject to management. See <i>C-Event Injection</i>
<b>C-Event Signal Target</b>	192	-	O	RW	<p>Each Signal Target 3-bit sub-field indicates how the associated event is to be signaled. If configured, then the component shall generate one or both component-local interrupts, an <i>Unsolicited Event (UE) Packet</i>, or both component local interrupt(s) and an <i>Unsolicited Event (UE) Packet</i>.</p> <p>0x0—No signal is generated 0x1—Component-local interrupt 0 0x2—Component-local interrupt 1 0x3—Component-local interrupts 0 and 1 0x4—<i>Unsolicited Event (UE) Packet</i> 0x5—Component-local interrupt 0 and an <i>Unsolicited Event (UE) Packet</i> 0x6—Component-local interrupt 1 and an <i>Unsolicited Event (UE) Packet</i> 0x7—Component-local interrupts 0 and 1, and an <i>Unsolicited Event (UE) Packet</i></p> <p>Sub-field 0 is located in byte 0, bit 0. Sub-field 1 is contiguous to sub-field 0. And so forth.</p>

<b>I-Event Detect</b>	64	-	O	-	Controls which interface events to signal to management. This field applies to all component interfaces. See <i>I-Event Detect</i>
<b>I-Event Injection</b>	64	-	O	-	Controls which interface events to inject management. This field applies to all component interfaces. See <i>I-Event Injection</i>
<b>I-Event Signal Target</b>	192	-	O	RW	<p>Each Signal Target 3-bit sub-field indicates how the associated event is to be signaled. If configured, then the component shall generate one or both component-local interrupts, an <i>Unsolicited Event (UE) Packet</i>, or both component local interrupt(s) and an <i>Unsolicited Event (UE) Packet</i>.</p> <p>0x0—No signal is generated  0x1—Component-local interrupt 0  0x2—Component-local interrupt 1  0x3—Component-local interrupts 0 and 1  0x4—<i>Unsolicited Event (UE) Packet</i>  0x5—Component-local interrupt 0 and an <i>Unsolicited Event (UE) Packet</i>  0x6—Component-local interrupt 1 and an <i>Unsolicited Event (UE) Packet</i>  0x7—Component-local interrupts 0 and 1, and an <i>Unsolicited Event (UE) Packet</i></p> <p>Sub-field 0 is located in byte 0, bit 0. Sub-field 1 is contiguous to sub-field 0. And so forth.</p>
<b>MV [n]</b>	1	-	M	RW	Indicates if the corresponding MGMT VC and MGMT Interface ID fields are configured.
<b>MGMT VC [n]</b>	5	-	M	RW	<p>If a component supports <i>Control OpClass In-Band Operations</i>, then this identifies the VC to use when transmitting an <i>Unsolicited Event (UE) Packet</i> on MGMT Interface [n].</p> <p>If MV[n] == 1b, then MGMT VC [n] is valid.</p>
<b>MGMT Interface ID [n]</b>	12	-	M	RW	<p>If a component supports <i>Control OpClass In-Band Operations</i>, then this identifies an egress interface that can be used to transmit an <i>Unsolicited Event (UE) Packet</i>.</p> <p>If MV[n] == 1b, then Management Interface ID [n] is valid.</p> <p>Multiple MGMT Interface ID fields may be valid.</p>

<b>* UEP Mask</b>					For each Bit <sub>K</sub> == 1b, the corresponding MGMT VC K and MGMT Interface K may be used to transmit an <i>Unsolicited Event (UE) Packet</i> to the component corresponding to this bit mask. <ul style="list-style-type: none"> <li>• PM UEP Mask corresponds to the Primary Manager identified by <i>Core Structure</i> PMCID.</li> <li>• PFM UEP Mask corresponds to the Primary Fabric Manager identified by <i>Core Structure</i> [PFMCID, PFMSID].</li> <li>• SFM UEP Mask corresponds to the Secondary Fabric Manager identified by <i>Core Structure</i> [SFMCID, SFMSID].</li> <li>• Error UEP Mask corresponds to the <i>Component Error and Signal Event Structure</i> [Error MGR CID, Error MGR SID].</li> <li>• Event UEP Mask corresponds to the Event Manager identified by <i>Component Error and Signal Event Structure</i> [Event MGR CID, Event MGR SID].</li> <li>• PWR UEP Mask corresponds to the Power Manager identified by <i>Core Structure</i> [PWR MGR CID, PWR MGR SID].</li> <li>• MECH UEP Mask corresponds to the Mechanical Manager identified by <i>Component Mechanical Structure</i> [Mechanical MGR CID, Mechanical MGR SID].</li> <li>• Media UEP Mask corresponds to the Media Manager identified by <i>Component Media Structure</i> [PWR MGR CID, PWR MGR SID].</li> </ul>	
	R0	20	-	-	-	RsvdP
	R1	36	-	-	-	RsvdP
	R2	10	-	-	-	RsvdP
	R3	10	-	-	-	RsvdP
	R4	28	-	-	-	RsvdP
	R5	64	-	-	-	Reserved

## 8.21.1. C-Error Fields

The following applies to the C-Error Status, C-Error Detect, C-Trigger, and C-Error Fault Injection fields:

- Each error Bit<sub>K</sub> shall be implemented as a set across the C-Error Status, C-Error Detect, and C-Trigger fields, i.e., Bit<sub>K</sub> shall be implemented in all fields or not at all. C-Error Fault Injection bits may be implemented.
- Each unsupported Bit<sub>K</sub> shall be hardwired to 0b.
- If Bit<sub>K</sub> is supported, then the default Bit<sub>K</sub> value shall be 0b.

### 8.21.1.1. C-Error Status

The C-Error Status field shall be as specified in *C-Error Status*. For each C-Error Status Bit<sub>K</sub> = 1b, the corresponding error condition has been recorded. Software clears Bit<sub>K</sub> by writing 1b. Unsupported C-Error Status bits shall be hardwired to 0b.

The Update C-Status column indicates which *Core C-Status* bit to update. These bits shall be updated independent of whether the component supports the *Component Error and Signal Event Structure*.

Update C-Status Legend:

- NFI = Update the *Core C-Status* Non-Fatal Internal Error status bit
- FI = Update the *Core C-Status* Fatal Internal Error status bit
- NTP = Update the *Core C-Status* Non-Transient Protocol Error status bit.
- If a NFI and the corresponding *Component Error and Signal Event Structure* C-Error Signal Target subfield indicates an *Unsolicited Event (UE) Packet* is to be transmitted, then use the “Recoverable error event detected” event type.
- If a FI or NTP and the corresponding *Component Error and Signal Event Structure* C-Error Signal Target subfield indicates an *Unsolicited Event (UE) Packet* is to be transmitted, then use the “Unrecoverable error event detected” event type.

Table 8-60: C-Error Status

Bit Location	Access	Update C-Status	Description
0	RW1CS	FI	<i>Component Containment</i> Recorded
1	RW1CS	NFI	Non-Fatal Internal Component Error Recorded
2	RW1CS	FI	Fatal Internal Component Error Recorded
3	RW1CS	NTP	End-to-End Unicast UR Recorded
4	RW1CS	NTP	End-to-End Unicast MP Recorded
5	RW1CS	NTP	End-to-End Unicast Packet Execution Error (EXE) Non-Fatal Recorded
6	RW1CS	NTP	End-to-End Unicast Packet Execution Error (EXE) Fatal Recorded
7	RW1CS	NTP	End-to-End Unicast UP Recorded
8	RW1CS	FI	AE Invalid Access Key Recorded

Bit Location	Access	Update C-Status	Description
9	RW1CS	FI	AE Invalid Access Permission Recorded
10	RW1CS	NTP	Switch / Router UP Recorded
11	RW1CS	FI	End-to-end Packet Retry Error Recorded
12	RW1CS	FI	Fatal Media Error Containment Triggered Recorded
13	RW1CS	FI	Security Error Recorded
14	RW1CS	NTP	End-to-End Multicast UR Recorded
15	RW1CS	NTP	End-to-End Multicast MP Recorded
16	RW1CS	NTP	End-to-End Multicast Packet Execution Error (EXE) Non-Fatal Recorded
17	RW1CS	NTP	End-to-End Multicast Packet Execution Error (EXE) Fatal Recorded
18	RW1CS	NTP	End-to-End Multicast UP Recorded
19	RW1CS	NTP	SOD UP Recorded
20	RW1CS	FI	Unexpected Component Power Loss Recorded
59:21	-	-	RsvdZ
63:60	RW1CS	-	Vendor-defined Error Status bits Each unsupported Vendor-defined error bit shall be Reserved, and shall be hardwired to 0b. The <i>Core Structure</i> C-UUID is used to identify the vendor-defined interpretation of these bits.

### 8.21.1.2. C-Error Detect

The C-Error Detect field shall be as specified in *C-Error Detect*. For each C-Error Detect Bit<sub>k</sub> = 1b, the corresponding error condition shall be recorded in the C-Error Status field. Unsupported C-Error Detect bits shall be hardwired to 0b. This table also indicates the severity of each error. The severity is used to determine whether the error should be logged based on the configured Error Logging Level within the Error Control field.

Table 8-61: C-Error Detect

Bit Location	Access	Error Severity	Description
0	RW	Critical	<i>Component Containment</i> Detect
1	RW	Caution	Non-Fatal Internal Component Error Detect
2	RW	Critical	Fatal Internal Component Error Detect

Bit Location	Access	Error Severity	Description
3	RW	Critical	End-to-End Unicast UR Detect
4	RW	Critical	End-to-End Unicast MP Detect
5	RW	Caution	End-to-End Unicast Packet Execution Error (EXE) Non-fatal Detect
6	RW	Critical	End-to-End Unicast Packet Execution Error (EXE) Fatal Detect
7	RW	Caution	End-to-End Unicast UP Detect
8	RW	Critical	AE Invalid Access Key Detect
9	RW	Critical	AE Invalid Access Permission Detect
10	RW	Caution	Switch / Router UP Detect—see <i>Packet Validation and Processing</i>
11	RW	Non-critical	End-to-end Packet Retry Error Detect—Exceeded the maximum number of request packet retransmissions
12	RW	Critical	Fatal Media Error Containment Triggered Detect
13	RW	Critical	Security Error Detect
14	RW	Caution	End-to-End Multicast UR Detect
15	RW	Caution	End-to-End Multicast MP Detect
16	RW	Caution	End-to-End Multicast Packet Execution Error (EXE) Non-Fatal Detect
17	RW	Caution	End-to-End Multicast Packet Execution Error (EXE) Fatal Detect
18	RW	Caution	End-to-End Multicast UP Detect
19	RW	Caution	SOD UP Detect
20	RW	Critical	Unexpected Component Power Loss Detect—detects non-managed or unexpected power loss events.  The <i>Component Error and Signal Event Structure</i> C-Error Signal Target sub-field corresponding to this error shall be hardwired to zero.
59:21	-	-	RsvdZ
63:60	RW	Caution	Vendor-defined Error Detect bits  Each unsupported Vendor-defined error bit shall be Reserved, and shall be hardwired to 0b.  The <i>Core Structure</i> C-UUID is used to identify the vendor-defined interpretation of these bits.

### 8.21.1.3. C-Error Trigger

The C-Error Trigger field shall be as specified *C-Error Trigger*. For each C-Error Trigger Bit<sub>k</sub> = 1b, if the corresponding error is detected, then the component shall trigger *Component Containment*. Unsupported C-Error Trigger bits shall be hardwired to 0b.

- 5 Component containment is also triggered if an Unexpected Physical Layer Failure—PHY-Down (see *Interface Error Fields*) triggered interface containment and *Interface I-CAP 1 Control Interface-Component Containment Enable* == 1b.

Table 8-62: C-Error Trigger

Bit Location	Access	Description
0	RW	<i>Component Containment Trigger</i> —Writing 1b to this bit shall clear <i>Component Containment</i> , and shall enable and arm <i>Component Containment</i> . Reads of this field shall return 0b.
1	RW	Non-Fatal Internal Component Error Trigger
2	RW	Fatal Internal Component Error Trigger
3	RW	End-to-End Unicast UR Trigger
4	RW	End-to-End Unicast MP Trigger
5	RW	End-to-End Unicast Packet Execution Error (EXE) Non-fatal Trigger
6	RW	End-to-End Unicast Packet Execution Error (EXE) Fatal Trigger
7	RW	End-to-End Unicast UP Trigger
8	RW	AE Invalid Access Key Trigger
9	RW	AE Invalid Access Permission Trigger
10	RW	Switch / Router Unexpected Packet (UP) Trigger—see <i>Switches and Transparent Router (TR)</i> . Only applicable if the component supports switch or TR packet relay.
11	RW	End-to-end Packet Retry Error Trigger
12	RW	Fatal Media Error Containment Trigger
13	RW	Security Error Trigger
14	RW	End-to-End Multicast UR Trigger
15	RW	End-to-End Multicast MP Trigger
16	RW	End-to-End Multicast Packet Execution Error (EXE) Non-Fatal Trigger
17	RW	End-to-End Multicast Packet Execution Error (EXE) Fatal Trigger
18	RW	End-to-End Multicast UP Trigger
19	RW	SOD UP Trigger

Bit Location	Access	Description
20	RW	Unexpected Component Power Loss Trigger
59:21	-	Reserved
63:60	RW	<p>Vendor-defined Error Trigger bits</p> <p>Each unsupported Vendor-defined error bit shall be Reserved, and shall be hardwired to 0b.</p> <p>The <i>Core Structure</i> C-UUID is used to identify the vendor-defined interpretation of these bits.</p>

#### 8.21.1.4. C-Error Fault Injection

The C-Error Fault Injection field shall be as specified in *C-Error Fault Injection*. If C-Error Fault Injection Bit<sub>K</sub> == 1b and the corresponding C-Error Detect Bit<sub>K</sub> == 1b, then the component shall initiate the corresponding error processing as though the error was detected.

Table 8-63: C-Error Fault Injection

Bit Location	Access	Description
0	-	Reserved
1	WO	Test Non-Fatal Internal Component Error
2	WO	Test Fatal Internal Component Error
3	WO	Test End-to-End Unicast UR Error
4	WO	Test End-to-End Unicast MP Error
5	WO	Test End-to-End Unicast Packet Execution Error (EXE) Non-fatal Error
6	WO	Test End-to-End Unicast Packet Execution Error (EXE) Fatal Error
7	WO	Test End-to-End Unicast UP Error
8	WO	Test AE Invalid Access Key Error
9	WO	Test AE Invalid Access Permission Error
10	WO	Test Switch / Router Unexpected Packet (UP) Error—see <i>Switches and Transparent Router (TR)</i> . Only applicable if the component supports switch or router packet relay.
11	WO	Test End-to-end Packet Retry Error
12	WO	Reserved
13	WO	Security Error
14	WO	Test End-to-End Multicast UR Error

Bit Location	Access	Description
15	WO	Test End-to-End Multicast MP Error
16	WO	Test End-to-End Multicast Packet Execution Error (EXE) Non-Fatal Error
17	WO	Test End-to-End Multicast Packet Execution Error (EXE) Fatal Error
18	WO	Test End-to-End Multicast UP Error
19	WO	Test SOD UP Error
20	WO	Test Unexpected Component Power Loss
59:21	-	Reserved
63:60	WO	<p>Test Vendor-defined Error</p> <p>Each unsupported Test Vendor-defined error bit shall be Reserved, and shall be hardwired to 0b.</p> <p>The <i>Core Structure</i> C-UUID is used to identify the vendor-defined interpretation of these bits.</p>

## 8.21.2. C-Event and I-Event Fields

The following applies to the C-Event Detect, C-Event Injection, I-Event Detect, and I-Event Injection fields:

- Each unsupported Bit<sub>k</sub> shall be hardwired to 0b.
- If Bit<sub>k</sub> is supported, then the default Bit<sub>k</sub> value shall be 0b.

### 8.21.2.1. C-Event Detect / C-Event Injection

The C-Event Detect shall be as specified in *C-Event Detect*. For each C-Event Detect Bit<sub>k</sub> = 1b, if the corresponding event is detected, then the component informs management of the event as indicated in the corresponding C-Event Signal Target subfield.

Table 8-64: C-Event Detect

Bit Location	Access	Description
0	RW	BIST Failure Event Detect
1	RW	Unable to Communicate with an Authorized Destination Event Detect
2	RW	Excessive RNR NAK Responses Event Detect
3	RW	Buffer Overflow Event Detect
4	RW	Component Thermal Shutdown Event Detect
5	RW	Possible Malicious Packet Event Detect

Bit Location	Access	Description
6	RW	Invalid Component Image Event Detect
7	RW	Component Low-power Entry Event Detect
8	RW	Component Low-power Exit Event Detect
9	RW	Component Deep Low-power Entry Event Detect (inform management prior to C-DLP entry)
10	RW	Component Deep Low-power Exit Event Detect
11	RW	Peer Component Deep Low-power Entry Event Detect
12	RW	Emergency Power Reduction Triggered Event Detect
13	RW	Fatal Media Error Containment Triggered Event Detect
14	RW	Component Power-off Transition Completed Event Detect
15	RW	Component Power Restoration Event Detect
16	RW	Primary Media Maintenance Required Event Detect
17	RW	Primary Media Maintenance Override Event Detect
18	RW	Secondary Media Maintenance Required Event Detect
19	RW	Secondary Media Maintenance Override Event Detect
20	RW	Component Thermal Throttle Event Detect
21	RW	Component Thermal Throttle Restoration Event Detect
22	RW	P2P Non-transient Operating Condition Detect
59:23	-	Reserved
63:60	-	Vendor-defined Event Detect Each unsupported Vendor-defined event bit shall be Reserved, and shall be hardwired to 0b. The <i>Core Structure</i> C-UUID is used to identify the vendor-defined interpretation of these bits.

The C-Event Injection field shall be as specified in *C-Event Injection*. If C-Event Injection Bit<sub>K</sub> == 1b and the corresponding C-Event Detect Bit<sub>K</sub> == 1b, then the component shall initiate the corresponding event processing as though the event was detected. Unsupported C-Event Injection bits shall be hardwired to 0b. Reads of this field shall return 0x0.

Table 8-65: C-Event Injection

Bit Location	Access	Description
0	WO	Inject BIST Failure Event

Bit Location	Access	Description
1	WO	Inject Unable to Communicate with an Authorized Destination Event
2	WO	Inject Excessive RNR NAK Responses Event
3	WO	Inject Buffer Overflow Event
4	WO	Inject Component Thermal Shutdown Event
5	WO	Inject Possible Malicious Packet Event
6	WO	Inject Invalid Component Image Event
7	WO	Inject Component Low-power Entry Event
8	WO	Inject Component Low-power Exit Event
9	WO	Inject Component Deep Low-power Entry Event
10	WO	Inject Component Deep Low-power Exit Event
11	WO	Inject Peer Component Deep Low-power Entry Event
12	WO	Inject Emergency Power Reduction Triggered Event
13	WO	Inject Fatal Media Error Containment Triggered Event
14	WO	Inject Component Power-off Transition Completed Event
15	WO	Inject Component Power Restoration Event
16	WO	Inject Primary Media Maintenance Required Event
17	WO	Inject Primary Media Maintenance Override Event
18	WO	Inject Secondary Media Maintenance Required Event
19	WO	Inject Secondary Media Maintenance Override Event
20	WO	Inject Component Thermal Throttle Event
21	WO	Inject Component Thermal Throttle Restoration Event
22	WO	Inject P2P Non-transient Operating Condition Event
59:23	-	Reserved
63:60	WO	Inject Vendor-defined Event Each unsupported Vendor-defined event bit shall be Reserved, and shall be hardwired to 0b. The <i>Core Structure</i> C-UUID is used to identify the vendor-defined interpretation of these bits.

### 8.21.2.2. I-Event Detect / I-Event Injection

The I-Event Detect shall be as specified in *I-Event Detect*. For each I-Event Detect Bit<sub>K</sub> = 1b, if the corresponding interface event is detected, then the component informs management of the event as indicated in the corresponding I-Event Signal Target subfield. The I-Event Detect field applies to all component interfaces.

5

Unsupported I-Event Detect bits shall be hardwired to 0b.

Table 8-66: I-Event Detect

Bit Location	Access	Description
0	WO	<i>Full Interface Reset Event</i>
1	WO	<i>Warm Interface Reset Event</i>
2	WO	New Peer Component Detected Event ( <i>Link RFC</i> packet received, out-of-band signal or event notification, etc.)
3	WO	Exceeded Transient Error Threshold Event (see <i>Interface Structure</i> )
4	WO	Active Link Failure Event (see <i>Interface Structure</i> )
5	WO	Interface Performance Degradation Event—interface is functional but is unable to deliver the configured maximum performance level, e.g., due to lane failure or the physical layer is unable to operate at the maximum supported signaling rate
6	WO	Interface Service Performance Degradation Event—interface is functional but one or more services accessed through this link are unable to deliver the expected performance
59:7	-	RsvdP
63:60	WO	Vendor-defined I-Event Each unsupported Vendor-defined event bit shall be Reserved, and shall be hardwired to 0b. The <i>Core Structure</i> C-UUID is used to identify the vendor-defined interpretation of these bits.

The I-Event Injection field shall be as specified in *I-Event Injection*. If I-Event Injection Bit<sub>K</sub> == 1b and the corresponding I-Event Detect Bit<sub>K</sub> == 1b, then the component shall initiate the corresponding event processing as though the event was detected.

10

- Unsupported I-Event Injection bits shall be hardwired to 0b.
- The component shall indicate Interface ID = 0x0 as the interface on which the event was detected.
- The *Core Structure* C-UUID is used to identify the Vendor-defined bits.

Table 8-67: I-Event Injection

Bit Location	Access	Description
0	WO	Inject <i>Full Interface Reset</i> Event
1	WO	Inject <i>Warm Interface Reset</i> Event
2	WO	Inject New Peer Component Detected Event
3	WO	Inject Exceeded Transient Error Threshold Event
4	WO	Inject Active Link Failure Event
5	WO	Inject Interface Performance Degradation Event—interface is functional but is unable to deliver the configured maximum performance level, e.g., due to lane failure or the physical layer is unable to operate at the maximum supported signaling rate
6	WO	Inject Interface Service Performance Degradation Event—interface is functional but one or more services accessed through this link are unable to deliver the expected performance
7	WO	Reserved
8	WO	Inject Vendor-defined I-Event  Each unsupported Vendor-defined event bit shall be Reserved, and shall be hardwired to 0b.  The <i>Core Structure</i> C-UUID is used to identify the vendor-defined interpretation of these bits.
59:7	-	RsvdP
60	WO	Inject the Vendor-defined I-Event associated with this bit
61	WO	Inject the Vendor-defined I-Event associated with this bit
62	WO	Inject the Vendor-defined I-Event associated with this bit
63	WO	Inject the Vendor-defined I-Event associated with this bit

### 8.21.3. Component Error Logs

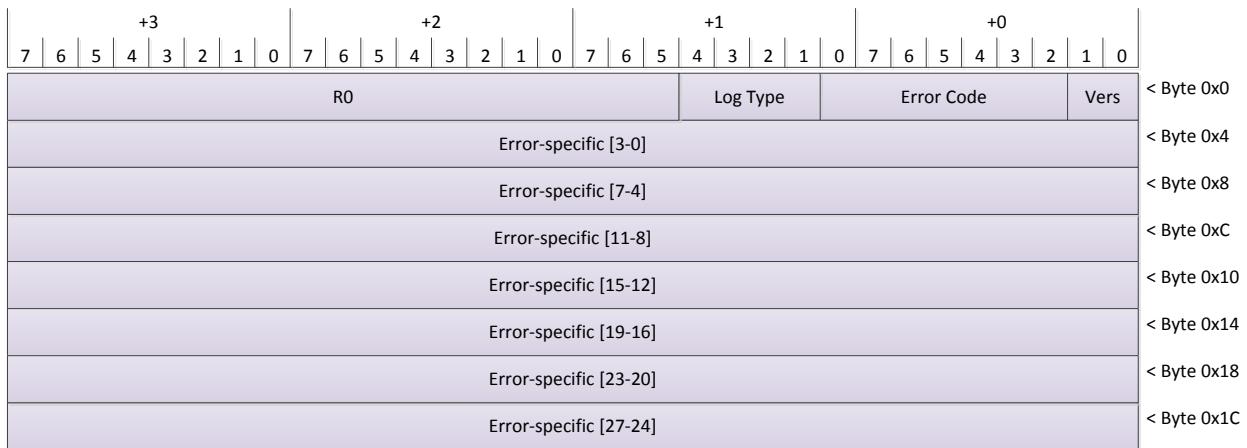


Figure 8-20: Component Error Log Format – Component Error

Table 8-68: Component Error Log – Component Error Format Fields

Field Name	Size (bits)	Value / Bit Location	Description
<b>Version (Vers)</b>	2	0x0	Log Entry Version—log layout and size are interlocked with version
<b>Error Code</b>	7	-	The error associated with this log entry. The error value is K, where K corresponds to Bit <sub>K</sub> within the Component Error Status field.
<b>Log Type</b>	4	-	<p>Identifies the source of the log, and log record format.</p> <p>0x0—Free Entry—For each log entry, the component shall set the Log Type to Free Entry upon power-up / <i>Component Reset</i> (any type). The component sets Log Type to a non-zero value when it consumes a log entry. Software shall set the Log Type to Free Entry once it completes processing the log entry.</p> <p>0x1—Component</p> <p>0x2—Component Fault Injection</p> <p>0x3—Interface—Protocol</p> <p>0x4—Interface—Non-transient Error</p> <p>0x5—Interface Fault Injection</p> <p>0x6—Component Internal Error</p> <p>0x7-0xE—Reserved</p> <p>0xF—Vendor-defined</p> <p>The error log record formats for a Component Internal Error or a Vendor-defined error are component-specific and outside of this specification's scope.</p>

Field Name	Size (bits)	Value / Bit Location	Description
Error-specific R0			A component may create a maximum of three log entries per error—a Component or Component Interface log entry, a Component Internal Error log entry, and / or a Vendor-defined log entry. The Error-specific field in a Component Fault Injection or an Interface Fault Injection log record may be Reserved.
	28 bytes	-	Error-specific bytes. Unused bytes shall be Reserved.
	19	-	RsvdP

7	6	5	4	3	2	1	0		7	6	5	4	3	2	1	0		7	6	5	4	3	2	1	0
Interface ID								Root Cause								Log Type				Error Code				Vers	
Error-specific [3-0]								Error-specific [7-4]								Error-specific [11-8]				Error-specific [15-12]				< Byte 0x0	
Error-specific [3-0]								Error-specific [7-4]								Error-specific [11-8]				Error-specific [15-12]				< Byte 0x4	
Error-specific [3-0]								Error-specific [7-4]								Error-specific [11-8]				Error-specific [15-12]				< Byte 0x8	
Error-specific [3-0]								Error-specific [7-4]								Error-specific [11-8]				Error-specific [15-12]				< Byte 0xC	
Error-specific [3-0]								Error-specific [7-4]								Error-specific [11-8]				Error-specific [15-12]				< Byte 0x10	
Error-specific [3-0]								Error-specific [7-4]								Error-specific [11-8]				Error-specific [15-12]				< Byte 0x14	
Error-specific [3-0]								Error-specific [7-4]								Error-specific [11-8]				Error-specific [15-12]				< Byte 0x18	
Error-specific [3-0]								Error-specific [7-4]								Error-specific [11-8]				Error-specific [15-12]				< Byte 0x1C	
Error-specific [3-0]								Error-specific [7-4]								Error-specific [11-8]				Error-specific [15-12]					

Figure 8-21: Component Error Log Format – Component Interface Error

Table 8-69: Component Error Log – Component Interface Error Format Fields

Field Name	Size (bits)	Value / Bit Location	Description
Version (Vers)	2	0x0	Log Entry Version—log layout and size are interlocked with log entry version
Error Code	7	-	The error associated with this log entry. The error value is K, where K corresponds to Bit <sub>K</sub> within <i>I-Error Status</i> .
Log Type	4	-	See <i>Component Error Log – Component Error Format Fields</i>
Root Cause	7		Provides additional information on the root cause of the error indicated by the Error Code field  0x0—Link-local Protocol Error. Error-specific[7-0] contains a copy of the link-local protocol packet.  0x1—Access Key Violation. Error-specific contains the first 28 bytes of the end-to-end protocol packet.

Field Name	Size (bits)	Value / Bit Location	Description
Interface ID			0x2—Link Hardware Failure. Error-specific may contain vendor-specific diagnostic information.
			0x3—Exceeded Transient Error Threshold. Error-specific field shall be Reserved.
			0x4—Unexpected PHY-Down Transition. Error-specific shall be Reserved.
Error-Specific	12 bytes	-	0x5-0x63—Reserved
			Identifier of the interface that detected the error
			Error-specific field. Unused bytes shall be Reserved.

## 8.22. Component Switch Structure

The Component Switch Structure provides integrated and discrete switch configuration and management. This section also specifies the per interface structures used by *Switches* to perform unicast and multicast packet relay.

- 5 The following are the features and requirements associated with the Component Switch Structure and the associated packet relay table structures:
- A discrete switch component or a component that contains an integrated switch shall support the Component Switch structure.
    - If supported, a component shall support a single Component Switch structure.
- 10 **Developer Note:** *If a switch is composed multiple chips / dies, then the implementation is responsible for distributing / replicating switch configuration information from the single Component Switch structure.*
- The Component Switch structure shall use the *Component Switch Structure Format*.
  - A switch shall provision a Linear Packet Relay Table (LPRT) for each interface used to relay single-subnet unicast packets (GC == 0b).
    - Each *Interface Structure* shall provision a LPRT PTR that contains the non-Null address of the corresponding LPRT.
    - The LPRT shall use the *LPRT / MPRT Format*.
    - A switch shall use the packet's DCID field to directly index the LPRT to locate a route entry row.
    - Each LPRT is enabled / disabled through the corresponding *Interface Structure*.
  - A switch may provision a Multi-subnet Packet Relay Table (MPRT) for each interface used to relay multi-subnet unicast packets (GC is present and GC == 1b). If supported,
    - Each *Interface Structure* shall provision a MPRT PTR that contains the non-Null address of the corresponding MPRT.
    - The MPRT shall use the *LPRT / MPRT Format*.

- A switch shall use the packet's DSID field to directly index the MPRT to locate a route entry row.
- Each MPRT is enabled / disabled through the corresponding *Interface Structure*.
- Wildcard Packet Relay is used to construct linear component topologies with minimal switch logic and resources. Linear component topologies may be inserted into any topology that supports explicit OpClass packets (for component interfaces that support the P2P-Core OpClass, see *Daisy-Chain Topologies*).
  - Upon receipt of an end-to-end packet not destined for the component on an ingress interface enabled for Wildcard Packet Relay, the component relays the packet to the configured egress interface. If the egress interface is not configured (invalid route entry) or does not support Wildcard Packet Relay, then the packet is handled as a UP.
  - A component shall provision two interfaces that support Wildcard Packet Relay per linear topology. Software configures the EI field within each interface's LPRT route entry table with the interface used for to relay packets on a given linear topology.
    - A component may support multiple linear topologies, however, at any given time, an interface enabled for Wildcard Packet Relay shall be configured to relay packets within a single linear topology.
  - If an interface supports LPRT Wildcard Packet Relay, then:
    - LPRT Size shall be hardwired to 0x0.
    - A single LPRT route entry shall be provisioned. This route entry shall be used to relay all end-to-end packets independent of the packet's DCID.
    - The LPRT MHC and HC route entry fields shall be hardwired to 0x0.
    - Max Routes shall be hardwired to 0x0.
  - If an interface supports MPRT Wildcard Packet Relay, then:
    - MPRT Size shall be hardwired to 0x0.
    - A single MPRT route entry shall be provisioned. This route entry shall be used to relay all end-to-end packets independent of the packet's DCID and DSID fields.
    - The MPRT MHC and HC route entry fields shall be hardwired to 0x0.

**Developer Note:** If both ends of a LPRT / MPRT Wildcard Packet Relay linear topology are attached to a switch topology, then software is responsible for preventing packet loops.

	Route N	Route 1	Route 0	LPRT PTR   MPRT PTR
DCID 0   LSID 0 >	EI, VCA, HC, V	...	EI, VCA, HC, V	EI, VCA, HC, V   MHC
DCID 1   LSID 1 >	EI, VCA, HC, V	...	EI, VCA, HC, V	EI, VCA, HC, V   MHC
...				
DCID K   LSID L >	EI, VCA, HC, V	...	EI, VCA, HC, V	EI, VCA, HC, V   MHC

Figure 8-22: LPRT / MPRT Format

- Each LPRT / MPRT route entry row shall contain Max Routes route entries.
  - Each route entry row and individual route entry shall use the *LPRT / MPRT Route Entry Row Format*.
  - The Minimum Hop Count (MHC) field is used to calculate the Computed Hop Count (CHC).
  - The Valid bit determines if the route entry is valid, i.e., the destination component is reachable using this route.

- The Valid bit shall not be set to 1b until all route entry fields have been successfully configured.
- Hop Count (HC) indicates the number of hops to the destination component from the indicated egress interface.
- VCA contains the index to the VCAT entry corresponding to this route entry.
  - The VCAT shall be successfully configured before packet relay is enabled.
- EI contains the interface identifier corresponding to this route entry.
- If the row size is not an integer 4-byte multiple, then it shall be padded by LPRT MPRT Pad Size bits.

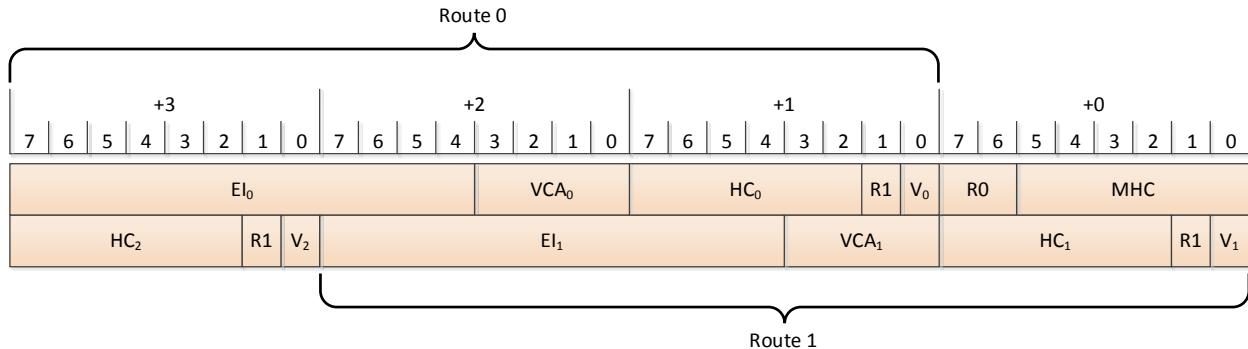


Figure 8-23: LPRT / MPRT Route Entry Row Format

Table 8-70: LPRT / MPRT Route Entry Row Fields

Field Name	Size (bits)	Access	Description
MHC	6	RW	<p>Minimum hop count</p> <p>This field shall be Reserved if hop count configuration is unsupported.</p>
V	1	RW	<p>Valid Entry Indicator</p> <p>0b—Invalid Route Entry (not configured)</p> <p>1b—Valid Route Entry (configured)</p>
HC	6	RW	<p>If a LPRT, then this field contains the number of switch crossings to destination component</p> <p>If a MPRT, then this field contains the switch crossings to a destination subnet</p> <p>This field shall be Reserved if hop count configuration is unsupported.</p> <p>Given the variety of topologies and dynamic operating conditions, management may configure this field with the actual number of switch hops or a relative number using policies outside of this specification's scope. If management uses a relative number, larger relative hop counts shall represent larger physical hop counts, e.g., relative hop count 1 represents more physical hops than</p>

Field Name	Size (bits)	Access	Description
VCA			hop count 0. This is necessary since the component cannot distinguish between actual and relative hop counts.
	4	RW	VC Action is used to select the action column within the VCAT.
EI	12	RW	Egress Interface Identifier
R0	2	-	RsvdP
R1	1	-	RsvdP

- If a component supports VC remapping (see *Unicast Packet Relay*), then the component shall provision a VC Action Table (VCAT) for each interface used to relay unicast packets.
  - A VCAT row is selected using the packet's VC field.
  - Each *Interface Structure* shall provision a VCAT PTR that contains the non-Null address of the corresponding VCAT.
    - If VCAT PTR == Null, then the component does not support VC remapping.
  - The number of VCAT columns shall be UVCATSZ.
    - Each column represents an action table entry that contains a VC Mask (VCM) and a hop-count comparison threshold.
    - A VCM is bit mask where Bit<sub>k</sub> == 1b indicates a VC that can be used, and Bit<sub>k</sub> == 0b indicates a VC that cannot be used.
      1. If VC<sub>k</sub> is unsupported, then Bit<sub>k</sub> shall be hardwired to 0b.
  - The number of VCAT rows shall be the maximum number of VCs supported by the switch.
  - The VCAT and each VCAT entry shall use the *VCAT and VCAT Entry Format*.
  - A switch shall use the VCA field within a given route entry to directly index the VCAT to locate the corresponding VC action.
  - If a row is modified using a Control Write request packet, then the size of the packet's payload shall be an integer 4-byte multiple.

5

10

15

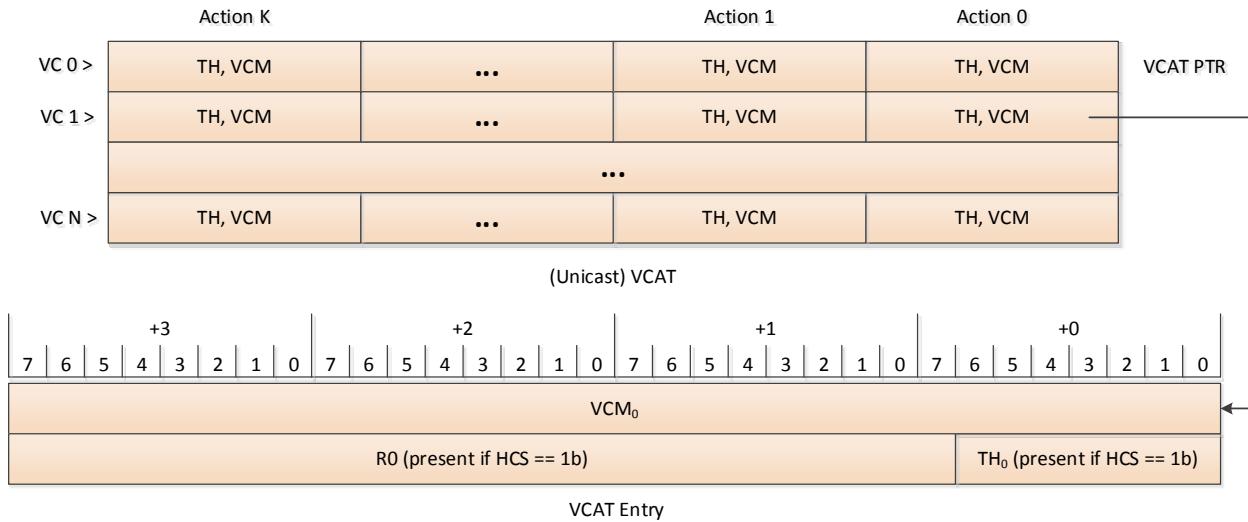


Figure 8-24: VCAT and VCAT Entry Format

Table 8-71: VCAT Fields

Field Name	Size (bits)	M / O	Access	Description
<b>VCM</b>	32	M	RW	VC Mask—Bits corresponding to unsupported VCs shall be hardwired to 0b.
<b>TH</b>	7	M	RW	Configured threshold This field shall be present only if <i>Route Control Field RC CAP 1 HCS == 1b</i> .
<b>R0</b>	25	-	-	RsvdP This field shall be present only if <i>Route Control Field RC CAP 1 HCS == 1b</i> .

- If a component supports multicast packet relay, then:
  - The component shall provision a Multicast Packet Relay Table (MCPRT) which is used to relay single-subnet multicast packets (GC == 0b).
    - The MCPRT shall use the *MCPRT / MSMCPRT Format*.
    - A switch shall use the packet's MGID field to directly index the MCPRT to locate a route entry row.
  - A component may provision a Multi-subnet Multicast Packet Relay Table (MSMCPRT) for each interface used to relay multi-subnet multicast packets (GC == 1b). If supported,
    - The MSMCPRT shall use the *MCPRT / MSMCPRT Format*.
    - A switch shall use the packet's GMGID field to locate a route entry row. Due to the potentially large number of global multicast groups, the mechanism applied to configure a route entry row (e.g., through a software provider library interface) is outside of this specification's scope.

5

10

15

	Egress Interface N	...	Egress Interface 1	Egress Interface 0	MCPRT PTR   MSMCPRT PTR
MGID 0   F(GMGID 0) >	MVCA, V	...	MVCA, V	MVCA, V	
MGID 1   F(GMGID 1) >	MVCA, V	...	MVCA, V	MVCA, V	
...					
MGID K   F(GMGID L) >	MVCA, V	...	MVCA, V	MVCA, V	

Figure 8-25: MCPRT / MSMCPRT Format

- Each MCPRT / MSMCPRT row shall contain Max Interface entries.
- MGID 0 and (Global Multicast Prefix 0 combined with MGID 0) shall be reserved for management. See *Multicast*.
- Each multicast entry row shall use the *MCPRT / MSMCPRT Row Format*.
  - The Valid bit determines if the received multicast packet shall be replicated and transmitted through the corresponding egress interface.
  - MVCA contains the index to the MVCAT entry corresponding to this egress interface. The MVCAT shall be successfully configured before packet relay is enabled.
  - A switch shall use the MVCA field within a given route entry to directly index the MVCAT to locate the corresponding VC action.
  - Each VC Mask (VCM) represents a list of VCs that may be used.
  - If a row is modified using a Control Write request packet, then the size of the packet's payload shall be an integer 4-byte multiple.
  - If the row size is not an integer 4-byte multiple, then it shall be padded by MCPRT MSMCPRT Pad Size bits.

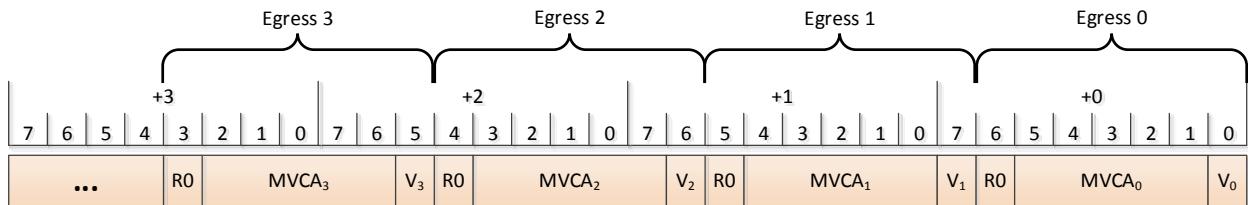


Figure 8-26: MCPRT / MSMCPRT Row Format

Table 8-72: MCPRT / MSMCPRT Row Fields

Field Name	Size (bits)	Access	Description
V	1	RW	Valid bit—Indicates the egress interface participates in the multicast group. If the egress interface participates, then the multicast request packet shall be replicated and transmitted through this egress interface. 0b—Does not participate 1b—Participates
MVCA	5	RW	VC Action as specified in the corresponding MVCAT
R0	1	-	RsvdP

- A component shall provision a Multicast VC Action Table (MVCAT) used to relay multicast packets.
  - The number of MVCAT entries shall be MVCATSZ.
  - The MVCAT shall use the *MVCAT Format*.
  - A switch shall use the MVCA field within a given route entry to directly index the MVCAT to locate the corresponding VC action.
  - Each Multicast VC Mask (MVCM) represents a list of VCs that may be used to remap the request packet's VC field.
  - If a row is modified using a Control Write request packet, then the size of the packet's payload shall be an integer 4-byte multiple.

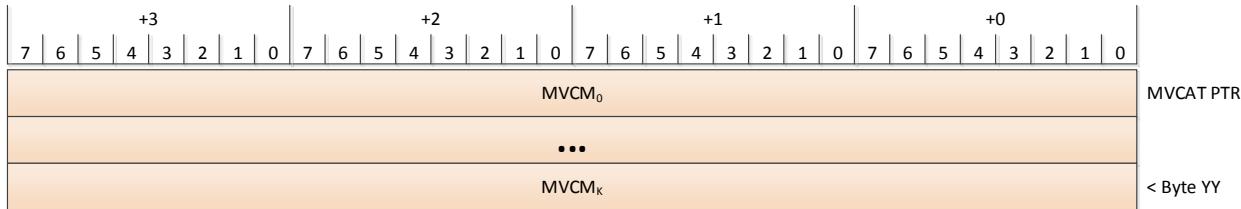


Figure 8-27: MVCAT Format

Table 8-73: MVCAT Fields

Field Name	Size (bits)	M / O	Access	Description
MVCM	32	M	RW	Multicast VC Mask—Bits corresponding to unsupported VCs shall be hardwired to 0b.

- If a switch does not support a multicast packet relay, then the switch shall relay all multicast packets through the configured Default Multicast Egress Interface or the Default Collective Egress Interface.
  - If the Default Collective Egress Interface is not configured, then the packet shall be relayed through the Default Multicast Egress Interface. Only multicast packets whose MGID / GMGID matches the collective multicast group identifier shall be relayed through the Default Collective Egress Interface.
  - If the Default Multicast Egress Interface is not configured and the packet cannot be relayed through the Default Collective Egress Interface, then the packet shall be handled as a UP.
- If a Control Write request packet is used to update a table entry within the LPRT, MPRT, VCAT, MCPRT, MCMCPRT, or MVCAT, then:
  - Management shall update only one table entry / row per Control Write request packet.
  - The Control Write request packet's payload shall be an integer 4-byte multiple.
- A switch may cache LPRT, MPRT, MCPRT, MCMCPRT, VCAT, or MVCAT contents. If cached, then the component shall set *Core Structure Component CAP 1 Cached Component Control Space Structure Support == 1b*.

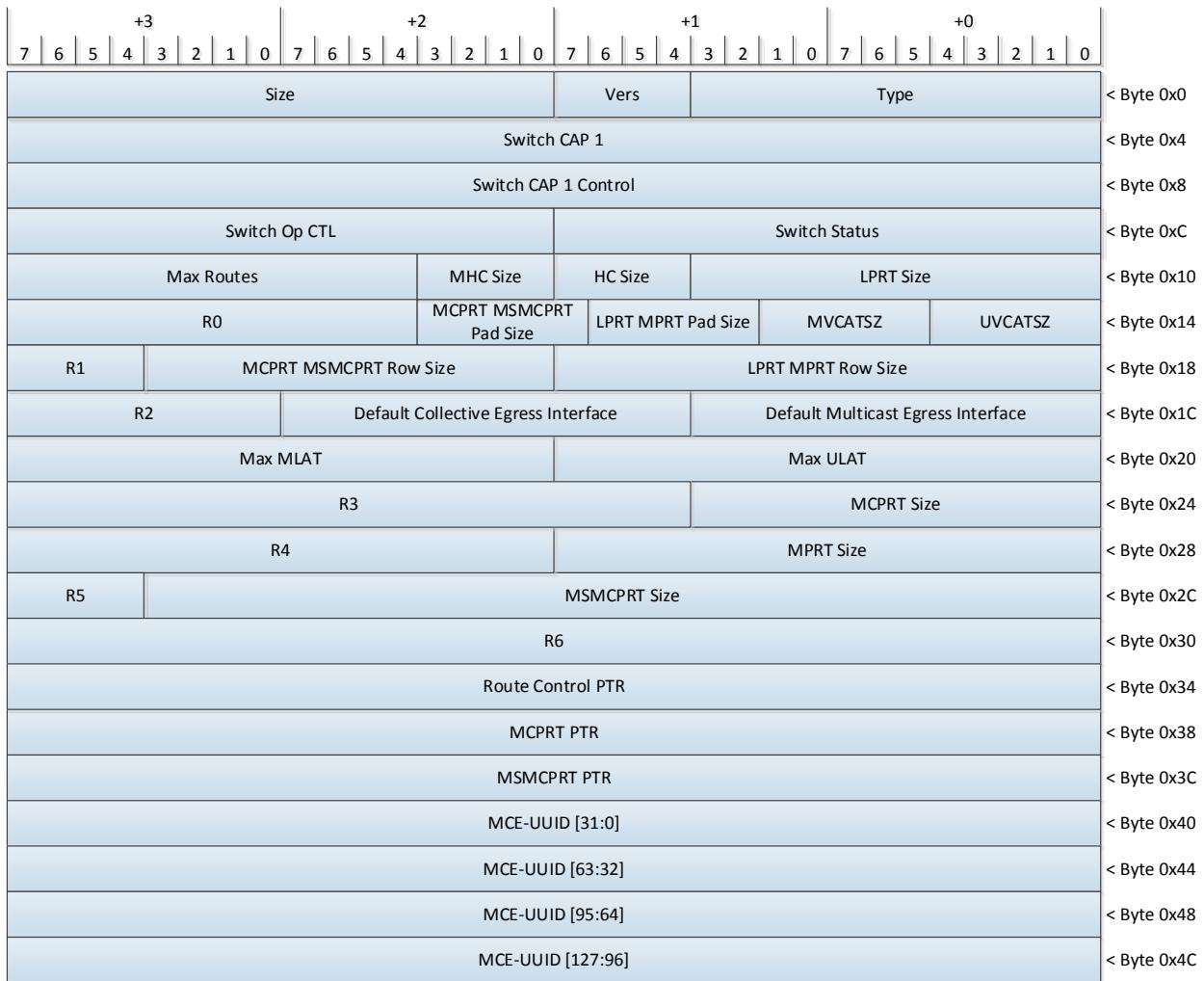


Figure 8-28: Component Switch Structure Format

Table 8-74: Component Switch Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>Version (Vers)</b>	4	0x1	M	RO	Structure Version
<b>Switch CAP 1</b>	32	-	M	RO	See <i>Switch CAP 1 Field</i>
<b>Switch CAP 1 Control</b>	32	-	M	RW	See <i>Switch CAP 1 Control</i>
<b>Switch Status</b>	16	-	M	RW1C	Switch Status
		Bits 15:0			RsvdZ
<b>Switch Op CTL</b>	16	-	M	RW	Switch Operation Control
		Bit 0			Enable Packet Relay

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
LPRT Size					0b—Disable packet relay—any packets not destined for the switch shall be silently discarded 1b—Enable packet relay This bit shall have no impact on <i>Directed Control Space Packet Relay</i> .
					RsvdP
HC Size	12	-	M	RO	Number of Linear Packet Relay Table entries If LPRT Size == 0x0, then the size shall be $2^{12}$ .
MHC Size	4	-	O	RO	Number of bits supported by the underlying component implementation for the HC fields. HC Size shall be $\leq 7$ .
Max Routes	4	-	O	RO	Number of bits supported by the underlying component implementation for the MHC table entry field. If zero, then the MHC shall be Reserved within the corresponding table entries. MHC Size shall be $\leq 7$ .
UVCATSZ	12	-	M	RO	Maximum number of route table entries per route entry row within a LPRT or MPRT. If Max Routes == 0x0, then the number of route entries shall be $2^{12}$ .
MVCATSZ	5	-	M	RO	The number of VC actions per row within the VCAT Table. If UVCATSZ == 0x0, then the number of VC actions shall be $2^6$ .
LPRT MPRT Pad Size	5	-	M	RO	The number of VC actions per row within the MVCAT Table. If MVCATSZ == 0x0, then the number of VC actions shall be $2^6$ .
					Indicates the number of pad bits provisioned per LPRT / MPRT row entry to

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>MCPRT MSMCPRT Pad Size</b>					maintain integer 4-byte alignment. Pad bits shall be Reserved.
	5	-	M	RO	Indicates the number of pad bits provisioned per MCPRT / MSMCPRT row entry to maintain integer 4-byte alignment. Pad bits shall be Reserved.
<b>LPRT MPRT Row Size</b>	16	-	O	RO	The size of a LPRT or MPRT row in bytes. The size shall be an integer 4-byte multiple. The LPRT MPRT Pad-Size indicates the number of pad bits appended to the row to ensure integer 4-byte alignment.
<b>MCPRT MSMCPRT Row Size</b>	16	-	O	RO	The size of a MCPRT or MSMCPRT row in bytes. The size shall be an integer 4-byte multiple. The MCPRT MSMCPRT Pad-Size indicates the number of pad bits appended to the row to ensure integer 4-byte alignment.
<b>Default Multicast Egress Interface</b>	12	-	M	RW	This contains the interface identifier used to relay multicast packets through if the switch does not support multicast packet relay.
<b>Default Collective Egress Interface</b>	12	-	O	RW	This contains the interface identifier used to relay multicast collective packets through if the switch does not support multicast packet relay.
<b>Max ULAT</b>	16	-	M	RO	Maximum Unicast latency—The worst case latency from when the first bit of a unicast Max_Payload_Size packet is received at the ingress interface until the last bit is transmitted from the egress interface. Assumes the egress interface has sufficient flow-control credits to transmit the packet. Unicast Latency = Max ULAT * ULAT Scale
<b>Max MLAT</b>	16	-	O	RO	Maximum Multicast Latency—The worst case latency from when the first bit of a multicast Max_Payload_Size packet is received until the last replicated packet has been transmitted.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					<p>Assumes a multicast packet targets all egress interfaces and all egress interfaces have sufficient flow-control credits to transmit the packet.</p> <p>Multicast Latency = MLAT * MLAT Scale</p> <p>If multicast is unsupported, then field is Reserved.</p>
<b>MCPRT Size</b>	12	-	O	RO	<p>Number of Single-subnet Multicast Packet Relay Table entries</p> <p>If MCPRT Size == 0x0, then the size shall be <math>2^{12}</math>.</p>
<b>MPRT Size</b>	16	-	O	RO	<p>Number of Multi-subnet Packet Relay Table entries</p> <p>If MPRT Size == 0x0, then the number of entries shall be <math>2^{16}</math>.</p>
<b>MSMCRPT Size</b>	28	-	O	RO	<p>Number of Multi-subnet Multicast Packet Relay Table entries</p> <p>If MSMCRPT Size == 0x0, then the number of entries shall be <math>2^{28}</math>.</p>
<b>Route Control PTR</b>	32	-	M	RO	<p>Pointer to the <i>Route Control Field</i>.</p> <p>This structure shall be shared by the <i>Component Destination Table Structure</i> (if supported) and the <i>Component Switch Structure</i>, and provides component-wide route control.</p>
<b>MCPRT PTR</b>	32	-	O	RO	<p>Pointer to the MCPRT (single-subnet Multicast Packet Relay Table).</p> <p>If the switch does not support multicast packet relay, then this field shall be Null.</p>
<b>MSMCPRT PTR</b>	32	-	O	RO	<p>Pointer to the MSMCPRT (multi-subnet Multicast Packet Relay Table).</p> <p>If the switch does not support multi-subnet multicast packet relay, then this field shall be Null.</p>
<b>MCE-UUID</b>	128	-	O	RO	If the switch supports multi-subnet multicast packet relay, then this UUID identifies the encoding method used to

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					access the multi-subnet multicast relay table. If MCE-UUID == 0x0, then the GMGID is used to linearly access the table.
R0	12	-	-	-	RsvdP
R1	4	-	-	-	RsvdP
R2	8	-	-	-	RsvdP
R3	20	-	-	-	RsvdP
R4	4	-	-	-	RsvdP
R5	4	-	-	-	RsvdP
R6	32	-	-	-	Reserved

Table 8-75: Switch CAP 1 Field

Bit Location	Access	Description
0	RO	Control OpClass Packet Filtering Support—If supported, then all interface used to relay packets shall support Control OpClass packet filtering. If supported, then each interface is enabled by setting <i>Interface I-CAP 1 Control Control OpClass Packet Filtering Enable</i> = 1b. 0b—Unsupported 1b—Supported
1	RO	ULAT Scale—Indicates the latency scale associated with the Max ULAT field. 0b—ns 1b—ps
2	RO	MLAT Scale—Indicates the latency scale associated with the Max MLAT field. 0b—ns 1b—ps
3	RO	PCO Communications Support—Indicates if a switch supports the packet relay semantics (see <i>Switches</i> ) required to support <i>PCIe-Compatible Ordering (PCO)</i> communications. 0b—Unsupported 1b—Supported
4	RO	Control Write MSG Packet Filtering Support—If supported, then all interface used to relay packets shall support Control Write MSG packet filtering. If supported, then each interface is enabled by setting <i>Interface I-CAP 1 Control Control Write MSG Packet Filtering Enable</i> = 1b.

Bit Location	Access	Description
5		0b—Unsupported 1b—Supported
	RO	Default Collective Packet Relay Support—Indicates if a switch supports multicast packet relay through the Default Collective Egress Interface. If supported, then the switch filters multicast packets based on the MGID / GMGID reserved for <i>Collective Operations</i> —see <i>Multicast</i> .
		0b—Unsupported 1b—Supported
31:6	-	Reserved

Table 8-76: Switch CAP 1 Control Field

Bit Location	Access	Description
0	RW	MCPRT Enable—Enables or disables single-subnet multicast packet relay. If single-subnet multicast is unsupported, then this field shall be hardwired to 0b. This impacts all single-subnet multicast packet relay.
		0b—Disabled 1b—Enabled
1	RW	MSMCPRT Enable—Enables or disables multi-subnet multicast packet relay. If multi-subnet multicast is unsupported, then this field shall be hardwired to 0b. This impacts
		0b—Disabled 1b—Enabled
2	RW	Default Multicast Packet Relay Enabled—Indicates the Default Multicast Egress Interface has been configured, and enables multicast packet relay through this interface.
		0b—Disabled 1b—Enabled
3	RW	Default Collective Packet Relay Enabled—Indicates the Default Collective Egress Interface has been configured, and enables multicast packet relay through this interface.
		0b—Disabled 1b—Enabled
31:4	-	Reserved

## 8.22.1. Route Control Field

The *Route Control Field* shall be as illustrated. The Route Control field contains a set of sub-fields that provide component-wide route control within a component that supports the *Component Switch Structure* and / or the *Component Destination Table Structure*.

- The Route Control PTR field within the *Component Switch Structure* and / or the *Component Destination Table Structure* shall point a single, shared Route Control field.
- Simultaneous Tables (ST) sub-field determines if the routing algorithms are to use the LPRT associated with the switch ingress interface or the SSDT even when the packet's DSID field does not equal the interface's *Interface Structure* LSID.
- Local Table First (LTF) sub-field determines if the routing algorithms use the LPRT associated with the switch ingress interface or the SSDT when the packet's DSID field does not equal the interface's *Interface Structure* LSID.
  - If ST[Bit<sub>K</sub>] == 0b, then shall set LTF[Bit<sub>K</sub>] = 0b.
  - If ST[Bit<sub>K</sub>] == 1b, then may set LTF[Bit<sub>K</sub>] = 1b.
- Threshold Enable (TE) sub-field determines if the routing algorithms shall use threshold comparison using the thresholds configured within the VCAT corresponding to each supported routing table. Each Bit<sub>K</sub> corresponds to the VC used when accessing a VCAT.
- If a switch and (ST[Bit<sub>K</sub>] == 1b && LTF[Bit<sub>K</sub>] = 0b), then the routing algorithms use the LPRT and the MPRT to make a route selection.
- If a Requester and (ST[Bit<sub>K</sub>] == 1b && LTF[Bit<sub>K</sub>] = 0b), then the routing algorithms use the SSDT and the MSDT to make a route selection.
- If a switch and (ST[Bit<sub>K</sub>] == 1b && LTF[Bit<sub>K</sub>] = 1b), then the routing algorithms use the LPRT first and then the MPRT to make a route selection.
- If a Requester and (ST[Bit<sub>K</sub>] == 1b && LTF[Bit<sub>K</sub>] = 1b), then the routing algorithms use the SSDT first and then the MSDT to make a route selection.

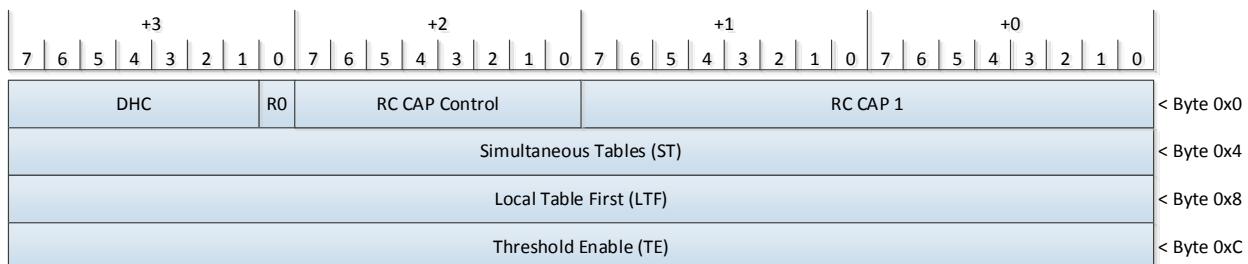


Figure 8-29: Route Control Field

Table 8-77: Route Control Sub-fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
RC CAP 1	16	-	M	RO	Route Control field size
		Bits 1:0			0x0—16 bytes 0x1-0x3—Reserved

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
RC Control	8	Bit 2			MSS—Indicates if the component supports multi-subnet routing. 0b—Unsupported 1b—Supported
		Bit 3			HCS—Indicates if the routing tables have provisioned the MHC field within any routing table (LPRT / MPRT / SSDT / MSDT) entry, and whether the component supports threshold comparison. If the MHC field is provisioned, then it shall be present in all supported routing tables. 0b—Unsupported / MHC Not Provisioned 1b—Supported / MHC Provisioned
		Bits 15:4			RsvdZ
		-	M	RW	
	6	Bit 0			Symmetric Subnet (SS) Enabled—if the component supports multi-subnet packet relay, and the fabric manager has configured each subnet such that all subnets have identical topologies and CID assignment e.g., the topology and CID assignment in subnet 0 is identical to that in subnet 1. 0b—Disabled 1b—Enabled
		Bits 7:1			RsvdP
		-	M	RW	Default Hop Count is the worst-case number of hops within a subnet, e.g., a topology with an asymmetric number of hops between components. The maximum DHC shall be less than or equal to $2^{\text{MHC Size}} - 1$ .
Simultaneous Table (ST)	32	-	M	RW	If Bit <sub>k</sub> == 0b, then the routing algorithms shall not use the LPRT / SSDT when the packet's DSID is not equal to the interface's LSID. If Bit <sub>k</sub> == 1b, then the routing algorithms may use the LPRT / SSDT when the packet's DSID is not equal to the interface's LSID.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Local Table First (LFT)	32	-	M	RW	If $Bit_K == 0b$ , then the routing algorithms may use the LPRT / SSDT when the packet's DSID is not equal to the interface's LSID. If $Bit_K == 1b$ , then the routing algorithms shall use the LPRT / SSDT when the packet's DSID is not equal to the interface's LSID. If MSS == 0b or HCS == 0b, then this field shall be hardwired to 0x0.
Threshold Enable (TE)	32	-	M	RW	If $Bit_K == 0b$ , then the routing algorithms shall not use threshold comparison using the thresholds configured within the VCAT corresponding to each supported routing table. If $Bit_K == 1b$ , then the routing algorithms shall use threshold comparison using the thresholds configured within the VCAT corresponding to each supported routing table. If HCS == 0b, then this field shall be hardwired to 0x0.
R0	1	-	-	-	RsvdP

## 8.23. Component Multicast Structure

The Component Multicast Structure is used to configure and manage Requester and Responder multicast services (see *Multicast*).

If a component supports multicast communications, then the following are the features and requirements associated with the Component Multicast Structure:

- Any Requester or Responder may support the Component Multicast structure.
- The Component Multicast structure shall use the *Component Multicast Structure Format*.
- A component shall support single-subnet unreliable multicast communications, and shall provision a Component Multicast Table (CMT). A Requester uses this structure to generate single-subnet unreliable multicast request packets (GC = 0b).
  - The CMT shall use the *Unreliable Multicast Table Format*.
  - A component shall use the multicast group's MGID to index the CMT to locate the CMT table entry, i.e., row.
- A component may support multi-subnet unreliable multicast communications. If supported, a component shall provision a Multi-subnet Component Multicast Table (MSCMT). A Requester uses this structure to generate multi-subnet unreliable multicast request packets (GC = 1b).
  - The MSCMT shall use the *Unreliable Multicast Table Format*.

- A component may use the multicast group's GMGID to index MSCMT, or may apply an implementation-specific mechanism to locate the corresponding the MSCMT table entry. If an implementation-specific mechanism is used, then the MCE-UUID field shall contain a non-zero UUID.
- A component may support single-subnet reliable multicast communications. If supported, a component shall provision Reliable Component Multicast Table (RCMT) located by the RCMPTR field. A Requester uses this structure to generate single-subnet reliable multicast request packets (GC = 0b).
  - The RCMT shall use the *Reliable Multicast Table Format*.
  - A component shall use the reliable multicast group's MGID to directly index the RCMT to locate the RCMT table entry.
- A component may support multi-subnet reliable multicast communications. If supported, a component shall provision a Multi-subnet Reliable Component Multicast Table (MSRCMT). A Requester uses this structure to generate multi-subnet reliable multicast request packets (GC = 1b).
  - The MSRCMT shall use the in *Reliable Multicast Table Format*.
  - A component may use the multicast group's GMGID to index MSRCMT, or may apply an implementation-specific mechanism to locate the corresponding the MSRCMT table entry. If an implementation-specific mechanism is used, then the MCE-UUID field shall contain a non-zero UUID, and the same mechanism shall be used for unreliable and reliable multicast table access.
- Once a multicast table entry is identified, then a Requester takes the following steps:
  - If Valid == 1b, then the component shall transmit one copy of the multicast request packet through each egress interface indicated by the EM (Egress Mask).
  - The VC field shall be copied to the multicast request packet's VC field.
  - If the row is associated with a reliable multicast group, then the Requester updates its retransmission logic to track this request packet; the Requester uses table entry's MTI field to select the corresponding MINTE.
- A Responder uses the multicast table row to validate if it is a member of the multicast group identified by the packet's MGID / GMGID.
  - If a reliable multicast group, then the Responder uses the *Component Destination Table Structure* and the request packet's SCID and VC fields to generate an acknowledgment packet (see *Reliable Multicast Acknowledgment*).
- If a CMT, MSCMT, RCMT, or MSRCMT row is modified using a Control Write request packet, then the size of the packet's payload shall be an integer 4-byte multiple.
- A component may cache CMT, MSCMT, RCMT, or MSRCMT contents. If cached, then the component shall set *Core Structure Component CAP 1 Cached Component Control Space Structure Support* == 1b.

MGID 0 / F(GMIGD 0) >	U-Pad, EM, VC, V	CMPTR / CGMPTR
MGID 1 F(GMIGD 1) >	U-Pad, EM, VC, V	
	...	
MGID K / F(GMIGD N) >	U-Pad, EM, VC, V	

Figure 8-30: Unreliable Multicast Table Format

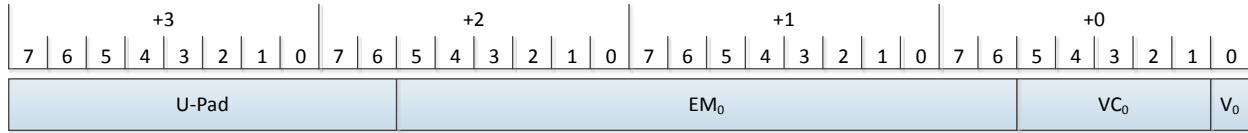


Figure 8-31: Unreliable Multicast Row Format

Table 8-78: Unreliable Multicast Table Fields

Field Name	Value / Bit Location	Size (bits)	M/O	Access	Description
Valid (V)	-	1	M	RW	Valid Interface Entry Indicator 0b—Non-participating multicast group interface 1b—Participating multicast group interface
VC	-	5	M	RW	Virtual Channel identifier to use in unreliable multicast request packets transmitted through the indicated egress interfaces.
EM	-	-	M	RW	Egress Mask—A bit mask where each Bit <sub>K</sub> corresponds to Requester Egress Interface <sub>K</sub> . If Bit <sub>K</sub> == 0b, then a multicast request packet is not transmitted through the corresponding Requester egress interface. If Bit <sub>K</sub> == 1b, then a multicast request packet is transmitted through the corresponding Requester egress interface. If Bit <sub>K</sub> corresponds to an unsupported egress interface, then this bit shall be hardwired to 0b.
U-Pad	-	-	M	RO	U-Pad Size pad bits associated with each unreliable multicast row entry to maintain integer 4-byte alignment.

MGID 0 / F(GMIGD 0) >	RSP-PTR, R-Pad, EM, MTI, Role, VC, V	RCMT PTR / MSRCMT PTR
MGID 1 F(GMIGD 1) >	RSP-PTR, R-Pad, EM, MTI, Role, VC, V	
	...	
MGID K / F(GMIGD N) >	RSP-PTR, R-Pad, EM, MTI, Role, VC, V	

Figure 8-32: Reliable Multicast Table Format

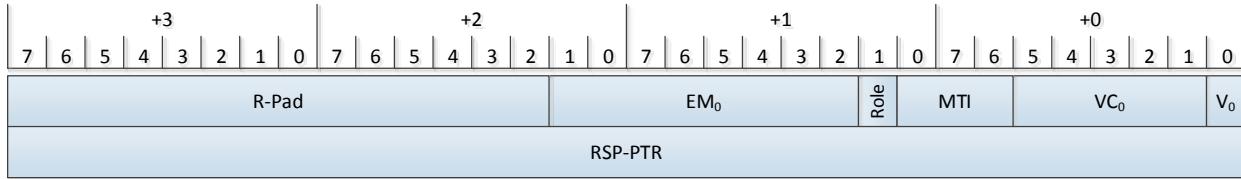


Table 8-79: Reliable Multicast Table Fields

Field Name	M / O	Access	Description
<b>Valid (V)</b>	1	RW	Valid Table Entry—determines if reliable multicast table entry is valid, i.e., the component participates in the reliable multicast group corresponding to this table entry.  If V = 0b, then the corresponding Reliable Multicast Responder Table shall be invalid.  0b—Invalid Table Entry 1b—Valid Table Entry
<b>VC</b>	5	RW	Virtual Channel Identifier to use in reliable multicast request packets transmitted through the indicated egress interfaces.
<b>Role</b>	1	RW	Component Multicast Role—Component's operational role for the corresponding multicast group  0b—Requester 1b—Responder
<b>MTI</b>	3	RW	Reliable Multicast Timer Index—Indicates the MINTE retransmission timer to use
<b>EM</b>	-	RW	Egress Mask—A bit mask where each Bit <sub>k</sub> corresponds to Requester Egress Interface <sub>k</sub> .  If Bit <sub>k</sub> == 0b, then a multicast request packet is not transmitted through the corresponding Requester egress interface.  If Bit <sub>k</sub> == 1b, then a multicast request packet is transmitted through the corresponding Requester egress interface.  If Bit <sub>k</sub> corresponds to an unsupported egress interface, then this bit shall be hardwired to 0b.
<b>RSP-PTR</b>	32	RO	Pointer to the table that identifies the set of Responders participating in this reliable multicast group  Each table shall contain Max RSP table entries.
<b>R-Pad</b>	-	RO	R-Pad Size pad bits associated with each reliable multicast row entry to maintain integer 4-byte alignment.

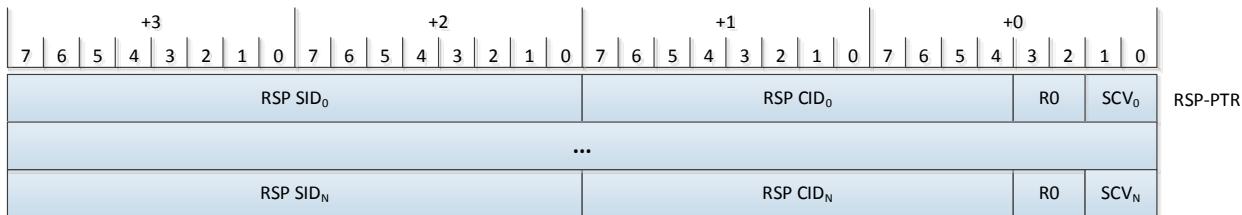


Figure 8-34: Reliable Multicast Responder Table

Table 8-80: Reliable Multicast Responder Table Fields

Field Name	M / O	Access	Description
<b>SCV</b>	2	RW	Determines if the table entry contains a valid Responder CID and SID (if applicable) 0x0—Invalid CID and Invalid SID 0x1—Valid CID and Invalid SID 0x2—Valid CID and Invalid SID 0x3—Reserved
<b>RSP CID</b>	12	RW	CID of the Responder participating in this reliable multicast group
<b>RSP SID</b>	16	RW	SID of the Responder participating in this reliable multicast group
<b>RO</b>	2	-	Reserved

Figure 8-35: Component Multicast Structure Format

Table 8-81: Component Multicast Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>Version (Vers)</b>	4	0x1	M	RO	Structure Version
<b>MCAST CAP 1</b>	16	-	M	RO	Multicast Capabilities
		Bit 0			Reliable multicast Support 0b—Unsupported 1b—Supported
		Bits 5:2			Provisioned Egress Mask Bits 0x0—2 0x1—4 0x2—8 0x3—12 0x4—16

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
MCAST CAP 1 Control	16				0x5—20 0x6—24 0x7—32 0x8—48 0x9—64 0xA—128 0xB—256 0xC—512 0xD—1024 0xE—2048 0xF—4096
					Bits 7:6  Reliable Multicast Role Support  If a Requester or a Requester-Responder, then the <i>Reliable Multicast Table Entry Row Format</i> RSP-PTR shall point to a <i>Reliable Multicast Responder Table</i> containing Max RSP table entries.  If a Responder, then the <i>Reliable Multicast Table Entry Row Format</i> RSP-PTR shall be Null in all, and the <i>Reliable Multicast Responder Table</i> shall not be provisioned. Further, the Max RSP field shall be Reserved.
					0x0—Requester 0x1—Responder 0x2—Requester-Responder 0x3—Reserved
					Bits 16:8  Reserved
					Multicast Capability Controls  Unreliable Multicast Enable  0b—Disable unreliable multicast operations 1b—Enable unreliable multicast operations  Reliable Multicast Enable  0b—Disable reliable multicast operations 1b—Enable reliable multicast operations  If reliable multicast is supported, then this bit determines if each entry in the Responder tracking table consists of a Responder CID or a Responder CID plus SID.  0b—Responder CID 1b—Responder CID plus SID

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
CMT Size		Bits 15:3			RsvdP
	12	-	M	RO	<p>Indicates the number of single-subnet unreliable multicast table entries associated with this component.</p> <p>If CMT Size == 0x0, then the number of entries shall be <math>2^{12}</math>.</p>
Max RSP	4	-	M	RO	<p>Used to calculate the maximum number of Responders a Requester can track per reliable multicast group.</p> <p>Number of Responders = <math>2^{\text{Max RSP}}</math></p> <p>If Max RSP == 0x0, then Number of Responders shall equal <math>2^{16}</math>.</p> <p>If reliable multicast is unsupported, then this field shall be Reserved.</p>
RCMT Size	12	-	M	RO	<p>Indicates the number of single-subnet reliable multicast table entries associated with this component.</p> <p>If RCMT Table Size == 0x0, then the number of entries shall be <math>2^{12}</math>.</p> <p>If reliable multicast is unsupported, then this field shall be Reserved.</p>
MINTE <sub>N</sub>	16	-	O	RW	<p>Used to calculate a reliable multicast retransmission timer.</p> <p>Timer = MINTE * <i>Core Structure Component CAP 1 Timer Unit</i> (-0%, + 25%).</p> <p>If reliable multicast is unsupported, then MINTE shall be Reserved.</p>
CMT PTR	32	-	M	RO	Indicates the start of the single-subnet unreliable multicast table
MSCMT PTR	32	-	M	RO	Indicates the start of the multi-subnet unreliable multicast table or Null if unsupported
RCMT PTR	32	-	M	RO	Indicates the start of the single-subnet reliable multicast table or Null if unsupported
MSRCMT PTR	32	-	M	RO	Indicates the start of the multi-subnet reliable multicast table or Null if unsupported

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
MSCMT Size	28	-	O	RO	<p>Indicates the number of multi-subnet unreliable multicast table entries provisioned by this component</p> <p>If MSCMT Size == 0x0, then the number of entries shall be <math>2^{28}</math>.</p> <p>If multi-subnet unreliable multicast is unsupported, then this field shall be Reserved.</p>
MSRCMT Size	28	-	O	RO	<p>Indicates the number of multi-subnet reliable multicast table entries provisioned by this component.</p> <p>If MSRCMT Size == 0x0, then the number of entries shall be <math>2^{28}</math>.</p> <p>If multi-subnet reliable multicast is unsupported, then this field shall be Reserved.</p>
U-Pad Size	5	-	M	RO	Indicates the number of pad bits provisioned per unreliable multicast row entry to maintain integer 4-byte alignment. Pad bits shall be Reserved.
R-Pad Size	5	-	M	RO	Indicates the number of pad bits provisioned per reliable multicast row entry to maintain integer 4-byte alignment. Pad bits shall be Reserved.
MCGPT	16	-	M	RW	<p>MCGPT indicates the minimum amount of time that shall elapse before a Requester may wrap the Tx Seq to 0x0 for a given multicast group in order to reuse sequence numbers.</p> <p>A Requester shall stall multicast packet generation for a given multicast group if the associated Tx Seq value would wrap to 0x0 before the MGPT expired.</p> <p>If an unreliable multicast group, a Requester may silently discard stalled multicast packets.</p> <p>If a reliable multicast group, a Requester may delay transmitting stalled multicast packets up to the time equivalent to the maximum number of retransmission attempts prior to informing the application or application middleware of the stalled condition.</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
MCE-UUID					MC Ghost Packet Timer = MCGPT ns
	128	-	O	RO	If the component supports multi-subnet multicast, then this UUID identifies the encoding method used to access the multi-subnet multicast tables. If MCE-UUID == 0x0, then the GMGID is used to linearly access the table.
R0	8	-	-	-	Reserved
R1	4	-	-	-	Reserved
R2	4	-	-	-	Reserved
R3	32	-	-	-	Reserved
R4	18	-	-	-	Reserved
R5	16	-	-	-	RsvdP

## 8.24. Component Extension Structure

The Component Extension Structure enables the number of Control Space structures to be expanded beyond what is accessible directly through the *Core Structure*.

The following are the features and requirements associated with the Component Extension Structure:

- A component may support the Component Extension Structure.
- The Component Extension structure shall use the *Component Extension Structure Format*.
- If supported, then the first Component Extension structure shall be located using the *Core Structure Component Extension PTR* field.
- Additional Component Extension structures may be provisioned. These structures should be located using the *Next Component Extension PTR* field, or they may be located using any Control Structure PTR field within this structure or within the *Core Structure*.

5

10

+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	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	7 6 5 4 3 2 1 0		
Next Component Extension PTR				Size		Vers	Type		
Control Structure PTR 1				Control Structure PTR 0		< Byte 0x0			
Control Structure PTR 3				Control Structure PTR 2		< Byte 0x8			
Control Structure PTR 5				Control Structure PTR 4		< Byte 0x10			
Control Structure PTR 7				Control Structure PTR 6		< Byte 0x18			
Control Structure PTR 9				Control Structure PTR 8		< Byte 0x20			
R0	Control Structure PTR 10						< Byte 0x28		
R1	Control Structure PTR 11						< Byte 0x30		

Figure 8-36: Component Extension Structure Format

Table 8-82: Component Extension Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>Version (Vers)</b>	4	0x1	M	RO	Structure Version
<b>Control Space Structure Pointers</b>	32 / 48	-	O	RO	Set of pointers to additional Control Space structure. A Null pointer indicates there is no corresponding structure present.
<b>Next Component Extension PTR</b>	32	-	O	RO	Pointer to the next Component Extension structure or Null.
<b>R0</b>	16	-	-	-	Reserved
<b>R1</b>	16	-	-	-	Reserved

## 8.25. Component Statistics Structure

The Component Statistics Structure provides configuration and control of standard statistics as well as access to vendor-defined statistics.

The following are the features and requirements associated with the Component Statistics Structure:

- A component may support zero or more Component Statistics Structures.
- The Component Statistics structure shall use the *Component Statistics Structure Format*.
- If the structure type (ST-Type) indicates the structure corresponds to an OpClass, then each statistic corresponds to an OpClass-specific OpCode. For example, if ST-Type = Core 64 OpClass then the OpCode 0 statistics field corresponds to Core 64 OpClass OpCode 0.
  - Statistics corresponding to OpCodes within a given OpClass that are reserved or unsupported should be hardwired to zero.
- If the structure type indicates the structure corresponds to a set of vendor-defined statistics, then each field is treated as a vendor-defined value.

- A component may support multiple statistics structures. These are located through the Next Statistics Pointer.
- Each Component Statistics structure is individually managed. Statistics operation specific to a given structure instance may be enabled, disabled, reset, etc. through the SCTL field.
- All Component Statistic Structure fields shall be set to zero whenever the structure the component is reset or exits a low-power state that precludes maintaining statistics.
- Statistic fields should be reset if the component enters a low power state that precludes maintaining this structure.
- Disabling statistics shall pause statistics gathering, but shall not reset the statistics field.
- Enabling statistics post disabling shall resume statistics gathering using the present values of each statistics field.
- All event statistic counters shall be incremented by one for each event. Counters shall not roll-over to zero. Counters may be set to zero by a hardware reset or state change or by management update.

ST-Type	CSTAT Status	CSTAT Control	CSTAT CAP1	Size	Vers	Type	< Byte 0x0
C-Snapshot PTR				Next Statistics PTR			
R0						EE Retry Exceeded	< Byte 0x10
R1							
OpCode 1 / Vendor-Defined 1				OpCode 0 / Vendor-Defined 0			
OpCode 3 / Vendor-Defined 3				OpCode 2 / Vendor-Defined 2			
OpCode 5 / Vendor-Defined 5				OpCode 4 / Vendor-Defined 4			
OpCode 7 / Vendor-Defined 7				OpCode 6 / Vendor-Defined 6			
OpCode 9 / Vendor-Defined 9				OpCode 8 / Vendor-Defined 8			
OpCode 11 / Vendor-Defined 11				OpCode 10 / Vendor-Defined 10			
OpCode 13 / Vendor-Defined 13				OpCode 12 / Vendor-Defined 12			
OpCode 15 / Vendor-Defined 15				OpCode 14 / Vendor-Defined 14			
OpCode 17 / Vendor-Defined 17				OpCode 16 / Vendor-Defined 16			
OpCode 19 / Vendor-Defined 19				OpCode 18 / Vendor-Defined 18			
OpCode 21 / Vendor-Defined 21				OpCode 20 / Vendor-Defined 20			
OpCode 23 / Vendor-Defined 23				OpCode 22 / Vendor-Defined 22			
OpCode 25 / Vendor-Defined 25				OpCode 24 / Vendor-Defined 24			
OpCode 27 / Vendor-Defined 27				OpCode 26 / Vendor-Defined 26			
OpCode 29 / Vendor-Defined 29				OpCode 28 / Vendor-Defined 28			
OpCode 31 / Vendor-Defined 31				OpCode 30 / Vendor-Defined 30			

Figure 8-37: Component Statistics Structure Format

Table 8-83: Component Statistics Structure Fields

Field Name	Size (bits)	Value / Bit Location	M/ O	Access	Description
<b>Version (Vers)</b>	4	0x1	M	RO	Structure Version
<b>CSTAT CAP 1</b>	8	-	M	RO	Component Statistics Capabilities
		Bit 0			Component Statistics Snapshot Support—Indicates if the component supports copying this statistics associated with this structure to the location specified by the C-Snapshot Pointer. This enables management to capture all statistics at a single point in time. 0b—Unsupported 1b—Supported
		Bits 7:1			RsvdP
<b>CSTAT Control</b>	8	-	M	RW	Control statistics operations
		Bit 0			Enable Statistics Gathering—Impacts only statistics associated with this structure 0b—Disable 1b—Enable
		Bit 1			Reset All Statistics—Impacts only statistics associated with this structure 1b—Reset Reads of this bit shall return 0b.
		Bit 2			Initiate Component Statistics Snapshot—if supported, copy all component statistics associated with this structure to the location identified by the C-Snapshot Pointer field. Statistics are contiguously copied starting from byte 0x10 through the end of this instance of the Component Statistics structure.  During the copy process, a component should continue to update all relevant statistics. As soon as the C-Snapshot Interval field is copied and no later than snapshot operation completion, the C-Snapshot Interval field shall be reset to 0x0 and resume operation. Further, once the snapshot operation completes, all component statistics associated with this Component Statistics structure shall be reset to zero.

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description	
CSTAT Status	8	Bits 7:3	M	-	1b—Perform Snapshot Reads of this bit shall return 0b.	
					RsvdP	
		Bit 0		-	Structure Status field	
				RW1C	Statistics Reset—Indicates if the statistics fields were reset to zero 0b—Incomplete / Not initiated 1b—Reset Completed	
				RW1C	Snapshot Status—Indicates if a component statistics snapshot request has been completed. If snapshot functionality is unsupported, then this field shall be hardwired to 0b. 0b—Incomplete / Unsupported 1b—Completed	
		Bit 1		-	RsvdZ	
				-		
				-		
				-		
		Bits 7:2		RO	Identifies the OpClass or vendor-defined statistics associated with this structure 0x0—Core 64 OpClass 0x1—Control OpClass 0x2—Atomic 1 OpClass 0x3—LDM 1 OpClass 0x4—Advanced 1 OpClass 0x5—Advanced 2 OpClass 0x6-0x14—Reserved OpClass 0x15—CTXID OpClass 0x16—Multicast OpClass 0x17—SOD OpClass 0x18—Vendor-defined 1 OpClass 0x19—Vendor-defined 2 OpClass 0x1A—Vendor-defined 3 OpClass 0x1B—Vendor-defined 4 OpClass 0x1C—Vendor-defined 5 OpClass 0x1D—Vendor-defined 6 OpClass 0x1E—Vendor-defined 7 OpClass 0x1F—Vendor-defined 8 OpClass 0x20—P2P-Core OpClass 0x21—P2P Vendor Defined OpClass 0x22-0x2F—Reserved	
				RO		
ST-Type	8	-	M	RO		

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description
					0x30-0x3F—Vendor-defined Statistics Structure
<b>Next Statistics PTR</b>	32	-	O	RO	Pointer to next component statistics structure or Null
<b>C-Snapshot PTR</b>	32	-	O	RO	If supported, then this field points to the location where the component statistics associated with this structure are copied upon command. If unsupported, then this field is Null.
<b>OpCode / Vendor Defined 0—15</b>	32	-	M	RO	Each field corresponds to either an OpCode or a vendor-defined statistic based on the ST-Type.
<b>EE Retry Exceeded</b>	16	-	M	RO	Number of times end-to-end packet retry exceeded the component's maximum. A component may treat this field as the sum across all supported OpClasses or on a per OpClass basis.
<b>R0</b>	48	-	-	-	RsvdP
<b>R1</b>	64	-	-	-	Reserved

## 8.26. Component Image Structure

The Component Image Structure is used to locate and manage an image within the component's Control Space or Data Space. Images may be an operating system boot image, a firmware image, an accelerator executable, etc.

- 5 The following are the features and requirements of the Component Image Structure:
- Any component type may support the Component Image structure.
  - A component may support multiple Component Image structures.
  - A Component Image structure contains a pointer to a table of Meta data entries, where each Meta data describes the corresponding image and its location.
  - 10 • The Component Image structure shall use the *Component Image Structure Format*.
  - Each image may uniquely identified by an Image UUID.
  - To validate image integrity, each image may be accompanied by the results of a hash function.
    - o A component may support zero or more hash functions.
    - o The Applied Hash field shall indicate the contents of the Image Hash field. If zero, i.e., no hash function was applied, then Image Hash field shall be Reserved.
    - o If one of the supported hash function was applied, then the Image Hash field shall contain the results.
  - 15 • A component may support image placement within its Control Space.

- A component may support image placement within its Data Space.
- Each Image Table entry may contain an Image Table entry index of an alternative image to use in case the image is invalid or is unable to be executed. If the image is invalid or cannot be executed, then the component shall use the alternative image (if configured).
- To ensure hardware-enforced image access isolation, images shall be isolated in dedicated pages as described below. This enables a component to use a *Region Key (R-Key)* at its discretion to enforce access isolation. Isolation enables a component to support an image management service which enables multiple components to independently read an image or enables a service provider to write an image to another.
  - Images in Data Space shall be isolated in pages with a minimum page size of 4 KiB. The page size should be one of the mandatory sizes for ZMMUs. See *Memory Management*.
  - If the component implements at least one *Component C-Access Structure*, each image in Control Space shall be isolated in pages matching the page size of the associated *Component C-Access Structure*.

5

10

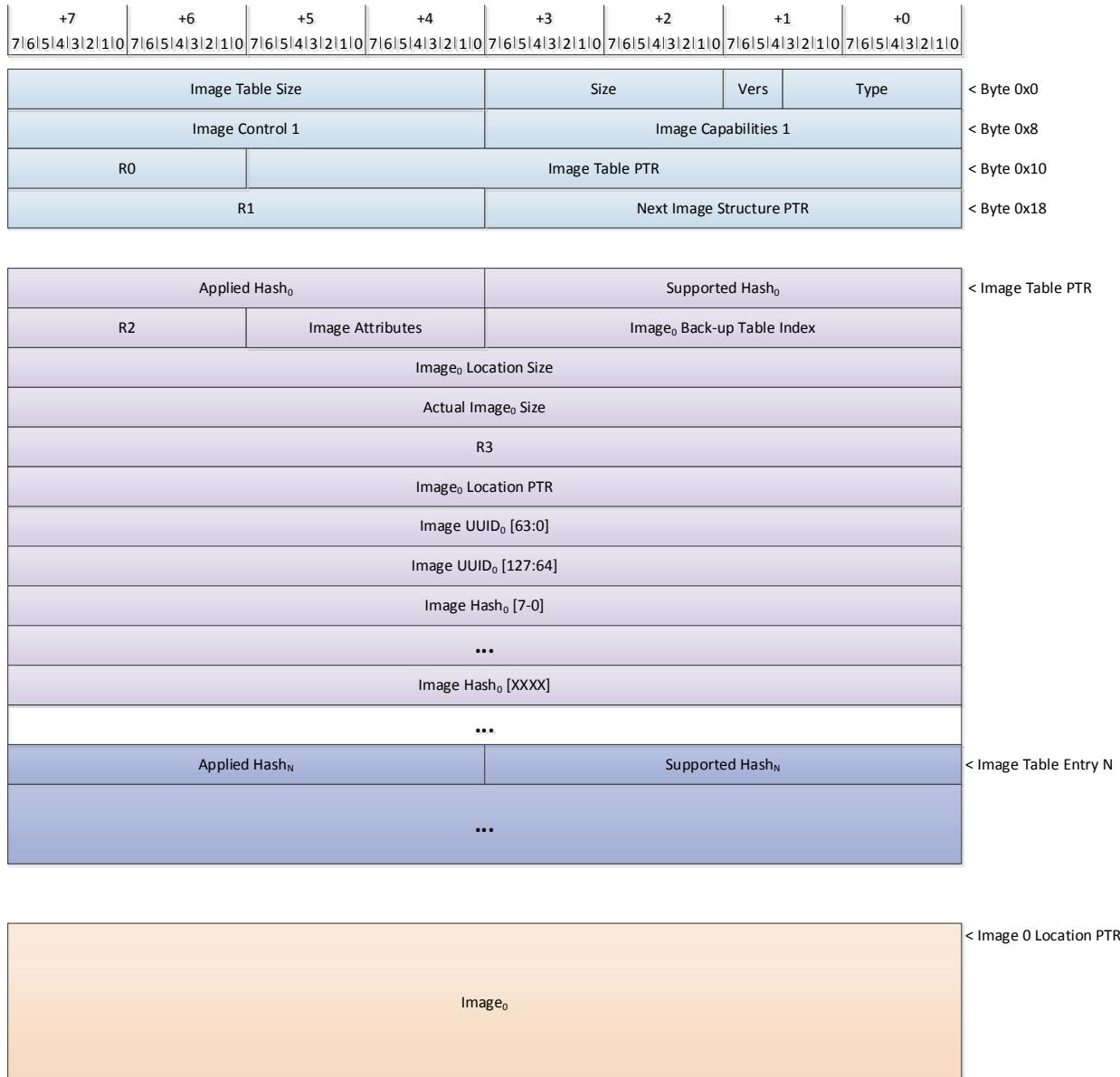


Figure 8-38: Component Image Structure Format

Table 8-84: Component Image Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>Version (Vers)</b>	4	0x1	M	RO	Structure Version
<b>Image Table Size</b>	32	-	M	RO	Number of image table entries. If Image Table Size == 0x0, then the number of image table entries equals $2^{32}$ .
<b>Image CAP 1</b>	32	-	M	RO	Image Structure Capabilities 1
		Bit 0			Read-only Image Location

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description		
					0b—Image location has read-write properties 1b—Image location is read-only, i.e., the component does not allow the image to be updated.		
		Bit 1			Control Space Image Location Support—indicates if the component supports image placement within Control Space. 0b—Unsupported 1b—Supported		
		Bit 2			Data Space Image Location Support—indicates if the component supports image placement within Data Space. 0b—Unsupported 1b—Supported		
		Bits 7:3			Maximum Image Hash Field Size 0x0—128 bits 0x1—256 bits 0x2—512 bits 0x3—1024 bits 0x4-0xF—Reserved		
		Bits 31:8			Reserved		
Image CAP 1 Control	32	-	M	RW	Image Structure Capabilities 1 Control		
		Bits 31:0			RsvdP		
Next Image Structure PTR	32	-	M	RO	Pointer to the next <i>Component Image Structure</i> . If unsupported, then this field shall be Null.		
Image Control	32	-	M	RW	Image controls associated with a given Image Table entry		
		Bit 0			Enable Executable 0b—Disable 1b—Enable  Upon enablement, the component should validate the image by executing the Applied Hash function and verifying its results with the Image Hash field.  If the results do not match, then the component shall notify management (if so configured, see the <i>Component Error and Signal Event Structure</i> ).		
					Delete Image		

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Supported Hash					1b—Deletes the stored image and overwrites the entire image location range with zero. Reads of this bit shall return 0b.
		Bit 2			Image Address Space Location 0b—Control Space 1b—Data Space (if supported)
		Bit 3			Back-up Image Configured—Determines if the corresponding Image Back-up Table Index has been successfully configured. 0b—Not configured 1b—Configured
		Bits 31:4			RsvdP
	32	-	RO	Bit mask indicating the supported hash functions associated with this image. If Bit <sub>k</sub> == 1b then the component supports the corresponding hash function; if Bit <sub>k</sub> == 1b, then it does not.	
		Bit 0			SHA-224
		Bit 1			SHA-256
		Bit 2			SHA-384
		Bit 3			SHA-512
		Bit 4			SHA-512/224
		Bit 5			SHA-512/256
		Bit 6			SHA3-224
		Bit 7			SHA3-256
		Bit 8			SHA3-384
		Bit 9			SHA3-512
		Bits 31:10			Reserved
Applied Hash	32	-	-	RW	Bit mask indicating the hash function applied to the associated image with the results placed in the Image Hash field. If Bit <sub>k</sub> == 1b, then the corresponding hash function was applied. All other bits shall be set to 0b.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Image Hash					If no hash function was applied, then this field shall be set to 0x0 and the corresponding Image Hash field shall be Reserved.
		Bit 0			SHA-224
		Bit 1			SHA-256
		Bit 2			SHA-384
		Bit 3			SHA-512
		Bit 4			SHA-512/224
		Bit 5			SHA-512/256
		Bit 6			SHA3-224
		Bit 7			SHA3-256
		Bit 8			SHA3-384
		Bit 9			SHA3-512
		Bits 31:10			RsvdP
Image Back-up Table Index	32	-	O	RW	Contains the index of the Image Table entry that may be used to locate an alternative image. This alternative image shall be used only if this image is invalid (e.g., hash validation failure) or is unable to be executed by used by the component.
Image Attributes	16	-	M	RW	Attributes of the corresponding image
					Image Space—Determines if the image is located in Control Space or Data Space. 0b—Control Space 1b—Data Space
					RsvdP
Image Location Size	64	-	M	RO	Size (in bytes) of the location (Image Location Pointer) where an image may be placed in the indicated address space.  If the location is in Control Space, then size shall be less than $2^{52}$ bytes.
Actual Image Size	64	-	M	RW	Size (in bytes) of the image stored at Image Location Pointer else Null.  If the location is in Control Space, then size shall be less than $2^{52}$ bytes.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Image Location PTR	64	-	M	RO	<p>Pointer to the image's location within the indicated address space.</p> <p>Images shall be located on integer 4096-byte address multiples, i.e., the lower 12 bits of this field shall be set to zero).</p> <p>If the location is within Control Space, then the maximum pointer size shall be 52 bits; unused upper bits shall be set to zero.</p>
Image UUID	128	-	M	RW	UUID associated with the stored image. An Image UUID should be unique per image.
Image Hash	-	-	M	RW	<p>Results of a hash function used to validate contents. Results [N:0] shall be placed at Image Hash [N:0] starting with Results [0] to Image Hash [0]. Unused upper Image Hash bits shall be Reserved.</p> <p>The size of this field is communicated by the Maximum Image Hash Field Size sub-field.</p> <p>If a hash function was not applied, then this field shall be Reserved.</p>
R0	16	-	-	-	Reserved
R1	32	-	-	-	Reserved
R2	16	-	-	-	Reserved
R3	64	-	-	-	Reserved

## 8.27. Component Precision Time Structure

The Component Precision Time Structure is used to configure *Precision Time* services.

The following are the features and requirements of the Component Precision Time Structure:

- Any component type may support the Component Precision Time Structure.
- A component may support multiple Component Precision Time Structures—one per Precision Time Domain (PTD). These structures shall be located using the Next PT PTR.
- A solution may provision multiple GTC per PTD, e.g., for resiliency. If multiple GTC are provisioned per PTD, then:
  - One Component Precision Time structure shall be provisioned per GTC per PTD.
  - Only one Component Precision Time structure shall be enabled to transmit PTREQ and / or PTRSP packets at a given time.
- The Component Precision Times structure shall use the *Precision Time Structure Format*.
- A component may act as a Precision Time Requester, a Precision Time Responder, or as both.

- A component shall not transmit PTREQ request packets unless Precision Time Requester Supported and Precision Time Requester Enabled are set to 1b.
- A component shall not execute PTREQ request packets nor transmit PTRSP response packets unless Precision Time Responder Supported and Precision Time Responder Enabled are Set.
- A component shall not act as a GTC unless Precision Time GTC Supported and Precision Time GTC Enabled are set to 1b.
- Management may configure an alternative Responder if the component supports multiple interfaces. A component may migrate to the alternative if communications with the Responder fail or if management informs the component to migrate. Migration may result in the component operating in a different PTD.
- The Domain PT Granularity represents the worst-case granularity supported by the PTD. This value may be equal to or greater than the component's Precision Time Granularity. By default, the Domain PT Granularity shall equal the component's PT Granularity.

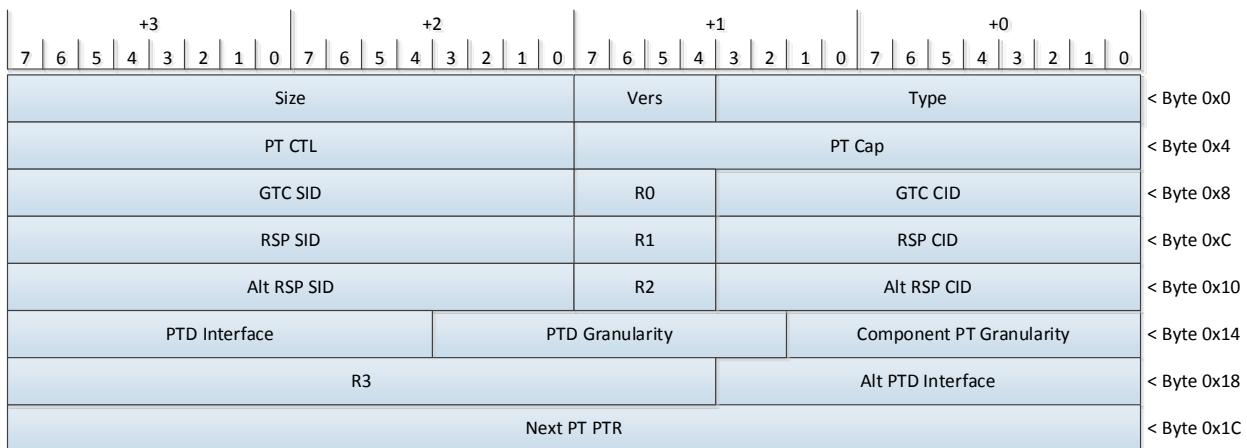


Figure 8-39: Precision Time Structure Format

Table 8-85: Precision Time Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Version (Vers)	4	0x1	M	RO	Structure Version
PT Cap	16	Bit 0	MC	RO	Precision Time Capabilities
					Precision Time Requester Support 0b—Unsupported 1b—Supported If a STC, then this field shall be hardwired to 1b. If a BTC and not a GTC, then this field shall be hardwired to 1b.
					Precision Time Responder Support 0b—Unsupported

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
PT CTL					1b—Supported If a GTC or a BTC, then this field shall be hardwired to 1b.
		Bit 2	MC		Precision Time GTC Support 0b—Unsupported 1b—Supported If a GTC, then this field shall be hardwired to 1b.
		Bit 3	M		Component Precision Time Granularity Unit 0b—ns (default) 1b—ps
		Bits 15:4	-		Reserved
	16			RW	Precision Time Control
		Bit 0	O	-	Precision Time Requester Enabled 0b—Component shall not transmit PTREQ 1b—Component may transmit PTREQ packets
		Bit 1	O		Precision Time Responder Enabled 0b—Component shall not transmit PTRSP 1b—Component may transmit PTRSP packets
		Bit 2	O		Precision Time GTC Enabled—Only one component shall be enabled as the GTC within a given PTD. 0b—Non-GTC Component 1b—GTC Component
		Bit 3	O		Migrate to Alternative Responder 1b—Component re-establishes Precision Time using the component identified by the Alt RSP CID. Reads of this bit shall return 0b.
		Bit 4	M		PTD Granularity Unit—Shall be configured by management. 0b—ns (default) 1b—ps
		Bit 5	O		GTC CID Location—Determines if the GTC is co-located within the same subnet 0b—Co-located, use GTC CID 1b—Not Co-located, use [GTC CID , GTC SID]
		Bits 15:6	-		RsvdP

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>GTC CID</b>	12	-	M	RW	If GTC CID Location == 0b, then this field contains the CID of the GTC of the active PTD.
<b>GTC SID</b>	16	-	O	RW	If multi-subnet communication is supported, then this field contains the SID of the subnet where the GTC of the active PTD is located.
<b>RSP CID</b>	12	-	M	RW	CID of the Precision Time Responder for this Precision Time Requester.
<b>RSP SID</b>	16	-	O	RW	If multi-subnet communication is supported, then this field contains the SID of the subnet where the Precision Time Responder for this component is located.
<b>Alt RSP CID</b>	12	-	M	RW	CID of the Precision Time Responder for this Precision Time Requester.  Only applicable if the component supports at least one additional interface that is attached to a Precision Time Responder.
<b>Alt RSP SID</b>	16	-	O	RW	If multi-subnet communication is supported, then this field contains the SID of the subnet where the alternative Precision Time Responder for this component is located.  Only applicable if the component supports at least one additional interface that is attached to a Precision Time Responder.
<b>PTD Interface</b>	12	-	M	RW	Component interface identifier used to exchange Precision Time request and response packets for the associated PTD.
<b>Alt PTD Interface</b>	12	-	M	RW	Component interface identifier of the alternative Precision Time Responder. Only applicable if the component supports at least one additional interface that is attached to a Precision Time Responder.
<b>Component PT Granularity</b>	10	-	M	RO	Precision Time Granularity—Indicates the period of the component's clock. Valid values are 1-1023. The period's unit of time is Precision Time Granularity Unit.
<b>PTD Granularity</b>	10	-	M	RO	Precision Time Domain Granularity—Indicates the Precision Time granularity used by the PTD's GTC. PTD granularity unit of time is the PTD Granularity Unit.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Next PT PTR	32	-	O	RO	Pointer to the next Component Precision Time structure or Null if not present.
R0	4	-	-	-	RsvdP
R1	4	-	-	-	RsvdP
R2	4	-	-	-	RsvdP
R3	20	-	-	-	RsvdP

## 8.28. Component Mechanical Structure

Components embodied within a mechanical module may be connected through an intermediate mechanical connector. *Example Mechanical Connectivity* illustrates a Requester (e.g., a processor) interface routed to a mechanical connector that is attached to a mechanical module with an on-board Responder. The architecture can support a variety of mechanical modules; these are specified in separate mechanical specifications.

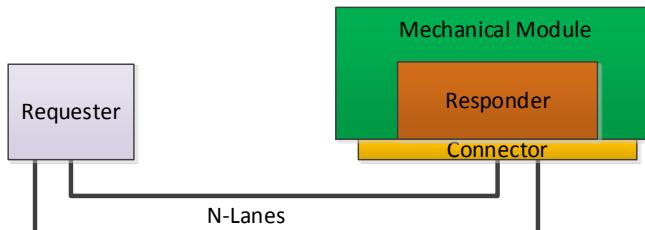


Figure 8-40: Example Mechanical Connectivity

The Component Mechanical structure is used to manage mechanical module operations, e.g., identification / location, power, dynamic insertion and removal, etc.

The following are the features and requirements associated with the Component Mechanical structure:

- The Component Mechanical structure shall be accessed through the *Interface Structure* of the component interface used to communicate with a component within a mechanical module.
  - If multiple interfaces are connected to the same mechanical module, then the Mechanical PTR field within each *Interface Structure* shall point to the same Component Mechanical structure.
  - The interface with the lowest numeric Interface ID with its link in L-Up should be used to control the mechanical module. If an interface with a lower numeric Interface ID subsequently transitions to I-Up, then management may transition to the new interface at its discretion. In-flight operations shall complete on the interface on which they were initiated unless physically precluded from doing so, e.g., the link transition to L-Down.
- The Component Mechanical structure may be supported by any component type.
- The Component Mechanical structure shall use the *Component Mechanical Structure Format*.
- Managed mechanical module insertion and removal functional elements:

- A Mechanical Module Controller (MMC) is an independent component within an enclosure used by software to:
  - Turn power on and off to each mechanical connector and inserted mechanical module.
  - Control individual Attention Indicators and the Power Indicators.
  - Engage and disengage individual electromechanical interlocks (if supported).
- An Attention Button is a momentary-contact push button switch physically located on or near the peer component's mechanical module. When pressed by a user, the MMC informs enclosure management that a user is about to dynamically insert or remove a mechanical module from the corresponding mechanical connector. Enclosure management should inform and seek permission from the impacted software entities before signaling the user to proceed.
  - If permission is granted and if a Power Indicator is supported and enabled, then the indicator will begin blinking, indicating the user may proceed. If permission is not granted or the operation fails, then the Power Indicator shall remain off and the management should log the event and any relevant reasons.
    - Once it begins blinking, if the user presses the Attention Button again within 5 seconds, the operation shall be aborted.
  - If a Power Indicator is unsupported or disabled, then dynamic insertion or removal could cause non-deterministic behavior and possibly damage hardware.
- A Power Controller is an independent set of one or more components used by software to control and monitor the main power, and if supported, the auxiliary power to a mechanical connector and to a mechanical module. The Power Controller is used to support dynamic insertion and removal of a mechanical module.
  - If a power controller is unsupported, then power shall be automatically managed based on the presence of the peer component, i.e., power off if no peer component is present and power on if a peer component is present.
  - If a power controller detects a main power fault, then it shall automatically shut off main power to the peer mechanical module; if supported, auxiliary power shall be unaffected. Main power shall not be restored until software disables and subsequently enables power through the Main Power Control Enable field.
  - If a power controller detects an auxiliary power fault, it shall shut off auxiliary power to the peer mechanical module and main power shall be unaffected. Auxiliary power shall not be restored until a mechanical module-specific method is invoked.
  - After main power is turned on, software shall not take any action until the associated link has transitioned to L-Up.
  - After main power is turned off, software shall wait at least 1 second before turning off the associated Power Indicator or turning main power back on.
- A Mechanical Retention Latch (MRL) is a retention mechanism that holds the mechanical module in the mechanical connectors and prevents the mechanical module's removal. A MRL may retain multiple mechanical modules if an enclosure does not support MRL Sensors.
- A MRL Sensor is any sensor type that detects and reports if the MRL is closed or open.
  - If the MRL Sensor detects an open MRL, then main power shall be automatically removed.

- If auxiliary power and any out-of-band signals or interfaces are interlocked with the MRL, then auxiliary power and these signals shall be automatically removed when the MRL is opened and restored when the MRL is closed.
- An Electromechanical Interlock is a mechanism for physically locking the mechanical module or MRL until software releases it.
  - The Electromechanical Interlock shall change state only under software control even if power is removed.
  - The Electromechanical Interlock shall engage or disengage and the Electromechanical Interlock Status shall be updated to reflect the new state within 200 ms of Electromechanical Interlock Control toggle.
  - Software shall wait at least 1 second between toggling Electromechanical Interlock.
- To compensate for variable MMC command execution times, Controller CMD Completion provides an asynchronous mechanism to inform software of when a MMC has completed executing a command.
  - The Controller CMD Completion may use an *Unsolicited Event (UE) Packet* to inform management. If an *Unsolicited Event (UE) Packet* is used, then software configures the Mechanical MGR CID and the Mechanical MGR SID (if applicable) prior to setting Controller CMD Completion Notification Enable.
  - The Controller CMD Completion may use a component-local interrupt to inform management. If a component-local interrupt is used, then software shall configure the Interrupt Address and Interrupt Data fields within the Component Mechanical structure prior to setting Controller CMD Completion Notification Enable.
- Mechanical modules and / or enclosures may support mechanical indicators, which are small optical devices, e.g., a LED.
  - Mechanical indicator-specific location and operation (e.g., on, off, and blink patterns) shall be as specified in the corresponding mechanical enclosure or mechanical form factor specification.
    - A component may specify additional blinking patterns, e.g., a block storage component might have a unique set of patterns per *SFF-8489 Specification for Serial GPIO IBPI (International Blinking Pattern Interpretation)*. These patterns shall be mapped to the Module Notification fields.
    - A mechanical enclosure or mechanical module may support additional module indicator notification patterns that are unique to the enclosure or mechanical module. These patterns may be associated with the mechanical module's Base Class or *Core Structure* C-UUID or may be managed through a *Vendor-Defined Structure* with a UUID located by the Mech-Vendor-Def PTR.
      - The UUID associated with *SFF 8489 Specification for Serial GPIO IBPI (International Blinking Pattern Interpretation)* shall be:  
b149c912c87d4435a052d4302eec3df9
  - Mechanical indicators are controlled through the Mechanical CAP 1 Control field.
  - A mechanical enclosure or mechanical module may support an Activity Indicator.
  - A mechanical enclosure or mechanical module may support an Attention Indicator.
    - The Attention Indicator shall not change state except when explicitly told to do so by software.
  - A mechanical enclosure or mechanical module may support a Power Indicator.

- The Power Indicator shall not change state except when explicitly told to do so by software.
- The Power Indicator shall reflect main power state. If auxiliary power is supported, then software shall determine if mechanical module insertion or removal is acceptable and may reflect that via the Power Indicator.
- If a Power Indicator is unsupported, then software shall provide other means to communicate to a user when a mechanical module may be inserted or removed.

**Developer Note:** *It is strongly recommended that enclosure designers consider the time required to ensure that the module's thermal level meets industry standards governing safe module handling. This could require delaying transitioning the corresponding power indicator to the off state until the enclosure's thermal management mechanisms have sufficiently cooled the module.*

- The Mechanical Signal Target field may be configured to generate one or two component-local interrupts, an *Unsolicited Event (UE) Packet*, or an *Unsolicited Event (UE) Packet* and a component-local interrupt(s) for each selected error or event.
  - If a component supports only *Out-of-band Management*, then software shall configure at least one component-local interrupt to inform management.
  - The Mechanical Event Detect field may be used to initiate management notification of mechanical events.
- Mechanical module presence detection shall be through the connector's Presence Detect pins.

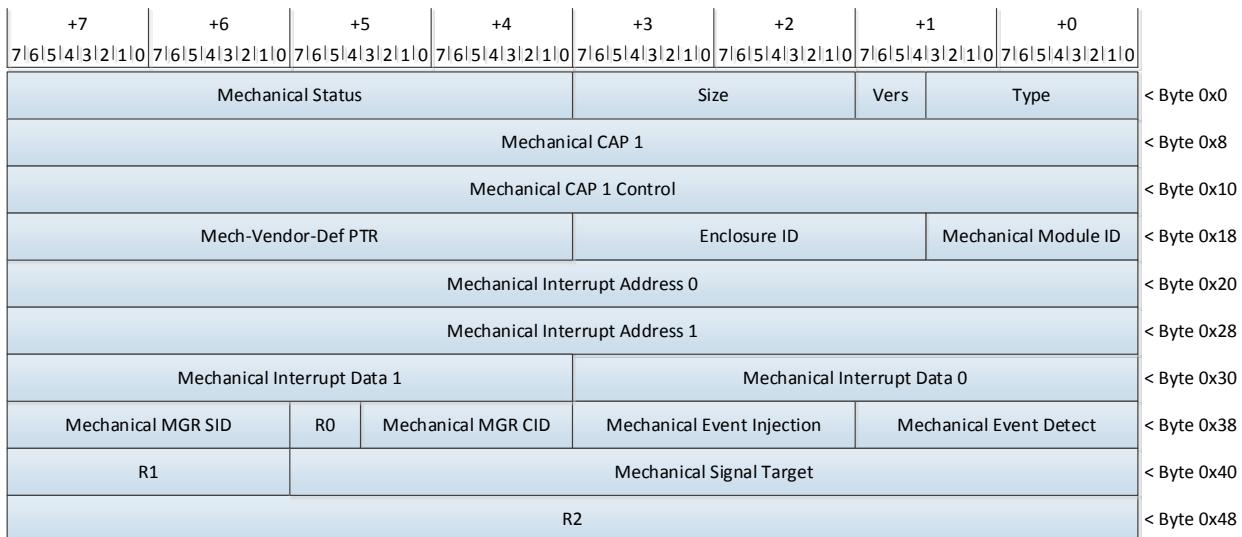


Figure 8-41: Component Mechanical Structure Format

Table 8-86: Component Mechanical Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Version (Vers)	4	0x1	M	RO	Structure Version
Mechanical Status	32	-	O		Current Mechanical status
		Bit 0		RW1C	Attention Button Pressed

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Mechanical CAP 1	64		O		0b—Not pressed 1b—Pressed
		Bit 1		RW1C	Power Fault Detected—Indicates if a main or auxiliary power fault has occurred. 0b—Not detected 1b—Detected
		Bit 2		RW1C	MRL Sensor State Change 0b—No change 1b—State change
		Bit 3		RO	MRL Sensor State 0b—MRL Closed 1b—MRL Open
		Bit 4		RW1C	Dynamic Insertion Removal Change—Indicates if a mechanical module has been dynamically inserted or removed. 0b—No change 1b—State change
		Bit 5		RO	Mechanical Component Presence—Indicates if a mechanical module has been inserted into the associated mechanical connector. Presence is detected using the mechanical connector's Presence Detect pins. If a mechanical module is inserted or removed, then Dynamic Insertion Removal Change shall be set to 1b. 0b—Not present 1b—Present
		Bit 6		RO	Electromechanical Interlock Status 0b—Disengaged 1b—Enabled
		Bit 7		RW1C	Controller CMD Completion—this bit is set to 1b when a controller has completed the outstanding controller command and can receive a new command. 0b—Completion Pending / No In-flight Command 1b—Completed
		Bits 31:8		-	RsvdZ
		Bit 0		HwInit	Mechanical Capabilities 1 Attention Button Support 0b—Unsupported

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					1b—Supported
		Bit 1		HwInit	Attention Indicator Support 0b—Unsupported 1b—Supported
		Bit 2		HwInit	Power Controller Support 0b—Unsupported 1b—Supported
		Bit 3		HwInit	Power Indicator Support 0b—Unsupported 1b—Supported
		Bit 4		HwInit	MRL Sensor Support 0b—Unsupported 1b—Supported
		Bit 5		HwInit	Electromechanical Interlock Support 0b—Unsupported 1b—Supported
		Bits 7:6		HwInit	Mechanical Insertion Removal Support—indicates if the component supports dynamic insertion or removal of a mechanical module 0x0—Unsupported 0x1—Managed Insertion and Removal 0x2—Asynchronous Insertion and Removal 0x3—Managed and Asynchronous Insertion and Removal
		Bit 8		HwInit	Common Reference Clock Support—Indicates if the mechanical connector supports common clock pins that can be used by an inserted module. 0b—Unsupported 1b—Supported
		Bit 9		HwInit	Controller CMD Completion Notification Support 0b—Unsupported 1b—Supported
		Bits 19:10		HwInit	Max_Mech_Power_LVL is used to calculate the maximum power a mechanical module supports. The corresponding mechanical module specification is responsible for specifying how power is physically provided. Max_Mech_Power = (Max_Mech_Power_LVL * Mech_Power_Scale) Watts.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
		Bits 22:20		HwInit	<p>Mech_Power_Scale</p> <ul style="list-style-type: none"> <li>• 0x0—1.0</li> <li>• 0x1—0.1</li> <li>• 0x2—0.01</li> <li>• 0x3—0.001</li> <li>• 0x4—0.0001</li> <li>• 0x5—0.00001</li> <li>• 0x6—0.000001</li> <li>• 0x7—0.0000001</li> </ul>
		Bits 30:23		HwInit	<p>Main Power Voltage Support—if Bit<sub>K</sub> == 1b, then the corresponding voltage is supported by the mechanical module.</p> <p>Bit 0: 12V (Non-high-power pins)          Bit 1: 12V (High-power pins)          Bit 2: 48V (High-power pins)          Bits 7:3 Reserved</p>
		Bit 31		HwInit	<p>Module Indicator Present Notification Support</p> <p>0b—Unsupported          1b—Supported</p>
		Bit 32		HwInit	<p>Module Indicator Locate Notification Support</p> <p>0b—Unsupported          1b—Supported</p>
		Bit 33		HwInit	<p>Module Indicator Failure Notification Support</p> <p>0b—Unsupported          1b—Supported</p>
		Bit 34		HwInit	<p>Module Indicator Notification 1 Support</p> <p>0b—Unsupported          1b—Supported</p>
		Bit 35		HwInit	<p>Module Indicator Notification 2 Support</p> <p>0b—Unsupported          1b—Supported</p>
		Bit 36		HwInit	<p>Module Indicator Notification 3 Support</p> <p>0b—Unsupported          1b—Supported</p>
		Bit 37		HwInit	<p>Module Indicator Notification 4 Support</p> <p>0b—Unsupported          1b—Supported</p>
		Bit 38		HwInit	<p>Module Indicator Notification 5 Support</p> <p>0b—Unsupported</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					1b—Supported
		Bit 39		HwInit	Module Indicator Notification 6 Support 0b—Unsupported 1b—Supported
		Bit 40		HwInit	Module Indicator Notification 7 Support 0b—Unsupported 1b—Supported
		Bit 41		HwInit	Module Indicator Vendor-defined 1 Support 0b—Unsupported 1b—Supported
		Bit 42		HwInit	Module Indicator Vendor-defined 2 Support 0b—Unsupported 1b—Supported
		Bits 44:43		HwInit	Module Indicator Interpretation—Indicates what to use as a basis to interpret Module Indicator Notification [1-7]. 0x0—Component’s Base Class 0x1—Component’s C-UUID 0x2—Vendor-defined (see Mech-Vendor-Def-PTR) 0x3—Reserved If Module Indicator Notification [1-7] are unsupported, then this field shall be Reserved.
		Bits 47:45		RO	Management Notification Support—Indicates the supported management notification mechanisms. 0x0—Unsupported 0x1—Component-local Interrupt 0x2— <i>Unsolicited Event (UE) Packet</i> using Mechanical MGR CID field 0x3— <i>Unsolicited Event (UE) Packet</i> using Mechanical MGR CID and Mechanical MGR SID fields 0x4—Component-local Interrupt and <i>Unsolicited Event (UE) Packet</i> using Mechanical MGR CID field 0x5—Component-local Interrupt and <i>Unsolicited Event (UE) Packet</i> using Mechanical MGR CID field and Mechanical MGR SID fields 0x6-0x7—Reserved

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Mechanical CAP 1 Control	64	Bits 63:48	O	RW	Reserved
		-			Mechanical Capabilities 1 Control
		Bit 0			Attention Button Enabled—When enabled, inform management when the attention button is pressed using the configured Management Notification Method. 0b—Disabled (default) 1b—Enabled
		Bits 2:1			Attention Indicator Control—If a mechanical enclosure or mechanical module supports an Attention Indicator, then this field drives its operation. If supported, then the default Attention Indicator Control shall be a non-zero value. If unsupported, then this field shall be hardwired to 0x0. 0x0—Reserved 0x1—On 0x2—Blink 0x3—Off
		Bit 3			Main Power Controller Enable 0b—Enable main power to the component's mechanical module (default) 1b—Disable main power to the component's mechanical module
		Bit 4			Power Fault Enabled—When enabled, inform management when a power fault is detected using the configured Management Notification Method. 0b—Disabled 1b—Enabled
		Bits 5			Power Indicator Control—If a mechanical enclosure or mechanical module supports a Power Indicator, then this field drives its operation. If supported, then the default Power Indicator Control shall be a non-zero value. If unsupported, then this field may be Read-Only and hardwired to 0x0. 0x0—Reserved 0x1—On

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					0x2—Blink 0x3—Off
		Bit 6			MRL Sensor Enabled—When enabled, inform management when the MRL Sensor changes state using the configured Management Notification Method.  0b—Disabled 1b—Enabled
		Bit 7			Electromechanical Interlock Control  1b—Toggle the interlock state  Reads of this bit shall return 0b.
		Bit 8			Dynamic Insertion Removal Enabled—When enabled, inform management when a dynamic mechanical component insertion or removal event is detected using the configured Management Notification Method.  0b—Disabled 1b—Enabled
		Bit 9			Controller CMD Completion Notification Enabled—When enabled, inform management when a mechanical controller command has completed using the configured Management Notification Method.  0b—Disabled / Unsupported 1b—Enabled
		Bits 14:10			Activity Indicator Control—If a mechanical enclosure or mechanical module supports an Activity Indicator, then this field drives its operation.  0x0—Module Present 0x1—Locate 0x2—Module Failure 0x3—Module Indicator Notification 1 0x4—Module Indicator Notification 2 0x5—Module Indicator Notification 3 0x6—Module Indicator Notification 4 0x7—Module Indicator Notification 5 0x8—Module Indicator Notification 6 0x9—Module Indicator Notification 7 0xA—Vendor-defined Notification 1 0xB—Vendor-defined Notification 2

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					0xC-0x1F—Reserved
		Bit 16:15			<p>Management Notification Method—Used to indicate the method to notify management when an event occurs.</p> <p>0x0—Not Configured</p> <p>0x1—Component-local Interrupt. Mechanical Interrupt Address and Mechanical Interrupt Data fields shall be valid prior to setting this value.</p> <p>0x2—UE Packet—Mechanical MGR CID and Mechanical MGR SID (if applicable) fields shall be valid prior to setting this value.</p> <p>0x3—Generate a component-local interrupt and a UE packet. Mechanical Interrupt Address, Mechanical Interrupt Data, Mechanical MGR CID, and Mechanical MGR SID (if applicable) shall be valid prior to setting this value.</p>
		Bit 18:17			<p>Mechanical MGR CID-SID Configured—Used to indicate if the Mechanical MGR CID and Mechanical MGR SID fields are configured. If these fields are not configured and Management Notification Method equals UE Packet, then see <i>Unsolicited Event (UE) Packet</i> to locate the destination component.</p> <p>0x0—Not configured</p> <p>0x1—Mechanical MGR CID Configured; Mechanical MGR SID is not configured</p> <p>0x2—Mechanical MGR CID and Mechanical MGR SID Configured</p> <p>0x3—Reserved</p>
		Bits 21:19			<p>Main Power Voltage Enable—Determines the main power voltage enabled.</p> <p>0x0—12V (Non-high-power pins)</p> <p>0x1—12V (High-power pins)</p> <p>0x2—48V (High-power pins)</p> <p>0x3-0x7—Reserved</p>
		Bits 63:22			RsvdP

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>Mechanical Module ID</b>	12	-	O	RW	Management may configure a mechanical module location identifier that is associated with the peer component attached through this interface.
<b>Enclosure ID</b>	20	-	O	RW	Management may configure an enclosure identifier to assist with locating a mechanical module among a set of mechanical enclosures.
<b>Mechanical Interrupt Address 0</b>	64	-	O	RW	The component-local address to target component-local interrupt 0.
<b>Mechanical Interrupt Address 1</b>	64	-	O	RW	The component-local address to target component-local interrupt 1
<b>Mechanical Interrupt Data 0</b>	32	-	O	RW	The interrupt data associated with component-local interrupt 0 Contents of this field are component-specific
<b>Mechanical Interrupt Data 1</b>	32	-	O	RW	The interrupt data associated with component-local interrupt 1 Contents of this field are component-specific
<b>Mech-Vendor-Def PTR</b>	32	-	O	RO	If non-Null, points to a vendor-defined structure.
<b>Mechanical MGR CID</b>	12	-	O	RW	If an <i>Unsolicited Event (UE) Packet</i> is used to notify management, then this field shall contain the CID of the component where the mechanical manager is located. The <i>Component Error and Signal Event Structure</i> MECH MGR UEP Mask indicates the [VC, egress interface] that can be used to transmit the <i>Unsolicited Event (UE) Packet</i> .
<b>Mechanical MGR SID</b>	16	-	O	RW	If an <i>Unsolicited Event (UE) Packet</i> is used to notify the management, then this field shall contain the SID (if applicable) of the component where the mechanical manager is located.
<b>Mechanical Event Detect</b>	16	-	O	RW	Controls which mechanical events to signal to the management. See <i>Mechanical Event Detect</i>
<b>Mechanical Event Injection</b>	16	-	O	RW	Controls which mechanical events to inject management. See <i>Mechanical Event Injection</i>
<b>Mechanical Signal Target</b>	48	-	O	RW	Each Signal Target 3-bit sub-field indicates how the associated error is to be signaled. If configured,

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
R0					<p>then the component shall generate one or both component-local interrupts, an <i>Unsolicited Event (UE) Packet</i>, or both component local interrupt(s) and an <i>Unsolicited Event (UE) Packet</i>.</p> <p>0x0—No signal is generated  0x1—Component-local interrupt 0  0x2—Component-local interrupt 1  0x3—Component-local interrupts 0 and 1  0x4—<i>Unsolicited Event (UE) Packet</i>  0x5—Component-local interrupt 0 and an <i>Unsolicited Event (UE) Packet</i>  0x6—Component-local interrupt 1 and an <i>Unsolicited Event (UE) Packet</i>  0x7—Component-local interrupts 0 and 1, and an <i>Unsolicited Event (UE) Packet</i></p> <p>Sub-field 0 is located in byte 0, bit 0. Sub-field 1 is contiguous to sub-field 0. And so forth.</p>
R0	4	-	-	-	RsvdP
R1	16	-	-	-	RsvdP
R2	64	-	-	-	Reserved

Table 8-87: Mechanical Event Detect

Bit Location	Default	Access	Description
0	0b	RW	Dynamic Module Insertion Detect—Presence is indicated by the mechanical connector's Presence Detect pins
	0b	RW	Dynamic Module Removal Detect—Removal is indicated by the mechanical connector's Presence Detect pins
	0b	RW	Attention Button Pressed Detect
	0b	RW	Attention Indicator On Continuously Detect
	0b	RW	Module Power Up Detect
	0b	RW	Module Power Off Detect
	0b	RW	MRL Open Detect
	0b	RW	MRL Close Detect
	0b	RW	Module Indicator Activated Detect
	0b	-	RsvdP

The Mechanical Event Injection field shall be as specified in *Mechanical Event Injection*. If Mechanical Event Injection Bit<sub>K</sub> == 1b and the corresponding Mechanical Event Detect Bit<sub>K</sub> == 1b, then the component shall initiate the corresponding event processing as though the event was detected. Unsupported Mechanical Event Injection bits shall be hardwired to 0b. Reads of this field shall return 0x0.

5

Table 8-88: Mechanical Event Injection

Bit Location	Default	Access	Description
0	0b	-	Inject Dynamic Module Insertion
	0b	RW	Inject Dynamic Module Removal
	0b	RW	Inject Attention Button Pressed
	0b	RW	Inject Attention Indicator On Continuously
	0b	RW	Inject Module Power Up
	0b	RW	Inject Module Power Off
	0b	RW	Inject MRL Open
	0b	RW	Inject MRL Close
	0b	RW	Inject Module Indicator Activated
	-	-	RsvdP

## 8.29. Component Destination Table Structure

The Component Destination Table structure serves the following purposes:

- In a multi-interface component, to select an egress interface to transmit a packet to a peer.
- To select the VC to use in the packet's VC field (single or multi-interface components).

10 The Component Destination Table primarily consists of a set of pointers to tables used to fulfill this structure's purposes:

- The SSDT (Single-Subnet Destination Table) is used by Requesters that support single-subnet communications. It contains a set of route table entries used to select an egress interface to reach a Responder.
- The MSDT (Multi-subnet Destination Table) is used by Requesters that support multi-subnet communications. It contains a set of route table entries used to select an egress interface to reach a destination subnet.
- The REQ-VCAT (VC Action Table) is used by Requesters to select the VC used in request packets.
- The RSP-VCAT (VC Action Table) is used by Responders to select the VC used in response packets.
- The RIT (Responder Interface Table) is used by multi-interface Responders to select egress interface to transmit a response packet.

20 The following are the features and requirements associated with the Component Destination Table Structure:

- Requesters and Responders that support explicit OpClass operations shall support the Component Destination Table structure.
  - Requester and Responder interfaces that do not support explicit OpClasses shall not be configured through this structure or associated tables.
  - If a component contains an integrated switch, then the Component Destination Table structure is used to select the egress interface and VC for component-generated packets generated. Switch packet relay tables (see *Interface Structure*) shall not be used.
  - If the component is a discrete switch and supports *In-band Management* or contains Data Space, then the Component Destination Table structure shall be used for switch-generated packets.
- The Component Destination Table structure shall use the *Component Destination Table Structure Format*.
- With the exception of an *Unsolicited Event (UE) Packet*, a Requester shall use the Component Destination Table to determine the egress interface and VC used to transmit an explicit OpClass request packet.
- With the exception of response packets to Control OpClass request packets with DR == 1b, a Responder shall use the Component Destination Table to determine the egress interface and VC used to transmit an explicit OpClass response packet.
  - A response packet to a Control OpClass request packet with DR == 1b shall be transmitted on VC 0, using the same interface on which the request packet was received, i.e., a Responder shall not use the Component Destination Table.
- If a Control Write request packet is used to update a table entry within the SSDT, MSDT, REQ-VCAT, RSP-VCA, or RIT tables, then:
  - Management shall update only one table entry / row per Control Write request packet.
  - The Control Write request packet's payload shall be an integer 4-byte multiple.
- If a Requester or a Responder contains an integrated switch, then:
  - The Component Destination Table structure shall be used to generate packets and to determine the egress interface to transmit packets.
  - The Component Destination Table structure shall not apply to packet relay between the external component interfaces.
- Within a directly-attached subnet, Requesters or Responders that support multiple CIDs may use any configured CID in the packet's SCID field when communicating with a peer component. Policies governing which CID to use are outside of this specification's scope.
- Within a multi-subnet topology, Requesters or Responders that support multiple GCIDs may use any GCID [CID, SID] in the packet's SCID and SSID fields when communicating with a peer component that supports multi-subnet communications. Policies governing which GCID to use are outside of this specification's scope.
- If a Requester supports single-subnet communications, then it shall provision a SSDT. A SSDT shall use the *SSDT / MSDT Format*. Each row shall correspond to a destination component within a directly-attached subnet. The SSDT shall be linearly indexed using the Responder's CID.

**Developer Note:** If a Requester communicates with Responders outside of a single enclosure, then Requesters are strongly encouraged to support the maximum SSDT size.

- If a Requester supports multi-subnet communications, then it shall provision a MSDT. A MSDT shall use the *SSDT / MSDT Format*. Each row shall correspond to a destination subnet. The MSDT shall be linearly indexed using the Responder's SID.

**Developer Note:** The SSDT/MSDT use the same basic structures, fields, and selection algorithms as used in a switch LPRT/MPRT. This enables nearly all of the underlying design and implementation to be reused for the two functional blocks.

	Route N	Route 1	Route 0	
DCID 0   DSID 0 >	EI, VCA, HC, V	...	EI, VCA, HC, V	EI, VCA, HC, V   MHC
DCID 1   DSID 1 >	EI, VCA, HC, V	...	EI, VCA, HC, V	EI, VCA, HC, V   MHC
		...		
DCID K   DSID L >	EI, VCA, HC, V	...	EI, VCA, HC, V	EI, VCA, HC, V   MHC

Figure 8-42: SSDT / MSDT Format

- Each SSDT / MSDT row shall contain Max Routes route entries.
  - Each SSDT / MSDT row shall use the *SSDT / MSDT Row Format*.
  - The Minimum Hop Count (MHC) field is used to calculate the Computed Hop Count (CHC).
  - For each route entry, the Valid Entry Indicator (V) bit determines if the route entry is valid, i.e., the destination component is reachable using this route.
  - The Hop Count (HC) field indicates the number of hops to the destination component or destination subnet from the indicated egress interface.
  - VCA is used to select a column within the REQ-VCAT corresponding to this route entry.
  - EI contains the egress interface identifier corresponding to this route entry.

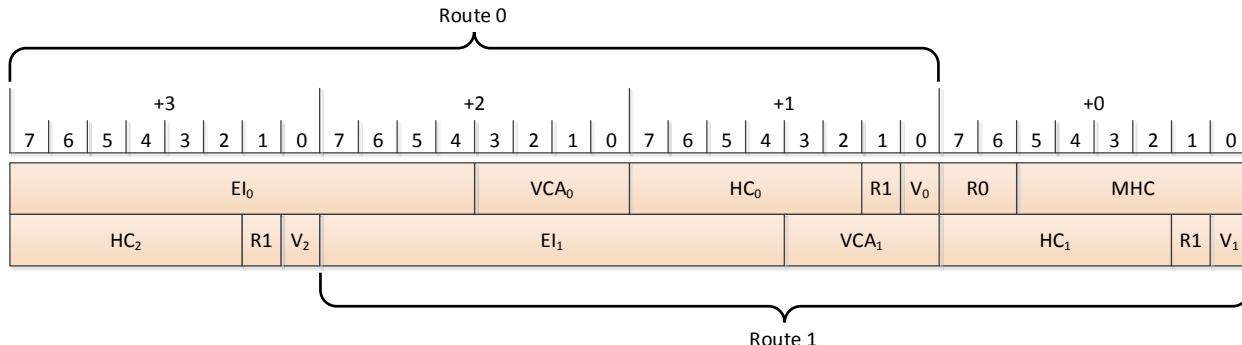


Figure 8-43: SSDT / MSDT Row Format

Table 8-89: SSDT / MSDT Row Fields

Field Name	Size (bits)	Access	Description
MHC	6	RW	Minimum hop count This field shall be Reserved if hop count configuration is unsupported.
V	1	RW	Valid Entry Indicator 0b—Invalid Route Entry (not configured) 1b—Valid Route Entry (configured)
HC	6	RW	If a SSDT, then this field contains the number of switch crossings to destination component

Field Name	Size (bits)	Access	Description
			<p>If a MSDT, then this field contains the number of switch crossings to a destination subnet</p> <p>This field shall be Reserved if hop count configuration is unsupported.</p> <p>Given the variety of topologies and dynamic operating conditions, management may configure this field with the actual number of switch hops or a relative number using policies outside of this specification's scope. If management uses a relative number, larger relative hop counts shall represent larger physical hop counts, e.g., relative hop count 1 represents more physical hops than hop count 0. This is necessary since the component cannot distinguish between actual and relative hop counts.</p>
VCA	4	RW	VCA is used to select the action column within the REQ-VCAT.
EI	12	RW	Egress Interface Identifier
R0	2	-	RsvdP
R1	1	-	RsvdP

- A Requester shall provision a Requester VC Action Table (REQ-VCAT). To better understand how this table is used, see *Single-Subnet Single-Route Egress Selection* and *Multi-subnet Requester Egress Selection*.
  - The REQ-VCAT and VCAT Entry format shall use the *REQ-VCAT and VCAT Entry Format*.
  - A REQ-VCAT shall contain one or more columns and each column shall contain one or more rows.
    - Each column represents a set of actions each containing a VC Mask (VCM) and a hop-count comparison threshold (if supported). A VCM is bit mask where Bit<sub>K</sub> == 1b indicates a VC that can be used in a request packet's VC field, and Bit<sub>K</sub> == 0b indicates a VC that cannot be used.
      1. If VC<sub>K</sub> is unsupported, then Bit<sub>K</sub> shall be hardwired to 0b.
    - The number of provisioned rows shall be equal to the maximum number of VCs supported on any Requester interface.

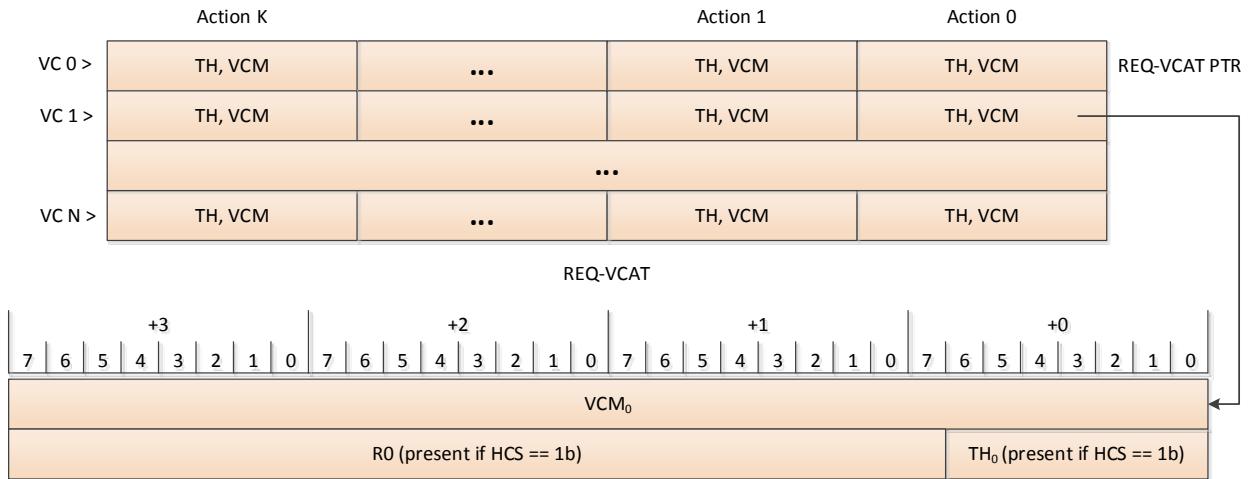


Figure 8-44: REQ-VCAT and VCAT Entry Format

Table 8-90: REQ-VCAT Fields

Field Name	Size (bits)	M / O	Access	Description
<b>VCM</b>	32	M	RW	VC Mask—Bits corresponding to unsupported VCs shall be hardwired to 0b.
<b>TH</b>	7	M	RW	Hop-count threshold comparison value. The configured threshold value shall be configured such that $TH = (HC + 1)$ . This field shall be present only if <i>Route Control Field RC CAP 1 HCS == 1b</i> .
<b>RO</b>	25	-	-	Reserved This field shall be present only if <i>Route Control Field RC CAP 1 HCS == 1b</i> .

- A Responder shall provision a Responder VC Action Table (RSP-VCAT). To understand how this table is used, see *Responder Egress and VC Selection*.
  - The RSP-VCAT Entry shall use the *RSP-VCAT Format*.
  - A RSP-VCAT shall contain one or more columns and each column shall contain one or more rows.
    - Each column represents a set of actions each containing a VC Mask (VCM). A VCM is bit mask where  $Bit_k == 1b$  indicates a VC that can be used in a response packet's VC field.
    - The number of provisioned rows shall be equal to the maximum number of VCs supported on any Responder interface.



Figure 8-45: RSP-VCAT Format

Table 8-91: RSP-VCAT Fields

Field Name	Size (bits)	M / O	Access	Description
VCM	32	M	RW	VC Mask—Bits corresponding to unsupported VCs shall be hardwired to 0b.

If a Responder supports multiple egress interfaces, then it shall provision a RIT. The RIT shall use the *RIT Format*. It contains a set of bit masks where each  $Bit_k == 1b$  indicates an egress interface that can be used to transmit a response packet.

5. • If the EIM size is not an integer 4-byte multiple, then it shall be padded by RIT Pad Size bits.



Figure 8-46: RIT Format

Table 8-92: RIT Fields

Field Name	Size (bits)	M / O	Access	Description
EIM	-	M	RW	Egress Interface Mask—a bit mask representing the allowable egress interfaces for the response packet. A Responder may choose from any valid egress interface / VC pair from the RSP-VCAT VCM and EIM. Each EIM contains <i>Core Structure Max Interface</i> bits. If interface $K$ does not support response packet transmission, then the corresponding $Bit_k$ shall be hardwired to 0b.

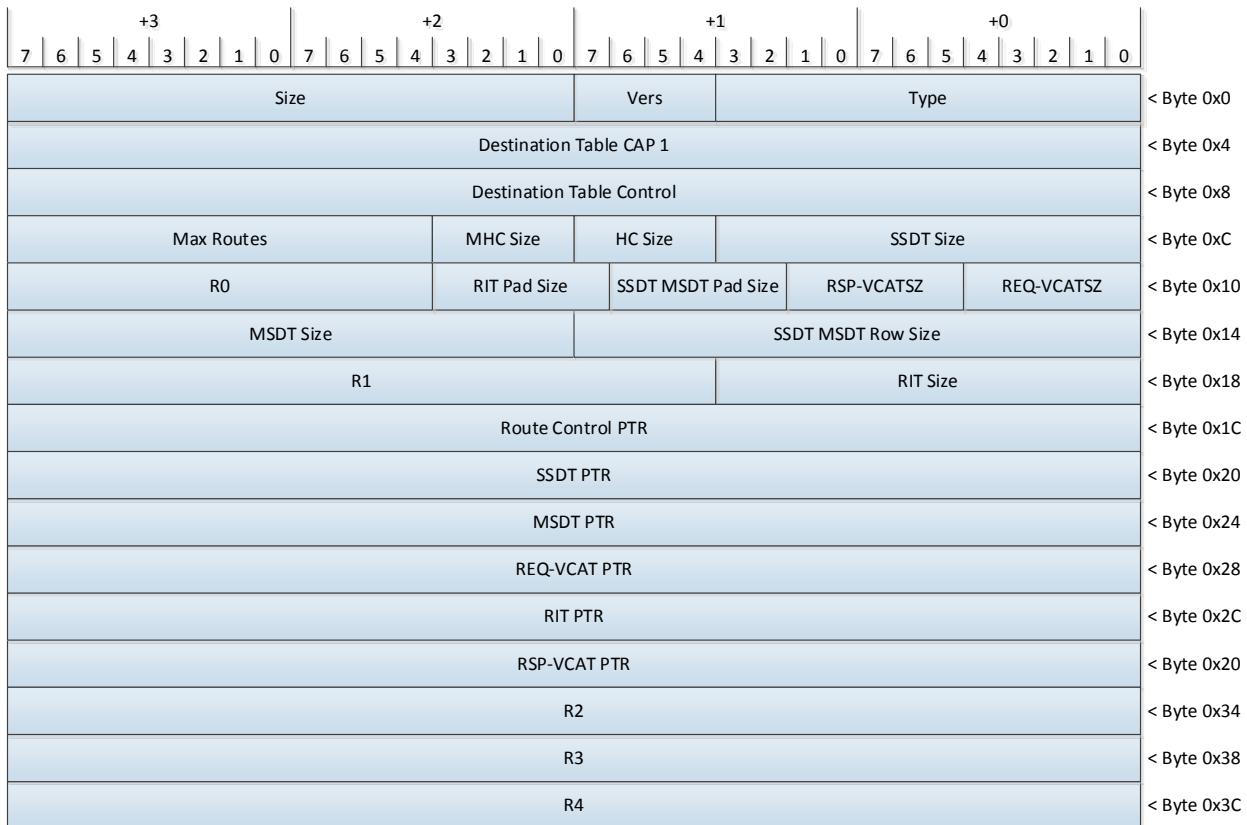


Figure 8-47: Component Destination Table Structure Format

Table 8-93: Component Destination Table Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>Version (Vers)</b>	4	0x1	M	RO	Structure Version
<b>Destination Table CAP 1</b>	32	-	M	RO	Destination Table Capabilities 1
		Bit 0			HCS—Indicates if the component supports minimum hop counts and threshold comparison. 0b—Unsupported 1b—Supported
		Bit 1			EI Support—Indicates if the SSDT / MSDT entries support an EI field. If unsupported, then the EI field shall not be present within a SSDT or MSDT table entry. 0b—Unsupported 1b—Supported
					<b>Developer Note:</b> All Requesters are strongly encouraged to support the EI field. However, for select component types (e.g., an embedded

Destination Table Control		Bit 2		<p><i>component such as a co-packaged memory media controller), component-specific egress interface selection can be used.</i></p>	
				<p>Wildcard SSDT Support—Indicates if the component supports a single SSDT table entry that is used to reach all local subnet components</p> <p>0b—Unsupported 1b—Supported</p>	
				<p>Wildcard MSDT Support—Indicates if the component supports a single MSDT table entry that is used to reach all multi-subnet components</p> <p>0b—Unsupported 1b—Supported</p>	
				<p>Reserved</p>	
	32	-	M	RW	This field controls the component's operation relative to this structure and sub-structures.
					<p>Peer Authorization Enable—Until enabled, a Responder shall not perform “Authorized Peer Component?” validation as described in MP and Security Error Detection.</p> <p>This enables management to configure the component including adding the management component to the SSDT and if applicable, the MSDT.</p> <p>0b—Disabled 1b—Enabled</p>
					<p>Threshold Enabled—if supported, enable the threshold comparison using the thresholds configured within the VCAT. If unsupported, then this field shall be hardwired to 0b.</p> <p>0b—Disabled 1b—Enabled</p>
					RsvdP
SSDT Size	12	-	O	RO	<p>Indicates the number of Single-subnet Destination Table entries</p> <p>If SSDT Size == 0x0, then the number of entries shall be <math>2^{12}</math>.</p> <p>If SSDT Size == 0x1 and Wildcard SSDT Support == 1b, then a single SSDT table entry shall be provisioned, and this entry shall be used to reach all local subnet components.</p>

HC Size	4	-	O	RO	<p>Number of bits supported by the underlying component implementation for the HC and TH table entry fields.</p> <p>If zero, then the MHC, HC, and TH fields shall be Reserved within the corresponding table entries.</p> <p>HC Size shall be <math>\leq 7</math>.</p>
MHC Size	4	-	O	RO	<p>Number of bits supported by the underlying component implementation for the MHC table entry field.</p> <p>If zero, then the MHC shall be Reserved within the corresponding table entries.</p> <p>MHC Size shall be <math>\leq 7</math>.</p>
Max Routes	12	-	O	RO	<p>Indicates the maximum number of route table entries (columns) per route entry row within a SSDT or MSDT.</p> <p>If Max Routes == 0x0, then the number of routes shall be <math>2^{12}</math>.</p>
REQ -VCATSZ	5	-	O	RO	<p>Indicates the number of VC actions (columns) per row within the REQ-VCAT.</p> <p>If REQ-VCATSZ == 0x0, then the number of VC actions shall be <math>2^5</math>.</p>
RSP-VCATSZ	5	-	O	RO	<p>Indicates the number of VC actions (columns) per row within the RSP-VCAT.</p> <p>If RSP-VCATSZ == 0x0, then the number of VC actions shall be <math>2^5</math>.</p>
SSDT MSDT Pad Size	5	-	O	RO	<p>Indicates the number of pad bits provisioned per SSDT / MSDT row entry to maintain integer 4-byte alignment.</p> <p>Pad bits shall be Reserved.</p>
RIT Pad Size	5	-	O	RO	<p>Indicates the number of pad bits provisioned per RIT row entry to maintain integer 4-byte alignment.</p> <p>Pad bits shall be Reserved.</p>
SSDT MSDT Row Size	16	-	O	RO	<p>The size of a SSDT or MSDT row in bytes. The size shall be an integer 4-byte multiple.</p> <p>The SSDT MSDT Pad-Size indicates the number of pad bits appended to the row to ensure integer 4-byte alignment.</p>

<b>MSDT Size</b>	16	-	O	RO	Indicates the number of Multi-subnet Destination Table entries  If MSDT Size == 0x0, then the number of entries shall be $2^{16}$ .  If MSDT == 0x1 and Wildcard MSDT Support == 1b, then a single MSDT table entry shall be provisioned, and this entry shall be used to reach all multi-subnet components.
	12	-	O	RO	The number of entries within the RIT  If RIT Size == 0x0, then the number of entries shall be $2^{12}$ .
<b>Route Control PTR</b>	32	-	M	RO	Pointer to the <i>Route Control Field</i> .  This field shall be shared by the <i>Component Destination Table Structure</i> and the <i>Component Switch Structure</i> (if supported), and provides component-wide route control.
	32	-	O	RO	If a Requester, then points to the provisioned SSDT, else this field shall be Null to indicate that the component does not support a SSDT.  If a SSDT is not supported, then unless explicitly stated otherwise by this specification, all SSDT-specific fields shall be Reserved.
<b>MSDT PTR</b>	32	-	O	RO	If a Requester that supports multi-subnet communications, then this shall point to the provisioned MSDT, else this field shall be Null to indicate that the component does not support a MSDT.  If a MSDT is not supported, then unless explicitly stated otherwise by this specification, all MSDT-specific fields shall be Reserved.
	32	-	O	RO	If a Requester, then this shall point to the provisioned REQ-VCAT, else this field shall be Null to indicate that the component does not support a REQ-VCAT.  If a REQ-VCAT is not supported, then unless explicitly stated otherwise by this specification, all REQ-VCAT-specific fields shall be Reserved.
<b>RIT PTR</b>	32	-	O	RO	If a Responder, then this points to the provisioned RIT, else this field shall be Null to indicate that the component does not support a RIT.

RSP-VCAT PTR					If a RIT is not supported, then unless explicitly stated otherwise by this specification, all RIT-specific fields shall be Reserved.
	32	-	O	RO	If a Responder, then this points to the provisioned RSP-VCAT, else this field shall be Null to indicate that the component does not support a RSP-VCAT.  If a RSP-VCAT is not supported, then unless explicitly stated otherwise by this specification, all RSP-VCAT-specific fields shall be Reserved.
R0	12	-	-	-	RsvdP
R1	20	-	-	-	RsvdP
R2	32	-	-	-	Reserved
R3	32	-	-	-	Reserved
R4	32	-	-	-	Reserved

### 8.29.1. Single-Subnet Single-Route Egress Selection

*Single-subnet Single-Route Requester Egress Selection* illustrates the structures and conceptual steps taken by a Requester that supports a single route to each destination to select an egress interface. The following assumes MSS = 1b and HCS = 1b.

- 5 1. The Requester accesses the SSDT, and uses the Responder's CID to directly index and locate the corresponding SSDT route entry row.
2. Since the Requester supports a single route entry (Max Routes = 0), the following applies:
  - a. VCA field is ignored.
  - b. If V == 1b, then the packet is relayed to the EI (egress interface), else it is handled as a component-local error.
- 10 3. The Requester uses the Traffic Class corresponding to the Responder's addressable resource to index the REQ-VCAT and selects a VC Mask entry. If a Requester supports a Requester ZMMU, then the Traffic Class from the corresponding *Requester PTE* entry is used; if not, then the Requester uses a component-specific method to derive the Traffic Class.
- 15 4. The Requester selects a valid VC (VCM[Bit<sub>K</sub> == 1b]), and copies this to the request packet's VC field.

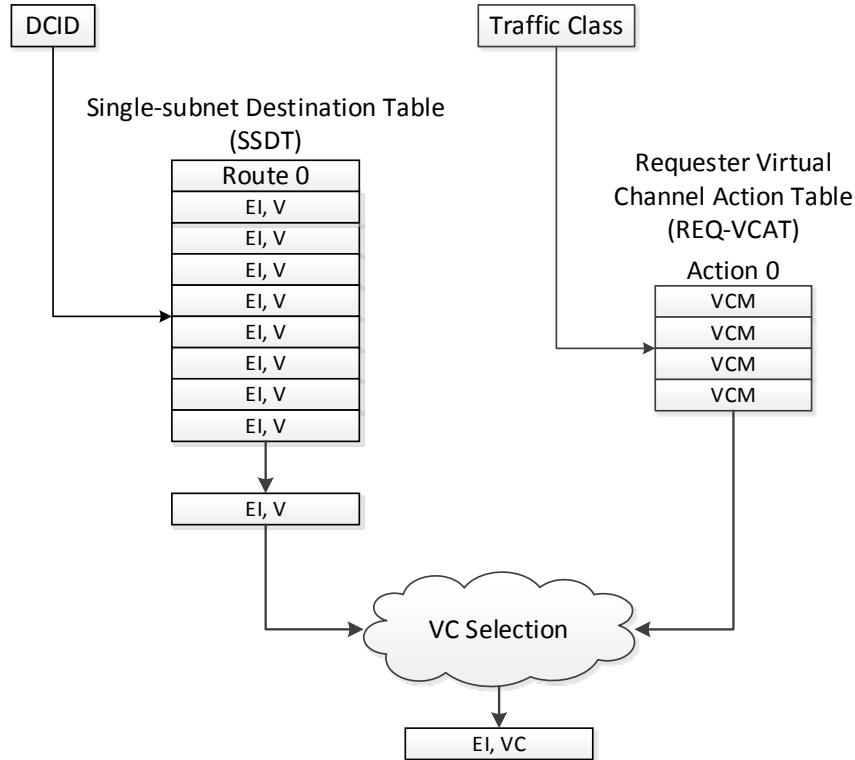


Figure 8-48: Single-subnet Single-Route Requester Egress Selection

### 8.29.2. Single-Subnet Multi-Route Egress Selection

*Single-subnet Multi-Route Requester Egress Selection* illustrates the structures and conceptual steps used by a Requester that supports multiple routes to each destination to select an egress interface. The following assumes MSS = 0b and HCS = 0b.

1. The Requester accesses the SSDT, and uses the Responder's CID to directly index and locate the corresponding SSDT route entry row.
2. The Requester uses the Traffic Class corresponding to the Responder's addressable resource to index the REQ-VCAT and selects a row. If a Requester supports a Requester ZMMU, then the Traffic Class from the corresponding *Requester PTE* entry is used; if not, then the Requester uses a component-specific method to manage derive the Traffic Class.
3. If a Requester uses random or adaptive route selection.
4. The Requester selects an egress interface and a valid VC (VCM[Bit<sub>K</sub> == 1b]), and copies this to the request packet's VC field.

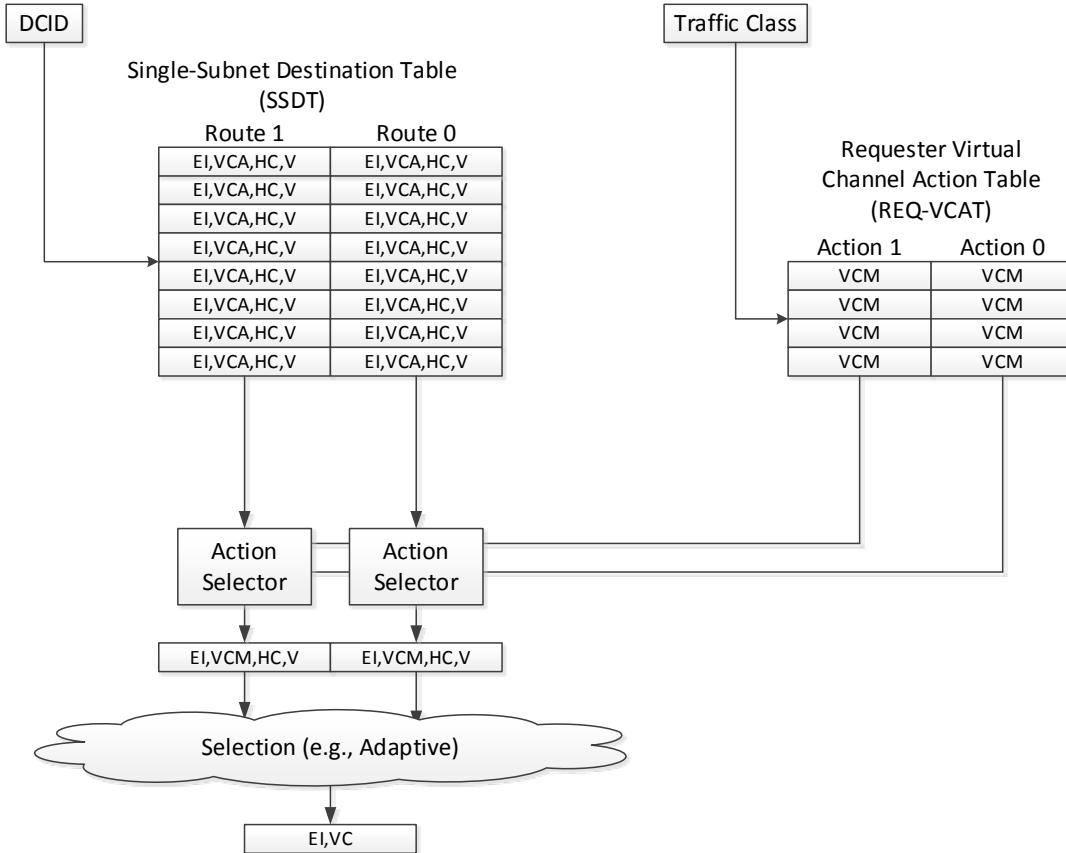


Figure 8-49: Single-subnet Multi-Route Requester Egress Selection

### 8.29.3. Responder Egress and VC Selection

*Responder Egress Selection* illustrates the structures and conceptual steps taken to select an egress interface:

1. The Responder uses the request packet's VC field to index the RSP-VCAT to locate the VC Mask. The VC Mask indicates the set of VCs that can be used in the corresponding response packet's VC field.
2. If the Responder supports a single egress interface, then it selects a VC ( $VCM[Bit_k == 1b]$ ) to use in the corresponding response packet's VC field, and schedules response packet transmission.
3. If the Responder supports multiple egress interfaces, then the Responder uses the ingress interface identifier on which the request packet arrived to index the RIT to locate the Egress Interface Mask. The Egress Interface Mask indicates the set of egress interfaces ( $EIM[Bit_k == 1b]$ ) that may be used to transmit the corresponding response packet.
4. If multiple VC and / or egress interfaces can be used, then the Responder applies a component-specific selection process, e.g., adaptive selection based on the least-congested egress interface.

5

10

15

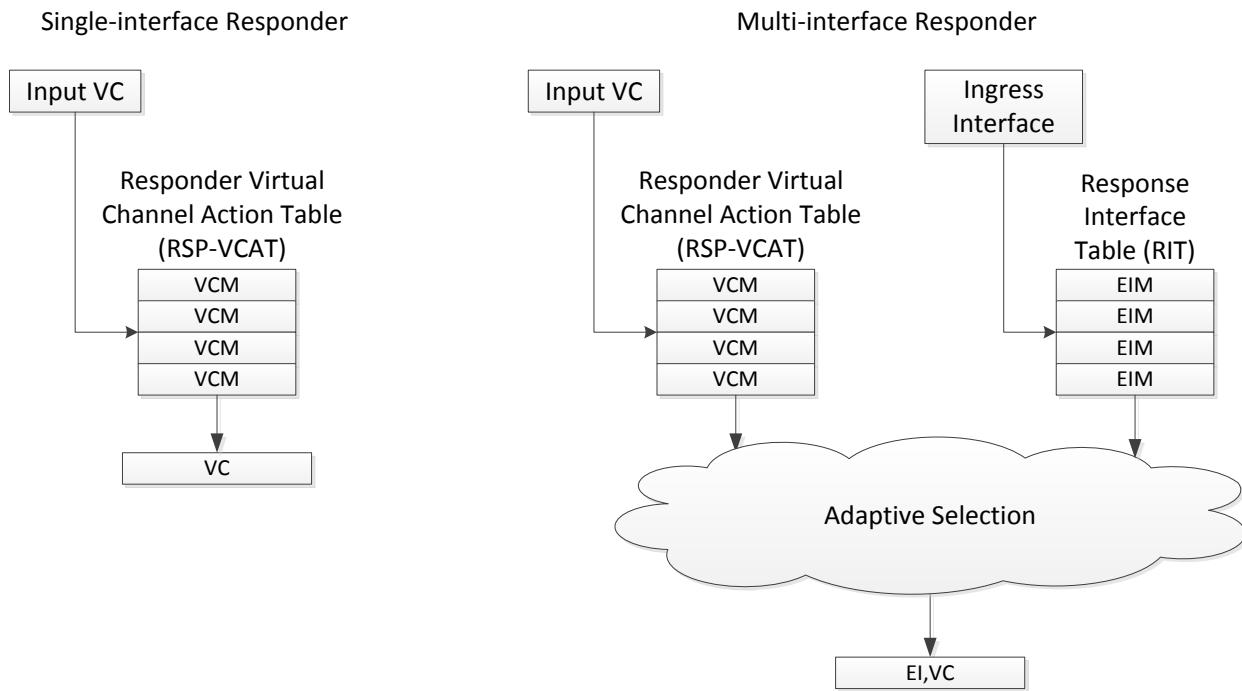


Figure 8-50: Responder Egress Selection

#### 8.29.4. Multi-subnet Requester Egress Selection

*Multi-subnet Requester Egress Selection* illustrates the structures and conceptual steps used by a Requester that supports multiple routes to each destination to select an egress interface. The following assumes MSS = 1b and HCS = 1b. :

1. The Requester accesses the MSDT, and uses the Responder's SID to directly index and locate the corresponding MSDT route entry row.
2. The Requester uses the Traffic Class corresponding to the Responder's addressable resource to index the VCAT and selects a row. If a Requester supports a Requester ZMMU, then the Traffic Class from the corresponding Requester PTE entry is used; if not, then the Requester uses a component-specific method to manage derive the Traffic Class.
3. The following pseudo-code illustrates table selection:

```

if ST[VC]:
    if GC && DSID ≠ LSID:
        if SSDT.MHC == 0:
            use MSDT routes
        elif LTF[VC]:
            use SSDT routes
        else:
            use MSDT and SSDT routes
    else:
        use SSDT routes
else:
    if GC && DSID ≠ LSID:
        use MSDT routes

```

else:  
 use SSDT routes

4. The following pseudo-code illustrates hop count computation:  
 if GC && DSID ≠ LSID:  
 if SS:  

$$CHC = SSDT.MHC + MSDT.MHC$$
  
 else:  

$$CHC = DHC + MSDT.MHC$$
  
 else:  

$$CHC = SSDT.MHC$$

5. If V == 1b, then the route entry can be used, else the route entry is ignored.

6. Action Selector is VCM = VCMs[VCA]

7. Use Traffic Class to index REQ-VCAT to obtain VCM and threshold (TH)

8. The following pseudo-code illustrates how a threshold comparison is performed:  
 if TE[VC]:  
 if  $CHC > TH$ :  

$$VCM = 0x00000000$$
  
 else:  

$$VCM = VCM$$
  
 else:  

$$VCM = VCM$$

9. Select an egress interface and VC, and copy the VC to the request packet's VC field.

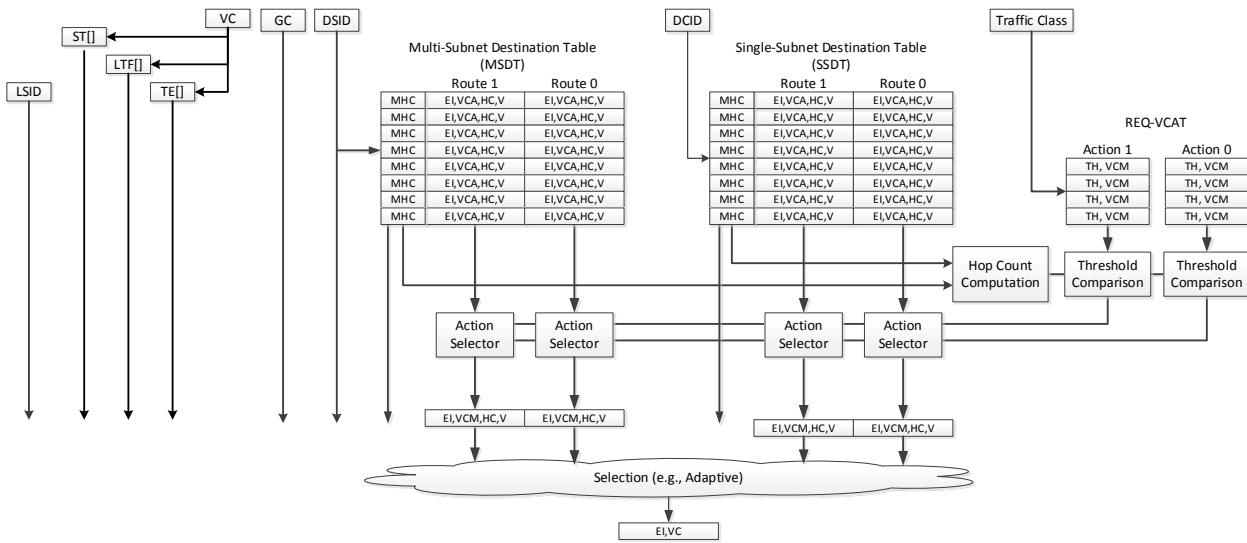


Figure 8-51: Multi-subnet Requester Egress Selection

## 8.30. Vendor-Defined Structure

A Vendor-Defined structure enables a component to surface vendor-defined Control Space structures using standardized access and location mechanisms.

The following are the features and requirements associated with the Vendor-Defined Structure:

- A component may support zero or more Vendor-Defined structures.
- A component may support any mix of Vendor-Defined and Vendor-Defined with UUID structures.

- If a control structure contains a pointer to a Vendor-Defined structure, then it may point to either type of Vendor-defined structure.
- The Vendor-Defined structure format shall use the *Vendor-Defined Structure Format*. The *Core Structure* C-UUID is used to identify this structure.
- The Vendor-Defined with UUID shall use the *Vendor-Defined with UUID Structure Format*. The VD-UUID may differ from the *Core Structure* C-UUID.
- Vendor-Defined and Vendor-Defined with UUID structures may be included in any configuration structure hierarchy that includes a Vendor-Defined Pointer field or a Control Structure Pointer field, e.g., the *Core Structure*.
- A Vendor-Defined structure may contain or point to additional Vendor-Defined structures.
- The contents, access, semantics, etc. of the Vendor-Defined Structure fields beyond the three requisite fields are outside of this specification's scope.

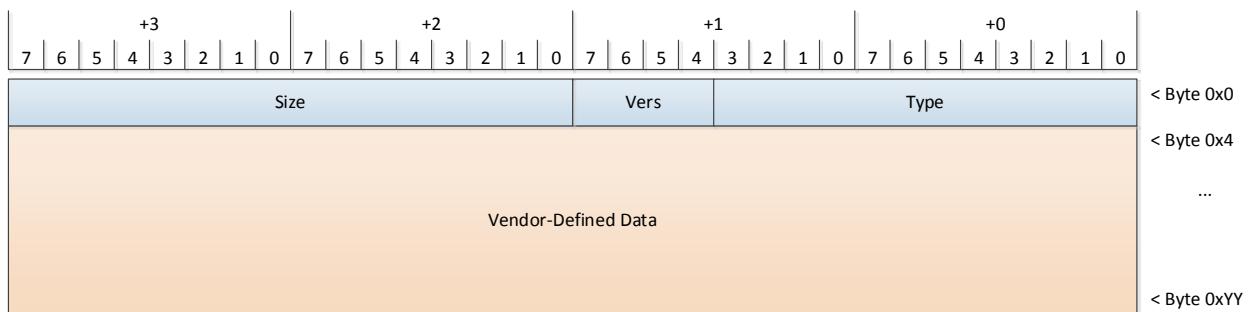


Figure 8-52: Vendor-Defined Structure Format

Table 8-94: Vendor-Defined Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Version (Vers)	4	0x1	M	RO	Structure Version
Vendor-Defined Data	-	-	-	-	Vendor-defined layout and access rules.

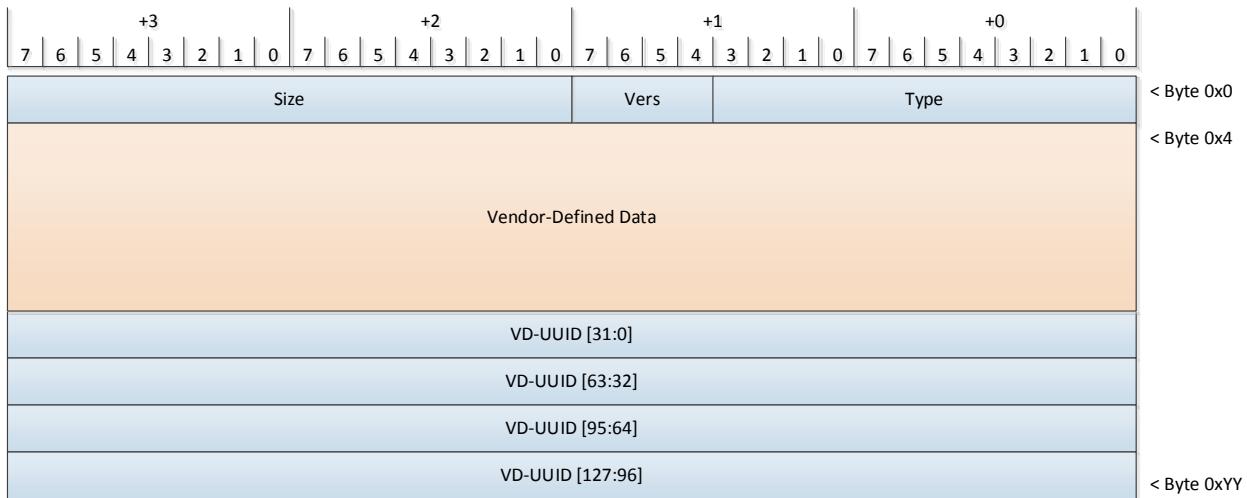


Figure 8-53: Vendor-Defined with UUID Structure Format

Table 8-95: Vendor-Defined with UUID Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Version (Vers)	4	0x1	M	RO	Structure Version
Vendor-Defined Data	-	-	-	-	Vendor-defined layout and access rules.
VD-UUID	128	-	M	RO	VD-UUID associated with this structure

## 8.31. Component Security Structure

The Component Security Structure is used to manage security certificates and Transaction Integrity Keys (TIK) and control security-related operations. It is also used to associate a given certificate and TIK pair with a given destination component's DCID. See *Security* for additional details on security certificates and TIKs.

The following are the features and requirements associated with the Component Security Structure:

- Any component type may support the Component Security Structure. If supported, then:
  - A component shall support a single Component Security Structure.
  - A component shall support the *Component PA (Peer Attribute) Structure*.
  - A component shall support the Null ART.
- The Component Security Structure shall use the *Component Security Structure Format*.
- The structure is divided into the following areas:
  - A C-Cert table contains a set of pointers to the component's own certificates within the Certificate Table.
  - A Certificate Table that supports one or more security certificate types with one or more certificate entries per certificate type.

- Certificates may be used for a variety of security services, e.g., HMAC services, encryption services, secured boot services, etc.
- The certificate table may be used independent of or in conjunction with the *Component PA (Peer Attribute) Structure* and TIK tables depending upon the security services supported by the component.
- Certificates may be accessed by multiple means depending upon the security service.
- A TIK table that supports one or more TIK table entries. Each TIK table entry is self-describing, i.e., it indicates the enabled hash function, the KDS-managed sequence number associated with this entry, and the size of the TIK (this is based on the enabled hash function).
  - Each TIK table entry shall support one TIK.
  - The length of each TIK table entry shall be 96 bytes. Due to the variable size of a given TIK, all unused table entry bytes shall be treated as Reserved.
- A component uses the *Component PA (Peer Attribute) Structure* to locate the Certificate Table entry and the TIK table entry to use when communicating with a given peer component or with a multicast group.
  - The component uses the SEC Index to locate two pointers: a Certificate Table entry pointer and a TIK table entry pointer.
  - A Null TIK Table pointer shall preclude transmitting HMAC authenticated packets.
  - If a destination receives a HMAC authenticated packet and the TIK Table pointer is Null, HMAC validation shall not be performed.
  - Authenticated communications may occur between this component and multiple destination components. If so, multiple Certificate Table entry and TIK Table entry pointer pairs may point to the same respective certificate and TIK table entries.
  - If multiple security services are associated with a given peer component or multicast group, then a component may provision multiple pointer pairs per SEC table entry.
- A component may support one or more ART (Anti-Replay Tag) methods.
  - If the Sequence Number ART is supported, a component shall provision resources to support one Sequence Number ART per TIK Table entry.
    - A component may support out-of-order request packet arrival. If supported, then the component indicates the maximum Sequence ART Window Size. See *Sequence Number ART* for additional details and requirements.
  - If the Precision Time ART is supported, then the component shall support Precision Time request and response packets, the *Component Precision Time Structure*, and Precision Time shall be enabled. All communicating components using the Precision Time ART shall participate in the same Precision Time Domain (PTD).
    - Due to variable end-to-end request transmission latencies, a component using the Precision Time ART shall support a Precision Time Window that arriving request packets with a non-zero Precision Time ART field are validated against. See *Precision Time ART* for additional details and requirements.
- A component's TIK should be periodically updated, e.g., on an annual basis. Updates may be dynamic, i.e., while the component continues to transmit and receive packets, or activated subsequent to a *Full Component Reset*, a *Warm Component Reset*. If a component supports dynamic updates, then:
  - A component shall support at least two TIK Table entries—at least one TIK Table entry is kept in a non-configured state for TIK transition purposes.

**Developer Note:** Using a non-configured TIK Table entry is conceptually an N+1 mechanism where any non-configured TIK Table entry can be used to configure a new TIK in place of an existing TIK. Management is responsible for ensuring at least one TIK Table entry is kept in the non-configured state for TIK transition purposes.

- 5
- Management shall configure a non-configured TIK Table entry with the new TIK (this can be any TIK Table entry).
  - Once configured, management configures the *Component PA (Peer Attribute) Structure* to point to the new TIK Table entry. Once updated, the component shall use the new TIK Table entry for all applicable packets.

10

**Developer Note:** There is an inherent race condition between in-flight packets using the old TIK and when the new TIK Table entry pointer is updated. Management is responsible for temporarily configuring components that support the Component Error and Signal Event Structure to avoid side-effects, e.g., triggering containment or generating a Standalone Acknowledgment indicating that a security error was detected in the corresponding request packet, until it is confident that there are no in-flight packets using the old TIK (management can calculate this time period by examining the retransmission times of the Requesters impacted by the TIK update).

15

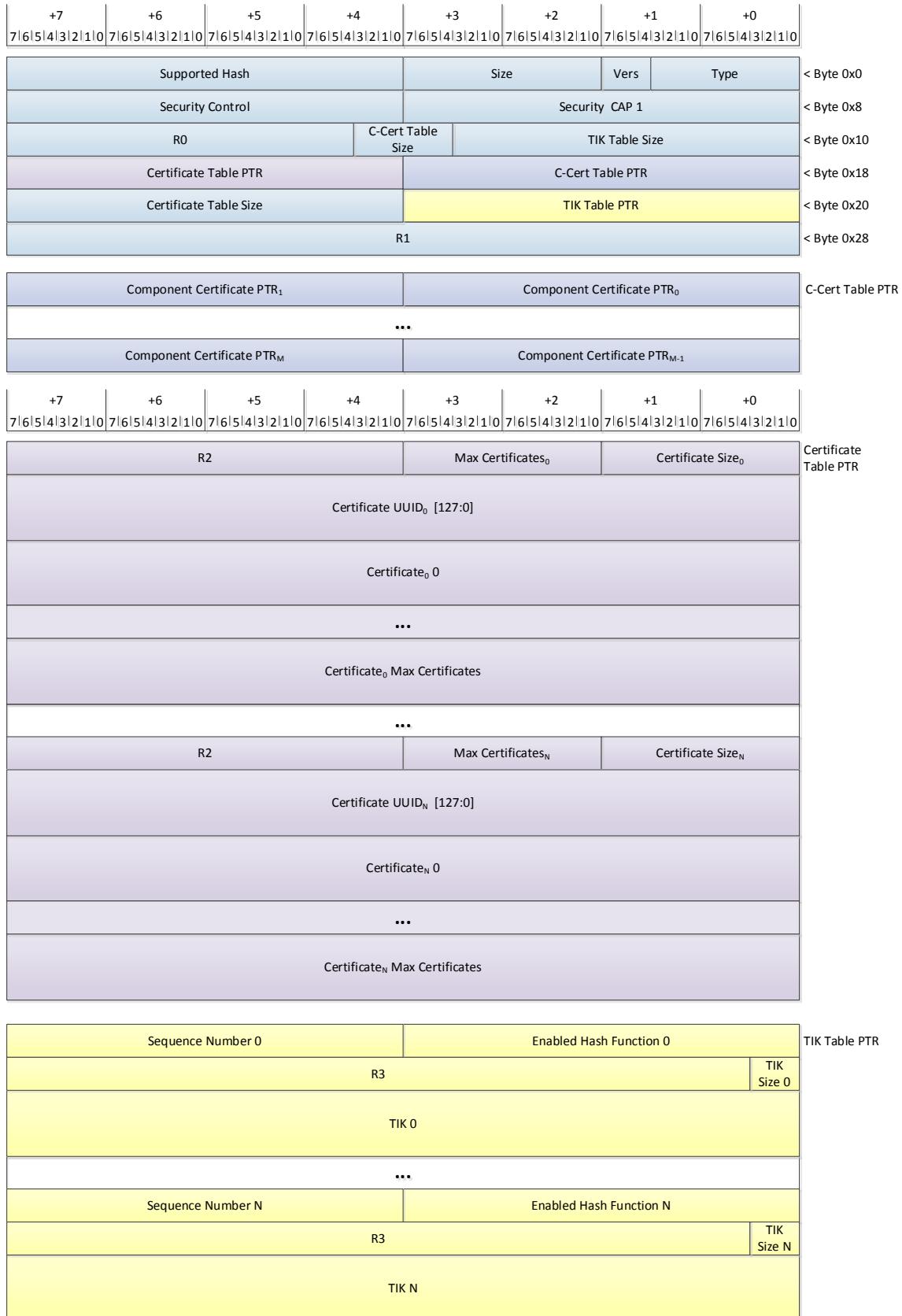


Figure 8-54: Component Security Structure Format

Table 8-96: Component Security Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>Version (Vers)</b>	4	0x1	M	RO	Structure Version
<b>Security CAP 1</b>	32	-	M	RO	Component Security Structure Capabilities
		Bit 0			HMAC Authentication Support 0b—Unsupported 1b—Supported
		Bit 1			Sequence Number ART Support 0b—Unsupported 1b—Supported
		Bit 2			Precision Time ART Support 0b—Unsupported 1b—Supported
		Bits 31:3			Reserved
<b>Security Control</b>	32	-	M	RW	Security Controls
		Bit 0			Security Enable 0b—Disable all security validation 1b—Enable all security validation
		Bit 1			HMAC Authentication Enable—This field enables the dynamic generation and validation of the HMAC field in explicit OpClass packets at the component level.  If enabled, then shall set <i>Core Structure Component CAP 1 Control Next Header Enable</i> = 1b.  The <i>Component PA (Peer Attribute) Structure</i> may be used to selectively enable the inclusion and validation of a dynamically-generated HMAC field. Selective enablement via the <i>Component PA (Peer Attribute) Structure</i> settings shall apply only if the HMAC Authentication Enable == 1b. If selective enablement is not enabled, then when HMAC Authentication Enable == 1b, all explicit OpClass packets shall contain a dynamically-generated HMAC field upon transmission, and the HMAC field shall be validated upon receipt.  Shall be enabled only if the component supports explicit OpClass packets and the <i>Explicit OpClass Next Header Field</i> .

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					<p>0b—Disabled 1b—Enabled</p> <p>Enable Sequence Number ART—This field enables the dynamic generation and validation of the Sequence Number ART field in explicit OpClass packets at the component level.</p> <p>If enabled, then the <i>Core Structure Component CAP 1 Control</i> Next Header Enable and HMAC Authentication Enable bits shall be set to 1b.</p> <p>The <i>Component PA (Peer Attribute) Structure</i> may be used to selectively enable the inclusion and validation of a dynamically-generated Sequence Number ART field. Selective enablement via the <i>Component PA (Peer Attribute) Structure</i> settings shall apply only if Enable Sequence Number ART f== 1b. If selective enablement is not enabled, then when Enable Sequence Number ART == 1b, all explicit OpClass packets shall contain a dynamically-generated Sequence Number ART field upon transmission, and the Sequence Number ART field shall be validated upon receipt.</p> <p>Shall be enabled only if the component supports explicit OpClass packets and the <i>Explicit OpClass Next Header Field</i>.</p> <p>If enabled, no other ART shall be enabled.</p> <p>0b—Disabled 1b—Enabled</p>
		Bit 3			<p>Enable Precision Time ART—This field enables the dynamic generation and validation of the Precision Time ART field in explicit OpClass packets at the component level.</p> <p>If enabled, then the <i>Core Structure Component CAP 1 Control</i> Next Header Enable field and the HMAC Authentication Enable bits shall be set to 1b.</p> <p>The <i>Component PA (Peer Attribute) Structure</i> may be used to selectively enable the inclusion and validation of a dynamically-generated Precision Time ART field. Selective enablement via the <i>Component PA (Peer Attribute) Structure</i> settings shall apply only if the Enable Precision Time ART</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					<p><math>\text{== 1b}</math>. If selective enablement is not enabled, then when Enable Precision Time ART <math>\text{== 1b}</math>, all explicit OpClass packets shall contain a dynamically-generated Precision Time ART field upon transmission, and the Precision Time ART field shall be validated upon receipt.</p> <p>Shall be enabled only if the component supports explicit OpClass packets and the <i>Explicit OpClass Next Header Field</i>.</p> <p>If enabled, no other ART shall be enabled.</p> <p>0b—Disabled 1b—Enabled</p>
		Bit 4			<p>Enable Null ART—This field enables the dynamic generation and validation of the Null ART field in explicit OpClass packets at the component level.</p> <p>If enabled, then the <i>Core Structure Component CAP 1 Control</i> Next Header Enable and the HMAC Authentication Enable bits shall be set to 1b.</p> <p>The <i>Component PA (Peer Attribute) Structure</i> may be used to selectively enable the inclusion and validation of a dynamically-generated Null ART field. Selective enablement via the <i>Component PA (Peer Attribute) Structure</i> settings shall apply only if Enable Null ART <math>\text{== 1b}</math>. If selective enablement is not enabled, then when Enable Null ART <math>\text{== 1b}</math>, all explicit OpClass packets shall contain a dynamically-generated Null ART field upon transmission, and the Null ART field shall be validated upon receipt.</p> <p>Shall be enabled only if the component supports explicit OpClass packets and the <i>Explicit OpClass Next Header Field</i>.</p> <p>If enabled, no other ART shall be enabled.</p> <p>0b—Disabled 1b—Enabled</p>
		Bits 12:5			<p>Sequence Number ART Window Size specifies the maximum allowed time window to accept out-of-order request packets.</p> <p>If Sequence Number ART Window Size <math>\text{== 0}</math> then</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Supported Hash	32				Out-of-order packets shall fail packet validation else SN Time Window = (Sequence Number ART Window Size * 10 ns)
		Bits 20:13			Precision Time ART Time Window specifies the maximum difference between the component's Master Time and the packet's Precision Time ART. If Precision Time ART Time Window == 0 then PT Difference = 1 us else PT Difference = (Precision Time ART Time Window * 50 ns)
		Bits 31:21			RsvdP
		-			Bit mask indicating the supported hash functions. If Bit <sub>k</sub> == 1b then the component supports the corresponding hash function; if Bit <sub>k</sub> == 0b, then it does not.
		Bit 0			SHA-224
		Bit 1			SHA-256
		Bit 2			SHA-384
		Bit 3			SHA-512
		Bit 4			SHA-512/224
		Bit 5			SHA-512/256
TIK Table Size	28	-	M	RO	Indicates the number of TIK table entries supported by the component. If TIK Table Size == 0x0, then the number of entries shall be 2 <sup>28</sup> .

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>C-Cert Table Size</b>	8	-	M	RO	Indicates the number of C-Cert Table entries If C-Cert Table Size == 0x0, then the number of entries shall be $2^8$ .
<b>C-Cert Table PTR</b>	32	-	M	RO	Pointer to the first C-Cert Table entry
<b>Component Certificate PTR<sub>K</sub></b>	32	-	M	RW	A Component Certificate Pointer points to a certificate assigned to the component itself or is Null.
<b>Certificate Table PTR</b>	32	-	M	RO	Pointer to the first Certificate Table entry
<b>TIK Table PTR</b>	32	-	M	RO	Pointer to the first TIK Table entry
<b>Certificate Table Size</b>	32	-	M	RO	Indicates the number of Certificate Table entries If Certificate Table Size == 0x0, then the number of entries shall be $2^{32}$ .
<b>Certificate Size</b>	16	-	M	RO	Size of each certificate (in bytes)
<b>Max Certificates<sub>K</sub></b>	16	-	M	RO	Maximum number of certificates associated with a given certificate table entry If Max Certificates == 0x0, then the number of entries shall be $2^{16}$ .
<b>Certificate UUID</b>	128	-	M	RO	Certificate UUID identifies the type of certificate associated with a given certificate table entry.
<b>Certificate</b>	-	-	M	RW	Public Key Certificate, e.g., a X.509 certificate. Each certificate is Certificate Size bytes in length. A certificate may serve multiple purposes.
<b>Enabled Hash</b>	32	-	M	RW	If Bit <sub>K</sub> == 1b, then the TIK uses the corresponding hash function. Only a single Bit <sub>K</sub> shall be set to 1b at any given time.
					SHA-224
					SHA-256
					SHA-384
					SHA-512
					SHA-512/224
					SHA-512/256

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Sequence Number		Bit 6			SHA3-224
		Bit 7			SHA3-256
		Bit 8			SHA3-384
		Bit 9			SHA3-512
		Bits 31:10			RsvdP
	32	-	M	RW	Sequence number corresponding to this TIK table entry. This number shall be managed by the KDS.
	TIK Size	4	M	RW	Size of the configured TIK
					Each TIK shall be placed such that TIK byte 0 is placed at TIK entry byte 0xC of a given TIK Table entry, and each subsequent byte is contiguously placed thereafter.
					0x0—28 bytes
					0x1—32 bytes
					0x2—48 bytes
TIK <sub>K</sub>		-	M	RW	0x3—64 bytes
					0x4-0xF—Reserved
					The TIK associated with a given TIK table entry, in which the payload is a cipher text Enc_Ki (TIK, S) with the length of the TIK in bits, and the Next Header includes a 32-bit S value, a 32-bit cleared bit string and a 64-bit HMAC value.
					See <i>Transaction Integrity Keys (TIK) Management</i> .
R0	28	-	-	-	Reserved
R1	64	-	-	-	Reserved
R2	32	-	-	-	Reserved
R2	60	-	-	-	RsvdP

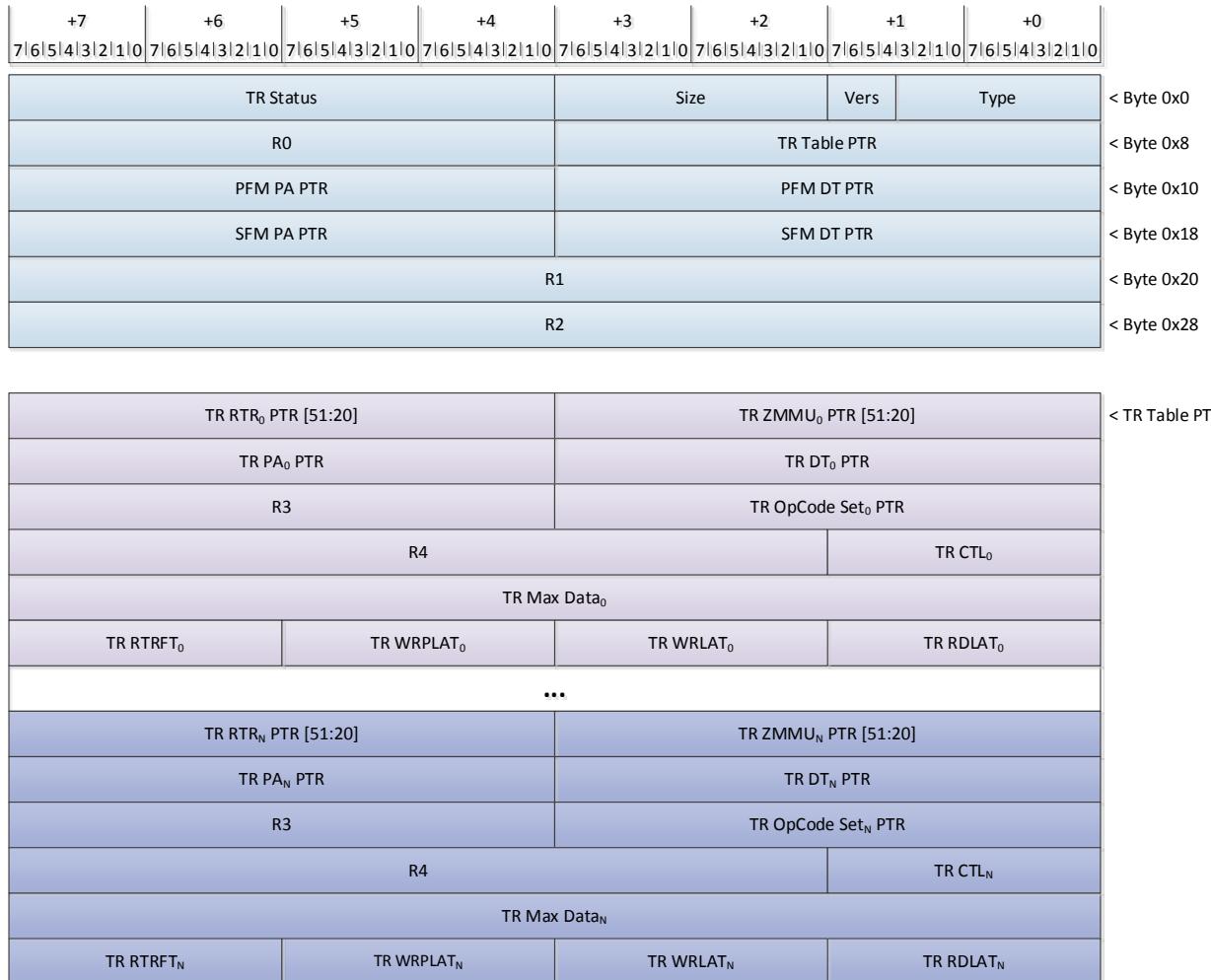
## 8.32. Component TR Structure

The Component TR Structure is used to configure a *Transparent Router (TR)*. The Component TR structure primarily consists of a set of pointers to tables and structures used to translate request and response packets into their destination subnet versions.

5 The following are the features and requirements of the Component TR Structure:

- Any component type may support the Component TR Structure.

- If supported, a component shall support a single Component TR Structure.
- The Component TR Structure shall use the *Component TR Structure Format*.
- Each TR Table entry contains the following fields:
  - A pointer to the ZMMU used to translate request packets. Each ZMMU shall be located on an integer  $2^{20}$  byte address multiple.
  - A pointer to the RTR used to track request packets and translate response packets. Each RTR shall be located on an integer  $2^{20}$  byte address multiple.
  - A pointer to the *Component Destination Table Structure* used to translate request packets.
  - A pointer to the *Component PA (Peer Attribute) Structure* used to generate and validate request and response packets.
  - A Max Data field that represents the aggregate of the addressable resources in the destination subnets.
  - A set of read and write latency fields used to calculate Requester retransmission timers. These timers reflect the worst-case latency when transparently accessing any Responder within any destination subnet through the subnet-local TR.
  - A failsafe timer release resources.
  - A pointer to the *OpCode Set Structure* that is used to configure the supported and enabled OpCodes within a destination subnet. This is used to validate and generate request and response packets.
- TR ZMMU implementation should support the Requester ZMMU semantics specified in *Memory Management* and the TR Requester PTE specified in *Transparent Router (TR)*.
- TR RTR implementation is outside of this specification's scope.
- A TR shall maintain a RTR failsafe timer (RTRFT) per subnet.
  - A TR may continuously run the RTRFT or start and stop it based on whether there are outstanding translated request packets.
  - If a response packet for an outstanding translated request packet is not received for at least two RTRFT expirations, then the TR shall release the corresponding RTR table entry.
  - The RTRFT time interval is calculated as follows:
    - $$RTRFT = 2^{TR.RTRFTE} * 100 \text{ ps, +50\%, -0\%}$$



The diagram illustrates the memory layout of the Component TR Structure. It consists of two main sections: a header and a body. The header is a fixed-size structure with fields: TR Status, Size, Vers, Type, R0, PFM PA PTR, PFM DT PTR, SFM PA PTR, SFM DT PTR, R1, and R2. The body is a variable-size structure starting with R3, followed by R4, and then repeating the pattern for N entries. Each entry in the body includes fields: TR RTR<sub>0</sub> PTR [51:20], TR ZMMU<sub>0</sub> PTR [51:20], TR PA<sub>0</sub> PTR, TR DT<sub>0</sub> PTR, TR OpCode Set<sub>0</sub> PTR, TR CTL<sub>0</sub>, TR Max Data<sub>0</sub>, TR RTRFT<sub>0</sub>, TR WRPLAT<sub>0</sub>, TR WRLAT<sub>0</sub>, TR RDLAT<sub>0</sub>, and an ellipsis. The body ends with R4, TR Max Data<sub>N</sub>, TR RTRFT<sub>N</sub>, TR WRPLAT<sub>N</sub>, TR WRLAT<sub>N</sub>, and TR RDLAT<sub>N</sub>. The entire structure is aligned to 8 bytes.

Field	Size (bits)	Value / Bit Location	M / O	Access	Description
Version (Vers)	4	0x1	M	RO	Structure Version
TR Status	32	-	M	-	See <i>TR Status</i>
PFM DT PTR	32	-	M	RW	Pointer to a <i>Component Destination Table Structure</i> corresponding to the subnet where the Primary Fabric Manager is located. The Primary Fabric Manager is identified by <i>Core Structure</i> PFMCID. A TR does not support Subnet Identifiers. As such, the <i>Core Structure</i> PFMSID shall be Null.
PFM PA PTR	32	-	M	RW	Pointer to a <i>Component PA (Peer Attribute) Structure</i> corresponding to the subnet where

Figure 8-55: Component TR Structure Format

Table 8-97: Component TR Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Version (Vers)	4	0x1	M	RO	Structure Version
TR Status	32	-	M	-	See <i>TR Status</i>
PFM DT PTR	32	-	M	RW	Pointer to a <i>Component Destination Table Structure</i> corresponding to the subnet where the Primary Fabric Manager is located. The Primary Fabric Manager is identified by <i>Core Structure</i> PFMCID. A TR does not support Subnet Identifiers. As such, the <i>Core Structure</i> PFMSID shall be Null.
PFM PA PTR	32	-	M	RW	Pointer to a <i>Component PA (Peer Attribute) Structure</i> corresponding to the subnet where

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					the Primary Fabric Manager is located. This structure is used to generate and validate packets.
<b>SFM DT PTR</b>	32	-	M	RW	<p>Pointer to a <i>Component Destination Table Structure</i> corresponding to the subnet where the Secondary Fabric Manager is located. The Secondary Fabric Manager is identified by <i>Core Structure SFMCID</i>.</p> <p>A TR does not support Subnet Identifiers. As such, the <i>Core Structure SFMSID</i> shall be Null.</p>
<b>SFM PA PTR</b>	32	-	M	RW	Pointer to a <i>Component PA (Peer Attribute) Structure</i> corresponding to the subnet where the Secondary Fabric Manager is located. This structure is used to generate and validate packets.
<b>TR Table PTR</b>	32	-	M	RO	Pointer to the TR Table
<b>TR ZMMU PTR</b>	32	-	M	RO	This field points to the ZMMU used to translate request packets.
<b>TR RTR PTR</b>	32	-	M	RO	This field points to the RTR used to track request and response packets.
<b>TR DT PTR</b>	32	-	M	RO	This field points to the <i>Component Destination Table Structure</i> used to select the component egress interface and VC to transmit a packet in a given subnet.
<b>TR PA PTR</b>	32	-	M	RO	This field points to the <i>Component PA (Peer Attribute) Structure</i> used to generate and validate packets within a given subnet.
<b>TR OpCode Set PTR</b>	32	-	M	RO	This field points to the <i>OpCode Set Structure</i> that is used to configure the supported and enabled OpCodes within a given subnet.
<b>TR Max Data</b>	64	-	M	RW	This field contains the Max Data the aggregate addressable resources of all destination subnets.
<b>TR CTL</b>	16	Bit 0	M	RW	Individual Subnet Control Field
					TR Relay Enable—Enable packet translation and relay to or from this subnet
					0b—Disable Packet Relay

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
TR RDLAT		Bit 1			1b—Enable Packet Relay TR ZMMU Reset—Reset the ZMMU used to translate request packets from this subnet. All associated resources shall be released and returned to their uninitialized state. 1b—Reset ZMMU Reads of this bit shall return 0b.
					TR RTR Reset—Reset the RTR associated with this subnet. All associated resources shall be released and returned to their uninitialized state. 1b—Reset RTR Reads of this bit shall return 0b.
		Bit 2			RsvdP
		Bit 15: 3			Worst-case latency from the time that the first bit of a Read request packet is received, translated, and transmitted until the last bit of the corresponding Read Response packet is received, translated, and transmitted by a TR. Latency = TR RDLAT * Core Latency Scale
TR WRLAT	16	-	M	RW	Worst-case latency from the time that the first bit of a Write request packet is received, translated, and transmitted until the last bit of the corresponding <i>Standalone Acknowledgment</i> is received, translated, and transmitted by a TR. Latency = WRLAT * Core Latency Scale
TR WRPLAT	16	-	M	RW	Worst-case latency from the time that the first bit of a Write Persistent request packet is received, translated, and transmitted until the last bit of the corresponding <i>Standalone Acknowledgment</i> is received, translated, and transmitted by a TR. Latency = TR WRPLAT * Core Latency Scale If the TR does not support Write Persistent translation, then this field shall be Reserved.
TR RTRFT	16	-	M	RW	Indicates the RTRFT in Core Latency Scale

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
R0	32	-	-	-	Reserved
	64	-	-	-	Reserved
	64	-	-	-	Reserved
	32	-	-	-	Reserved
	48	-	-	-	Reserved

Table 8-98: TR Status

Bit Location	Access	Description
Bit 0	RW1C	Request Packet Relay Failure—Indicates one or more request packets were discarded due to request packet translation failure
Bit 1	RW1C	Response Packet Relay Failure—Indicates one or more response packets were discarded due to response packet translation failure
Bit 2	RW1C	Access Key Status—Indicates one or more packets were discarded due to Access Key validation failure
Bit 31:1	-	RsvdZ

### 8.33. Service UUID Structure

If a component supports multiple service types, then it should provision a Service UUID structure. The Service UUID structure communicates one or more component class encodings and UUIDs that are used to identify services.

The following are the features and requirements of the Service UUID structure:

- Any component type may support the Service UUID structure.
- The Service UUID structure shall use the *Service UUID Structure Format*.
- A component that supports a single service may support the Service-UUID structure. Management shall use the Service-UUID instead of the *Core Structure* C-UUID to load service-specific software (e.g., the C-UUID is used to identify the hardware, vendor-defined capabilities, etc., and the Service-UUID is used to identify the service-specific software).
- If the *Core Structure* Base C-Class == Multi-Class Component, then the component shall provision the Service UUID structure.
- A component may support up to Max-UUID Class and Max-SI fields tuples.
  - With the exception of the Multi-Class Component and Reserved encodings, a Class field may contain any encoding specified in *Appendix C Component Class Encodings*.
  - Multiple Class fields may contain the same component class encoding.

- Each Class field shall be associated with a single Service-UUID, i.e., Class 0 is associated with Service-UUID 0, Class 1 is associated with Service-UUID 1, and so forth.
- Each Max-SI field shall be associated with a single Service-UUID, i.e., Max-SI 0 is associated with Service-UUID 0, Max-SI 1 is associated with Service-UUID 1, and so forth.
- 5     ● A component may support up to Max-UUID Service-UUID fields. Multiple Service-UUID fields may contain the same UUID value.
- 10    ● Each Class, Max-SI, and Service-UUID fields shall contain non-zero values, i.e., shall contain a valid component class encoding, a maximum number of service instance entries per Service-UUID, and a UUID value.
- 15    ● If *Core Structure* Base C-Class ≠ Multi-Class Component, then all Class fields shall be set to Base C-Class.
- 20    ● For each Service-UUID, a component may support up to Max-SI service instance table entries. Each table entry contains:
  - A SI-DS Base. Each SI-DS Base shall be an integer 4096-byte address multiple that identifies a resource of SI-DS Length in the component's Data Space. If a component supports multiple SI-DS Base, then each SI's Data Space resource shall be independent from any other SI's Data Space resource accessed through a SI-DS Base, i.e., the resources shall not overlap with one another.
  - A SI-CS Base. Each SI-CS Base shall be an integer 4096-byte address multiple that identifies a resource of SI-CS Length in the component's Control Space. If a component supports multiple SI-CS Base, then each SI's Control Space resource shall be independent from any other SI's Control Space resource accessed through a SI-CS Base, i.e., the resources shall not overlap with one another.

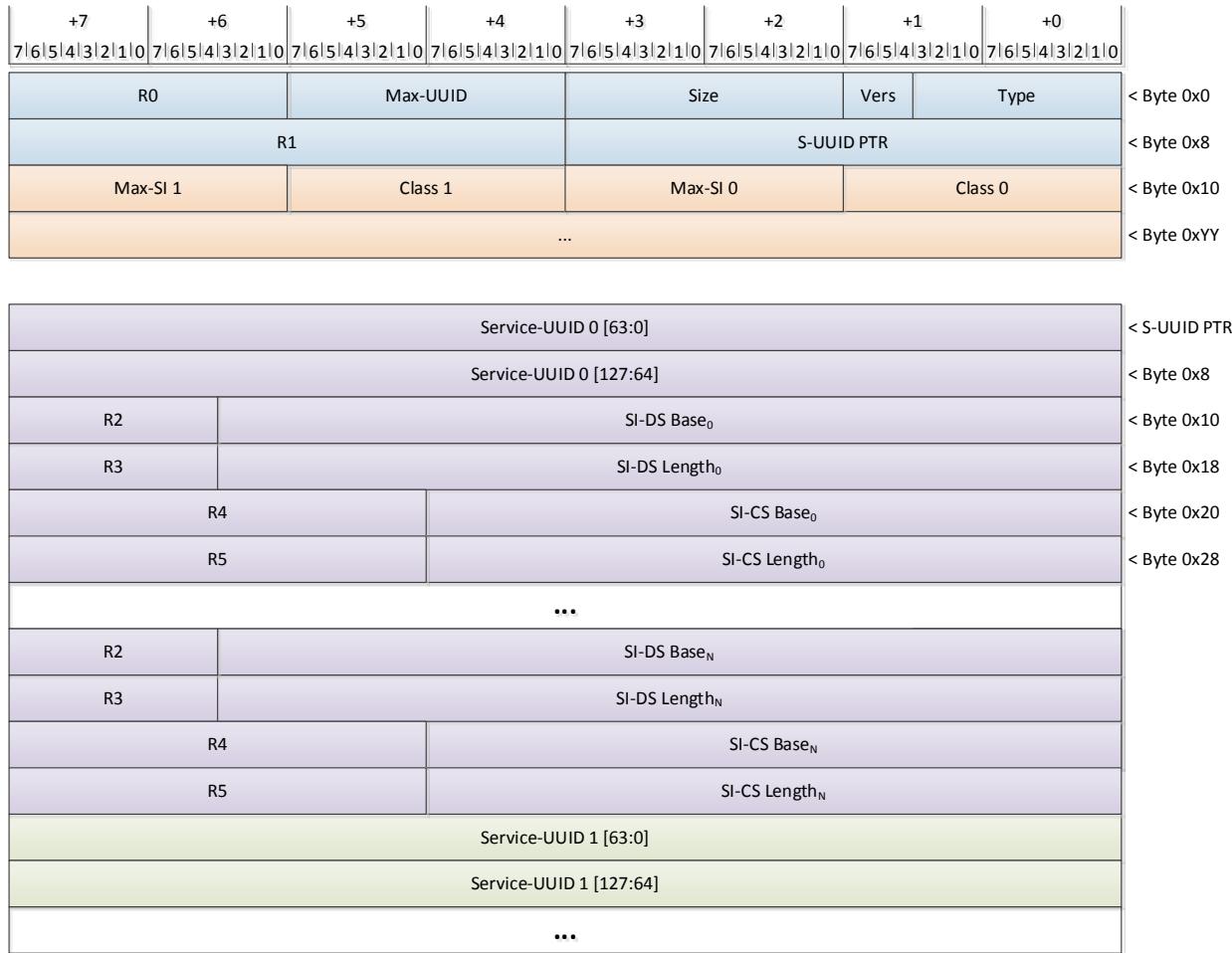


Figure 8-56: Service UUID Structure Format

Table 8-99: Service UUID Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Version (Vers)	4	0x1	M	RO	Structure Version
Max-UUID	16	-	M	RO	Maximum number of Class and Service-UUID fields contained in this structure. If Max-UUID == 0x0, then the number of Service-UUID and Class fields shall be $2^{16}$ .
S-UUID PTR	32	-	M	RO	Pointer to byte 0 of the first Service-UUID table entry (Service-UUID 0).
Class <sub>N</sub>	16	-	M	RO	Class associated with a given Service-UUID  If the number of Class fields is less than what is required to ensure the structure size is an integer 16-byte multiple, then pad bytes shall be appended after

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					the last Class entry to make the structure size an integer 16-byte multiple.
Max-SI <sub>N</sub>	16	-	M	RO	Maximum number of service instances associated with each Service-UUID. If Max-SI == 0x0, then $2^{16}$ service instances shall be provisioned.
Service-UUID <sub>N</sub>	128	-	O	RO	Each Service-UUID identifies a given component-specific service.
SI-DS Base <sub>K</sub>	52	-	O	RO	SI-DS Base shall be an integer 4096-byte address multiple that identifies a resource of SI-DS Length in the component's Data Space that is associated with the service instance.
SI-DS Length <sub>K</sub>	40	-	O	RO	Length of the Data Space resource associated with the service instance equals (SI-DS <sub>Length</sub> * 4096 bytes).
SI-CS Base <sub>K</sub>	40	-	O	RO	SI-CS Base shall be an integer 4096-byte address multiple that identifies a resource of SI-CS Length in the component's Control Space that is associated with the service instance.
SI-CS Length <sub>K</sub>	40	-	O	RO	Length of the Control Space resource associated with the service instance equals (SI-CS <sub>Length</sub> * 4096 bytes).
R0	16	-	-	-	RsvdP
R1	32	-	-	-	Reserved
R2	12	-	-	-	RsvdP
R3	12	-	-	-	RsvdP
R4	24	-	-	-	RsvdP
R5	24	-	-	-	RsvdP

## 8.34. Component C-Access Structure

The Component C-Access structure is used to provide fine-grain access isolation within a component's Control Space when accessed using Control OpClass request packets. By selectively placing Control Space structures into different regions, management can provide fine-grain access control to Control Space, e.g., different management entities can be permitted access to a subset of the regions.

The following are the features and requirements of the Component C-Access structure:

- If a component supports the Control OpClass, then it should support the Component C-Access structure.
- The Component C-Access structure shall use the *Component C-Access Structure Format*.
- Each Component C-Access structure supports a single page size (see C-Page Size field).

- If a component supports multiple page sizes, then it shall support one Component C-Access structure per page size. Additional structures are located using the Next C-Access PTR field.
- The size of the region covered by a Component C-Access structure is (C-Access R-Key Table Size \* page size).
- A Component C-Access structure specifies a Base Address used to locate byte zero of the first page covered by the structure.
  - If the size of the region is a power of 2, then the Base Address shall be a multiple of the size of the region.
  - If the size of the region is not a power of 2, then the Base Address shall be a multiple of the quantity (region size rounded up to the next power of 2). For example, for a 14-entry structure with 1 MiB pages, the Base Address shall be a multiple of 16 MiB.
  - Bits 11:0 of the Base Address are not present and are implied to be zero.
- The address range covered by a Component C-Access structure is:  
Base Address to (Base Address + region size – 1)
- If a control structure or a control structure-specific resource is larger than its associated Component C-Access structure's page size, then it shall be located on multiple contiguous pages. If multiple pages are consumed, then software shall configure each C-Access R-Key Table entry with the same R-Keys.
- Each C-Access R-Key Table entry contains two fields—a C-Access RO R-Key and a C-Access RW R-Key. These R-Keys shall be as specified in a *Region Key (R-Key)*, and used to validate Control OpClass request packet access as specified in *Access Permission Validation*.
  - A Responder masks off portions of a request packet's address field to locate the corresponding Component C-Access structure and generate an index into the C-Access R-Key Table.
- The first Component C-Access structure shall be pointed to by the *Core Structure*, and its first page shall cover the *Core Structure*.
- Any remaining Component C-Access structures shall be located by following Next C-Access PTR fields, starting with the one in the first Component C-Access structure.

**Developer Note:** Software on some Requesters needs to efficiently access the *Core Structure* and a number of other Control Space structures. It's desirable to map the entire set of pages covered by the first Component C-Access structure statically. To make this practical, the following is recommended:

- Developers should place multiple Control Space structures on a common page when they don't require distinct R-Keys. See *Control Structure Organization*.
- Use a page size small enough to avoid excessive wasting of Control Space mapped by these pages, e.g., if the Control Space structures are small, use 4 KiB pages.

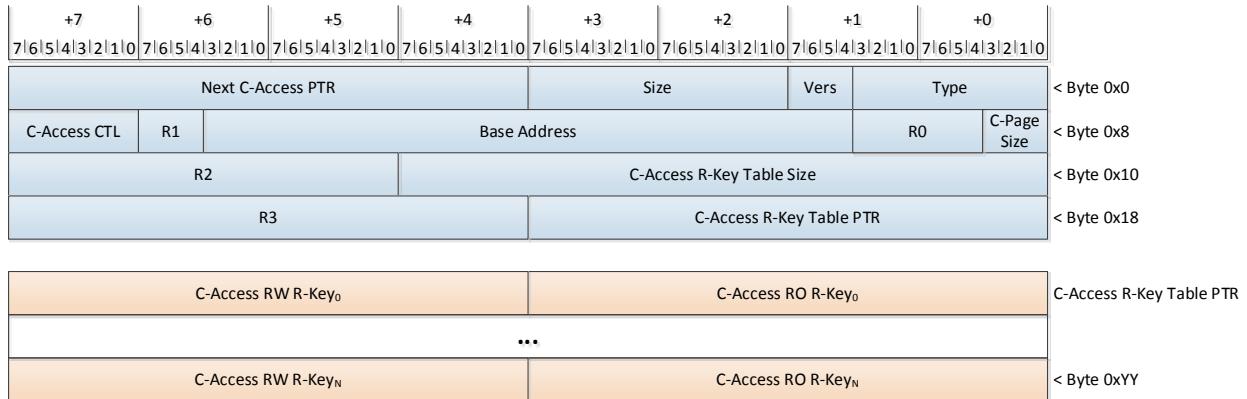


Figure 8-57: Component C-Access Structure Format

Table 8-100: Component C-Access Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>Version (Vers)</b>	4	0x1	M	RO	Structure Version
<b>Next C-Access PTR</b>	32	-	M	RO	If a component supports multiple Component C-Access structures, then this field links these together. If there are no additional Component C-Access structures, then this field shall be set to Null.
<b>C-Page Size</b>	4	-	M	RO	Page size associated with this instance of the Component C-Access structure. 0x0— $2^{12}$ bytes (4 KiB) 0x1— $2^{16}$ bytes (64 KiB) 0x2— $2^{20}$ bytes (1 MiB) 0x3— $2^{25}$ bytes (32 MiB) 0x4-0xF—Reserved
<b>C-Access CTL</b>	8	-	M	RW	C-Access control field
	Bit 0				Enable C-Access R-Key Validation—if enabled, then the R-Key validation shall be performed on all Control OpClass request packets. 0b—Disabled 1b—Enabled
					Reset C-Access R-Keys—if 1b, then all C-Access RO R-Key and C-Access RW R-Key fields shall be set to the Default R-Key 1b—Reset C-Access R-Keys Upon initiating a reset, reads of this bit shall return 1b.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Base Address		Bits 7:2			Upon reset completion, reads of this bit shall return 0b.
					Updates to C-Access RO R-Key or C-Access RW R-Key fields prior to reset completion may result in non-deterministic behavior.
					RsvdP
Base Address	40	-	M	RO	The address of byte 0 of the first page within Control Space that is protected by this instance of the Component C-Access structure. Base Address bits 11:0 are not present and are implied to be 0x0.
C-Access R-Key Table Size	40	-	M	RO	Number of R-Key Table entries within this structure.
C-Access R-Key Table PTR	32	-	M	RO	Pointer to the C-Access R-Key Table associated with this structure
C-Access RO R-Key	32	-	M	RW	Read-only R-Key associated with a given page
C-Access RW R-Key	32	-	M	RW	Read-Write R-Key associated with a given page
R0	8	-	-	-	RsvdP
R1	4	-	-	-	RsvdP
R2	24	-	-	-	Reserved
R3	32	-	-	-	Reserved

## 8.35. Component PA (Peer Attribute) Structure

The Component PA structure is used to configure the following tables used to generate and validate the peer component-specific or multicast group-specific portions of request and response packets:

- If a component supports multiple peer attribute configurations to communicate with diverse peer components, then a component uses the PA Table to configure the peer attributes used to generate and validate packets. Alternatively, if a component supports a single peer attribute entry, then it may use the wildcard peer attribute.
- If a component supports multiple security certificates or TIKs (see the *Component Security Structure*), then components use the SEC Table to associate security certificates and TIKs with a peer component or multicast group. Alternatively, if a component supports a single security certificate and a single TIK, then it may use the wildcard certificate and TIK.

- The SSAP, MSAP, MCAP, and MSMCAP tables contain a set of indices used to access the PA Table and SEC Table. Requesters use these tables to generate request packets and validate response packets. Responders use these tables to validate request packets and generate response packets.

- Within a single-subnet topology,

- When generating a unicast request packet, a Requester uses the destination component's CID to index the SSAP table. When validating a unicast response packet, a Requester uses the response packet's SCID to index the SSAP table.
- When generating a multicast request packet, a Requester uses the destination MGID to index the MCAP table. For Reliable multicast acknowledgments, a Requester uses the response packet's SCID to index the SSAP table.
- When validating a unicast request packet or generating a unicast response packet, a Responder uses the request packet's SCID to index the SSAP table.
- When validating a multicast request packet, a Responder uses the request packet's MGID to index the MCAP table. To generate a Reliable multicast acknowledgment, a Responder uses the request packet's SCID to index the SSAP table.
- Due to the potentially large number of components within a multi-subnet topology, the mechanisms to index the MSAP and MSMCAP tables are outside of this specification's scope. Instead, component-specific functions are used to locate the appropriate MSAP or MSMCAP table index. For example, by organizing components into groups and applying intelligent [CID, SID] or [MGID, Global Multicast Prefix] assignment, a subset of bits from each identifier can be used to construct the corresponding index.
  - When generating a unicast request packet, a Requester applies a component-specific function to the destination component's [CID, SID] to index the MSAP table. When validating a unicast response packet, a Requester applies a component-specific function to the response packet's [SCID, SSID] to index the MSAP table.

**Developer Note:** *A Requester could save the results of the component-specific function applied during request packet generation (e.g., the MSAP table index) and use this to improve response packet validation efficiency.*

- When generating a multicast request packet, a Requester applies a component-specific function to the destination multicast group's [MGID, Global Multicast Prefix] to index the MSMCAP table. When validating a *Reliable Multicast Acknowledgment*, a Requester applies a component-specific function to the response packet's [SCID, SSID] to index the MSAP table.
- When validating a unicast request packet or generating a unicast response packet, a Responder applies a component-specific function to the request packet's [SCID, SSID] to index the MSAP table.
- When validating a multicast request packet, a Responder applies a component-specific function to the request packet's [MGID, Global Multicast Prefix] to index the MSMCAP table. If a *Reliable Multicast* request packet, then the Responder applies a component-specific function to the request packet's [SCID, SSID] to index the MSAP table to generate a *Reliable Multicast Acknowledgment*.
- If a component supports multiple Access Keys, then components use the AKey field within the SSAP, MSAP, MCAP, and MSMCAP tables to configure the Access Key used to generate or validate packets. Alternatively, if a component supports a single Access Key, then it may use the wildcard Access Key.

The following are the features and requirements of the Component PA structure:

- Any component type that supports explicit OpClass packets should support the Component PA structure. If a component supports Access Keys, packets with the *Explicit OpClass Next Header Field* present, or multi-subnet communications, then it shall support the Component PA structure.
- The Component PA structure shall use the *Component PA Structure Format*.
- If a component supports a single Access Key, then the Wildcard Access Key field should be used in place of the AKey field within the SSAP, MSAP, MCAP, and MSMCAP tables. If used, then:
  - Requesters shall use the Wildcard Access Key field to generate request packets and validate response packets.
  - Responders shall use the Wildcard Access Key field to validate request packets and generate response packets.
- If a component supports a single PEER-ATTR, then the Wildcard PEER-ATTR field should be used in place of the PA Table. If used, then:
  - PA Index Field Size shall be hardwired to 0x0.
  - Requesters shall use the Wildcard PA field to generate request packets and validate response packets.
  - Responders shall use the Wildcard PA field to validate request packets and generate response packets.
- If a component supports a single security pointer pair, then the Wildcard Certificate PTR and Wildcard TIK PTR should be used in place of the SEC Table. If used, then:
  - SEC Index Field Size shall be hardwired to 0x0.
  - The Wildcard TIK PTR shall be non-Null and point to a TIK Table entry within the *Component Security Structure*. This TIK Table entry shall contain a single TIK.
  - The Wildcard Certificate PTR shall be non-Null and point to a Certificate Table entry within the *Component Security Structure*. This Certificate Table entry shall contain a single certificate.
  - Requesters shall use the Wildcard TIK PTR and Wildcard Certificate PTR fields to locate the corresponding security information required to generate request packets and validate response packets.
  - Responders shall use the Wildcard TIK PTR and Wildcard Certificate PTR fields to locate the corresponding security information required to validate request packets and generate response packets.
- The PA Index is used to directly index the PA Table and locate the peer component or multicast group's attributes. Each table entry shall contain a *PEER-ATTR Field*.
- Each SSAP, MSAP, MCAP, and MSMCAP table entry contains an AKey field (a single Access Key).
  - The AKey field may contain any Access Key value. A *Component Reset* returns a component to its uninitialized state, i.e., all AKey entries are set to the Default Access Key.
  - When a Requester generates a request packet and Access Key validation is enabled:
    - If an explicit OpClass packet, then the AKey field is copied to the request packet's Access Key field.
    - If generating a non-explicit OpClass packet, then the component shall validate that the AKey field is the Default Access Key and, if not, shall perform component-local error handling and cease processing this request packet.
  - When a Requester or a Responder validates a packet and Access Key validation is enabled:
    - If a packet does not contain an Access Key field and the AKey field's Access Key equals the Default Access Key, then the packet is permitted, else, the component initiates AE error handling.

- If a packet contains an Access Key field and the AKey field equals the packet's Access Key, then the packet is permitted, else the component initiates AE error handling.
- Each SSAP, MSAP, MCAP, and MSMCAP table entry shall contain an ACREQ. The ACREQ field is 2-bit used by Requesters and Requester-Responders to enforce access control on non-Control explicit OpClass packets. The ACREQ field is encoded as follows:
  - 0x0—No Access
    - If a Requester is to transmit a request packet, then the packet shall be handled as a component-local error. If ACREQ is dynamically modified to be No Access and packet transmission scheduling or transmission reached a point where it cannot be stopped, then packet transmission may continue.
  - 0x1—R-Key Required
    - Access permitted, however a Requester shall set RK = 1b and provision the R-Key field in all applicable explicit OpClass request packets when targeting a page configured with a non-Default R-Key (see *Requester PTE*).
  - 0x2—Reserved
  - 0x3—Full Access (Trusted Responder)
    - A Requester shall set RK = 0b in all applicable explicit OpClass request packets irrespective of the R-Key configured for the targeted page, i.e., for select Responders, the Requester may ignore the R-Key configure in the *Requester PTE* R-Key field.
- A component may support Wildcard Requester Access Control (W-ACREQ). If supported, then the W-ACREQ is used to validate access. W-ACREQ uses the same encoding as the ACREQ field in the \*AP tables.
- Each SSAP, MSAP, MCAP, and MSMCAP table entry shall contain an ACRSP. The ACRSP field is 2-bit used by Responders and Requester-Responders to enforce access control on non-Control explicit OpClass packets. The ACRSP is encoded as follows (the same encoding applies to the W-ACRSP field if wildcard semantics are used in place of these tables):
  - 0x0—No Access
    - If a Responder receives a request packet, then the packet shall be handled as a MP.
  - 0x1—R-Key Required
    - Access permitted, however a Responder shall perform R-Key validation on all applicable request packets.
  - 0x2—Reserved
  - 0x3—Full Access (Trusted Requester)
    - A Responder shall ignore the RK bit and shall not perform R-Key validation.
- A component may support Wildcard Responder Access Control (W-ACRSP). If supported, then the W-ACRSP is used to validate access. W-ACREQ uses the same encoding as the ACRSP field in the \*AP tables.
- The SEC index is used to directly index the SEC Table and locate the pointers used to access the security certificate and TIK associated with the peer component or multicast group.
  - Each SEC Table entry contains one or more pointer pairs, where a pointer pair consists of a Certificate PTR and a TIK PTR. If multiple pointer pairs are provisioned per table entry, then these contiguously placed one pair after another, i.e., [Certificate PTR<sub>0</sub>, TIK PTR<sub>0</sub>], [Certificate PTR<sub>1</sub>, TIK PTR<sub>1</sub>], and so forth.

- If additional pointer pairs are provisioned per table entry, then the mechanism to select a specific pair is based on security management mechanisms outside of this specification's scope.
- The Certificate PTR points to a Certificate Table entry. The Certificate Table is located by the Certificate Table Pointer within the *Component Security Structure*.
- The TIK PTR points to a TIK Table entry. The TIK Table is located by the TIK Table Pointer within the *Component Security Structure*.
- See *Security* for additional details on certificate and TIK management.
- The PA Table and SEC Table shall use the respective *PA and SEC Table Formats*.

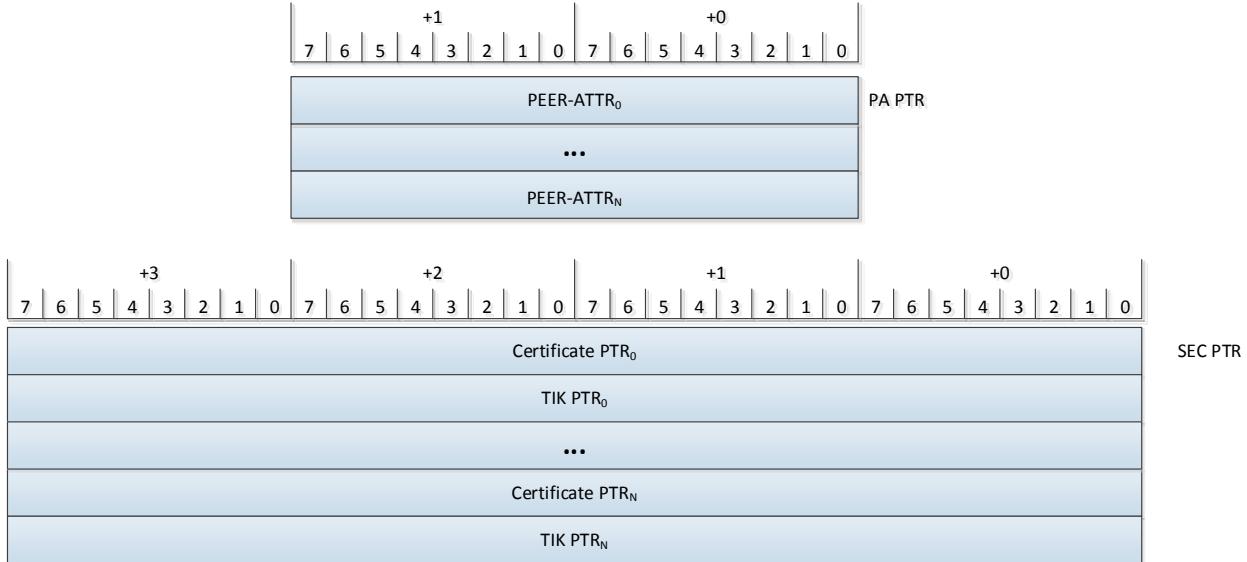


Figure 8-58: PA and SEC Table Formats

- The SSAP, MSAP, MCAP, and MSMCAP table shall use the respective *SSAP, MSAP, MCAP, and MSMCAP Table Formats*.
  - If the PA Table is unsupported, then the PA Index field shall not be provisioned. If so, then an implementation should support the Wildcard PEER-ATTR field.
  - If the SEC Table is unsupported, then SEC Index field shall not be provisioned. If so and security is required, then an implementation should support the Wildcard Certificate and Wildcard TIK fields.

CID 0 >	Pad	SEC Index	ACRSP	ACREQ	AKey	PA Index	SSAP PTR
...							
CID N >	Pad	SEC Index	ACRSP	ACREQ	AKey	PA Index	
MGID 0 >	Pad	SEC Index	ACRSP	ACREQ	AKey	PA Index	MCAP PTR
...							
MGID N >	Pad	SEC Index	ACRSP	ACREQ	AKey	PA Index	
F(CID <sub>k</sub> , SID <sub>k</sub> ) >	Pad	SEC Index	ACRSP	ACREQ	AKey	PA Index	MSAP PTR
...							
F(CID <sub>N</sub> , SID <sub>N</sub> ) >	Pad	SEC Index	ACRSP	ACREQ	AKey	PA Index	
F(GMGID <sub>k</sub> ) >	Pad	SEC Index	ACRSP	ACREQ	AKey	PA Index	MSMCAP PTR
...							
F(GMGID <sub>N</sub> ) >	Pad	SEC Index	ACRSP	ACREQ	AKey	PA Index	

Figure 8-59: SSAP, MSAP, MCAP, and MSMCAP Table Formats

Table 8-101: PEER-ATTR Field

Bits	Access	Description
Bits 2:0	RW	Identifies the <i>OpCode Set Structure</i> (see SET ID field) to use when communicating with this destination component
Bit 3	RW	<p>Latency Domain—Determines if the peer Requester or Responder is in a low-latency domain or a non-low-latency domain.</p> <p>If the Responder is in a low-latency domain, then:</p> <ul style="list-style-type: none"> <li>The <i>Core Structure</i> TO<sub>k</sub> shall be used to determine request packet retransmission. TO<sub>k</sub> shall correspond to the VC used to transmit the request packet.</li> <li>The <i>Core Structure</i> LL Request Deadline shall be used in request packets.</li> <li>The <i>Core Structure</i> LL Response Deadline shall be used in response packets.</li> </ul> <p>If the Responder is in a non-low-latency domain, then:</p> <ul style="list-style-type: none"> <li>The <i>Core Structure</i> LTO<sub>k</sub> shall be used to determine request packet retransmission. LTO<sub>k</sub> shall correspond to the VC used to transmit the request packet.</li> <li>The <i>Core Structure</i> NLL Request Deadline shall be used in request packets.</li> </ul>

Bits	Access	Description
Bit 4		<ul style="list-style-type: none"> <li>The <i>Core Structure</i> NLL Response Deadline shall be used in response packets.</li> </ul> <p>0b—Low-latency Domain 1b—Non-low-latency Domain</p>
Bit 4	RW	<p>Peer Explicit OpClass Next Header Enable—if enabled, all explicit OpClass packets exchanged between this component and the indicated destination component shall contain an <i>Explicit OpClass Next Header Field</i>.</p> <p>If the <i>Core Structure Component CAP 1 Next Header Support</i> field is 0b, then this field shall be hardwired to 0b.</p> <p>If <i>Core Structure Component CAP 1 Control</i> Next Header Enable field is 0b, then shall set Peer Explicit OpClass Next Header Enable = 0b.</p> <p>0b—Disable 1b—Enable</p>
Bit 5	RW	<p>Peer Control OpClass Next Header Enable—if enabled, then all Control OpClass packets exchanged between this component and the indicated destination component shall contain an <i>Explicit OpClass Next Header Field</i>.</p> <p>If the <i>Core Structure Component CAP 1 Next Header Control OpClass Support</i> field is clear, then this field shall be hardwired to 0b.</p> <p>If <i>Core Structure Component CAP 1 Control</i> Next Header Control OpClass Enable field is clear, then shall set Peer Control OpClass Next Header Enable = 0b.</p> <p>0b—Disable 1b—Enable</p>
Bit 6	RW	<p>Peer Precision Time Enable—if enabled, all explicit OpClass packets exchanged between this component and the indicated destination component shall contain an <i>Explicit OpClass Next Header Field</i> with a <i>Precision Time</i> value.</p> <p>If the <i>Core Structure Component CAP 1 Next Header Precision Time Support</i> field is 0b, then this field shall be hardwired to 0b</p> <p>If <i>Core Structure Component CAP 1 Control</i> Next Header Precision Time Enable field is 0b, then shall set Peer Precision Time Enable = 0b.</p> <p>0b—Disable 1b—Enable</p> <p>If Peer HMAC Authentication is enabled, then this field shall be set to 0b.</p>
Bit 7	RW	<p>Peer HMAC Authentication Enable—if enabled, explicit OpClasses enabled for Next header between this component and the indicated destination component shall contain an <i>Explicit OpClass Next Header Field</i> with a dynamically-generated HMAC which is validated upon receipt.</p> <p>If the <i>Component Security Structure Security CAP 1 HMAC Authentication Support</i> == 0b, then this field shall be hardwired to 0b. If <i>Component Security Structure</i></p>

Bits	Access	Description
		<p>Security Control HMAC Authentication Enable == 0b, then this field shall be set to 0b.</p> <p>0b—Disable 1b—Enable</p> <p>If Peer Precision Time is enabled, then shall set Peer HMAC Authentication Enable = 0b.</p>
Bits 9:8	RW	<p>Destination ART—If Peer HMAC Authentication is Enabled, then this field indicates the ART to use when generating or validating a packet. If the destination's HMAC Authentication Enable is Disabled, then this field shall be set to Reserved.</p> <p>0x0—Null ART (default) 0x1—Sequence Number ART 0x2—Precision Time ART 0x3—Reserved</p> <p>If <i>Component Security Structure</i>'s Sequence Number ART Support == 0b or the Sequence Number ART Enable == 0b, then this field shall not be set to Sequence Number ART.</p> <p>If the <i>Component Security Structure</i>'s Precision Time ART Support field == 0b or the Precision Time Number ART Enable field == 0b, then this field shall not be set to Precision Time ART.</p>
10	RW	<p>Write MSG Embedded Read Enable—If enabled, then the component may transmit and receive <i>Write MSG</i> request packets with ER = 1b when communicating with this peer component.</p> <p>0b—Disabled 1b—Enabled</p>
11	RW	<p>Meta Read-Write Enable—If enabled, then the component may include the Meta field in applicable explicit OpClass packets when communicating with this peer component. See <i>Meta Read Response and Meta Write</i></p> <p>0b—Disabled 1b—Enabled</p>
Bits 15:12	-	RsvdP

Table 8-102: SEC Table Entry Fields

Bits	Description
Certificate PTR	Pointer to a Certificate Table entry within the <i>Component Security Structure</i>
TIK PTR	Pointer to a TIK Table entry within the <i>Component Security Structure</i>

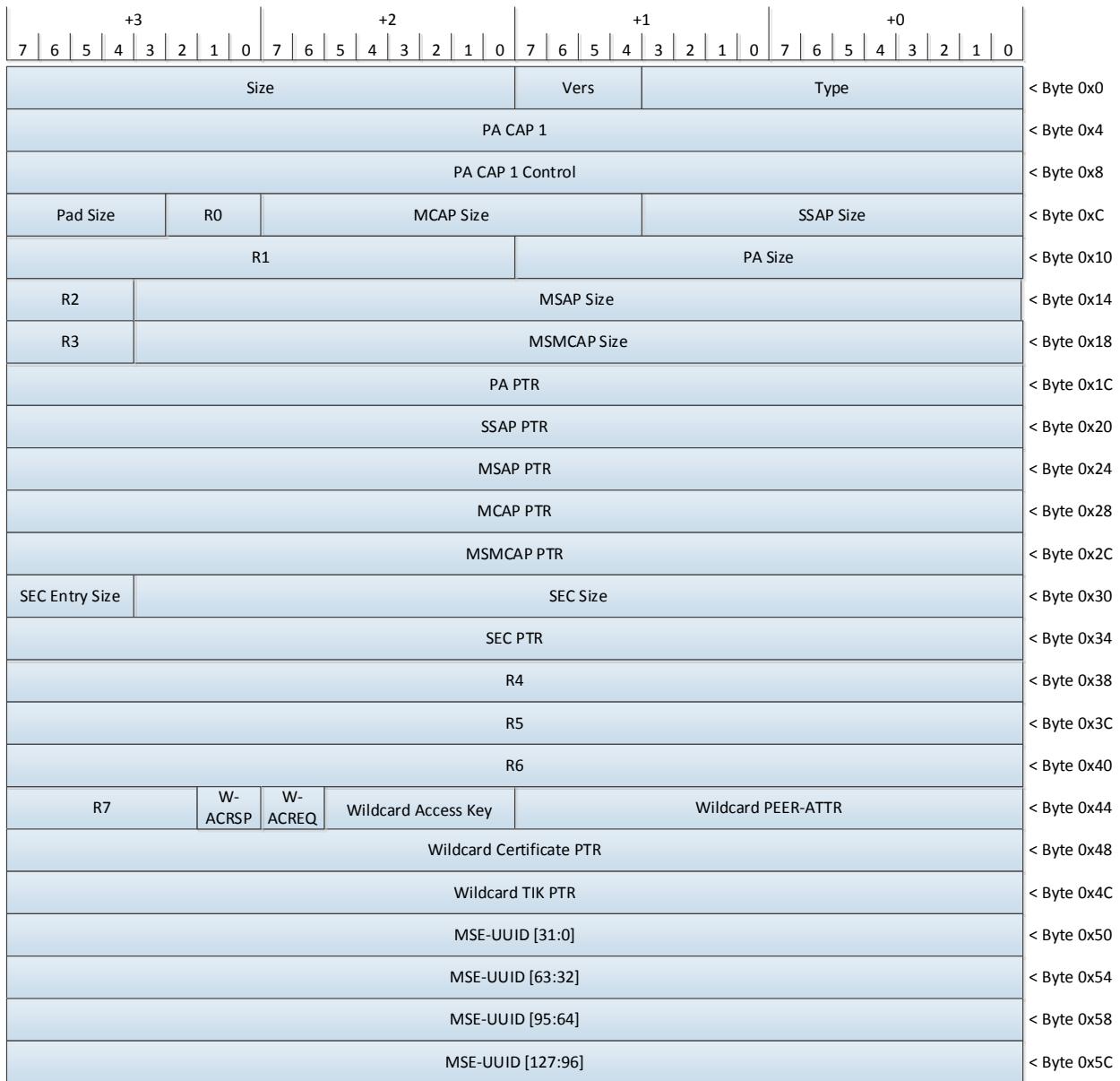


Table 8-103: Component PA Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Version (Vers) PA CAP 1	4	0x1	M	RO	Structure Version
	32	-	M	RO	Peer Attributes Capabilities 1
		Bits 1:0			PA Index Field Size—size of the PA Index field within a supported SSAP Table, MSAP Table, MCAP Table, or MSMCAP Table. If 0x0, then the PA PTR shall be Null.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					<p>If the number PA entries is less than the number of entries supported by a given field size (e.g., if the number of PA entries == 16 and the field size is 8-bits), then the unused upper bits within the field shall be ignored when the field is modified and shall return 0x0 when the field is read.</p> <p>0x0—0 bits 0x1—8 bits 0x2—16 bits 0x3—Reserved</p>
		3:2			<p>PA Entry Size—Size of a PA Table entry in bits (i.e., the size of the PEER-ATTR field).</p> <p>0x0—16 bits 0x1-0x3—Reserved</p>
		Bits 5:4			<p>SEC Index Field Size—size of the SEC Index field within a supported SSAP Table, MSAP Table, MCAP Table, or MSMCAP Table. If 0x0, then the SEC PTR shall be Null.</p> <p>If number SEC entries is less than the number of entries supported by a given field size (e.g., if number SEC entries == 16 and the field size is 8-bits), then the unused upper bits within the field shall be ignored when the field is modified and shall return 0x0 when the field is read.</p> <p>0x0—0 bits 0x1—1 bit 0x2—8 bits 0x3—16 bits</p>
		Bit 6			<p>Wildcard AKey Support—Indicates if the component uses the Wildcard Access Key field instead of the AKey within the SSAP, MSAP, MCAP, and MSMCAP tables to manage Access Keys.</p> <p>If the component does not use the Wildcard Access Key field, then this field shall be hardwired to 0b.</p> <p>0b—AKey 1b—Wildcard</p>
		Bit 7			Wildcard PEER-ATTR Support—Indicates if the component uses the Wildcard PEER-ATTR field instead of the PA Table to manage peer attributes.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
PA CAP 1 Control	32				If the component does not use the Wildcard PEER-ATTR field, then this field shall be hardwired to 0b.  0b—PA Table 1b—Wildcard
					Wildcard SEC Support—Indicates if the component uses the Wildcard Certificate and Wildcard TIK fields instead of the SEC Table to manage security.  0b—SEC Table 1b—Wildcard
					Wildcard ACREQ Support—Indicates if the component uses the W-ACREQ instead of the ACREQ field in the SSAP, MSAP, MCAP, and MSMCAP tables to manage Requester access control.  If the component does not use the W-ACREQ field, then this field shall be hardwired to 0b.  0b—Use ACREQ field in the *AP Tables 1b—Use W-ACREQ
					Wildcard ACRSP Support—Indicates if the component uses the W-ACRSP instead of the ACRSP field in the SSAP, MSAP, MCAP, and MSMCAP tables to manage Responder access control.  If the component does not use the W-ACRSP field, then this field shall be hardwired to 0b.  0b—Use ACRSP in the * AP Tables 1b—Use W-ACRSP
					Reserved
		-	M	RW	
					Access Key Enable—used to enable or disable component Access Key generation and validation  If a component does not support Access Keys validation, then this field shall be hardwired to 0b.  0b—Disabled 1b—Enabled
					RsvdP

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
SSAP Size	12	-	M	RO	Indicates the number of SSAP table entries If SSAP Size == 0x0, then the number of table entries shall be $2^{12}$ .
MCAP Size	12	-	O	RO	Indicates the number of MCAP table entries If MCAP Size == 0x0, then the number of table entries shall be $2^{12}$ .
Pad Size	5	-	M	RO	Indicates the number of pad bits provisioned per SSAP / MCAP / MSAP / MSMCAP row entry to maintain integer 4-byte alignment. Pad bits shall be Reserved.
PA Size	16	-	M	RO	Indicates then number of PA table entries If PA Size == 0x0, then the number of table entries shall be $2^{16}$ . If PA PTR == Null, then this field shall be Reserved and the PA Index field shall not be present in the SSAP, MSAP, MCAP, and MSMCAP Table entries.
MSAP Size	28	-	M	RO	Indicates the number of MSAP table entries If MSAP Size == 0x0, then the number of table entries shall be $2^{28}$ .
MSMCAP Size	28	-	M	RO	Indicates the number of MSMCAP table entries If MSMCAP Size == 0x0, then the number of table entries shall be $2^{28}$ .
PA PTR	32	-	M	RO	Pointer to the PA Table. If Wildcard PEER-ATTR Support == 1b, then this field shall be Null.
SSAP PTR	32	-	M	RO	Pointer to the SSAP table. If SSAP PTR == Null, then SSAP Size shall be Reserved.
MSAP PTR	32	-	O	RO	Pointer to the MSAP table. If multi-subnet unicast packet exchange is unsupported, then this field shall be Null. If MSAP PTR == Null, then MSAP Size shall be Reserved.
MCAP PTR	32	-	M	RO	Pointer to the MCAP table. If single-subnet multicast packet exchange is unsupported, then this field shall be Null. If MCAP PTR == Null, then MCAP Size shall be Reserved.
MSMCAP PTR	32	-	O	RO	Pointer to the MSMCAP table. If multi-subnet multicast packet exchange is unsupported, then

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
SEC Size					this field shall be Null. If MSMCAP PTR == Null, then MSMCAP Size shall be Reserved.
SEC Entry Size	28	-	O	RO	Indicates the number of SEC table entries If SEC Size == 0x0, then the number of table entries shall be $2^{28}$ .
SEC PTR	4	-	O	RO	Indicates the number of pointer pairs per SEC table entry. 0x0—1 0x1—2 0x2—4 0x3—8 0x4—16 0x5-0xF—Reserved
PA Index	32	-	O	RO	Pointer to the Security table
AKey	-	-	M	RW	Index into the PA table.
ACREQ	6	-	M	RW	Access Key used to generate and validate unicast and multicast packets.
ACRSP	2	-	M	RW	Requester Access Control—see previous discussion for details. 0x0—No Access 0x1—R-Key Enabled 0x2—Reserved 0x3—Full Access (Trusted Responder)
SEC Index	2	-	M	RW	Responder Access Control—see previous discussion for details. 0x0—No Access 0x1—R-Key Enabled 0x2—Reserved 0x3—Full Access (Trusted Requester)
Wildcard PEER-ATTR	-	-	O	RW	Index into the SEC table.
Wildcard Access Key	16	-	O	RW	If the component supports Wildcard PEER-ATTR functionality, then this field may be configured with the peer attributes (see <i>PEER-ATTR Field</i> ) to use to generate and validate packets.
	6	-	O	RW	If the component supports Wildcard AKey functionality, then this field may be configured with the Access Key to use to generate and validate packets.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
W-ACREQ	2	-	M	RW	<p>Wildcard Requester Access Control—see previous discussion for details.</p> <p>0x0—No Access 0x1—R-Key Enabled 0x2—Reserved 0x3—Full Access (Trusted Responder)</p>
W-ACRSP	2	-	M	RW	<p>Wildcard Responder Access Control—see previous discussion for details.</p> <p>0x0—No Access 0x1—R-Key Enabled 0x2—Reserved 0x3—Full Access (Trusted Requester)</p>
Wildcard Certificate PTR	32	-	O	RO	If non-Null, then this points to the Certificate Table entry to use to generate and validate packets that require HMAC authentication.
Wildcard TIK PTR	32	-	O	RO	If non-Null, then this points to the TIK Table entry to use to generate and validate packets that require HMAC authentication.
MSE-UUID	128	-	O	RO	<p>If the component supports multi-subnet unicast communications, then this UUID identifies the encoding method used to access the MSAP.</p> <p>Components that support multi-subnet multicast communications shall use the MCE-UUID within the <i>Component Multicast Structure</i> to identify the encoding method used to access the MSMCAP.</p>
R0	3	-	-	-	Reserved
R1	16	-	-	-	Reserved
R2	4	-	-	-	Reserved
R3	4	-	-	-	Reserved
R4	32	-	-	-	Reserved
R5	32	-	-	-	Reserved
R6	32	-	-	-	Reserved
R7	6	-	-	-	RsvdP

## 8.36. Component Event Structure

The Component Event structure is used to configure how a Responder signals management upon receipt of an *Unsolicited Event (UE) Packet*.

The following are the features and requirements of the Component Event structure:

- 5 • Any component type may support the Component Event structure. To support this structure, the component shall support the *Unsolicited Event (UE) Packet* as a Responder or as a Requester-Responder. Further, shall set *Core Structure Component CAP 1 Management Services Support* = 1b to indicate that a manager capable of executing an *Unsolicited Event (UE) Packet* may be co-located at this component.
- 10 • The Component Event structure shall use the *Component Event Structure Format*.
- 15 • The Event Signal field is composed of 256 2-bit sub-fields. Each sub-field indicates one of four actions to take upon receipt of an *Unsolicited Event (UE) Packet*: do not signal management or trigger one of three possible component-local interrupts. A solution may support multiple component-local interrupts to enable different management entities, e.g., a subset of events could be handled by firmware, a subset by an operating system, and a subset by a higher-level manager.
  - o Coordination of event management that spans multiple management entities, i.e., multiple signal targets, is outside of this specification's scope.
- 20 • For each configured Interrupt Address, management shall allocate an Event Record circular queue that contains Max Event Records entries. Each Event Record entry shall use the *Event Record Entry Format*. Upon receipt of an *Unsolicited Event (UE) Packet*, the component selects the next available Event Record in monotonically-increasing order (modulo the maximum number of event records), and copies the relevant fields from the UE packet to the Event Record.
  - o If no Event Records are available, then the component shall silently discard the *Unsolicited Event (UE) Packet* (management relies upon packet retransmission to compensate for transient resource shortages).
  - o If Event Control indicates the interrupt service routine associated with the corresponding component-local interrupt has exited, then the component shall generate that interrupt.
  - o Upon component initialization or reset, the component shall target entry 0x0 to store the first Event Record associated with a given Event Record queue.
- 25
- 30

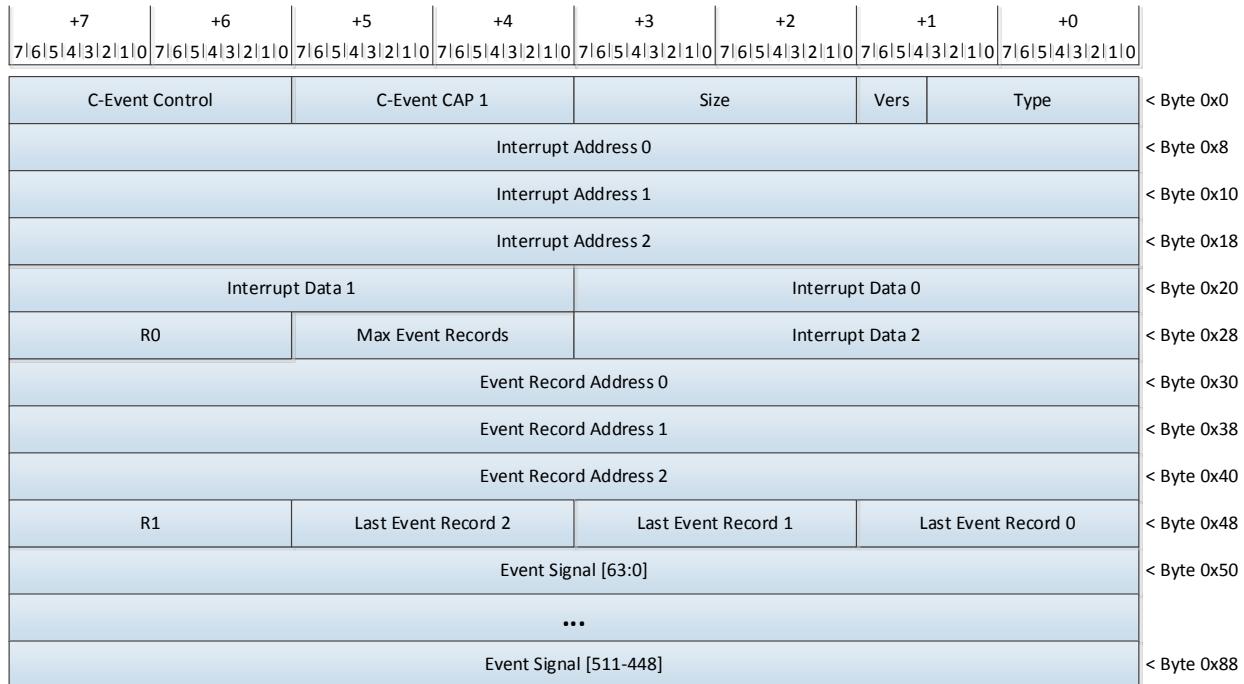


Figure 8-61: Component Event Structure Format

Table 8-104: Component Event Structure Fields

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description	
Version (Vers)	4	0x1	M	RO	Structure Version	
C-Event CAP 1	16	-	M	RO	Capabilities associated with the Component Event structure.	
		Bits 2:0			Event Signal Size—Indicates the size of the Event Signal field 0x0—512 bits 0x1-0x7—Reserved	
		Bits 15:3			Reserved	
C-Event Control	16	-	M	-	Component Event structure control fields.	
		Bit 0		RW	Interrupt 0 Enable—Determines if the Interrupt Address 0 and Interrupt Data 0 fields are configured and enabled for use. 0b—Disabled 1b—Enabled	
					Interrupt 1 Enable—Determines if the Interrupt Address 1 and Interrupt Data 1 fields are configured and enabled for use. 0b—Disabled	
		Bit 1		RW		

<b>Interrupt Address 0</b>  <b>Interrupt Address 1</b>  <b>Interrupt Address 2</b>  <b>Interrupt Data 0</b>  <b>Interrupt Data 1</b>  <b>Interrupt Data 2</b>				1b—Enabled
				Bit 2
				RW
				Interrupt 2 Enable—Determines if the Interrupt Address 2 and Interrupt Data 2 fields are configured and enabled for use.  0b—Disabled 1b—Enabled
				Bit 3
				RW
				Management writes 1b to this bit to indicate that it has updated the Last Event Record 0, and that the interrupt service routine associated with Interrupt Address 0 has exited.  Reads of this bit shall return 0b.
				Bit 4
				RW
				Management writes 1b to this bit to indicate that it has updated the Last Event Record 1, and that the interrupt service routine associated with Interrupt Address 1 has exited.  Reads of this bit shall return 0b.
				Bit 5
				RW
				Management writes 1b to this bit to indicate that it has updated the Last Event Record 2, and that the interrupt service routine associated with Interrupt Address 2 has exited.  Reads of this bit shall return 0b.
				Bits 15:6
				-
				RsvdP
	64	-	O	RW
	64	-	O	RW
	64	-	O	RW
	32	-	O	RW
	32	-	O	RW
	32	-	O	RW

					Contents of this field are component-specific
<b>Max Event Records</b>	16	-	M	RW	Indicates the maximum number of event records associated with a given Event Record circular queue.  If Max Event Records == 0x0, then the queue shall contain $2^{16}$ Event Records.
<b>Event Record Address 0</b>	64	-	M	RW	This is the component-local address of the circular queue where the component writes Event Records that are associated with Interrupt 0.
<b>Event Record Address 1</b>	64	-	M	RW	This is the component-local address of the circular queue where the component writes Event Records that are associated with Interrupt 1.
<b>Event Record Address 2</b>	64	-	M	RW	This is the component-local address of the circular queue where the component writes Event Records that are associated with Interrupt 2.
<b>Last Event Record 0</b>	16	-	M	RW	Indicates the index of the last Event Record associated with the Event Record Address 0 queue that was processed by management.
<b>Last Event Record 1</b>	16	-	M	RW	Indicates the index of the last Event Record associated with the Event Record Address 1 queue that was processed by management.
<b>Last Event Record 2</b>	16	-	M	RW	Indicates the index of the last Event Record associated with the Event Record Address 2 queue that was processed by management.
<b>Event Signal</b>	512	-	M	RW	Each Event Signal 2-bit sub-field indicates how the associated event is to be signaled. If configured, the component shall generate the indicated component-local interrupt.  0x0—No signal action is taken 0x1—Trigger component-local interrupt 0 0x2—Trigger component-local interrupt 1 0x3—Trigger component-local interrupt 2  Sub-field 0 is located in byte 0, bit 0. Sub-field 1 is contiguous to sub-field 0. And so forth.
<b>R0</b>	16	-	-	-	RsvdP
<b>R1</b>	16	-	-	-	RsvdP

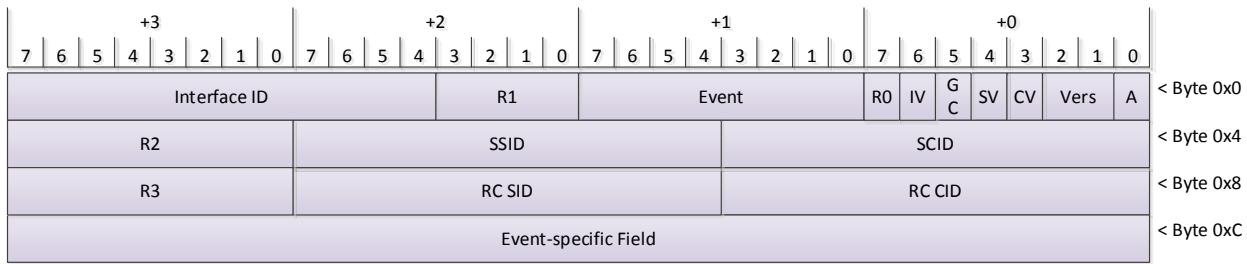


Figure 8-62: Event Record Entry Format

Table 8-105: Event Record Fields

Field Name	Size (bits)	Access	Description
<b>A</b>	1	RW	Indicates if the record is available to be overwritten by the component with a new <i>Unsolicited Event (UE) Packet</i> . 0b—Available 1b—Unavailable
<b>Vers</b>	2	RW	Version of this Event Record entry format. For this version of the specification, Vers shall be set to 0x1.
<b>CV</b>	1	RW	If 1b, then this indicates the RC CID field contains a valid CID.
<b>SV</b>	1	RW	If 1b, then this indicates the RC SID field contains a valid SID.
<b>GC</b>	1	RW	If 1b, then this indicates the SSID field contains a valid SID.
<b>IV</b>	1	RW	If 1b, then this indicates the Interface ID field contains a valid Interface ID.
<b>Event</b>	8	RW	Copy of the <i>Unsolicited Event (UE) Packet</i> 's Event field.
<b>Interface ID</b>	12	RW	Copy of the <i>Unsolicited Event (UE) Packet</i> 's Interface ID field.
<b>SCID</b>	12	RW	Copy of the <i>Unsolicited Event (UE) Packet</i> 's SCID field.
<b>SSID</b>	16	RW	Copy of the <i>Unsolicited Event (UE) Packet</i> 's SSID field (if applicable).
<b>RC CID</b>	12	RW	Copy of the <i>Unsolicited Event (UE) Packet</i> 's RC CID field (if applicable).
<b>RC SID</b>	16	RW	Copy of the <i>Unsolicited Event (UE) Packet</i> 's RC SID field (if applicable)
<b>Event-specific Field</b>	32	RW	Copy of the <i>Unsolicited Event (UE) Packet</i> 's Event-specific Field.

## 8.37. Component SOD Structure

The Component SOD structure by components that support *Strong Ordering Domain (SOD)* to manage SODs between components. This structure is used to configure two tables that share a common row layout.

- 5 The SSOD Table is used to configure individual SODs that exchange packets within a directly-attached subnet.
- 10 The MSOD Table is used to configure individual SODs that exchange packets in a multi-subnet topology. Due to the potentially large number of components within a multi-subnet topology, the mechanisms to index the MSOD is outside of this specification's scope. Instead, a component-specific function is used to locate the appropriate MSOD Table index. For example, by organizing components into groups and applying intelligent [CID, SID], a subset of bits from each identifier can be used to construct the corresponding index.
- 15 Each table row is composed of up to SODE entries. Each table row is directly-indexed using the SOD's SODID.

15 The following are the features and requirements of the Component SOD structure:

- Any component type may support the Component SOD structure.
- The Component SOD structure shall use the *Component SOD Structure Format*.
- Each SSOD / MSOD Table row is composed of the following fields:
  - o V—this bit determines if the SSODID entry is valid. This field shall not be set to 1b until the PRI EI and DSODID fields have been configured.
    - Software should set V = 1b in the Responder prior to setting V in the Requester. If V ≠ 1b, then the Responder shall silently discard request packets that target this SODID.
    - If both components are Requester-Responders, then software may set V = 1b in either component first.
  - o ARM—if the interface supports SOD interface migration, then this determines if the SOD can be migrated and which interface to target.
    - Once SOD interface migration is triggered, then SOD interface migration for this SOD shall be disabled. To enable SOD interface migration, the ARM field shall be modified to reflect the new target interface.
  - o PRI EI—Indicates the primary interface to use upon SOD initialization
  - o SEC EI—Indicates the alternative interface to use if SOD interface migration is supported
  - o DSODID—Contains the Destination SOD Identifier associated with this SSODID. Within each SSOD / MSOD Table row, each DSODID entry shall contain a unique DSODID value.

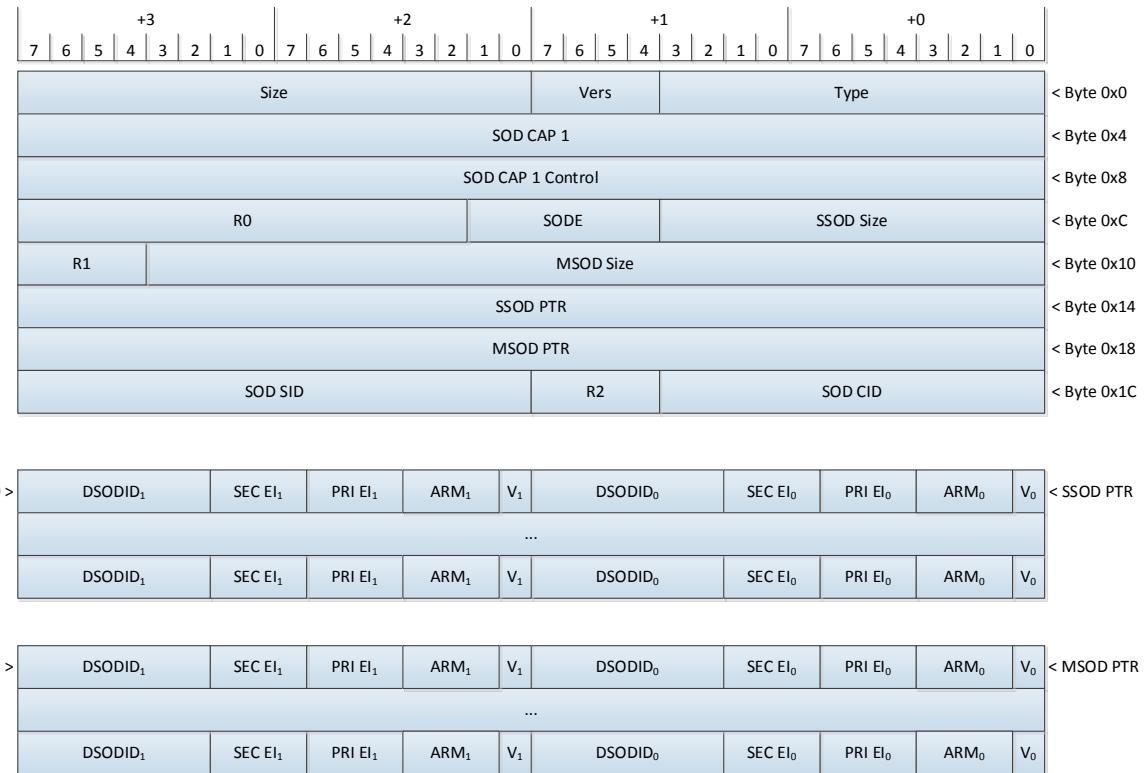


Figure 8-63: Component SOD Structure Format

Table 8-106: Component SOD Structure Fields

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description
<b>Version (Vers)</b>	4	0x1	M	RO	Structure Version
<b>SOD CAP 1</b>	32	-	M	RO	SOD Capabilities
		Bit 0			Multi-subnet Support—Indicates if the component supports multi-subnet communications. 0b—Unsupported 1b—Supported
		Bit 1			Interface Migration Support—Indicates if the component supports SOD migration between component interfaces 0b—Unsupported 1b—Supported
		Bits 31:2			Reserved
<b>SOD CAP 1 Control</b>	32	-	M	RW	SOD Capability Control
		Bit 0			Enable SOD Communications—Controls SOD packet exchange

<b>Component</b>  <b>Table 1</b>  <b>Component Structure</b>	<b>Bit 1</b>	<b>Bits 31:2</b>	<b>0b—Disabled</b> <b>1b—Enabled</b>		<b>SOD CID-SID Configured</b> —Used to indicate if the SOD CID and SOD SID fields are configured.  0x0—Not configured 0x1—SOD CID Configured 0x2—SOD CID and SOD SID Configured 0x3—Reserved	
			<b>RsvdP</b>			
	<b>SSOD Size</b>	<b>12</b>	<b>-</b>	<b>M</b>	<b>RO</b>	Number of SSOD Table entries  If SSOD Size == 0x0, then the number of entries shall be $2^{12}$ .
	<b>SODE</b>	<b>6</b>	<b>-</b>	<b>O</b>	<b>RO</b>	Indicates the number of row entries per SOD / MSOD Table entry  If SODE == 0x0, then the number of entries shall be $2^7$ .
		<b>SSOD PTR</b>	<b>32</b>	<b>-</b>	<b>M</b>	<b>RO</b>
	<b>MSOD PTR</b>	<b>32</b>	<b>-</b>	<b>M</b>	<b>RO</b>	Pointer to the multi-subnet SOD table. If the component does not support multi-subnet topologies, then this field shall be Reserved.
		<b>SOD CID</b>	<b>12</b>	<b>-</b>	<b>M</b>	<b>RW</b>
	<b>SOD SID</b>	<b>16</b>	<b>-</b>	<b>O</b>	<b>RW</b>	If a component supports multi-subnet communications, then this field shall contain the SID to use to exchange all multi-subnet SOD packets. The SOD SID may be any configured <i>Core Structure</i> SID n.  If the component does not support multi-subnet topologies, then this field shall be Reserved.
		<b>MSOD Size</b>	<b>28</b>	<b>-</b>	<b>M</b>	<b>RO</b>
	<b>V</b>	<b>1</b>	<b>-</b>	<b>M</b>	<b>RW</b>	Determines if a row entry contains a valid PRI EI, SEC EI, and DSODID. If Valid, then the corresponding SOD [DCID   GDCID, SSODID, DSODID] may be used to exchange SOD request and response packets.

					If V == Valid and software sets this field to Invalid, then the SOD shall be disabled from further communications. Software shall disable the SOD at both components prior to reinitializing the SOD.  0b—Invalid 1b—Valid
ARM	3	-	M	RW	Determines if interface migration is enabled and that the target interface is to migrate the SOD to upon receipt of a SOD packet with the Migrate Interface (MI) == 1b.  Software may trigger migration from the PRI EI to the SEC EI at its discretion, e.g., to initiate planned downtime.  0x0—Unarmed 0x1—Armed, Target SEC EI 0x2—Armed, Target PRI EI 0x3—Trigger Migration, Target SEC EI 0x4—Trigger Migration, Target Pri EI 0x5-0x7—Reserved
PRI EI	3	-	M	RW	Component interface used to exchange SOD packets upon SOD initialization.  SOD packets may be exchanged on any supported interface [0-7].
SEC EI	3	-	M	RW	Alternative component interface used to exchange SOD packets.  SOD packets may be exchanged on any supported interface [0-7].
DSODID	6	-	M	RW	The DSODID at the peer component associated with this SSODID table entry
R0	14	-	-	-	Reserved
R1	4	-	-	-	RsvdP
R2	4	-	-	-	RsvdP

## 8.38. Component ATP Structure

The Component ATP structure is used to manage *Address Translation and Page Services*.

The following are the features and requirements of the Component ATP structure:

- 5
- Any component type may support the Component ATP structure.
  - The Component ATP structure shall use the *Component ATP Structure Format*.

- The Component ATP structures shall apply to only *Address Translation and Page Services* packets.
- The Component ATP structure is used to manage *Address Translation and Page Services* at the component level. A component may support per context management using mechanisms outside of this specification's scope, e.g., if a *Logical PCI Device (LPD)*, then through the Address Translation Services capabilities and control registers as specified in the *PCI Express Base Specification*. Per context management may impose additional constraints but shall not provide additional capabilities beyond what is configured through this structure.
- Address translation services translate and invalidate address translations in STUs. STU size is calculated as  $STU = 2^{STUE} * 4096$  bytes. The minimum  $STU = 4096$  ( $STUE = 0x0$ ). The maximum  $STU = 8$  TiB ( $STUE == 0x1F$ ).

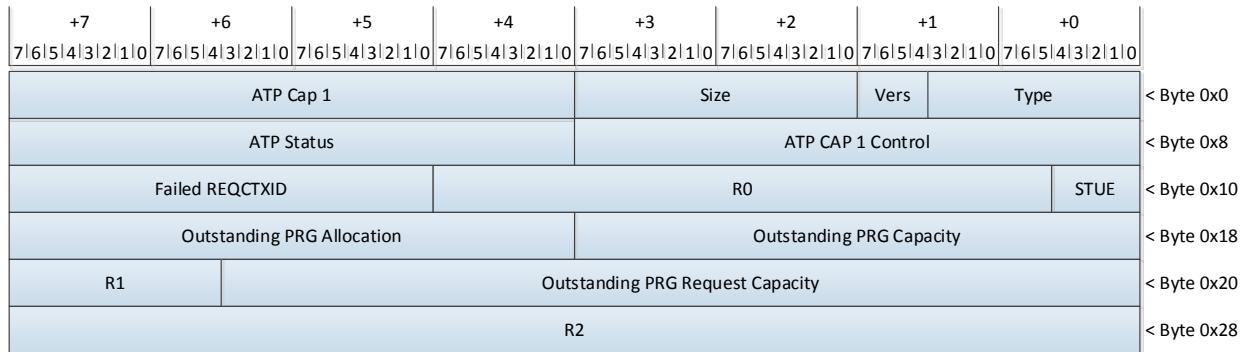


Figure 8-64: Component ATP Structure Format

Table 8-107: Component ATP Structure Fields

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description
Version (Vers)	4	0x1	M	RO	Structure Version
ATP CAP 1	32	-	M	RO	ATP Capabilities 1
		Bit 0			PASID Support—Indicates if the component supports placing a PASID value in the PASID field in applicable packets. If PASID is supported, then a component shall support the entire 20-bit PASID space. If unsupported, then the PASID field shall be Reserved and the PV field shall be set to 0b. 0b—PASID Reserved 1b—PASID Value
		Bit 1			PRG RSPN PASID Required—If a component supports <i>Page Services</i> , then this field indicates if a <i>PRG Response Notification</i> packet is required to contain a valid PASID value. If PRG RSPN PASID Required == 0b, then the component shall ignore the PASID field in any

					<p><i>PRG Response Notification</i> packet. Further, PRG Index values shall be shared across all component PASIDs.</p> <p>If PRG RSPN PASID Required == 1b, then the component shall copy the <i>PRG Request</i> packet's PASID field to the <i>PRG Response Notification</i> packet's PASID field.</p> <p>0b—PASID Ignored 1b—PASID Required</p> <p>This field shall apply only to a non-<i>Logical PCI Device (LPD)</i> or a component that does not support per context management.</p> <p>A <i>Logical PCI Device (LPD)</i> uses the Page Request Status register as defined in the <i>PCI Express Base specification</i>.</p>
	Bit 2				<p>Address Translation Services Support—Indicates if the component supports <i>Address Translation Services</i>.</p> <p>0b—Unsupported 1b—Supported</p>
	Bit 3				<p>Page Services Support—Indicates if the component supports <i>Page Services</i>.</p> <p>0b—Unsupported 1b—Supported</p>
	Bit 4				<p>Context Management Support—Indicates if the component supports per context management (e.g., a LPD supports per context management).</p> <p>0b—Unsupported 1b—Supported</p>
	Bit 5				<p>Privileged Mode Support—Indicates if the component supports Privileged Mode operations. Though associated with solution protection models, the specific meaning of Privileged Mode and Non-Privileged Mode are outside of this specification's scope.</p> <p>0b—Unsupported 1b—Supported</p>
	Bit 6				<p>Execution Permission Support—Indicates if the component supports applicable request packets with the Execute bit set to 1b.</p> <p>0b—Unsupported 1b—Supported</p>

ATP CAP 1 Control		Bit 7			Global Mapping Support—Indicates if a component may create a single mapping entry within a given context's cache that is applicable to all supported PASID values. 0b—Unsupported 1b—Supported
		Bit 8			Global Invalidate Support—Indicates if a component supports <i>Translation Invalidate Request</i> packets within the Global Invalidate field set to 1b. If unsupported, then the component shall silently ignore the Global Invalidate field in all <i>Translation Invalidate Request</i> packets. 0b—Unsupported 1b—Supported
		Bits 31:10	Reserved		
		32	-	M	RW
	Bit 0				
		PASID Enable—Determines if the component may use the PASID field in applicable packets (the PV field determines if the PASID field contains a PASID value). If disabled, then the PASID field shall be Reserved and PV = 0b. 0b—Disabled 1b—Enabled			
		Bit 1		Address Translation Services Enable—Determines if the component may exchange address translation packets, e.g., <i>Translation Request</i> packets. 0b—Disabled 1b—Enabled	
		Bit 2		Page Services Enable—Determines if the component may exchange page services packets, e.g., <i>PRG Request</i> packets. 0b—Disabled 1b—Enabled	
		Bit 3		Reset PRI—If Page Services Enable == 0b and Reset PRI == 1b, then the PRI shall be reset, i.e., all implementation-specific PRI tracking logic and resources shall be reset to the uninitialized state. Reads of this bit shall return 0b.	

	Bit 4			Context Management Override—If context management is supported, then this determines if it is enabled to override settings within this structure. For example, if an unmodified OS is used and the component supports LPDs, then management may elect to have the OS manage address translation and page services.  0b—Disabled 1b—Enabled
	Bit 5			Privileged Mode Enable—If Privileged Mode is supported, then this determines if a component is enabled to act upon or set the Priv field in applicable packets.  0b—Disabled 1b—Enabled
	Bit 6			Execute Permission Enable—If Execute Permission is supported, then this determines if a component is enabled to act upon or set the Execute field in applicable packets.  0b—Disabled 1b—Enabled
	Bit 7			Global Mapping Enable—If Global Mapping is supported, then this determines if the component is enabled to create a single mapping entry within a given context's cache that is applicable to all supported PASID values.  0b—Disabled 1b—Enabled
	Bit 8			Global Invalidate Enable—If Global Invalidate is supported, then this determines if the component is enabled to validate and act upon the Global Invalidate field in <i>Invalidate Request</i> packets. If unsupported or disabled, then the component shall silently ignore the Global Invalidate field in all <i>Translation Invalidate</i> Request packets.  0b—Disabled 1b—Enabled
	Bit 9			Address Translation Cache Enable—Used to enable the component to cache address translations returned in a <i>Translation Response</i> packet.  0b—May Not Cache

ATP Status	32	Bit 10	Bits 15:11	Bits 31:11	RsvdP				
		1b—May Cache	This field shall apply only to a non- <i>Logical PCI Device (LPD)</i> ; a <i>Logical PCI Device (LPD)</i> uses the ATS Control register as defined in the <i>PCI Express Base Specification</i> .						
		Max Context ID—Determines the maximum REQCTXID value.	If a component supports <i>Logical PCI Device (LPD)</i> , then shall set Max Context ID = 0b.						
		0b— $2^{16} - 1$	1b— $2^{24} - 1$						
		Smallest Translation Unit (STU)—Used to calculate memory region size communicated in a <i>Translation Response packet</i> or a <i>Translation Invalidate Request packet</i> . For example, a solution that supports 4096-byte address translation will set STU = 0x0.	Memory region size = $2^{\text{STU}} * 4096$ bytes						
If the component does not support <i>Address Translation</i> , then this field shall be Reserved.									
This field shall apply only to a non- <i>Logical PCI Device (LPD)</i> ; a <i>Logical PCI Device (LPD)</i> uses the ATS Control register as defined in the <i>PCI Express Base Specification</i> .									
RsvdP									
-		M	RW1C	Status Register					
Bit 0		<i>PRG Response Notification</i> Failure—Indicates that the component received a <i>PRG Response Notification</i> packet indicating Response Failure. The component shall ignore all subsequent <i>PRG Response Notification</i> packets that target the indicated REQCTXID. Failed REQCTXID shall contain the last context to encounter this condition.							
Bit 1		<i>Unexpected PRG Index</i> —Indicates receipt of a <i>PRG Response Notification</i> packet that contains a PRG Index that does not match any outstanding <i>PRG Request</i> packet. Failed REQCTXID shall contain the last context to encounter this condition.							

					If Unexpected PRG Index == 0b, then Failed REQCTXID does not contain a valid context identifier.
					Stopped—When Page Services Enable transitions from 1b to 0b, then shall set Stopped = 0b. Once the component has stopped transmitting new <i>PRG Request</i> packets and all outstanding <i>PRG Request</i> packets have been completed, then shall set Stopped = 1b. If Page Services Enable == 1b, then this field is undefined.
					RsvdZ
<b>STUE</b>	5	-	M	RW	Smallest Translation Unit Exponent (STUE) is used to calculate the Smallest Translation Unit (STU).  $STU = 2^{STUE} * 4096 \text{ bytes}$  Software shall set ATP CAP 1 Control Address Translation Services Enable = 0b prior to modifying this field.
<b>Failed REQCTXID</b>	24	-	M	RO	Last REQCTXID that suffered a failure condition. This field is updated only upon detection of a context-specific failure or error condition as reflected in the ATP Status field. This field is non-deterministic until a failure or error condition is detected.
<b>Outstanding PRG Capacity</b>	32	-	O	RO	Indicates the maximum number of pages that a component can request to be resident at any given time.
<b>Outstanding PRG Allocation</b>	32	-	O	RW	If a Requester, then this indicates the maximum number of outstanding <i>PRG Request</i> packets. If a component exceeds this value, then its PRI may be disabled.
<b>Outstanding PRG Request Capacity</b>	52	-	O	RO	If a Responder, then this indicates the maximum number of outstanding <i>PRG Request</i> packets that it can receive.
<b>R0</b>	35	-	-	-	RsvdP
<b>R1</b>	12	-	-	-	RsvdP
<b>R2</b>	64	-	-	-	Reserved

## 8.39. Congestion Management Structure

If a Requester supports explicit OpClass packets, then the Congestion Management structure may be used to configure Requester congestion management behavior.

5 The following are the features and requirements of the Congestion Management structure:

- Only a Requester may support the Congestion Management structure. If a Requester supports only non-explicit OpClass packets, then it shall not support this structure.
- The Congestion Management structure shall use the *Congestion Management Structure Format*.
- If Strict Increment Mode == 0b, then a Requester may apply a vendor-defined amount to adjust the PIDT index used to control packet injection rate, e.g., the amount may vary by congestion event type. If set to 1b, then a Requester shall adjust only the PIDT index used by one irrespective of the congestion event type.
- Once congestion events stop being received, the Requester may decrement the PIDT index to increase the packet injection rate until a congestion events are received. Once received, the Requester increments the PIDT index. At this point the packet injection should be stabilize and the PIDT index will oscillate between two indices. The Requester may decrement the PIDT index only once per Congestion Sampling Window.
- A Requester may support congestion management on a component-wide basis.
- A Requester may support congestion management using a vendor-defined basis controlled through a *Vendor-Defined Structure* accessed through the structure's Vendor-defined PTR.
- A Requester may support more fine-grain congestion management based on a resource type or group, e.g., per interface, per [interface, VC], per protocol engine, etc. If supported, then:
  - The component shall provision a Resource Array located through the Resource Array PTR.
  - The Resource Array shall support a single Resource Type.
    - If Resource Type equals Interface ID, then each table entry shall correspond to a single Interface ID in monotonically-increasing order, i.e., table entry 0 corresponds to interface 0, table entry 1 corresponds to interface 1, and so forth.
      - If an interface supports only non-explicit OpClass packets, then congestion management shall not be applied and the R-Min Index shall be hardwired to 0x0.
    - If Resource Type equals Interface ID and VC, then there shall be one table entry for each provisioned interface and each supported VC. Each table entry shall correspond to a single [Interface ID, VC] in monotonically-increasing order, i.e., table entry 0 corresponds to [interface 0, VC0], table entry 1 corresponds to [interface 0, VC 1], and so forth.
    - If Resource Type equals Protocol Engine ID, then each table entry shall correspond to a single protocol engine in monotonically-increasing order, i.e., table entry 0 corresponds to protocol engine 0, table entry 1 corresponds to protocol engine 1, and so forth.
    - If Resource Type equals Resource Group ID, then each table entry shall correspond to a single Resource Group ID in monotonically-increasing order, i.e., table entry 0 corresponds to Resource Group ID 0, table entry 1 corresponds to Resource Group ID 1, and so forth.
  - The R-Min Index in each table entry shall independently managed. The component shall be responsible for dynamically adjusting an implementation-specific index associated with

resource table entry. This implementation-specific index shall always be greater than or equal to its corresponding R-Min Index.

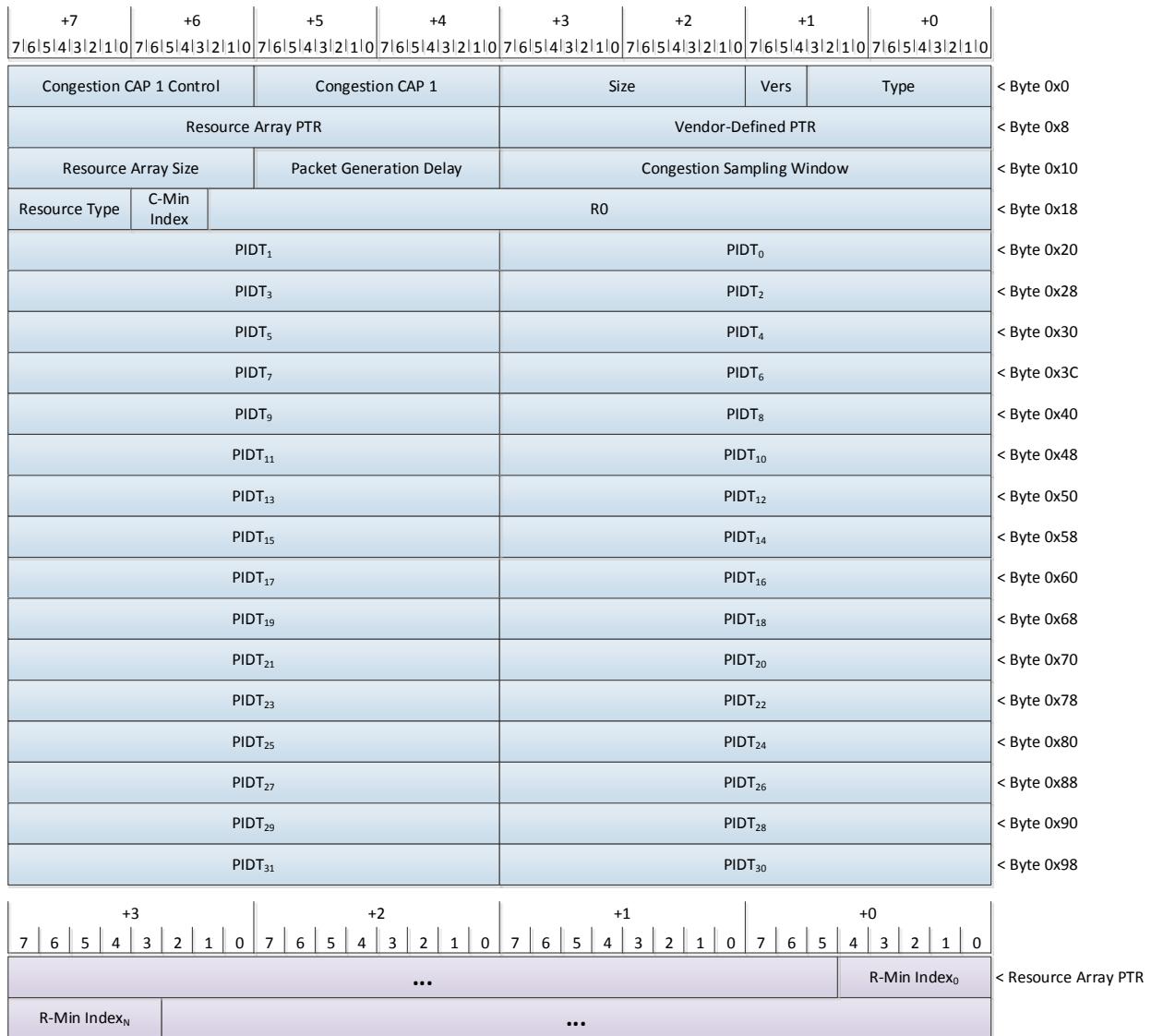


Figure 8-65: Congestion Management Structure Format

Table 8-108: Congestion Management Structure Fields

Field Name	Size (bits)	Value / Bit Location	M O	Access	Description
Version (Vers)	4	0x1	M	RO	Structure Version
Congestion CAP 1	16	-	-	RO	Congestion Management Capabilities
		Bit 0			Strict Increment Mode—This bit indicates whether to use a vendor-defined mechanism to adjust which index within the Packet Injection Delay Table (PIDT) to use or to

<b>Congestion CAP 1 Control</b>					adjust the current index by 1 value when a congestion event is detected. See <i>Congestion Management</i> .
					0b—Vendor-defined 1b—Strict By 1 Adjustment
					Reserved
					Congestion Management Capability Control
	16	-	RW		Congestion Management Control—By default, congestion management shall be performed at the component-level.
					If a component supports Resource-based congestion management, then congestion management may be performed at the component-level or at the resource level.
					If a component supports Vendor-defined-based congestion management, then congestion management may be performed at the component-level or at a vendor-defined level.
					0x0—Component 0x1—Resource 0x2—Vendor-defined 0x3-0x7—Reserved
					Reserved
	32	-	M	RW	This field is configured with the number of ns a component shall wait before adjusting the PIDT index used to control packet injection rate. The field should be set to at least the minimum round-trip latency between a Requester and any peer Responder.
	16	-	M	RO	This field indicates the number of ps required to generate and transmit a 16-byte explicit OpClass packet on any component interface.
	32	-	O	RO	If non-Null, then the component supports component-level and vendor-defined level congestion management. This field shall point to a <i>Vendor-Defined Structure</i> .
	32	-	O	RO	If non-Null, then the component supports component-level and resource-level congestion management. This field shall point to a Resource Array.

<b>Resource Array Size</b>	16	-	O	RO	<p>Indicates the number of Resource Array table entries.</p> <p>If Resource Array Size == 0x0, then the number of entries shall be <math>2^{16}</math>.</p> <p>If Resource Array PTR == Null, then this field shall be Reserved.</p>
	5	-	M	RW	<p>This field indicates the minimum PIDT index that the component may use to determine packet injection rate, i.e., the hardware shall not auto adjust any PIDT index to a value less than C-Min Index.</p> <p>This field shall take precedence over all R-Min Index fields.</p>
	8	-	O	RO	<p>If Resource Array PTR is non-Null, then this field indicates the resource that is associated with each Resource Array table entry.</p> <p>0x0—Interface ID 0x1—Interface ID and VC 0x2—Protocol Engine ID 0x3—Resource Group ID (Group composition is managed through the Vendor-defined structure) 0x4-0xFF—Reserved</p>
	32	-	M	RW	<p>Packet Injection Delay Table (PIDT) defines a set of packet injection rates.</p> <p>PIDT<sub>0</sub> shall be set to 0 to indicate the maximum packet injection rate.</p> <p>Each PIDT<sub>k</sub> entry shall contain a larger value than PIDT<sub>k-1</sub> entry.</p> <p>Each PIDT value represents the minimum time delay between the start of the transmission of one packet on a given interface and the start of the transmission of the next packet on the same interface. This minimum time is calculated as:</p> $\text{Time} = \text{PIDT}_k * \text{Packet Generation Delay}$
	5	-	O	RW	This field indicates the minimum PIDT index that the resource may use to determine packet injection rate, i.e., the hardware shall not auto adjust any PIDT index for this resource to a value less than R-Min Index.

RO	51	-	-	-	This field shall be greater than or equal to the C-Min Index field.		
					RsvdP		

## 8.40. Component RKD Structure

The Component RKD structure is used to manage the authorization for accessing an *R-Key Domain (RKD)* for Requesters that support transmitting request packets with a *Region Key (R-Key)*.

- 5 The following are the features and requirements of the Component RKD structure:
- 10
- The Component RKD structure shall use the *Component RKD Structure Format*.
  - Each Requester that supports transmitting request packets with R-Keys shall support the Component RKD structure.
  - A 4096-bit RKD Authorization bit array controls if access to each RKD is permitted or not. The Component RKD structure shall be configured by management.
  - A Trusted Thread Enable bit controls if trusted threads are inherently authorized to access every RKD, independent of the RKD Authorization bit values.

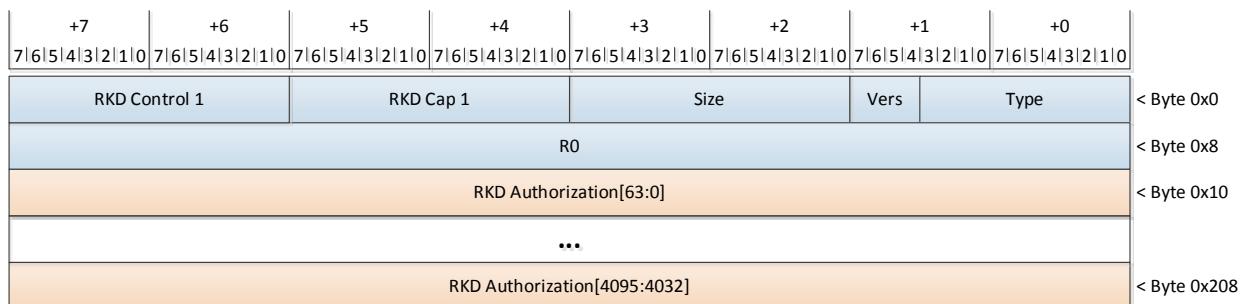


Figure 8-66: Component RKD Structure Format

Table 8-109: Component RKD Structure Fields

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description
<b>Version (Vers)</b>	4	0x1	M	RO	Structure Version
<b>RKD Cap 1</b>	16	-	M	RO	RKD Capabilities
		Bits 2:0			RKD Mechanism Table Type
					0x0: Fixed 4096-bit array 0x1-0x7: Reserved
		Bits 15:3			Reserved
<b>RKD Control 1</b>	16	-	M	RW	RKD Control
		Bit 0			RKD Validation Enable—Controls RKD validation using the RKD Authorization bit array

RKD Authorization	4096	-	M	RW	0b—Disable 1b—Enable
			Bit 1		Trusted Thread Enable—Controls whether trusted threads are inherently authorized to access every RKD, independent of RKD Authorization bit values. 0b—Disable 1b—Enabled
			Bits 15:2		RsvdP
	64	-	-	-	RKD Authorization—A bit array used to control whether access to each RKD is permitted. If Bit <sub>k</sub> == 1b, then access to the corresponding RKD is permitted. If Bit <sub>k</sub> == 0b, then access to the corresponding RKD is not permitted.
RO					Reserved

## 8.41. Component RE Table Structure

The Component RE Table structure is configured with a series of regular expressions. Once configured, an application can invoke operations such as *Pattern Requests* to apply a regular expression to an address range. The applied regular expression is identified by RE Table ID and an index within the RE Table pointed to by this structure.

The following are the features and requirements of the Component RE Table structure:

- The Component RE structure shall use the *Component RE Table Structure Format*.
- If a Responder supports multiple regular expression syntaxes, then it may provision multiple Component RE tables, with each unique syntax identified by the RE-UUID field.

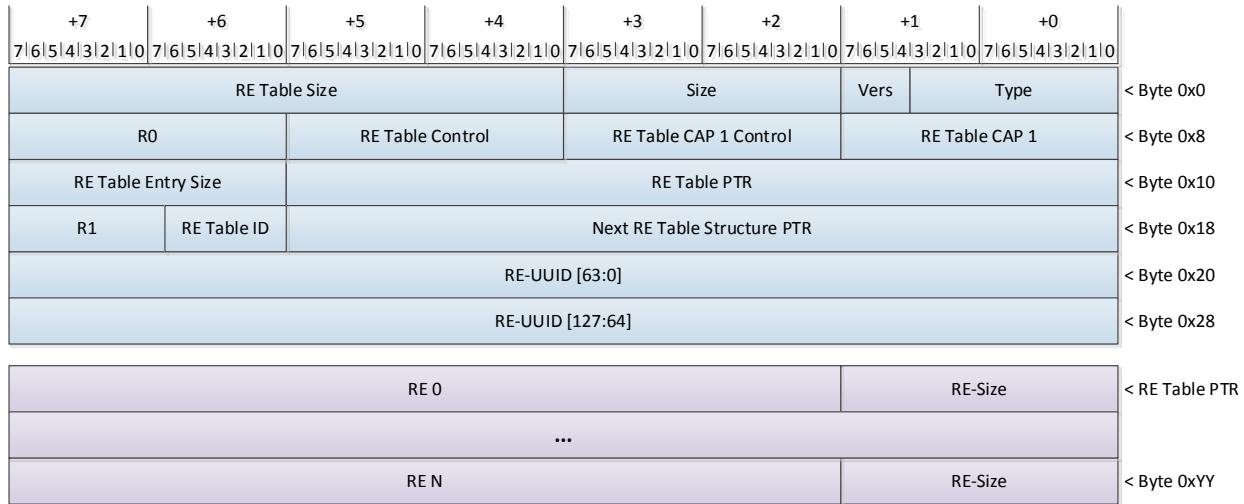


Figure 8-67: Component RE Table Structure Format

Table 8-110: Component RE Table Structure Fields

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description
Version (Vers)	4	0x1	M	RO	Structure Version
RE Table Size	32	-	M	RO	Number of entries within the RE Table. RE Table Size == 0x indicates the RE Table contains $2^{32}$ RE Table entries.
RE Table CAP 1	16	-	M	RO	RE Table Capabilities 1
	16	Bits 15:0			Reserved
RE Table CAP 1 Control	16	-	M	RW	RE Table CAP 1 Control
	16	Bits 15:0			RsvdP
RE Table Control	16	-	M	RW	RE Table Control
	16	Bit 0			RE Table Enable—if enabled, then operations (e.g., pattern requests may index this table to apply the indexed regular expression to the addressed resource range. 0b—Disable 1b—Enable
	16	Bits 15:1			RsvdP
RE Table PTR	48	-	M	RO	Points to byte zero of the RE Table associated with this structure.
RE Table Entry Size	16	-	M	RO	Maximum number of bytes provisioned per RE Table entry. RE Table Entry Size shall be an integer 4-byte multiple.

RE-UUID	7	-	M	RO	Provides a unique identifier for each supported RE Table. Identifiers are assigned in monotonically-increasing order starting with 0x0.
	48	-	M	RO	If the component supports multiple Component RE Table structures, then this points to the next structure, else shall be Null.
	128	-	M	RO	<p>The RE-UUID identifies the regular expression syntax supported by this table.</p> <p>The following UUID shall identify the POSIX Basic Regular Syntax (BRE): cb970b48f5a047f993badea847a05a15</p> <p>The following UUID shall identify the POSIX Extended Regular Syntax (ERE): 15d0fdb261474d2ebf909c49de961975</p> <p>The following UUID shall identify the Perl 6 Syntax: 738cc30124da4a65995b815a1c8e8c80</p>
RE-Size	16	-	M	RW	Indicates the length of the regular expression that has been configured in this RE Table entry. If RE-Size == 0x0, then the entry is not configured.
RE [n]	-	-	M	RW	Configured regular expression
R0	9	-	-	-	RsvdP
R1	16	-	-	-	RsvdP

## 8.42. Component PM Structure

In addition to gathering statistics, a component may gather additional packet-specific information used by performance management tools to better understand and optimize application and topology behavior. The packet-specific information is stored in one of two performance log record types. The Component PM structure is used to manage and access the performance management log.

The following are the features and requirements of the Component PM structure:

- The Component PM structure shall use the *Component PM Structure Format*.
- If a packet suffers a data integrity error or contains a stomped ECRC, then a performance log record shall not be recorded.
- If the Deadline sub-field within a packet Congestion field is decremented to zero prior to the packet being transmitted, then the Egress Delta Timestamp field shall be set to 0x0 to indicate that the packet was silently discarded.

- Performance log records are written to starting at log record zero in monotonically increasing order modulo Max\_Perf\_Records. A component may overwrite valid performance log records if all performance log records have been consumed.
  - Software may clear all log records by writing 1b to the Clear Performance Marker Log bit.
  - Once processed, software shall clear the Ingress Timestamp field within a performance log record to indicate this record is free. Software shall clear performance log records in monotonically-increasing order modulo Max\_Perf\_Records.
  - The Performance Log Record Type 0 shall use the *Performance Log Record Type 0 Format*.
  - The Performance Log Record Type 1 shall use the *Performance Log Record Type 1 Format*.

5

Table 8-111: Performance Log Record Fields

Field Name	Size (bits)	Description
<b>DCID</b>	12	Copy of the packet's DCID
<b>SCID</b>	12	Copy of the packet's SCID
<b>OCL</b>	5	Copy of the packet's OCL
<b>OpCode</b>	5	Copy of the packet's OpCode
<b>Length</b>	7	Copy of the packet's Length
<b>RK</b>	1	Copy of the packet's RK
<b>GC</b>	1	Copy of the packet's GC
<b>NH</b>	1	Copy of the packet's NH
<b>LP</b>	1	Copy of the packet's LP
<b>Ingress VC</b>	5	Copy of the packet's VC when the packet was received
<b>EI</b>	1	Copy of the packet's Explicit Congestion sub-field when the packet was received
<b>EO</b>	1	Copy of the packet's Explicit Congestion sub-field when the packet was transmitted
<b>Tag</b>	12	Copy of the packet's Tag
<b>Ingress Timestamp</b>	64	Timestamp when the packet was received. Each tick represents 1 ns.
<b>Egress Delta Timestamp</b>	64	Delta time (ns) from Ingress Timestamp of when the packet was transmitted. Timestamp should be a close approximation of when the packet was actually transmitted.  If Egress Delta Timestamp == 0x0, then the packet was discarded by the switch due to an error or the Deadline sub-field within the packet's Congestion field reaching zero prior to transmission.

Field Name	Size (bits)	Description
Ingress Packet Deadline	10	Copy of the 10-bit Deadline sub-field within the packet's Congestion field at the time the packet was received at the ingress interface
Ingress Interface	12	Interface on which the packet arrived
Egress Interface	12	Interface on which the packet was transmitted
Vendor-defined 0	8	Vendor-defined field
Vendor-defined 1	22	Vendor-defined field
DSID	16	Copy of the packet's DSID
SSID	16	Copy of the packet's SSID

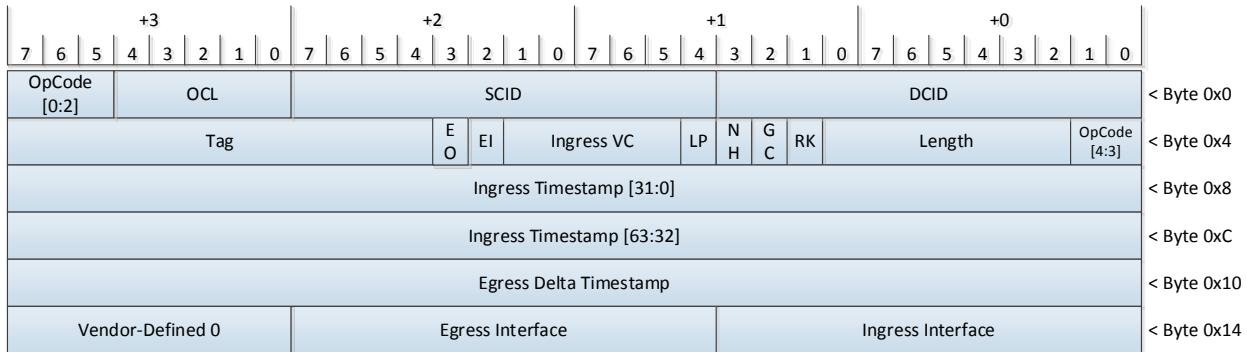


Figure 8-68: Performance Log Record Type 0 Format

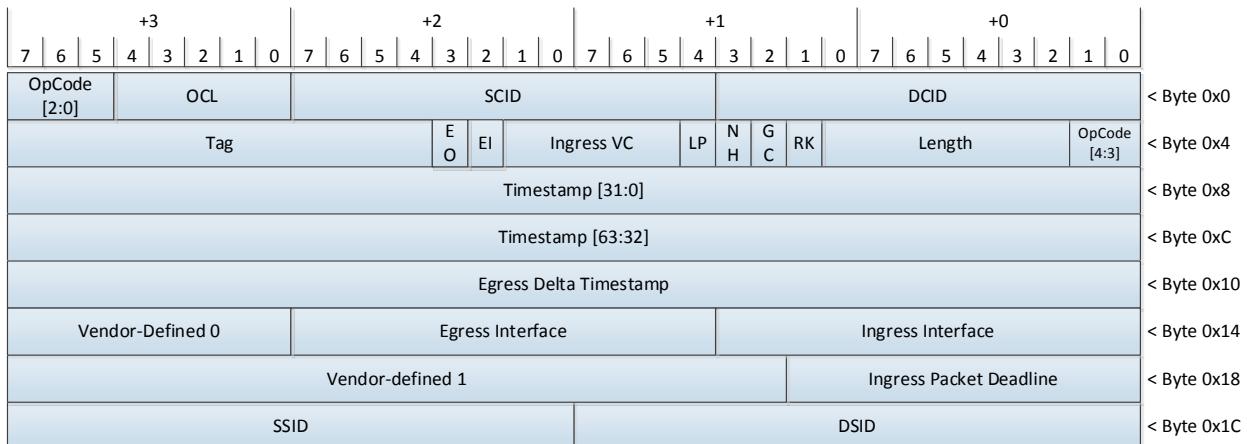


Figure 8-69: Performance Log Record Type 1 Format

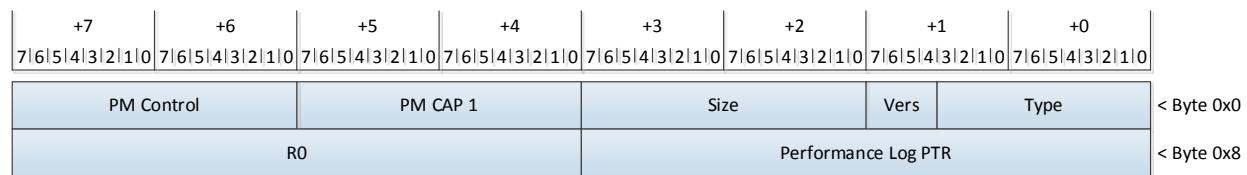


Figure 8-70: Component PM Structure Format

Table 8-112: Component PM Structure Fields

Field Name	Size (bits)	Value / Bit Location	M/O	Access	Description
<b>Version (Vers)</b>	4	0x1	M	RO	Structure Version
<b>PM CAP 1</b>	16	-	M	RO	Performance Marker Support—Indicates if the component supports gathering packet-specific data if an explicit OpClass packet contains the PM bit and PM == 1b. If non-zero, indicates the type of information that it is capable of gathering. 0x0—Unsupported 0x1—Generates Performance Log Record Type 0 0x2—Generates Performance Log Record Types 1 0x3-0x7—Reserved
		Bits 2:0			
		Bits 7:3			
		Bits 15:8			
<b>PM Control 1</b>	16	-	M	RW	Performance Log Record Enable—if enabled, then the component shall generate a performance log record upon receipt of any packet with PM == 1b. 0b—Disabled 1b—Enabled
		Bit 0			
		Bit 1			
		Bits 15:2			

<b>Performance Log PTR</b>	32	-	M	RO	If supported, then this field points to the performance log.
<b>RO</b>	32	-	-	-	Reserved

## 9. Switches

A switch relays end-to-end packets between an ingress interface and an egress interface.

The following are the features and requirements associated with switches:

- A switch shall support unicast packet relay within a single subnet.
- A switch may support multi-subnet unicast packet relay.
- A switch may be a discrete component or may be integrated within a Requester or Responder component.
- A switch shall support a *Component Switch Structure*.
- A switch may support 2-4096 interfaces.
  - Each component interface that supports unicast packet relay shall provision the Access Key Table PTR, VCAT PTR, LPRT PTR, and the MPRT PTR fields specified in the *Interface Structure*.
  - A component with an integrated switch may support interfaces that do not support packet relay. For example, a component that supports packet relay between a subset of the component interfaces, and supports the P2P-Core OpClass on a subset of interfaces.
- A switch shall not relay link-local packets or end-to-end packets that do not contain a DCID or MGID field (e.g., P2P-Core OpClass packets).
- A switch may support ingress and egress interface Access Key validation (see *Component Switch Structure* for configuration details).
- A switch shall perform packet validation as specified in *Switch Packet Processing*.
- A switch shall perform *Prelude CRC (PCRC)* validation prior to relaying a packet.
  - Upon successful PCRC validation, a switch may relay a packet without waiting for the entire packet to be received and validated. This is referred to as cut-through.
- A switch shall be capable of stomping the packet's *End-to-end CRC (ECRC)* field.
- A switch may relay an end-to-end packet with a stomped ECRC field as it would a valid packet. If packet transmission has not started, then a switch should silently discard a packet that contains a stomped ECRC field.
- A switch shall perform *End-to-end CRC (ECRC)* validation prior to completing packet transmission through an egress interface.
  - A switch may delay packet relay until the entire packet is received and ECRC validation is successfully completed. This is referred to as store-and-forward.
  - If packet transmission has not started, then a switch should silently discard a packet that fails ECRC validation.
- Unless explicitly stated otherwise by this specification (e.g., stomping an ECRC field or performing VC remapping), an end-to-end packet shall not be modified by a switch.
- Each interface used to relay packets shall provision a distinct Linear Packet Relay Table (LPRT). The LPRT is located using the LPRT PTR located in the corresponding *Interface Structure*.
- If a switch supports multi-subnet packet relay, then each interface used to relay multi-subnet packets shall provision a distinct Multi-subnet Packet Relay Table (MPRT). The MPRT is located using the MPRT PTR located in the corresponding *Interface Structure*.
- If a unicast packet does not contain a DSID field (see *Explicit OpClass Multi-subnet Field*), then the switch shall relay the packet using the LPRT associated with the ingress interface.

- If a unicast packet contains a DSID field and the switch supports multi-subnet packet relay, then the switch shall relay the packet using the MPRT and the LPRT associated with the ingress interface.
- A switch may relay a packet to the same interface that received the packet if the packet relay tables are so configured. Management shall ensure the packet relay tables are configured to prevent resource dependency cycles.
- A unicast packet stalled on egress interface  $VC_k$  shall not stall packet relay of unicast or multicast packets destined for a different egress interface  $VC_k$ .
- If a switch supports multicast packet relay, then the following requirements apply:
  - A switch shall perform multicast packet relay only on multicast packets (explicit OpClass packets with  $OCL == \text{Multicast OpClass}$ ).
  - A switch may interleave and transmit unicast and multicast packets independent of the order received.
  - If a multicast packet does not contain a Global Multicast Prefix field (see *Explicit OpClass Multi-subnet Field*), then the switch shall relay the packet using the Multicast Packet Relay Table (MCPRT).
  - If a multicast packet contains a Global Multicast Prefix field, then the switch shall relay the packet using the Multi-subnet Multicast Packet Relay Table (MSMCPRT).
- If a switch does not support multicast packet relay, then unless explicitly stated otherwise by this specification, a switch shall relay all multicast packets through the configured Default Multicast Egress Interface (see *Component Switch Structure*).
  - If the Default Multicast Egress Interface is not configured, then the packet shall be handled as a UP.
  - If the switch is a member of the multicast group (e.g., the management multicast group), then the switch shall create a copy of the packet for its consumption prior to relaying the packet through the Default Multicast Egress interface.
- A switch interface may support Control OpClass packet filtering. Control OpClass packet filtering is used to prevent a component from communicating with non-management components, e.g., while the component is in the C-CFG state. If enabled, then
  - A switch shall silently discard any non-Control OpClass packet.
  - A switch shall silently discard any Control OpClass packet with  $GC == 0b$  whose DCID does not match one of the following: the switch's configured PMCID, PFMCID, or SFMCID.
  - If a switch has not been configured to support multi-subnet management, then the switch shall silently discard any Control OpClass packet with  $GC == 1b$ .
  - If a switch has been configured to support multi-subnet management and the packet's  $GC == 1b$ , then the switch shall silently discard any packet with a [DCID, DSID] that does not match either the switch's [PFMCID, PFMSID] or [SFMCID, SFMSID].
  - Control OpClass packet filtering is independently enabled per component interface.
  - A switch may be configured to treat a discarded non-Control OpClass packet as an AE error (see and inform management based on AE handling configuration (see *Component Error and Signal Event Structure*).
- A switch shall support the semantics specified in *Directed Control Space Packet Relay*.
- A switch shall ensure that all control structures, ancillary structures, and data and control paths are protected at all times by implementation-specific data integrity techniques that can detect data corruption.
- A switch may support PCO communications. If supported, then:
  - All interfaces used to relay packets should support PCO communications.

- If the ingress interface  $VC_K$  and egress interface  $VC_N$  are enabled to use PCO communications, then:
  - The switch shall relay packets between the ingress interface VC and egress interface VC in the order received.
  - The switch shall not update the Deadline sub-field within the Congestion field within packets relayed between the ingress interface VC and the egress interface VC.
  - The switch shall not drop packets relayed between the ingress interface VC and the egress interface VC for any reason with the exception of the egress interface is unable to transmit packets due to switch failure / disable, link failure / disable, *Component Reset*, or *Interface Reset*.

## 9.1. Unicast Packet Relay

Gen-Z relay table structures are flexibly defined such that systems of all sizes can be implemented with high performance. The following sub-sections illustrate different solution scales, topologies, and routing algorithms. Additional routing algorithms may be supported.

If an interface's *Interface I-Control* LPRT Enable == 0b, then packet relay is disabled on this interface, i.e., the interface shall not be used to relay packets to other component interfaces.

If an interface's *Interface I-Control* MPRT Enable == 0b, then multi-subnet packet relay is disabled on this interface, i.e., the interface shall not be used to relay multi-subnet packets to other component interfaces.

Within the following sub-sections, the LPRT variable is valid only if *Interface I-Control* LPRT Enable == 1b, and the MPRT variable is valid only if *Interface I-Control* MPRT Enable == 1b.

### 9.1.1. Single Subnet – Single Route

A switch may support single-subnet packet relay and provision a single route per destination component. *Single-subnet—Single Route* illustrates the structures and conceptual steps involved:

1. The switch accesses the ingress interface's LPRT, and uses the packet's DCID to directly index and locate the corresponding LPRT route entry row.
2. Since the switch supports a single route entry (MSS = 0b and HCS = 0b), the following applies:
  - a. VCA field is ignored.
  - b. VC remapping is not be performed.
  - c. If V == 1b, then the packet is relayed to the EI (egress interface), else it is handled as a UP error.

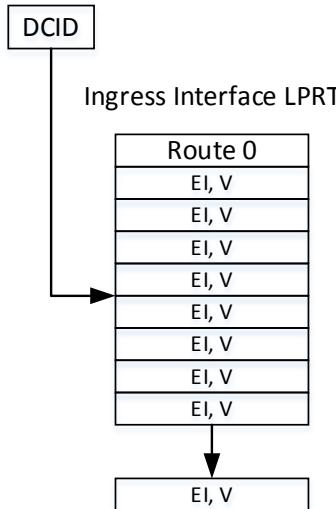


Figure 9-1: Single-subnet—Single Route

### 9.1.2. Single-Route VC Remapping

A switch may support single-subnet or multi-subnet packet relay with VC remapping. *Single Route—VC Remapping* illustrates the structures and conceptual steps involved in multi-subnet with VC remapping (in the following steps, if supporting a single subnet, then all MPRT and GC bit-related steps can be ignored):

1. In parallel, the switch accesses the ingress interface's LPRT and MPRT.
  - a. The switch uses the packet's DCID field to directly index and locate the corresponding LPRT route entry row.
  - b. If GC == 1b, then the switch uses the packet's DSID field to directly index and locate the corresponding MPRT route entry row.
2. Since the switch supports a single route entry, the following applies to the LPRT and MPRT route entry rows:
  - a. MHC and HC fields shall be ignored.
  - b. If V == 1b, then the route entry can be used to determine the egress interface, else the route entry is ignored.
  - c. The switch uses the VCA route entry to access the VCAT table and derive a VCM (VC mask).
3. If (GC && (DSID ≠ LSID)), then use MPRT routes, else use LPRT routes. If V == 0b, then the packet is handled as a UP error. This step may be performed prior to accessing the LPRT or MPRT, in which case only one table is accessed to determine the route and egress interface.
4. Prior to relaying the packet, the switch uses the VCM to perform VC remapping. If the packet's VC is a valid output VC (VCM[Bit<sub>VC</sub> == 1b]), then the VC does not need to be remapped. If (VCM[Bit<sub>VC</sub> == 0b]), then any VC where VCM[Bit<sub>VC</sub> == 1b] may be used to remap the packet's VC field. If there are multiple valid output VC, then output VC selection may be random, adaptive (e.g., based on congestion information), etc.

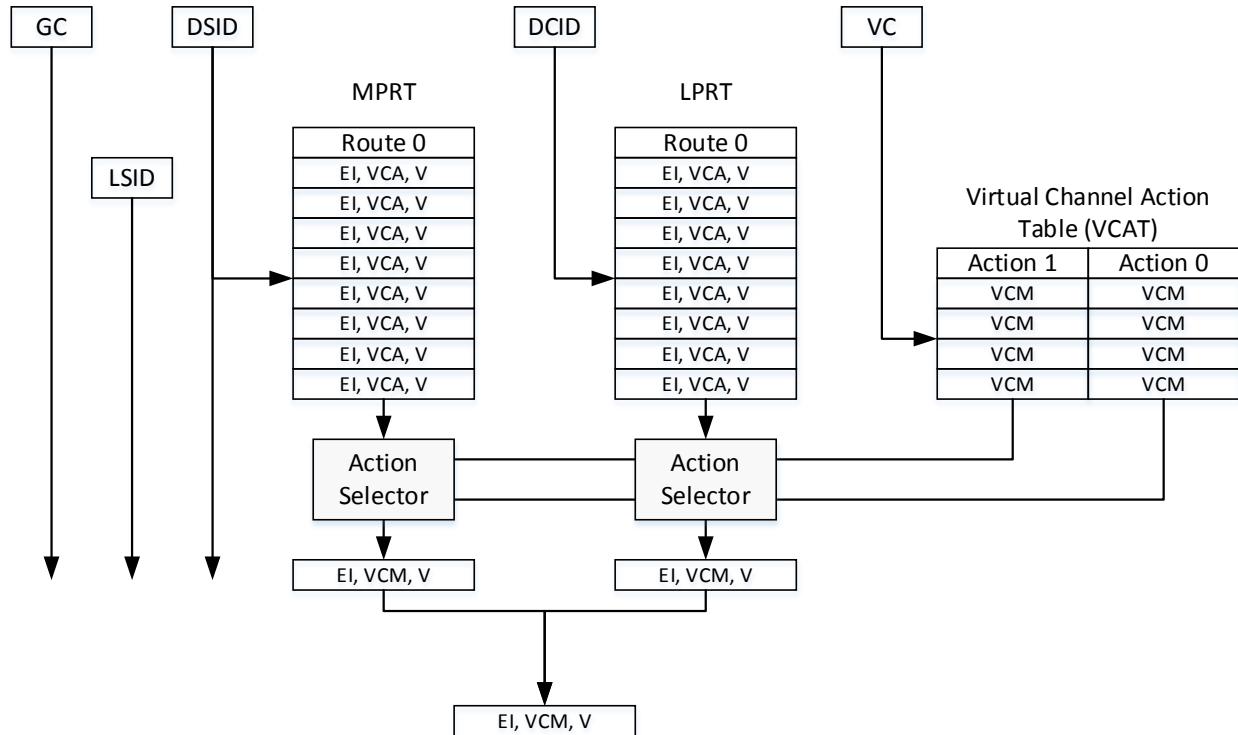


Figure 9-2: Single Route—VC Remapping

**Developer Note:** VC remapping is applicable to multiple topologies and scenarios. For example, in a solution where Requester and Responder components support a small number of VCs (e.g., 1-2) and the switch components supports many VCs (e.g., 8-32). VC remapping is used to prevent deadlocks and enable robust topologies while supporting any mix of simple-to-robust Requester and Responder components.

Another example is to support more complex routing algorithms and topologies. Though some topologies such as a Folder-Clos, Mesh, Flattened Bufferfly, and Hyper-X do not require VC remapping, other topologies such as a Torus or Dragonfly do. When VC remapping is used in conjunction with the routing tables, the architecture is capable of supporting a wide-range of topologies without sacrificing performance. Further, in topologies where VC remapping is not required, it can still be used to increase performance.

### 9.1.3. Single-Subnet Multipath Routing

A switch may support single-subnet packet relay with multiple routes to a given destination. *Single-subnet—Multipath Selection* illustrates the structures and conceptual steps involved in single-subnet, multipath selection routing. In this example, MSS = 0b and HCS = 0.

1. The switch accesses the ingress interface's LPRT.
  - a. The switch uses the packet's DCID field to directly index and locate the corresponding LPRT route entry row.
2. The following applies to the LPRT route entry rows:
  - a. The switch uses random or adaptive route selection.
  - b. If V == 1b, then the route entry can be used to determine the egress interface, else the route entry is ignored.

- c. The switch uses the VCA within the route entry to access the VCAT table, and to derive a VCM (VC mask).
3. If all LPRT route entries have  $V == 0b$ , then the packet is handled as a UP error.
4. Action Selector:  $VCM = VCMs[VCA]$
5. a. If the packet's VC is a valid output VC ( $VCM[Bit_{VC}] == 1b$ ), then the VC does not need to be remapped. If ( $VCM[Bit_{VC}] == 0b$ ), then any VC whose corresponding  $VCM[Bit_k == 1b]$  may be used to remap the packet's VC field. If there multiple valid output VC, then output VC selection may be random, adaptive (e.g., based on congestion information), etc.

**Developer Note:** Random route selection provides statistical load-balancing in the steady state.

10 Adaptive route selection typically selects the route with the lowest expected latency and uses egress interface congestion state as an input to the selection criteria.

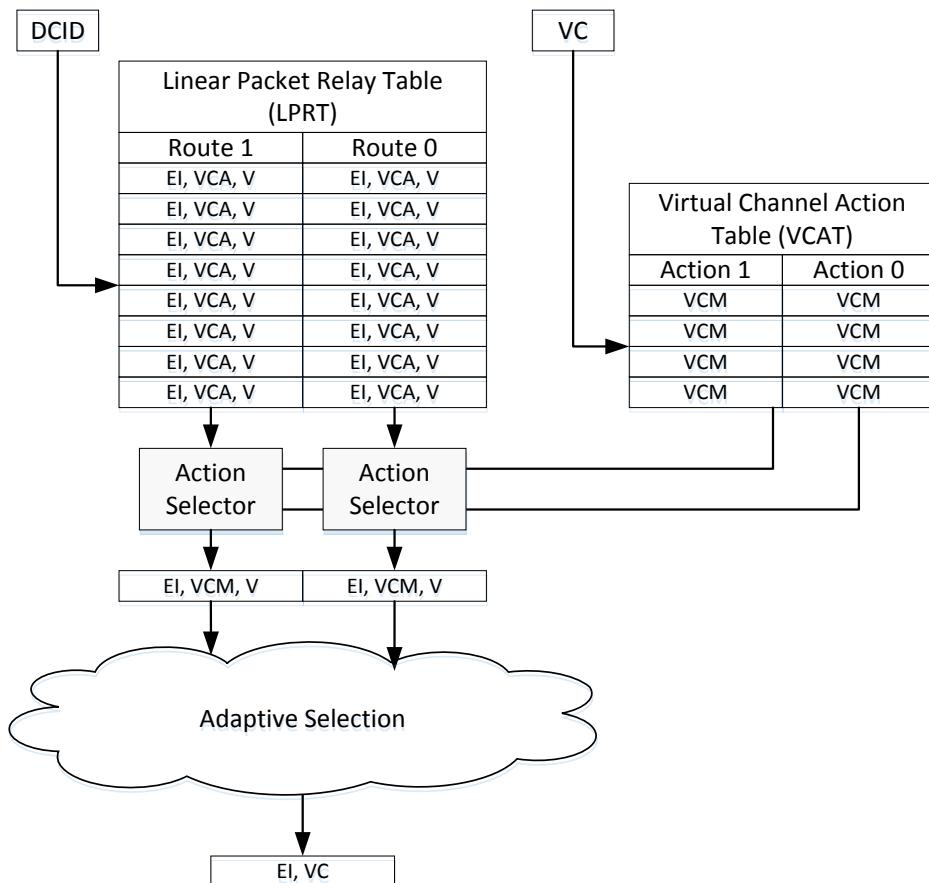


Figure 9-3: Single-subnet—Multipath Selection

#### 9.1.4. Multi-Subnet Multipath Routing

15 A switch may support multi-subnet packet relay with multiple routes to a given destination. *Multi-subnet Multipath Selection (MSS = 1b and HCS = 1b)* and *Multi-subnet Multipath Selection (MSS = 1b and HCS = 0b)* illustrate the conceptual structures and steps involved in multi-subnet-multipath selection routing.

The following steps assume MSS = 1b and HCS = 1b.

1. The following pseudo-code illustrates table selection:

```
if ST[VC]:  
    if GC && DSID ≠ LSID:  
        if LPRT.MHC == 0:  
            use MPRT routes  
        elif LTF[VC]:  
            use LPRT routes  
        else:  
            use MPRT and LPRT routes  
    else:  
        use MPRT routes  
    else:  
        if GC && DSID ≠ LSID:  
            use MPRT routes  
        else:  
            use LPRT routes
```

2. The following pseudo-code illustrates hop count computation:

```
if GC && DSID ≠ LSID:  
    if SS:  
        CHC = LPRT.MHC + MPRT.MHC  
    else:  
        CHC = DHC + MPRT.MHC  
    else:  
        CHC = LPRT.MHC
```

3. If V == 1b, then the route entry can be used, else the route entry is ignored.

4. Action Selector is VCM = VCMs[VCA]

5. Use packet's VC to index VCAT to obtain VCM and threshold (TH)

6. The following pseudo-code illustrates how a threshold comparison is performed:

```
if TE[VC]:  
    if CHC > TH:  
        VCM = 0x00000000  
    else:  
        VCM = VCM  
    else:  
        VCM = VCM
```

7. Select an egress interface and VC.

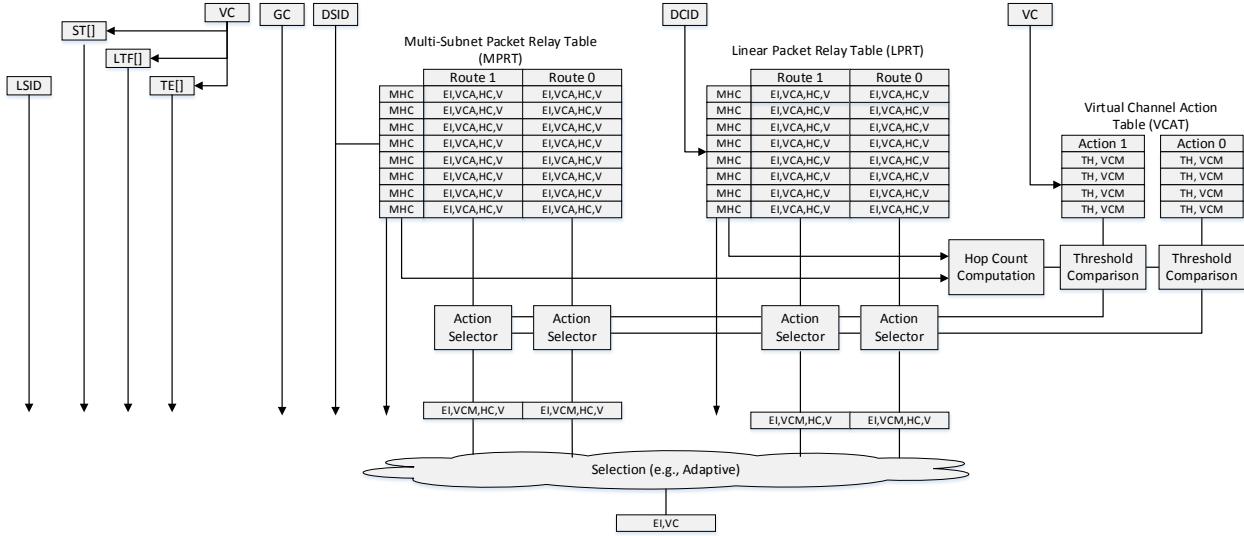


Figure 9-4: Multi-subnet Multipath Selection (MSS = 1b and HCS = 1b)

The following steps assume MSS = 1b and HCS = 0b.

1. The following pseudo-code illustrates table selection:  
 if GC && DSID  $\neq$  LSID:  
     use LPRT routes  
 else:  
     use MPRT routes
2. If V == 1b, then the route entry can be used, else the route entry is ignored.
3. Action Selector is VCM = VCMs[VCA]
4. Use packet's VC to index VCAT to obtain VCM and threshold (TH)
5. Select an egress interface and VC.

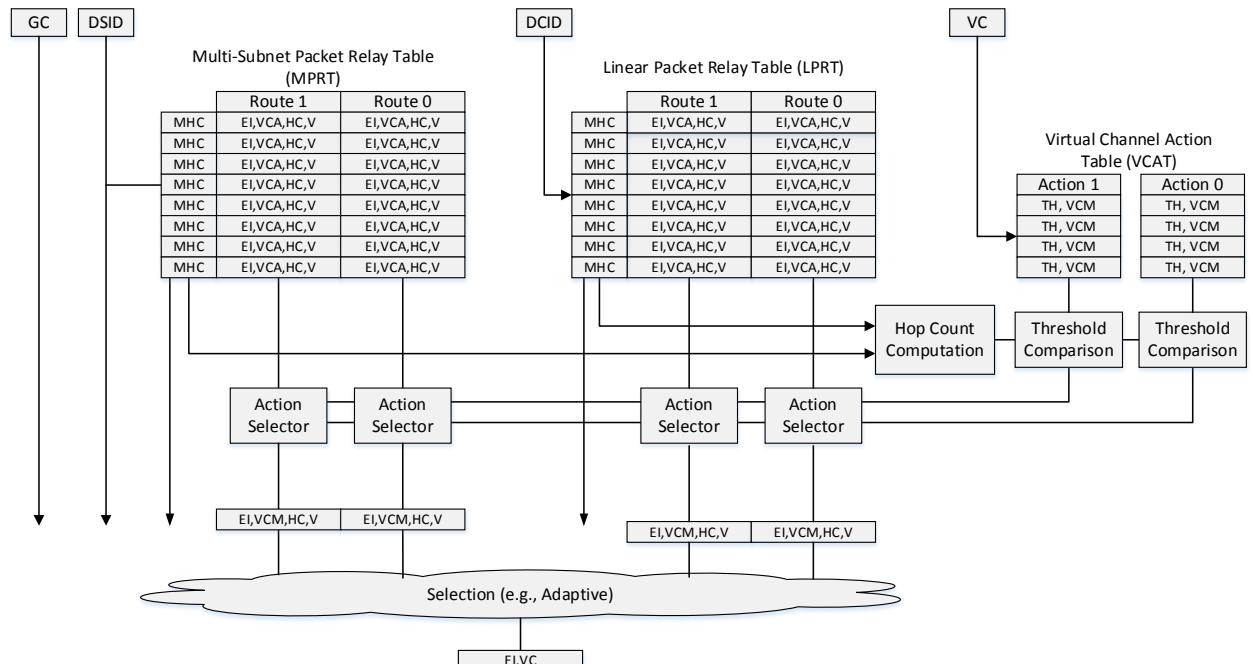


Figure 9-5: Multi-subnet Multipath Selection (MSS = 1b and HCS = 0b)

## 9.2. Routing Tables and Corrupt Packet Relay

Switches that support cut-through packet relay perform PCRC validation prior to accessing the packet relay tables. PCRC validation does not cover the packet's CID or SID fields; these fields are covered by the packet's ECRC. If the packet's DCID and / or DSID fields are corrupted, then the switch will not detect this corruption until it validates the ECRC. As a result, a corrupted packet could be incorrectly relayed to an egress interface, and depending upon the topology and routing algorithm, this could result in a deadlock.

To prevent such deadlocks, the packet relay tables need to be configured with all restricted routes removed. For example, topologies that use XY dimension routing do not allow packets to be relayed from a Y-ingress interface to an X-dimension egress interface. The relay tables need to be configured to prevent such packet relay, and in doing so, cause any packet whose corrupted DCID and / or DSID fields would violate this rule to be silently discarded.

## 9.3. Directed Control Space Packet Relay

Directed Control Space packet relay is used for two purposes:

- To configure an uninitialized component
- To enable two independent fabric managers to indirectly communicate through intermediate switches without comprehending the peer management component's CID / GCID.

### 9.3.1. Configuration Using Directed Packet Relay

Configuration of an uninitialized component using in-band Control OpClass packets uses Directed Control Space packet relay. Until a CID and SID (if applicable) and the *Component Destination Table Structure* are configured on the uninitialized component and the switch component's LPRT and MPRT tables are updated, management cannot directly communicate with the component. Instead, management communicates with the directly-attached switch which is directly-attached to the component, and the switch relays request and response packets using the interface identifier associated with the switch interface directly attached to the component as illustrated in *Example Directed-Control Space Packet Relay*.

The following are the features and requirements associated with Directed Control Space packet relay:

- Management shall set the packet's [DCID, DSID (if applicable)] to match one of the configured [CID, SID (if applicable)] of the switch, DR = 1b and DR-Interface = switch egress interface in the Control Read, Control Write, or Control Write MSG request packet.
- Upon receipt of a Control Read, Control Write, or Control Write MSG request packet that is destined to the switch, the switch shall validate the packet as specified in *Switch Packet Processing*.
  - If an error is detected, then the switch shall handle the error, and shall not relay the packet. If no error is detected, then the switch shall set the packet's VC = VC 0, and shall relay the packet through the DR-Interface egress interface. The switch shall calculate a new ECRC, and place it in the packet's ECRC field.
- The destination component shall validate and execute the request packet, and shall transmit a Control OpClass response packet. While the component is in the C-CFG State, the component

shall not validate the request packet's DCID and DSID (if applicable) fields. Further, the component shall copy the request packet's DCID and DSID (if applicable) respectively to the response packet's SCID and SSID fields.

- 5 Management shall configured the component's [CID 0, SID 0] and the *Component Destination Table Structure* using Directed Control Space Packet Relay access, before continuing configuration using Control Read and Control Write directly, not using Directed Control Space Packet Relay. If applicable, management may configure multiple CID and SID (if applicable).
- 10 Management shall not use Directed Control Space Packet Relay to communicate with the component while in any state other than C-CFG. The Control Write request packet that sets *Core C-Control Component Enable* = 1b shall not use Directed Control Space Packet Relay.
- 15 Management shall configure the component's [CID 0, SID 0] in all switch LPRT and MPRT (if supported), prior to transitioning the component to the C-Up state.

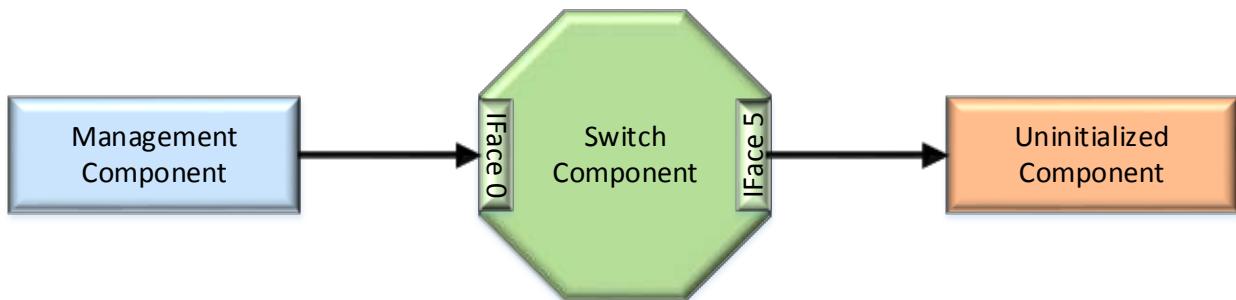


Figure 9-6: Example Directed-Control Space Packet Relay

### 15 9.3.2. Indirect Manager Communication

If two configured topologies are connected, then the managers need to communicate to reconcile component identifiers, packet relay tables, etc. to enable the two topologies to communicate. Indirect manager communication shall use only unreliable *Control Write MSG* request packets.

20 *Example Indirect Manager Communication* illustrates an example mechanism to enable the disparate managers to communicate. To exchange unreliable *Control Write MSG* request packets, the following steps are taken:

1. The respective managers configure switch A and switch B to notify management upon detecting a new component, e.g., to generate an *Unsolicited Event (UE) Packet*.
2. Once Switch A and switch B are connected and upon the link transitioning to L-Up, each interface initiates a Peer-Attribute *Link CTL* exchange. Switch A and switch B inform management that a new component has been detected (in this example, manager A transmits an *Unsolicited Event (UE) Packet*). Manager A and Manager B determine if the respective *Interface Structure* Peer State Peer Component C-State == C-Up. If in C-Up, then Manager A and Manager B may transmit a Control Read request packet with DR = 1b, DR-Interface set to the respective switch interface, and a non-zero MGR-UUID field to determine the manager of the new component.
  - a. Upon receipt of the Control Read request packet, each switch validates if the request packet was transmitted by a valid manager, and when this fails, silently discards the packet.
3. If a manager fails to receive a Control Read Response packet and the manager supports indirect manager communication, then a manager sets *Interface I-Control* Ingress DR Enabled = 1b in the

switch egress interfaces. This enables the switch to receive an unreliable *Control Write MSG* request packets with DR == 1b.

4. Manager A transmits an unreliable *Control Write MSG* request packet with DR = 1b and DR-Interface set to switch A egress X. The maximum message size shall be less than or equal to Max\_Packet\_Payload.
5. a. Manager B may take the same actions as manager A but targeting switch B egress Y.
5. If switch B, egress Y *Interface I-Control* Ingress DR Enabled == 1b, then upon receipt, switch B updates the packet's DCID and DSID (if applicable) to target manager B (manager B is either switch B's Primary Fabric Manager or Secondary Fabric Manager), updates the packet's SCID and SSID (if applicable) to indicate switch B, and updates the packet's Tag field to ensure it does not collide with a Tag used in any other request packet. Further, switch B sets SDR = 1b and DR-Interface = ingress interface upon which the unreliable *Control Write MSG* request packet arrived (these indicate the interface to target in a subsequent unreliable *Control Write MSG* request packet transmitted by manager B). Switch B transmits the packet to manager B.
6. Upon receipt, manager B interprets the packet's payload using a method outside of this specification's scope. From this point onward, managers A and B exchange of a series of *Control Write MSG* request packets that are directed through switches A and B.

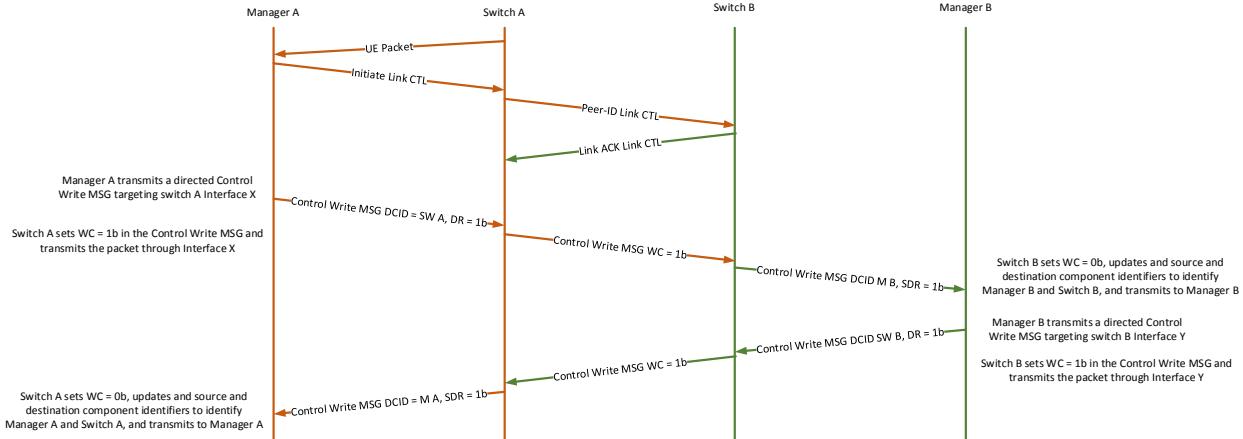


Figure 9-7: Example Indirect Manager Communication

## 10. Transparent Router (TR)

5 A TR proxies request packets from the Requester's subnet to the Responder's subnet and response packets from the Responder's subnet to the Requester's subnet. Since the TR's presence is transparent to the Requester and the Responder, the TR is protocol-aware and maintains state to correlate request packets with response packets. In other words, inter-subnet and intra-subnet packets are identical in format and semantics, and this requires the TR to actively participate in the communication process.

10 The following are the features and requirements associated with a TR:

- 15 • A TR may be a discrete component or integrated within another component type.
- A Requester within a given subnet sees a TR as a Responder or a Requester-Responder. A Responder within a given subnet sees a TR as a Requester or a Requester-Responder. A TR within a given subnet sees a TR as a Requester-Responder.
  - o If a TR has a Responder role, then the TR shall translate request packets only from the Requester's subnet, and shall translate response packets only from the Responder's subnet.
- A TR presents a single, contiguous Data Space to each directly-attached subnet.
  - o A TR may present a different sized Data Space to each subnet.
  - o The Data Space presented to each directly-attached subnet may be mapped to multiple subnets. For example, *Processors Communicating Through a TR* illustrates three subnets. For each subnet, the TR advertises a unique Data Space, e.g., subnet A sees a single Data Space that enables transparent access to subnets B and C, subnet B sees a single Data Space that enables transparent access to subnets A and C, and subnet C sees a single Data space that enables transparent access to subnets A and B.
  - o A Responder shall treat a TR as a Requester or a Requester-Responder.
  - o A Requester shall treat a TR as a Responder or a Requester-Responder with addressable resources.
  - o A TR shall treat a peer TR as a Responder or a Requester-Responder with addressable resources.

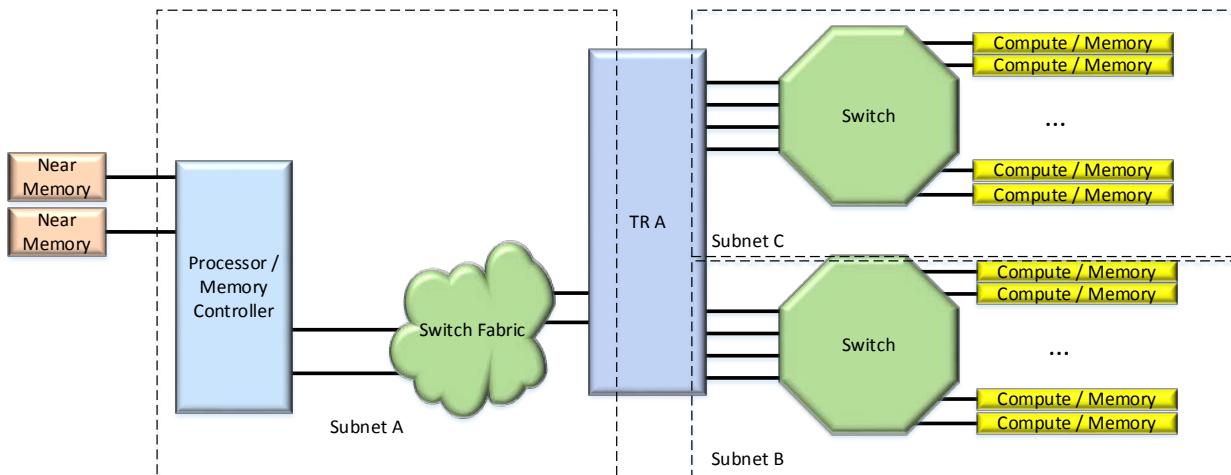


Figure 10-1: Processors Communicating Through a TR

- 30 • Within each subnet, a TR shall be assigned a subnet-local CID. In the above example, the TR is assigned one CID per attached subnet—CID A, CID B, and CID C.

- Two subnets may be connected via multiple TRs, e.g., as illustrated in *Two Subnets Connected by a Two TRs*. The TRs may be directly connected to each other (as illustrated), or may communicate through intermediate switches in either subnet.

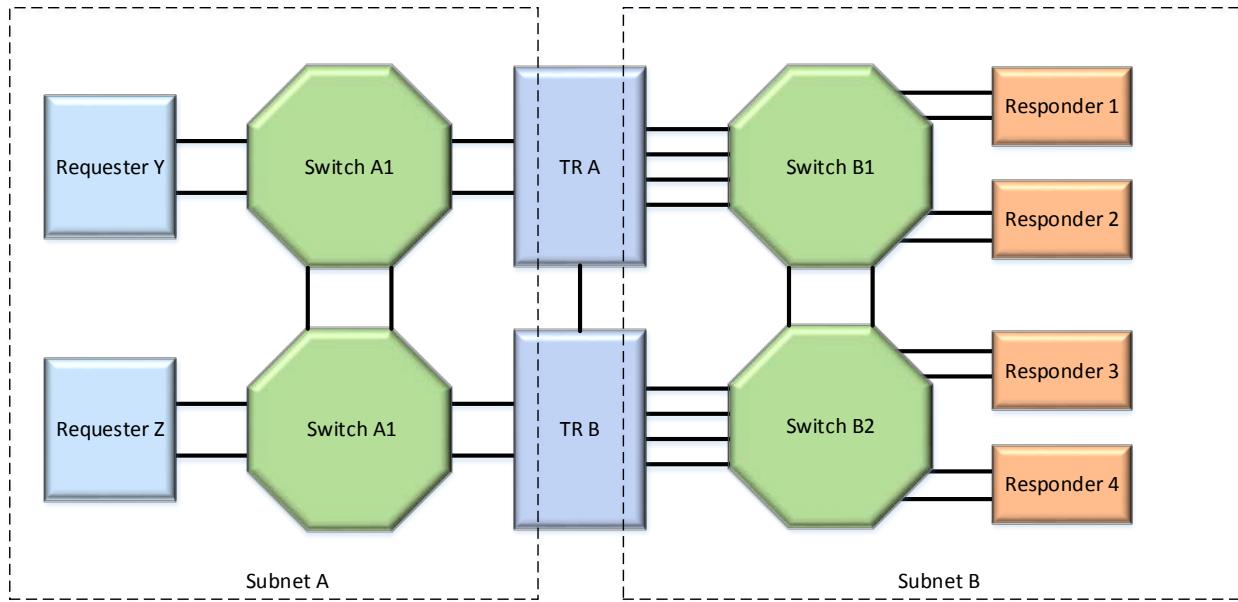


Figure 10-2: Two Subnets Connected by a Two TRs

5

- A TR may support a maximum of 16 subnets. If a solution needs to support more than 16 subnets, then a TR may be attached to one or more TRs to access additional subnets.
- A TR may be directly-attached to a subnet composed primarily of TR components, e.g., as illustrated in *Multiple Subnets Connected by Multiple TRs*. See *TR-to-TR Communications*.

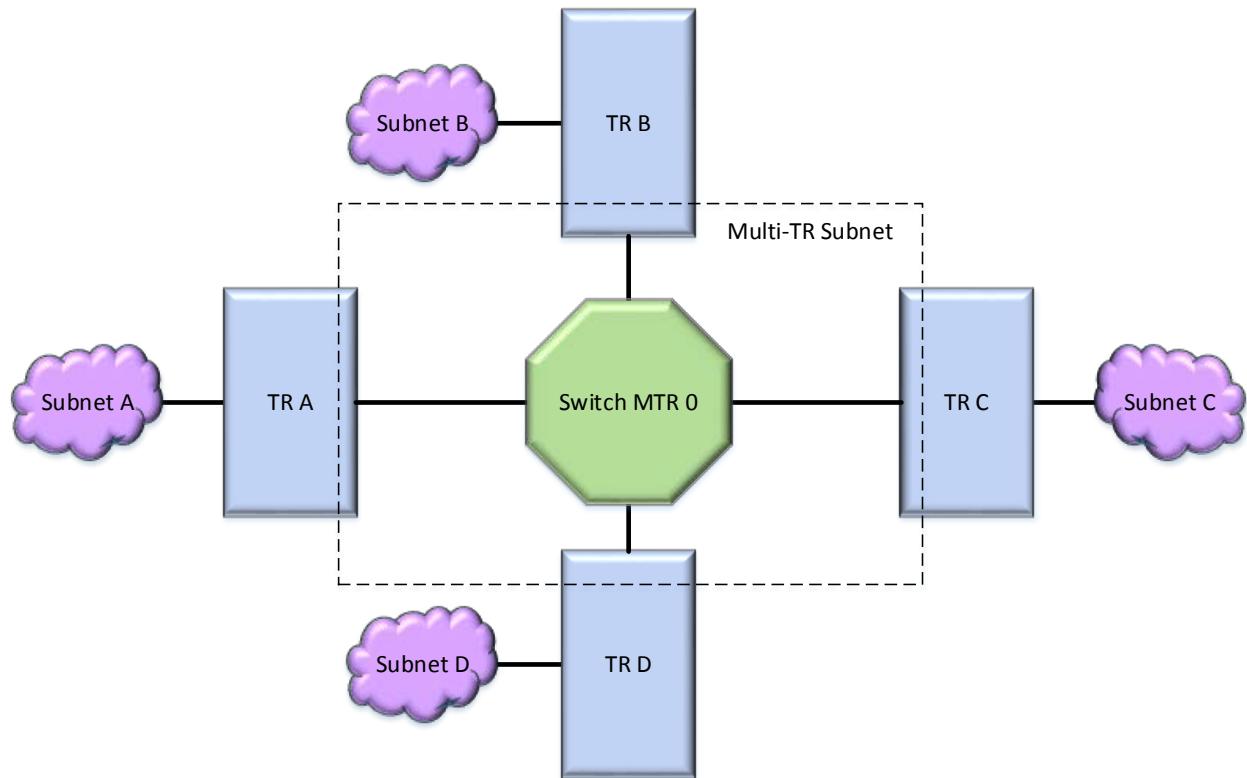


Figure 10-3: Multiple Subnets Connected by Multiple TRs

- A TR validates and translates a request packet into one or more destination subnet request packets. A TR validates and translates response packets into destination subnet response packets.
  - A TR shall not be responsible for Reliable Delivery of a request packet.
  - A TR shall provide the requisite latency values to enable management to calculate Requester retransmission timers.
    - If a TR is directly-attached to a subnet composed of non-TR components, then the TR shall provide a single set of latency values representing the worst-case latency across all components that any requester in the source subnet can communicate with in the destination subnet.
    - If a TR is directly-attached to a multi-TR subnet, then the TR provides a single set of latency values representing the worst-case latency across all accessible subnets.
    - If a TR supports non-deterministic request translation, then the TR shall support the semantics specified in *Non-Deterministic Request Execution*.
  - A TR shall maintain a Request Forward Progress Timer (RFPT). See *Component TR Structure* for additional details.
- If support is indicated by the TR *OpCode Set Structure* associated with the source subnet, a request packet may be translated into multiple request packets within the destination subnet. *Series of Packets between a Requester, a TR, and a Responder* illustrates a LDM 1 Read Request being translated into multiple smaller Read request packets targeting the same component or multiple destination components in a one or more destination subnets.
- A TR shall not track the execution results of a non-idempotent request packet. The Requester and Responder shall support the semantics specified in *Non-idempotent Request (NIR)*.
- A TR shall not translate or relay link-local packets between subnets.

- A TR may support 2-4096 interfaces.
  - A component with an integrated TR may support interfaces that do not support packet relay. For example, interfaces to / from component logic operating above the TR packet relay logic.

5                   **Developer Note:** Developers are strongly encouraged to support homogeneous interfaces, i.e., the supported signaling rates, the number of transmit lanes, the number of receive lanes, etc. should be the same on all interfaces used to transmit packets associated with the TR.

- If a TR supports multicast packet relay, then:
  - A TR may drop multicast packets if conditions warrant (see *Multicast*).
  - All components that directly or transparently participate in a multicast group shall be configured to use the same MGID, i.e., a TR shall not translate a request packet's MGID field.
  - If a TR supports *Reliable Multicast*, then it shall act as a proxy for the source subnet's Requester and aggregate all destination subnet response packets prior to transmitting a *Reliable Multicast Acknowledgment* to the source subnet's Requester.
- A TR should silently discard an end-to-end packet with stomped ECRC (see *End-to-end CRC (ECRC)*).
- A TR is managed through a *Component TR Structure*.
- Unless explicitly stated otherwise by this specification, a TR shall process and validate packets per *Transparent Router Unicast Packet Processing*.
- If relaying an explicit OpClass packet, then a TR shall update each packet's Congestion field using the same logic as a switch would, i.e., to reflect any congestion experienced and to decrement the Deadline sub-field to reflect the TR processing and TR packet residence time.

20                   **Developer Note:** Solutions that incorporate TRs are encouraged to use components that support *Link-Level Reliability (LLR)* to mitigate end-to-end performance impacts caused by transient packet errors.

- For each supported subnet, a TR shall contain a Requester ZMMU (or equivalent). Each Requester ZMMU is used to translate a request packet from the source subnet to one or more destination subnets. The request packet's Address field is used as the input address to locate a *Requester PTE* that is modified as illustrated in *TR Requester PTE*.
  - Common TR Requester PTE fields shall be as specified in *Requester PTE*.
  - The TR Index field is used to locate the per destination subnet's *Component Destination Table Structure* and *Component PA (Peer Attribute) Structure* used to generate and validate packets and the Request Tracking Table (RTR) table used to track request packets relayed to the destination subnet.
  - If request packet translation fails, then, if applicable, the TR shall transmit a *Standalone Acknowledgment* (Reason = the detected error).

Local Destination	TC	Write Mode	NSE	LPE	PS	FF	PME	WPE	SKE	CE	CCE	RKE	DT	ST	D-ATTR	ET
ADDR [63:12]																
R-Key	Security Key ID										CO	TR Index				

40                   Figure 10-4: TR Requester PTE

Table 10-1: TR Requester PTE Fields

Field Name	Size (bits)	Access	Description
Local Destination	12	RW	The contents of this field depend upon the D-ATTR field.
D-ATTR	3	RW	<p>This field indicates how to interpret the Local Destination and Global Destination fields.</p> <p>0x0—Invalid Entry</p> <p>0x1—Interleave Table Index</p> <p>0x2—Unicast Single-subnet Responder CID. Explicit OpClass packets shall set GC bit = 0b. The Local Destination field shall contain the destination subnet Responder's CID.</p> <p>0x3—Reserved</p> <p>0x4—Multicast Single-subnet Group Identifier. Multicast OpClass packets shall set GC = 0b to indicate the multicast group spans only the local subnet. The Location Destination field shall contain the local subnet's Multicast Group Identifier.</p> <p>0x5-0x7—Reserved</p>
TR Index	4	RW	<p>The field is an index into TR Table (see <i>Component TR Structure</i>).</p> <p>The TR Index field is used to locate the per destination subnet's <i>Component Destination Table Structure</i> and <i>Component PA (Peer Attribute) Structure</i> used to generate and validate packets and the Request Tracking Table (RTR) table used to track request packets relayed to the destination subnet.</p>
CO	2	RW	<p>If the Requester and the Responder do not support a common OpClass for read and write requests, then this field indicates the OpClass to use to translate the request packet.</p> <p>0x0—P2P-Core</p> <p>0x1—P2P-Coherency</p> <p>0x2—Core 64</p> <p>0x3—Reserved</p>

- Each RTR entry contains a subset of the original request packet's protocol fields (e.g., the [SCID, Tag, Access Key, OpClass]) and a subset of the translated request packet's protocol fields (e.g., the [DCID, Tag]). The combination of these fields enables the TR to correlate response packets with the original request packet and to translate the response packet into the Requester's expected response packet (recorded OpClass is used to indicate if the packet format needs to be translated to the original request packet's OpClass).
  - If response packet translation fails, then the TR shall silently discard the packet.
  - This document specifies RTR functional semantics; all other aspects are outside of this specification's scope.

- Requester ZMMU, *Component Destination Table Structure*, and *Component PA (Peer Attribute) Structure* entries may be updated at any time.
  - A TR may pause processing or silently discard any request packet impacted by an update. A TR relies upon Requesters to retransmit any silently discarded packets.
  - A TR may release an RTR entry associated with any Outstanding Request Packet that is impacted by an update.
  - The TR shall silently discard any subsequently received response packets associated with an update that precludes response packet delivery to the original Requester.
- Once packet translation is completed, the TR shall dynamically generate a new PCRC and ECRC and place these new values in the respective fields in the translated request or response packet prior to completing packet transmission through the egress interface.
- A TR shall ensure that the Requester ZMMU, ancillary structures, data and control paths, etc. are protected at all times by implementation-specific data integrity techniques.
- A TR shall ensure all in-flight packets are protected at all times by implementation-specific data integrity techniques that provide equal or better data protection than what is provided by the protocol.

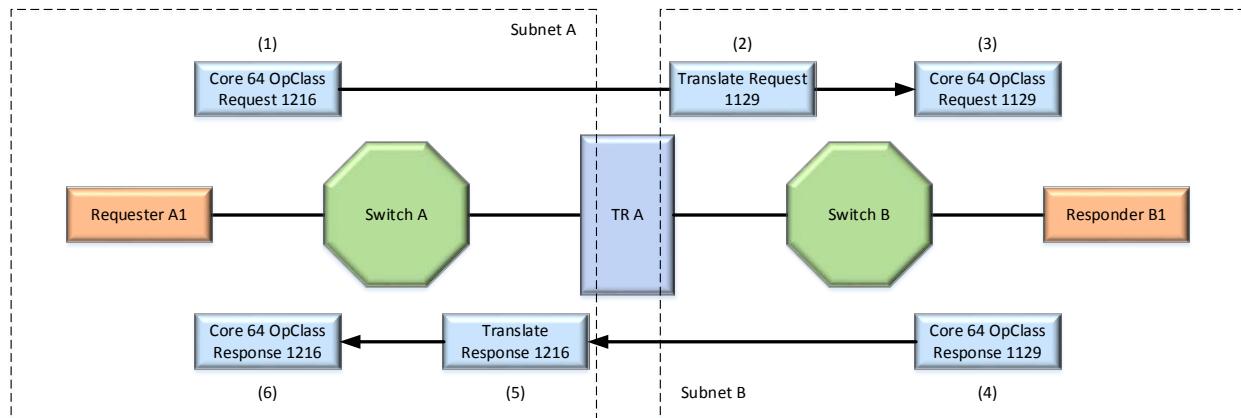


Figure 10-5: Example Packet Flow between a Requester, a TR, and a Responder

Using, *Example Packet Flow between a Requester, a TR, and a Responder*, the following steps are taken to transmit request and response packets between subnet A and subnet B:

1. Requester A1 transmits (1) request 1216 to TR A.
  - a. TR A generates a local RTR and translates (2) the request packet.
    - i. The TR uses the Requester PTE TR Index to locate the destination subnet's *Component Destination Table Structure*. The Local Destination field is used to index the Component Destination Table structure to identify the potential egress interfaces and the associated VC.
    - ii. The TR uses the Requester PTE TR Index to locate the destination subnet's *Component PA (Peer Attribute) Structure*. The Local Destination field is used to index the supported *Component PA (Peer Attribute) Structure*'s tables to translate specific request packet protocol fields, e.g., the Access Key, Next Header, etc., and to take applicable security actions.
    - iii. The TR uses the Requester PTE TR Index to locate the destination subnet's RTR, and to allocate and populate an RTR table entry to reflect the translated request packet.
  - b. TR A transmits (3) the translated request packet to Responder B1.

- 5
- c. Responder B1 executes the request packet, and transmits (4) the response packet to TR A.
  - d. TR A identifies the RTR, and translates (5) the response packet.
    - i. The TR uses the TR Index in ingress interface's *Interface Structure* to locate the RTR associated with the source subnet.
    - ii. The TR uses response packet protocol fields, e.g., [SCID, Tag], to locate the RTR table entry and translate the response packet.
  - e. TR A transmits (6) the response packet to Requester A1.

10 *Series of Packets between a Requester, a TR, and a Responder* illustrates how a TR translates a series of request and response packets. The term "destination subnet" refers to the subnet of the destination component associated with a given packet.

15

- Request T0 is translated into destination subnet TR REQ 0.
- TR RSP 0 is translated into destination subnet response RSP 0.
- Request packets T1-T4 are translated into destination subnet request packets TR REQ 1-TR REQ4.
- TR RSP 1-TR RSP4 are translated into destination subnet response packets RSP 1-RSP 4.

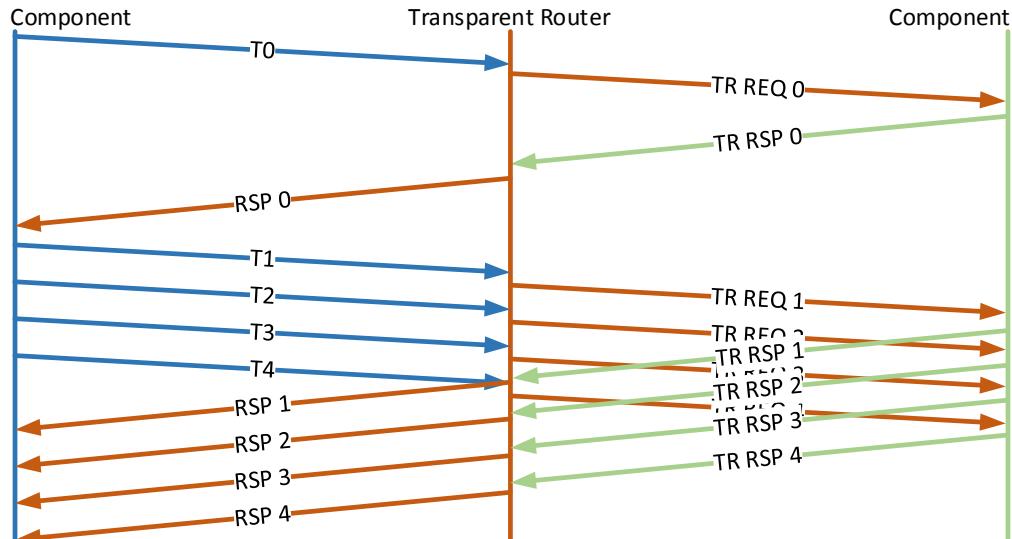


Figure 10-6: Series of Packets between a Requester, a TR, and a Responder

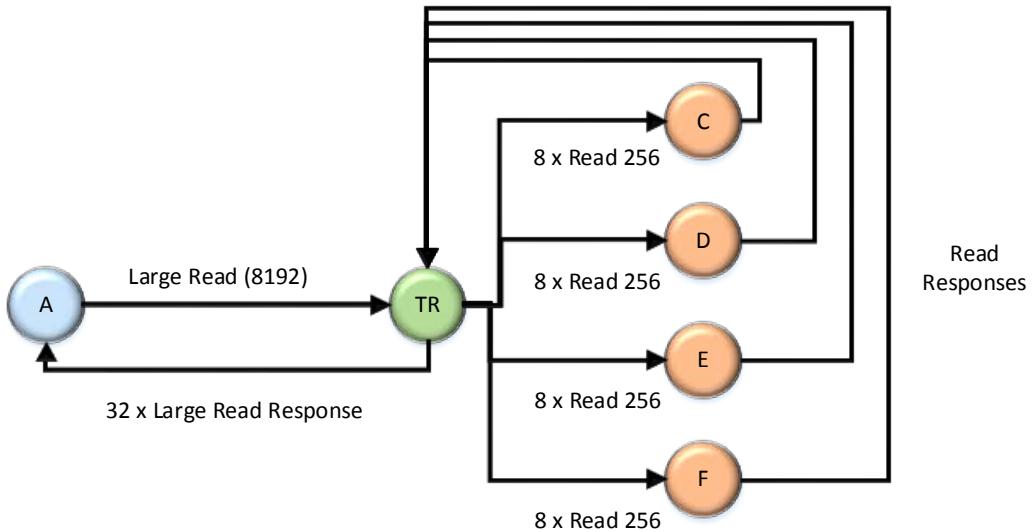


Figure 10-7: One Logical Packet, Multiple Physical Packets

## 10.1. TR-to-TR Communications

5 TR-to-TR communication involves multiple components and one or more packet translations. The following figures illustrate example request-response exchanges that traverse two TRs.

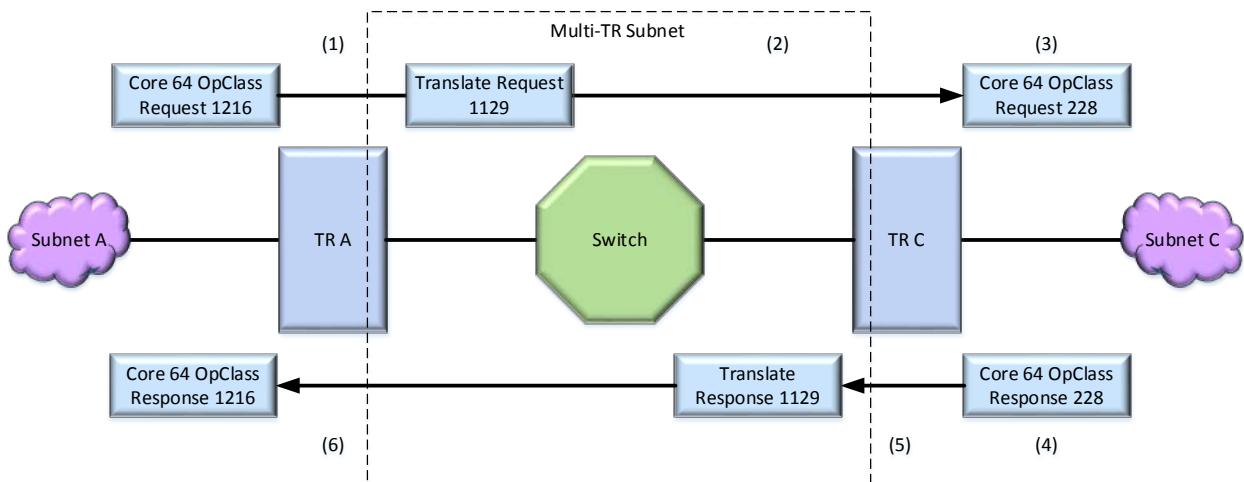


Figure 10-8: Example TR-to-TR Request-Response Exchange

10 Using *Example TR-to-TR Request-Response Exchange*, the following steps are taken to transmit request and response packets between subnet A and subnet C:

- 15
1. TR A's Requester ZMUU is configured such that TR A is able to determine that the request packet is destined for a Responder in subnet C.
  2. TR A receives request 1216 for the Requester. To forward the packet across the multi-TR subnet:
    - a. TR A generates a local RTR, translates (1) the request packet, and transmits (2) the packet to TR C.
    - b. TR C generates a local RTR, translates (3) the request packet into a subnet C-local request packet, and transmits the packet to the Responder.

- 5
- c. The Responder executes the request packet, and transmits a response packet to TR C.
  - d. TR C receives the corresponding response from the subnet-C Responder, and translates (4) the packet into the TR A response packet.
  - e. TR C translates the response packet (5), transmits the packet to TR A, and releases the local RTR.
  - f. TR A receives the response packet, translates (6) the packet, transmits the response packet to the original Requester, and releases the local RTR.

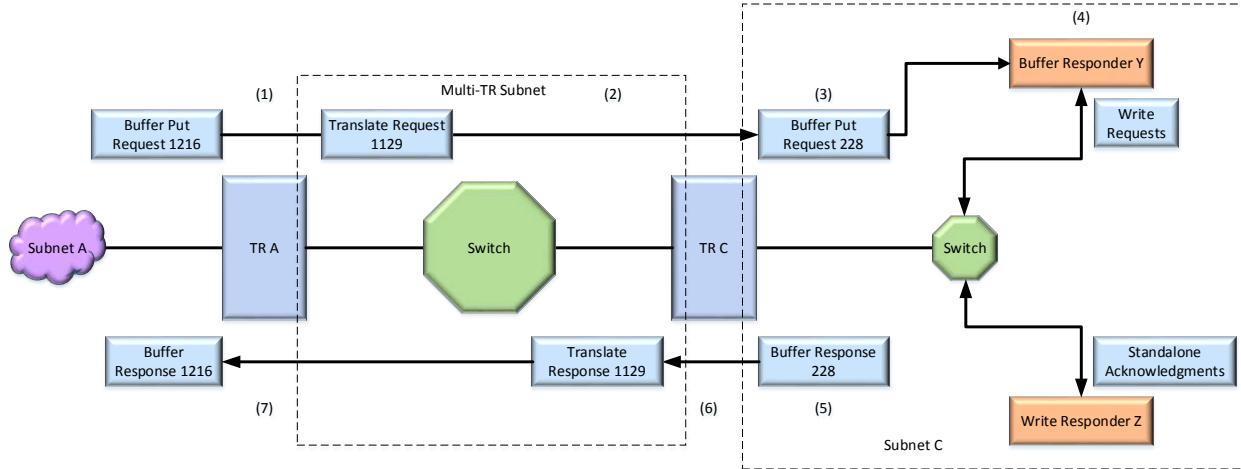


Figure 10-9: Example TR-to-TR Request-Response Exchange, Remote Buffer Operation

10 *Example TR-to-TR Request-Response Exchange, Remote Buffer Operation* illustrates a remote Buffer Put operation where the Requester believes it is migrating data within a single Responder component.

- 15
1. TR A receives request 1216 for the Requester. To forward the packet across the multi-TR subnet:
    - a. TR A generates a local RTR. In this particular case, TR A needs to examine the two addresses (A and B) within the Buffer Put request packet to confirm both map to subnet C.
    - b. TR A translates (1) the request packet (in general, only the fields required to traverse the multi-TR subnet and the Tag field need to be translated), and transmits (2) the request packet to TR C.
    - c. TR C generates a local RTR, translates (3) the request packet into a subnet C-local Buffer Put request packet, and transmits the packet to the Buffer Responder Y.
    - d. The Buffer Responder Y executes the Buffer Put request packet, and generates (4) a series of subnet C-local write request packets to move buffer A in Responder Y to buffer B in the Responder Z.
    - e. Once Buffer Responder Y successfully executes the Buffer Put request packet, it generates and transmits (5) a *Standalone Acknowledgment* to TR C.
    - f. TR C translates the *Standalone Acknowledgment* (6), transmits the packet to TR A, and releases the local RTR.
    - g. TR A receives the *Standalone Acknowledgment*, translates (7) the packet, transmits the *Standalone Acknowledgment* to the original Requester, and releases the local RTR.

20

25

30

# 11. Link Specification

This section specifies the link protocol and link state machine. The link protocol is a point-to-point protocol used to exchange link-local packets between the two attached interfaces.

The link state machine closely parallels the physical layer abstraction state machine (see *PHY States*).

## 5 11.1. Link States

Table 11-1: Link States and Descriptions

Link State (LS)	Description
L-Up	<p>The Link Up (L-Up) state is the normal operation state used to exchange link-local and end-to-end packets.</p> <p>Whenever a link transitions to L-Up, the interface shall transmit a minimum of one Link Flow-control packet per enabled VC.</p> <p>Upon transitioning from L-Down to L-Up, the interface shall complete the following steps in order:</p> <ol style="list-style-type: none"><li>1. An interface shall perform a Tx Packet Alignment <i>Link CTL</i>-Link ACK <i>Link CTL</i> packet exchange.</li><li>2. If <i>Interface I-CAP 1 Control</i> Auto Peer-Attribute <i>Link CTL</i> Exchange == 1b, then the interface shall perform a Peer-Attribute <i>Link CTL</i>-Link ACK <i>Link CTL</i> packet exchange. If this exchange reveals the peer interface's attributes precludes interoperability, then the link shall transition to L-Down.</li><li>3. If <i>Link-level Reliability (LLR)</i> is supported and enabled, then the interface shall perform LLR initialization. LLR initialization shall be completed (success or failure) prior to exchanging <i>Explicit Flow Control</i> packets or transmitting any end-to-end packet.</li><li>4. If the interface supports <i>Explicit Flow Control</i>, then it shall transmit a minimum of one <i>Explicit Flow Control</i> packet per enabled VC (<i>Interface Structure</i> HVE).</li><li>5. If the interface supports <i>In-band Discovery and Initialization</i> and <i>Interface I-Control Disable Link</i> RFC == 0b, then the interface shall start transmitting <i>Link RFC</i> packets.</li><li>6. If the interface supports <i>In-band Discovery and Initialization</i> and the directly-attached interfaces support multiple common OpClasses, then to enable <i>In-band Management</i> packets to be exchanged, management shall set <i>Interface I-Control OpClass Select</i> to indicate which OpClass shall be used on this interface, and shall subsequently perform a Peer-Set-Attribute <i>Link CTL</i>-Link ACK <i>Link CTL</i> packet exchange. If a single common OpClass is supported, then this step should be skipped.</li></ol> <p>If the interface supports <i>Explicit Flow Control</i>, then upon transitioning from L-LP to L-Up, the interface shall transmit a minimum of one <i>Explicit Flow Control</i> packet per enabled VC.</p> <p>Links whose physical layer does not support <i>IDLE Symbols</i> shall transmit <i>Link Idle</i> packets whenever not transmitting a different link-local packet type or end-to-end</p>

Link State (LS)	Description
	<p>packet. This ensures both interfaces remain synchronized as to where a given packet starts.</p> <p>If a link support the PCIe LTSSM, then the PCIe L0, L0s, Recovery, and Loopback states shall be mapped to the L-Up state for Gen-Z configuration, management, and protocol purposes.</p> <p>Physical layer states are specified in <i>PHY States</i>.</p>
L-Down	<p>The Link Down (L-Down) state is used to indicate that a link is non-operational.</p> <p>Prior to dynamic mechanical module removal, the link shall transition to L-Down. This transition should be completed prior to power being removed from the component.</p> <p>If a link support the PCIe LTSSM, then the PCIe Disabled, Hot Reset, Configuration, Polling, and Detect states shall be mapped to the L-Down state for Gen-Z configuration, management, and protocol purposes.</p>
L-LP	<p>The Link Low Power (L-LP) state corresponds to any of the PHY-LP [1-4] states (physical layer low-power non-operational states). If a physical layer does not support PHY-LP [1-4], then the L-LP state shall not be entered.</p> <p>While in this state, the link remains initialized but shall not transmit or receive link-local or end-to-end packets.</p> <p>If a link support the PCIe LTSSM, then the PCIe L1 / L1 sub-states, L2, and L3 states shall be mapped to the L-LP state for Gen-Z configuration, management, and protocol purposes.</p>
L-Up-LP	<p>The Link Up Low Power (L-Up-LP) state corresponds to any of the PHY-Up-LP [1-4] states (physical layer low power operational states).</p> <p>If a physical layer does not support the PHY-Up-LP [1-4] states then the L-Up-LP state shall not be entered.</p> <p>While in this state, the link may transmit and receive link-local and end-to-end packets, but operates at a lower power / lower performance level.</p>

The link state machine shall be as illustrated in *Link State Machine*. Upon power up or a Warm or Full *Interface Reset*, the link starts in the L-Down state.

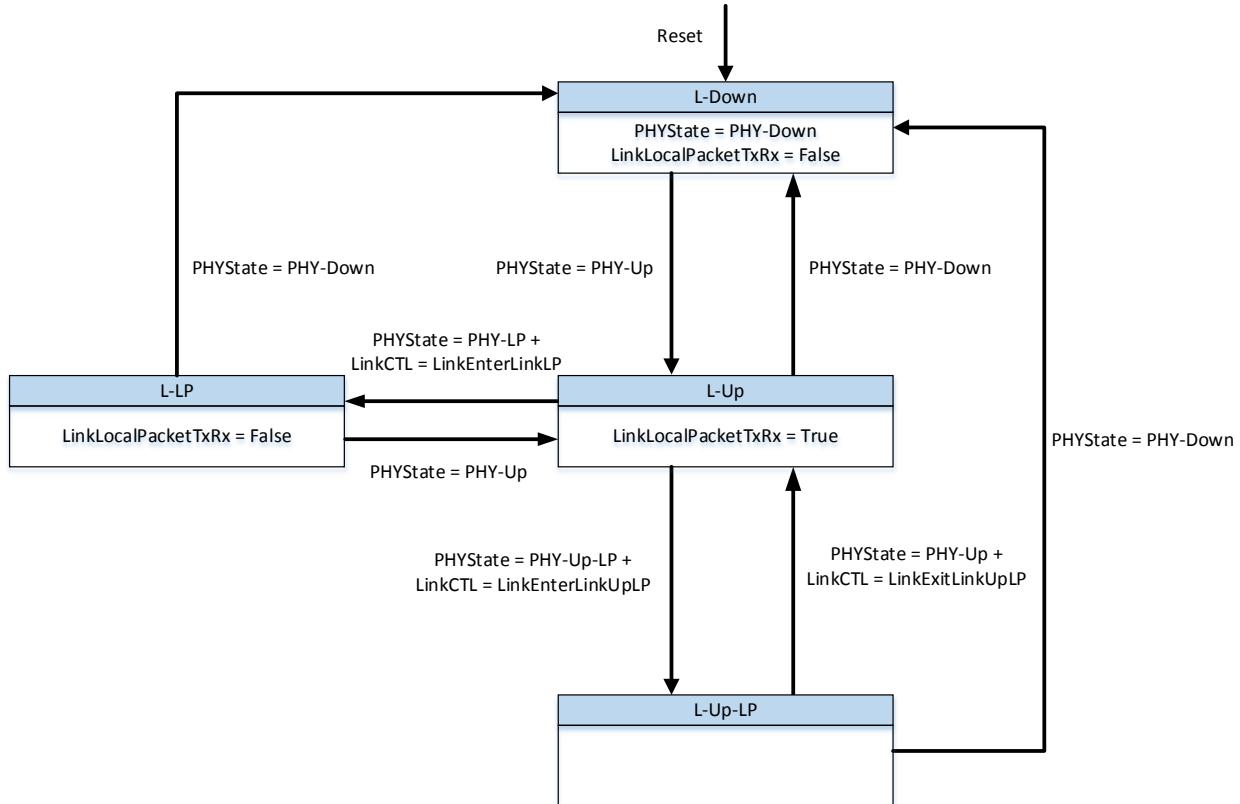


Figure 11-1: Link State Machine

Link Reset—an internal signal to reset the link. This signal is generated at power up or as a result of a component or interface reset event.

- 5 LinkLocalPacketTxRx—if True, then the link may transmit link-local or end-to-end packets. If False, then the link shall not transmit link-local or end-to-end packets.

When Link Reset is asserted, the link shall take the following steps:

- If the link state is L-Up or L-Up-LP, then the link shall exchange Link CTL (Link Reset Event) packets to inform the peer interface that the link is transitioning to L-Down due to a reset event.
- 10 • If the Link CTL packet exchange is completed or if the Link CTL packet exchange fails or cannot be performed, then the physical layer shall transition to PHY-Down.
- Once the physical layer transitions to PHY-Down, the link shall:
  - o Silently discard all scheduled or queued end-to-end and link-local packets.
  - o Silently discard any new packets.
  - o Reset all *Link-Local Flow Control* tracking logic and resources to the uninitialized state.
  - o Reset all *Link-level Reliability* tracking logic and resources to the uninitialized state.
  - o Reset all *Link CTL* tracking logic and resources to the uninitialized state.

**Developer Note:** A component's internal logic and packet transmission can fail mid-transmission yet the associated interface link and physical layers remain operational. To detect this condition, receivers can run an implementation-specific timer that starts with the receipt of a packet's first byte. If the timer expires prior to the receipt of the last byte of the packet's CRC, then it is likely that the transmitting component has failed.

## 11.2. Link Protocol

The link protocol is used to exchange link-local packets between two directly-attached interfaces. The link itself is primarily managed through the *Interface Structure*, which leaves the link protocol to simple packet exchanges focused on exchanging flow-control credits and link control packets.

## 11.3. Link-local Packet Protocol Formats

There are two link-local packet sizes—32-bit and 128-bit. The 32-bit packet size is used only with one operation type, and the 128-bit packet size is used for all others. The 128-bit general packet format is illustrated in *128-bit Link-local General Packet Format*. A link-local packet is transmitted starting at bit 0, and consecutively transmits subsequent bits 1 through bit 31 or 1 through bit 127 (the last bit in the packet).

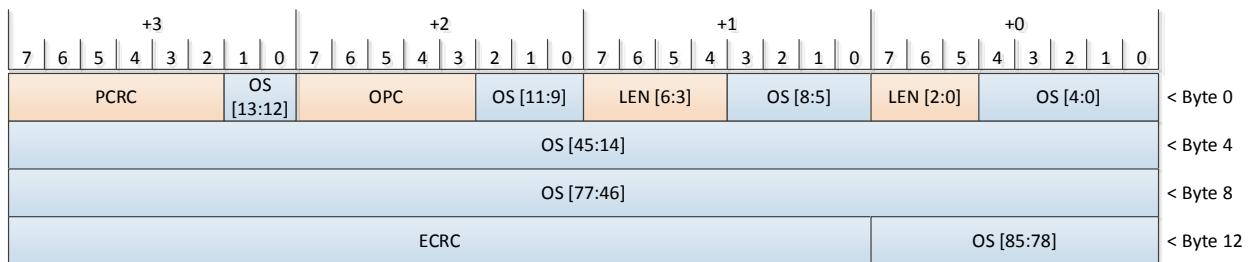


Figure 11-2: 128-bit Link-local General Packet Format

Table 11-2: Common Link-Local Packet Protocol Fields

Field Name	Field Abbreviation	Size (bits)	Description
Length	LEN	7	A link-local packet shall set Length = 0x7F. Actual packet length is delineated by the OPC.
Operation Code	OPC	5	OpCode implies packet length and payload size.

Table 11-3: Common Link-Local 128-bit Packet Protocol Fields

Field Name	Field Abbreviation	Size (bits)	Description
OpCode-Specific Prelude CRC	OS	-	Operation code-specific fields
	PCRC	6	Integrity field that protects select protocol fields that determine packet type and length. The protected fields vary by packet type but are always at the same bit locations.  PCRC shall be dynamically generated at the source interface, and validated at the destination interface.  See <i>Prelude CRC (PCRC)</i> .

Field Name	Field Abbreviation	Size (bits)	Description
End-to-end CRC	ECRC	24	<p>Packet Data integrity field</p> <p>ECRC shall be dynamically generated at the link transmitter interface and validated at the peer link receiver interface.</p> <p>See <i>End-to-end CRC (ECRC)</i>.</p>

Link-local packets have the following attributes:

- Unless specifically stated otherwise in this specification, link-local packets shall not be acknowledged.
- Link-local packets shall not be subject to flow control, and shall not consume flow-control credits.
- Link-local packets shall be transmitted without regard to virtual channels.
- Link-local packets shall be transmitted and received only by the interfaces directly attached to the link, i.e., link-local packets shall not be relayed to another link.
- Link-local packets are received and executed in the order they are transmitted.
- Link-local packets shall be protected by a 6-bit PCRC and a 24-bit ECRC.
- A source interface shall dynamically generate the PCRC and place it into the transmitted link-local packet's PCRC field.
- A source interface shall dynamically generate the ECRC and place it into the transmitted link-local packet's ECRC field.
- Link-local packets may be interleaved with end-to-end packets.
- Multiple link-local packets may be consecutively transmitted on a link.
- Link-local packets shall be limited to 32-bit and 128-bit formats.
- Link-local packets may be discarded for various reasons including data integrity errors, interface or operation error detection, unsupported operation type, etc.
- Receivers should provision sufficient resources to ensure link-local packet processing does not diminish link performance.

## 11.4. OpCodes

A link-local packet OpCode is a 5-bit field that represents a *Link-Local Packet OpCodes*. The OpCode Description column entry provides a sub-section link to the associated details.

Table 11-4: Link-Local Packet OpCodes

OpCode Name	OpCode Encoding	Packet Size (bits)	OpCode Description
Link Single-VC FC LLR ACK	0x0	128	See <i>Explicit Link Flow Control</i>
	0x1	128	See <i>Explicit Link Flow Control</i>
	0x2	128	See <i>Link-local RFC Packet</i>

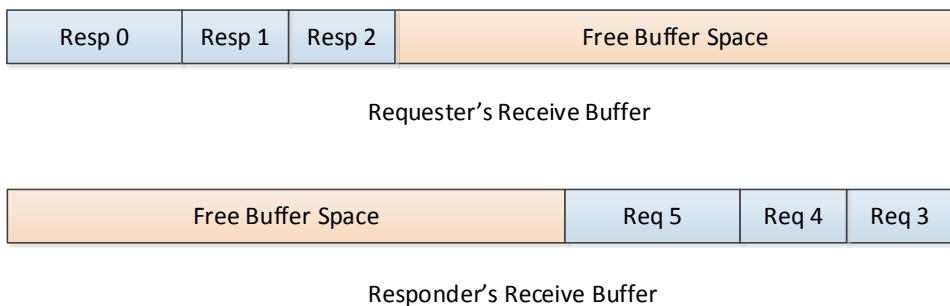
OpCode Name	OpCode Encoding	Packet Size (bits)	OpCode Description
Link SR	0x3	128	See <i>Link Synchronization</i>
Link RR	0x4	128	See <i>Link Resynchronization Request (Link RR)</i>
Link RC	0x5	128	See <i>Link Resynchronization Completion (Link RC)</i>
Link CTL	0x6	128	See <i>Link CTL</i>
LLR	0x7	128	See <i>Link-level Reliability (LLR)</i>
Reserved	0x8-0x1B	-	Reserved
Link Idle	0x1C	32	See <i>Link Idle</i>
Reserved	0xD-0x1E	-	Reserved
Link RP	0x1F	128	See <i>Link Resynchronization Pattern (Link RP)</i>

## 11.5. Link-Local Flow Control

Each receiving interface has a finite buffer space to receive end-to-end packets. To prevent buffer overflow and subsequent end-to-end packet loss, each transmitter tracks its peer receiver's available buffer space. Tracking is accomplished using implicit flow control or explicit flow control.

### 5 11.5.1. Implicit Flow Control

Implicit flow control shall be constrained to point-to-point topologies consisting of two components—a component on one end that is exclusively the Requester and a component on the other end that is exclusively the Responder. A link using implicit flow control does not exchange flow-control packets. Instead, implicit flow control relies on the Requester to prevent receive buffer overflow at each link interface. *Implicit Flow-Control Example* illustrates the Requester's and Responder's receive buffers. The Requester's buffer contains three response packets and some amount of free space capable of containing the response packets to the three request packets contained in the Responder's receive buffer.



15

Figure 11-3: Implicit Flow-Control Example

- An interface that supports the P2P-Core or the P2P-Vendor-defined OpClass may support implicit flow control. If *Interface I-CAP 1 Implicit Flow Control Support == 1b*, then the interface supports implicit flow control.

- To enable implicit flow control, the following shall be true:
  - Both directly-connected interfaces support implicit flow control.
  - A Requester shall not transmit unreliable request packets, e.g., a P2P-Core Unreliable Write.
  - Management shall set *Interface I-CAP 1 Control Omit P2P Standalone Acknowledgment* = Ob.
- If implicit flow control is enabled, then flow-control packets shall not be transmitted by either interface; any flow-control packet shall be silently discarded.
- The Requester shall track each interface's available receive buffer space (see *Interface Structure Max Implicit FC Credits*). The Requester shall transmit a request only if both the Responder's interface has sufficient receive buffer space to receive the request packet and the Requester's interface has sufficient receive buffer space to receive the corresponding response packet.

### 11.5.2. Explicit Flow Control

Explicit flow control may be used in any topology, and is the default link-local flow-control mechanism for end-to-end packets. Explicit flow control uses link-local packets to exchange each receiver's present flow-control credit state, which is used by the corresponding transmitter to determine if the receiver has sufficient buffer space to receive an end-to-end packet. Within this section, the transmitter refers to the interface injecting the end-to-end packet and the receiver refers to the interface consuming the end-to-end packet.

The following are the features and requirements associated with flow-control packets:

- Flow-control packets shall be transmitted only when link state L-Up or L-Up-LP.
  - When the link state transitions to L-Down, for all supported VC, all flow-control credits shall be released and all flow-control tracking logic shall be set to the uninitialized state.
  - Flow-control packets may be interleaved with end-to-end packets corresponding to VCs with available flow-control credits.
- The maximum number of flow-control credits per VC shall be  $\text{Max\_FC} = 2^{16}$ .
- Each flow-control credit represents a 16-byte unit of buffer space, i.e., Credit Size = 16. Packet transmission shall consume one or more flow-control credits. Partial flow-control credit consumption shall not be allowed. If the packet's length is not an integer 16-byte multiple, then the number of flow-control credits consumed shall be equal to the packet's length rounded up to the next integer 16-byte multiple.
- Flow-control credits shall reflect the number of currently available credits for a particular VC. For example, if an interface's VC currently has physical buffer resources to receive 256 bytes (payload and / or protocol), then the number of available credits for that VC is 16.
  - The phrase 'currently available' refers to the time when the flow-control packet is scheduled, and need not reflect any updates between the time it is scheduled and the time it is transmitted. For example, at time T, a flow-control packet reflects 16 credits. At time T+1, 16 additional credits become available which are communicated via a new flow-control packet that reflects 32 credits. To reflect such increases, multiple flow-control packets may be scheduled at any given time.
- Flow-control credits provisioned per supported VC may vary, e.g., the number provisioned for VC0 may differ than the number provisioned for VC1.
- For each VC, an interface shall support at least the minimum number of flow-control credits ( $\text{VC}_k \text{ MIN}$ ) to receive at least one packet of the largest supported packet type (protocol plus associated

payload). For example, a Core 64 Write request packet carrying a 256-byte payload with every optional field present requires 20 flow-control credits to transmit.

- Flow-control credits may be statically or dynamically (e.g., *Adaptive Flow Control*) provisioned per VC. Credit provisioning policies and controls are outside of this specification's scope.
  - The number of flow-control credits per enabled VC may be dynamically adjusted while the link is active. The number of credits advertised per enabled VC may be dynamically increased. With the exception of the link transitioning to L-Down, the number of advertised credits shall not be dynamically decreased until they are consumed by the transmitter, at which time, the receiver interface may return fewer credits than freed in a subsequent flow-control packet. Any reduction, e.g., a flow-control packet that effectively reduces the previously advertised flow-control credits, other than when the link is transitioning to the L-Down state shall be treated as a non-transient error.
- If an interface receives flow-control credits for an unsupported or non-enabled VC, then the interface shall silently discard the packet.
- Each receiver shall maintain a single, interface-local FCTT (Flow-Control Transmission Timer).
  - The FCTT period shall be  $2^{24}$  UI.
  - An interface should transmit flow-control packets whenever its receiver has available flow-control credits and no other higher precedence packets are awaiting transmission.
  - While link state is L-Up or L-Up-LP,
    - For each enabled VC, an interface shall transmit at least one flow-control packet every FCTT period.
    - For each enabled VC, an interface should transmit one flow-control packet at every  $(\text{Max\_FC} * \text{Credit Size} * 8 / \text{Max\_Lanes})$  UI, where Max\_Lanes represents the maximum number of enabled receiver lanes.
  - FCTT shall be suspended when the link state transitions to L-LP or L-Down and during physical layer retraining. When the link state transitions L-Up, the FCTT shall be reset to zero and shall resume running.
  - If an interface fails to receive at least one flow-control packet per enabled VC from the link's peer interface for two consecutive FCTT time periods, then the interface shall initiate physical layer retraining.
- Periodic transmission of flow-control packets does not guarantee a transmitter can make forward progress, i.e., if the number of available flow-control credits on a given VC does not increase such that there are sufficient credits to transmit the next pending end-to-end packet, then the transmitter stalls end-to-end packet transmission on that VC. The effects of a stalled transmitter impact not only the peer interface, but can negatively impact peer component operation and cause congestion spreading.
  - A Requester component shall be able to receive response packets to all Outstanding Request Packets without relying on more than temporary VC backpressure as a result of received end-to-end packet processing and the subsequent return of the associated flow-control credits.
  - A Responder component shall be able to receive request packets up to *Core Structure* Max Requests without relying on more than temporary VC backpressure as a result of received end-to-end packet processing and the subsequent return of the associated flow-control credits.
  - If a Responder component supports NIR operations, it should not apply VC backpressure due to NIR resource exhaustion.
  - A switch component ensures forward progress per *Switches*.

- Available flow-control credits are calculated using a sliding window algorithm. For each supported VC:
  - The receiver on each link end maintains a logical circular number space (see *Mid and Wrap-Around Flow-Control Window Examples*). The logical circular number space size shall be `Max_FC`.

5

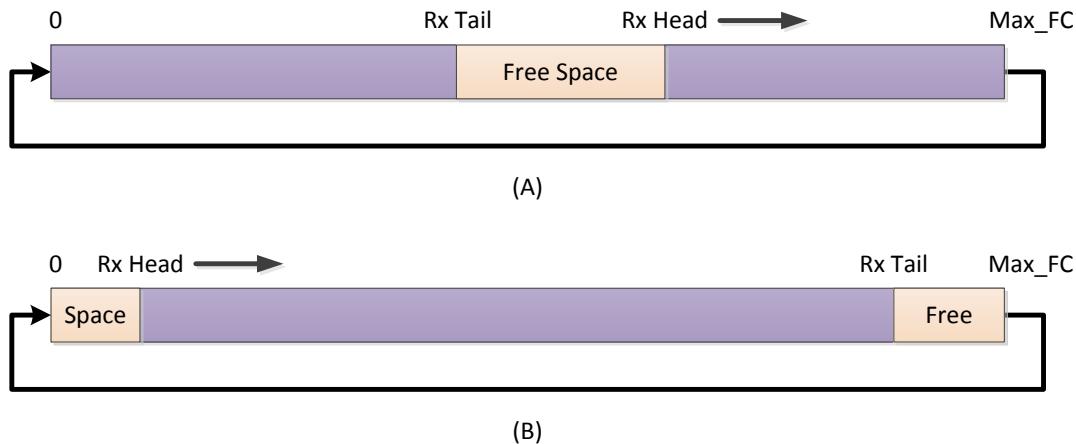


Figure 11-4: Mid and Wrap-Around Flow-Control Window Examples

- The receiver slides a logical window across this number space. The window's leading edge is referred to as the head (Rx Head) and the trailing edge as its tail (Rx Tail). As receive buffer space becomes available, the receiver increments the window's head by the corresponding number of flow-control credits. As buffer space is consumed, the receiver increments the window's tail by the corresponding number of flow-control credits. For example, if 17 flow-control credits of additional buffer space became available, then the receiver increments  $\text{Rx Head} = (\text{Rx Head} + 17) \bmod \text{Max\_FC}$ . If an end-to-end packet arrives that consumed 8 flow-control credits, then the receiver increments  $\text{Rx Tail} = (\text{Rx Tail} + 8) \bmod \text{Max\_FC}$ .
- The transmitter on each link end maintains an analog of the receiver's logical circular number space and sliding window. The window's leading edge is referred to as the head (Tx Head) and the trailing edge as its tail (Tx Tail). When the receiver advertises additional flow-control credits (packet's Rx HD field), the transmitter sets  $\text{Tx Head} = \text{Rx HD}$ . When the transmitter transmits an end-to-end packet, the transmitter increments the window's tail by the number of flow-control credits consumed, i.e.,  $\text{Tx Tail} = (\text{Tx Tail} + Y) \bmod \text{Max\_FC}$ . The transmitter calculates the available flow-control credits by calculating  $\text{Diff} = (\text{Tx Head}, \text{Tx Tail})$ .
- An end-to-end packet can suffer a transient error during transmission. As a result, the receiver will discard the packet. If the transient error was a PCRC error, then the receiver does not know how many flow-control credits would have been consumed by the packet's receipt. To ensure that the receiver and the transmitter remain in sync, each flow-control packet contains the transmitter's present understanding of the receiver's tail. If the transmitter's understanding is incorrect, then the receiver updates its Rx Head value to reflect the current flow-control credits and transmits a new flow-control packet to the transmitter. Upon receipt of a new flow-control packet, the transmitter takes the following actions:
  - If  $\text{Rx Tail} < \text{TRx TL}$ , then:
    - $\text{Delta} = \text{Diff}(\text{TRx Tail}, \text{Rx Tail})$

10

15

20

25

30

35

- Rx Head += Delta (modulo Max\_FC)
- Rx Tail = TRx TL
- Transmit a new flow-control packet that reflects updated Rx Head value
- Upon transitioning from L-Down to L-Up, for each enabled VC:
  - The receiver calculates the available free buffer space and sets its Rx Head and Rx Tail variables to indicate the available free buffer space. For example, if 1024 bytes are available, then Rx Head = 64 and Rx Tail = 0.
  - The transmitter sets its Tx Head and Tx Tail variables to zero to indicate it has zero flow-control credits.
  - The receiver transmits at least one flow-control packet with the Rx HD field set to the receiver's corresponding local head variable.
  - The transmitter updates its Tx Head = Rx HD and Tx Tail remains zero. The transmitter has Diff (Tx Head, Tx Tail) = 64 flow-control credits.
- Upon transitioning from L-LP to L-Up, for each enabled VC, each interface shall schedule at least one flow-control packet.
- A transmitter shall schedule an end-to-end packet only if the available flow-control credits represent sufficient receive buffer space to contain the entire packet (protocol fields plus any payload data).
  - If a transmitter transmits an end-to-end packet without having sufficient flow-control credits, then the receiver shall silently discard the packet without consuming any flow-control credits. Upon detecting this error, the receiver should transmit a flow-control packet corresponding to the associated packet's VC.
- A receiver may transmit a flow-control packet for  $VC_k$  whenever there is a non-zero number of unadvertised flow-control credits. At a minimum, whenever the number of unadvertised flow-control credits for  $VC_k$  is greater than ( $VC_k$  maximum provisioned flow-control credits DIV 3), a receiver shall transmit a flow-control packet for  $VC_k$ .
- An interface shall support the Link Single-VC FC LLR ACK packet.
  - If an interface is enabled to exchange P2P-Core OpClass packets, then the link shall use the Link Single-VC FC LLR ACK Flow-Control Packet format with VC = 0x0.
  - A Link Single-VC FC LLR ACK packet shall use the *Link Single-VC FC LLR ACK Flow-Control Packet Format*.
    - If the interface does not support *Link-level Reliability (LLR)*, then the RSEQ and TSEQ fields shall be Reserved.
    - If the interface supports *Link-level Reliability (LLR)*, then the RSEQ and TSEQ fields shall be updated as specified for a LLR ACK.
- If an interface supports two or more VCs, then it shall support the Link Dual-VC FC operation.
  - The Link Dual-VC FC shall use the *Link Dual-VC Flow-Control Packet Format*.
- With one exception (described below), a component shall not make the return of flow-control credits conditional upon any other end-to-end packet's arrival or upon any flow-control packet's arrival on any enabled VC of any interface.
  - The one exception is a component that supports multiple VCs may delay returning flow-control credits for a VC used to transmit request packets while awaiting the return of flow-control credits needed for outbound forward progress for a VC used to transmit response packets on the same or any other interface.

- A P2P-Core component in a daisy-chain topology shall base its return of link-local flow-control credits only upon its ability to accept packets for forwarding along the daisy-chain, without regard to its ability to accept additional request packets on its own behalf.

### 11.5.2.1. Adaptive Flow Control (AFC)

5 Adaptive flow-control enables a transmitter to optimize the aggregate flow-control credits across all VCs to meet current transmission needs. For example, a receiver that supports an aggregate of 4096 flow-control credits equally distributed across 8 VCs, i.e., 512 credits per VC. Dynamic workload conditions could result in some VCs being underutilized and some being oversubscribed, i.e., congested. AFC enables a portion of the flow-control credits from the underutilized VCs (e.g., 256 credits) to be used by 10 the oversubscribed VCs to alleviate congestion. By adaptively adjusting flow-control credit usage, a receiver can avoid overprovisioning resources to meet worst-case workload behavior.

#### 11.5.2.1.1. AFC initialization and communication

15 AFC support is indicated by the *Interface I-CAP 1 Adaptive FC Credit Support* bit and enabled by the *Interface I-CAP 1 Control Adaptive FC Credit Enable* bit. If enabled, then each interface detects if its peer supports and is enabled to use AFC through the flow-control packet protocol:

- If FP == 1b, then the transmitter supports and is enabled to use AFC.
- If either transmitter transmits FP == 0b, then the interfaces shall not use AFC.
- If both transmitters transmit FP == 1b, then the interfaces shall use AFC.
- If the interfaces are enabled to use and software disables AFC, then to take effect, software shall 20 initiate a *Warm Interface Reset*. Similarly, if the interfaces are not enabled and software enables AFC, then to take effect, software shall initiate a *Warm Interface Reset*.
- If *Interface I-CAP 1 Control PCO Communications Enable* == 0b, then all VCs may participate in AFC.
  - If *Interface I-CAP 1 Control PCO Communications Enable* == 1b and *Interface Structure VC PCO Enabled Bit<sub>K</sub>* == 1b, then VC<sub>K</sub> shall not participate in AFC, i.e., VC<sub>K</sub>'s flow-control credits shall be exclusive to VC<sub>K</sub>.

#### 11.5.2.1.2. Resource Management and Packet Transmission

The following applies to each VC<sub>K</sub> enabled to use AFC:

- Irrespective of the flow-control credit source, a transmitter shall not transmit an end-to-end packet on VC<sub>K</sub> unless there are sufficient flow-control credits.
- For each VC<sub>K</sub>, a transmitter shall ensure that the VC<sub>K</sub> has at least VC<sub>K</sub> MIN flow-control credits prior to allocating any flow-control credits to AFC.
- When an end-to-end packet is transmitted on VC<sub>K</sub>, the transmitter shall consume AFC credits prior to consuming VC<sub>K</sub> MIN credits.
- Upon receipt of a flow-control packet, the transmitter shall calculate the available flow-credits and update VC<sub>K</sub> flow-control credits as follows:
  - If VC<sub>K</sub> has less than VC<sub>K</sub> MIN, then the transmitter shall increment VC<sub>K</sub>'s flow-control credits. If there are sufficient flow-control credits, then the transmitter shall increment to at least VC<sub>K</sub> MIN flow-control credits.
  - If any available flow-control credits remain, then the transmitter may increment the available AFC credits.

- The maximum number of  $VC_k$  flow-control credits shall be less than `Max_FC`, i.e.,  $FC_{redits} + VC_k MIN$  shall be less than `Max_FC`.

**Developer Note:** The following example pseudo code illustrates how explicit flow-control credits could be implemented. Further, it illustrates the incremental changes to support AFC. This example does not calculate the available flow-control credits using  $(head - tail)$ . This alternative approach improves timing and eliminates the potential for the tail pointer to pass the head pointer that could break AFC support.

```

5   force_credit_sent = new_rxredits > (FC_SZ/3);           // how to know when to force a credit return
tail_diff = (fc_update.TRx_TL - rx_tail);                  // # dropped credits is tail_from_TX - tail_in_RX
total_new_credits = (fc_update.Rx_HD - tx_head);          // new credits is new_head - old head
10
15
if (reset) {
    rx_head = FC_SZ;                                     // track's head of local FIFO, increments as FIFO pop's
    new_rxredits = FC_SZ;                                // just used to know when to send new credits
    tx_head = 0;                                         // track's head pointer sent via fc_update packets
    tx_tail = 0;                                         // track's total credits of packets sent to the link
    avail_credits = 0;                                    // tx side number of credits available to send packets
    rx_tail = 0;                                         // track's total credits of packets received from the link
} else {
    rx_head = rx_head + tail_diff + fifo_pop;
    new_rxredits = send_fc_update ? 0 : new_rxredits + fifo_pop;
    tx_head = fc_update ? fc_update.Rx_HD : tx_head;
    tx_tail = tx_tail + credits_sent;
    avail_credits = avail_credits + total_new_credits - pktcredits_sent;
    rx_tail = fc_update ? fc_update.TRx_TL : (rx_tail + pkts_received);
25
}

// Changes to support adaptive crediting (changes/additions to above pseudo code)
30
avail_credits = avail_credits + new_avail_credits - pktcredits_sent_without_pool;
pool_credits = pool_credits + new_pool_credits - pktcredits_sent_with_pool;

35
if (MAX_PKT_SZ == avail_credits)           // no credits needed for VC
    new_pool_credits = total_new_credits;
else {                                     // grab needed credits for VC, the rest to the pool
    if ((MAX_PKT_SZ - avail_credits) < total_new_credits) {
        new_avail_credits = (MAX_PKT_SZ - avail_credits);
        new_pool_credits = total_new_credits - (MAX_PKT_SZ - avail_credits);
    } else                                // all credits needed for VC
        new_avail_credits = total_new_credits;
}

```

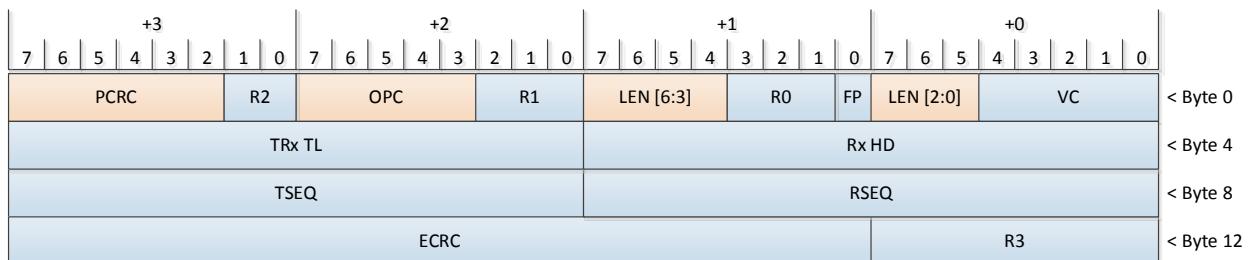


Figure 11-5: Link Single-VC FC LLR ACK Flow-Control Packet Format

Table 11-5: Link Single-VC FC LLR ACK Flow-Control Packet-specific Fields

Field Name	Field Abbreviation	Size (bits)	Description
Virtual Channel	VC	5	Indicates the virtual channel targeted by this flow-control packet.
FC Pool	FP	1	FP shall be equal to <i>Interface I-CAP 1 Control Adaptive FC Credit Enable</i> .
Receiver Head	Rx HD	16	The current location of the head of the receiver's available logical buffer space associated with the indicated VC.
Transmitter Tail	TRx TL	16	The transmitter's understanding of the peer receiver's tail associated with the indicated VC.
RSEQ	-	16	If a Link Single-VC FC LLR ACK operation, then this field contains the number of end-to-end packets (modulo $2^{16}$ ) received by the interface; else this field shall be Reserved.
TSEQ	-	16	If a Link Single-VC FC LLR ACK operation, then this field contains the number of end-to-end packets transmitted (modulo $2^{16}$ ) by the interface; else this field shall be Reserved.
R0	-	3	Reserved
R1	-	3	Reserved
R2	-	2	Reserved
R3	-	8	Reserved

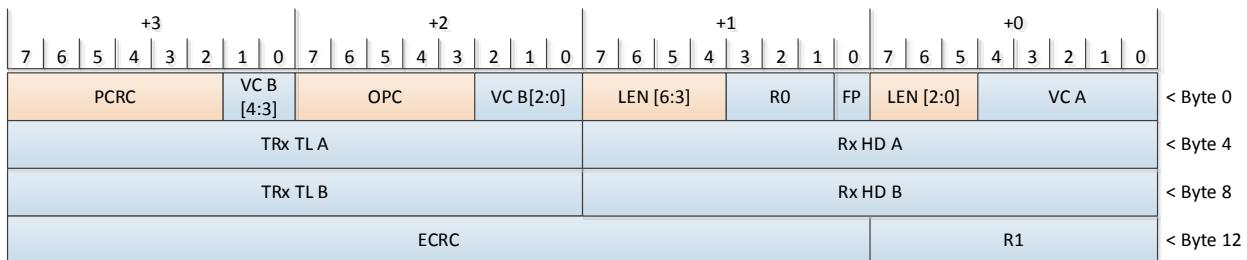


Figure 11-6: Link Dual-VC Flow-Control Packet Format

Table 11-6: Link Dual-VC Flow-Control Packet-specific Fields

Field Name	Field Abbreviation	Size (bits)	Description
Virtual Channel A	VC A	5	Indicates the first virtual channel targeted by this flow-control packet.
Virtual Channel B	VC B	5	Indicates the second virtual channel targeted by this flow-control packet.

Field Name	Field Abbreviation	Size (bits)	Description
Receiver Head A	Rx HD A	16	The current location of the head of the receiver's available logical buffer space associated with VC A.
Transmitter Tail A	TRx TL A	16	The transmitter's understanding of the peer receiver's tail associated with VC A.
Receiver Head B	Rx HD B	16	The current location of the head of the receiver's available logical buffer space associated with VC B.
Transmitter Tail B	TRx TL B	16	The transmitter's understanding of the peer receiver's tail associated with VC B.
FC Pool	FP	1	FP shall be equal to <i>Interface I-CAP 1 Adaptive FC Credit Enable</i> .
R0	-	1	Reserved
R1	-	8	Reserved

*Example Flow-Control Packet Exchange* illustrates an example interleaved flow-control and end-to-end packet exchange across a single link between two component interfaces.

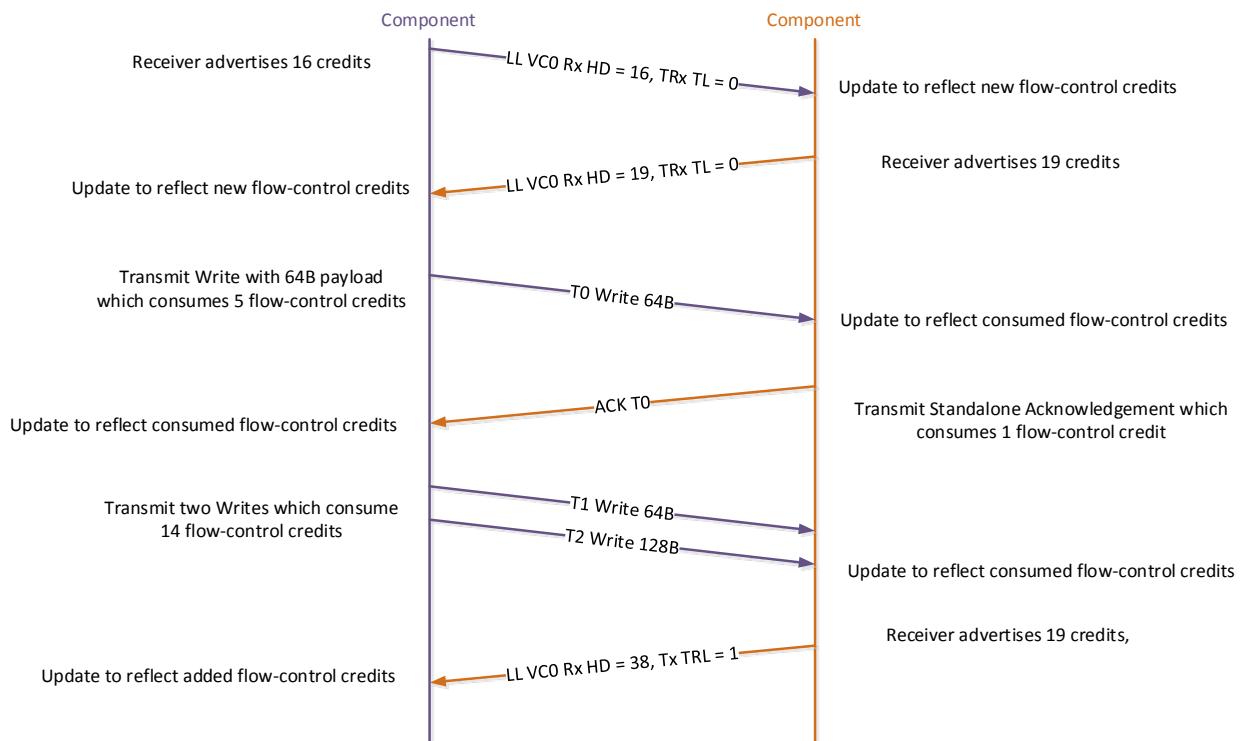


Figure 11-7: Example Flow-Control Packet Exchange

## 11.6. Link RFC

If a component supports *In-band Discovery and Initialization*, then the component shall support Link RFC (Ready for Configuration) packets. Transmission of a Link RFC packet indicates the component has reached an internal operational state where it can execute *In-band Management* packets, or the component is performing self-initialization that will be completed in the indicated time.

The following are the features and requirements associated with Link RFC packets:

- The Link RFC packet shall use the *128-bit Link-local General Packet Format*.
  - If the component is not ready for configuration, then the OS field shall reflect the time remaining until the component has completed self-initialization and is ready for configuration. Remaining time is calculated as (Time Units \* Unit of Time)
    - OS[15:0] indicates the number of Time Units.
      - If non-zero, then upon receipt of a Link RFC packet, the interface shall copy this value to the *Interface Structure* Peer Component TTC field and set *Interface I-Status* Peer Link RFC TTC = 1b.
      - If zero, then upon receipt of a Link RFC packet, the interface shall set *Interface I-Status* Peer Link RFC TTC = 0b and set *Interface I-Status* Peer Link RFC Ready = 1b.
    - OS[17:16] indicates the Unit of Time
      - 0x0—1  $\mu$ s
      - 0x1—1 ms
      - 0x2—1 s
      - 0x3—Reserved
  - Once the component has completed self-initialization and is ready for configuration, then OS[15:0] shall be set to 0x0.
- If *Interface I-Control* Disable Link RFC == 0b, then an interface shall schedule Link RFC packets according to the order specified in *Link States and Descriptions* L-Up.
- Until configured otherwise, an interface shall transmit a Link RFC packet at least once every 10 ms, or the component transitions to C-CFG or C-Up. See *Component States and Descriptions*.
- Upon scheduling a Link RFC packet, the interface shall transition to I-CFG.

## 11.7. Link Synchronization (Link SR)

A given physical layer may require periodic synchronization to ensure lane alignment. The Link SR packet can be used to provide this synchronization.

The following are the features and requirements associated with Link Synchronization packets:

- The Link SR packet may be used on any physical layer that requires periodic synchronization.
- The Link SR packet format shall use the *128-bit Link-local General Packet Format*.
  - The OpCode-specific payload shall be Reserved.
- The *Interface Structure* LSRINT field controls Link SR packet enablement and the maximum transmission period. If LSRINT ≠ 0 then the Link SR packet shall be transmitted at least once every  $2^{\text{LSRINT}}$  UI. The scheduling interval should be greater than equal to  $2^{16}$  UI.

## 11.8. Link Idle

Receivers require a deterministic method to identify the start of a packet. A Link Idle packet is used to maintain symbol lock in the absence of other link-local or end-to-end packet transmission.

The following are the features and requirements associated with Link Idle packets:

- 5
- The Link Idle packet shall be supported.
  - The Link Idle packet shall use the *Link Idle Packet Format*.

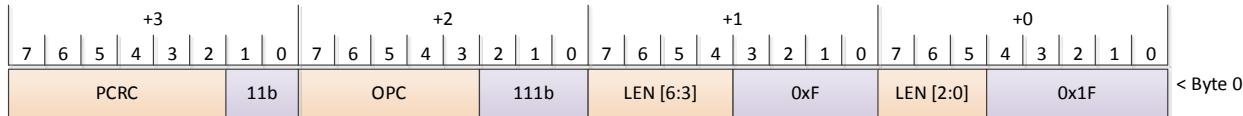


Figure 11-8: Link Idle Packet Format

## 11.9. Steps Following Physical Layer Retraining

10 The following steps shall be taken in the order listed following physical layer retraining:

1. If *Link-level Reliability (LLR)* is supported and enabled, then the interface shall perform LLR recovery. LLR recovery shall be successfully completed prior to exchanging *Explicit Flow Control* packets or transmitting any end-to-end packet.
2. If the interface supports *Explicit Flow Control*, then it shall transmit a minimum of one *Explicit Flow Control* packet per enabled VC.

15

## 11.10. Link Resynchronization

Link resynchronization is used to synchronize a transmitter and a receiver interfaces such that the start of each link-local or end-to-end packet is deterministically identified. Resynchronization is a multi-step process as illustrated in *Example Link Resynchronization Exchange*.

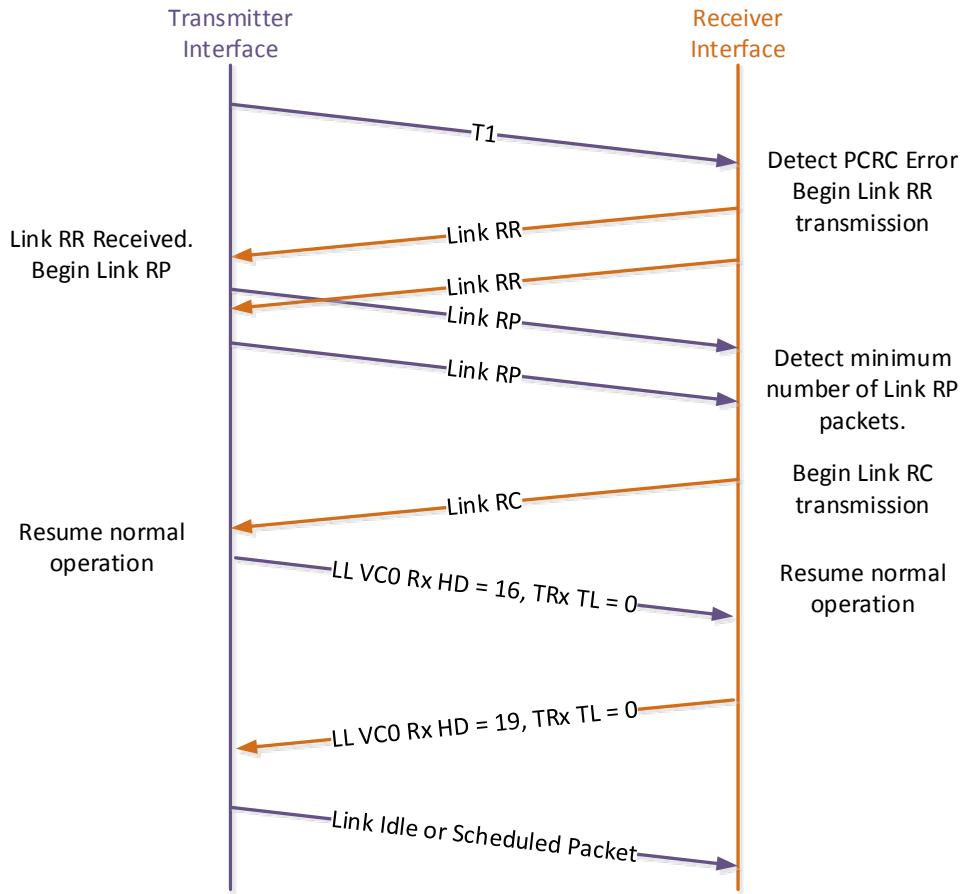


Figure 11-9: Example Link Resynchronization Exchange

The steps are as follows:

- Upon detecting a transient error where the start of a packet cannot be determined, e.g., a PCRC error, the receiver interface shall start transmitting *Link Resynchronization Request (Link RR)* packets. The receiver shall enter a mode where it is explicitly searching the receive bit-stream for a series of Link RP packets. All received bits shall be silently discarded until a Link RP packet is detected.
- Upon receiving a *Link Resynchronization Request (Link RR)*, the transmitter interface shall start transmitting *Link Resynchronization Pattern (Link RP)* packets.
- Upon receiving the minimum number of Link RP packets, the receiver interface shall start transmitting *Link Resynchronization Completion (Link RC)* packets. At this point, the transmitter and receiver interfaces are synchronized.
- If supported and enabled, then the interface shall initiate *Link-level Reliability (LLR)* recovery.
- If the interfaces support and are enabled to use *Explicit Flow Control*, then for each supported and enabled VC, the transmitter and receiver interfaces shall generate flow-control packets.
- The transmitter and receiver interfaces shall transmit any pending end-to-end packets.

It is possible for both receiver interfaces to detect a transient error where the start of a packet cannot be determined in their respective directions either simultaneously or during the resynchronization process. If this occurs, then the interfaces shall initiate physical layer retraining.

### 11.10.1. Link Resynchronization Request (Link RR)

The Link RR packet is transmitted by the receiver interface upon detection of a transient error where the start of a packet cannot be determined, e.g., a PCRC error.

The following are the features and requirements associated with Link RR packets:

- 5 • The Link RR packet shall be supported.
- The Link RR packet format shall use the *128-bit Link-local General Packet Format*.
  - o The OpCode-specific fields shall be Reserved.
- The Link RR packet shall be immediately scheduled as the next packet to transmit upon detecting a transient error.
  - 10 o At the end of each subsequent 8192 UI, a Link RR packet shall be scheduled as the next packet to transmit until the minimum number of *Link Resynchronization Pattern (Link RP)* packets required to complete resynchronization are received, or upon exceeding a maximum of 255 Link RR transmissions. This enables end-to-end packets on any VC with sufficient flow-control credits to be transmitted in the unaffected direction and ensures the Link RR packet is periodically transmitted.
  - 15 o If the minimum number of *Link Resynchronization Pattern (Link RP)* packets is not received upon exceeding the maximum number of transmissions, then the interface shall initiate physical layer retraining.

### 11.10.2. Link Resynchronization Pattern (Link RP)

20 A Link RP packet transmission sequence is initiated upon the receipt of a *Link Resynchronization Request (Link RR)*.

The following are the features and requirements associated with the Link RP packet:

- 25 • The Link RP packet shall be supported.
- The Link RP packet format shall use the *128-bit Link-local General Packet Format*.
  - o The OpCode-specific fields shall contain a fixed value of [0x1 FFFF FFFF FFFF FFFF].
- Upon receiving a *Link Resynchronization Request (Link RR)* packet, the interface shall transmit a contiguous sequence of Link RP packets (no other packets shall be transmitted while this Link RP packet sequence is being transmitted). The minimum number of Link RP packets in this sequence shall be 48.
  - 30 o At the end of each subsequent 8192 UI, the interface shall transmit one Link RP packet.
  - o Link RP packet transmission shall stop upon receipt of at least one *Link Resynchronization Completion (Link RC)* packet, or upon exceeding a maximum of 255 additional Link RP packet transmissions, i.e., 255 Link RP packets subsequent to the initial Link RP packet sequence.
  - 35 o If a *Link Resynchronization Completion (Link RC)* packet is not received upon exceeding the maximum number of Link RP packet transmissions, then the interface shall initiate physical layer retraining.

### 11.10.3. Link Resynchronization Completion (Link RC)

40 The Link RC packet is transmitted by the receiver interface upon detection of at least a minimum number of Link RP packets.

The following are the features and requirements associated with Link RC packets:

- The Link RC packet shall be supported.
- The Link RC packet format shall use the *128-bit Link-local General Packet Format*.
  - The OpCode-specific fields shall be Reserved.
- The Link RC packet shall be immediately scheduled upon the receiver detecting a minimum of 32 *Link Resynchronization Pattern (Link RP)* packets.
  - The interface shall transmit at least one Link RC packet every 8192 UI until a non-Link RP packet is received or upon exceeding a maximum of 255 Link RC packet transmissions.
  - If a non-Link RP packet is not received upon exceeding the maximum number of Link RC packet transmissions, then the interface shall initiate physical layer retraining.
- Upon transmission of a Link RC packet, the interface shall exit link resynchronization mode.
  - If supported and enabled, then the interface shall initiate *Link-level Reliability (LLR)* recovery.
  - For each enabled VC, the interface shall transmit flow-control packets, and then any pending link-local or end-to-end packets.
- Upon receipt of a Link RC packet, the interface shall exit link resynchronization mode.

### 11.11. Link CTL

Link CTL packets serve multiple purposes:

- To explicitly manage the physical layer
- To inform the directly-attached interface of link and component changes
- To direct the directly-attached component to perform component power and reset management.

A Link CTL exchange is initiated by one interface transmitting a Link CTL request packet with any non-Link ACK or non-Link NAK sub-code. The receiving interface executes the Link CTL request packet, and returns a Link ACK or Link NAK Link CTL packet.

The following are the features and requirements associated with Link CTL packets:

- All component interfaces shall support Link CTL.
- Link CTL packets shall use the *Link CTL General Packet Format*.
- Unless explicitly stated otherwise by this specification, the following shall apply to Link CTL packets:
  - A Link ACK Link CTL packet shall not be acknowledged.
  - A Link NAK Link CTL packet shall not be acknowledged.
  - A Link ACK Link CTL packet or a Link NAK Link CTL packet shall correspond to a single outstanding Link CTL request.
  - If a Link ACK Link CTL packet or a Link NAK Link CTL packet is received and there is no outstanding Link CTL request, then the packet shall be silently discarded.
  - A link shall initiate only one Link CTL exchange at any given time.
    - Unless explicitly stated otherwise by this specification, the requesting interface shall transmit the Link CTL packet at least once every 1 ms until it receives a Link ACK Link CTL or a Link NAK Link CTL packet or the link transitions to L-Down. After 32 Link CTL packet transmissions without receiving a corresponding Link ACK Link CTL or Link NAK Link CTL, the interface shall initiate physical layer retraining. Post physical layer retraining, if another 32 Link CTL packet transmissions occur

5

10

15

20

- without receiving a corresponding Link ACK Link CTL or Link NAK Link CTL, then the link shall transition to L-Down.
- Some Link CTL exchanges may require the interface to continuously transmit the Link CTL packet until acknowledged or power is removed.
  - For each received non-Link ACK Link CTL and non-Link NAK Link CTL packet, the receiving interface shall execute the Link CTL packet (if capable), and shall transmit a Link ACK Link CTL packet or a Link NAK Link CTL packet.
  - If the communicating components' states and corresponding peer interfaces' states allow, Link CTL packets may be interleaved with non-Link CTL link-local packets and end-to-end packets.
  - When a Link CTL packet is acknowledged, the interface shall set *Interface I-Status Link CTL Completed* = 1b.
  - All unused OS bits in a given Link CTL packet shall be Reserved.
  - The Peer-Attribute Link CTL is used to discover peer interface attributes to simplify and to ensure interoperability. If any peer interface attributes preclude interoperability, e.g., incompatible OpClass, incompatible *Link-Local Flow Control*, etc., then the interface shall set *Interface I-Status Peer Interface Incompatibility Detected* = 1b, and shall transition the link to the L-Down state.

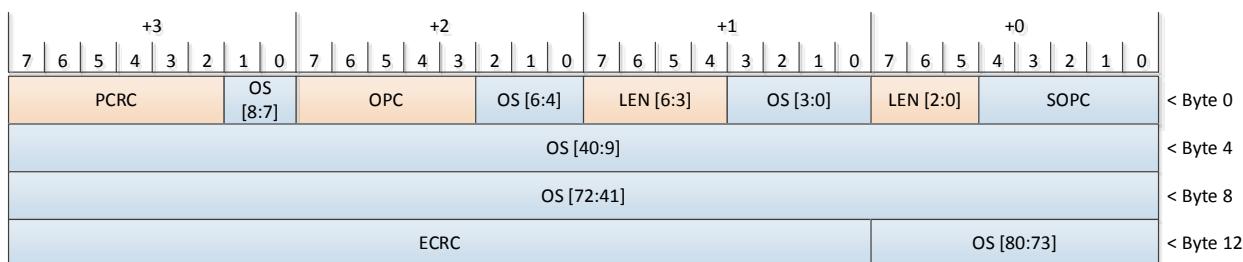


Figure 11-10: Link CTL General Packet Format

Table 11-7: Link CTL Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
SOPC	Sub-OpCode	5	See <i>Link CTL Sub-OpCodes</i>

Table 11-8: Link CTL Sub-OpCodes

Sub-OpCode Name	Sub-OpCode Encoding	Description
Link ACK	0x0	<p>Link-local positive acknowledgment packet used in response to a Link CTL packets. A Link ACK shall be returned when the peer interface supports the requested Link CTL operation and has successfully completed the Link CTL request.</p> <p>Unless explicitly stated otherwise by this specification, the OS field shall be Reserved.</p> <p>Upon receipt, the interface shall set <i>Interface I-Status Link CTL Completion Status</i> = 0x1.</p>

Sub-OpCode Name	Sub-OpCode Encoding	Description
Link NAK	0x1	<p>Link-local negative acknowledgment packet. A Link NAK shall be returned when the peer interface does not support the requested Link CTL operation, does not support an optional link-local operation or capability (e.g., <i>Link-level Reliability (LLR)</i>), or failed to complete the Link CTL request (e.g., to transition link state).</p> <p>OS[4:0] shall be set to one of the following:</p> <ul style="list-style-type: none"> <li>0x0—Reserved (Never returned)</li> <li>0x1—Reserved (Never returned)</li> <li>0x2—Unsupported Link CTL Operation</li> <li>0x3—Unable to complete Link CTL request</li> <li>0x4—Unsupported Link-local Operation: LLR, Link SR</li> <li>0x5-0x1F—Reserved</li> </ul> <p>Upon receipt, the interface shall set <i>Interface I-Status Link CTL Completion Status</i> = OS[4:0].</p>
Enter-Link-LP	0x2	<p>Request the peer interface to transition to the Link-LP state.</p> <p>Prior to transitioning to the Link-LP state, the receiving interface shall transmit a Link ACK Link CTL packet.</p> <p>Upon receiving the Link ACK Link CTL packet, or if the requesting interface does not receive a Link ACK Link CTL packet within 1 ms, it shall automatically transition to the Link-LP state.</p> <p>OS[2:0] shall be set to a value of [1-4] to indicate the requested PHY-LP [1-4] state. If OS[2:0] ≠ [1-4], then the interface does not take any action, and transmits a Link ACK Link CTL.</p> <p>If the physical layer does not support any PHY-LP [1-4] states, then the receiving interface shall return a Link NAK Link CTL.</p> <p>If the physical layer does not support the requested PHY-LP [1-4] state, then the physical layer shall transition to the nearest supported higher power PHY-LP [1-4] state.</p> <p>If an interface is in L-Up or L-Up-LP, then it may schedule an Enter-Link-LP Link CTL packet.</p> <p>Each interface may independently schedule an Enter-Link-LP Link CTL packet.</p>
Enter-Link-Up-LP	0x3	<p>Request the peer interface to transition to the Link-Up-LP state.</p> <p>Prior to transitioning to or if presently in the Link-Up-LP state, the receiving interface shall transmit a Link ACK Link CTL packet.</p> <p>Upon receiving the Link ACK Link CTL packet, the requesting interface shall automatically transition to the Link-Up-LP state.</p>

Sub-OpCode Name	Sub-OpCode Encoding	Description
		<p>OS[2:0] shall be set to a value of [1-4] to indicate the requested PHY-Up-LP [1-4] state. If OS[2:0] ≠ [1-4], then the interface does not take any action, and transmits a Link ACK Link CTL.</p> <p>If the physical layer does not support any PHY-Up-LP [1-4] states, then the receiving interface shall return a Link NAK Link CTL.</p> <p>If the physical layer does not support the requested PHY-Up-LP [1-4] state, then the physical layer shall transition to the nearest supported higher power PHY-Up-LP [1-4] state. If an interface is in L-Up, then it may schedule an Enter-Link-Up-LP Link CTL packet.</p> <p>Each interface may independently schedule an Enter-Link-Up-LP Link CTL packet.</p>
Exit-Link-Up-LP	0x4	<p>Request the peer interface to transition from the Link-Up-LP to the Link-Up state.</p> <p>Prior to transitioning to or if presently in the Link-Up state, the receiving interface shall transmit a Link ACK Link CTL packet.</p> <p>Upon receiving the Link ACK Link CTL packet, the requesting interface shall automatically transition to the L-Up state, and the link to the Link-Up state.</p> <p>If an interface is in L-Up-LP, then it may schedule an Exit-Link-Up-LP Link CTL packet.</p> <p>Each interface may independently schedule an Exit-Link-Up-LP Link CTL packet.</p>
Peer-C-Reset	0x5	<p>Request the peer component to perform a <i>Full Component Reset</i>.</p> <p>Prior to initiating a <i>Full Component Reset</i>, the receiving interface shall transmit a Link ACK Link CTL packet. Once the receiving interface has transmitted a Link ACK Link CTL packet, it shall not silently discard all link-local and end-to-end packets.</p> <p>Upon receiving a Link ACK Link CTL packet or detecting the physical layer transitioning to PHY-Down, the requesting interface shall stop transmitting Peer-C-Reset Link CTL packets, and shall update the <i>Interface Structure</i> Peer State peer component's C-state bits.</p> <p>If an interface is in L-Up or L-Up-LP, then it may schedule a Peer-C-Reset Link CTL packet.</p> <p>Each interface may independently schedule a Peer-C-Reset Link CTL packet.</p>
Peer-C-Up	0x6	Request the peer component to transition from C-DLP to C-Up.

Sub-OpCode Name	Sub-OpCode Encoding	Description
		<p>Prior to transitioning to or if the component is already in C-Up, the receiving interface shall transmit a Link ACK Link CTL packet.</p> <p>If a Link ACK CTL is received, then update the <i>Interface Structure</i> Peer State peer component's C-state bits.</p>
Peer-Enter-C-DLP	0x7	<p>Inform the peer component interface that this component is entering or is already in C-DLP.</p> <p>If OS[0] == 0b, then the receiving interface shall transition to I-LP and the link to L-LP.</p> <p>If OS[0] == 1b, then the receiving interface state shall not change state and if supported, then the link shall transition to L-Up-LP else it shall remain in L-Up.</p> <p>Upon receipt, the interface shall transmit a Link ACK Link CTL packet, and update the <i>Interface Structure</i> Peer State peer component's C-state bits.</p>
Peer-Exit-C-DLP	0x8	<p>Inform the peer component interface that this component is exiting C-DLP.</p> <p>The receiving interface shall transmit a Link ACK Link CTL packet. Upon receipt, the component shall transition all component interfaces to the I-Up state and the respective links to L-Up.</p> <p>Each interface may independently schedule a Peer-Exit-C-DLP Link CTL packet.</p>
Initiate PHY Retraining	0x9	<p>Direct the peer component interface to initiate physical layer retraining.</p> <p>Prior to initiating physical layer retraining, the receiving interface shall transmit four back-to-back Link ACK Link CTL packets.</p> <p>Upon receiving a Link ACK Link CTL packet or detecting the physical layer retraining has been initiated, the requesting interface shall stop transmitting Initiate PHY Retraining Link CTL packets.</p> <p>Each interface may independently schedule an Initiate PHY Retraining Link CTL packet.</p>
End Data Stream	0xA	<p>Inform the peer component interface that there are presently no packets other than Link Idle packets scheduled for transmission.</p> <p>Upon receiving a Link ACK Link CTL packet or a <i>Link Idle</i> packet, the requesting interface shall stop transmitting End Data Stream Link CTL packets.</p> <p>The OS field shall be Reserved.</p>

Sub-OpCode Name	Sub-OpCode Encoding	Description
<b>Link Reset Event</b>	0xB	<p>Inform the peer component interface that a link reset event has been initiated and the peer is to execute a link reset and transition to L-Down and PHY-Down.</p> <p>Upon receiving a Link ACK Link CTL packet or detecting the physical layer has transitioned to the PHY-Down state, the requesting interface shall stop transmitting Link Reset Event packets and shall complete its transition to L-Down and PHY-Down. Once the receiving interface has transmitted a Link ACK Link CTL packet, it shall silently discard all link-local and end-to-end packets. Once the PHY has transitioned to PHY-Down, both interfaces shall silently discard any new packet scheduled on the respective interface.</p> <p>The OS field shall be Reserved.</p>
<b>Peer-Attribute</b>	0xC	<p>Request the peer component to return a set of component attributes. Upon receipt, the receiving interface shall transmit a Link ACK Link CTL packet with OS field set as follows:</p> <p>OS[0] = 1b if the peer component supports the P2P-Core OpClass on this interface; if not, then OS[0] = 0b. Upon Peer-Attribute receipt, set <i>Interface Structure Peer State Peer Interface P2P-Core OpClass Support</i> = OS[0].</p> <p>OS[1] = 1b if the peer component supports the P2P-Coherency OpClass on this interface; if not, then OS[1] = 0b. Upon Peer-Attribute receipt, set <i>Interface Structure Peer State Peer Interface P2P-Coherency OpClass Support</i> = OS[1].</p> <p>OS[2] = 1b if the peer component supports the P2P-Vendor-defined OpClass on this interface; if not, then OS[2] = 0b. Upon Peer-Attribute receipt, set <i>Interface Structure Peer State Peer Interface P2P-Vendor-defined OpClass Support</i> = OS[2].</p> <p>OS[3] = 1b if the peer component supports the explicit OpClass on this interface; if not, then OS[3] = 0b. Upon Peer-Attribute receipt, set <i>Interface Structure Peer State Peer Interface Explicit OpClass Support</i> = OS[3].</p> <p>OS[4] = <i>Core Structure Component CAP 1 Control Manager Type</i>. Upon Peer-Attribute receipt, set <i>Interface Structure Peer State Peer Component Manager Type</i> = OS[4].</p> <p>OS[5] = 1b if CID 0 of the peer component is configured; if not, then OS[5] = 0b. If configured, then OS[38:27] shall contain the peer's CID 0 value, else these bits shall be Reserved. Upon Peer-Attribute receipt, set <i>Interface Structure Peer State Valid Peer CID</i> = OS[5].</p> <p>OS[6] = 1b if any of CID [1-3] fields of the peer component is configured; if not, then OS[6] = 0b. Upon Peer-Attribute receipt,</p>

Sub-OpCode Name	Sub-OpCode Encoding	Description
		<p>set <i>Interface Structure Peer Component Multiple CID Configuration</i> = OS[6].</p> <p>OS[7] = 1b if SID 0 of the peer component is configured; if not, then OS[7] = 0b. If configured, then OS[54:39] shall contain the peer's SID 0 value, else these bits shall be Reserved. Upon Peer-Attribute receipt, set <i>Interface Structure Peer State Valid Peer SID</i> = OS[7].</p> <p>OS[8] = 1b if any SID [1-3] fields of the peer component is configured; if not, then OS[8] = 0b. Upon Peer-Attribute receipt, set <i>Interface Structure Peer Component Multiple SID Configuration</i> = OS[8].</p> <p>OS[9] = 1b if the peer component supports a Home Agent; if not, then OS[9] = 0b. Upon Peer-Attribute receipt, set <i>Interface Structure Peer Component Home Agent Support</i> = OS[9].</p> <p>OS[10] = 1b if the peer component supports a Caching Agent; if not, then OS[10] = 0b. Upon Peer-Attribute receipt, set <i>Interface Structure Peer Component Caching Agent Support</i> = OS[10].</p> <p>OS[26:11] = Peer <i>Core Structure</i> Base C-Class. Upon Peer-Attribute receipt, shall set <i>Interface Structure Peer Base C-Class</i> = OS[26:11] and shall set <i>Interface Structure Peer State Valid Peer Base C-Class</i> = 1b.</p> <p>OS[38:27] = Peer <i>Core Structure</i> CID 0. Upon Peer-Attribute receipt, if <i>Interface I-CAP 1 Control Peer CID Configured</i> == 0b and OS[5] == 1b, then shall set Peer CID = OS[38:27] and shall set <i>Interface Structure Peer State Valid Peer CID</i> = 1b.</p> <p>OS[54:39] = Peer <i>Core Structure</i> SID 0. Upon Peer-Attribute receipt, if <i>Interface I-CAP 1 Control Peer SID Configured</i> == 0b and OS[7] == 1b, then shall set Peer SID = OS[54:39] and shall set <i>Interface Structure Peer State Valid Peer SID</i> = 1b.</p> <p>OS[66:55] = Peer <i>Interface Structure</i> Interface ID. Upon Peer-Attribute receipt, shall set <i>Interface Structure Peer Interface ID</i> = OS[66:55] and shall set <i>Interface Structure Valid Peer Interface</i> = 1b.</p> <p>This field shall be copied into the <i>Interface Structure Peer Interface ID</i> field, and the corresponding Peer State bit shall be set to 1b.</p> <p>OS[69:67] = Encoded peer component's C-State as follows:</p> <ul style="list-style-type: none"> <li>0x1—C-Down</li> <li>0x2—C-CFG</li> <li>0x3—C-Up</li> <li>0x4—C-LP</li> <li>0x5—C-DLP</li> </ul>

Sub-OpCode Name	Sub-OpCode Encoding	Description
		<p>0x0, 0x6-0x7—Reserved</p> <p>OS[74:70] = <i>Interface Structure</i> Peer HVS. If Peer HVS <math>\leq</math> <i>Interface Structure</i> HVS, then the field shall be copied to <i>Interface</i> HVE. If Peer HVS <math>&gt;</math> <i>Interface Structure</i> HVS, then <i>Interface Structure</i> HVS shall be copied to <i>Interface Structure</i> HVE.</p> <p>OS[76:75] = Encoded peer interface's supported flow-control as follows:</p> <ul style="list-style-type: none"> <li>0x0—<i>Implicit Flow Control</i></li> <li>0x1—<i>Explicit Flow Control</i></li> <li>0x2—<i>Implicit Flow Control</i> and <i>Explicit Flow Control</i></li> </ul> <p>Upon Peer-Attribute receipt, set <i>Interface Structure</i> Peer Interface Flow-control Support = OS[76:75].</p> <p>Each interface may independently schedule a Peer-Attribute Link CTL packet.</p> <p>Prior to software initiating a Peer-Attribute request, software shall disable <i>Interface I-CAP 1 Control</i> Source CID Packet Validation Enable and Source SID Packet Validation Enable fields.</p>
PM_Enter_L1	0xD	<p>PM_Enter_L1 is a semantic analog of PCI Express PM_Enter_L1. If an interface supports and is enabled to use the PCI Express physical layer and LTSSM, then the PM_Enter_L1 Link CTL is used by the interface to inform its peer that it is transitioning to the PCI Express L1 link state.</p> <p>Prior to transmitting PM_Enter_L1 Link CTL packets, the interface shall ensure it has accumulated sufficient flow-control credits on each enabled VC to transmit at least the largest supported end-to-end packet. Once it has sufficient flow-control credits, the interface shall stop transmitting end-to-end packets.</p> <p>Once the interface begins transmitting PM_Enter_L1 Link CTL packets, it shall not transmit end-to-end packets. If end-to-end packets arrive that require transmission, then the interface shall complete the transition to L1 and then transition back to L0 to enable such transmission.</p> <p>The interface shall continuously transmit PM_Enter_L1 Link CTL packets with no more than 8 (8b/10b encoding) or 32 (128b/130b encoding) Symbol times of idle between PM_Enter_L1 Link CTL packets until the packet is acknowledged or the physical layer transitions to electrical idle. An interface may transmit non-Link CTL packets and SKP Ordered Sets between PM_Enter_L1 Link CTL packets; these shall not contribute to the idle time limit.</p>

Sub-OpCode Name	Sub-OpCode Encoding	Description
PM_Enter_L23		<p>Upon receipt of a PM_Enter_L1 Link CTL packet, the peer interface shall transmit a Link ACK Link CTL packet and then transition the physical layer to electrical idle.</p> <p>PM_Enter_L1 shall be supported only by an interface that supports and is enabled to use the PCI Express physical layer and LTSSM. If PM_Enter_L1 Link CTL packet is received and the interface is not enabled to use PCI Express physical layer, then a Link NAK with Unsupported Link CTL Operation shall be returned.</p>
PM_Enter_L23	0xE	<p>PM_Enter_L23 is a semantic analog of PCI Express PM_Enter_L23. If an interface supports and is enabled to use the PCI Express physical layer and LTSSM, then the PM_Enter_L23 Link CTL is used by the interface to inform its peer that the interface is ready for link power and clock to be removed.</p> <p>The interface shall continuously transmit PM_Enter_L23 Link CTL packets with no more than 8 (8b/10b encoding) or 32 (128b/130b encoding) Symbol times of idle between PM_Enter_L23 Link CTL packets until the packet is acknowledged or power is removed.</p> <p>Once the interface begins transmitting PM_Enter_L23 Link CTL packets, it shall not transmit end-to-end packets. Upon successful completion, the link enters the L2 / L3 Ready power management state.</p> <p>Upon receipt of a PM_Enter_L23 Link CTL packet, the peer interface shall transmit a Link ACK Link CTL packet.</p> <p>PM_Enter_L23 shall be supported only by interfaces that support the PCI Express physical layer and LTSSM. If PM_Enter_L1 Link CTL packet is received and the interface is not enabled to use the PCI Express physical layer and the LTSSM, then a Link NAK with Unsupported Link CTL Operation shall be returned.</p>
PM_Active_State_Request_L1	0xF	<p>PM_Active_State_Request_L1 is a semantic analog of PCI Express PM_Active_State_Request_L1. If an interface supports and is enabled to use the PCI Express physical layer and LTSSM, then the PM_Active_State_Request_L1 is used by the interface to initiate ASPM negotiation.</p> <p>The interface shall continuously transmit PM_Active_State_Request_L1 Link CTL packets with no more than 8 (8b/10b encoding) or 32 (128b/130b encoding) Symbol times of idle between PM_Active_State_Request_L1 Link CTL packets until the packet is acknowledged. An interface may transmit non-Link CTL packets and SKP Ordered Sets between</p>

Sub-OpCode Name	Sub-OpCode Encoding	Description
		<p>PM_Active_State_Request_L1 Link CTL packets; these shall not contribute to the idle time limit.</p> <p>If the peer interface rejects the PM_Active_State_Request_L1, then it shall transmit a Link NAK “Unable to complete Link CTL request”, which is analogous to a PCI Express PM_Active_State_Nak message. Once rejected, an interface shall wait a minimum of 10 <math>\mu</math>s before transmitting a new PM_Active_State_Request_L1 Link CTL packet.</p> <p>PM_Active_State_Request_L1 shall be supported only by interfaces that support the PCI Express physical layer and LTSSM. If PM_Enter_L1 Link CTL packet is received and the interface is not enabled to use the PCI Express physical layer and the LTSSM, then a Link NAK with Unsupported Link CTL Operation shall be returned.</p>
Path Time	0x10	<p>Request the peer to transmit a Link ACK with minimal processing and transmission latency. The requesting interface uses a Path Time Link CTL packet exchange to calculate and set the <i>Interface Structure</i> Path Propagation Time field.</p> <p>The requesting interface should ensure that the link is operating at its maximum performance level prior to transmitting this Link CTL packet.</p>
Tx Packet Alignment	0x11	<p>The transmitter shall copy <i>Interface Structure</i> Rx Packet Alignment to Tx Packet Alignment Link CTL OS[7:0], and <i>Interface Structure</i> Rx Min Packet Start to OS[15:8]. The transmitter shall copy <i>Interface Structure</i> Rx LLRT ACK to OS[35:16].</p> <p>Upon receipt, the receiver shall set <i>Interface Structure</i> Tx Packet Alignment = OS[7:0], and <i>Interface Structure</i> Tx Min Packet Start = OS[15:8]. Upon receipt, the receiver shall set <i>Interface Structure</i> Tx LLRT ACK = OS[35:16].</p> <p>Transmission of all non-<i>Link Idle</i> packets scheduled subsequent to the Link ACK Link CTL packet shall use the Tx Packet Alignment and Tx Min Packet Start values.</p> <p>Upon transitioning from L-Down to L-Up, each interface shall perform a Tx Packet Alignment Link CTL-Link ACK Link CTL packet exchange prior to exchanging any non-<i>Link Idle</i> packets.</p>
Nonce Notification	0x12	<p>The Nonce Notification Link CTL packet is used to communicate the component’s <i>Core Structure</i> Component Nonce value (copied to OS[63:0]) to the peer interface.</p> <p>If the receiver’s <i>Interface Structure</i> Peer Nonce does not match the received Component Nonce, then the receiver shall take the configured <i>Interface Error Fields</i> Interface AE actions.</p>

Sub-OpCode Name	Sub-OpCode Encoding	Description
		Upon receipt of a Nonce Notification Link CTL packet, the peer interface shall transmit a Link ACK Link CTL packet. See <i>Mitigating In Situ Insertion</i> for additional details.
Peer-Set-Attribute	0x13	OS[2:0] = <i>Interface I-CAP 1 Control</i> P2P-Enabled
Reserved	0x14-0x1F	Reserved

## 11.12. Link-level Reliability (LLR)

Any component interface may support LLR packets to provide Reliable Delivery of an end-to-end packet between two directly-attached interfaces. Link-level reliability does not modify the end-to-end packet format. Instead, each interface maintains an implicit sequence number associated with each transmitted end-to-end packet, and retains a copy of the end-to-end packet until the associated sequence number is acknowledged. *Example Error-free LLR Packet Exchange* illustrates an example of error-free link end-to-end packet reliability operation:

- Upon the link transitioning to L-Up, the two components perform a three-way packet exchange to reliably communicate the number of end-to-end packets received and transmitted by the interface. In many cases, both interfaces initiate a three-way packet exchange at the same time (not illustrated in *Example Error-free LLR Packet Exchange*), if so, then each interface shall transmit and receive at least one Init ACK packet before initialization is completed.
- Once the two interfaces have initialized their implicit sequence number counters, the components begin transmitting end-to-end packets. The purple component interface (left) transmits an end-to-end packet with an implicit sequence number of 1. Upon receipt, the orange component interface (right) increments its local receive counter, and generates a LLR ACK packet with the updated value to acknowledge the receipt of end-to-end packet 1. Upon receiving the LLR ACK packet, the purple component interface releases its copy of end-to-end packet 1.
- Similarly, the orange component interface generates an end-to-end packet with an implicit sequence number of 1, which in turn, is acknowledged by a LLR ACK packet.
- As the exchange progresses, multiple end-to-end packets are exchanged and acknowledged. In some cases, a LLR packet (of any type) acknowledges multiple end-to-end packets.

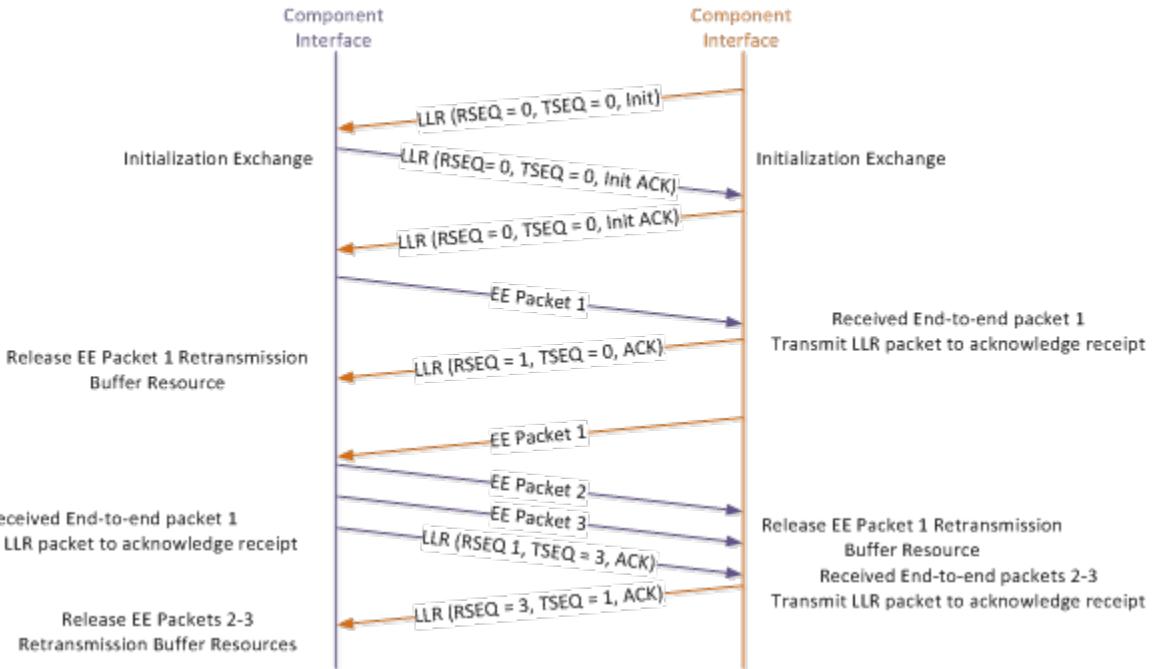


Figure 11-11: Example Error-free LLR Packet Exchange

*Example Error Handling LLR Packet Exchange* illustrates how the protocol handles an error:

- When the orange component interface (right) detects a PCRC or ECRC error, it discards the packet and enters an end-to-end packet discard mode. While in this discard mode, each subsequently received end-to-end packet in the affected direction is silently discarded.
- Upon receiving a LLR Discard packet, the purple component interface (right) begins transmitting LLR Clear Discard packets.
- For each received LLR Clear Discard packet, the orange component interface transmits a LLR Exit Discard packet.
- Upon receiving a LLR Exit Discard packet, the purple component interface knows all outstanding end-to-end packets have been successfully discarded from the link, and both interfaces have synchronized their received and transmitted counters. At this point, both interfaces resume normal link-local and end-to-end packet exchange.

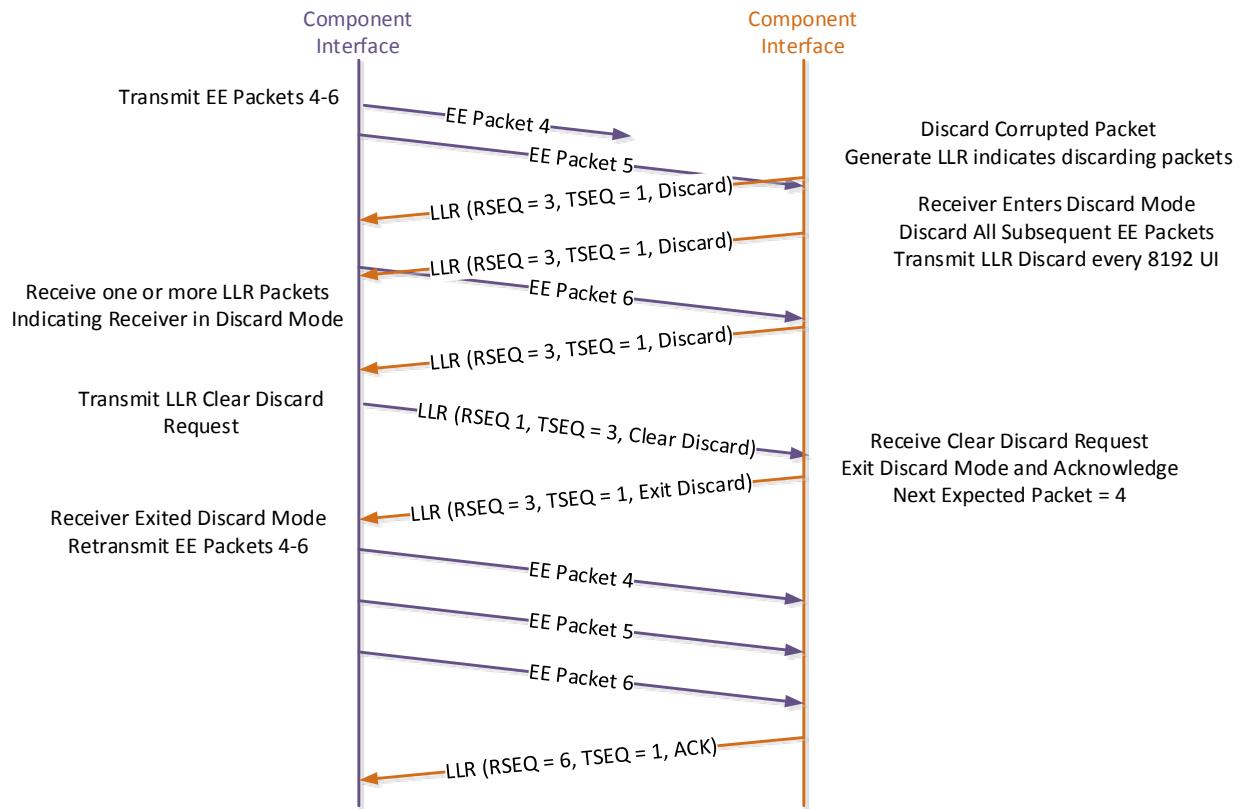


Figure 11-12: Example Error Handling LLR Packet Exchange

The following are the features and requirements associated with LLR packets:

- Interfaces that support the P2P-Core OpClass or the P2P-Coherency OpClass shall support LLR. If an interface is configured to use the P2P-Core, the P2P-Coherency OpClass, or the P2P-Vendor-defined OpClass, then LLR shall be enabled.
- LLR packets shall use the *LLR Packet Format*.
- LLR packets shall be transmitted only while the link is in L-Up or L-Up-LP.
- Upon transitioning from L-Down to L-Up, if supported and enabled, LLR initialization shall be performed prior to transmitting the first end-to-end packet.
- Once LLR is successfully initialized, each egress interface shall transmit end-to-end packets in the order they were received. Similarly, when LLR recovery is initiated, an egress interface shall retransmit end-to-end packets in the original order that they were previously transmitted with one exception:
  - If retransmitting an explicit OpClass packet, then prior to retransmission, the interface shall verify that if the packet's Congestion Deadline sub-field will be less than or equal to zero by the time the packet would be received by the peer interface, then the interface shall silently discard the explicit OpClass packet. If an explicit OpClass packet is discarded, then retransmission continues with the next end-to-end packet.
- If an interface does not support LLR, then upon receipt of any LLR packet, it shall transmit a *Link CTL* Link NAK (Unsupported Link-Local Operation: LLR) packet. Upon receipt of a Link CTL Link NAK (Unsupported Link-Local Operation: LLR) packet, an interface shall suspend LLR operation.
- LLR initialization shall be as follows:
  - Upon transitioning to L-Down, all link end-to-end packet reliability tracking logic shall be set to the uninitialized state.

- Upon transitioning from L-Down to L-Up and prior to transmitting any end-to-end packets, each interface shall transmit one LLR Init packet.
  - The interface shall transmit at least one LLR Init packet every subsequent 8192 UI. If an interface transmits 256 LLR Init packets without receiving at least one LLR Clear Discard or a *Link CTL* Link NAK (Unsupported Link-Local Operation: LLR) packet, then the interface shall initiate physical layer retraining.
    - If initialization fails a second time, then the link shall transition to L-Down.
    - If the link transitioned from L-Down to L-Up, then the implicit end-to-end packet received and transmitted sequence numbers shall be set to zero.
- If LLR is supported and enabled, then upon receipt of a LLR Init packet, an interface shall transmit a LLR Init ACK packet.
- Once the interface has transmitted and received at least one LLR Init ACK packet, the two interfaces are synchronized and may exchange end-to-end packets.
  - If an interface receives an end-to-end packet prior to completing LLR initialization, then the interface shall silently discard the end-to-end packet.
- Each interface shall maintain a single Tx LLRT (LLR timer) that continuously runs while in L-Up and LLR is enabled.
  - Tx LLRT = *Interface Structure* Tx LLRT ACK UI (+0%, -25%)
  - Tx LLRT shall be suspended when the link state transitions to L-LP or L-Down. When the link transitions from L-LP or L-Down to L-Up, the Tx LLRT shall be reset to zero, and shall resume running.
  - When the Tx LLRT expires, an interface shall transmit at least one LLR ACK packet.
    - An interface should transmit a LLR ACK packet whenever it has packets to acknowledge and no other higher precedence packets are awaiting transmission.
- Each interface shall maintain a single Rx LLRT (LLR timer) that continuously runs while in L-Up and LLR is enabled.
  - Rx LLRT = *Interface Structure* Rx LLRT ACK UI (+25%, -0%)
  - Rx LLRT shall be suspended when the link state transitions to L-LP or L-Down. When the link transitions from L-LP or L-Down to L-Up, the Rx LLRT shall be reset to zero, and shall resume running.
  - When the Rx LLRT expires, an interface should have received at least one LLR ACK packet.
    - If an interface fails to receive at least one LLR ACK packet from the link's peer interface for four consecutive Rx LLRT periods, then the interface shall initiate physical layer retraining.
- If an end-to-end packet remains unacknowledged for three consecutive Rx LLRT periods or upon completion of *Link Resynchronization* or if the interface detects an end-to-end packet with an ECRC error and *Link-level Reliability (LLR) CRC Trigger == 0b*, then the interface shall initiate the LLR retransmission process:
  - The interface shall stop transmitting new end-to-end packets.
  - The interface shall transmit a LLR Discard packet with the RSEQ and TSEQ fields indicating the implicit sequence numbers of the last successfully received and transmitted end-to-end packet. The interface shall transmit at least one LLR Discard packet every 8192 UI. If an interface transmits 256 LLR Discard packets without receiving at least one LLR Clear Discard, then the interface shall initiate physical layer retraining.
  - Upon receipt of a LLR Discard packet, the peer interface shall stop transmitting end-to-end packets.
  - The peer interface shall start transmitting LLR Clear Discard packets with the RSEQ and TSEQ fields indicating the implicit sequence numbers of the last successfully received and

transmitted packet. The interface shall transmit at least one LLR Clear Discard packet every 8192 UI until a LLR Exit Discard packet is received. If an interface transmits 256 LLR Clear Discard packets without receiving at least one LLR Exit Discard packet, then the interface shall initiate physical layer retraining.

- 5        o Upon receipt of a LLR Clear Discard packet, the interface shall transmit a LLR Exit Discard packet every 8192 UI until an end-to-end packet is received (receipt of an end-to-end packet completes LLR recovery). If an interface transmits 256 LLR Exit Discard packets without receiving at least one end-to-end packet, then the interface shall initiate physical layer retraining.
- 10        o Upon receipt of a LLR Exit Discard packet, the peer interface shall stop transmitting LLR Clear Discard packets, and shall retransmit all outstanding end-to-end packets in the order that the packets were previously transmitted, and subsequently transmit all new end-to-end packets.
- 15        o Once an interface transmits or receives a LLR Discard packet, it shall not initiate a new retransmission process until it receives a LLR Exit Discard or transmits a LLR Exit Discard.
- 20        o If both interfaces initiate the retransmission process at the same time, then each interface shall complete the retransmission process as specified.
- 25        o If *Link Resynchronization* or physical layer retraining is initiated during the LLR retransmission process, then the interfaces shall complete *Link Resynchronization* or physical layer retraining, and then resume the LLR retransmission process from the point where it was interrupted. Upon resumption, the interfaces shall reset LLR packet counters to 0x0.
- 30        • If four consecutive retransmission attempts occur with no advance in the value of the RSEQ value received from the remote interface, then the interface shall initiate physical layer retraining.
- 35        • A LLR packet indicates the number of received and transmitted end-to-end packets at the time the packet was scheduled.
  - o The RSEQ field within any LLR packet shall indicate the number of end-to-end packets successfully received by the interface (independent of the VC) at the time of the packet was scheduled. A successfully received end-to-end packet is a packet with a valid length, a valid PCRC field, a valid ECRC field, and one that did not suffer any encoding errors.
    - An end-to-end packet with a stomped ECRC field shall be treated as a valid end-to-end packet for link end-to-end reliability purposes.
    - RSEQ shall be incremented by one modulo  $2^{16}$  for each successfully received end-to-end packet.
  - o The TSEQ field within any LLR packet shall indicate the number of end-to-end packets transmitted by the interface (independent of the VC) at the time of the packet was scheduled.
    - TSEQ shall be incremented by one modulo  $2^{16}$  for each transmitted end-to-end packet.
  - o A receiver may schedule multiple LLR ACK packets at any given time.
  - o A receiver may use the Link Single-VC FC LLR ACK packet to update explicit flow-control credits for the indicated VC and to acknowledge the number end-to-end packets successfully received by the interface and the number of packets transmitted.
  - o A transmitter shall suspend end-to-end packet transmission if the number of unacknowledged end-to-end packets equals  $(2^{16} - 1)$ .
- 40        • Any LLR packet may acknowledge multiple end-to-end packets.
- 45        • Upon detecting an invalid packet:
  - o The interface shall silently discard all subsequently received end-to-end packets.

- The interface shall initiate the retransmission process as specified above.

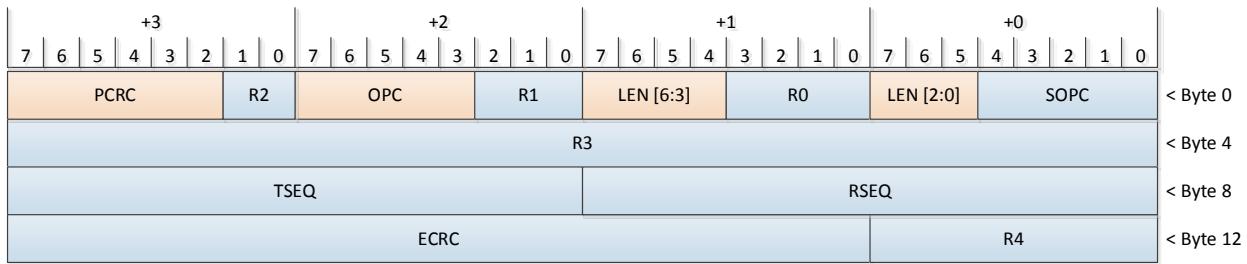


Figure 11-13: LLR Packet Format

Table 11-9: LLR Packet Fields

Field Name	Size (bits)	Description
RSEQ	16	Number of end-to-end packets (modulo $2^{16}$ ) received by the interface
TSEQ	16	Number of end-to-end packets transmitted (modulo $2^{16}$ ) by the interface
SOPC	5	<p>Sub Operation Code</p> <p>0x0—LLR ACK A LLR ACK packet acknowledges the receipt of end-to-end packets on at the directly-attached interface.</p> <p>0x1—LLR Init A LLR Init packet initiates LLR synchronization between two directly-attached interfaces.</p> <p>0x2—LLR Init ACK A LLR Init ACK packet completes LLR synchronization between two directly-attached interfaces.</p> <p>0x3—LLR Discard A LLR Discard packet informs the directly-attached interface that one or more end-to-end packets have been silently discarded.</p> <p>0x4—LLR Clear Discard A LLR Clear Discard packet informs the directly-attached interface that the interface has stopped transmitting new end-to-end packets and is awaiting a LLR Exit Discard packet.</p> <p>0x5—LLR Exit Discard A LLR Exit Discard packet informs the directly-attached interface that the interface has received the LLR Clear Discard packet and is ready to receive new end-to-end packets.</p> <p>0x6-0x1F—Reserved</p>

Field Name	Size (bits)	Description
R0	3	Reserved

### 11.13. Link-local Packet Validation and Processing

*Link-local Packet Validation* shall be as illustrated in the following sub-sections.

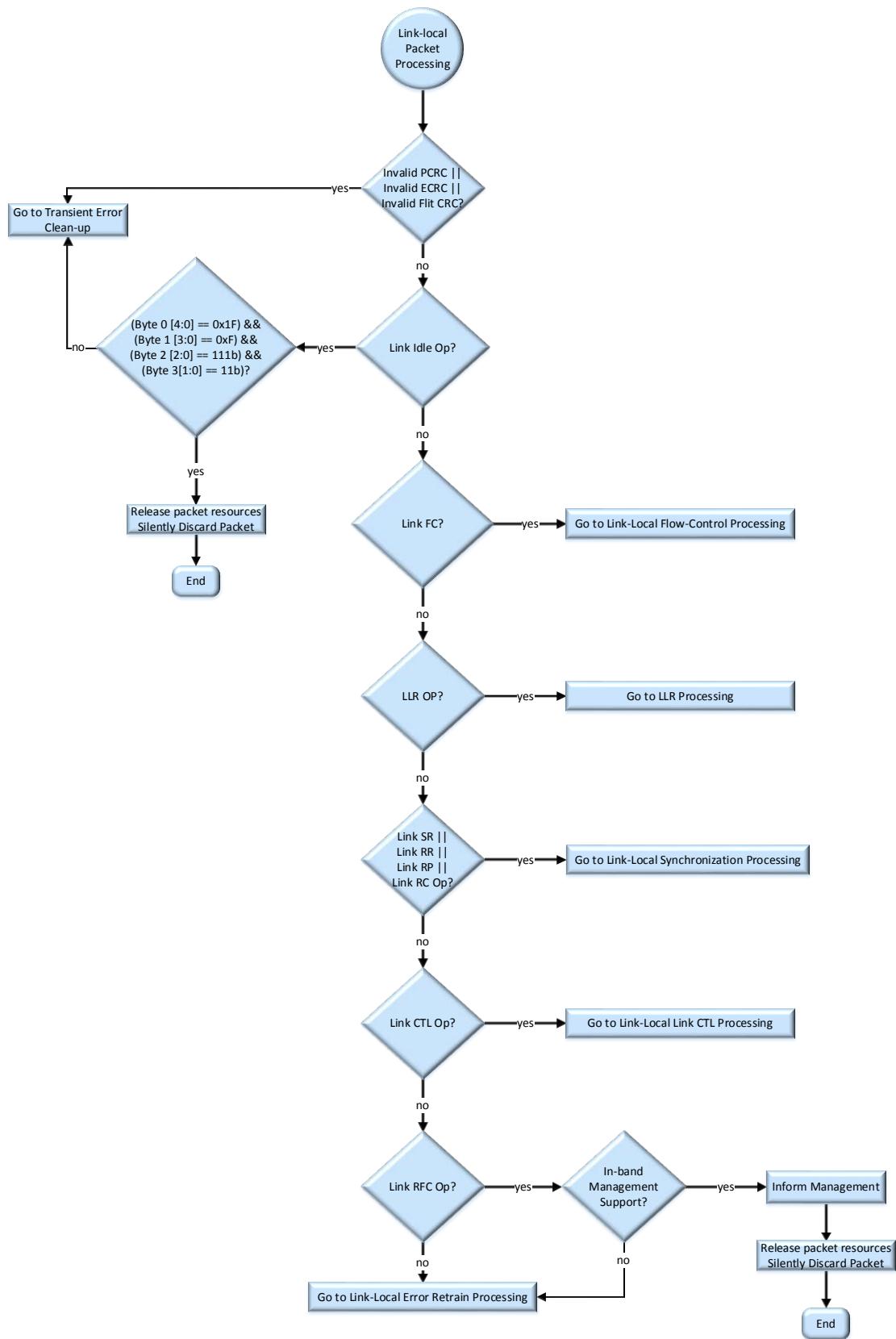


Figure 11-14: Link-local Packet Validation

Table 11-10: Link-local Packet Validation Details

Validation Step	Explanation
<b>Invalid PCRC    Invalid CRC   Invalid Flit CRC?</b>	If the packet's PCRC and ECRC fields do not match the dynamically-generated PCRC and ECRC values, or the flit's Flit CRC does not match the dynamically-generated Flit CRC, then the packet has suffered a transient error; proceed to <i>Transient Error Clean-Up</i> .
<b>Link Idle Op?</b>	Determine if the OpCode field indicates if this is a Link Idle packet.
<b>Link Idle: All OS bits == 1b?</b>	If a <i>Link Idle</i> packet, validate that all OpCode-specific bits are set to 1b. If not all 1b, then treat as if the packet suffered a PCRC error, and proceed to <i>Transient Error Clean-Up</i> . If all 1b, then silently discard the packet.
<b>Link FC?</b>	Determine if the OpCode field indicates this is a flow-control operation. If so, then proceed to Link Flow-Control Processing.
<b>LLR Op?</b>	Determine if the OpCode field indicates this is a LLR operation. If so, then proceed to LLR Processing.
<b>Link SR    Link RR    Link RP    Link RC Op?</b>	Determine if the OpCode field indicates this is a link-local synchronization operation. If so, then proceed to Link-local Synchronization Processing.
<b>Link CTL Op?</b>	Determine if the OpCode field indicates this is a link-local Link CTL operation. If so, then proceed to Link-local Link CTL Processing.
<b>Link RFC Op?</b>	Determine if the OpCode field indicates this is a link-local Link RFC operation. If not, proceed to Link-local Error Retrain Processing.
<b>Link RFC: In-Band Management Support?</b>	If <i>Core Structure Component CAP 1 Control In-band Management Support == 0b</i> , then proceed to Link-local Error Retrain Processing.

### 11.13.1. Non-AFC Link-local Flow-Control Processing

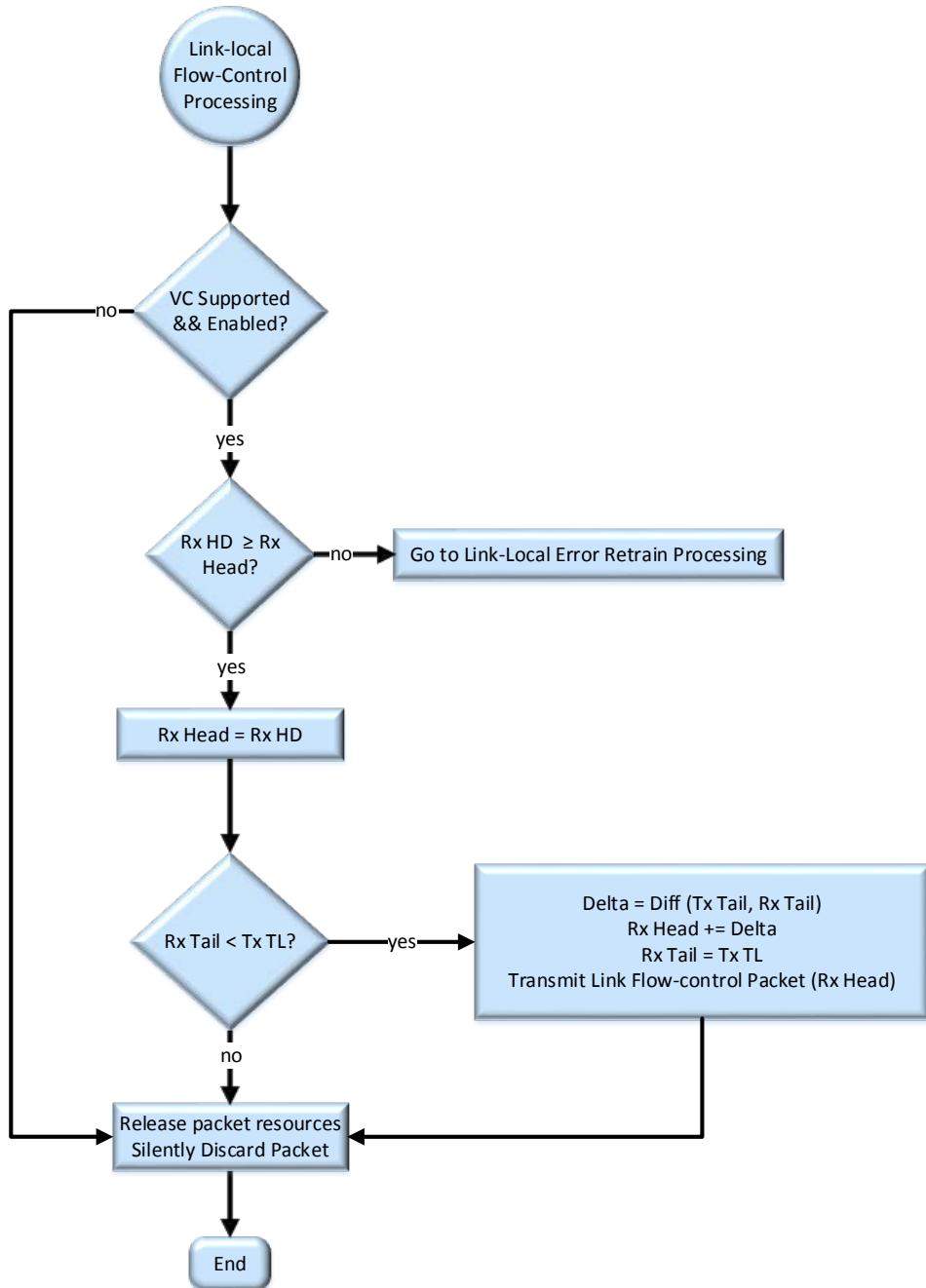


Figure 11-15: Non-AFC Link-local Flow-Control Processing

Table 11-11: Non-AFC Link-local Flow-Control Processing Details

Validation Step	Explanation
<b>VC Supported and Enabled?</b>	If unsupported or not enabled, bypass VC flow-control credit processing steps and silently discard the packet

Validation Step	Explanation
<b>Rx HD <math>\geq</math> Rx Head?</b>	Is the updated receiver head greater than or equal to the current receiver head? If yes, then set Rx Head = Rx HD. If no, then negative credits were returned which is a non-transient error condition. Go to Link-Local Error Retrain Processing.
<b>Rx Tail &lt; Tx TL?</b>	Taking Max_FC into account, determine if the transmitter's Tx Tail less than the receiver's tail. If not less than, then the flow-control credit update was successful. If less than, then synchronize the two sides and transmit a new flow-control packet with the updated Rx Head.

### 11.13.2. Link-Local Link CTL Processing

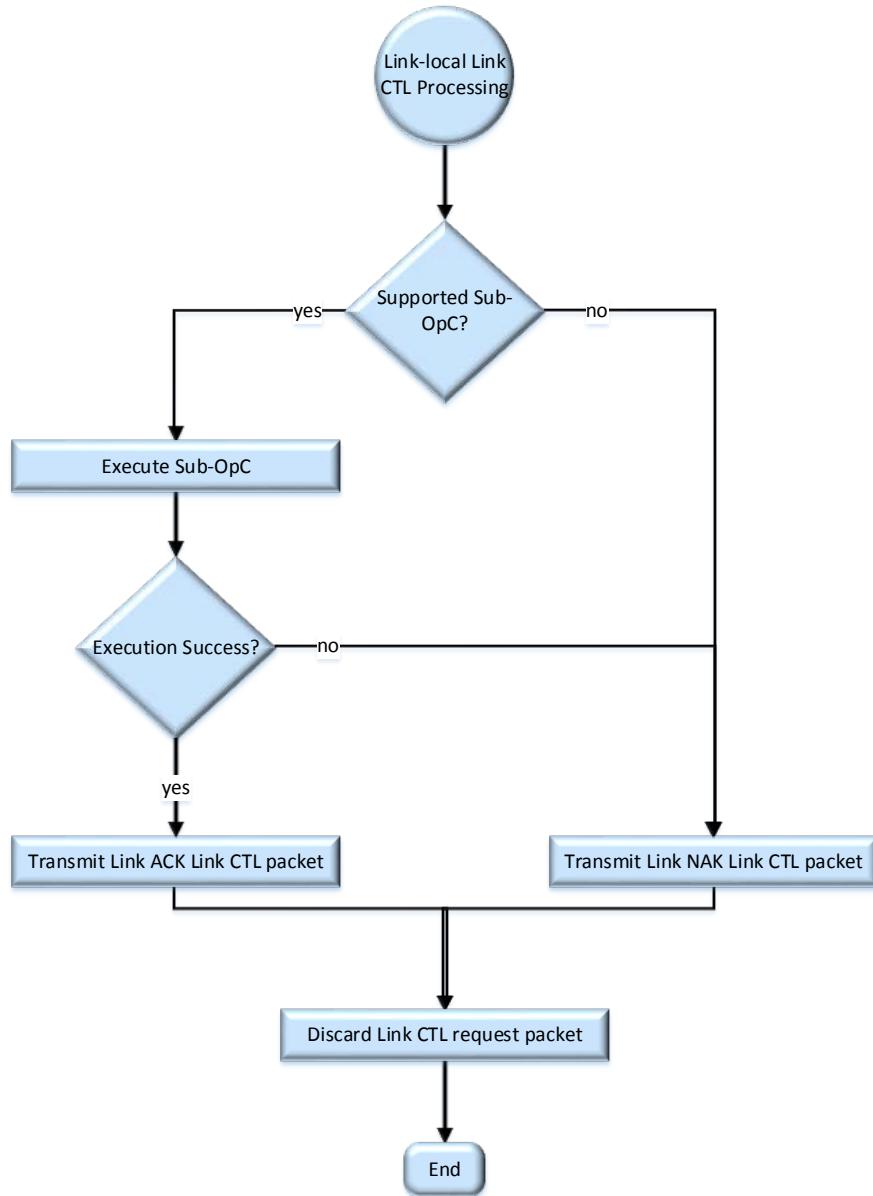


Figure 11-16: Link-local Link CTL Processing

Table 11-12: Link-local Link CTL Processing Details

Validation Step	Explanation
<b>Supported Link Sub-OpC?</b>	If an unsupported sub-operation, then transmit a Link NAK Link CTL packet, and discard the Link CTL request packet.
<b>Supported Link Sub-OpC: Execute Sub-OpC</b>	Execute the sub-operation

Validation Step	Explanation
<b>Execute Sub-OpC: If success?</b>	If execution is successful, then transmit a Link ACK <i>Link CTL</i> packet. If unsuccessful, then transmit a Link NAK <i>Link CTL</i> packet. Discard Link CTL request packet.

### 11.13.3. Link-Local Synchronization Processing

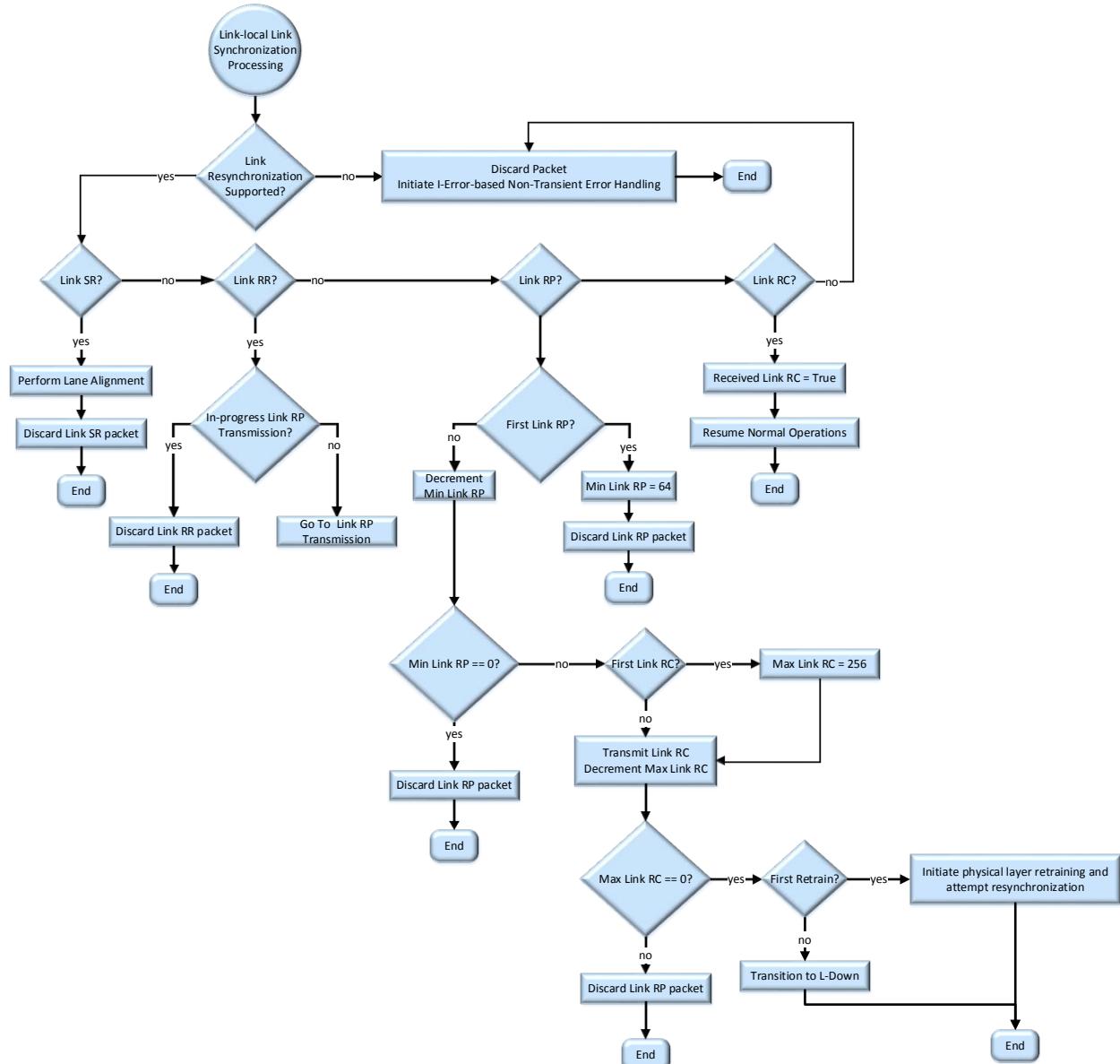


Figure 11-17: Link-local Synchronization Processing

Table 11-13: Link-local Synchronization Processing Details

Validation Step	Explanation
<b>Link Resynchronization Supported?</b>	If not, then discard the packet and initiate the non-transient error processing as configured in <i>Interface Error Fields</i> .
<b>Link SR?</b>	If a Link SR packet, then perform lane alignment and discard packet.
<b>Link RR?</b>	If not a Link RR packet, then proceed to next check to determine the packet type.
<b>Link RR: In-progress Link RP Transmission?</b>	If Link RP transmission has already started, then this is a duplicate Link RR and it can be silently discarded.
	If Link RP transmission has not started, then go to Link RP Transmission to initiate.
<b>Link RP?</b>	If not a Link RP packet, then proceed to the next check to determine the packet type.
<b>Link RP: First Link RP?</b>	If this is the first Link RP, then initialize minimum counter and discard the packet.
	If this isn't the first Link RP, then decrement the counter
<b>Link RP: Max Link RP == 0?</b>	If non-zero, then the discard the packet.
<b>Max Link RP: First Link RC?</b>	If the first Link RC packet, then set the max tracking variable for Link RC packets.
	Transmit a Link RC packet and decrement the max tracking variable.
<b>Max Link RP: Max Link RC == 0?</b>	If the max tracking variable reaches zero and the first retrain, then initiate physical layer retraining. If second retrain, then transition the link to L-Down.
<b>Link RC?</b>	If not a Link RC packet, then this is an unknown link-local packet format, and the link proceeds to Link-local Error Retrain Processing.  If a Link RC packet, then set Link RC = True to halt Link RP Transmission, and return to normal link-local and end-to-end packet exchange.

#### 11.13.4. Link-Local Link RP Transmission Processing

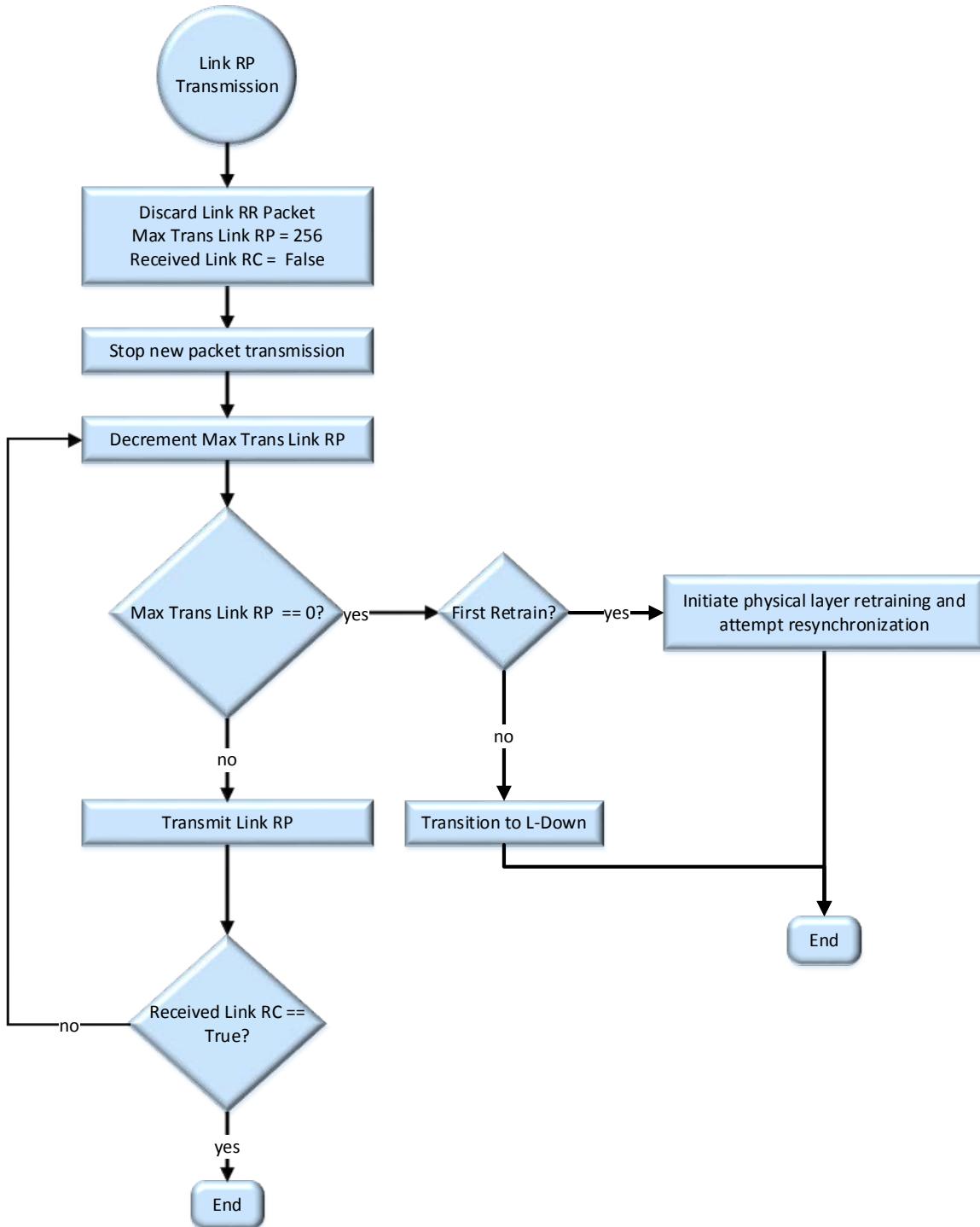


Figure 11-18: Link-local Link RP Transmission Processing

Table 11-14: Link-local Link RP Transmission Processing Details

Processing Step	Explanation
<b>Discard Link RR Packet</b>	Set up the transmission control variables to detect when a Link RC packet is received or when the maximum number of Link RP packets has been transmitted
<b>Max Trans Link RP = 256</b>	
<b>Received Link RC = False</b>	
<b>Stop new packet transmission</b>	Complete the current packet's transmission and halt all subsequent packet transmission
<b>Decrement the Max Trans Link RP</b>	Start Link RP transmission Loop
	Decrement the maximum Link RP transmission counter
<b>Max Trans Link RP == 0?</b>	If zero and the first retrain, then initiate physical layer retraining. If second retrain, then transition the link to L-Down.
	If non-zero, then transmit a Link RP packet
<b>Link RP packet: Link RC == True?</b>	If a Link RC packet has been received, then exit this loop, else jump to the start of the loop

### 11.13.5. LLR Processing

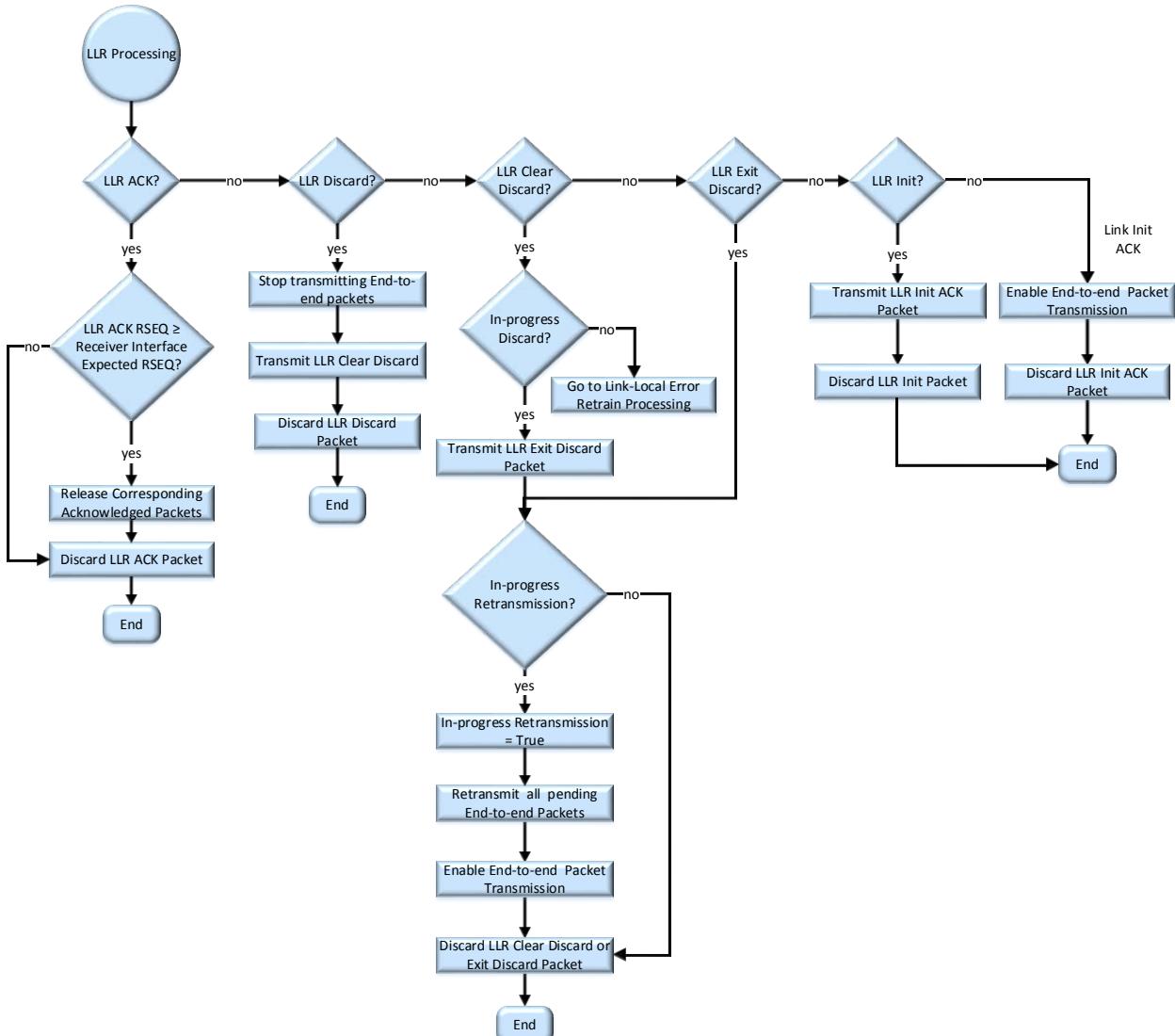


Figure 11-19: LLR Processing

Table 11-15: LLR Processing Details

Validation Step	Explanation
<b>LLR ACK?</b>	If not a LLR ACK, proceed to next packet classification step (LLDR Discard)
<b>LLR ACK: If LLR ACK RSEQ ≥ Receiver Interface Expected RSEQ?</b>	Does the LLR ACK packet's RSEQ field contain a value that is greater than or equal to the receiver's expected RSEQ? If yes, then release the corresponding acknowledged packets, and advance the receiver's expected RSEQ. Discard the LLR ACK packet.

Validation Step	Explanation
<b>LLR Discard?</b>	<p>If not a LLR Discard packet, proceed to the next packet classification step (LLR Clear Discard).</p> <p>If a LLR Discard packet, then:</p> <ul style="list-style-type: none"> <li>• Stop transmitting end-to-end packets</li> <li>• Transmit a LLR Clear Discard packet</li> <li>• Discard the LLR Discard packet</li> </ul>
<b>Link Clear Discard?</b>	<p>If not a LLR Clear Discard packet, proceed to the next packet classification step (LLR Exit Discard)</p>
<b>Link Clear Discard: In-progress Discard?</b>	<p>If not, then go to Link-Local Error Retrain Processing.</p>
<b>Link Clear Discard: Transmit LLR Exit Discard</b>	<p>Transmit a LLR Exit Discard packet</p>
<b>Link Clear Discard or Link Exit Discard: In-progress Retransmission?</b>	<p>If this is the first Link Clear Discard packet received, then initialize and retransmit all pending end-to-end packets, and then enable end-to-end packet transmission. All end-to-end packets are retransmitted / transmitted in the order received at the egress interface.</p> <p>Discard LL Clear Discard or LLR Exit Discard Packet</p>
<b>LLR Exit Discard?</b>	<p>If not a LLR Exit Discard packet, then proceed to next packet classification step (LLR Init).</p> <p>If a LLR Exit Discard packet, then follow the LLR Clear Discard steps to retransmit end-to-end packets and enable end-to-end packet transmission.</p>
<b>LLR Init?</b>	<p>If not a LLR Init packet, then proceed to the next packet classification step (LLR Init ACK).</p> <p>If a LLR Init packet, then transmit a LLR Init ACK packet and discard the LLR Init packet.</p>
<b>LLR Init ACK</b>	<p>LLR Init ACK packet received. Enable end-to-end packet transmission and discard the LLR Init ACK packet.</p>

### 11.13.6. Link-Local Error Retrain Processing

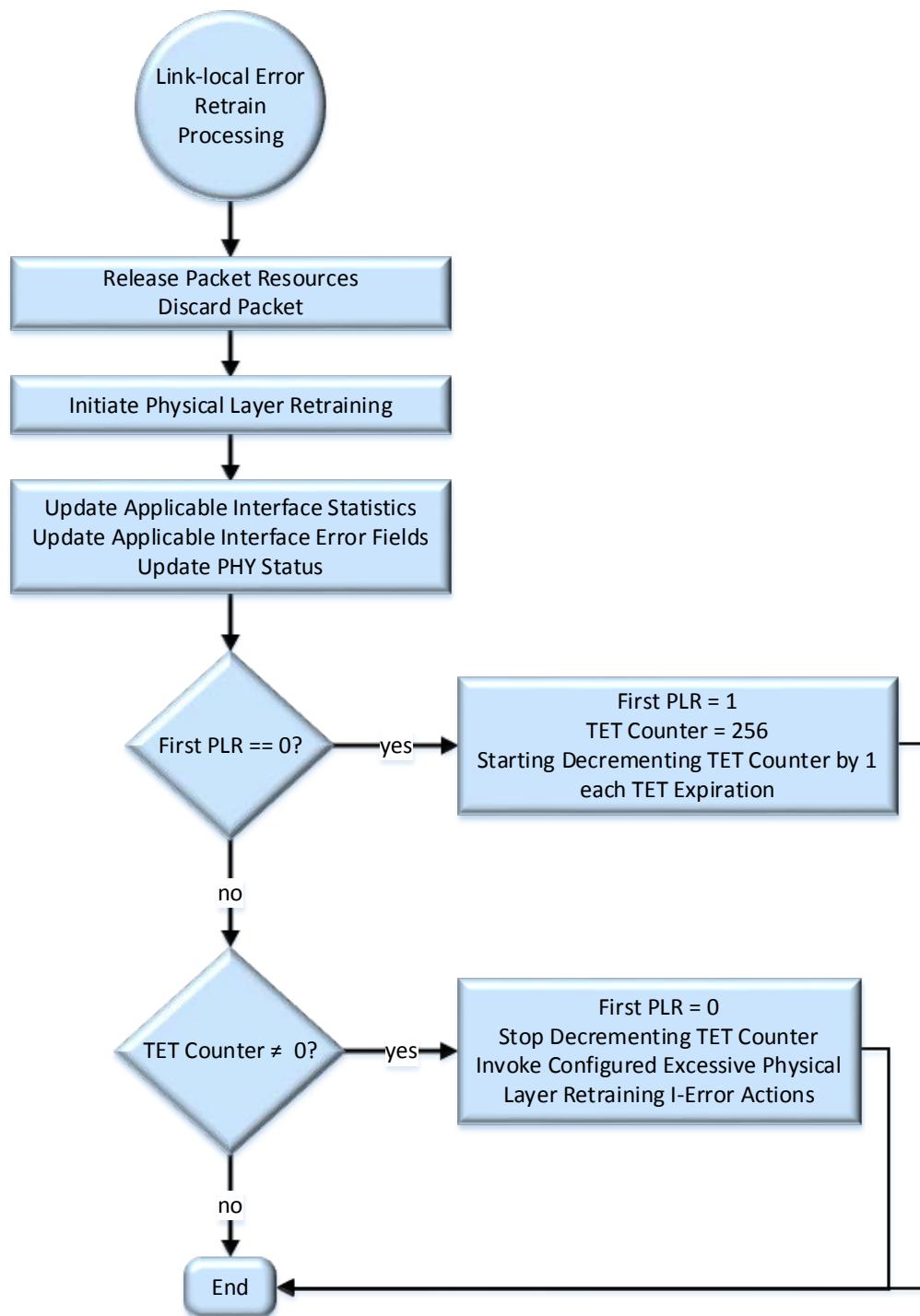


Figure 11-20: Link-Local Error Retrain Processing

Table 11-16: Link-Local Error Retrain Processing Details

Validation Step	Explanation
<b>First PLR == 0?</b>	<p>If this is the first physical layer retraining event being tracked, then:</p> <ul style="list-style-type: none"> <li>• Set First PLR = 1 and initialize TET Counter</li> <li>• For each subsequent TET expiration, decrement the TET Counter variable by 1</li> </ul>
<b>First PLR 1: TET Counter ≠ 0?</b>	<p>If this was not the first physical layer retraining event, then determine if the TET Counter had reached zero prior to a new physical layer training event occurring.</p> <p>If zero and <i>I-Error Detect Excessive Physical Layer Retraining Event == 1b</i>, then take configured error handling actions.</p>

# 12. End-to-End Packet Reliability / Validation

## 12.1. Unicast Reliable Delivery

Unicast request packet Reliable Delivery is achieved using an end-to-end request-acknowledgment paradigm. PCO communications uses a different paradigm, and shall comply with the requirements specified in *PCIe-Compatible Ordering (PCO)*.

The following are the features and requirements associated with non-PCO unicast OpClass request packet Reliable Delivery:

- Request packet exchange uses datagram communications. Each request packets that require end-to-end Reliable Delivery is acknowledged.
  - For request packets with request packet-specific response packets, the response packet is treated as an acknowledgment, e.g., a Read Response packet acknowledges a Read request packet.
  - For request packets without request packet-specific response packets, a *Standalone Acknowledgment* acknowledges the request packet, e.g., a *Standalone Acknowledgment* acknowledges a Core 64 Write request packet.
  - For vendor-defined request packets, a *Standalone Acknowledgment* or a vendor-defined response packet acknowledges a request packet.

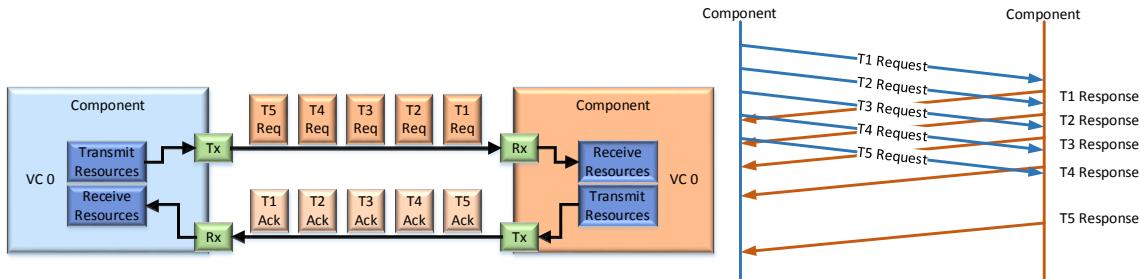


Figure 12-1: Unicast Packet Request-Acknowledgment Exchange

- A Responder shall positively acknowledge a request packet that requires Reliable Delivery upon successful validation and execution.
- If a Responder fails to successfully validate or execute a request packet, then the Responder shall transmit a negative acknowledgement (NAK).
  - A NAK is a *Standalone Acknowledgment* with a Reason code set to indicate the highest precedence error.
  - A NAK shall impact only the corresponding request packet at the Requester. Unrelated request packets and response packets may continue to be transmitted and received.
  - If the NAK indicates a non-transient error, the Requester shall fail the request packet. If the *Component Error and Signal Event Structure* is supported and enabled, then the error shall be handled as configured. If the request packet was initiated by a host processor read or write through a Requester ZMMU or equivalent, then local processor error handling should be determined by the Processor Exception Control (PEC) bit in *Requester PTE Fields*.
- A Requester shall maintain retransmission timers to detect lost request packets and response packets required for Reliable Delivery. A retransmission timer determines the minimum time before the Requester may retransmit the unacknowledged request packet.

- *Core Structure Component CAP 1* Retransmission Timer Support indicates the supported retransmission timer mechanisms.
- If a Requester supports a component-wide retransmission timer mechanism, then it shall use the *Core Structure* MUTO.
  - Components that support the P2P-Core OpClass may support MUTO or use  $TO_0$  ( $TO_0$  corresponds to the implicit interface  $VC_0$ ).
  - Components that support the P2P-Coherency OpClass may support MUTO or use  $TO_K$ .
- If a Requester supports per VC retransmission timers, then for each interface  $VC_K$ , it shall use  $TO_K$  to configure the minimum retransmission timeout value for request packets transmitted in the low-latency domains associated with  $VC_K$ , and  $LTO_K$  to configure the minimum retransmission timeout value for request packets transmitted in the non-low-latency domains associated with  $VC_K$ .
- For long-lived request packets (e.g., *Buffer Requests*), the Requester should use the methods specified in *Non-Deterministic Request Execution*.
- A Requester may implement proprietary timers and retransmission algorithms managed through means outside of this specification's scope.
- A retransmission timer associated with explicit OpClass request packets shall be greater than  $((Request\ Deadline + Responder\ Deadline + Response\ Deadline) * Deadline\ Tick)$ . See *Deadline Semantics*.
- When a request packet's retransmission timeout expires, a Requester shall retransmit the request packet unless it is explicitly stated otherwise by this specification or it is physically impossible, e.g., there is no route to a Responder. A Requester may take the following actions (these actions will vary by implementation and design choices):
  - If this is the first timeout for a given request packet, then the retry counter is set to a non-zero value. The non-zero value is Requester-specific, i.e., the Requester may set the retry counter to any value based on policies outside of this specification's scope.
  - The Requester decrements the retry counter by one.
  - If the retry counter is non-zero, then the Requester retransmits the request packet.
    - Unless explicitly stated otherwise by this specification, the retransmitted request packet shall contain the same protocol-field values as the original request packet with the exception of the PCRC and ECRC fields, which shall be dynamically generated as the request packet is transmitted.
    - A Requester may retransmit a request packet through a different interface or using a different VC.
  - If the retry counter is zero and / or no path exists between the Requester and the Responder, then the Requester shall fail the request.
    - If the *Component Error and Signal Event Structure* is supported and enabled, then the error shall be handled as configured.
    - If the request packet was initiated by a host processor read or write through a Requester ZMMU or equivalent, then local processor error handling should be determined by the Processor Exception Control (PEC) bit in *Requester PTE fields*.

**Developer Note:** *If multiple paths exist between a Requester and a Responder, then upon retransmission timer expiration, the Requester can retransmit the request on a different path (e.g., per request retransmission using round-robin selection from all enabled paths). Though this can quickly identify a viable path to the Responder, it can also increase the time to detect a failed path.*

- *Unicast Packet Timeout with Retransmission Example* illustrates elements pertinent to unicast request packet Reliable Delivery:
  - Request packets are independently acknowledged and, unless explicitly stated otherwise by this specification, acknowledgments may be returned in any order.
  - Error recovery impacts only a single request packet; all unrelated request packets shall continue to be transmitted, executed, and acknowledged.

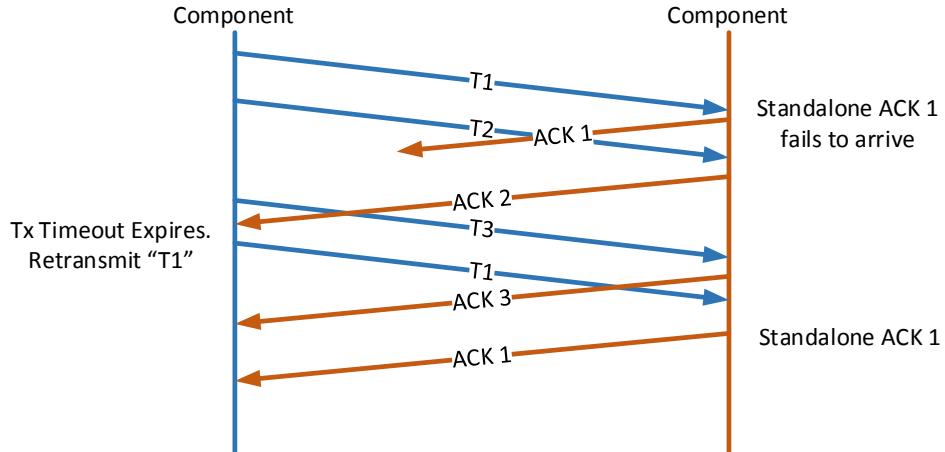


Figure 12-2: Unicast Packet Timeout with Retransmission Example

## 12.2. P2P-Core / P2P-Coherency Reliable Delivery

The following are the features and requirements specific to P2P-Core OpClass and P2P-Coherency OpClass request packet Reliable Delivery:

- Since P2P-Core and P2P-Coherency interfaces are required to support and have *Link-level Reliability (LLR)* enabled, only non-transient events can cause response packets to be lost.
- P2P-Core and P2P-Coherency request packets shall not be retransmitted.
- Retransmission timers used for P2P-Core and P2P-Coherency request packets for which a response packet is required shall be used to detect a non-transient operating condition (e.g., Responder failure or containment). If a retransmission timer (*Core Structure* MUTO or  $T_{O_k}$ ) expires and if configured, then the Requester shall inform management of a P2P Non-transient Operating Condition (see *C-Event Detect*), and shall initiate component-local error handling.
- A Responder shall validate and execute a request packet and transmit the corresponding response packet in less than or equal to the worst-case completion latency associated with any supported and enabled request packet type. Retransmission timers shall be set to a value greater than or equal to this worst-case completion latency.

## 12.3. Explicit OpClass Reliable Delivery

The following are the features and requirements specific to unicast explicit OpClass request packet Reliable Delivery:

- A Responder shall validate and execute a request packet and transmit the corresponding response packet in less than or equal to the time calculated using the *Core Structure* Responder Deadline.

If a Responder is unable to do so, then the Responder shall silently discard the request packet, and shall not transmit the corresponding response packet.

- A Responder that supports non-idempotent request packets shall comply with the Responder-specific requirements specified in *Non-idempotent Request (NIR)* to ensure non-idempotent request packet Reliable Delivery.

## 12.4. Non-Deterministic Request Execution

Some request packets are referred to as ‘non-deterministic’ because of the wide variation of their execution times. For example, *Buffer Requests* and *Pattern Requests* can take significantly longer to execute than a single read or write request packet. To ensure Reliable Delivery and to determine execution success, these request packets require two Reliable Delivery mechanisms:

- The first mechanism uses the retransmission timer and acknowledgment algorithm described in *Reliable Delivery* to ensure the request packet was successfully received by the Responder. The Responder shall perform request packet validation of all protocol fields and payload. If validation is successful, then it shall transmit a *Standalone Acknowledgment* (Reason = ND ACK). If request packet validation fails, then the Responder shall transmit a *Standalone Acknowledgment* indicating the error.
  - The Responder shall generate the *Standalone Acknowledgment* within *Core Structure* RDLAT for P2P-Core / P2P-Coherency / P2P-Vendor-defined OpClass request packets, or *Core Structure* Responder Deadline for explicit OpClass request packets.
- The second mechanism uses a variable execution timer (VET) and a request-specific response packet to determine the execution success or failure of a given request. The variable request execution timer may be set to SVETO, LVETO, ATSTO, or an implementation-specific value.
  - VET = SVETO, + 100% / -0% if the operation is considered small
  - VET = LVETO, + 100% / -0% if the operation is considered large
  - VET = ATSTO, +50% / -0% if an *ATS Translation Invalidate Request* or a *PRG Request* operation
  - If the VET expires without receiving a request-specific response packet, then the Requester shall retransmit the request packet unless it is physically impossible, e.g., there is no route to a Responder.
- To ensure a Requester and a Responder remain consistent with one another:
  - A Responder shall transmit a *Standalone Acknowledgment* (Reason = ND ACK) prior to transmitting the request-specific response packet.
    - If a Requester does not receive the *Standalone Acknowledgment*, then it shall retransmit the request packet.
    - If a Responder receives a duplicate request packet prior to receipt of a *Non-Idempotent Request Release (NIRR)*, then it shall transmit a new *Standalone Acknowledgment* (Reason = ND ACK).
  - Explicit OpClass non-deterministic request packets are non-idempotent. If explicit OpClass requests are used, then Requesters and Responders shall take the steps specified in *Non-idempotent Request (NIR)*.
    - A Requester shall not transmit a *Non-Idempotent Request Release (NIRR)* packet until it has received a *Standalone Acknowledgment* (Reason = ND ACK) and a request-specific response packet.

- A Requester shall not reuse a non-deterministic request packet's Tag value in any other request packet targeting the Responder until the non-deterministic request is completed (if an explicit OpClass, then targeting the Responder's [CID, SID]).
- A Requester shall silently discard any duplicate request-specific response packets.
- A non-deterministic request packet is completed when:
  - If a *Standalone Acknowledgment* (Reason = ND ACK) is received prior to expiration of the initial retransmission timer associated with the non-deterministic request packet, then the non-deterministic request is completed upon receipt of the *Standalone Acknowledgment* associated with the *Non-Idempotent Request Release (NIRR)* packet.
  - If the non-deterministic request packet was retransmitted due to expiration of the corresponding retransmission timer, then the non-deterministic request is completed upon both receipt of the *Standalone Acknowledgment* associated with the *Non-Idempotent Request Release (NIRR)* packet and expiration of the VET corresponding to the last retransmission of the non-deterministic request packet (the VET continues to run independent of receiving a request-specific response packet to ensure any duplicate request-specific response packets are properly handled).

*Non-deterministic Request Reliable Delivery Exchange* illustrates the additional acknowledgment packet exchange as well as when the timers are started and stopped.

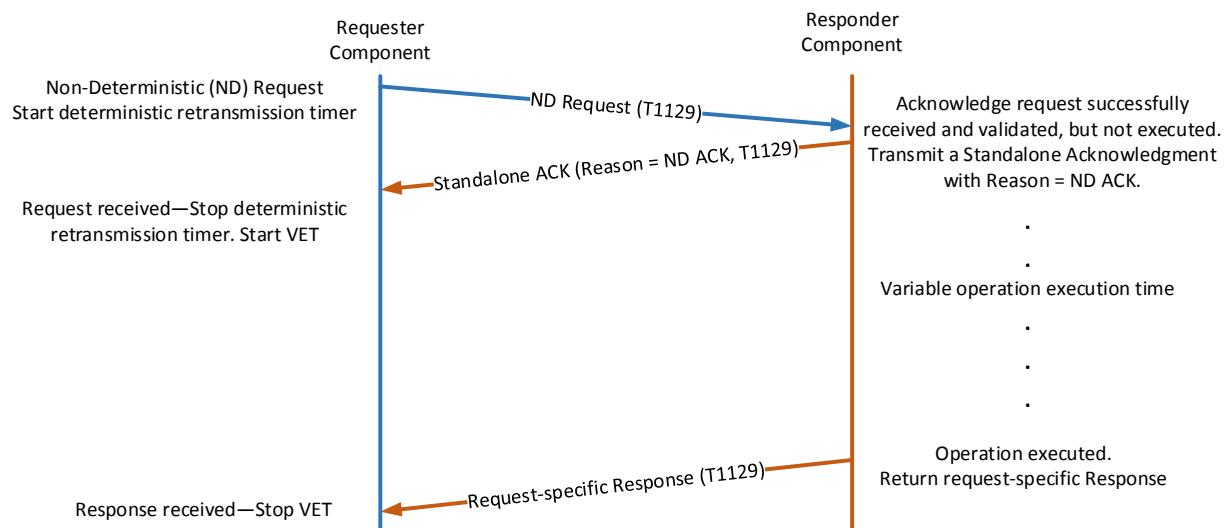


Figure 12-3: Non-deterministic Request Reliable Delivery Exchange

## 12.5. Non-idempotent Request (NIR)

Non-idempotent request packets (NIRs) are end-to-end request packets that cannot be re-executed without side-effects. When a NIR's retransmission timer expires, the Requester cannot determine on its own the root cause for the expected Response's failure to be returned. Only the Responder knows whether the NIR was successfully received and executed.

The following are the requirements to support NIR operations:

- The Responder shall maintain an implementation-specific resource that uniquely identifies each outstanding NIR operation and the corresponding execution results.

- If a new NIR request packet arrives and the Responder has insufficient implementation-specific resources, then the Responder may transmit a *Responder Not Ready (RNR) NAK* or silently discard the new NIR request packet.
- To delineate a new NIR request packet from a duplicated NIR request packet or a multi-request packet NIR operation, the Responder compares a subset of the NIR request packet fields with the corresponding values stored in the implementation-specific resources (e.g., SCID, DCID, Tag).
  - If a duplicate NIR request packet and the NIR operation has completed, then the Responder shall not execute the duplicate NIR request packet, and shall transmit a request-specific response with the execution results.
  - If a new or duplicate NIR request packet in a multi-request packet NIR operation is received and the NIR operation has not completed, then the Responder shall validate and execute the request packet, transmit a request-specific response packet, and update the execution results as appropriate.
- A Responder shall maintain the implementation-specific resource and the corresponding execution results until the Responder receives a *Non-Idempotent Request Release (NIRR)* packet, or until the resources are released due to the expiration of the corresponding failsafe timer.
  - The execution results shall be returned in a request-specific acknowledgment upon receipt of a retransmitted NIR request packet.
  - Upon receiving the request-specific acknowledgment that completes the NIR operation (success or failure), a Requester shall transmit a *Non-Idempotent Request Release (NIRR)* packet to release the corresponding NIR's implementation-specific resources.
  - Upon receiving a *Non-Idempotent Request Release (NIRR)* packet, a Responder shall transmit a *Standalone Acknowledgment*.
    - If a *Non-Idempotent Request Release (NIRR)* packet is received and there is no corresponding outstanding NIR operation, then the Responder shall transmit a *Standalone Acknowledgment* indicating successful release.
  - For NIR operations that have deterministic execution completion times, a Responder should maintain a single failsafe NIR Timer (*Core Structure NIRT*).
    - A Responder may continuously run the NIRT, or start and stop it based on whether there are implementation-specific resources awaiting release.
    - If a *Non-Idempotent Request Release (NIRR)* packet corresponding to a given implementation-specific resource is not received for at least two NIRT expirations, then the Responder shall release the implementation-specific resource.
  - For NIR operations that have non-deterministic execution completion times (e.g., Atomic Fence, Buffer, Pattern, etc.), the Responder should maintain a single extended failsafe NIR Timer (*Core Structure ENIRT*).
    - A Responder may continuously run the ENIRT, or start and stop it based on whether there are implementation-specific resources awaiting release.
    - If a *Non-Idempotent Request Release (NIRR)* packet corresponding to a given implementation-specific resource is not received for at least two ENIRT expirations (depending upon the timeout value, long-duration requests could take more than two ENIRT expirations to complete), then the Responder shall release the implementation-specific resource.
- If a component supports non-idempotent operations, then the component shall perform the steps illustrated in *Non-Idempotent Request Packet Processing*.

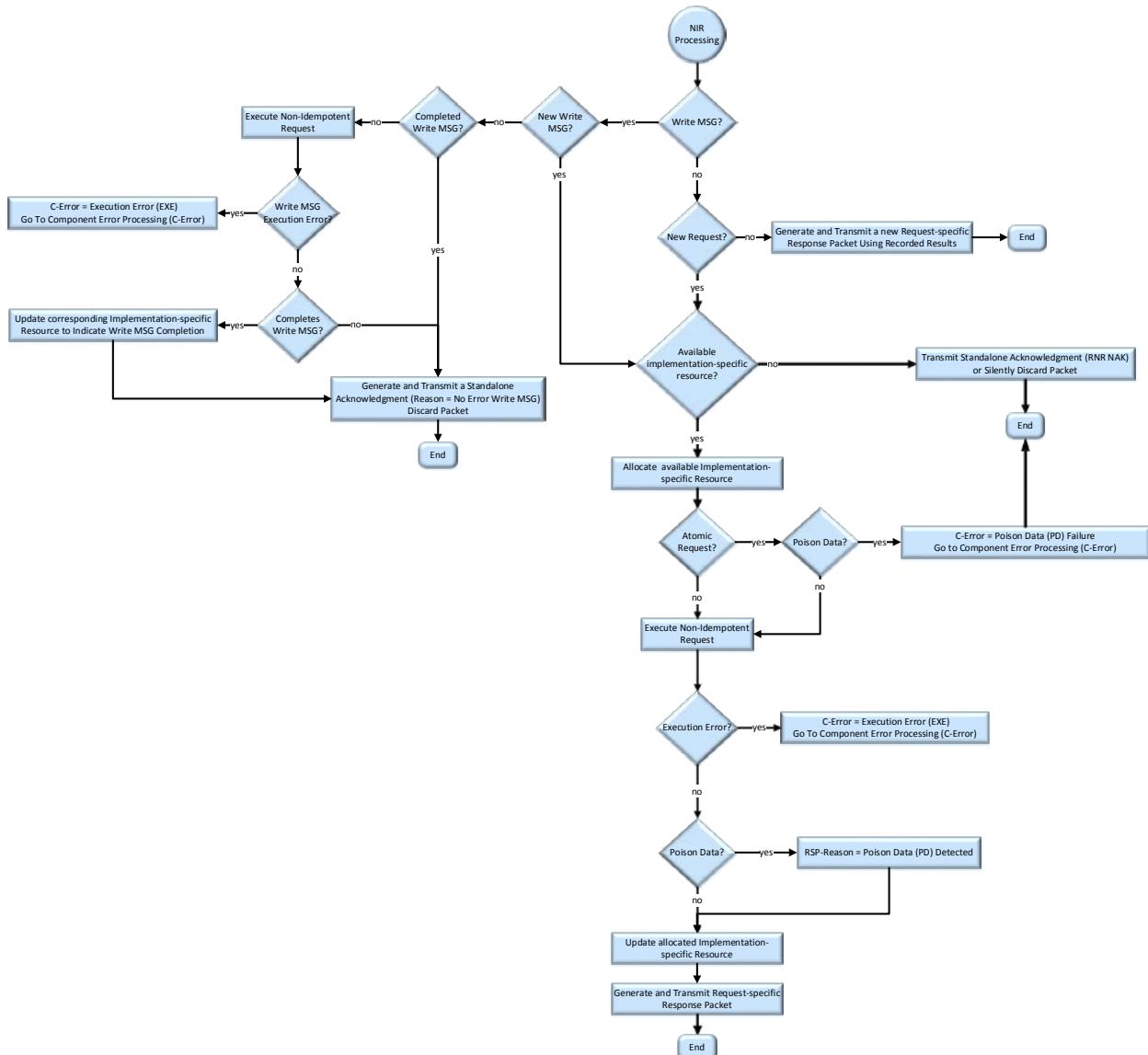


Figure 12-4: Non-Idempotent Request Packet Processing

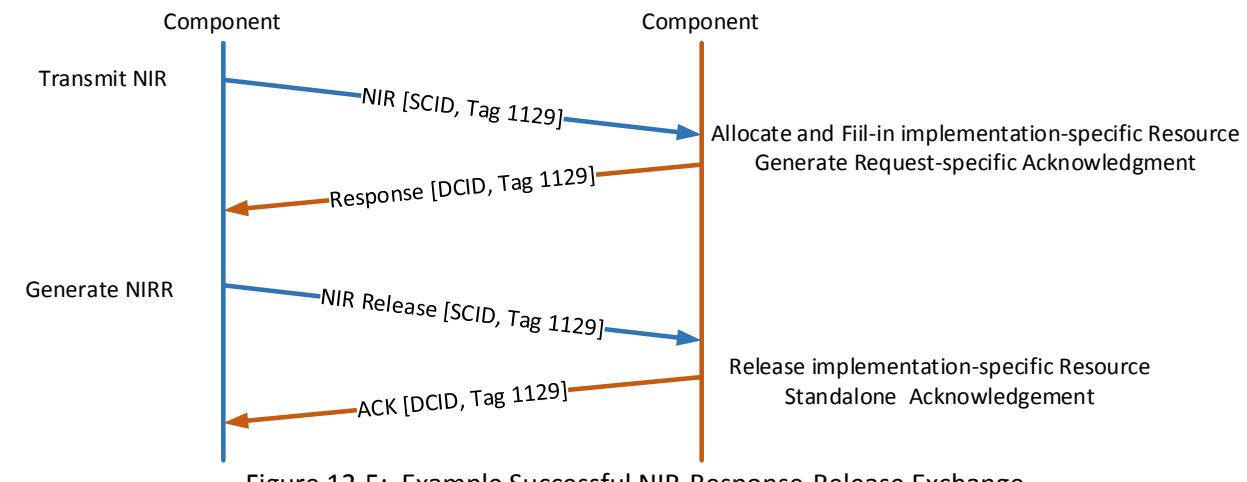
Table 12-1: Non-Idempotent Request Packet Details

Processing Step	Explanation
<b>Write MSG?</b>	If a reliable <i>Write MSG</i> request packet, then perform the <i>Write MSG</i> -specific packet validation and processing steps. Unreliable <i>Write MSG</i> request packets are idempotent, and are not processed per this flow chart.
<b>Write MSG: New Write MSG?</b>	If the <i>Write MSG</i> request packet does not correspond to an outstanding <i>Write MSG</i> operation, then this is a new operation and the Responder proceeds to the “Available Implementation-specific Resource” check to continue processing the request packet.

Processing Step	Explanation
<b>Write MSG: Completed Write MSG?</b>	<p>If the <i>Write MSG</i> request packet corresponds to an outstanding <i>Write MSG</i> operation, then determine if the operation was previously completed.</p>
<b>Write MSG: Write MSG Execution Error?</b>	<p>If the <i>Write MSG</i> request packet corresponds to a completed <i>Write MSG</i> operation, then transmit a <i>Standalone Acknowledgment</i> (Reason = No Error Write MSG), and discard the packet.</p> <p>If the <i>Write MSG</i> request packet does not correspond to a completed <i>Write MSG</i> operation, then execute the <i>Write MSG</i> request packet. Restart fail-safe timer.</p>
<b>Write MSG: Completes Write MSG?</b>	<p>If an execution error was detected, then proceed to <i>Component Error Processing</i> (EXE).</p>
<b>New Request?</b>	<p>If successful execution of the <i>Write MSG</i> request packet completes the operation, then update the implementation-specific resource to reflect this.</p> <p>Independent of whether the request packet completed the <i>Write MSG</i> operation, transmit a <i>Standalone Acknowledgment</i> (Reason = No Error Write MSG), and discard the packet.</p>
<b>Available Implementation-specific Resource?</b>	<p>If a new non-idempotent request packet, then proceed to the “Available Implementation-specific Resource?” step.</p> <p>If a duplicate non-idempotent non-<i>Write MSG</i> request packet (e.g., an <i>Atomic</i>), then transmit a new request-specific response packet using the results stored within the implementation-specific resource.</p>
<b>Atomic Request?</b>	<p>Does the component have implementation-specific resources available to track the new non-idempotent request packet and the associated execution results?</p> <p>If so, then allocate a resource, else then generate a <i>Responder Not Ready (RNR) NAK</i> or silently discard the request packet.</p>
<b>Execution Error?</b>	<p>If an <i>Atomic</i> request packet and poison data was detected, then the Responder shall proceed to <i>Component Error Processing</i> (Poison Data (PD) Failure) without executing the request packet.</p> <p>Else, execute the non-idempotent request packet.</p>
	<p>If non-idempotent request packet execution failed, then <i>Component Error Processing</i> (EXE).</p> <p>If a request packet execution is non-deterministic, then the Responder shall handle the request packet as specified in <i>Non-Deterministic Request Execution</i>. Once execution has</p>

Processing Step	Explanation
No EXE Error: Poison Data?	completed, then the Responder shall proceed with NIR packet processing as described in the subsequent table entries.
No EXE Error:	If poison data was detected during request packet execution, then set RSP-Reason = Poison Data (PD) Detected.  Perform the following steps: <ul style="list-style-type: none"> <li>• Update tracking logic and record the results. Multi-packet operations require additional tracking logic.</li> <li>• Start a failsafe timer to ensure tracking logic and resources are guaranteed to be released</li> <li>• Generate a request-specific response packet.</li> </ul>

*Example Successful NIR-Response-Release Exchange* illustrates an example of a normal NIR-Response-Release exchange, and *Example NIR-Response-Release Exchange with Retries* illustrates a more complex example where the NIR is retransmitted multiple times based on transient operating conditions.



5

Figure 12-5: Example Successful NIR-Response-Release Exchange

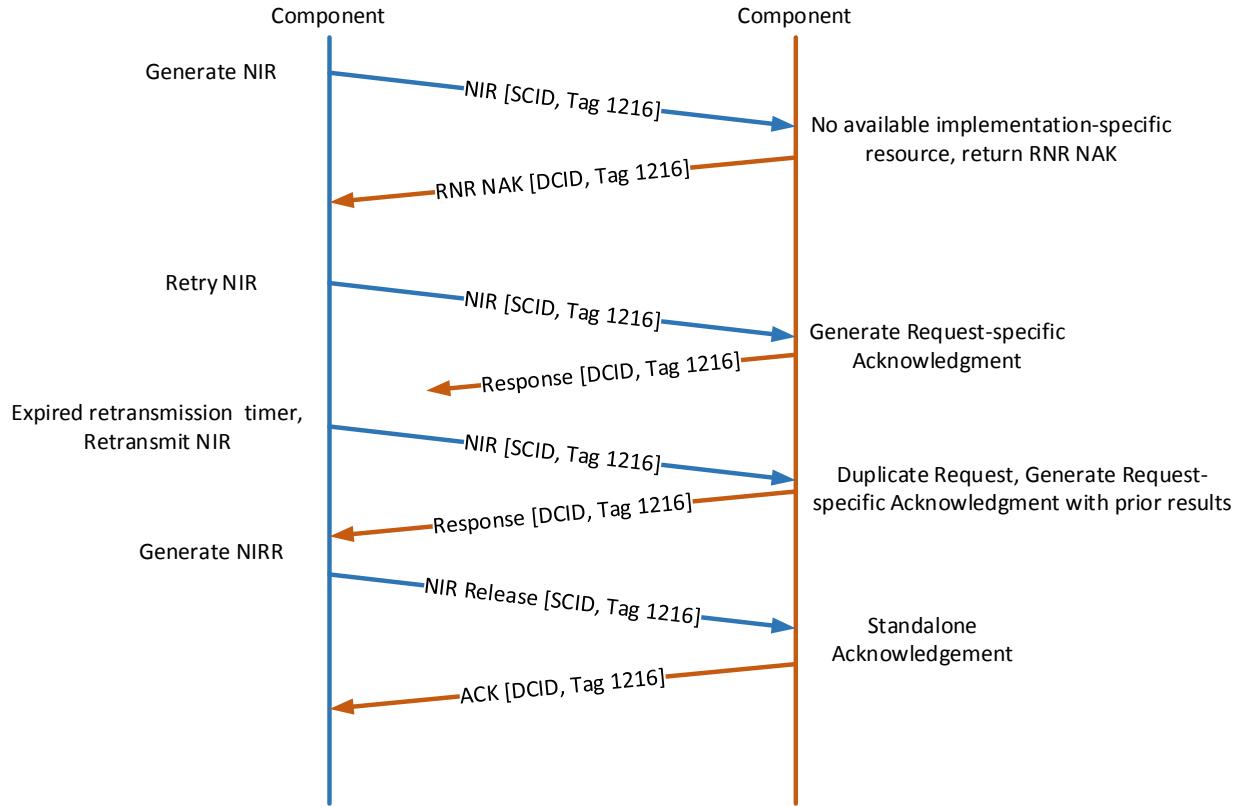


Figure 12-6: Example NIR-Response-Release Exchange with Retries

## 12.6. Packet Data Integrity

Gen-Z provides packet data integrity through a combination of CRCs (cyclic redundancy check) and packet protocol validation. Integrity is further enhanced if the underlying physical layer supports encoding. Gen-Z uses two CRCs per packet, and distributes specific packet protocol fields to provide:

- 100% detection of random 4-bit errors
- 100% detection of single-burst errors up to length 8
- >99.9999999% detection of more severe random errors and single-burst errors, as well as double-burst errors.

In addition, a Flit-based CRC may be used for an additional layer of protection (see *Physical Layer Abstraction*). In combination with the packet CRCs, this provides:

- 100% detection of random 6-bit errors.
- 100% detection of single-burst errors up to length 32.
- 100% detection of double-burst errors up to length 7, 7.
- ~100% detection of more severe random errors and multi-burst errors

Note: packet data integrity is orthogonal to component media data integrity (see *Memory Media RAS Architecture*).

## 12.6.1. Prelude CRC (PCRC)

To protect against errors which could cause the packet type and length to be incorrectly interpreted, all packet formats contain a Prelude CRC (PCRC).

The following are the features and requirements associated with PCRC and all packets:

- The PCRC code word is distributed across the first four bytes of each packet format. Distribution enhances burst error and lane failure detection.
- The PCRC shall be the 6-bit CRC specified in *Appendix B CRC*.
- Whenever a packet is (re-)transmitted, the source component's interface shall dynamically generate a PCRC and place it into the transmitted packet's PCRC field.
- All component ingress interfaces shall perform packet PCRC validation.
  - As a packet is received, the interface shall dynamically generate a PCRC.
  - If the dynamically-generated PCRC matches the packet's PCRC, then the packet proceeds to the next processing step. If the comparison fails, then the packet shall be handled per *Transient Error Clean-Up*.
  - An intermediate switch or transparent router shall not relay a packet until the PCRC has been successfully validated.
- If a component supports remapping any field covered by the PCRC, e.g., VC remapping, then the component shall perform PCRC validation prior to performing field remapping.

## 12.6.2. End-to-end CRC (ECRC)

With the exception of the *Link Idle* packet, all packet formats contain an ECRC field (for a link-local packet, end-to-end is between two interfaces, and for an end-to-end packet, end-to-end is between two or more components). The following are the features and requirements associated with ECRC:

- The 24-bit ECRC shall be as specified in *Appendix B CRC*.
- Whenever a packet is (re-)transmitted, the source component's interface shall dynamically generate an ECRC and place it into the transmitted packet's ECRC field.
- All component ingress interfaces shall perform packet ECRC validation.
  - As a packet is received, the interface shall dynamically generate an ECRC.
  - If the dynamically-generated ECRC matches the packet's ECRC field, then no further ECRC-related actions shall be taken.
  - If a link-local packet and the dynamically-generated ECRC does not match the link-local packet's ECRC field, then the packet shall be handled per *Transient Error Clean-Up*.
    - A *Link Idle* packet does not contain the ECRC field. If a *Link Idle* packet is indicated, the PCRC is valid, and packet validation fails, then the failure shall be treated as an ECRC error and handled per *Transient Error Clean-Up*.
  - If an end-to-end packet and the dynamically-generated ECRC does not match the end-to-end packet's ECRC field, then the stomped version of the dynamically-generated ECRC is compared to the packet's ECRC field. The stomped version is the one's complement of the dynamically-generated ECRC.
    - If these do not match, then the packet shall be handled per *Transient Error Clean-Up*.
    - If these match, then:

- 5
- If the packet has arrived at the destination component, then the packet shall be handled per *Transient Error Clean-Up*.
  - If the packet arrived at an intervening switch or transparent router and packet transmission has not begun on the component's egress interface, then the packet shall be silently discarded. If transmission has begun, then packet transmission shall be completed.
  - If a component supports remapping any field covered by the ECRC, e.g., VC remapping, then the component shall perform ECRC validation prior to performing field remapping.

### 12.6.3. Flit CRC

10 A Flit CRC is used with Flit Encoding to improve data integrity. The following are the features and requirements associated with Flit CRC:

- The 16-bit ECRC shall be as specified in *Appendix B CRC*.
- When a Flit CRC error is detected, a group of packets shall generally be treated as though all 15 packets suffered an ECRC error. This group includes all packets that are fully or partially within the Flit CRC code word. Further, this group shall be treated only as a single error event with regard to decrementing the Transient Error Threshold Counter, or updating other applicable statistics.
- When an ECRC error is detected, a *pair* of packets shall be treated as though both packets suffered an ECRC error. This pair includes the packet with the ECRC error, *and the subsequent packet*. If both packets in the pair suffered an ECRC error, then *Link Resynchronization* shall be initiated.

## 12.7. Error Detection and Recovery

25 Requirements for error detection and recovery vary by solution. Before delving further, within this document, the term 'silently discard' is used to describe a common portion of the packet error recovery process. The term is applicable to link-local and end-to-end packets and translates into the following actions:

- The component shall not execute the operation contained within the received packet.
- Resources associated with the packet shall be released and made available for use. If applicable, flow-control credits shall be updated to reflect the available resources.
- Unless explicitly indicated by this specification, the component shall not generate an error log or error event. A component may update applicable supported statistics.

## 12.8. Physical Layer Error Handling

There are two architectural errors defined for the physical layer:

- A *PHY\_Error* indicates the physical layer is non-operational. The underlying specific error is communicated via the *Interface PHY Structure*.
  - *PHY\_Error* transitions the *PHY-state* to *PHY-Down*.

- A PHY\_Degraded indicates the physical layer is operational but cannot deliver the performance or services possible under normal operating conditions.
    - PHY\_Degraded does not change the physical state, i.e., the physical remains in the PHY-Up state.
- 5 Physical layer errors shall take precedence over link-local and end-to-end packet errors.

## 12.9. Packet Validation and Processing

The following flow diagrams illustrate packet validation and error handling, starting with packet type identification through optional service-specific validation. The first step is to determine the packet type and the type of packet processing to be performed by the receiving component.

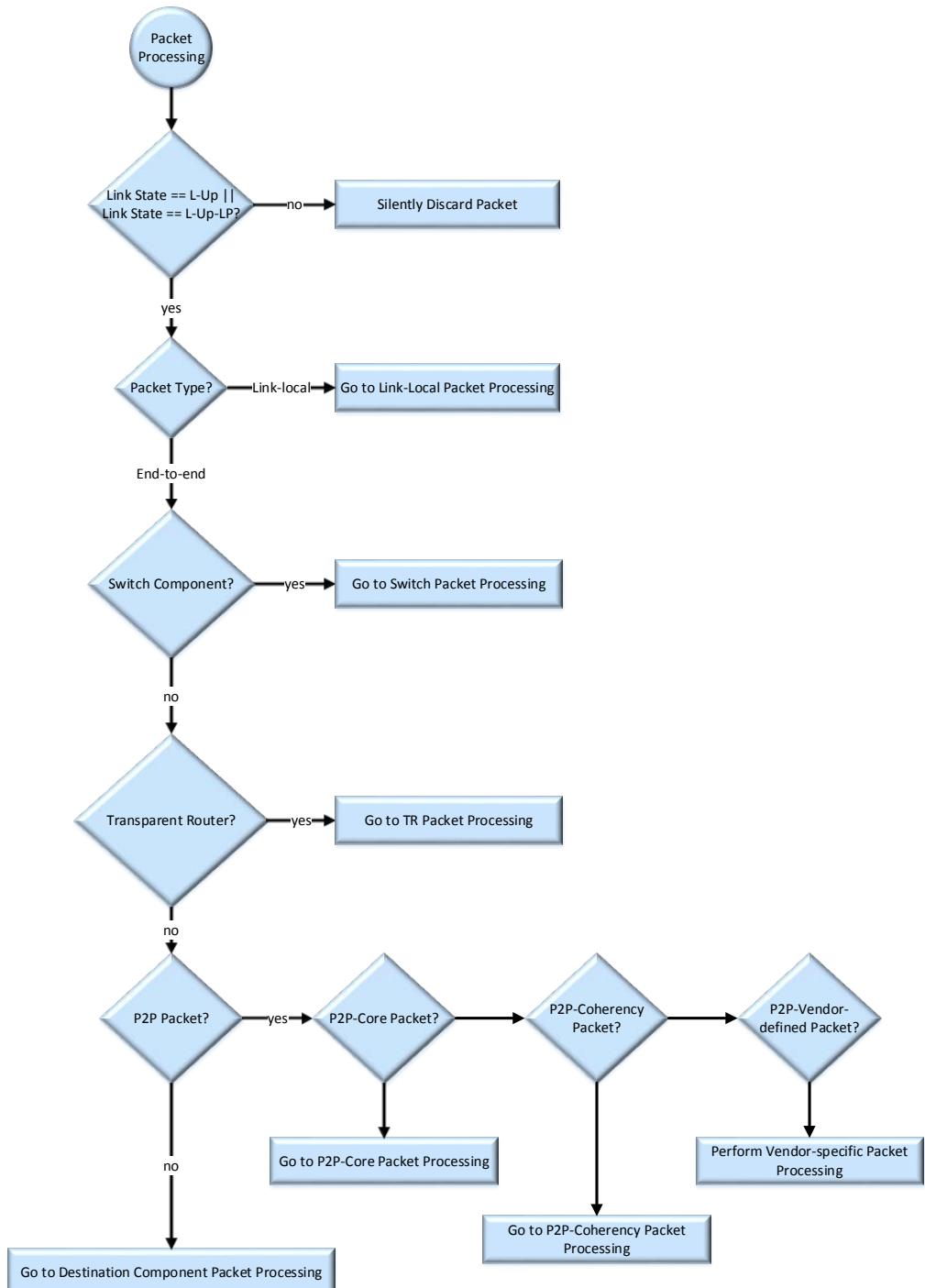


Figure 12-7: Packet Type Validation

Table 12-2: Packet Type Validation Details

Validation Step	Explanation
<b>Link State == L-Up    Link State == L-Up-LP?</b>	Is the link in an operational state? If not, then silently discard the packet.

Validation Step	Explanation
<b>Packet Type?</b>	<p>The packet's L field is examined to determine if this is a link-local packet or an end-to-end packet.</p> <p>If a link-local packet, proceed to <i>Link-local Packet Validation and Processing</i>.</p>
<b>Switch Component?</b>	<p>If the packet was received by a switch component, then proceed to <i>Switch Packet Processing</i>.</p>
<b>Transparent Router?</b>	<p>If the packet was received by a TR, then proceed to <i>Transparent Router Unicast Packet Processing</i>.</p>
<b>P2P?</b>	<p>If the packet was received on a component interface that is enabled for the P2P-Core, P2P-Coherency, or P2P-Vendor-defined OpClass, then determine which type, and proceed to the appropriate packet processing steps (<i>P2P-Core Packet Processing</i>, <i>P2P-Coherency Packet Processing</i>, perform vendor-specific packet processing), else, proceed to <i>Destination Component Packet Processing</i>.</p>

### 12.9.1. P2P-Core Packet Processing

If a component supports the P2P-Core OpClass on an interface, then it shall perform *P2P-Core Packet Processing* validation and processing as illustrated.

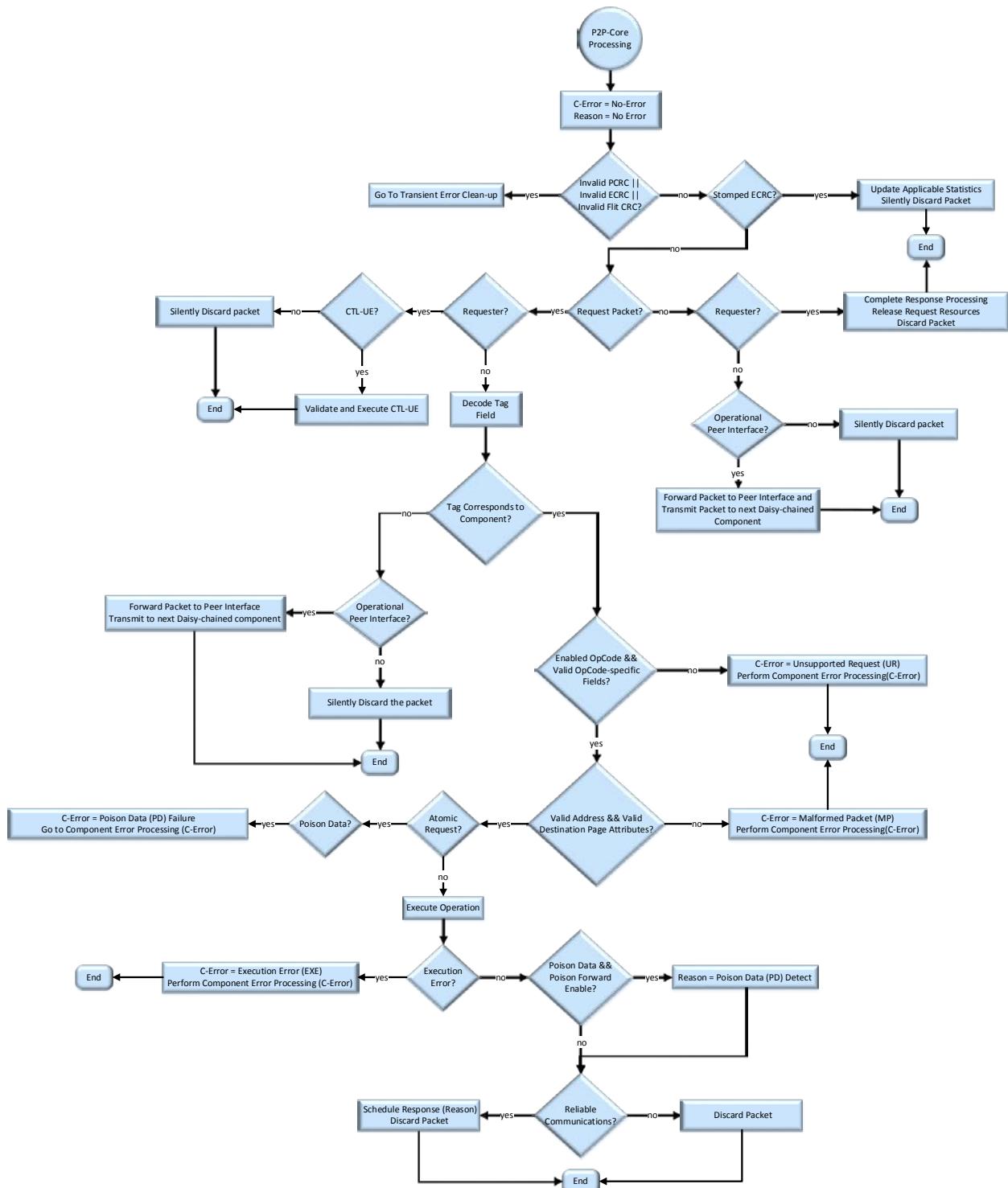


Figure 12-8: P2P-Core Packet Processing

Table 12-3: P2P-Core Packet Processing Details

Validation Step	Explanation
<b>Invalid PCRC    Invalid CRC   Invalid Flit CRC?</b>	If the packet's PCRC and ECRC fields do not match the dynamically-generated PCRC and ECRC values, or the flit's Flit CRC does not match the dynamically-generated Flit CRC, then the packet has suffered a transient error; proceed to <i>Transient Error Clean-Up</i> .
<b>Stomped ECRC?</b>	If the ECRC field was stomped, then the packet is invalid and is silently discarded. If not, then the packet proceeds to the next processing step.
<b>Request Packet?</b>	If a Request Packet, then continue with request packet processing path, else proceed with response processing path.
<b>Request: Requester?</b>	If a request packet on an interface enabled to exchange P2P-Core OpClass packets ( <i>Interface I-CAP 1 Control OpClass Select == 0x1</i> ) and the destination is a Requester, then determine if a <i>CTL-UE Packet</i> .
<b>Requester: CTL-UE?</b>	If a <i>CTL-UE Packet</i> , then validate and execute it. A <i>CTL-UE Packet</i> is the only request packet that may be received by a Requester in a daisy-chain topology. If not, then silently discard the packet. The error / misconfiguration will be detected by the originating Requester.
<b>Request: Tag Corresponds to Component?</b>	Determine if the request packet is intended for this component.
<b>Not this Component: Operational Peer Interface?</b>	If <i>Interface I-CAP 1 Control Daisy-chain Peer Interface Configured == 1b</i> , and the interface identified by the <i>Interface Structure Peer Daisy Interface ID</i> field is in the I-Up state, then forward the request packet to the peer interface and transmit the packet to the next component. If there isn't an operational peer interface, then silently discard the packet.
<b>Enabled OpCode &amp;&amp; Validate OpCode-specific fields?</b>	Validate the P2P-Core OpClass and associated OpCode are enabled, and the OpCode-specific fields are valid. If any are invalid, then proceed to <i>Component Error Processing (UR)</i> .
<b>Valid Address &amp;&amp; Valid Destination Page Attributes?</b>	Validate the address and operation length are within the component's Data Space or Control Space. Verify that the destination page attributes support the requested operation. If any are invalid, then proceed to <i>Component Error Processing (MP)</i> .

Validation Step	Explanation
<b>Atomic Request?</b>	<p>If an Atomic request packet and poison data was detected, then the Responder shall proceed to <i>Component Error Processing</i> (Poison Data (PD) Failure) without executing the request packet.</p> <p>Else, execute the request packet.</p>
<b>Execution Error?</b>	<p>Was an error detected during the execution of the packet? There are numerous potential causes depending upon the OpCode, component state, etc. If an error was detected, then the packet proceeds to <i>Component Error Processing</i> (EXE).</p>
<b>Poison Data and Poison Forward Enable?</b>	<p>If poison data was detected and (Primary Media and <i>Primary Media CAP 1 Control</i> Poison Forwarding Enabled == 1b) or (Secondary Media and <i>Secondary Media CAP 1 Control</i> Poison Forwarding Enabled == 1b), then update the applicable statistics, log the error information, and return the request-specific response packet with (Reason = Poison Data (PD) Detected).</p>
<b>Reliable Communications?</b>	<p>If the request packet requires a request-specific response packet or <i>Interface I-CAP 1 Control</i> Omit P2P Standalone Acknowledgment == 0b, then the Responder shall transmit a response packet.</p>

### 12.9.2. P2P-Coherency Packet Processing

If a component supports the P2P-Coherency OpClass on an interface, then it shall perform *P2P-Coherency Packet Processing*.

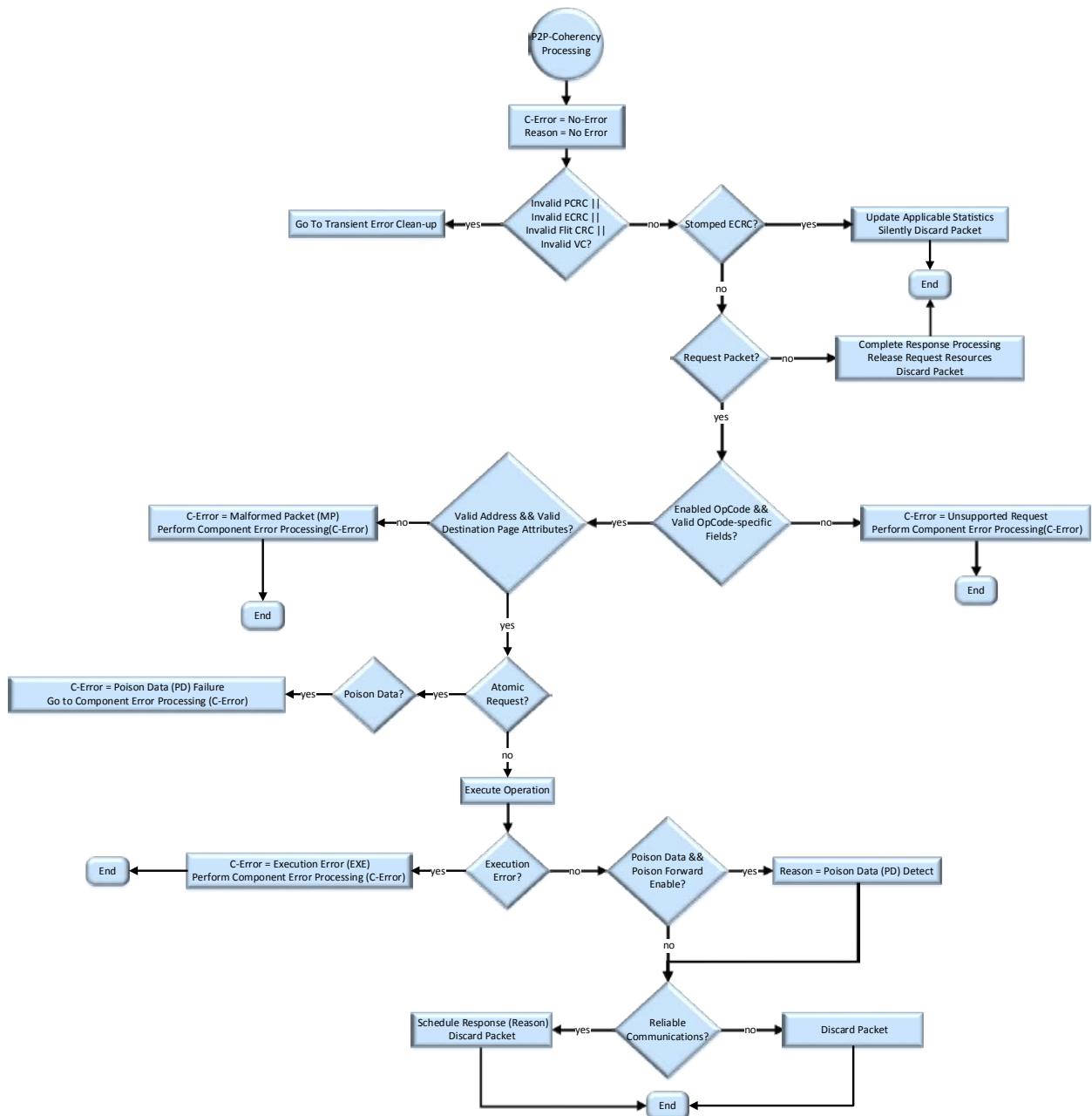


Figure 12-9: P2P-Coherency Packet Processing

Table 12-4: P2P-Coherency Packet Processing Details

Validation Step	Explanation
<b>Invalid PCRC    Invalid CRC   Invalid Flit CRC    Invalid VC?</b>	If the packet's PCRC and ECRC fields do not match the dynamically-generated PCRC and ECRC values, or the flit's Flit CRC does not match the dynamically-generated Flit CRC, or the packet's VC is invalid (packet's VC field does not contain a VC that is enabled on the receiving interface), then the packet has suffered a transient error; proceed to <i>Transient Error Clean-Up</i> .

Validation Step	Explanation
<b>Stomped ECRC?</b>	If the ECRC field was stomped, then the packet is invalid and is silently discarded. If not, then the packet proceeds to the next processing step.
<b>Request packet?</b>	If no, then complete response packet processing, release request tracking resources, and discard the packet. If yes, then proceed with packet validation.
<b>Enabled OpCode &amp;&amp; Validate OpCode-specific fields?</b>	Validate the P2P-Coherency OpClass and associated OpCode are enabled, and the OpCode-specific fields are valid. If any are invalid, then proceed to <i>Component Error Processing</i> (UR).
<b>Valid Address &amp;&amp; Valid Destination Page Attributes?</b>	<p>Validate the address and operation length are within the component's Data Space or Control Space.</p> <p>Verify that the destination page attributes support the requested operation, e.g., if a coherency request packet, then in addition to validating that the page permits coherency operations, validate that the Responder cache agent associated with the page is permitted to execute the operation on the destination page.</p> <p>If any are invalid, then proceed to <i>Component Error Processing</i> (MP).</p>
<b>Atomic Request?</b>	<p>If an Atomic request packet and poison data was detected, then the Responder shall proceed to <i>Component Error Processing</i> (Poison Data (PD) Failure) without executing the request packet.</p> <p>Else, execute the request packet.</p>
<b>Execution Error?</b>	Was an error detected during the execution of the packet? There are numerous potential causes depending upon the OpCode, component state, etc. If an error was detected, then the packet proceeds to <i>Component Error Processing</i> (EXE).
<b>Poison Data and Poison Forward Enable?</b>	If poison data was detected and (Primary Media and <i>Primary Media CAP 1 Control</i> Poison Forwarding Enabled == 1b) or (Secondary Media and <i>Secondary Media CAP 1 Control</i> Poison Forwarding Enabled == 1b), then update the applicable statistics, log the error information, and return the request-specific response packet with (Reason = Poison Data (PD) Detected).
<b>Reliable Communications?</b>	If the request packet requires a request-specific response packet or <i>Interface I-CAP 1 Control</i> Omit P2P Standalone Acknowledgment == 0b, then the Responder shall transmit a response packet.

### 12.9.3. Destination Component Packet Processing

If a component supports explicit OpClass, then it shall perform end-to-end packet validation and processing as illustrated in the following figures.

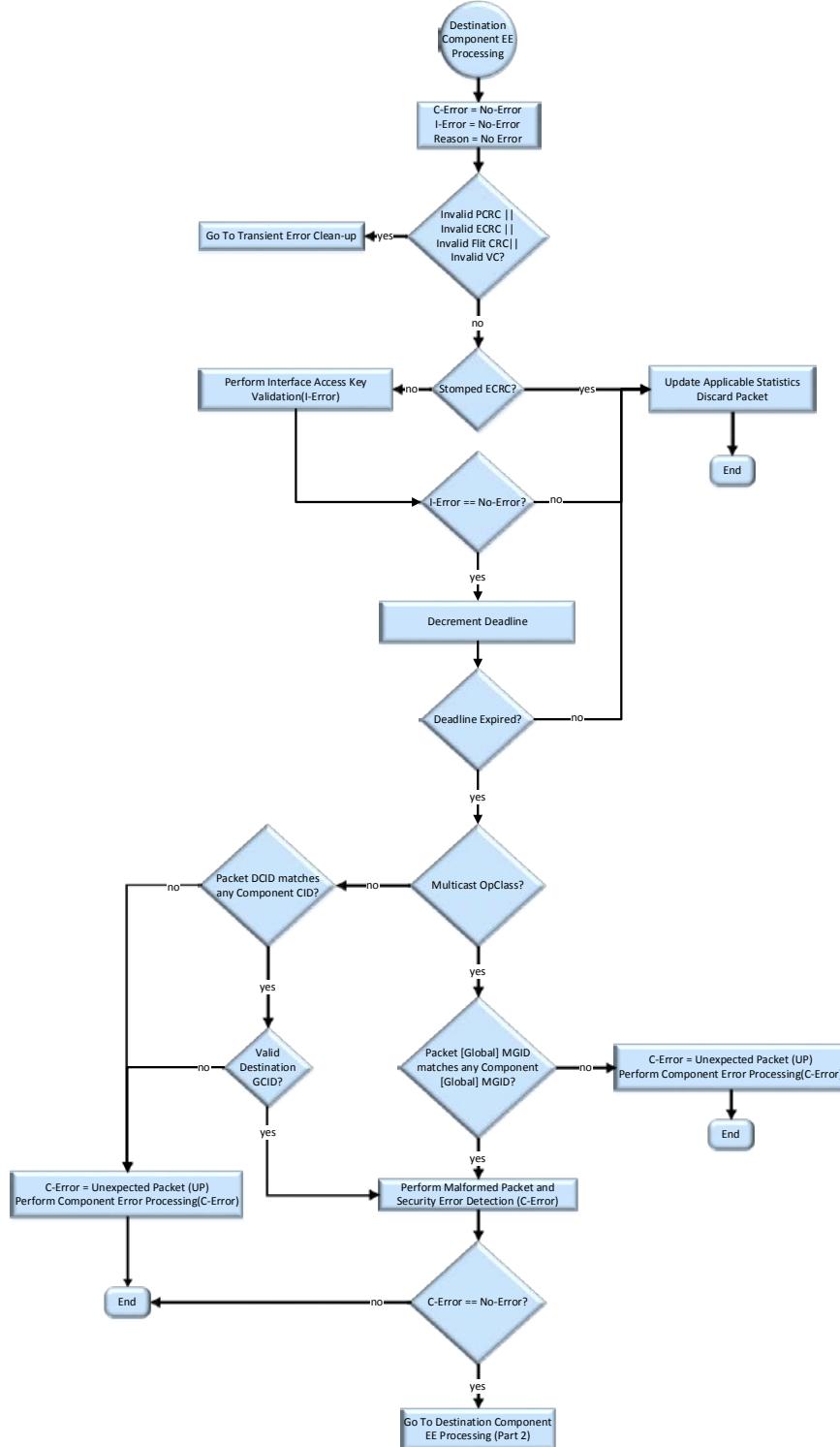


Figure 12-10: Destination Component Packet Processing (Part 1)

Table 12-5: Destination Component Packet Processing Part 1 Details

Validation Step	Explanation
<b>Invalid PCRC    Invalid CRC   Invalid Flit CRC    Invalid VC?</b>	<p>If the packet's PCRC and ECRC fields do not match the dynamically-generated PCRC and ECRC values, or the flit's Flit CRC does not match the dynamically-generated Flit CRC, or the packet's VC is invalid (packet's VC field does not contain a VC that is enabled on the receiving interface), then the packet has suffered a transient error; proceed to <i>Transient Error Clean-Up</i>.</p>
<b>Stomped ECRC?</b>	<p>If the ECRC field was stomped, then the packet is invalid and is silently discarded. If not, then the packet proceeds to the next processing step.</p>
<b>Interface Access Key Validation: I-Error == No Error?</b>	<p>Validate that the packet's Access Key matches an Access Key configured on the receiving interface. If an error is detected, then update applicable statistics, and discard the packet. If and <i>I-Error Detect Interface Access Key Violation Detect == 1b</i>, then take configured error handling actions.</p>
<b>Decrement Deadline: Deadline Expired?</b>	<p>Decrement Deadline (see <i>Deadline Semantics</i>), and if the packet's Deadline is not valid, then discard the packet and update applicable statistics.</p>
<b>Multicast OpClass?</b>	<p>Determine if a multicast or unicast packet.</p>
<b>Unicast: Packet DCID matches any Component CID?</b>	<p>If the component is in the C-CFG state, then skip this step.</p> <p>If the component is not in the C-CFG state and Up and the packet's DCID field does not match any configured component CID, then proceed to <i>Component Error Processing (UP)</i>.</p>
<b>Unicast: Valid Destination GCID?</b>	<p>If the component is in the C-CFG state, then skip this step.</p> <p>If the component is not in the C-CFG state and Up and GC == 1b and the packet's DSID field does not match the configured component SID, then proceed to <i>Component Error Processing (UP)</i>.</p>
<b>Multicast: Packet [Global] MGID matches any Component [Global] MGID?</b>	<p>If the packet's MGID / GMGID field does not match any configured component MGID / GMGID, then proceed to <i>Component Error Processing (UP)</i>.</p>
<b>Perform Malformed Packet and Security Error Detection(C-Error)</b>	<p>The component executes the steps specified in <i>MP and Security Error Detection</i>.</p>
<b>C-Error == No Error?</b>	<p>If an error was detected, then stop further processing as the error was handled during <i>MP and Security Error Detection</i>.</p>

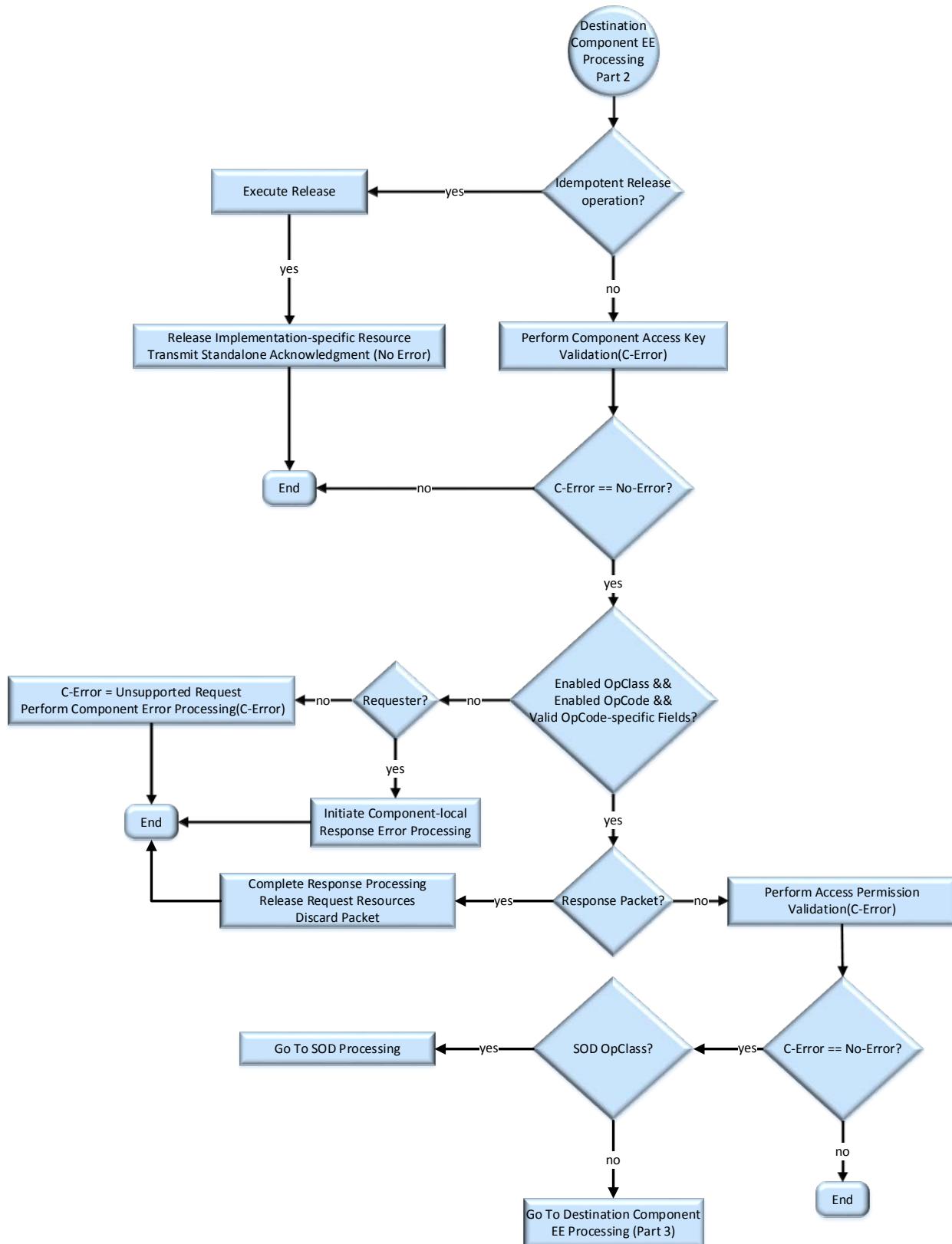


Figure 12-11: Destination Component Packet Processing (Part 2)

Table 12-6: Destination Component Packet Processing Part 2 Details

Validation Step	Explanation
<b>Idempotent Release?</b>	If a <i>Non-Idempotent Request Release (NIRR)</i> packet, then execute the request packet, release application-specific resources, and transmit a <i>Standalone Acknowledgment</i> (Reason = No Error).
<b>Perform Access Key Validation</b>	The component executes the steps specified in <i>Access Key Validation</i> .
<b>C-Error == No Error?</b>	If an error was detected, then stop further processing as the error was handled during <i>Access Key Validation</i> .
<b>Enabled OCL &amp;&amp; Enabled OpCode &amp;&amp; Valid OpCode-specific Fields?</b>	Validate the packet's OpClass (implicit or explicit) and associated OpCode are enabled, and the OpCode-specific fields are valid.
<b>Invalid Op: Requester?</b>	If the component's role for processing this packet is as a Requester, then initiate component-local error processing. If the component's role for processing this packet is as a Responder, then proceed to <i>Component Error Processing (UR)</i> .
<b>Response Packet?</b>	If a response packet, then complete the response processing, release request resources, and discard the packet. If there are any errors while processing the response packet, then initiate component-local response error processing. End packet processing.
<b>Valid Op: Perform Access Permission Validation</b>	The component executes the steps specified in <i>Access Permission Validation</i> .
<b>C-Error == No Error?</b>	If an error was detected, then stop further processing as the error was handled during <i>Access Permission Validation</i> .
<b>SOD OpClass?</b>	If a SOD OpClass packet, then proceed to <i>SOD OpClass Packet Processing</i> , else proceed to Destination Component EE Processing 3.

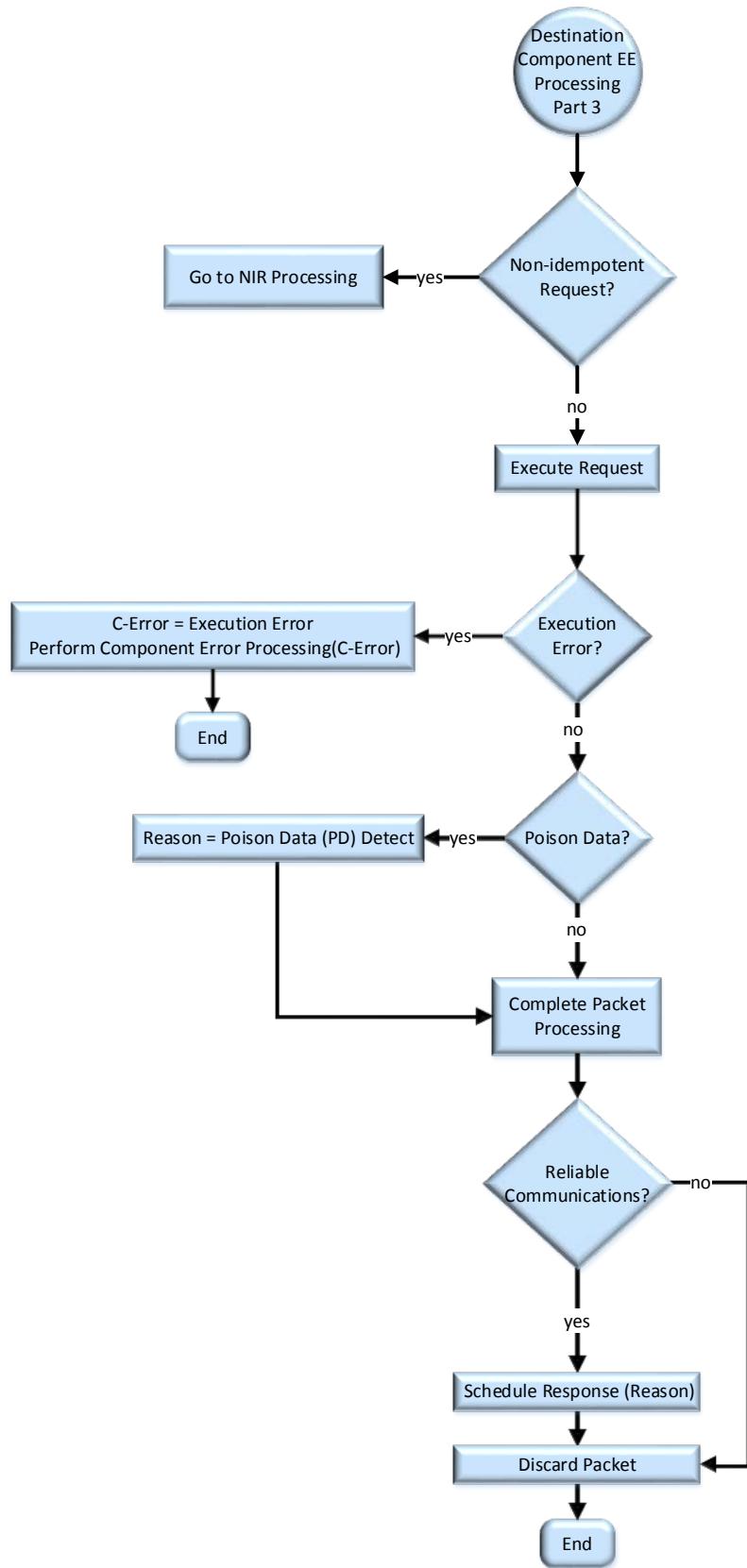


Figure 12-12: Destination Component Packet Processing (Part 3)

Table 12-7: Destination Component Packet Processing Part 3 Details

Validation Step	Explanation
<b>Non-Idempotent Request?</b>	If a non-idempotent request packet, then perform the steps illustrated in <i>Non-Idempotent Request Packet Processing</i> .
<b>Execution Error?</b>	Was an error detected during the execution of the packet? There are numerous potential causes depending upon the OpCode, component state, etc. If an error was detected, then the packet proceeds to <i>Component Error Processing</i> (EXE).
<b>Poison Data?</b>	If poison data encountered while executing the packet, then set temporary variable to (Reason = Poison Data (PD) Detected).
<b>Reliable Communications?</b>	If the component requires a response packet for the request packet, then schedule the corresponding response packet with the Reason value.

#### 12.9.4. Component Error Processing

A component shall perform component error processing as illustrated in *Component Error Processing*.

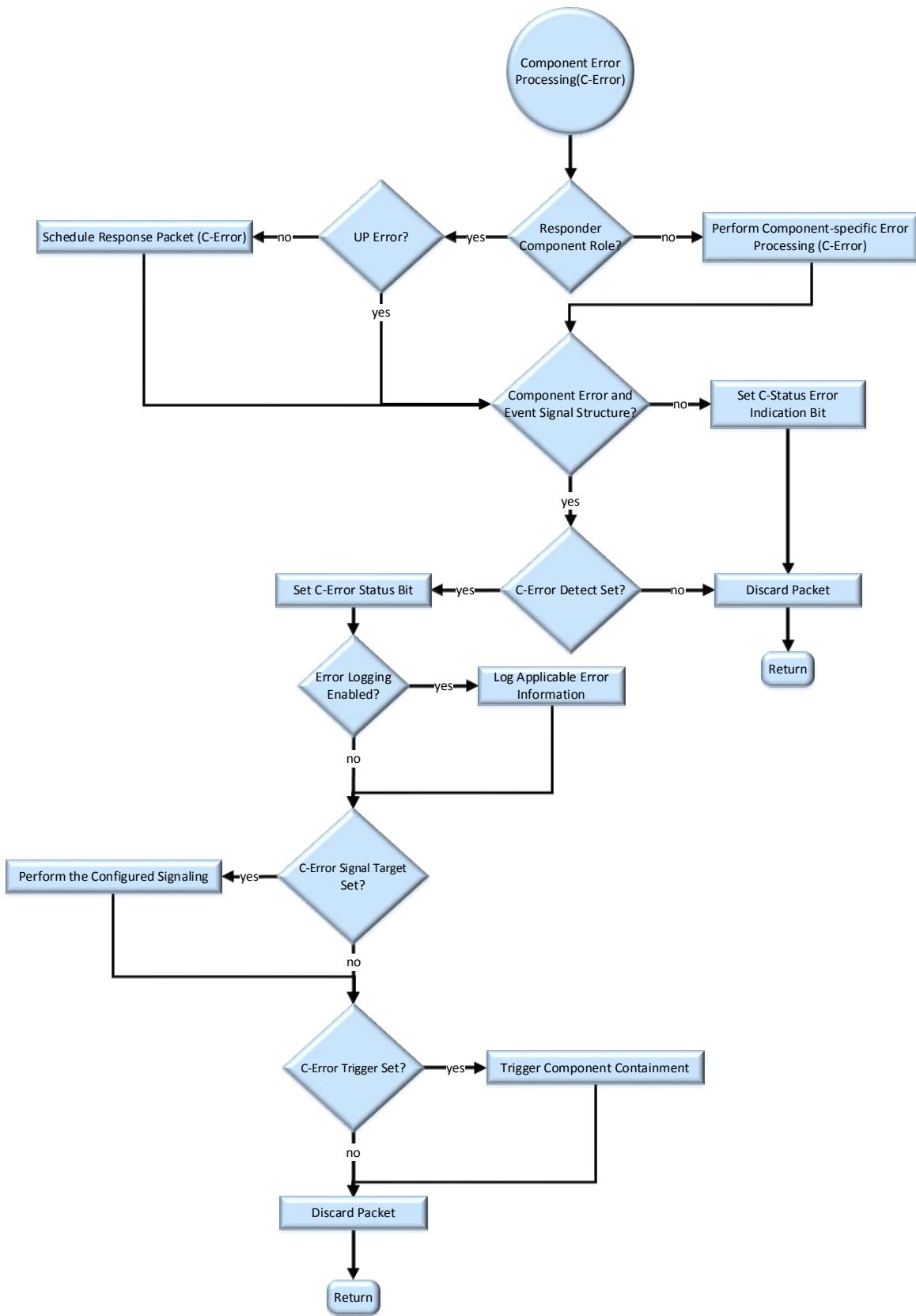


Figure 12-13: Component Error Processing

Table 12-8: Component Error Processing Details

Validation Step	Explanation
<b>Responder Role</b> <b>Requester Role</b>	If a UP error was detected, then do not generate a response packet (misrouted packets are dropped, not acknowledged). If a non-UP error was detected, then generate a response packet with the indicated C-Error.
<b>Component Error and Signal Event Structure?</b>	Perform the component-specific error processing associated with the packet. If the component does not support the <i>Component Error and Signal Event Structure</i> , then update the <i>Core C-Status</i> field to indicate an error was detected and is basic type (e.g., fatal, non-fatal, etc.), but no details can be provided. Discard the packet, and perform any associated component-specific processing.
<b>C-Error Detect Set?</b>	If the bit corresponding to C-Error within the <i>C-Error Detect</i> field is not set to 1b, then silently discard the packet, and perform any associated component-specific processing. Else, set the bit corresponding to C-Error within the <i>C-Error Status</i> field to 1b.
<b>Error Logging Enabled?</b>	If error logging is enabled and there are available log entries, then log the applicable error information.
<b>C-Error Signal Target Set?</b>	If the subfield corresponding to C-Error within the <i>Component Error and Signal Event Structure</i> C-Error Signal Target subfield is non-zero, then perform the configured signaling actions.
<b>C-Error Trigger Set?</b>	If the bit corresponding to C-Error within the <i>C-Error Trigger</i> is set to 1b, then trigger <i>Component Containment</i> .

### 12.9.5. Switch Packet Processing

Switches shall validate relayed end-to-end packets as illustrated in the following figures.

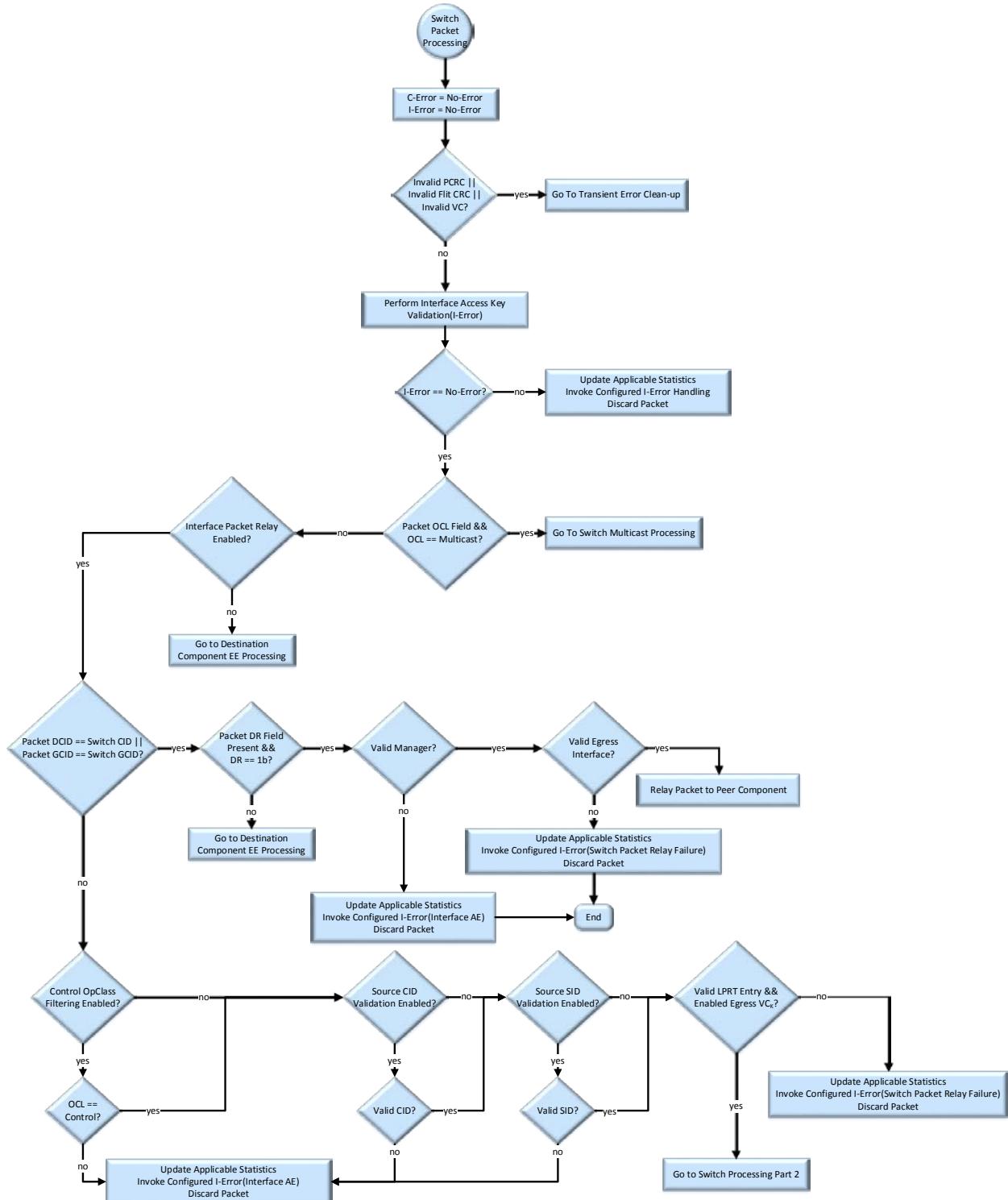


Figure 12-14: Switch Packet Processing (Part 1)

Table 12-9: Switch Packet Processing Part 1 Details

Validation Step	Explanation
<b>Invalid PCRC    Invalid Flit CRC    Invalid VC?</b>	If the packet's PCRC field does not match the dynamically-generated PCRC value, or the flit's Flit CRC does not match the dynamically-generated Flit CRC, or the packet's VC is invalid (packet's VC field does not contain a VC that is enabled on the receiving interface), then the packet has suffered a transient error; proceed to <i>Transient Error Clean-Up</i> .
<b>Interface Access Key Validation: I-Error == No Error?</b>	Validate that the packet's Access Key matches an Access Key configured on the receiving interface. If an error is detected, then update applicable statistics, and discard the packet. If and <i>I-Error Detect Interface Access Key Violation Detect == 1b</i> , then take configured error handling actions.
<b>Packet OCL Field &amp;&amp; OCL == Multicast?</b>	If the OCL field is present and it equals the Multicast OpClass, then go to Switch Multicast Processing.
<b>Interface Packet Relay Enabled?</b>	If <i>Interface I-CAP 1 Control LPRT</i> or <i>MPRT</i> is enabled, then proceed with switch packet processing, else proceed to <i>Destination Component Packet Processing</i> to complete packet processing.
<b>Packet DCID == Switch CID    Packet GCID == Switch GCID?</b>	Is this packet destined for this switch component? Compare the packet's DCID and DSID (if applicable) to the component <i>Core Structure</i> CIDs and SIDs (if applicable).
<b>Switch: Packet DR Field Present &amp;&amp; Packet DR == 1b?</b>	If a packet contains a DR field and DR == 1b, then proceed with DR packet validation steps, else proceed to <i>Destination Component Packet Processing</i> to complete packet processing.
<b>DR == 1b: Valid Manager?</b>	Is the packet is from a configured management component, i.e., does the packet's SCID and SSID, if applicable, match the switch's component identifier or one of the following: Primary Manager, Primary Fabric Manager, or Secondary Fabric Manager.
	<p>If the <i>Core Structure</i> MGR-UUID is zero, then no additional access permission checks are required.</p> <p>If non-zero and the packet contains a MGR-UUID field, then compare that to the <i>Core Structure</i> MGR-UUID. If they match, then access permission is granted; if they do not match, then update applicable statistics, discard the packet, and take error handling actions as configured in the <i>Interface Error Fields</i> (Interface AE).</p>
<b>DR == 1b: Valid Egress Interface?</b>	Validate the DR-Interface matches an actual Switch Interface. If valid, then relay the packet through the egress interface.

Validation Step	Explanation
<b>Control OpClass Packet Filtering Enabled?</b> <b>Enabled: OCL == Control?</b>	If not valid, then update applicable statistics, discard the packet, and take error handling actions as configured in the <i>Interface Error Fields</i> (Switch Packet Relay Failure). If <i>Interface I-CAP 1 Control</i> Control OpClass Packet Filtering Enable == 1b, then validate CID else proceed to next step.
<b>Source CID Validation Enabled?</b> <b>Enabled: Valid CID?</b>	If packet's OCL == Control, then proceed to next validation step, else update applicable statistics, discard the packet, and take error handling actions as configured in the <i>Interface Error Fields</i> (Interface AE).
<b>Source SID Validation Enabled?</b> <b>Enabled: Valid SID?</b>	If <i>Interface I-CAP 1 Control</i> Source CID Packet Validation Enable == 1b, then validate CID else proceed to next step If packet's SCID == Peer CID, then proceed to next validation step, else update applicable statistics, discard the packet, and take error handling actions as configured in the <i>Interface Error Fields</i> (Interface AE).
<b>Valid LPRT Entry &amp;&amp; Enabled Egress VC<sub>k</sub>?</b>	If <i>Interface I-CAP 1 Control</i> Source SID Packet Validation Enable == 1b, then validate SID else proceed to next step If packet's GC == 0b    (GC == 1b && SSID == Peer SID), then proceed to next validation step, else update applicable statistics, discard the packet, and take error handling actions as configured in the <i>Interface Error Fields</i> (Interface AE).
	If the packet's DCID does not correspond to a valid Linear Packet Relay Table entry or if the packet's VC does not correspond to an enabled egress interface VC, then update applicable statistics, invoke error handling as configured in <i>Interface Error Fields</i> (Switch Packet Relay Failure) handling, and discard packet. If valid and enabled, then proceed to the next set of switch processing.

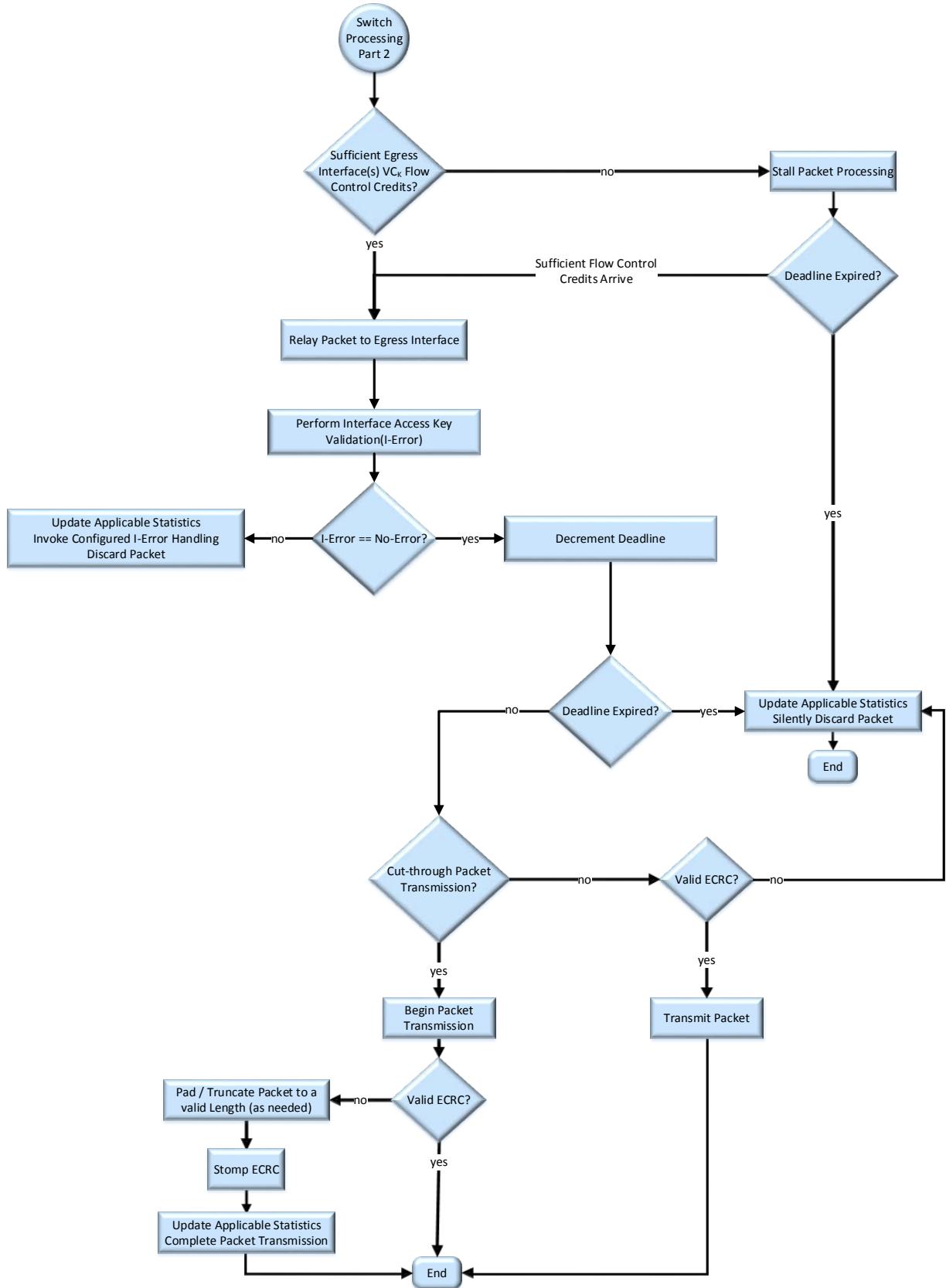


Figure 12-15: Switch Packet Validation and Processing (Part 2)

Table 12-10: Switch Packet Validation and Processing Part 2 Details

Validation Step	Explanation
<b>Sufficient Egress Interface(s) VC<sub>k</sub> Flow-Control Credits?</b>	Does the VC of the egress interface(s) corresponding to the packet's VC field have sufficient flow-control credits to enable packet relay?
<b>Switch Egress Access Key Validation Enabled?</b>	Is egress Access Key validation enabled?
<b>Interface Access Key Validation: I-Error == No Error?</b>	Validate that the packet's Access Key matches an Access Key configured on the transmitting interface. If an error is detected, then update applicable statistics, and discard the packet. If and <i>I-Error Detect</i> Interface Access Key Violation Detect == 1b, then take configured error handling actions.
<b>Decrement Deadline: Deadline Expired?</b>	Decrement Deadline (see <i>Deadline Semantics</i> ), and if the packet's Deadline is not valid, then discard the packet and update applicable statistics.
<b>Cut-through Packet Transmission?</b>	Is the packet being relayed and transmitted using cut-through or store-and-forward logic?
<b>Not Cut-Through: Valid ECRC?</b>	If an invalid ECRC, then update appropriate statistics and silently discard the packet.
<b>Cut-through: Begin Packet Transmission</b>	If a valid ECRC, then transmit the packet.
<b>Cut-through: Valid ECRC?</b>	Begin transmitting the packet
<b>Cut-through: Invalid ECRC?</b>	If during packet relay or transmission, then the packet's ECRC field is valid, then packet processing is complete.
	A valid ECRC field is one that either matches the dynamically-generated ECRC or the stomped version of the dynamically-generated ECRC. If either is not true, then an invalid ECRC is detected.
	Prior to transmitting the ECRC field, was an invalid ECRC detected? If not then:
	<ul style="list-style-type: none"> <li>Pad or truncate the packet to match the packet's LEN field.</li> <li>Place the stomped version of the dynamically-generated ECRC into the packet's ECRC field.</li> <li>Update applicable statistics and complete packet transmission.</li> </ul>

### 12.9.5.1. **Switch Multicast Packet Handling**

A switch shall perform multicast packet handling as illustrated in the following figures.

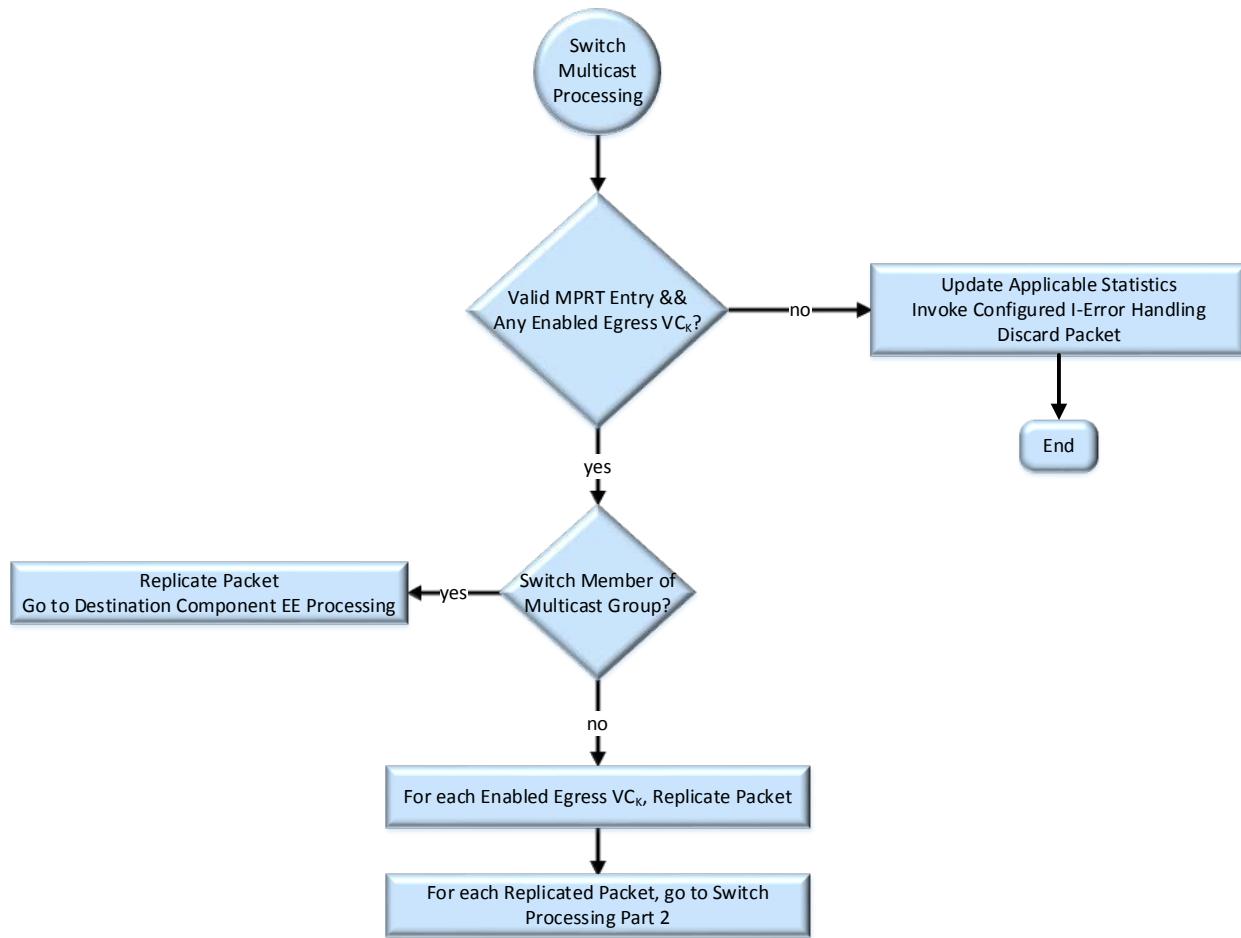


Figure 12-16: Switch Multicast Processing

Table 12-11: Switch Multicast Processing Details

Validation Step	Explanation
<b>Valid MPRT Entry &amp;&amp; Any Enabled Egress VC<sub>k</sub>?</b>	If the packet's MGID field does not match an enabled Multicast Packet Relay Table or if the packet's VC field does not match VC of any of the egress interfaces associated with this multicast group, then update applicable statistics, invoke configured <i>Interface Error Fields</i> Switch Packet Relay Failure handling, and discard packet.
<b>Switch Member of Multicast Group?</b>	If the switch is a member of the multicast group (subnet-local or global multicast group), then replicate the packet for its own consumption and proceed to <i>Destination Component Packet Processing</i> .
<b>For each enabled Egress VC<sub>k</sub>, Replicate Packet</b>	For each egress interface associated with the multicast group and with a valid VC, replicate the packet, and proceed to Switch Packet Processing Part 2 to relay and transmit the packet on each egress interface.

## 12.9.6. Transparent Router Unicast Packet Processing

A TR shall perform unicast packet processing as illustrated in the following figures.

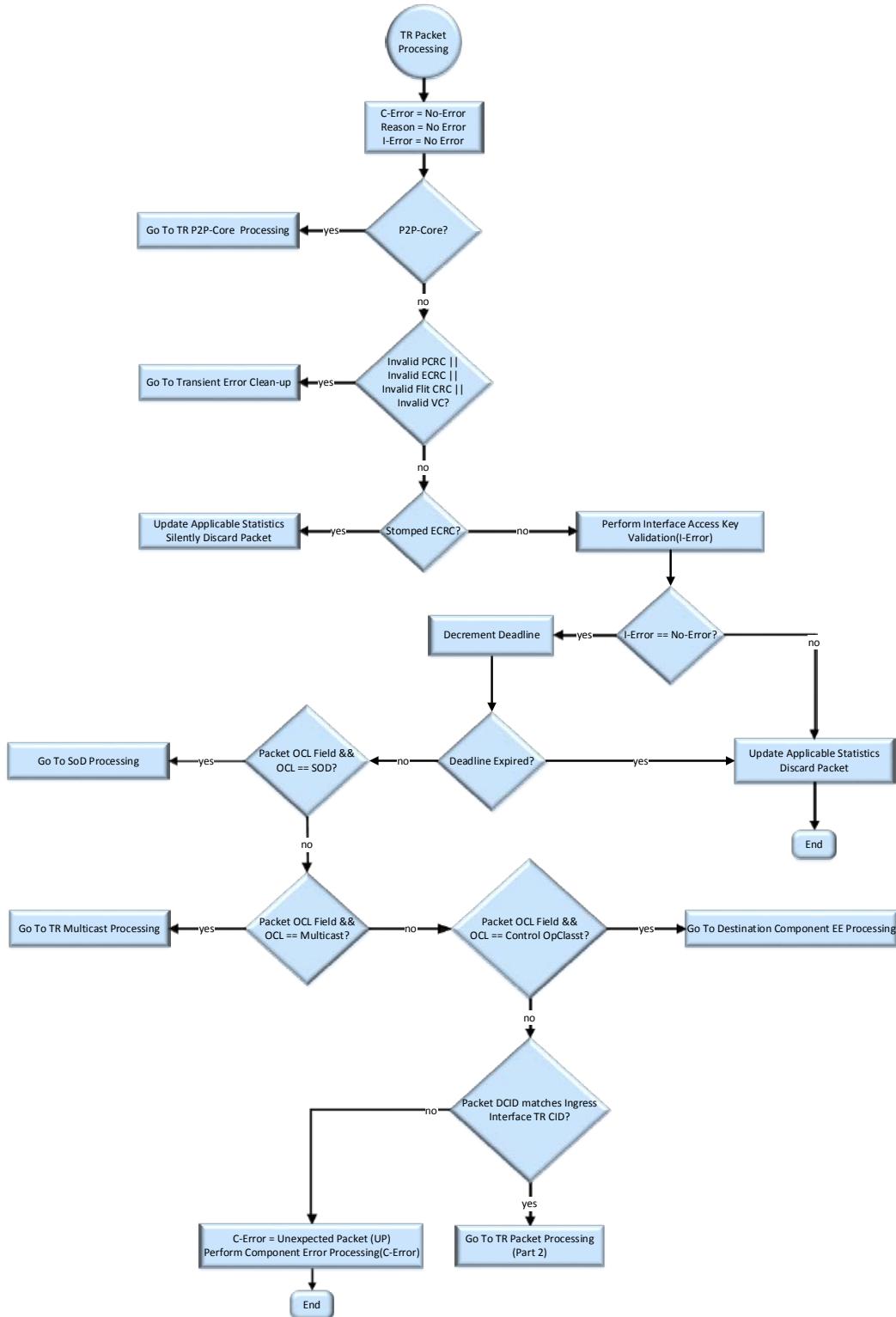


Figure 12-17: TR Unicast Packet Part 1 Validation and Processing

Table 12-12: TR Unicast Packet Part 1 Validation Processing Details

Validation Step	Explanation
<b>P2P-Core?</b>	If the component interface is enabled to support the P2P-Core OpClass, then proceed to <i>TR P2P-Core Processing</i> .
<b>Invalid PCRC    Invalid CRC   Invalid Flit CRC    Invalid VC?</b>	If the packet's PCRC and ECRC fields do not match the dynamically-generated PCRC and ECRC values, or the flit's Flit CRC does not match the dynamically-generated Flit CRC, or the packet's VC is invalid (packet's VC field does not contain a VC that is enabled on the receiving interface), then the packet has suffered a transient error; proceed to <i>Transient Error Clean-Up</i> .
<b>Stomped ECRC?</b>	If the ECRC field was stomped, then the packet is invalid and is silently discarded. If not, then the packet proceeds to the next processing step.
<b>Interface Access Key Validation: I-Error == No Error?</b>	Validate that the packet's Access Key matches an Access Key configured on the transmitting interface. If an error is detected, then update applicable statistics, and discard the packet. If and <i>I-Error Detect</i> Interface Access Key Violation Detect == 1b, then take configured error handling actions.
<b>Decrement Deadline: Deadline Expired?</b>	Decrement Deadline (see <i>Deadline Semantics</i> ), and if the packet's Deadline is not valid, then discard the packet and update applicable statistics.
<b>Packet OCL Field &amp;&amp; OCL == SOD?</b>	If the packet contains an OCL field and the OCL field equals the SOD OpClass, then proceed to <i>SOD OpClass Packet Processing</i>
<b>Packet OCL Field &amp;&amp; OCL == Multicast?</b>	If the packet contains an OCL field and the OCL field equals the Multicast OpClass, then proceed to <i>TR Multicast Processing</i> .
<b>Packet OCL Field &amp;&amp; OCL == Control OpClass?</b>	If the packet contains an OCL field and the OCL field equals the Control OpClass, then proceed to <i>Destination Component Packet Processing</i> .
<b>Packet DCID matches Ingress Interface TR CID?</b>	If the packet does not match any CID configured for subnet that the packet was received, proceed to <i>Component Error Processing (UP)</i> . The DCID in request packets is validated against the ingress interface's <i>Interface Structure</i> TR CID field.
<b>Valid Destination GCID?</b>	If GC == 1b and the packet's DSID does not match the configured component SID, then proceed to <i>Component Error Processing (UP)</i> .

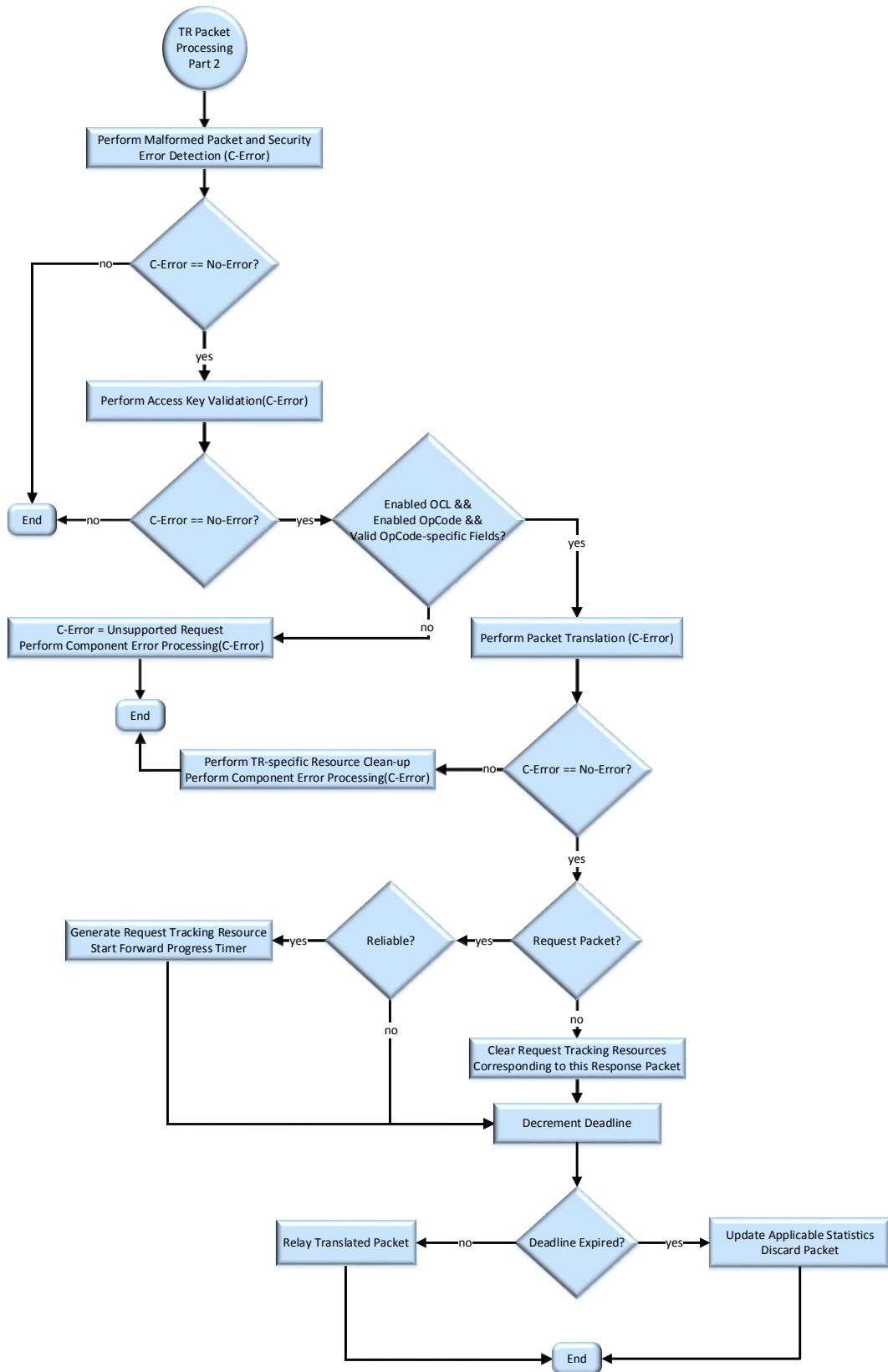


Figure 12-18: TR End-to-end Unicast Packet Part 2 Validation

Table 12-13: TR End-to-end Unicast Packet Part 2 Validation Details

Validation Step	Explanation
<b>Perform Malformed Packet and Security Error Detection(C-Error)</b>	The component executes the steps specified in <i>MP and Security Error Detection</i> .
<b>C-Error == No Error?</b>	If an error was detected, then stop further processing as the error was handled during <i>MP and Security Error Detection</i> .
<b>Perform Access Key Validation</b>	The component executes the steps specified in <i>Access Key Validation</i> .
<b>C-Error == No Error?</b>	If an error was detected, then stop further processing as the error was handled during <i>Access Key Validation</i> .
<b>Enabled OCL &amp;&amp; Enabled OpCode &amp;&amp; Valid OpCode-specific Fields?</b>	Validate the packet's OpClass (implicit or explicit) and associated OpCode are enabled, and the OpCode-specific fields are valid. If invalid, then proceed to <i>Component Error Processing</i> (UR).
<b>Perform Packet Translation (C-Error)</b>	The TR translates the packet to a packet that can traverse the destination subnet to the destination component.
<b>C-Error == No Error?</b>	If an error was detected, then perform TR-specific resource clean-up, and proceed to <i>Component Error Processing</i> (C-Error). C-Error reflects the specific issue encountered during the translation process.
<b>Request Packet?</b>	If a response packet, then clear the corresponding request tracking resources, and relay the packet to egress interface attached to the destination subnet.
<b>Request: Reliable?</b>	If the request packet requires Reliable Delivery, then generate request tracking resources and start a forward progress timer.
<b>Decrement Deadline: Deadline Expired?</b>	Decrement Deadline (see <i>Deadline Semantics</i> ), and if the packet's Deadline is not valid, then discard the packet and update applicable statistics, else relay the translated packet to the egress interface attached to the destination subnet.

### 12.9.6.1. TR P2P-Core Processing

A TR shall perform *TR P2P-Core OpClass Packet Processing* as illustrated.

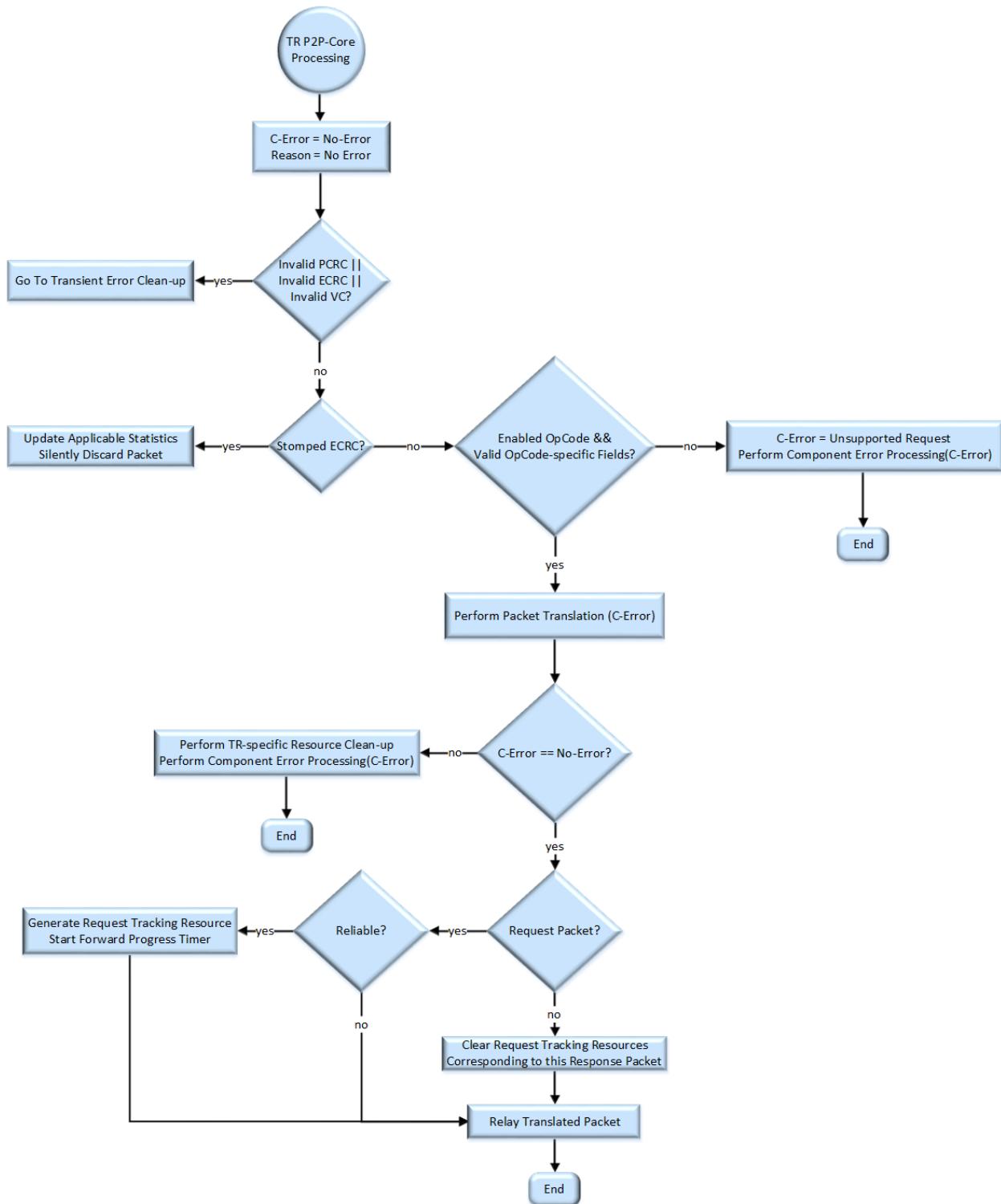


Figure 12-19: TR P2P-Core OpClass Packet Processing

Table 12-14: TR P2P-Core OpClass Packet Processing Details

Validation Step	Explanation
<b>Invalid PCRC    Invalid ECRC?</b>	Transient error detection. If true, then proceed to <i>Transient Error Clean-Up</i> .
<b>Stomped ECRC?</b>	If the ECRC field was stomped, then the packet is invalid and is silently discarded. If not, then the packet proceeds to the next processing step.
<b>Enabled OpCode &amp;&amp; Valid OpCode-specific Fields?</b>	Validate the OpCode is enabled, and the OpCode-specific fields are valid. If invalid, then proceed to <i>Component Error Processing (UR)</i> .
<b>Perform Packet Translation (C-Error)</b>	The TR translates the packet to a packet that can traverse the destination subnet to the destination component.
<b>C-Error == No Error?</b>	If an error was detected, then perform TR-specific resource clean-up, and proceed to <i>Component Error Processing (C-Error)</i> . C-Error reflects the specific issue encountered during the translation process.
<b>Request Packet?</b>	If a response packet, then clear the corresponding request tracking resources, and relay the packet to egress interface attached to the destination subnet.
<b>Request: Reliable?</b>	If the request packet requires Reliable Delivery, then generate request tracking resources and start a forward progress timer. Relay the translated packet to the egress interface attached to the destination subnet.

### 12.9.6.2. *TR Multicast Processing*

A TR shall perform *TR Multicast Processing* as illustrated.

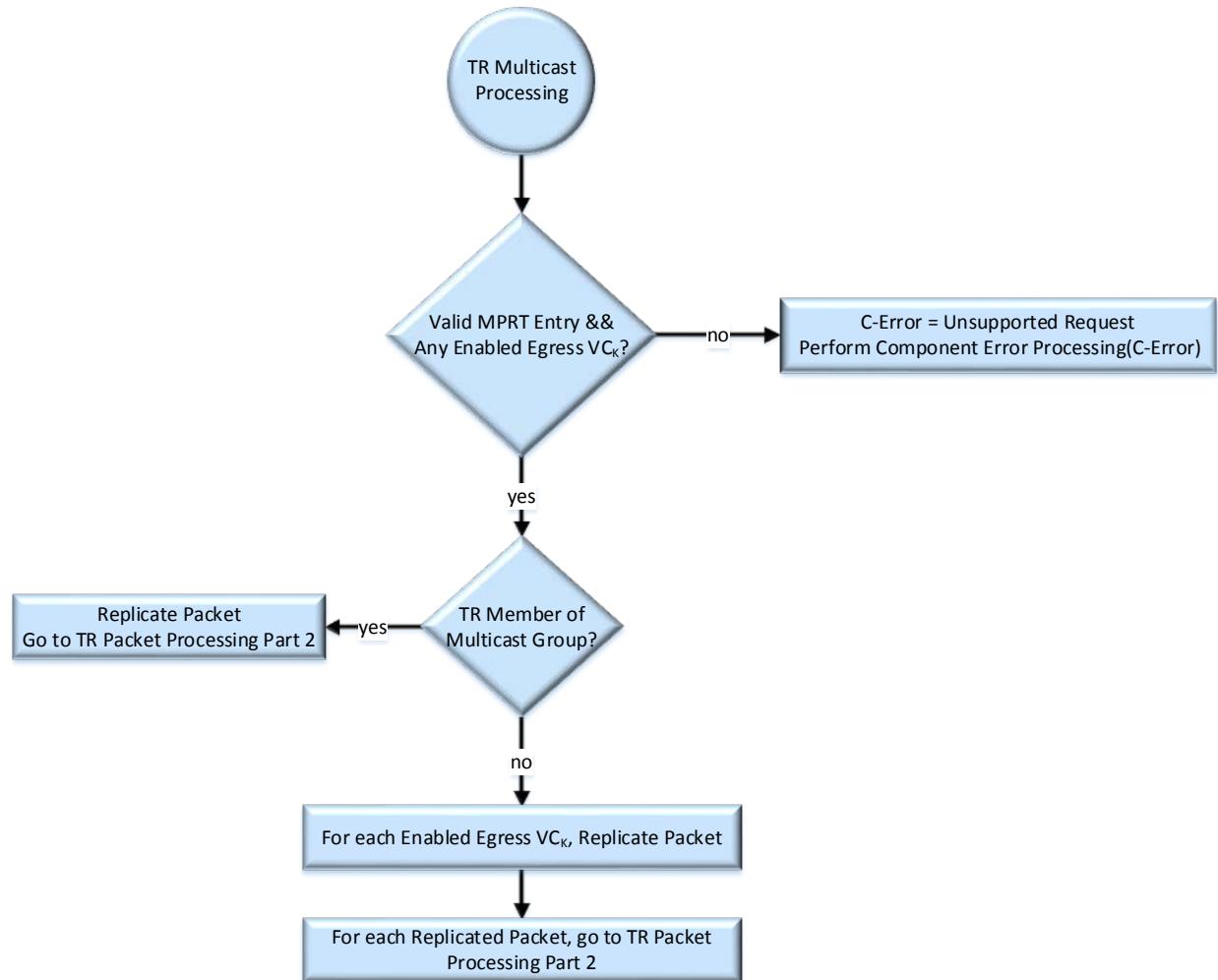


Figure 12-20: TR Multicast Processing

Table 12-15: TR Multicast Processing Details

Validation Step	Explanation
<b>Valid MPRT Entry &amp;&amp; Any Enabled Egress VCK?</b>	If the packet's MGID field does not match an enabled Multicast Packet Relay Table or if the packet's VC field does not match VC of any of the egress interfaces associated with this multicast group, then proceed to <i>Component Error Processing (UP)</i> .
<b>TR Member of Multicast Group?</b>	If the TR is a member of the multicast group, then replicate the packet and proceed to <i>Destination Component Packet Processing</i> .
<b>For each enabled Egress VCK, Replicate Packet</b>	For each egress interface associated with the multicast group and with a valid VC, replicate the packet, and proceed to TR Packet Processing Part 2 Validation.

## 12.9.7. MP and Security Error Detection

A component shall perform *MP and Security Error Detection* as illustrated.

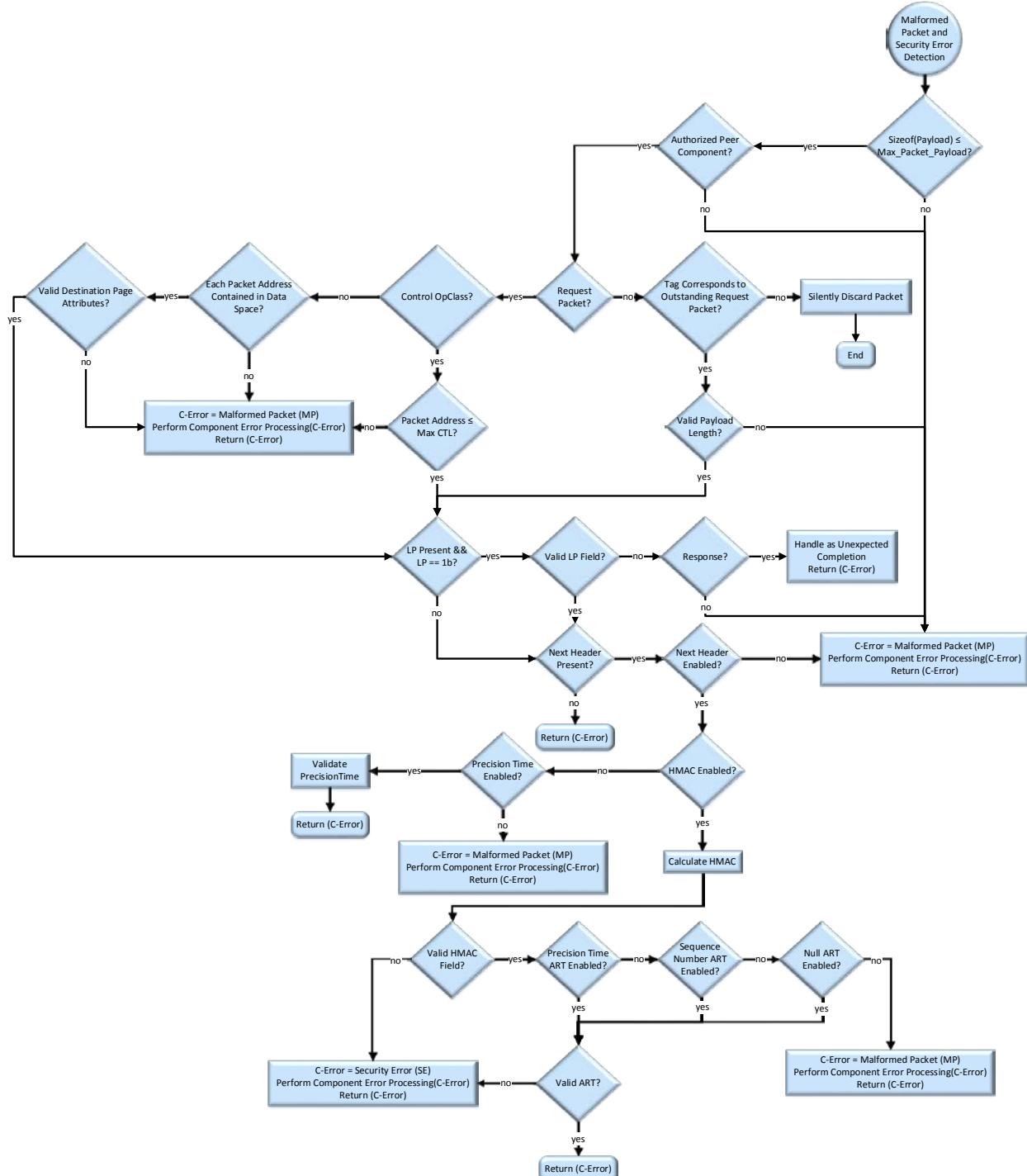


Figure 12-21: MP and Security Error Detection

Table 12-16: MP and Security Error Detection Details

Validation Step	Explanation
<b>Sizeof(Payload) ≤ Max_Packet_Payload?</b>	If payload is present and the size is greater than Max_Packet_Payload, then proceed to <i>Component Error Processing</i> (MP).
<b>Authorized Peer Component?</b>	<p>If <i>Component Destination Table Structure</i> Peer Authorization Enable == 0b, then the peer component is authorized.</p> <p>If <i>Component Destination Table Structure</i> Peer Authorization Enable == 1b, then to be authorized, the following shall be true:</p> <ul style="list-style-type: none"> <li>• The packet's SCID and if applicable, the packet's SSID shall match a configured SSDT / MSDT table entry within the <i>Component Destination Table Structure</i>.</li> <li>• If a component supports the <i>Component PA (Peer Attribute) Structure</i>, then the ACREQ / Wildcard ACREQ or ACRSP / Wildcard ACRSP fields shall be validated. If the component does not support the <i>Component PA (Peer Attribute) Structure</i>, then the component shall validate access using a mechanism outside of this specification's scope.</li> </ul> <p>If unauthorized, then proceed to <i>Component Error Processing</i> (MP).</p>
<b>Request Packet?</b>	If this is a request packet, then continue with request packet processing, else continue with response packet processing.
<b>Request: OCL == Control OpClass?</b>	If the packet contains an OCL field and that OCL equals the Control OpClass, then validate the Address field relative to Control Space.
<b>Request: Each Packet Address field Contained in Data Space?</b>	<p>The packet is not a Control OpClass packet. Verify that each packet Address field and each Address plus associated operation length is contained within the component's Data Space, e.g., if zero-based addressing is used (<i>Core Structure Component CAP 1 Address Field Interpretation</i> == 0b), then compare each Address field and Address plus length relative to the <i>Core Structure Max Data</i>.</p> <p>If any are not contained in the component's Data Space, then proceed to <i>Component Error Processing</i> (MP).</p>
<b>Request: Valid Destination Page Attributes?</b>	Verify that the destination page attributes (see <i>Responder PTE Page Attributes</i> ) support the requested operation, e.g., if a Write Persistence or a Persistent Flush, then Persistence is enabled. If a coherency request packet, then in addition to validating that the page permits coherency operations, validate that the Responder cache agent associated with the

Validation Step	Explanation
<b>Request Control OpClass: Packet Address <math>\leq</math> Max CTL</b>	<p>page is permitted to execute the operation on the destination page.</p> <p>Verify that packet's Address field and Address plus associated operation length is contained within the component's Control Space, e.g., if zero-based addressing is used (<i>Core Structure Component CAP 1 Address Field Interpretation == 0b</i>), then compare each Address field and Address plus length relative to the <i>Core Structure Max CTL</i>.</p> <p>If any are not contained in the component's Control Space, then proceed to <i>Component Error Processing</i> (MP).</p>
<b>Response: Tag Corresponds to Outstanding Request Packet?</b>	<p>If the packet's Tag field (a SOD response packet contains an Rx Seq in place of a Tag field) does not correspond to any Outstanding Request Packet, then silently discard the packet and stop packet processing.</p>
<b>Response: Valid Payload Length?</b>	<p>If the payload length does not match the expected response payload size, then proceed to <i>Component Error Processing</i> (MP).</p> <p>For example, if the packet is a Read Response for a Read 64 request packet, then a 64-byte payload is expected.</p>
<b>LP Present <math>\&amp;\amp;</math> LP == 1b?</b>	<p>Validate if the <i>LPD Field</i> is present else proceed to Next Header Present check.</p>
<b>Valid LPD Field?</b>	<p>If LPD fields (e.g., LPD BDF matches a valid PCI BDF value) and sub-fields are valid, then proceed to Next Header Present check.</p>
<b>Invalid LPD Field: Response?</b>	<p>If a response packet, then handle as an Unexpected Completion as specified in the <i>PCI Express Base Specification</i> and return C-Error.</p>
<b>Next Header Present?</b>	<p>If the <i>Explicit OpClass Next Header Field</i> is not present, then return.</p>
<b>Next Header Enabled?</b>	<p>If the <i>Explicit OpClass Next Header Field</i> is present, and the component is not enabled to process Next Headers (<i>Core Structure Component CAP 1 Control Next Header Enable == 0b</i>), then proceed to <i>Component Error Processing</i> (MP).</p>
<b>HMAC Enabled?</b>	<p>Is the component enabled to process HMAC?</p>
<b>No HMAC: Precision Time Enabled?</b>	<p>If the component is not enabled to process Precision Time, then proceed to <i>Component Error Processing</i> (MP).</p> <p>Validate the precision time portion of the <i>Explicit OpClass Next Header Field</i>, and return.</p>

Validation Step	Explanation
<b>HMAC: Calculate HMAC</b>	As the packet was received, a HMAC was dynamically generated.
<b>HMAC: Valid HMAC Field?</b>	If the dynamically-generated HMAC does not match the packet's HMAC field, then proceed to <i>Component Error Processing</i> (SE).
<b>HMAC: Precision Time ART Enabled?</b>	If the component is not enabled to use a precision time anti-replay tag, then proceed to next ART classification.
<b>HMAC Sequence Number ART Enabled?</b>	If the component is not enabled to use a sequence number anti-replay tag, then proceed to next ART classification.
<b>HMAC Null ART Enabled?</b>	If the component is not enabled to use a Null anti-replay tag, then proceed to proceed to <i>Component Error Processing</i> (MP).
<b>HMAC: Valid ART?</b>	If an invalid ART, then proceed to <i>Component Error Processing</i> (SE), else return.

## 12.9.8. Access Key Validation

A component shall perform *Component Access Key Validation* as illustrated.

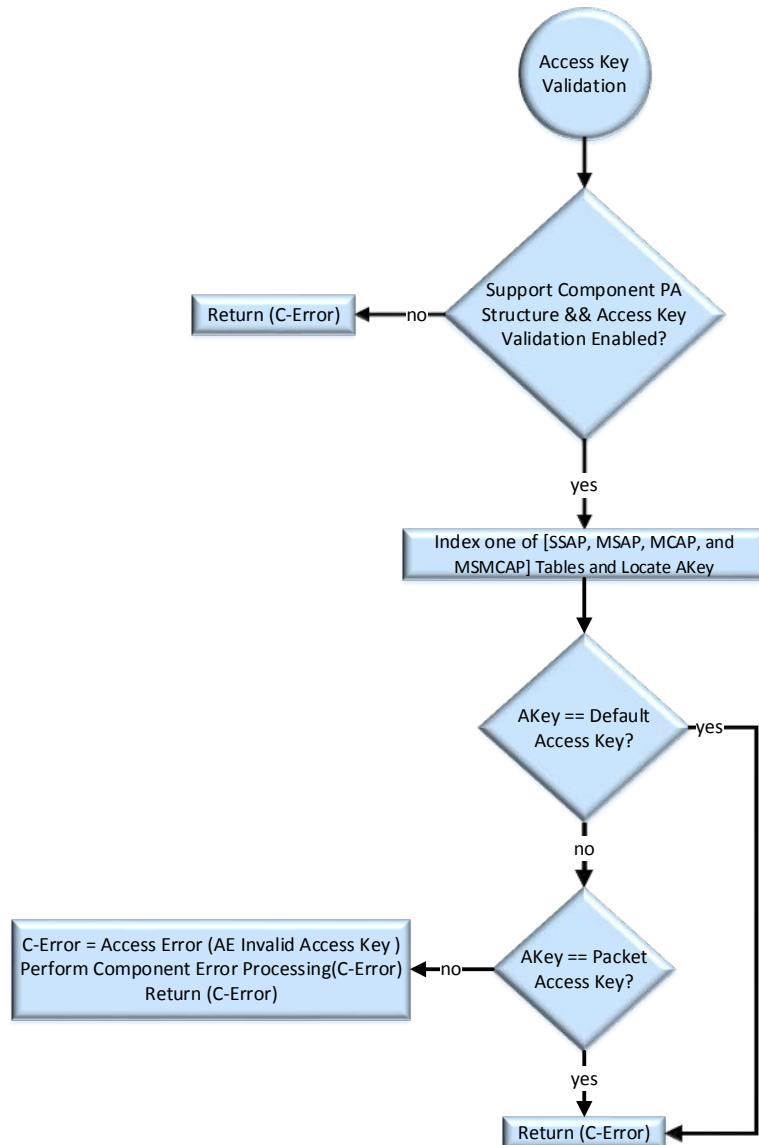


Figure 12-22: Component Access Key Validation

Table 12-17: Component Access Key Validation Details

Validation Step	Explanation
<b>Support Component PA Structure and Access Key Validation Enabled?</b>	If the component does not support the <i>Component PA (Peer Attribute) Structure</i> or Access Key validation is disabled, then the packet is permitted.
<b>Index one of [SSAP, MSAP, MCAP, and MSMCAP] Tables and Locate AKey</b>	If the packet is a unicast packet, then use either the SSAP (single subnet) or the MSAP (multi-subnet) table to locate the AKey. If the packet is a multicast packet, then use either the MCAP (single subnet) or the MSMCAP (multi-subnet) table to locate the AKey.

Validation Step	Explanation
<b>AKey == Default Access Key?</b>	Verify if the AKey equals the Default Access Key. If true, then the packet is permitted. If false, then proceed to <i>Component Error Processing</i> (AE Invalid Access Key).
<b>AKey == Packet Access Key?</b>	Verify if the AKey equals the packet's Access Key. If true, then the packet is permitted. If false, then proceed to <i>Component Error Processing</i> (AE Invalid Access Key).

An interface shall perform *Interface Access Key Validation* as illustrated.

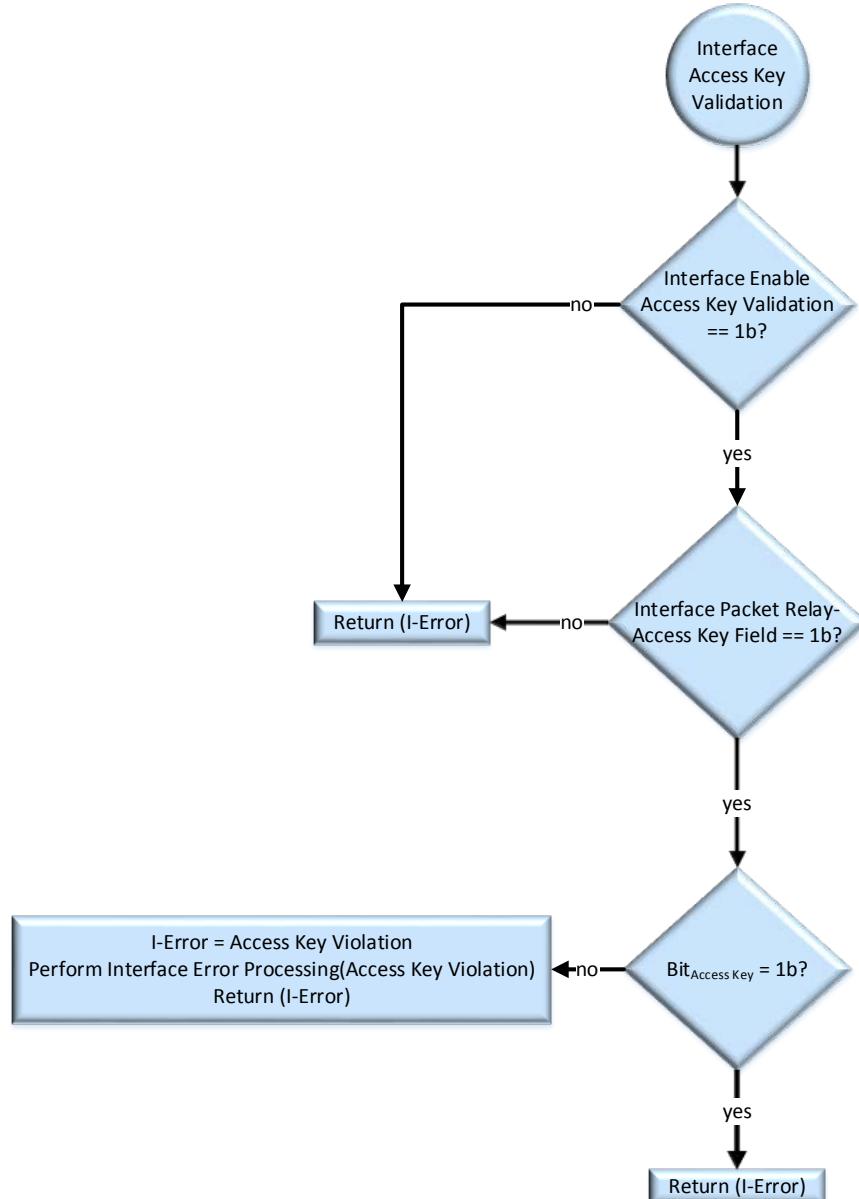


Figure 12-23: Interface Access Key Validation

Table 12-18: Interface Access Key Validation Details

Validation Step	Explanation
<b>Interface Access Key Validation</b> <b>Enable == 1b?</b>	If disabled, then return no error.
<b>Interface Packet Relay-Access Key Field == 1b?</b>	Are the Interface Ingress Access Key Mask and Interface Egress Access Key Mask fields provisioned? If not, then return no error.
<b>Bit<sub>Access Key</sub> == 1b?</b>	Using the packet's Access Key value as an index into the appropriate Access Key Mask field, verify if it set to 1b. If yes, then return no error, if not, then initiate interface error processing.

## 12.9.9. Access Permission Validation

A component shall perform *Access Permission Validation* as illustrated. If a proprietary page structure is used, then the semantics used to perform access permission validation shall be adhered to independent of which structure is used.

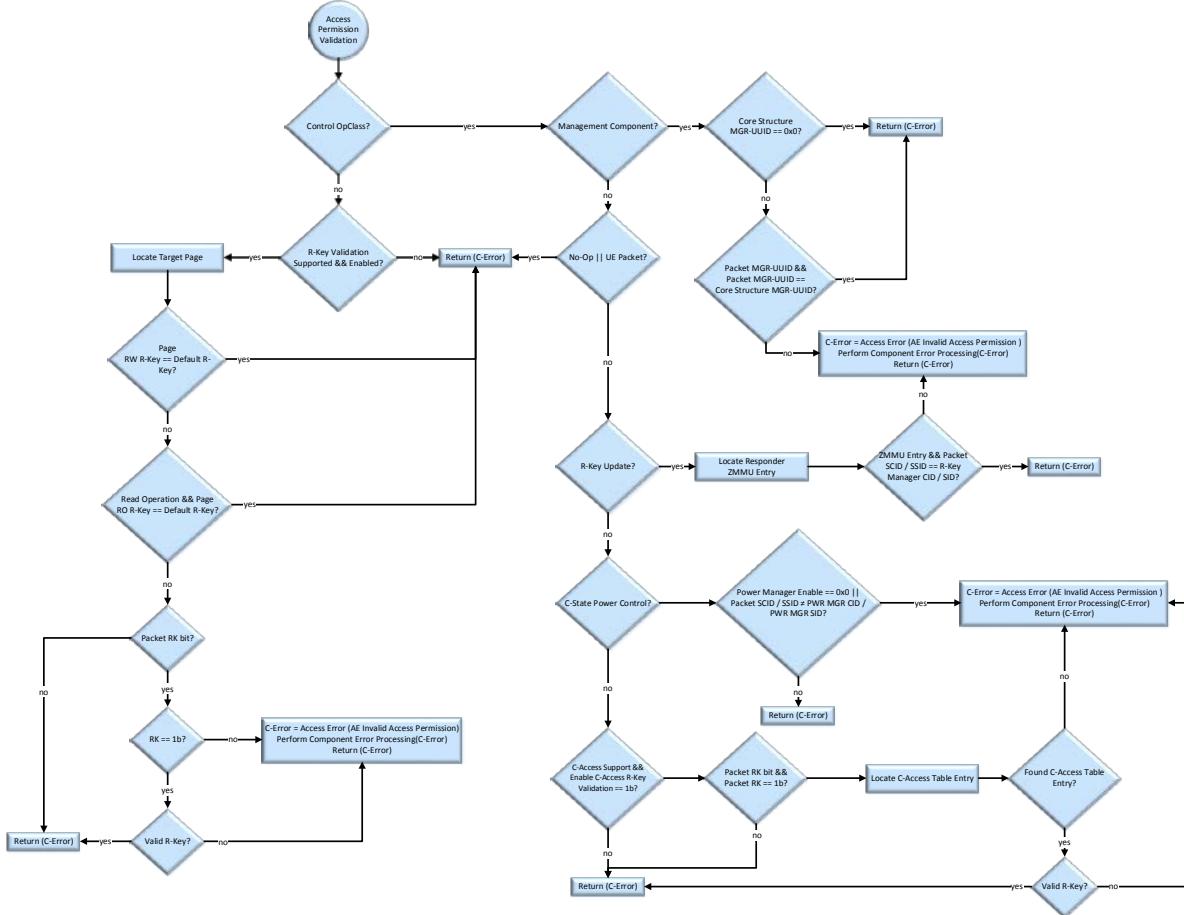


Figure 12-24: Access Permission Validation

Table 12-19: Access Permission Validation Details

Validation Step	Explanation
<b>Control OpClass?</b>	<p>Is this is a Control OpClass request?</p>
<b>Control OpClass: Management Component?</b>	<p>If <i>Core Structure Component CAP 1 Control Manager Type</i> == 0b and the packet's SCID does not match the <i>Core Structure</i> PMCID, then proceed to <i>Component Error Processing</i> (AE Invalid Access Permission).</p> <ul style="list-style-type: none"> <li>• If packet's SCID matches or the <i>Core Structure</i> PMCID is not configured, then the packet's SCID shall be handled as a valid management component CID.</li> <li>• If the <i>Core Structure</i> PMCID is not configured and the packet is a Control Write request packet, then the packet's SCID shall be copied to the <i>Core Structure</i> PMCID.</li> </ul> <p>If <i>Core Structure Component CAP 1 Control Manager Type</i> == 1b and the packet's SCID does not match the <i>Core Structure</i> PFMCID or SFMCID, and if packet GC == 1b and the packet's SSID does not match the <i>Core Structure</i> PFMSID or SFMSID, then proceed to <i>Component Error Processing</i> (AE Invalid Access Permission).</p> <ul style="list-style-type: none"> <li>• If the packet's SCID matches or the <i>Core Structure</i> PFMCID and SFMCID are not configured, then the packet's SCID shall be handled as a valid management component CID.</li> <li>• If GC == 1b and the packet's SSID matches or the PFMCID, SFMCID, PFMSID, and SFMSID are not configured, then the SSID shall be handled as a valid management component SID.</li> <li>• If the <i>Core Structure</i> PFMCID is not configured and the packet is a Control Write request packet, then the packet's SCID shall be copied to the <i>Core Structure</i> PFMCID.</li> <li>• If GC == 1b and the <i>Core Structure</i> PFMCID and PFMSID are not configured and the packet is a Control Write request packet, then the packet's SCID shall be copied to the <i>Core Structure</i> PFMCID and the packet's SSID shall be copied to the <i>Core Structure</i> PFMSID.</li> </ul>
<b>Management Component: Core Structure MGR-UUID == 0x0?</b>	<p>If the <i>Core Structure</i> MGR-UUID is zero, then no additional access permission checks are required.</p> <p>If non-zero and the packet contains a MGR-UUID field, then compare that to the <i>Core Structure</i> MGR-UUID. If they match, then access permission is granted; if they do not match, then</p>

Validation Step	Explanation
<b>Control OpClass: No-Op    UE packet?</b>	<p>proceed to <i>Component Error Processing</i> (AE Invalid Access Permission).</p>
<b>Control OpClass: R-Key Update?</b>	<p>If a <i>No-Op</i> or an <i>Unsolicited Event (UE) Packet</i>, then access permission is granted.</p>
<b>Control OpClass: C-State Power Control?</b>	<p>If an <i>R-Key Update</i> packet, then locate the Responder ZMMU PTE and compare the packet's SCID / SSID with the PTE's R-Key Manager CID / SID. If they match, then access permission is granted; if they do not match, then proceed to <i>Component Error Processing</i> (AE Invalid Access Permission).</p>
<b>Control OpClass: C-Access Support &amp;&amp; Enable C-Access R-Key Validation == 1b?</b>	<p>If a C-State Power Control packet, then validate if Power Manager Enable is 1b and that the Packet's SCID / SSID match the <i>Core Structure</i> PWR MGR CID / PWR MGR SID. If they match, then access permission is granted; if they do not match, then proceed to <i>Component Error Processing</i> (AE Invalid Access Permission).</p>
<b>Non-Control OpClass: R-Key Validation Supported and Enabled?</b>	<p>If the component does not support the <i>Component C-Access Structure</i> or R-Key validation using the C-Access structure is disabled, then access permission is granted.</p> <p>If not, and the packet contains a RK bit and RK == 1b, then locate the C-Access Table entry and validate the R-Key. If the entry cannot be found or the packet's R-Key does not match the entry's appropriate R-Key, then proceed to <i>Component Error Processing</i> (AE Invalid Access Permission); else access permission is granted.</p>
<b>Page RW R-Key == Default R-Key?</b>	<p>If unsupported or disabled, then access permission is granted.</p> <p>If supported and enabled, then locate the target page.</p>
<b>Read Operation &amp;&amp; Page RO R-Key == Default R-Key?</b>	<p>If the page's RW R-Key is the Default R-Key, then access permission is granted.</p>
<b>Packet RK bit?</b>	<p>If the operation only reads the page and the page's RO R-Key is the Default R-Key, then access permission is granted.</p>
<b>RK Bit: RK == 1b?</b>	<p>If the packet does not contain an RK bit, then access permission is granted.</p> <p>If RK ≠ 0b, then proceed to <i>Component Error Processing</i> (AE Invalid Access Permission); else validate the packet's R-Key field. If a read operation, then compare the packet's R-Key with the RO R-Key. If a write / data modification operation, then compare the packet's R-key with the RW R-Key. If the R-Key does not match, then proceed to <i>Component Error Processing</i> (AE Invalid Access Permission); else access permission is granted.</p>

## 12.9.10. Transient Error Clean-Up

When a transient error is detected, the component shall perform *Transient Error Packet Clean-up Processing* as illustrated.

5

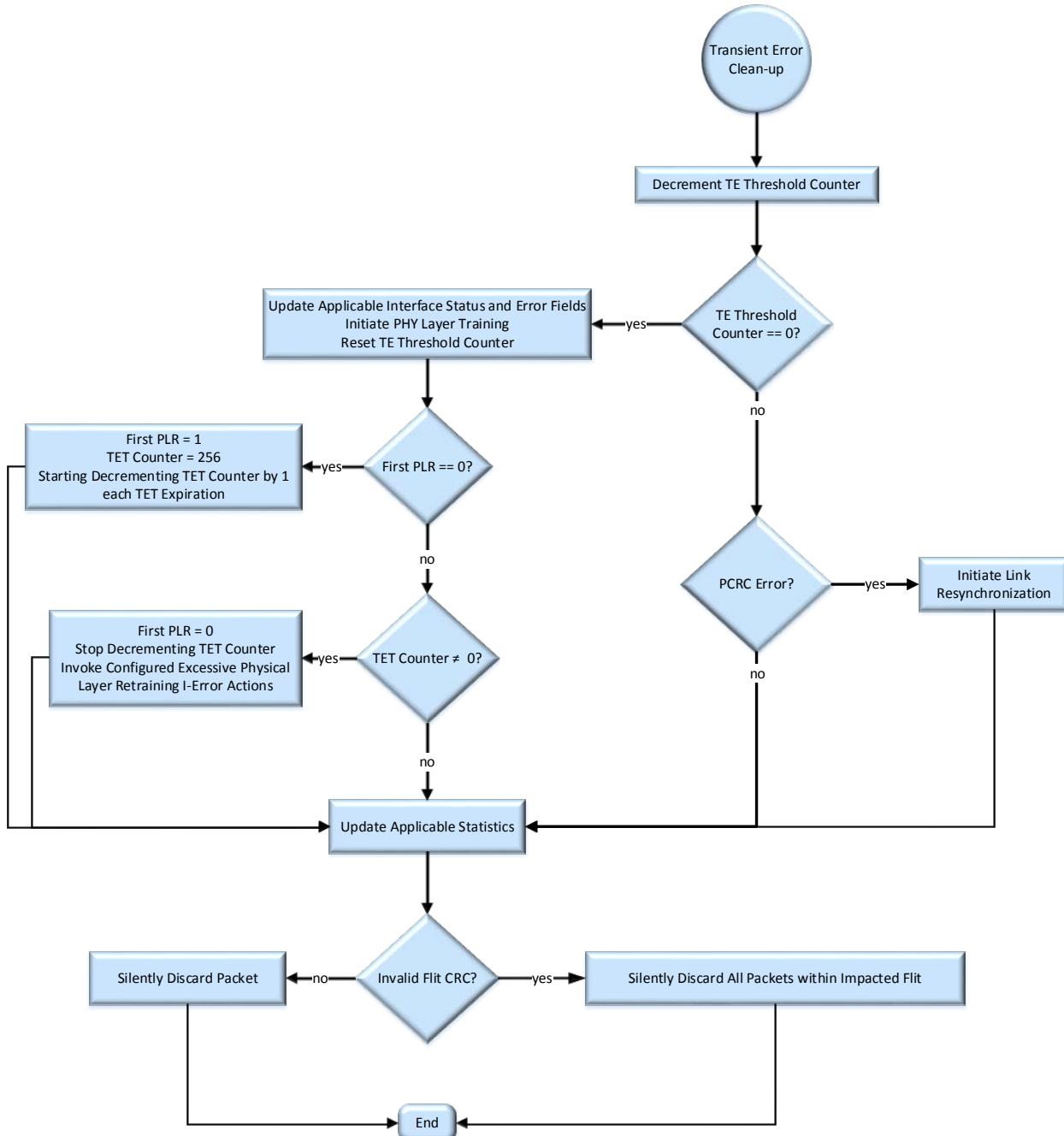


Figure 12-25: Transient Error Packet Clean-up Processing

Table 12-20: Transient Error Packet Clean-up Processing Details

Validation Step	Explanation
<b>Decrement TE Threshold Counter</b>	Decrement the Transient Error Threshold Counter. This counter is used to determine if the link has suffered too many transient errors.
<b>TE Threshold Counter == 0?</b>	<p>If the number of transient errors exceed the threshold allowed within allotted time, then:</p> <ul style="list-style-type: none"> <li>• Update applicable interface status and error fields</li> <li>• Initiate physical layer retraining</li> <li>• Reset the TE Threshold Counter to its configured value.</li> <li>• Update applicable statistics.</li> <li>• Silently discard the packet.</li> </ul>
<b>First PLR == 0?</b>	<p>If this is the first physical layer retraining event being tracked, then:</p> <ul style="list-style-type: none"> <li>• Set First PLR = 1 and initialize TET Counter</li> <li>• For each subsequent TET expiration, decrement the TET Counter variable by 1</li> </ul>
<b>First PLR 1: TET Counter ≠ 0?</b>	<p>If this was not the first physical layer retraining event, then determine if the TET Counter had reached zero prior to a new physical layer training event occurring.</p> <p>If zero and <i>I-Error Detect Excessive Physical Layer Retraining Event == 1b</i>, then take configured error handling actions.</p>
<b>PCRC Error?</b>	<p>If a PCRC error was detected, then packet synchronization has been lost. Initiate <i>Link Resynchronization</i>. Upon completion, update applicable statistics and silently discard the packet.</p>
<b>Invalid Flit CRC?</b>	<p>If an invalid Flit CRC was detected, then drop all packets within the Flit (this includes any end-to-end packet that was partially contained within the flit—trailing or beginning bits of a packet). If not an invalid Flit CRC, then drop the impacted packet.</p>

## 12.10. Error Containment

Error containment is an optional capability intended to isolate the impact of a non-transient error or event, and to reduce the probability of silent data corruption. The architecture supports component and interface containment. Each level may be triggered in response to a particular processing error or event (e.g., physical layer failure) or may be explicitly triggered by software.

5 Error containment is an optional feature managed through Control Space structures.

## 12.10.1. Component Containment

Component containment may be explicitly triggered by software or in response to any error or fault condition detected at the component level that has been configured to trigger containment.

Component containment can also be triggered if an Unexpected Physical Layer Failure—PHY-Down (see *Interface Error Fields*) triggered interface containment and *Interface I-CAP 1 Control Interface-Component Containment Enable == 1b*.

Independent of the underlying cause, when component containment is triggered, the following applies:

- If a packet protocol error triggered containment, then the packet shall not be executed.
- The actions configured in the *Component Error and Signal Event Structure* shall be taken.
- If a component contains an integrated switch or transparent router, all packets not destined for the component shall be relayed unless containment recovery precludes continued packet relay operation.
- Until component containment is cleared:
  - The Responder shall transmit a *Standalone Acknowledgment* (Reason = Component Containment Triggered) for any request packet that targets Data Space.
  - The Responder shall validate and execute any request packet that targets Control Space.
- The component shall validate and execute all packets received prior to the packet that triggered component containment, and schedule appropriate response packets for transmission on any interface that is capable of reaching the destination component.
- As a result of component containment, if any component egress interface cannot transmit packets, then all packets targeting that interface shall be silently discarded.
- Component containment shall be enforced until the Component Containment Trigger bit (see *C-Error Trigger*) is cleared or the component is reset (Full or Warm).

## 12.10.2. Interface Containment

Interface Containment is triggered due to the unexpected link state transition to L-Down (see *Link States*). Interfaces enabled for Interface Containment reporting shall treat the transition to L-Down as an error except under the following conditions:

- Software initiated the transition by initiating an Interface Reset.
- Software initiated the transition by triggering an Interface Containment event.
- Software initiated the PHY-Down transition.
- Interface Containment detection is disabled.

Once Interface Containment is detected:

- If a packet triggered containment, then it shall not be executed.
- The *Interface I-Status* shall be updated to indicate interface containment has been triggered.
- The actions configured in the *Interface Error Fields* shall be taken.
- The link shall remain in the L-Down state until interface containment is cleared. Once cleared, the link shall proceed through normal link initialization.
- Interface Containment shall be enforced until cleared or the interface is reset (Full or Warm).

## 12.11. Packet Scheduling Precedence

When multiple packets of differing types are ready to be transmitted, the component shall schedule transmission using the following precedence (highest-to-lowest precedence):

- *Link-level Reliability (LLR)* ACK packets upon LLRT expiration (if LLR supported and enabled)
- *Link-level Reliability (LLR)* Discard packets (if LLR supported and enabled)
- *Explicit Flow Control* update packets upon FCTT expiration (if supported and enabled)
- An *Explicit Flow Control* packet when the number of  $VC_k$  unadvertised flow-control credits is greater than ( $VC_k$  maximum provisioned flow-control credits DIV 3).
- End-to-end packets, selected according to the three-tier arbitration specified below.
- *Link-level Reliability (LLR)* ACK packets prior to LLRT expiration (if LLR supported and enabled), and *Explicit Flow Control* update packets prior to FCTT expiration (if supported and enabled). These have equal precedence (if both packet types are ready for transmission, then round-robin arbitration should be used).

End-to-end packets shall be selected for transmission on the basis of a three-tier arbitration:

- A winning VC is selected per *Virtual Channels*, from the set of VCs for which sufficient flow-control credits are available end-to-end packets are awaiting transmission, considering all sources.
- Within the winning VC, a winning packet source is selected based upon starvation-free arbitration amongst those sources with packets awaiting transmission. A source may consist of a component-internal requester or responder, or may be an internal transmission queue.
- If the winning source is a component-internal requester, then using starvation-free arbitration, the component shall select between new request packets, retransmitted request packets, and *Non-Idempotent Request Release (NIRR)* packets. *Non-Idempotent Request Release (NIRR)* packets should be given highest priority. New request packets shall not starve out retransmitted request packets, nor vice versa.

Once packet transmission has begun it shall not be preempted by another packet. Transmission shall be stopped only due to a condition that causes the link to transition to L-Down.

## 12.12. Reset Functionality

Operating and / or error conditions may require a component to be returned to a known state. As specified in Component Management, components always power-up in a known state with all of the resources in their default state. However, returning a single component to its power-on default state while the rest of the solution remains operational can disrupt normal solution operation. There are two reset classes—component and interface. Each reset class is further subdivided into one or more reset types.

Irrespective of the reset class,

- Non-volatile resources shall be unaffected by a reset. To reset a non-volatile resource, software shall be responsible for overwriting the non-volatile resource with a resource-specific value or pattern, or a mechanism outside this specification's scope shall be used.
- If the component transitions to the C-Down or the C-CFG state and the component supports *In-band Management*, then the component shall execute and schedule only *In-band Management* read and write request packets until it transitions to the C-Up state.

- Management shall not attempt to communicate with the component until sufficient time has elapsed to ensure all packets targeting the component have been drained from the topology. The minimum time delay shall be the maximum calculated retransmission timer used by all Requesters communicating with the reset component.

5      **12.12.1. Component Reset**

The following are the features and requirements associated with Component Reset:

- Unless explicitly stated otherwise by this specification, component reset may be initiated:
  - Through an out-of-band mechanism, e.g., a mechanical form-factor-specific power-down event.
  - Through *Core Structure* controls, i.e., performing an in-band or out-of-band write to the *Core C-Control Component Reset* bit.
  - As part of component containment recovery
  - As a result of the expiration of the Configuration Completion timer (see *Configuration Completion Timer and Release*)
  - Power removal or disablement
- There are four types of component reset—a *Full Component Reset* to return a component to its power-on initialization state, a *Warm Component Reset* to return a component to its power-on initialization state with the exception of fields classified as sticky (e.g., error bits), a *Warm Non-Switch Component Reset* which is identical to a *Warm Component Reset* except it does not impact the component’s integrated switch logic, and a content component reset that resets internal component-specific content but not configuration structures to the power-on initialization state.

**12.12.1.1. Full Component Reset**

A component shall support Full Component Reset.

- A Full Component Reset shall occur after a power-up event.
- When a Full Component Reset occurs, the following actions shall be initiated:
  - If previously triggered, *Component Containment* shall be cleared.
  - If previously triggered, *Component Media Structure Primary Media Status* and *Secondary Media Status Fatal Media Error Containment Triggered* shall be cleared.
  - For each interface, a *Full Interface Reset* shall be initiated.
  - All component Control Space structures and component-internal resources (registers, timers, tables, state structures, etc.) shall be reset to the uninitialized state, i.e., all Control Space structures and implementation-specific resources shall be returned to their initial power-up contents. With the exception of hardware-initialized fields, all non-RO Control Space structure fields and volatile media shall be set to zero.
  - All volatile media may be reset to the uninitialized state based on policies outside of this specification’s scope. Poisoned media locations shall be cleared.
  - All non-volatile media shall be unaffected.
    - If a component does not use non-volatile media for persistence, then the non-volatile media shall be reset to the uninitialized state. Poisoned media locations shall be cleared.
  - Additional actions are specified throughout this specification.
- The component shall transition to the C-Down state.

### 12.12.1.2. Warm Component Reset

A component shall support Warm Component Reset.

- A Warm Component Reset shall occur only if power remains uninterrupted and a Warm Component Reset event is initiated.
  - Unless explicitly stated otherwise by this specification, if the component contains an integrated switch, then management may initiate a *Warm Non-Switch Component Reset* instead of a Warm Component Reset.
- When a Warm Component Reset occurs, the following actions shall be initiated:
  - If previously triggered, component containment is cleared.
  - For each interface, a *Warm Interface Reset* shall be initiated.
  - All component Control Space structures and component-internal resources (registers, timers, tables, state structures, etc.) shall be reset to the uninitialized state with the exception of fields identified as sticky.
  - A Sticky field shall retain its state across a warm reset. A subset of control fields is designated as Sticky to facilitate initialization and error identification and recovery.
  - All volatile media and non-volatile media shall be unaffected.
  - Additional actions are specified throughout this specification.
- The component shall transition to the C-Down state.

### 12.12.1.3. Warm Non-Switch Component Reset

If a component supports an integrated switch, then it shall support Component Warm Non-Switch Reset.

- A Warm Non-switch Component Reset shall occur only if power remains uninterrupted and a Warm Non-Switch Component Reset event is initiated.
- When a Warm Non-Switch Component Reset occurs, the following actions shall be initiated:
  - If previously triggered, component containment is cleared.
  - All interfaces used by the integrated switch and their corresponding control structures shall not be impacted by the reset event.
  - All packet relay control structures, the *Component Switch Structure*, and the *Component Multicast Structure* shall not be impacted by the reset event. All other component control structures and component-internal resources (registers, timers, tables, state structures, etc.) shall be reset to the uninitialized state with the exception of fields identified as sticky.
  - A Sticky field shall retain its state across a warm reset. A subset of control fields is designated as Sticky to facilitate initialization and error identification and recovery.
  - All volatile media and non-volatile media shall be unaffected.
- With the exception of the integrated switch functionality and associated switch interfaces, the rest of the component shall transition to the C-Down state.

### 12.12.1.4. Content Component Reset

A component may support Content Component Reset.

- A Content Component Reset shall be initiated through the *Core Structure*.
- When a Content Component Reset occurs, the following actions shall be initiated:

- All component Control Space structures and component-internal resources shall be unaffected.
- The component shall reset component-specific state, structures, and volatile memory / resources to their power-on initialization state. The specific methods used by a component to perform this reset is outside of this specification's scope.

5

## 12.12.2. Interface Reset

The following are the features and requirements associated with Interface Reset:

- An interface may be reset:
  - Through an out-of-band mechanism
  - Through *Interface Structure* controls
  - As a result of a *Full Component Reset* or a *Warm Component Reset*
- Resetting a given component interface shall not impact other component interfaces with the exception that a reset disables packet forwarding or relay to / from this interface until the interface transitions to the Interface Up (I-Up) state. Resetting shall not impact unrelated component operations and services.
- There are two types of interface reset—a Full Interface Reset is used to return the interface to its power-up initialization state, and a Warm Interface Reset is used to return the interface to its power-up initialization state with the exception of sticky configuration fields.

### 12.12.2.1. Full Interface Reset

An interface shall support Full Interface Reset. When initiated, the following actions shall be taken:

- If previously triggered, interface containment shall be cleared.
- The associated link shall be reset (see *Link States* for the specific steps).
- The *Interface Structure*, the *Interface Statistics Structure*, the *Interface PHY Structure*, vendor-defined structures associated with this interface, and all associated interface-internal resources (registers, tables, tracking logic, state structures, etc.) shall be reset to the uninitialized state and all volatile media associated with this interface shall be set to zero.
- If a packet relay component, then the component shall disable this interface as a valid egress interface from all impacted packet relay tables.
- If a Requester or a Responder, then the component shall disable this interface in all tables within the *Component Destination Table Structure* that target this interface.
- Additional actions are specified throughout this specification.

### 12.12.2.2. Warm Interface Reset

An interface shall support Warm Interface Reset. When initiated, the following actions shall be taken:

- If previously triggered, interface containment shall be cleared.
- The associated link shall be reset, i.e., is transitioned to the L-Down state (see *Link States* for the specific steps). All in-flight / queued packets shall be silently discarded.
- If *Interface I-CAP 1 Control* Aggregated Interface Control == 0x0, then the physical layer shall transition to PHY-Down, and subsequently initiate physical layer training. If *Interface I-CAP 1 Control* Aggregated Interface Control ≠ 0x0, then the physical layer shall re-examine its *Interface PHY Structure*, and subsequently initiate physical layer retraining.

- 5 • The *Interface Structure*, the *Interface Statistics Structure*, vendor-defined structures associated with this interface, and all associated interface-internal resources (registers, tables, tracking logic, state structures, etc.) shall be reset to the uninitialized state except for interface configuration fields labeled as sticky; these shall be unaffected. All volatile media associated with this interface shall be set to zero.
- 10 • If a packet relay component, then the component shall disable this interface as a valid egress interface from all impacted packet relay tables.
- If a Requester or a Responder, then the component shall disable this interface in all tables within the *Component Destination Table Structure* that target this interface.
- The *Interface PHY Structure* shall not be reset to the uninitialized state.
- Additional actions are specified throughout this specification.

# 13. Physical Layer Abstraction

The *Gen-Z Core Architecture Specification* specifies a Physical Layer Abstraction (PLA). This section defines minimum required Physical Layer functions for Gen-Z and provides a common interface between the Physical Layer and the Gen-Z Core regardless of the physical transmission media (electrical or optical), signal transmission type (single-ended or differential), signal modulation (e.g., NRZ or PAM-4) and Physical Layer implementation that is selected. This section will not define the actual design of the Physical Layer; the *Gen-Z Physical Specification* specifies details of the Physical Layer implementations.

## 13.1. Required Physical Layer Features

The Gen-Z Physical Layer enables data packets to traverse between the physical transmission clock domains to the Gen-Z Core internal clock domain. At a minimum the Physical Layer shall support the features defined in the following sections.

### 13.1.1. Link Training

The Physical Layer is responsible for training the link and implementing all the implementation specific state machines for training the link. The Physical Layer shall handle generation and reception of any Training Sequences required for link training. The Physical Layer shall account for any lane-to-lane skew on the link, and handle any data scrambling and de-scrambling that is necessary.

### 13.1.2. PHY States

The Physical Layer shall support the link states defined in *Required PHY States*, and shall use the *Physical Layer State Machine*.

- The PHY may implement physical layer-specific sub-states which are defined in the PHY specifications.
- The PHY may support additional sub-states which are defined in *PHY Low-power States*.

Table 13-1: Required PHY States

State	Description
PHY-Down	<p>The PHY is down, and cannot send or receive data across the link. PHY transmitters and receivers are disabled. This state is entered when CORE_PHYRESET is asserted or when directed by the Gen-Z Core to disable the link or to initiate physical layer retraining. If physical layer retraining fails, then the PHY shall remain in PHY-Down.</p> <p>Configuration registers shall not reset when this state is entered, but PHY transmitter and receiver settings (e.g., transmitter equalization and receiver equalization) shall be cleared.</p>
PHY-Up	Physical layer training has been successfully completed, and the PHY is capable of sending and receiving data across the link. This state has the highest

State	Description
	performance and highest power consumption. The PHY can be directed to go to PHY-Down, PHY-Down-Retrain, or to one of the optional Low Power states.

### 13.1.3. State Transitions

The Gen-Z Core will direct the Physical Layer to transition between states using CORE\_STATEREQ. The Physical Layer signals its current state using the PHY\_TX\_STATE and the PHY\_RX\_STATE signals. The *Physical Layer State Machine* shall be as illustrated.

5

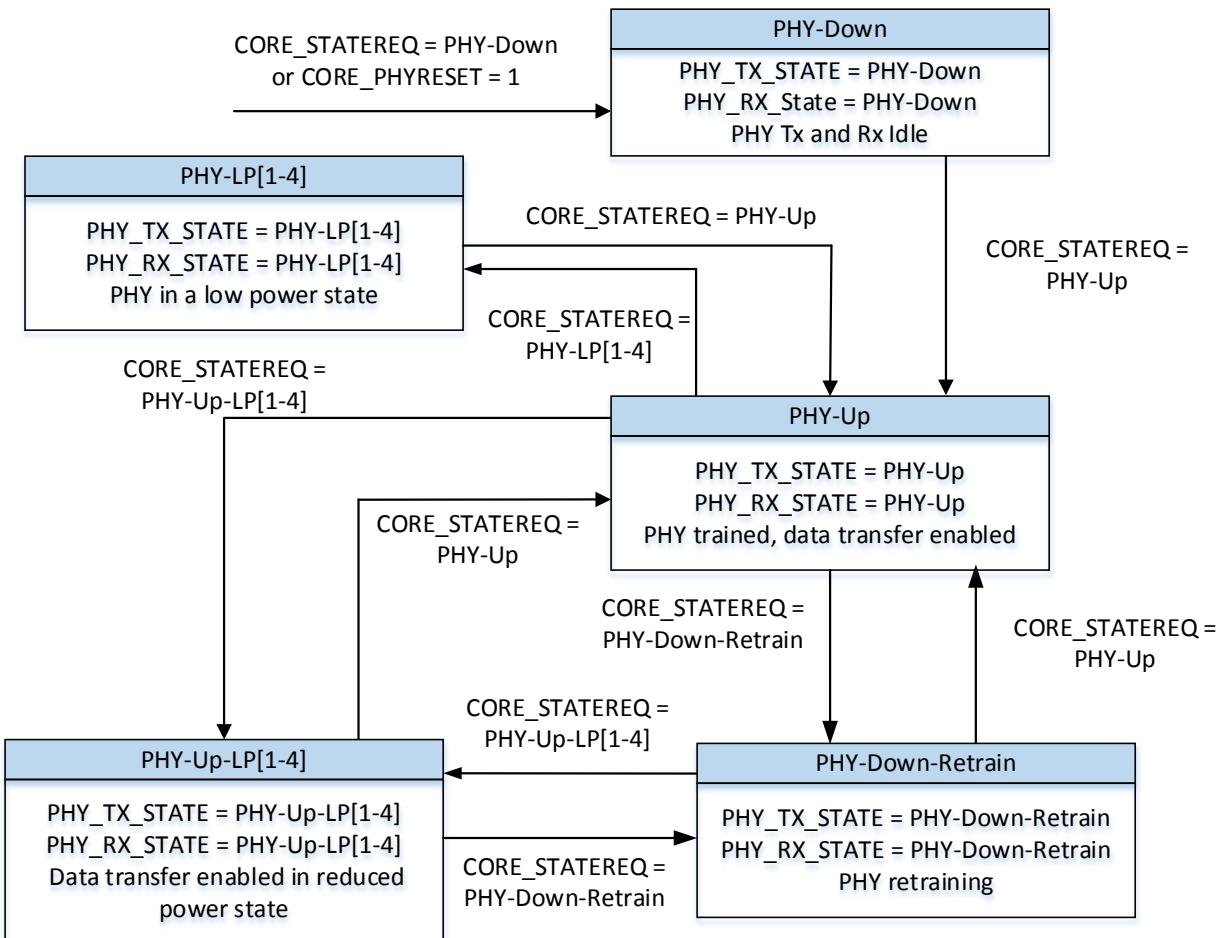


Figure 13-1: Physical Layer State Machine

### 13.1.4. Configuration Registers

The Physical Layer shall implement all Configuration Registers defined in the Physical Layer Configuration section.

10

### 13.1.5. Bandwidth Matching

The Physical Layer is responsible for matching the bandwidth of the Gen-Z Core and the link. The Physical Layer shall insert backpressure on the Gen-Z Core when the PHY needs to insert PHY specific bits for symbol encoding, or if the Gen-Z Core runs faster than the link bandwidth for link clock margin.

5 Backpressure is inserted using the PHY\_RXVALID and PHY\_TXDATAREQ signals on the PLA interface.

Bandwidth shall be matched by the Physical Layer regardless of the clocking architecture including common reference clock designs, separate independent reference clock designs, and separate independent reference clock designs with spread spectrum clocking. See the Gen-Z Physical Specification for additional clocking architecture details.

### 10 13.1.6. Data Striping

The Physical Layer shall be responsible for mapping data received from the Gen-Z Core on the PLA interface to the link, and the Physical Layer shall be responsible for mapping received data from the link to the PLA interface. The link is divided into one or more lanes. For example, lanes of an electrical link consist of wires whereas optical lanes are wavelengths on a fiber. The number of lanes on a link is PHY dependent, and the number of lanes may be different for each direction.

- A link may support 1, 2, 3, 4, 5, 6, 7, 8, 12, 16, 32, 64, 128, or 256 transmit lanes.
  - For links that support 4 or more transmit lanes, PLA data received from the Gen-Z Core shall be distributed over the lanes using byte-striping.
    - *Data Striping on a Four-lane link with a 64-bit PLA Interface* illustrates byte-striping on 4 transmit lanes. The PLA shall place byte 0 on lane 0, byte 1 on lane 1, byte 2 on lane 2, byte 3 on lane 3, byte 4 on lane 0, byte 5 on lane 1, and so forth.
  - For links that support 2 or 3 transmit lanes, PLA data received from the Gen-Z Core shall be distributed over the lanes using 4-byte striping.
    - *Data Striping on a Two-lane link with a 128-bit PLA Interface* illustrates quad-byte striping on 2 transmit lanes. The PLA shall place bytes [0-3] on lane 0, bytes [4-7] on lane 1, bytes [8-11] on lane 0, bytes [12-15] on lane 1, and so forth.
  - For links that support 1 transmit lane, PLA data received from the Gen-Z Core shall be distributed in the received byte-order on lane 0.
- A link may support 1, 2, 3, 4, 5, 6, 7, 8, 12, 16, 32, 64, 128, or 256 receive lanes.

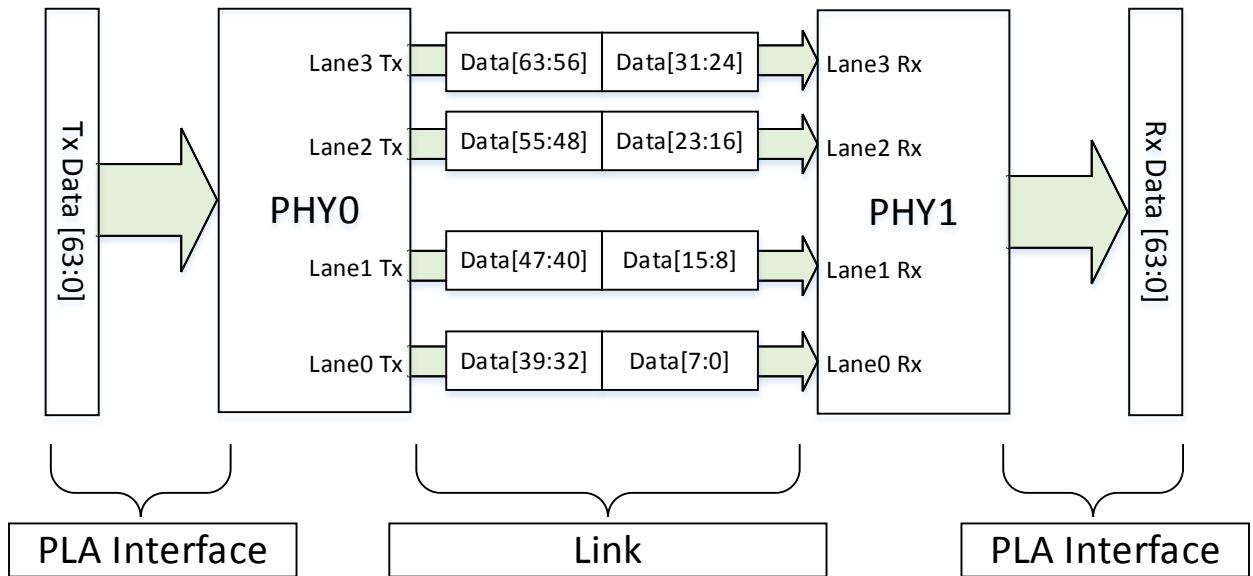


Figure 13-2: Data Striping on a Four-lane link with a 64-bit PLA Interface

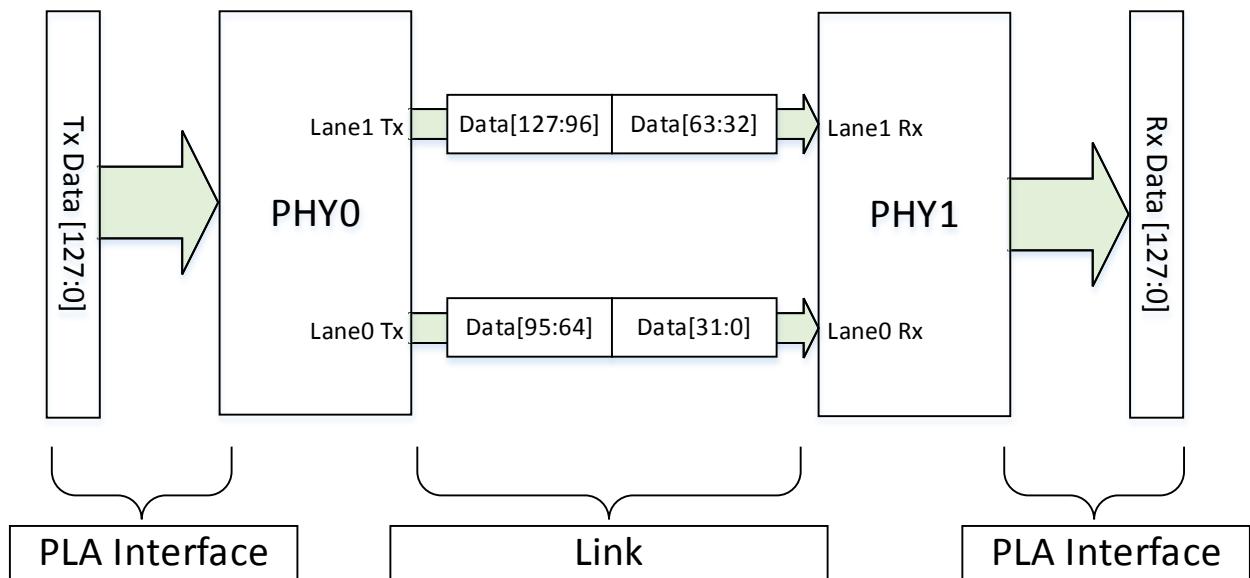


Figure 13-3: Data Striping on a Two-lane link with a 128-bit PLA Interface

### 5 13.1.7. Data Width

The width of `PLA_RXDATA_WIDTH` and `PLA_TXDATA_WIDTH` on the PLA data interface shall be a multiple of the number of lanes times the data striping to ensure proper alignment.

- Each width shall be an integer 4-byte multiple. For example, data widths for a two-lane link shall be an integer 8-byte multiple (2 lanes \* 4-byte striping), and data widths for an eight-lane link shall be an integer 8-byte multiple (8 lanes \* 1-byte striping).
- TX and RX widths may be different, i.e., an asymmetric link.

### 13.1.8. Packet Alignment

The Gen-Z Core is responsible for packing packets on the PLA transmit data interface.

- An implementation may limit the number of packets it receives per assertion of `PHY_RXVALID` by enforcing packet alignment (i.e. defining where in the byte stream that start-of-packet occurs in terms of an integer 4-byte multiple).
- The minimum packet alignment is determined by the maximum number of non-*Link Idle* packets the Gen-Z Core can receive and process per `PHY_RXVALID` assertion.
  - Receiver-imposed packet alignment is reported in *Interface Structure* Rx Packet Alignment, and uses the equation  $((\text{PLA_RXDATA_WIDTH} / \text{Max_Packets_Per_RXVALID}) / 4 \text{ Bytes})$ , rounded up, where `Max_Packet_Per_RXVALID` is implementation-specific. For example, a receiver that can process 16 packets per `PHY_RXVALID` assertion with a 256-byte PLA interface would use integer 16-byte alignment (256 Bytes / 16 Bytes) packets per `PHY_RXVALID`, and report Rx Packet Alignment = 4, i.e., (16 Bytes / 4 Bytes).
  - The transmitter packet alignment for non-*Link Idle* packets that the Gen-Z Core can send per `CORE_TXVALID` assertion shall be limited by *Interface Structure*'s Tx Packet Alignment.
    - Starting with the first assertion of `CORE_TXVALID` in the `PHY-Up*` states, the Gen-Z Core shall start a non-*Link Idle* packet on a (`Tx Packet Alignment * 4`) boundary (i.e., only *Link Idle* packets may start outside of a (`Tx Packet Alignment * 4`) byte boundary). At the end of a packet, the Gen-Z Core shall transmit *Link Idle* packets until reaching the (`Tx Packet Alignment * 4`) boundary.
    - A new non-*Link Idle* packet may be started only after at least (`Tx Min Packet Start * 4 bytes`) from the start of the prior packet.

### 13.1.9. Error Management

The Physical Layer is responsible for detecting errors. The errors are communicated to the PLA using Physical Layer Configuration registers and the PLA interface. Since detection mechanisms for high-level errors such as CRC are built into the protocol, packet errors are not identified by the Physical Layer. Error detection by the Physical Layer is restricted to decoding, disparity and buffer errors.

### 13.1.10. PCIe Physical Layer

Gen-Z supports interfacing with a PCIe Physical Layer. Refer to the Gen-Z PHY Specification for Physical Layer details such as PLA Link State mapping to LTSSM states. After a PCIe Physical Layer state transition from `PHY-Down*` or `PHY-LP*` states to `PHY-Up*` states, the PCIe Physical Layer shall transmit Gen-Z packets using the following rules:

- The PCIe Physical Layer shall start requesting Gen-Z packets from the Gen-Z Core by asserting `PHY_TXDATAREQ` on the PLA interface.
- The PCIe Physical Layer shall backpressure the Gen-Z Core by de-asserting `PHY_TXDATAREQ` signal as necessary to account for sync header removal or when PCIe Order Set or PCIe Framing Tokens are sent on the link.
- The PCIe Physical Layer shall de-assert `PHY_RXVALID` as needed to account for removal of sync headers, PCIe Ordered Sets, or PCIe Framing Tokens.

- Optional Gen-Z PLA IDLE Symbols are not supported by the PCIe Physical Layer. The Gen-Z Core shall be responsible for inserting Link Local Idle packets on the PLA interface if the Gen-Z Core has nothing to send.
- If Flit CRC encoding is supported and 8b/10b encoding is used and an encoding receiver error is detected as defined by the *PCI Express Base Specification*, then the PCIe Physical Layer shall treat an encoding receiver error as a Flit CRC error by asserting `PHY_RX_FLITCRC_ERR` for the entire Flit CRC code word. If Flit CRC encoding is not supported, then `PHY_RX_TRANSIENT_ERR` shall be asserted for one cycle instead.
- If 128b/130b encoding is used and a framing error is detected as defined in the *PCI Express Base Specification*, then the receiver shall signal an error to the Gen-Z Core by asserting `PHY_RX_RETRAIN_ERR` on the PLA interface, and shall follow the framing error recovery sequence defined in the PCIe Base Specification.

## 13.2. Optional Physical Layer Features

### 13.2.1. Power States

Power management allows the physical layer to minimize power consumption during periods of low activity. Three power states P0, P1 and P2 are defined. Support for these states is reported in the P0, P1, and P2 capabilities registers.

The PLA specifies two low-power states—PHY-LP and PHY-Up-LP as specified in *PHY Low-power States*. A PHY may support these low-power states.

Table 13-2: PHY Low-power States

State	Description
PHY-LP[1-4]	In the PHY-LP states, the PHY is down and cannot transmit or receive. Entry and exit latencies depend on the type of low-power features invoked in each state. For example, transitioning to electrical idle or turning off the transmitters and receivers.
PHY-Up-LP[1-4]	The PHY is up and capable of sending and receiving data across the link, but operates at a non-maximum bandwidth level, e.g., due to dynamic link width reduction. Entry and exit latencies depend on the type of low-power features invoked in each state. Packet transmission and receipt and receiving data may stall during transitions into and out of PHY-Up-LP states.
PHY-Down-Retrain	PHY-Down-Retrain is a faster retraining mechanism than going from PHY-Up to PHY-Down. The PHY transmitter remains active and transmitter equalization settings are held. The PHY receiver remains powered on. The PHY cannot send or receive data in this state. Configuration registers are not reset.

### 13.2.2. Link Width Reduction

The Physical Layer may adjust the width of the link during link training, e.g., due to lane failure. In the event the link width is reduced, the width of `PLA_RXDATA_WIDTH` and `PLA_TXDATA_WIDTH` on the PLA

data interfaces shall not change. The PHY shall account for the reduced width by reducing bandwidth by constraining **PHY\_TXDATAREQ** and **PHY\_RXVALID** signals.

### 13.2.3. Link Speed Reduction

The Physical Layer may adjust the speed of the link during link training, e.g., to meet BER requirements.

### 5 13.2.4. Retrain State

The Physical Layer may support a PHY-Down-Retrain state. PHY-Down-Retrain may optimize the retraining steps (e.g., eliminate detection) to reduce the physical retraining time.

### 13.2.5. IDLE Symbols

10 The Physical Layer may support symbol encoding on top of Gen-Z protocol traffic. Symbol encoding may be used to interrupt a Gen-Z packet with an IDLE symbol. Without support for symbol encoding, the Gen-Z Core shall assert **CORE\_TXVALID** after **<TxRequest To Valid Delay>** number of cycles following the assertion of **PHY\_TXDATAREQ**. If IDLE symbols are supported, then the Gen-Z Core may de-assert **CORE\_TXVALID** which signals the PHY to send an IDLE symbol.

15 If IDLE symbols are supported, then the Gen-Z Core may de-assert **CORE\_TXVALID** within packet boundaries of an end to end packet; this signals the PHY to send an IDLE symbol. This enables cut-through routing which reduces latency by allowing the Gen-Z Core transmitter to start packet transmission prior to receiving and buffering the entire packet. **CORE\_TXVALID** may not be de-asserted when the Gen-Z Core transmitter is not in the middle sending a packet. Instead, Link Local Idle Packets shall be inserted between end-to-end and link local packets by the Gen-Z Core when Gen-Z Core has 20 nothing to send."

### 13.2.6. Interface Aggregation PLA Impacts

25 A solution may require multiple interfaces to be aggregated and operate as a single interface. If the interface is configured to be a SAI, then the **CORE\_TXDATA** and **PHY\_RXDATA** signals from all aggregated interfaces shall be combined and all other signals shall use the SAI. All NAI valid and control signals shall be ignored. Aggregation shall be configured prior to the start of link training (see *Interface Structure*).

### 13.2.7. Flit Encoding

30 An optional encoding / decoding layer is supported between the physical layer and the Gen-Z link layer. This layer is used to provide stronger error protection for links with higher BER or error propagation resulting from the use of Rx Decision Feedback Equalization (DFE). The *Gen-Z Physical Specification* specifies if Flit Encoding is required for a given channel. This layer is used to encode 512 packet bits as a 528-bit flit with a 16-bit CRC, referred to as the *Flit CRC*. Flit CRC protection supplements, but does not replace packet CRC protection.

35 *Transportation of Flit on Four Lane Link* illustrates how a flit would be transported on a 4-lane link. The flit is striped across lanes with byte-granularity. A subsequent flit is contiguously placed, i.e., it starts with a byte transported on lane 2, and so on. Each flit payload contains 16 32-bit units of the packet stream, and the packet stream can consist of a single packet, a partial packet, or multiple packets.

lane3		lane0	
24	16	8	0
25	17	9	1
26	18	10	2
27	19	11	3
28	20	12	4
29	21	13	5
30	22	14	6
31	23	15	7
56	48	40	32
57	49	41	33
58	50	42	34
59	51	43	35
60	52	44	36
61	53	45	37
62	54	46	38
63	55	47	39
•		•	
504	496	488	480
505	497	489	481
506	498	490	482
507	499	491	483
508	500	492	484
509	501	493	485
510	502	494	486
511	503	495	487
		crc8	crc0
		crc9	crc1
		crc10	crc2
		crc11	crc3
		crc12	crc4
		crc13	crc5
		crc14	crc6
		crc15	crc7

Figure 13-4: Transportation of Flit on Four Lane Link

### 13.2.8. Flit Encoding Implementation Overview

The flit encoding / decoding layer may be inserted between the physical layer and the Gen-Z link layer, as illustrated in *Optional Flit Encode/Decode Layer*. The physical layer is unaware of Gen-Z packet boundaries or flit boundaries – it transmits and receives bits on its PLA interface. Though the Gen-Z link layer is unaware of flit encoding (or absence thereof), it is aware of packet boundaries. Therefore, the

flit decoding layer shall remove the flit CRC, and shall contiguously place flit payloads as it passes data to the Gen-Z link layer.

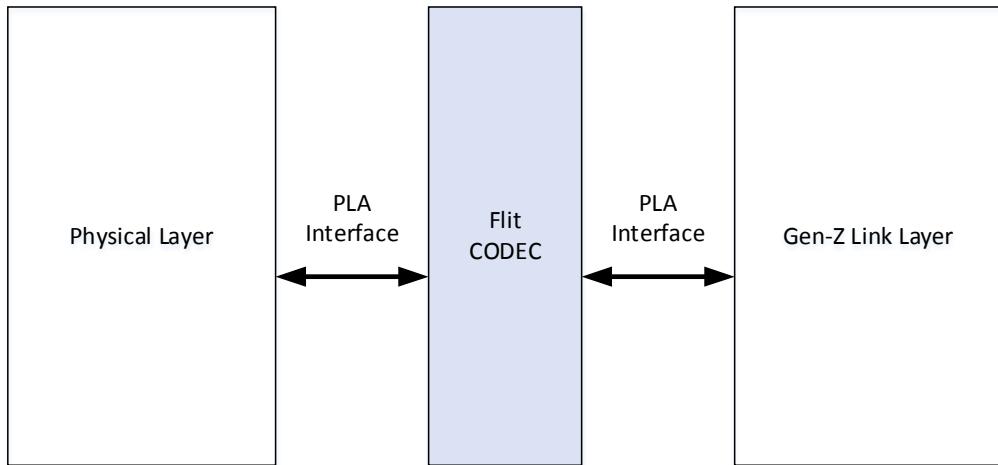


Figure 13-5: Optional Flit Encode/Decode Layer

- 5 The Flit CODEC uses `PHY_RX_FLITCRC_ERR` to signal flit CRC errors to the link layer. The PHY uses `PHY_RX_TRANSIENT_ERR` to signal lower-level PHY errors which require a link resynchronization.
- When the Flit CODEC is not present, the `PHY_RX_TRANSIENT_ERR` output from the PHY is directly connected to the link layer input, and the `PHY_RX_FLITCRC_ERR` input is hardwired to zero.
- 10 When the Flit CODEC is present, the `PHY_RX_TRANSIENT_ERR` signal from the PHY is an input to the Flit CODEC, and the Flit CODEC drives its own `PHY_RX_TRANSIENT_ERR` signal to the link layer.
- PHY\_RX\_TRANSIENT\_ERR shall be realigned by the Flit CODEC such that it's conservatively asserted along with any bits that the physical layer flagged with `PHY_RX_TRANSIENT_ERR`.
- 15 The Flit CODEC shall reset by clearing its internal encode and decode state when `CORE_PHYRESET` is set to 1 from the Gen-Z Core. Whenever the `PHY_TX_STATE` from the physical layer is in any of the PHY-Down\* or PHY-LP\* states, the FLIT CODEC encode state shall be cleared. Similarly, whenever the `PHY_RX_STATE` from the physical layer is in any of the PHY-Down\* or PHY-LP\* states, the FLIT CODEC decode state shall be cleared.
- In some use cases, the Flit CODEC layer may be physically present, but unused. To support this, the Flit CODEC may have a bypass mode where the PLA interface is effectively a pass through. Dynamic 20 reconfiguration of this mode is not supported.

### 13.2.9. Flit Encode

Since the flit encode is adding bits to the stream of Gen-Z packets, it uses PLA interface flow control when necessary to limit link-layer packet transmission rate. *Example Implementation of Flit Encode Layer* illustrates a high-level implementation of the flit encode logic. A 256-bit PLA interface is assumed; so the flit encoder calculates and appends CRC for two consecutive cycles of packet bits received from the link layer. The resulting 528-bit flit enters a rate matching buffer where it is realigned as necessary and transmitted on the PHY-side PLA interface.

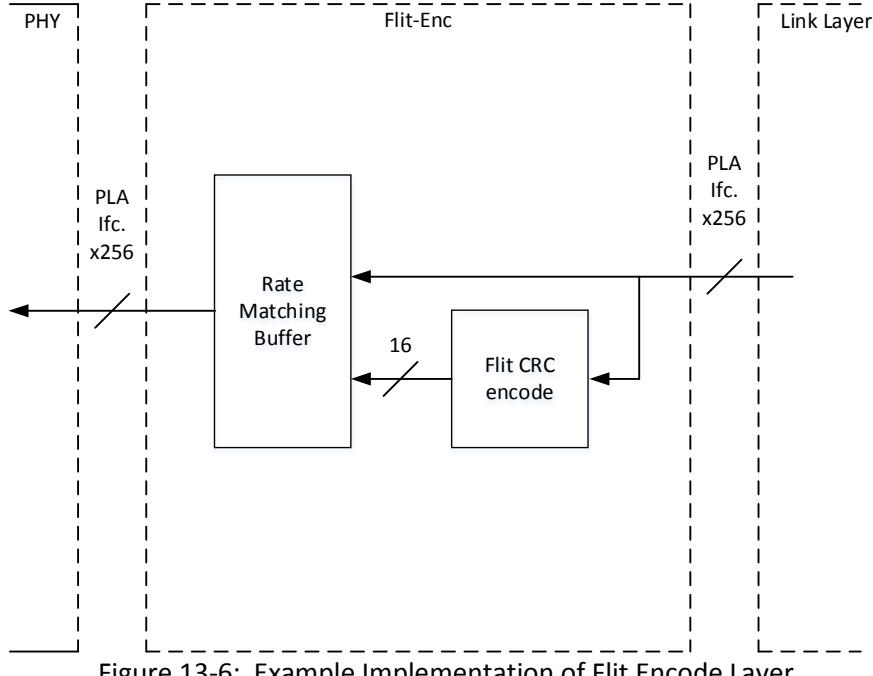


Figure 13-6: Example Implementation of Flit Encode Layer

If the PHY does not support a symbol encoding which allows insertion of IDLE symbols, the Flit CODEC shall transmit valid bits on CORE\_TXDATA (with CORE\_TXVALID assertion) a fixed number of cycles after assertion of PHY\_TXDATAREQ. This may require some orchestration of the PLA interface \*STATEREQ and \*\_STATE handshakes such that the Flit CODEC can assert PHY\_TXDATAREQ to the link layer earlier than the PHY asserts PHY\_TXDATAREQ to the Flit CODEC.

### 13.2.9.1. Flit Decode

*Example Implementation of Flit Decode Layer* illustrates a high-level implementation of flit decode logic. This example assumes a 256-bit PLA interface. Bits received from the PHY on PHY\_RXDATA are collected in a rate matching buffer. When a full 528-bit flit is available, the flit CRC decode is performed, while transmitting the first 256 bits of the flit payload to the Link Layer. If a CRC error is detected, PHY\_RX\_FLITCRC\_ERROR is asserted to the Link Layer along with the packet bits. The second 256 bits of the flit payload are staged and transmitted to the Link Layer in a second cycle. If a CRC error was detected, PHY\_RX\_FLITCRC error will also be asserted in the second cycle.

In the link layer, PHY\_RX\_FLITCRC\_ERR feeds into the packet CRC decode. The link layer tracks packet boundaries, and it shall force detection of a CRC error for any packet that contains bits received on PHY\_RXDATA while PHY\_RX\_FLITCRC\_ERR is asserted.

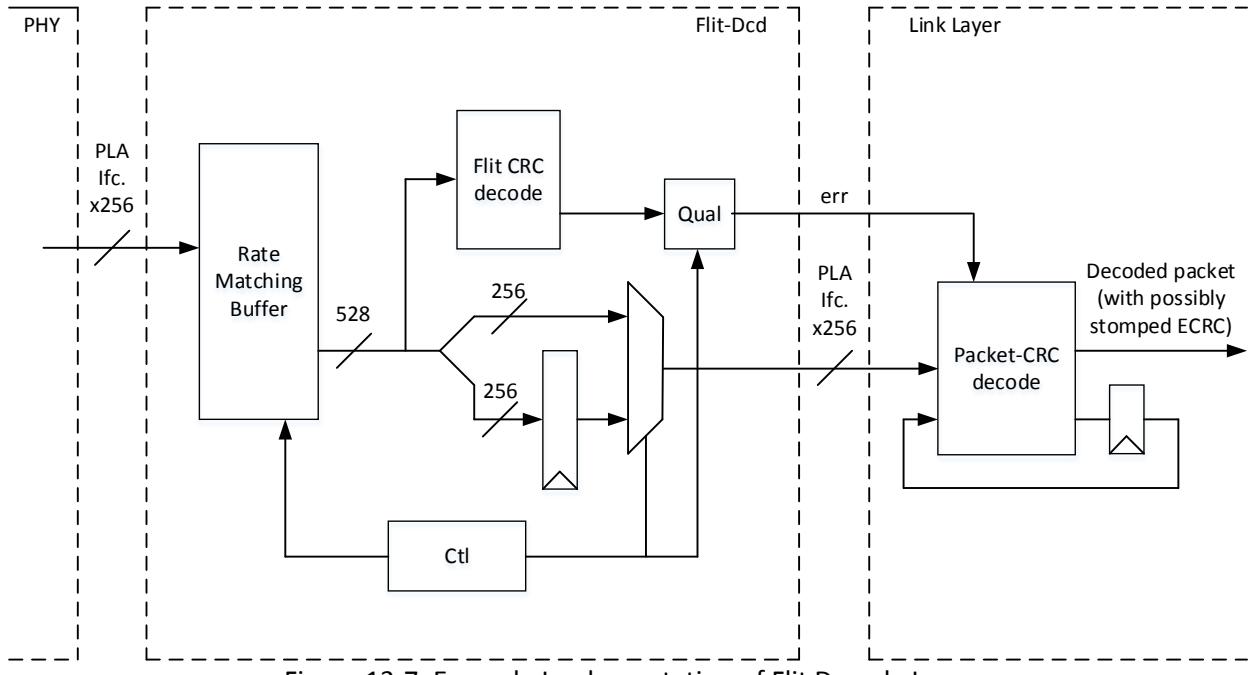


Figure 13-7: Example Implementation of Flit Decode Layer

### 13.3. PLA Interface

The *PLA Interface* shall be as illustrated. The PLA interface connects the Gen-Z Core and the PHY. This abstracts the PHY-specific details, and enables the Gen-Z Core to be compatible with multiple PHY.

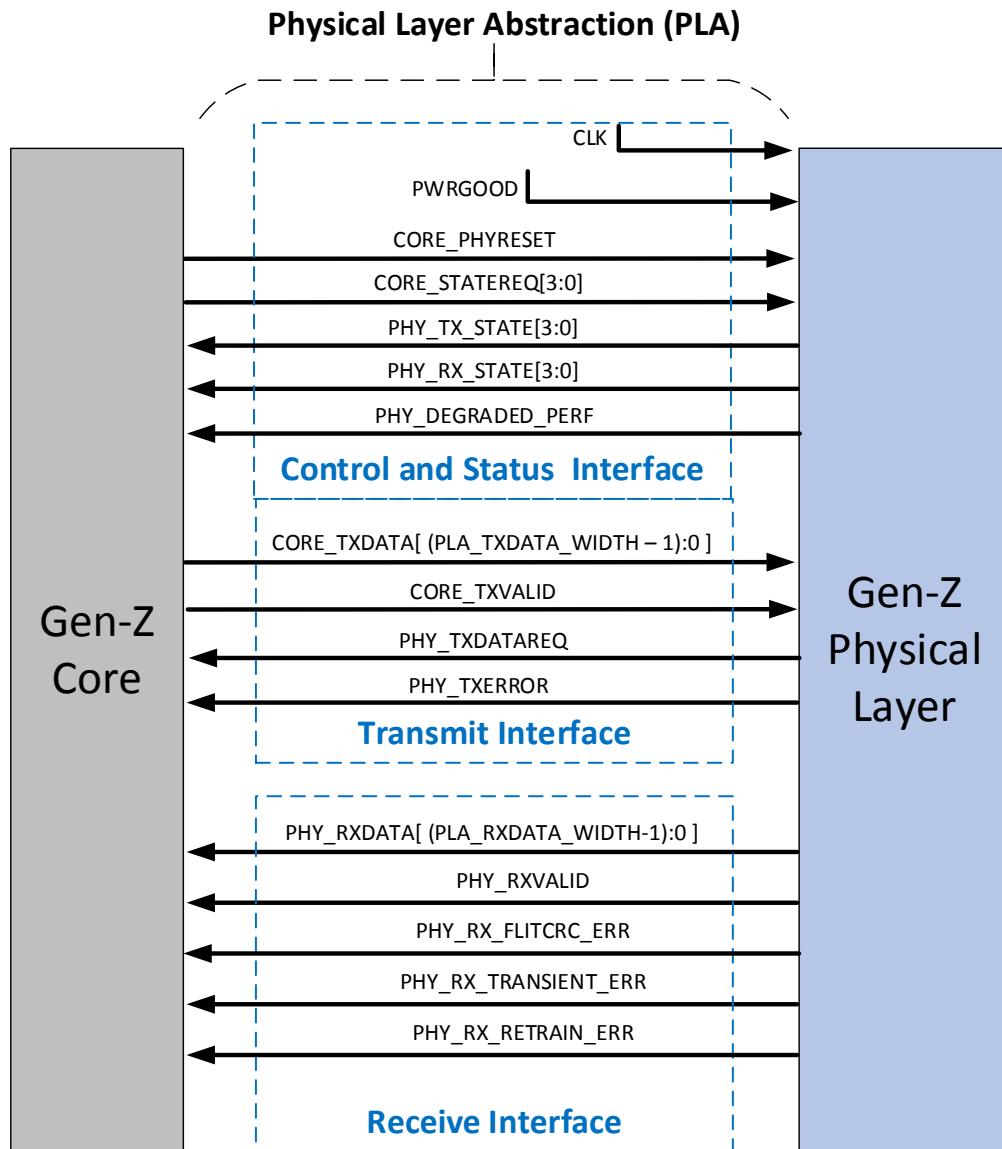


Figure 13-8: PLA Interface

### 13.4. PLA Interface Signals

The following terminology is used in this section:

- 5 • ‘Direction’ column indicates the direction of the signals from the Physical Layer perspective.
- An ‘Input’ signal is driven from the Gen-Z Core to the Physical Layer.
- An ‘Output’ signal is driven from the Physical Layer to the Gen-Z Core.
- ‘Detection Type’ column indicates the relevant condition of the signal.
  - ‘Level’ means that signal will present information as high or low voltage.
  - ‘Transition’ means that signal will present information using a voltage transition from low-to high or from high-to-low voltage level.
  - ‘Clock’ means that the signal will be used for synchronization of other signals.
  - ‘Async’ means that signal will present information that is not synchronized to any clock.

Table 13-3: Control and Status Interface Signals

Signal Name	Direction	Detection Type	Description
<b>CLK</b>	Input	Clock	Primary clock for the Gen-Z Core and Physical Layer.
<b>PWRGOOD</b>	Input	Asynch	Device powergood signal used to initialize logic states upon power-up
<b>CORE_PHYRESET</b>	Input	Level	Reset for the PHY from the Gen-Z Core
<b>CORE_STATEREQ[3:0]</b>	Input	Level	<p>Control Request from the Gen-Z Core to change the state of the link. The following encoding is used:</p> <ul style="list-style-type: none"> <li>○ 0x0: PHY-Down</li> <li>○ 0x1: PHY-Up</li> <li>○ 0x2: PHY-Down-Retrain</li> <li>○ 0x3-0x6: PHY-Up-LP[1-4]</li> <li>○ 0x7-0xA: PHY-LP[1-4]</li> <li>○ 0xB-0xF: Reserved</li> </ul> <p>To initiate a PHY retrain, the Gen-Z Core should request a transition to PHY-Down or PHY-Down-Retrain, wait for both the <b>PHY_TX_STATE</b> and <b>PHY_RX_STATE</b> to transition to the requested state, and then request a transition to PHY-Up or a Low Power state.</p>
<b>PHY_TX_STATE[3:0]</b>	Output	Level	<p>Signal from the Physical Layer indicating the transmitter link state which uses the same encoding as <b>CORE_STATEREQ</b>.</p> <p>The PHY has completed the Request signaled by <b>CORE_STATEREQ</b> when <b>PHY_TX_STATE</b> and <b>PHY_RX_STATE</b> reach the same value as <b>CORE_STATEREQ</b>.</p>
<b>PHY_RX_STATE[3:0]</b>	Output	Level	<p>Signal from the Physical Layer indicating the receiver link state which uses the same encoding as <b>CORE_STATEREQ</b>.</p> <p><b>PHY_TX_STATE</b> and <b>PHY_RX_STATE</b> may be different due to PHY state transition skew between the transmitter and receiver. Gen-Z Core shall not issue a new <b>CORE_STATEREQ</b> until both the transmitter and receiver have reached the current <b>CORE_STATEREQ</b> value.</p>

Signal Name	Direction	Detection Type	Description
<b>PHY_Degraded_PERF</b>	Output	Level	<p>Signal from the Physical Layer that when set to 1 indicates the link is running with unplanned reduced performance due to lane failure or reduced Physical Layer signaling rate. When set to 0, the link is performing at the level expected for the current <b>PHY_TX_STATE</b> and <b>PHY_RX_STATE</b>.</p> <p>If set to 1, then the <i>Interface Error Fields</i> Unexpected Degraded Link Performance actions shall be taken as configured.</p>

The following figures illustrates different PHY state transitions.

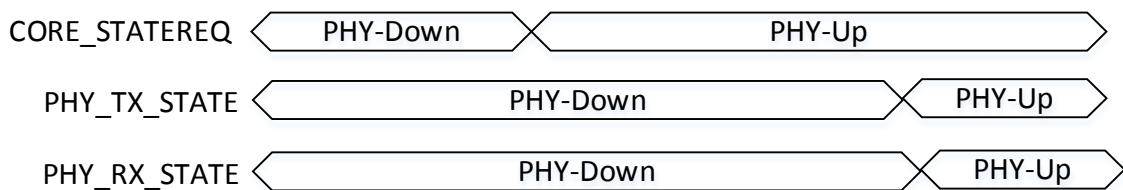


Figure 13-9: PLA PHY-Down to PHY-Up State Transition

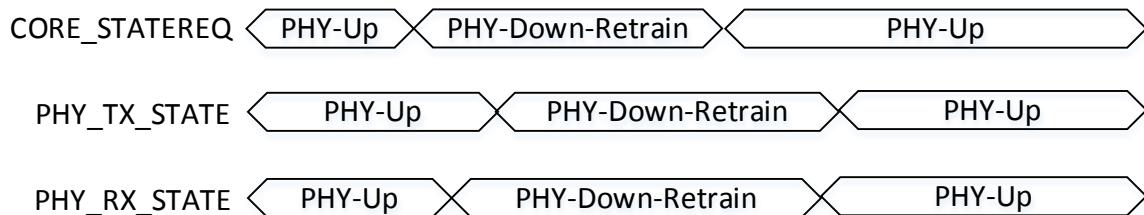


Figure 13-10: PLA PHY-Down-Retrain to PHY-Up State Transition

Table 13-4: Transmitter Interface Signals

Signal Name	Direction	Detection Type	Description
<b>CORE_TXDATA [PLA_TXDATA_WIDTH-1:0]</b>	Input	Level	<p>Parallel data bus from the Gen-Z Core to the Physical Layer.</p> <p>The <b>PLA_TXDATA_WIDTH</b> is implementation-specific, and shall meet the requirements defined in <i>Data Width</i>.</p>
<b>PHY_TXDATAREQ</b>	Output	Level	Data Request indicator from the Physical Layer to the Gen-Z Core for the entire <b>CORE_TXDATA</b> bus.

Signal Name	Direction	Detection Type	Description
			<p>The Physical Layer uses this signal to match the link bandwidth with the Gen-Z Core bandwidth.</p> <p>When this signal is asserted, the Physical Layer is requesting data CORE_TXDATA &lt;TxRequest To Valid Delay&gt; number of cycles in the future.</p> <p>When this signal is not asserted, the Physical Layer backpressures the Gen-Z Core such that the Gen-Z Core shall stall for each cycle that PHY_TXDATAREQ is not asserted.</p> <p>&lt;TxRequest To Valid Delay&gt; is a programmable configuration setting, TxDLY, in the <i>Interface PHY Structure</i>.</p>
CORE_TXVALID	Input	Level	Signal from the Gen-Z Core to the Physical Layer indicating CORE_TXDATA is valid. This is a delayed version of PHY_TXDATAREQ.
PHY_TXERROR	Output	Level	<p>Signal from the Physical Layer to the Gen-Z Core indicating that a transmitter error was detected (e.g. transmit phased-lock loop lock lost).</p> <p>The Gen-Z Core shall initiate a link retrain to recover from TXERRORs.</p> <p>This signal may be asserted with or without CORE_TXVALID.</p>

*PLA Tx Data Interface with <TxRequestToValidDelay> = 2* illustrates an example of Tx Flit transfers:

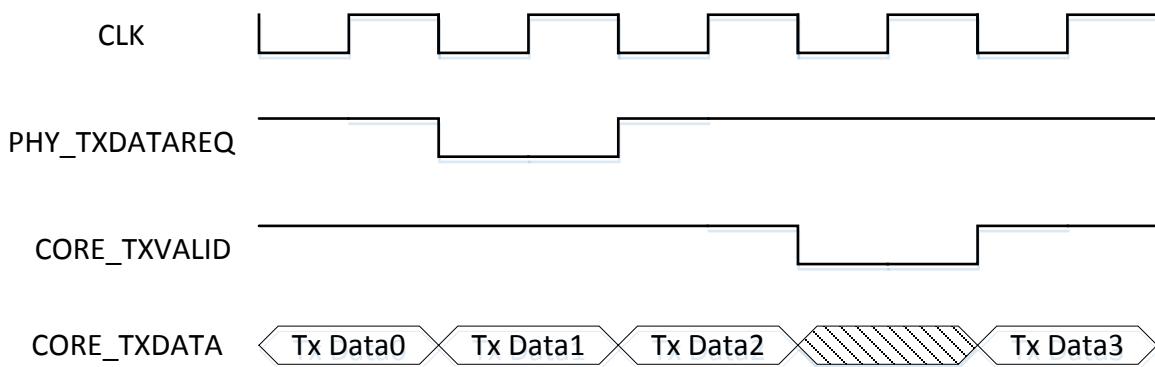


Figure 13-11: PLA Tx Data Interface with  $<\text{TxRequestToValidDelay}> = 2$

*PLA Tx Data Interface with <TxRequestToValidDelay> = 1* illustrates the same example as *PLA Tx Data Interface with <TxRequestToValidDelay> = 2* with the  $<\text{TxRequestToValidDelay}>$  set to 1.

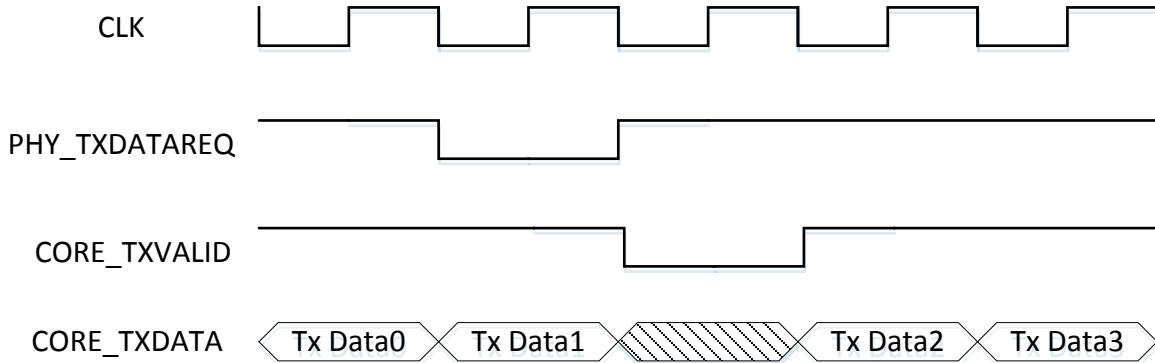
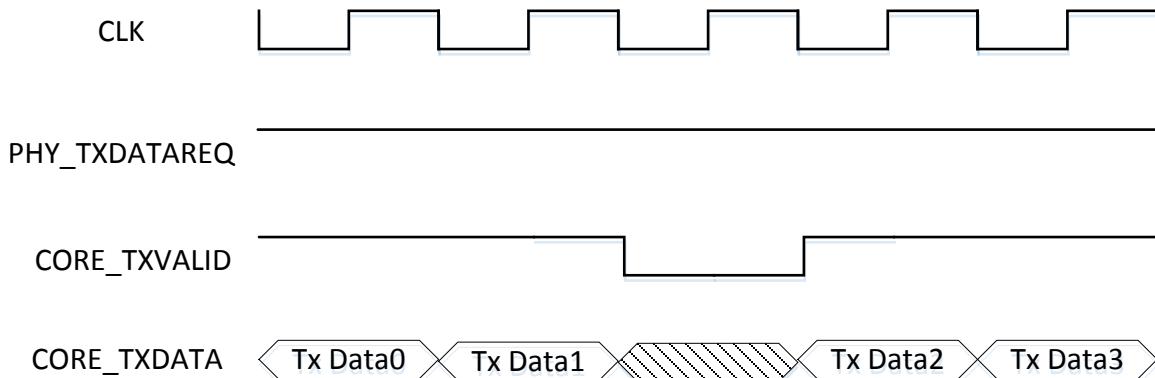


Figure 13-12: PLA Tx Data Interface with  $\langle \text{TxRequestToValidDelay} \rangle = 1$

*PLA Tx Data Interface with IDLE Symbol Support* illustrates a transmit bubble inserted by the Gen-Z Core when IDLE symbols are supported.



5

Figure 13-13: PLA Tx Data Interface with IDLE Symbol Support

Table 13-5: Receiver Interface Signals

Signal Name	Direction	Detection Type	Description
<b>PHY_RXDATA [PLA_RXDATA_WIDTH-1:0]</b>	Output	Level	Parallel data bus from the Physical Layer to the Gen-Z Core. The PLA_RXDATA_WIDTH is implementation specific, and shall meet the requirements defined in <i>Data Width</i> .
<b>PHY_RXVALID</b>	Output	Level	Signal from the Physical Layer to the Gen-Z Core indicating CORE_TXDATA is valid.
<b>PHY_RX_FLITCRC_ERR</b>	Output	Level	Signal indicating that a flit CRC error was detected. Only valid when optional flit-based CRC encoding is used, as described later in this chapter.  The Gen-Z Core shall force detection of a packet CRC error for any packet fully or partially within PHY_RXDATA on a cycle when PHY_RX_FLITCRC_ERR is asserted.

Signal Name	Direction	Detection Type	Description
PHY_RX_FLITCRC_ERR			PHY_RX_FLITCRC_ERR is only valid with PHY_RXVALID is asserted.
PHY_RX_TRANSIENT_ERR	Output	Level	<p>Signal from the Physical Layer to the Gen-Z Core indicating that a transient receiver error was detected which requires a link resynchronization.</p> <p>The Gen-Z Core shall decrement the TEC counter (see <i>Interface Structure</i>) for every cycle that PHY_RX_TRANSIENT_ERR is asserted.</p> <p>The Gen-Z Core shall initiate <i>Link Resynchronization</i>.</p> <p>The PHY shall keep PHY_RXVALID low on clock cycles when PHY_RX_TRANSIENT_ERR is asserted.</p>
PHY_RX_RETRAIN_ERR	Output	Level	<p>Signal from the Physical Layer to the Gen-Z Core indicating that the PHY receiver experienced an error that requires a PHY retrain to recover. The Gen-Z Core shall initiate a PHY retrain when PHY_RX_RETRAIN_ERR is asserted.</p> <p>The PHY shall keep PHY_RXVALID low on clock cycles when PHY_RX_RETRAIN_ERR is asserted.</p>

The following timing figures illustrate an Rx Flit transfer, a PLA transient error, and a PLA receiver retrain error.

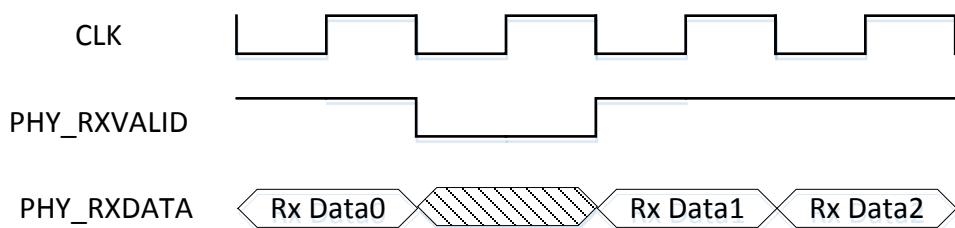


Figure 13-14: PLA Rx Data Interface

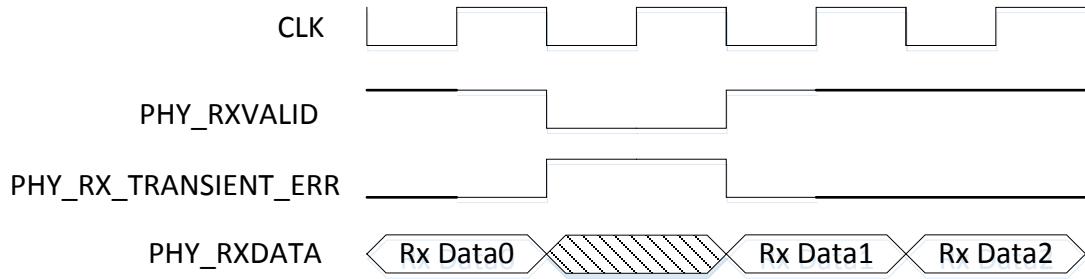


Figure 13-15: PLA Transient Error

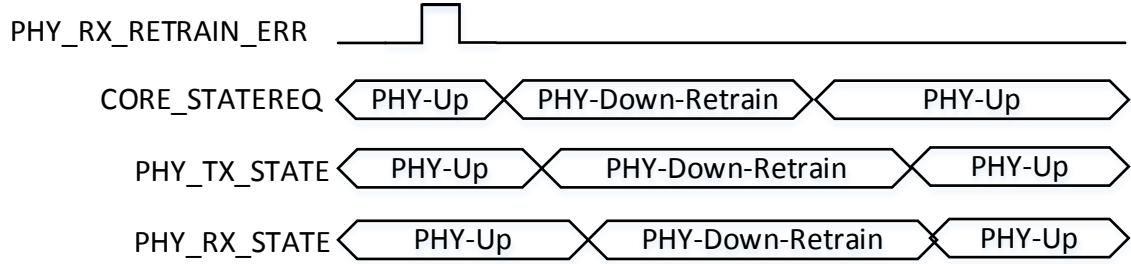


Figure 13-16: PLA Receiver Retrain Errors

## 5 13.5. Physical Layer Configuration

Physical layer configuration shall be performed through the *Interface PHY Structure*. If the interface supports the PCI Express physical layer, then PCI Express physical layer-specific configuration shall be performed as specified in the *PCI Express Base Specification*.

## 14. Multicast

Multicast uses one-to-many or many-to-many communications to send datagrams to a group of destination components starting with a single packet transmission. *Example Unreliable Multicast with Intervening Component Replication* illustrates a single Requester that transmits multicast request packets to a set of Responders. To support multicast operations, the following needs to occur:

- Multiple components need to be organized into a multicast group. Each multicast group needs to be configured such that there are no loops, e.g., a logical tree topology, that is overlaid onto a physical switch topology.
- Each multicast group is identified by a multicast group identifier.
  - If a multicast group spans a single subnet, then a multicast group identifier (MGID) is used in place of the unicast DCID in request packets.
  - If a multicast group spans multiple subnets, then a GMGID is constructed using the Global Multicast Prefix and a MGID (see *Explicit OpClass Multi-subnet Field*).
  - Within a single-subnet topology, MGID 0 shall be used only for management operations, e.g., service discovery, initialization, etc. Within a multi-subnet topology, the Global Multicast Prefix 0 combined with MGID 0 shall be used only for management operations.
  - Within a single-subnet topology, MGID 1 shall be used only for *Collective Operations*. Within a multi-subnet topology, the Global Multicast Prefix 1 combined with MGID 1 shall be used only for *Collective Operations*.
  - Switches use the MGID / GMGID to determine the egress interface to relay each copy of the multicast packet. In *Example Unreliable Multicast with Intervening Component Replication*, Requester A transmits packet T5. Component B replicates T5 and transmits a copy to components C, D, and E.
  - Responders use the MGID to validate multicast packets and forward to the payload to the associated application or service.

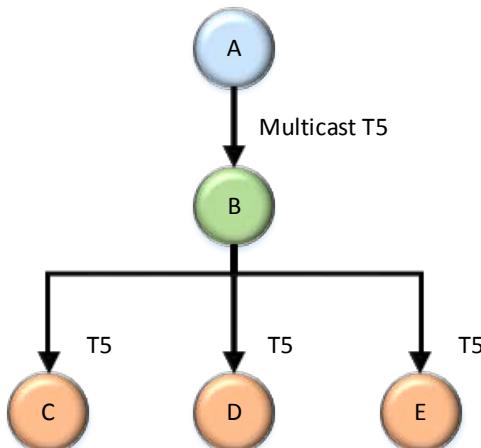


Figure 14-1: Example Unreliable Multicast with Intervening Component Replication

The following are the features and requirements associated with multicast operation:

- Any component type may participate in a multicast group as a Requester, a Responder, or a Requester-Responder.
- Within a subnet, each multicast group shall be identified by a unique MGID.
- Multicast groups that span multiple subnets shall be identified by a unique GMGID.

- Requesters and Responders that support multicast operations shall provision a *Component Multicast Structure*. A Requester or Responder joins or leaves a multicast group by updating the tables within this structure to reflect active or inactive MGIDs / GMGIDs.
- Switches and TRs that support multicast packet relay shall provision multicast packet relay tables as specified in the *Component Switch Structure*.
  - Each multicast relay table shall be configured to prevent relay loops among a given multicast group's members. A multicast group operates over a single logical hierarchical topology (conceptually, a tree topology) such that a request packet is not relayed to a component that has already processed the request packet.
- Unicast and multicast packets may be interleaved on the same physical topology.
- Multiple multicast groups may coexist on the same physical topology. Packets targeting different multicast groups may be interleaved on the same physical topology.
- Multicast request packets shall use the explicit Multicast OpClass. Packet relay components examine a request packet's OCL field to determine if the packet's destination field is treated as a DCID (unicast DCID) or a MGID (Multicast Group Identifier).
  - If the request packet's GC == 1b, then the packet's MGID field is combined with the Global Multicast Prefix field to create a GMGID.
    - Switches that do not support GMGIDs shall relay multicast packets using the request packet's MGID field. As a result, multiple global multicast groups may be mapped to a given subnet-local MGID.
- When a switch receives a multicast packet:
  - The switch shall validate the packet to ensure it can be relayed.
  - The switch shall replicate and relay the packet, exactly once, on each egress interface used to relay packets for the multicast group, except for the interface on which the packet arrived.
  - If a multicast packet cannot be relayed through a configured multicast group egress interface, e.g., the link has transitioned to the L-Down state or a forward progress timer expired, then the packet shall be silently discarded.
- Responders that receive a multicast packet in error shall silently discard the packet. If a component supports *In-band Management* and the *Component Error and Signal Event Structure*, then the Responder shall inform management as configured.
- Each multicast group shall be associated with an address range.
  - If supported, then software configures a *Requester PTE* with the MGID / GMGID, Address, and R-Key. The component uses the Requester PTE in conjunction with the Access Key and VC derived from the *Component Multicast Structure* entry to generate a request packet through each participating egress interface. If unsupported, then the component uses means outside of this specification's scope to configure the corresponding Address and R-Key.
  - If supported, software configures a *Responder PTE* with the R-Key. The component uses the Responder PTE in conjunction with the Access Key and VC derived from the *Component Multicast Structure* entry to validate a request packet. If unsupported, then the component uses means outside of this specification's scope to validate the corresponding Address and R-Key.

**Developer Note:** To optimize performance and improve software interoperability, Requesters are strongly encouraged to support a Requester ZMMU and Responders to support a Responder ZMMU.

Fields common to all unicast and multicast request and response packets are specified in *Base Formats for End-to-end Packets*. Unless explicitly stated otherwise by this specification, fields common to the unicast version of a given multicast request shall be as specified in the corresponding unicast request specification section.

## 5 14.1. Unreliable Multicast

The following are the features and requirements associated with all unreliable multicast request packets:

- Unreliable multicast does not provide Reliable Delivery.
  - If a Responder supports the *Component Error and Signal Event Structure* and if so configured, then Responder shall signal management if a request packet fails packet validation. If not, then Responders shall silently discard request packets that fail packet validation.
  - Requesters shall not maintain retransmission resources or timers associated with request packets.
  - Requesters shall stomp the ECRC field of unreliable multicast packets if conditions warrant.

### 14.1.1. Unreliable Multicast Write

The following are the features and requirements associated with all unreliable Multicast Write request packets:

- Unreliable Multicast Write request packets shall use the *Unreliable Multicast Write Request Packet Format*.
  - Unless explicitly stated otherwise by this specification, unreliable Multicast Write request packets shall operate as specified in *Write Requests*.

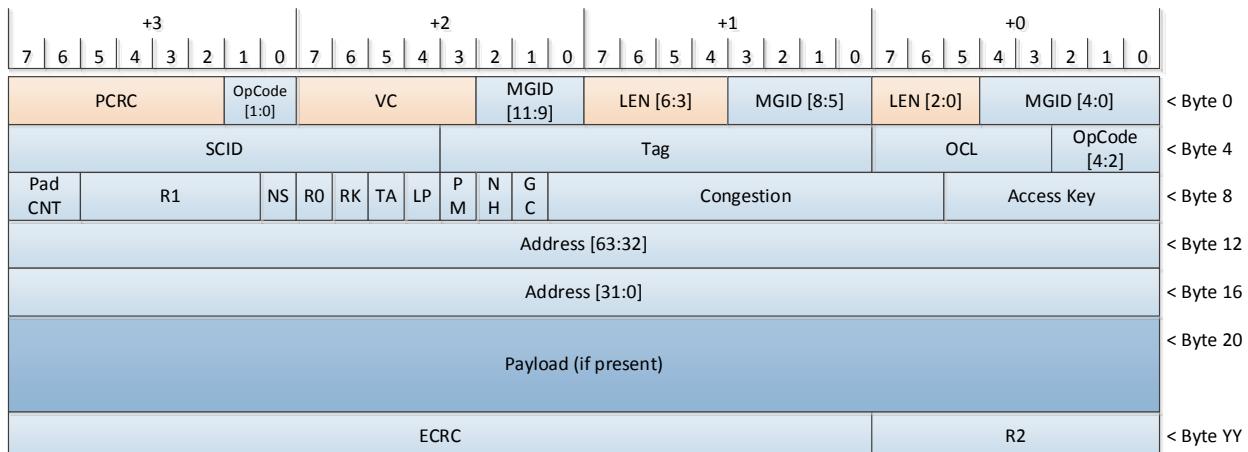


Figure 14-2: Unreliable Multicast Write Request Packet Format

Table 14-1: Unreliable Multicast-specific Write Request Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>MGID</b>	MG   MGID	12	Multicast Group Identifier within a single subnet.

Field Name	Field Abbreviation	Size (bits)	Description
R0			If GC == 1b, then the Global Multicast Prefix is combined to construct a Global Multicast Group Identifier (GMGID).
	-	1	Reserved
	-	5	Reserved
	-	8	Reserved

### 14.1.2. Unreliable Multicast Write MSG

The following are the features and requirements associated with all unreliable multicast Write MSG request packets:

- Unreliable multicast Write MSG request packets shall use the *Unreliable Multicast Write MSG Request Packet Format*.
  - Unless explicitly stated otherwise in *Unreliable Multicast-specific Write MSG Request Packet Fields*, all Unreliable Multicast Write MSG request packet protocol fields and associated semantics shall be as specified in *Write MSG*. Unlike its unicast analog, an unreliable multicast Write MSG does not contain context identifiers. Instead, the MGID / GMGID is used to locate a Responder-local context resource in order to target a destination buffer.

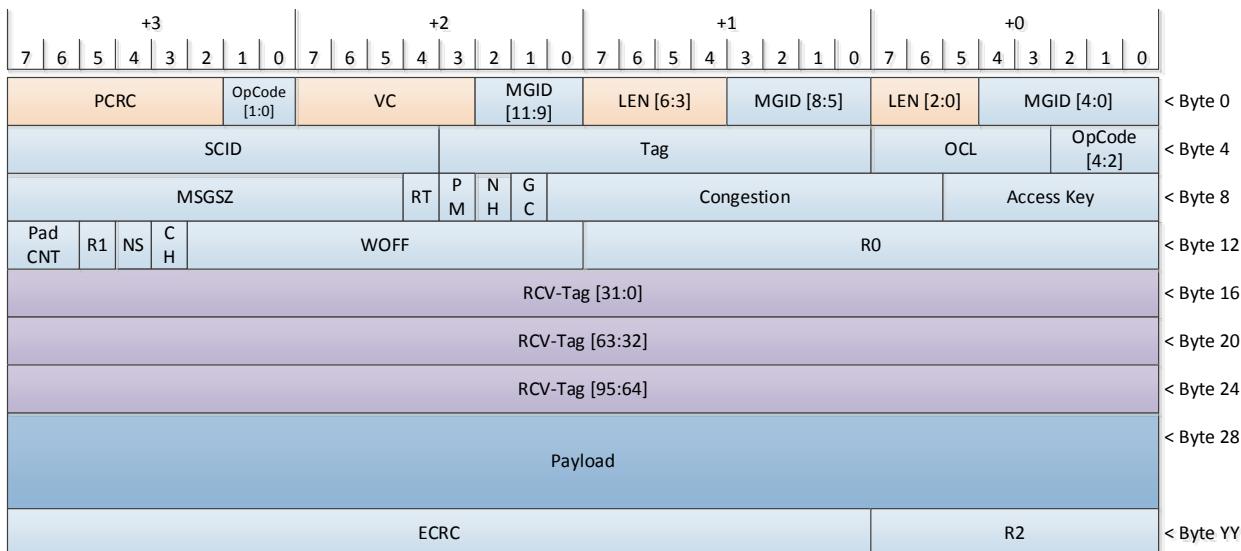


Figure 14-3: Unreliable Multicast Write MSG Request Packet Format

Table 14-2: Unreliable Multicast-specific Write MSG Request Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>MGID</b>	MG   MGID	12	Multicast Group Identifier within a single subnet.

Field Name	Field Abbreviation	Size (bits)	Description
			If GC == 1b, then the Global Multicast Prefix is combined to construct a Global Multicast Group Identifier (GMGID).
R0	-	16	Reserved
R1	-	1	Reserved
R2	-	8	Reserved

### 14.1.3. Unreliable Multicast Encapsulation Packet

An Unreliable Multicast Encapsulation packet is used to encapsulate a unicast OpClass request packet to all components participating within the associated multicast group. The multicast encapsulation header uses the same fields used to transmit and relay a packet to the multicast group as a non-encapsulated multicast request packet.

The following are the features and requirements of the Unreliable Multicast Encapsulation request packet:

- Unreliable multicast Encapsulation request packets shall use the *Unreliable Multicast Encapsulation Request Packet Format*.
- An Unreliable Multicast Encapsulation packet shall not encapsulate a Unicast Encapsulation packet, a SOD OpClass packet, or a Multicast OpClass packet.
- The length of an Unreliable Multicast Encapsulation packet shall be less than or equal to the maximum supported packet length.
- Packet validation shall be performed in two steps:
  - The first step shall validate the Unreliable Multicast Encapsulation packet's protocol fields—the header and tail fields that surround the encapsulated packet. These fields shall be validated as specified in *Destination Component Packet Processing*.
    - The Unreliable Multicast Encapsulation packet's ECRC shall protect the entire encapsulated packet.
  - The second step validates the actual encapsulated packet. The encapsulated packet shall be independently validated using encapsulated packet-specific protocol validation.
    - If present, then the encapsulated packet's VC, DCID, PCRC, SCID, Congestion, Access Key, and ECRC fields shall be Reserved.

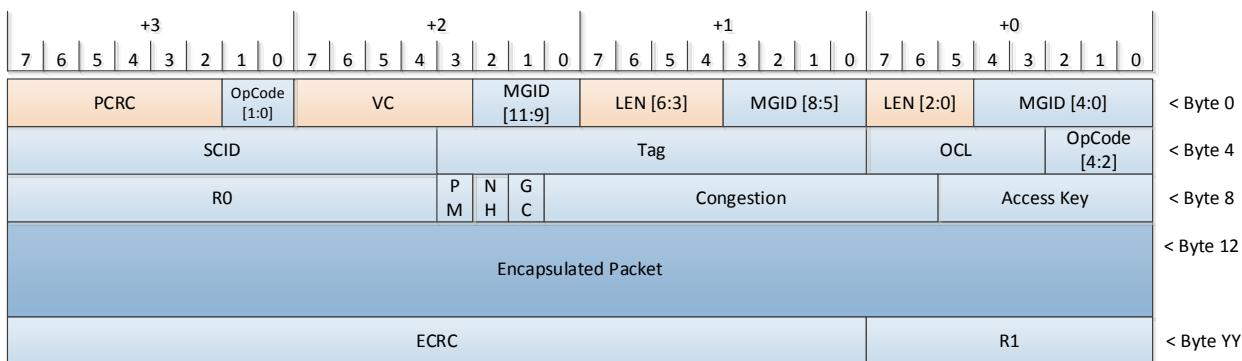


Figure 14-4: Unreliable Multicast Encapsulation Request Packet Format

Table 14-3: Unreliable Multicast Encapsulation-specific Request Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
MGID	MG   MGID	12	Multicast Group Identifier within a single subnet. If GC == 1b, then the Global Multicast Prefix is combined to construct a Global Multicast Group Identifier (GMGID).
R0	-	12	Reserved
R1	-	8	Reserved

## 14.2. Reliable Multicast

The following are the features and requirements associated with reliable multicast:

- Reliable multicast provides Reliable Delivery.
- A reliable multicast group shall have a single Requester.
- Responders shall use the *Reliable Multicast Acknowledgment* packet to return success or error notification.
  - Vendor-defined reliable multicast request packets may substitute a vendor-defined response in place of the Reliable Multicast Acknowledgment packet without impacting component interoperability. Reliable delivery is treated as vendor-defined.
  - A Responder shall validate and execute a reliable multicast request packet and transmit the corresponding response packet in less than or equal to the worst-case completion latency associated with the request packet type. If a Responder is unable to do so, then the Responder shall silently discard the reliable multicast request packet, and shall not transmit the corresponding response packet.
- A Requester shall maintain a retransmission timer for outstanding reliable multicast request packets to detect lost request packets and missing acknowledgments (e.g., due to transient and non-transient errors and events). The retransmission timer tracks the minimum time, before the Requester retransmits the unacknowledged request packet.
- Requesters shall stomp the ECRC field of reliable multicast packets if conditions warrant.
- A sequence number is associated with a given Reliable Multicast Sequence Number Space.
  - Each Reliable Multicast Sequence Number Space shall be associated with a single reliable multicast ordering domain. Each reliable multicast ordering domain is associated with a single Requester.
  - Each Reliable Multicast Sequence Number Space represents  $2^{16}$  Reliable Multicast Sequence Numbers.
    - Each Requester shall monotonically increment this field by one (modulo  $2^{16}$ ).
      - If GC == 0b, then Responders shall use the packet's [SCID, Tx Seq, MGID] to detect out-of-order, missing, or ghost packets.
      - If GC == 1b, then Responders shall use the packet's [SGCID, Tx Seq, GMGID] to detect out-of-order, missing, or ghost packets.
      - Each Responder in a reliable multicast group shall periodically or upon detecting a packet ordering issue transmit a Reliable Multicast

Acknowledgment packet that indicates the last successfully received request packet associated with this multicast group.

- A Requester shall never have more than  $((2^{16} - 1) / 2)$  outstanding multicast request packets.
- 5     ○ Reliable multicast Requesters and Responders shall support the entire Multicast Sequence Number Space.
- 10    ○ Reliable multicast sequence numbers shall be assigned to outstanding packets in monotonically increasing order (modulo  $2^{16}$ ).
  - Whenever a reliable multicast ordering domain is initialized, the first multicast end-to-end packet transmitted by a Requester shall set the multicast sequence number to one.
  - Whenever a Responder joins a reliable multicast group and the associated multicast ordering domain, the Responder shall set the next expected sequence number associated with this ordering domain to one. If the first valid request packet to arrive has a different sequence number, then the Responder shall treat that sequence number as valid and update the next expected sequence number to be that sequence number plus one.
  - If no multicast sequence numbers are available for use, then communication within the impacted reliable multicast ordering domain shall stall until outstanding packets are successfully acknowledged and the associated sequence numbers are released for reuse.
- 15    ○ Responders shall examine the reliable multicast sequence number to detect missing, duplicate, or out-of-order reliable multicast packets.
- 20    ○

#### 14.2.1. Reliable Multicast Write and Write Persistent

The following are the features and requirements associated with reliable multicast Write and Write Persistent request packets:

- Multicast OpClass Write and Write Persistent request packets shall use the packet format as illustrated in *Reliable Multicast Write and Write Persistent Request Packet Format*.
- 30    • Unless explicitly stated otherwise by this specification, the semantics of multicast Write and Write Persistent request packets shall as specified in *Write*.

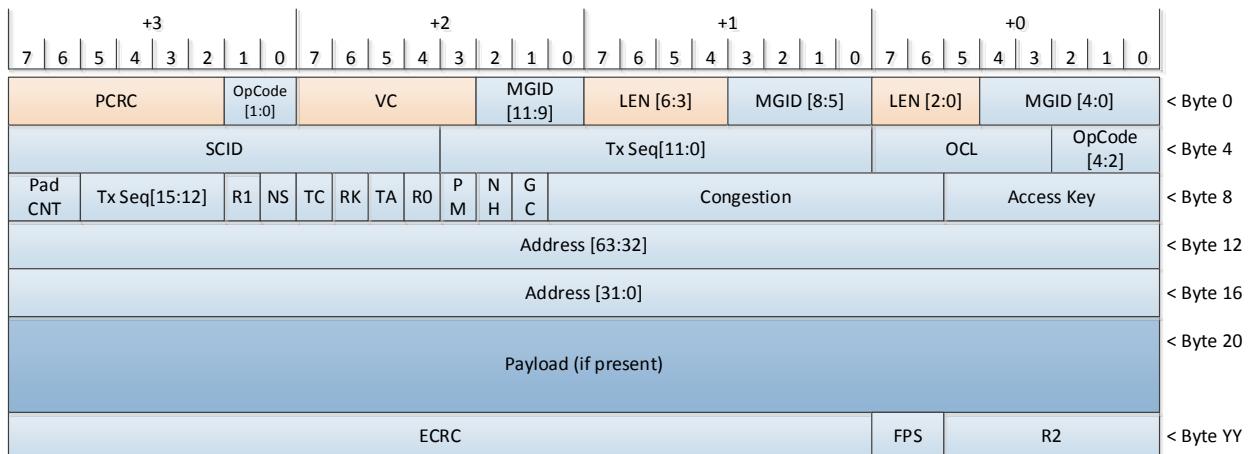


Figure 14-5: Reliable Multicast Write and Write Persistent Request Packet Format

Table 14-4: Reliable Multicast-specific Write Request Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
MGID	MG   MGID	12	Multicast Group Identifier within a single subnet. If GC == 1b, then the Global Multicast Prefix is combined to construct a Global Multicast Group Identifier (GMGID).
Tx Sequence Number	Tx Seq	16	Reliable multicast transmission end-to-end sequence number
R0	-	1	Reserved
R1	-	1	Reserved
R2	-	6	Reserved

## 14.2.2. Reliable Multicast Write MSG

The following are the features and requirements associated with reliable multicast Write and Write Persistent request packets:

- Multicast OpClass Write MSG request packets shall use the *Reliable Multicast Write MSG Request Packet Format*. Write MSG-specific packet fields are specified in *(Unreliable) Write MSG Request-specific Packet Fields*.
- A Responder shall transmit one Reliable Multicast Acknowledgment packet returned for each successfully received and executed Write MSG packet. The Request-specific field shall be set to the Write Offset of the corresponding request.
- Unlike its unicast analog, a reliable multicast Write MSG does not contain context identifiers. Instead, the MGID / GMGID is used to locate a Responder-local context resource in order to target a destination buffer.

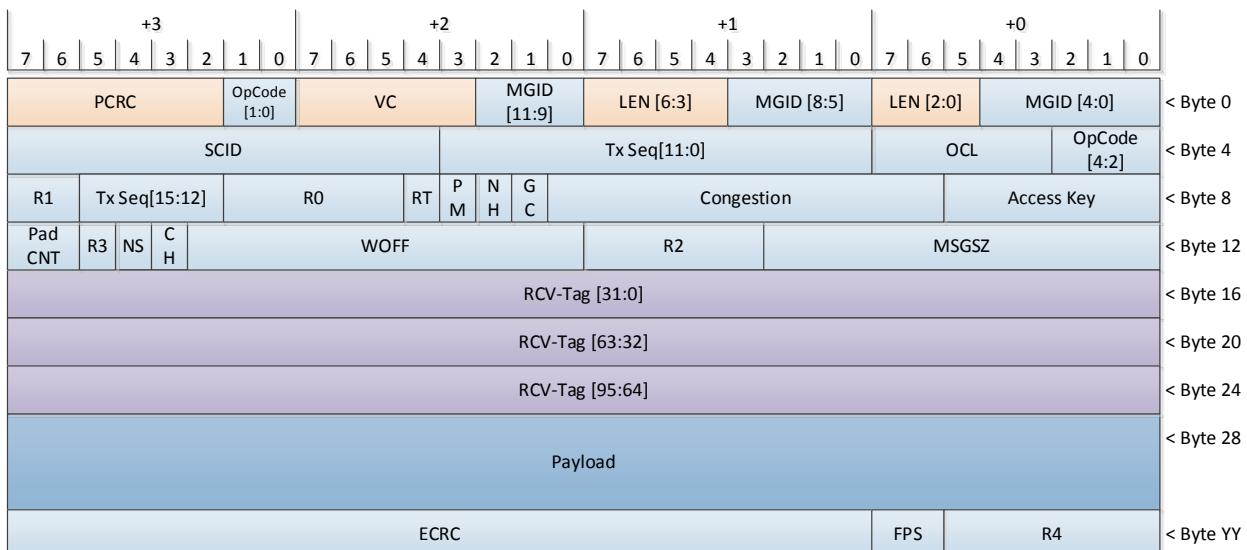


Figure 14-6: Reliable Multicast Write MSG Request Packet Format

Unless explicitly stated otherwise in *Reliable Multicast-specific Write MSG Request Packet Fields*, all protocol fields and associated semantics shall be as specified in *Write MSG*.

Table 14-5: Reliable Multicast-specific Write MSG Request Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>MGID</b>	MG   MGID	12	Multicast Group Identifier within a single subnet. If GC == 1b, then the Global Multicast Prefix is combined to construct a Global Multicast Group Identifier (GMGID).
<b>Tx Sequence Number</b>	Tx Seq	16	Reliable multicast transmission end-to-end sequence number.
<b>R0</b>	-	5	Reserved
<b>R1</b>	-	2	Reserved
<b>R2</b>	-	5	Reserved
<b>R3</b>	-	1	Reserved
<b>R4</b>	-	6	Reserved

### 14.2.3. Reliable Multicast Encapsulation Packet

A Reliable Multicast Encapsulation packet is used to encapsulate a unicast OpClass request packet to all components participating within the associated multicast group. The multicast encapsulation header uses the same fields used to transmit and relay a packet to the multicast group as a non-encapsulated multicast request packet.

The following are the features and requirements of the Reliable Multicast Encapsulation request packet:

- Reliable multicast Encapsulation request packets shall use the *Reliable Multicast Encapsulation Request Packet Format*.
- A Reliable Multicast Encapsulation packet shall not encapsulate a Unicast Encapsulation packet, a SOD OpClass packet, or a Multicast OpClass packet.
- The length of a Reliable Multicast Encapsulation packet shall be less than or equal to the maximum supported packet length.
- If applicable, then the *Forward Progress Screen (FPS)* Epoch shall be indicated using the encapsulated packet's FPS field.
- Packet validation shall be performed in two steps:
  - The first step shall validate the Reliable Multicast Encapsulation packet's protocol fields—the header and tail fields that surround the encapsulated packet. These fields shall be validated as specified in *Destination Component Packet Processing*.
    - The Reliable Multicast Encapsulation packet's ECRC shall protect the entire encapsulated packet.
  - The second step validates the actual encapsulated packet. The encapsulated packet shall be independently validated using encapsulated packet-specific protocol validation.
    - If present, then the encapsulated packet's VC, DCID, PCRC, SCID, Congestion, Access Key, and ECRC fields shall be Reserved.

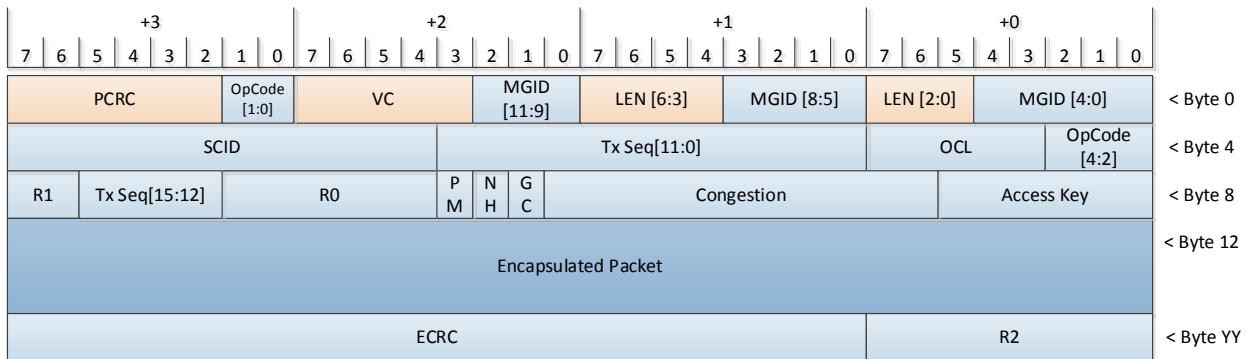


Figure 14-7: Reliable Multicast Encapsulation Request Packet Format

Table 14-6: Reliable Multicast Encapsulation-specific Request Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>MGID</b>	MG   MGID	12	Multicast Group Identifier within a single subnet. If GC == 1b, then the Global Multicast Prefix is combined to construct a Global Multicast Group Identifier (GMGID).
<b>R0</b>	-	6	Reserved
<b>R1</b>	-	2	Reserved
<b>R2</b>	-	8	Reserved

#### 14.2.4. Reliable Multicast Acknowledgment

The following are the features and requirements associated with reliable multicast acknowledgments:

- All reliable multicast packets shall be acknowledged.
  - Reliable multicast Write, Write Persistent, and Write MSG request packets shall be acknowledged by a Reliable Multicast Acknowledgment (MACK) packet.
  - Reliable multicast acknowledgments do not contain an RNR field. If a Responder is unable to execute the request packet, then it shall generate a MACK packet with the (Reason = Insufficient Responder Resources).
  - Vendor-defined reliable multicast request packets may be acknowledged by a vendor-defined acknowledgment packet or a MACK packet.
- MACK packets shall use the *Reliable Multicast Acknowledgment Packet Format*. MACK packets shall be unicast packets and are associated with the Advanced 2 OpClass.
- When a packet is positively acknowledged by all Responders, the Requester shall release all resources associated with the corresponding request packet.
- Requesters shall maintain a retransmission timer to detect missing response packets. The retransmission timer measures the time since the last reliable multicast packet was acknowledged by all Responders and released.
  - A Requester is not required to maintain a per request packet retransmission timer. Tracking the logical head of the Outstanding Request Packet list is sufficient.
  - If there are no Outstanding Request Packets and one is scheduled, then the Requester shall start the request packet retransmission timer.

- The request packet retransmission timer shall continuously run until all Outstanding Request Packets have been acknowledged, at which point, the retransmission timer shall stop.
- When a timeout is detected, the Requester shall retransmit the request packet at the head of the Outstanding Request Packet list. The Requester may retransmit all Outstanding Request Packets on the Outstanding Request Packet list or may wait until all Responders generate an acknowledgment to avoid flooding the topology.
- A MACK packet indicates the last valid packet received within a specific ordering domain at the time the acknowledgment was scheduled for transmission.
  - An acknowledgment shall be cumulative, that is, it acknowledges all Outstanding Request Packets up to and including the request packet indicated by the multicast sequence number.
    - If a Responder detects an error while processing a reliable multicast request packet, it shall immediately schedule a reliable multicast acknowledgement packet to communicate the last successfully received request packet and the corresponding error.
    - If a Responder successfully validates and executes a reliable multicast request packet and it wants to cumulatively acknowledge multiple request packets, then it may delay scheduling a reliable multicast acknowledgement packet for no more than the *Core Structure* Responder Deadline time.
  - An acknowledgment from any Responder that does not correlate to an outstanding request shall be silently discarded.
- A negative acknowledgment (NAK) is a MACK or vendor-defined acknowledgment packet with Reason set to the detected error value.
  - A NAK shall be cumulative; a NAK acknowledges all outstanding valid reliable multicast packets up to and including the packet indicated by the reliable multicast acknowledgment sequence number.
  - A NAK shall impact only its associated multicast ordering domain. Unrelated multicast ordering domains continue to flow, independent of the impacted multicast ordering domain's error recovery process.
  - A NAK shall indicate the highest precedence detected error in the Reason field. Reliable multicast use the same Reason codes as specified in *Standalone Acknowledgment*.
  - If a NAK indicates a transient error, then the Requester shall retransmit the packet at the head of the outstanding request list. The Requester may retransmit all packets on the outstanding request list, or may wait until all Responders generate an acknowledgment to avoid flooding the Responders.
  - If a NAK indicates a non-transient error, then the Requester shall be responsible for executing error recovery per its specific needs.

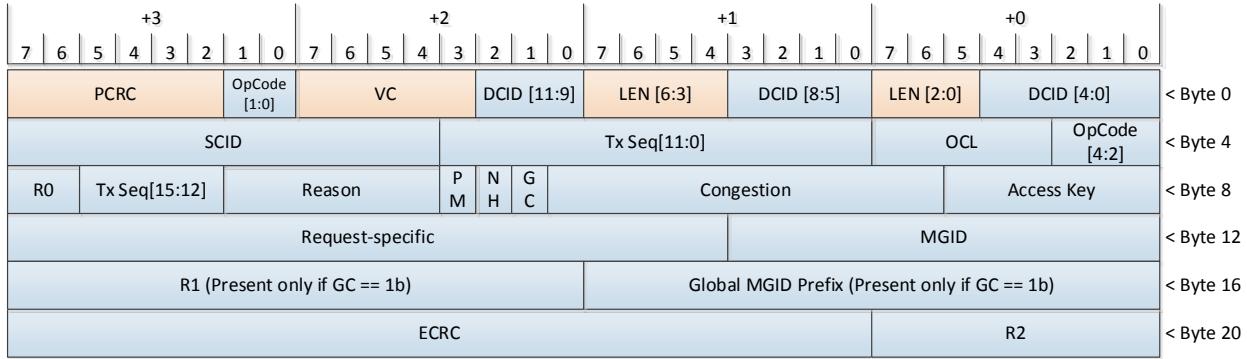


Figure 14-8: Reliable Multicast Acknowledgment Packet Format

Table 14-7: Reliable Multicast Acknowledgment Packet Fields

Field Name	Field Abbreviation	Size (bits)	Description
<b>Multicast Group ID</b>	MGID	12	If GC == 0b, then the MGID field contains the Multicast Group Identifier associated with this acknowledgment. If GC == 1b, then the MGID shall be combined with the Global Multicast Prefix to create the GMGID associated with this acknowledgment.
<b>Global Multicast Prefix</b>	-	16	If GC == 0b, then this field shall be Reserved. If GC == 1b, then this field shall contain the Global Multicast Prefix used to construct the GMGID associated with this acknowledgment.
<b>Multicast Received Seq</b>	Rx Seq	16	The last successfully received and executed multicast packet associated with the indicated multicast group by the Responding component.
<b>Reason</b>	Reason	6	Requesters shall examine the Reason field. See <i>Reason Field Encodings</i> .
<b>Request-specific</b>	-	20	This field communicates any request-specific information not covered by the Reason field.
<b>R0</b>	-	2	Reserved
<b>R1</b>	-	16	Reserved (Present only if GC == 1b)
<b>R2</b>	-	8	Reserved

The following figures illustrate reliable multicast acknowledgments.

*Reliable Multicast Write with Associated Acknowledgment* illustrates three components attached through separate point-to-point interfaces. Component A is the Requester and components B and C, the Responders. Requester A transmits two replicas of T5 and Responders B and C return ACKs to indicate success.

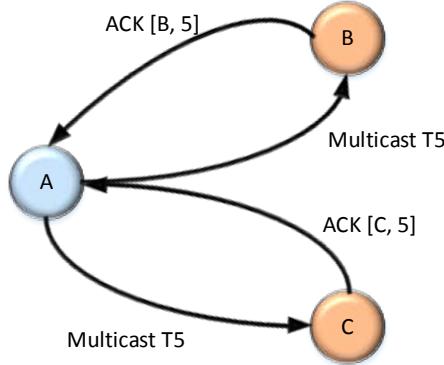


Figure 14-9: Reliable Multicast Write with Associated Acknowledgment

*Reliable Multicast through Intervening Component* illustrates five components with Requester A transmitting a multicast packet T5 which component B replicates and relays the replicas to components C, D, and E. Components C, D, and E each generate an ACK indicating success.

5

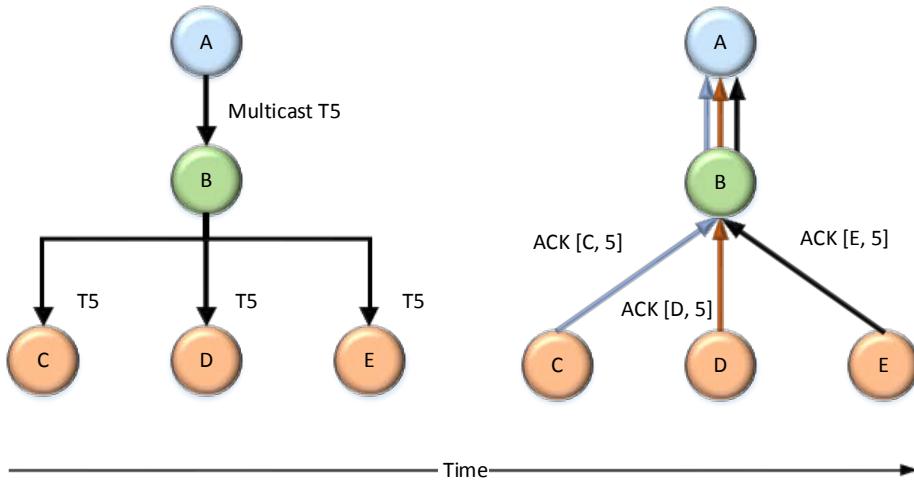


Figure 14-10: Reliable Multicast through Intervening Component

*Reliable Multicast with Lost Packet Error Recovery* illustrates five components with Requester A transmitting a multicast packet 5 which component B replicates and relays the replicas to components C, D, and E. Components C and D return ACKs and component E returns a NAK. Component A retransmits packet 5 and component B replicates and relays the replicas to components C, D, and E. Components C and D detect this is a duplicate and return an ACK without executing the duplicate. Component E executes packet 5 and returns an ACK.

10

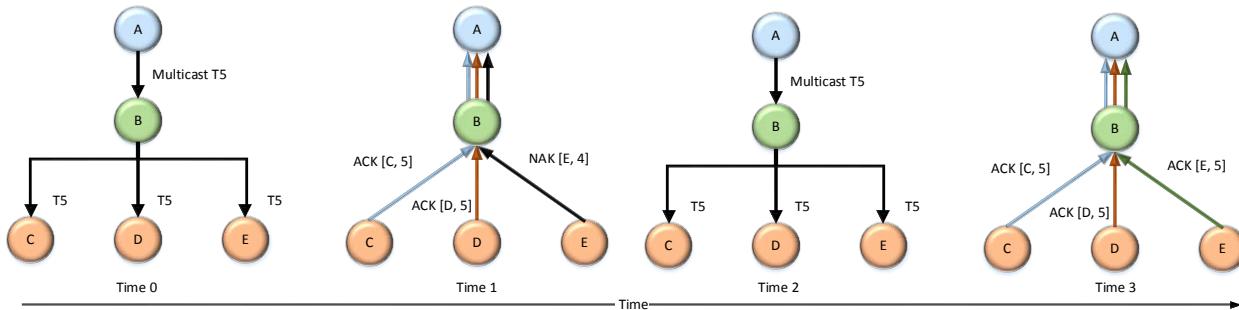


Figure 14-11: Reliable Multicast with Lost Packet Error Recovery

15

## 14.3. Multicast Group Management

This specification does not define a multicast group management protocol. Instead this section focuses on the algorithms and steps associated with each management activity and the specific configuration structures specified in section *Configuration and Management*. Multicast management is handled by a multicast management application, which for this specification is treated as an abstract software entity.

### 14.3.1. Multicast Group Create

A group shall be created prior to a multicast join:

1. An (administrative) application defines one or more packet address ranges to be associated with a multicast group operation. It also specifies Access Keys, etc.
2. The application requests the multicast management application to create the multicast group. Minimally, this requires allocating a multicast MGID / GMGID within the local subnet.
3. If multiple subnets participate in a multicast group, then the multicast management application works with router management to configure the attached hardware.

### 14.3.2. Multicast Join

The multicast management application performs the following high-level join algorithm on behalf of the target component—failure at any step results in multicast join failure:

1. Validate the multicast group has been created.
2. Verify the path from the target component’s interface (leaf) to the multicast group supports the requested multicast services.
3. Verify the target component is functionally compatible with the multicast group, e.g., the maximum packet size, reliable vs. unreliable services, etc.
4. Verify the target component may be successfully added to the multicast switching topology. Update each impacted multicast switch or router’s multicast relay table to reflect the modified multicast switching topology.
5. Enable the target component interface to begin multicast operation.

### 14.3.3. Multicast Leave

The multicast management application performs the following high-level leave algorithm on behalf of the target component—failure at any step results in management-specific error recovery.

1. Update each impacted multicast switch or router’s multicast relay table to reflect the target component’s interface is no longer a member of the multicast group.
2. Rebalance the multicast switching topology as necessary.
3. Remove the target component interface from the multicast group.

### 14.3.4. Multicast Prune

To improve subnet efficiency, the multicast management application should periodically verify that all components within a multicast group are still participating and if not, it should prune such components using the multicast leave operation.

### 14.3.5. Multicast Group Delete

When the multicast management application deems there is no need for a multicast group or there are no components participating in the group, the multicast group may be deleted.

1. All participating switch and router multicast relay tables are updated to remove all references to the associated MGID / GMGID.
2. All participating *Component Multicast Structures* are updated to remove all references to the associated MGID / GMGID.
3. The MGID / GMGID is released and no longer associated with a multicast group

# 15. Congestion and Packet Deadline

Congestion is a fabric condition where the explicit OpClass packet injection rate exceeds the packet service rate; this creates queuing delays which degrade performance. Gen-Z manages explicit OpClass packet congestion due to:

1. A traffic pattern that results in link bandwidth oversubscription, e.g., multiple source components targeting a single destination component.
2. A destination component that is unable to execute request and response packets at the provisioned bandwidth. The root cause can vary, e.g., due to cache miss rates, longer than expected memory media access, etc.

Though it is possible to mitigate (1) by overprovisioning the link(s) to the destination component, (1) will devolve into (2). To manage congestion, Gen-Z focuses on rapid congestion detection and removal by reducing packet injection rates of those components that contribute to congestion.

Packet deadline is a mechanism used to track packet age and to silently discard any packet whose Deadline has expired. Packet deadline ensures packets are discarded within a bounded period of time (required to prevent ghost packets). By bounding time, a Requester may use retransmission timers optimized for normal operating conditions rather than worst-case round-trip latencies to increase fabric utilization and efficiency.

## 15.1. Congestion Protocol Field

The following are the features and requirements associated with the Congestion Protocol field:

- Explicit OpClass packets contain a Congestion field. The Congestion field shall use the *Congestion Sub-fields Format*.
- The Explicit Congestion Notification (ECN) bit may be updated by an intermediate switch or a Responder when it detects congestion.
  - A Requester shall set ECN = 0b in all request packets prior to transmission.
  - If an intermediate switch detects congestion, then it shall set ECN = 1b.
  - Upon receipt of a request packet with ECN == 1b, the Responder shall set ECN = 1b in the corresponding response packet.
  - If a Responder detects congestion prior to response packet transmission, then it shall set ECN = 1b in the response packet.
  - Upon receipt of a response packet with ECN == 1b, the Requester may adjust its packet injection rate (see *Congestion Management*).

**Developer Note:** *Though multiple technologies have attempted to construct a single, robust formula to detect congestion that works across multiple topologies and operating environments, none have been truly successful. As a result, what constitutes congestion in Gen-Z components is vendor-defined. Though vendor-defined, a quite common technique is to declare congestion and set ECN = 1b in any packet within a queue that becomes full or surpasses an implementation-specific threshold.*

- The Deadline field is used to track a packet's age.

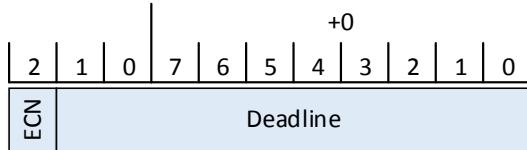


Figure 15-1: Congestion Sub-fields Format

Table 15-1: Congestion Sub-fields

Field Name	Size (bits)	Description
ECN	1	The ECN bit indicates if congestion was detected. 0b—No Congestion Detected 1b—Congestion Detected
Deadline	10	The Deadline field shall be decremented as a packet progresses through the fabric. If Deadline == 0x0, then the packet shall not be transmitted and shall be silently discarded.

## 15.2. Deadline Semantics

The following specifies how the Deadline sub-field is updated as a packet is processed by a given component.

- Requesters and Responders shall set the Deadline field prior to packet transmission.
  - If a Requester or a Responder does not support the *Component PA (Peer Attribute) Structure*, then management shall configure a Requester's *Core Structure* LL Request Deadline and NLL Request Deadline with the same value and a Responder's *Core Structure* LL Response Deadline and NLL Response Deadline with the same value.
  - If a request packet is associated with a low-latency domain or the Requester does not support the *Component PA (Peer Attribute) Structure*, then a Requester shall set the Deadline field to the *Core Structure* LL Request Deadline. To configure the LL Request Deadline, management calculates the one-way latency for any path on any VC to any Responder participating in a low-latency domain. Management then sets:
    - LL Request Deadline = One-way Latency DIV Deadline Tick
    - If One-way Latency Mod Deadline Tick > 0, then LL Request Deadline++
  - If a request packet is not associated with a low-latency domain, then a Requester shall set the Deadline field to the *Core Structure* NLL Request Deadline. To configure the NLL Request Deadline, management calculates the one-way latency for any path on any VC to any Responder participating in a non-low-latency domain. Management then sets:
    - NLL Request Deadline = One-way Latency DIV Deadline Tick
    - If One-way Latency Mod Deadline Tick > 0, then NLL Request Deadline++
  - If the Requester is associated with a low-latency domain or the Responder does not support the *Component PA (Peer Attribute) Structure*, then a Responder shall set the Deadline field to the *Core Structure* LL Response Deadline. To configure the LL Response Deadline, management calculates the one-way latency for any path on any VC to any communicating Requester. Management then sets:
    - LL Response Deadline = One-way Latency DIV Deadline Tick

- If One-way Latency Mod Deadline Tick > 0, then LL Response Deadline++
- If the Requester is associated with a non-low-latency domain, then a Responder shall set the Deadline field to the *Core Structure* NLL Response Deadline. To configure the NLL Response Deadline, management calculates the one-way latency for any path on any VC to any communicating Requester. Management then sets:
  - NLL Response Deadline = One-way Latency DIV Deadline Tick
  - If One-way Latency Mod Deadline Tick > 0, then NLL Response Deadline++
- Each Deadline represents the number of Deadline Ticks (see *Core Structure*) until the packet expires. Upon reaching zero, the packet shall be silently discarded.
- If a TR is interposed between a Requester and a Responder, then the configured Deadline values need to be increased to account for the TR processing and the transparent crossing of the other subnet.
- When any component interface schedules a packet for transmission, the component calculates the explicit OpClass packet's transmit time (T-Time) using some of the egress interface's  $VC_k$  transmit queue depth,  $VC_k$  service rate, available  $VC_k$  flow-control credits, available transmit bandwidth, link-local packet transmit queue depth, Path Propagation Time (see *Interface Structure*), etc.
- Conceptually, a component should take the following steps to determine the amount to decrement the Deadline field and whether it is permitted to transmit an explicit OpClass packet:
  - $DQ = ((\text{Component packet processing time} + \text{T-Time}) \text{ DIV Deadline Tick})$
  - If  $(\text{T-Time Mod Deadline Tick}) > 0$ , then  $DQ++$
  - If  $DQ > 0$ , then  $\text{Deadline} = \text{Deadline} - DQ$
  - If  $\text{Deadline} \leq 0$ , then silently discard the packet
- To avoid implementation complexity (e.g., extremely accurate fine-grain time calculations), a component may round the amount to decrement the Deadline value up such that a packet is pessimistically discarded, i.e., the packet is silently discarded earlier but never later than the absolute time ( $\text{Deadline} * \text{Deadline Tick}$ ) ns.
- When calculating the amount to decrement the Deadline
- The amount to decrement the Deadline
- *Explicit OpClass Packet Flow and Deadline Updates* illustrates an example switch topology and how the Deadline field is updated during a request packet-response packet exchange.
  - The Requester generates a request packet and sets the Deadline sub-field to either the LL Request Deadline or the NLL Request Deadline.
  - Prior to transmission, the Requester calculates the expected transmission time and decrements the Deadline field and starts the corresponding retransmission timer.
  - If Deadline is non-zero, then the Requester transmits the packet. Transmission and propagation across a given path will consume a non-zero amount of time. Depending upon the physical path length, this time should be accounted for in T-Time.
  - The first switch receives and relays the packet to the egress interface. The switch calculates packet residency and T-Time and updates the Deadline sub-field. If non-zero, then the switch transmits the packet to the second switch.
  - The second switch performs the same steps as the first and if Deadline is non-zero, it transmits the packet to the Responder.
  - The Responder receives and validates the request packet and decrements the request packet's Deadline sub-field. If non-zero, then the Responder executes the request packet. If the request packet's execution time is less than ( $\text{Responder Deadline} * \text{Deadline Tick}$ ), then the Responder generates a response packet and copies the Response Deadline into

the Deadline sub-field. The Responder calculates T-time and updates the Deadline sub-field; if non-zero, then the Responder transmits the packet to the switch.

- The two switches perform the same steps as they did for the request packet and transmit the packet to the Requester.
- The Requester receives and validates the response packet. If the time to perform these steps is less than the response packet's (Deadline \* Deadline Tick), then the Requester removes the request packet from the corresponding retransmission queue and executes the response packet.

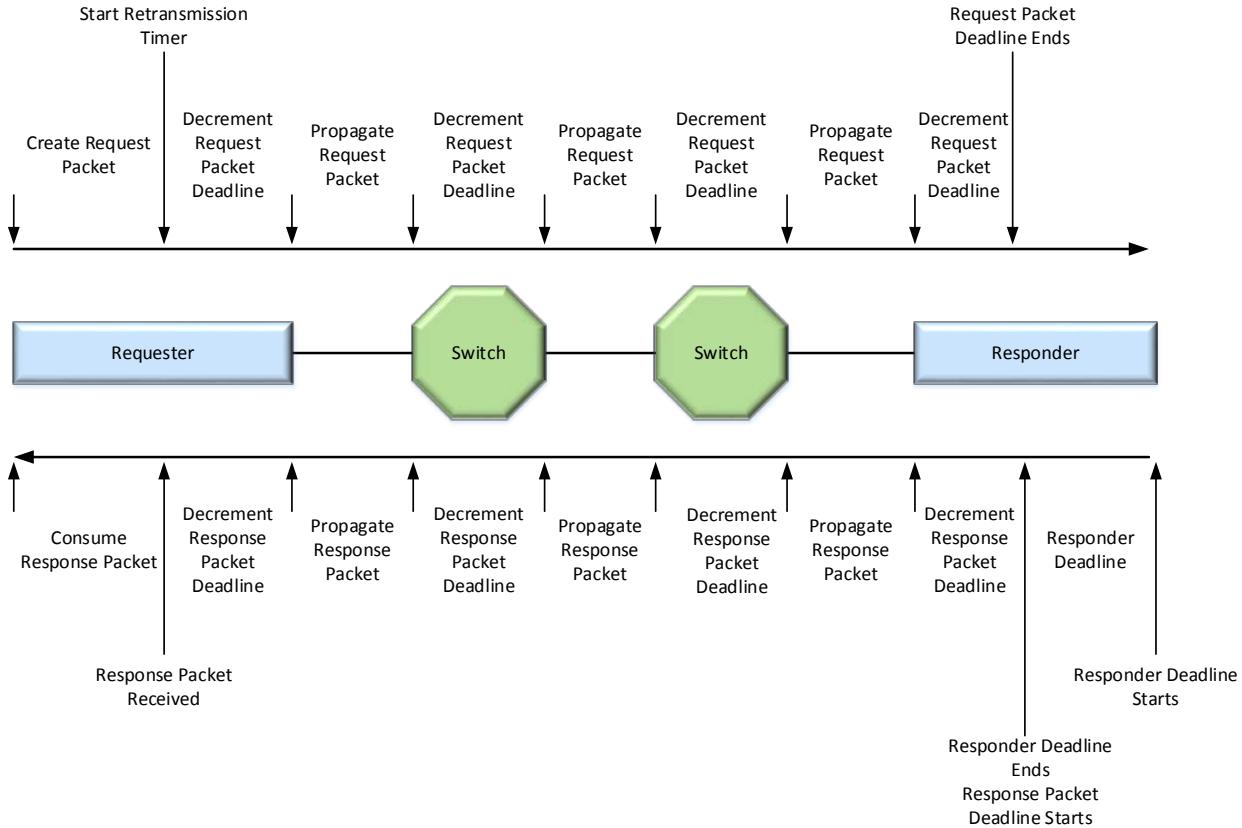


Figure 15-2: Explicit OpClass Packet Flow and Deadline Updates

- If physical layer retraining or *Link Resynchronization* is initiated prior to packet transmission, then the component shall recalculate T-Time and update the Deadline sub-field.
- If *Link-level Reliability (LLR)* is enabled, then upon initiating link-level packet retransmission, the component shall recalculate T-time for each retransmitted packet and update the Deadline field. One or more end-to-end packets could be silently discarded instead of retransmitted.
- If Responder load is high and the expected time to validate and execute a request packet will exceed the (Responder Deadline \* Deadline Tick), then the Responder shall silently discard the request packet. If applicable, then the Responder shall transmit a *Responder Not Ready (RNR) NAK*.
- Upon receipt of a request packet and decrement of its Deadline sub-field, if the request packet's Deadline value is non-zero, then the Responder may use the remaining request packet's Deadline to transparently increment the Responder Deadline or the Response Deadline. Similarly, if the Responder Deadline is non-zero subsequent to executing the request packet and generating a response packet, then the Responder may increment the Deadline field in the response packet.

Though this does increase the time available to generate a response packet and / or transmit it to the Requester, it shall not change the maximum time permitted for the request packet-response packet exchange.

**Developer Note:** Consider the following when setting the Deadline values in Requester and Responder components:

- Solutions composed of homogeneous components, switches, links, and signaling rates, and operate under uniform packet traffic patterns can use the same Deadline value for request and response packets. Similarly, simple topologies where Responders are equidistant to all Requesters (e.g., a single enclosure) can use the same Deadline value for request and response packets. If neither is true, then Deadline values could differ.
- Each Deadline value represents the one-way latency across the topology before the packet is dropped. Though smaller Deadline values enable more aggressive retransmission timers, too small a value could lead to excessive packet drops.
- Consider a topology's worst-case BER (Bit Error Ratio) to determine the probability of packet loss due to a transient error. For example, if the BER is  $10^{-15}$ , then packet loss due to transient errors will be very infrequent and the associated performance impacts negligible. If this is true, then an aggressive retransmission timer to compensate for transient errors is unwarranted and a larger Deadline value can be used.
  - Developers are strongly cautioned against assuming the BER of a single link in a single enclosure or package extends to a multi-link, multi-connector, or rack-scale solution or that the BER is largely unchanged across all signaling rates or in any physical environment. Developers are strongly encouraged to characterize the BER of each solution to properly configure retransmission timers and Deadline values.
- Consider the probability of congestion events based on the topology and packet traffic patterns. The higher the probability, the larger the Deadline values to reduce an unacceptable rate of packet drops and associated performance loss.
- To characterize a solution, developers can take the following steps:
  - Start with very large Deadline values, e.g., (Deadline \* Deadline Tick) is 10-100 times the worst-case one-way latency for any communicating pair of components.
  - Place maximum load on the fabric using a uniform packet traffic pattern or one that reflects the worst-case application traffic pattern. Measure packet loss and adjust the Deadline values downward until the packet drop rate reaches an unacceptable level. Ideally, packet loss is only 1-in- $10^4$  or 1-in- $10^5$  packets due to a Deadline reaching zero.

### 15.3. Congestion and Packet Injection Rate

Requesters use a packet injection delay table (PIDT) as specified in the *Congestion Management Structure* to control packet injection rate. A Requester increments or decrements the current index into the PIDT to reduce or increase the packet injection rate based on the receipt or detection of one of the following congestion events.

- Timeout Event—if a request packet is retransmitted two or more times, then this indicates packet loss due to Deadline expiration as a result of congestion. The Requester shall increment the

corresponding PIDT index to reduce the packet injection rate. If the Requester supports *Link-level Reliability (LLR)*, then any packet loss is likely caused by congestion and the Requester should increment the corresponding PIDT index to reduce the packet injection rate.

- Explicit Congestion Notification—If a Requester receives a response packet with ECN == 1b, then the Requester should increment the corresponding PIDT index to reduce the packet injection rate.
- High-latency Event—a Requester may calculate the expected time required to execute a request packet-response packet exchange under reasonable fabric load. If the Requester detects an exchange that significantly exceeds the time, then this should be treated as a congestion event. If multiple such events occur within a vendor-defined multiple of the *Congestion Management Structure's Congestion Sampling Window*, then the Requester should increment the corresponding PIDT index to reduce the packet injection rate.

5

10

# 16. Security

A topology can contain malicious components that attack and disrupt non-malicious component operations. *Attack Pathologies* illustrates possible attack pathologies. ‘In situ attacks’ includes crocodile clips or the equivalent on the subnet, as well as attacks mounted at endpoints. Broadly the flow is from left to right, with the end of the flow being the manifestation of the attack: a malicious component or unauthorized component removal. The purpose of security is to detect, protect and recover from the effects of a malicious component or unauthorized component removal.

The primary effects of unauthorized component removal are lost data availability and confidentiality. These effects can be addressed by a combination of data replication, error recovery, and encryption services which operate above the Gen-Z protocol and, therefore, are outside of this specification’s scope.

This rest of this section is focused on mitigation—the processes and mechanisms put in place to detect and protect against malicious components.

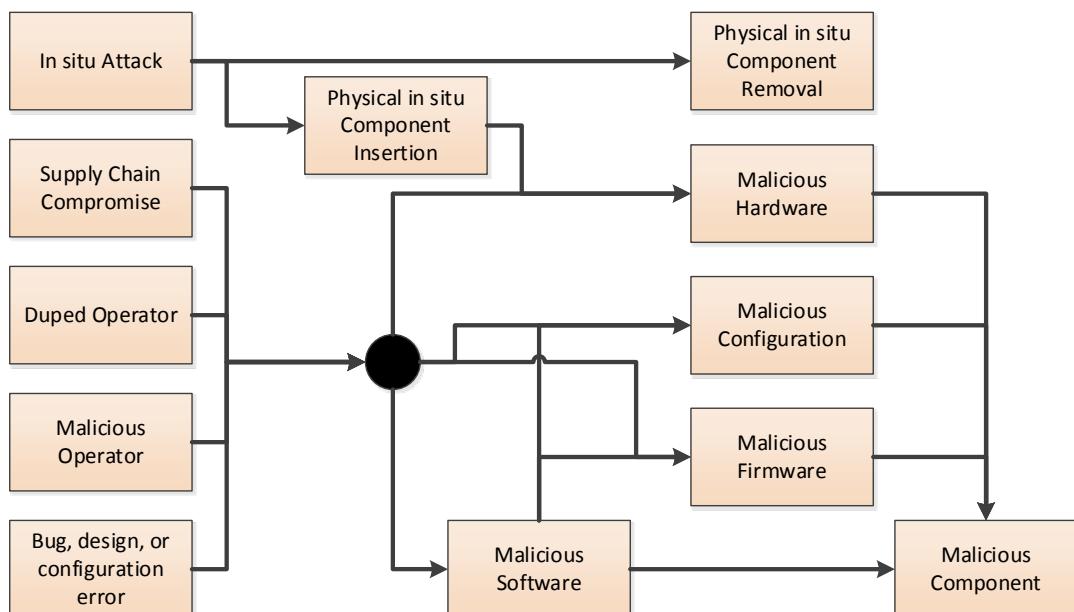


Figure 16-1: Attack Pathologies

## 16.1. Malicious Component Threats

The attacks that can be mounted by a malicious component can be broken down as follows:

- Attacks on packets traversing the subnet from other components:
  - Packet eavesdropping
  - Packet destruction
  - Packet modification
- Denial of service attacks:
  - Destruction of in-flight packets by an intermediate component
  - Extreme packet injection rates to cause resource exhaustion and congestion collapse

- Data destruction—Request acknowledgment without successful execution or acknowledgment of deliberately modified data
- Resource exhaustion by failure to send expected packets. This can occur if two components are interlocked via a specific packet exchange sequence and one prevents the sequence from progressing causing resources to be consumed for long time periods.
- Unauthorized packet injection including, but not limited to, unauthorized attempts to read or write the Data Space or Control Space of other components.
  - A variant of this is the replay attack. A replay attack is when the malicious component captures a legitimate packet, and at a later time injects it back into the subnet. Without mitigation, the receiving component has no way to distinguish the injected packet from a correct packet and may perform an unauthorized action such as writing to an address space overwriting any changes that have been made between receipt of the legitimate packet and the unauthorized replay.
- Precision Time manipulation to force time forwards or backwards.

## 16.2. Malicious Component Threat Mitigation

Malicious component threats can be mitigated as follows:

- Attacks on packets traversing the subnet:
  - Data encryption to protect against eavesdropping
  - Tight timeout domains combined with immediate response scheduling upon request execution completion to detect packet destruction.
  - Cryptographically-secure message authentication to detect malicious modification.
- Denial of service attacks:
  - Packet destruction detection as described above.
  - Extreme packet injection rate protection through:
    - Source packet injection rate controls
    - Component and switch adaptive routing controls
    - Control OpClass packet filtering
    - Packet filtering via the *Component Destination Table Structure*
  - Data destruction protection through:
    - Data replication across two or more components
    - High-level data authentication and verification (outside of this specification's scope)
    - Component, Data Space, and Control Space access control via *Access Keys* and *Region Key (R-Key)*
  - Resource exhaustion protection through:
    - Proper implementation and enforcement of the end-to-end packet protocol. In general, idempotent request packets do not require the Responder to maintain state therefore, resources are released upon the execution of a given request. Non-idempotent request packets require resources to be explicitly released. Responders use a failsafe timer to ensure all resources are eventually released.
    - Proper configuration and execution of packet deadline (see *Congestion and Packet Deadline*) to release resources.

- Unauthorized packet injection detection through:
  - Cryptographically-secure message authentication
  - Interface Access Key use and validation
  - Page R-Key use and validation
  - Replay attack detection using a packet uniqueness property built on top of cryptographically-secure message authentication
- Protection of Precision Time request and response packets through use of cryptographically-secure authentication, e.g. a HMAC within the *Explicit OpClass Next Header Field*. Once a component pair have established a common Master Time, replay attacks may be mitigated using the Precision Time ART (a Responder may copy the updated Master Time to the ART field within the Next Header). This provides end-to-end validation of Master Time.

From the above, the following fundamental mitigations can be identified:

1. Thorough and complete packet validation per *End-to-End Packet Reliability / Validation*.
2. Cryptographically-secure packet authentication
3. Cryptographically-secure determination of packet uniqueness
4. Congestion and Access Key and R-Key-based access control
5. Encryption, authentication, data replication, and some forms of access control (non-Access Keys or R-Keys).

The following sub-sections focus on the fundamental mitigations within this specification leaving all others as outside of this specification's scope. Security management is configured through the *Component Security Structure*.

### 16.3. Additional Packet Validation Requirements

This document specifies the following additional packet validations:

- Ingress and Egress Access Key
- Requester / Responder Page Region Key (R-Key)
- Requester / Responder Component Destination Table
- Switch Packet Relay Table

If Access Key-based validation is required, components should support explicit OpClasses (packets that contain an Access Key) and the *Component Destination Table Structure*.

In order to configure error handling and notify management via an *Unsolicited Event (UE) Packet* of validation failures, components should support the *Component Error and Signal Event Structure*.

### 16.4. Cryptographically Secure Authentication

Components may use the explicit OpClass Next Header for cryptographic authentication. The source component may insert a Hash-based Message Authentication (HMAC) Code and an Anti-Replay Tag (ART) into the *Explicit OpClass Next Header Field*. The destination component may use the Hash-based Message Authentication Code and ART to determine that the received packet is authentic and unique—not a replay.

### 16.4.1. HMAC Code Calculation

This section specifies how the HMAC shall be calculated and inserted into the *Explicit OpClass Next Header Field*. Requesters shall calculate a new HMAC for each request packet (new or retransmitted). Responders shall calculate a new HMAC for each response packet.

5 Calculate HMAC( $K$ , text) and truncate the output to 64 bits according to *FIPS198-1* with:

- HMAC: The secure hash function is any of the supported hash functions listed in *Component Security Structure Fields*.
- $K$ : The Transaction Integrity Key (TIK). A TIK is a  $k$ -bit secret and is shared between the source component and destination component.  $k$  is the length of a given supported has function. For example,  $k$  is 224 bits for a 224-bit hash function, 256 bits for a 256-bit hash function, and so forth.
- text: the data string shall start with bit 0, byte 0 of a packet and progresses until the end of the packet with the exception of the following fields which shall be excluded: PCRC, VC, Congestion, ECRC, Next Header[127:64].

10 Once calculated, the HMAC[63:0] shall be placed in Next Header[127:64] starting with HMAC[0] to Next Header[64] and progressing through HMAC[63] to Next Header[127] (inclusive).

#### 16.4.1.1. HMAC Code Validation

If the Next Header and the HMAC Next Header capability are enabled, the following shall apply to all received explicit OpClass packets:

- The destination component shall perform HMAC Code validation as part of its packet validation process.
- The destination component shall dynamically generate a HMAC Code using the steps specified in *HMAC Code Calculation* applied to the received packet and the TIK which is shared with the source component.
- The destination component shall compare the dynamically-generated HMAC Code with the value in Next Header[127:64]. If they match, then the packet is authenticated. If an invalid HMAC Code is detected, then:
  - If the *Component Error and Signal Event Structure* is supported, then the component takes the steps configured for a Security Error. If the component is configured to generate an *Unsolicited Event (UE) Packet* and multiple Security Errors are detected, then to prevent a management denial of service situation, a single *Unsolicited Event (UE) Packet* shall be generated per observed time period. The observed time period is implementation-specific.
  - The component shall silently discard the packet.

## 16.5. Anti-Replay Tag (ART)

35 A component shall construct an ART using one of three methods:

- Incrementing a per TIK unique 64-bit sequence number prior to packet (re-) transmission
- Copying the component's current Master Time prior to packet (re-) transmission
- Inserting a Null ART—a Null ART disables anti-replay validation.

If an ART is present, ART[63:0] shall be placed in Next Header[63:0] starting with ART[0] to Next Header[0] and progressing through ART[63] to Next Header[63] (inclusive). If a Null ART is enabled, Next Header[63:0] shall be Reserved.

5 Communicating components shall support the same ART method(s) and have the same method enabled. At a minimum, all components which support an ART shall support the Null ART.

### 16.5.1. Sequence Number ART

The following are the features and requirements associated with a Sequence Number ART (SNA):

- Requester and Responder components shall provision one sequence number per shared TIK.
  - The Requester shall generate a new sequence number associated with a given shared TIK and place it in the Request's *Explicit OpClass Next Header Field* as described in *Anti-Replay Tag (ART)*.
    - A sequence number shall be associated only with a single request.
    - A retransmitted request shall contain a new sequence number.
  - The Responder shall reflect the Request's SNA and place it in the Response's *Explicit OpClass Next Header Field* as described in *Anti-Replay Tag (ART)*.
- The sequence number used as a SNA is a 64-bit unsigned integer that uniquely identifies a request packet and its corresponding response packet, and may be used to detect packet replay attacks.
- A sequence number shall be monotonically increased (1, 2, 3, etc.).
  - The rate at which a sequence number will wrap depends upon the Request injection rate for a given shared TIK and the available bandwidth. For example, if 32-byte packets are continuously injected on a 2.56 Tbps link and starting at sequence number 1, then the sequence number will wrap in 58.46 years.
  - Prior to a given sequence number wrap, the corresponding TIK shall be replaced.
- Prior to transmitting the first request associated with a given TIK, the Requester shall set the sequence number to 1.
- Prior to receiving the first request associated with a given TIK, each Responder shall set the NESN (Next Expected Sequence Number) = 1.
- As part of request packet validation by the Responder, the SNA is compared to the Responder's NESN.
  - If the  $SNA \geq NESN$ , then the ART is valid and the request shall be executed upon successfully completing all other packet validation steps, else the ART is invalid and packet validation fails.
    - $NESN = SNA + 1$ .
    - If a TIK is shared by more than two components, or if a prior request suffers a transient error or is lost or discarded within the topology, then the Responder may observe non-sequential SNA values.
  - If multiple paths are actively used between a Requester and a Responder, then request packets may arrive out of the transmitted order due disparate path operating conditions, e.g., congestion. To avoid excessive packet Silent Discards, a Responder may take the following additional out-of-order actions:
    - If  $SNA = NESN$ , then in-order packet delivery occurred and no out-of-order actions are required.
    - If  $SNA > NESN$ , then out-of-order packet delivery could have occurred.

- 5           • Responder sets Highest Out-of-Order Sequence Number (HOSN) = SNA but does not advance NESN.
- 10           • Responder starts a *Core Structure* OOOTO timer that represents the configured window in which out-of-order request packets can arrive and still be validated and executed.
- 15           • If subsequent request packets arrive before the timer expires with  $SNA \geq NESN$ , then the request packets shall be treated as having a valid ART and packet validation proceeds.
- 20           • If the request packet's  $SNA > HOSN$  then  $HOSN = SNA$ 
  - When the timer expires,  $NESN = HOSN + 1$ . All subsequent request packet SNA values shall be compared to the updated NESN.
- 25           • As part of response packet validation by the Requester, the Sequence Number ART is compared to the request packet's original Sequence Number ART. If they are not equal, then an invalid ART has been detected.
- 30           • If the ART is required to be updated prior to each request packet's transmission, then the component shall update the ART prior to retransmission.
- 35           • If an invalid ART is detected, then:
  - o If the *Component Error and Signal Event Structure* is supported, then the component takes the steps configured for a Security Error. If the component is configured to generate an *Unsolicited Event (UE) Packet* and multiple Security Errors are detected, then to prevent a management denial of service situation, a single *Unsolicited Event (UE) Packet* shall be generated per observed time period. The observed time period is implementation-specific.
  - o The component shall silently discard the packet.

### 25       16.5.2. Precision Time ART

The following are the features and requirements associated with a *Precision Time* ART:

- 30           • All communicating components shall participate in the same Precision Time Domain.
- 35           • Each component shall obtain and maintain Master Time as specified in *Precision Time*.
- 40           • Just prior to transmission or retransmission, the present Master Time shall be placed in the packet's *Explicit OpClass Next Header Field* as described in *Anti-Replay Tag (ART)*.
- 45           • TIKs shall be replaced prior to Master Time wraps. *Time to Master Time Wrap* illustrates when Master Time will wrap as a function of Grandmaster Time Component (GTC) time granularity.
- 50           • As part of packet validation, the Precision Time ART is compared to the receiving component's current Master Time. If  $X$  = the packet's Precision Time ART,  $Y$  = Receiving Component's Master Time, and  $Z$  = Precision Time Window, then the Precision Time ART is valid if  $X \leq (Y + Z)$ . If an invalid Precision Time ART is detected, then:
  - o If the *Component Error and Signal Event Structure* is supported, then the component takes the steps configured for a Security Error. If the component is configured to generate an *Unsolicited Event (UE) Packet* and multiple Security Errors are detected, then to prevent a management denial of service situation, a single *Unsolicited Event (UE) Packet* shall be generated per observed time period. The observed time period is implementation-specific.
  - o The component shall silently discard the packet.

## 16.6. Mitigating In Situ Insertion

Mitigation may involve one or more of the following:

- A component may support the *Component Security Structure*. This structure may contain one or more security certificates. A security certificate can be used to ascertain whether the component can be trusted, e.g., determine if it authentic or a counterfeit, or determine if has been subverted.
  - Components that support *Out-of-band Management* may directly access this structure.
  - Components that support *In-band Management* may access this structure through *In-band Management* read and write request packets.
- A switch may be configured to restrict packet relay from a directly-attached component until management has ascertained if the component can be trusted. See *Switches*.
- Although the specific protocol is outside of this specification's scope, management may use a challenge-response protocol to force a newly discovered component to authenticate its public / private key pair using standard cryptographic protocols. These protocols may be exchanged using Core 64 Read and Write request packets or through explicit OpClass vendor-defined packets associated with an industry-specified UUID. The public key may be accessed through *Out-of-band Management* and stored in the *Component Security Structure* (one of the stored certificates) or a separate NVM resource.
- A Component Nonce is used to determine if the directly-attached peer component was transparently replaced during a low-power event. If supported, then:
  - Management shall generate a unique, non-zero, random 64-bit value for each component within a topology. Management shall write this unique value to each component's *Core Structure* Component Nonce field.
    - There shall be one Component Nonce value per component.
    - The Component Nonce field shall be a volatile, write-only register
    - The Component Nonce field shall be reset to zero whenever a *Full Component Reset* or a *Warm Component Reset* is initiated.
  - For each pair of directly connected components, management shall write the peer's Component Nonce value into the *Interface Structure* Peer Nonce field of each interface that directly connects the two components.
    - The Peer Nonce field shall be a volatile, write-only register
    - The Peer Nonce field shall be reset to zero whenever a *Full Interface Reset* or a *Warm Interface Reset* is initiated, the interface transitions to L-Down, or the link transitions to L-Down.
    - The Peer Nonce field in each interface that connects a pair of directly-attached components shall be configured with the peer component's Component Nonce value. For example, for each component 0 interface<sub>K</sub> that connects to component 1, component 0 interface<sub>K</sub> Peer Nonce = component 1 Component Nonce. Similarly, component 1 interface<sub>K</sub> Peer Nonce = component 0 Component Nonce.
    - Interfaces that are directly connected to different peer components shall be configured with the Component Nonce value of the respective peer component.
  - If *Interface I-CAP 1 Control* Nonce Enable == 1b, then whenever the link transitions to L-Up or L-Up-LP from the L-LP state, then the transmitter shall initiate a Nonce Notification *Link CTL* -Link ACK *Link CTL* packet exchange.

## 16.7. Transaction Integrity Keys (TIK) Management

Assume an out-of-band mechanism is used to establish a 256-bit AES key shared between a component and a Key Distribution Server (KDS) that is a trusted manager. The KDS shares a different AES key with each unique component.

- 5 Two components may establish a shared TIK from a Key Distribution Server (KDS) as illustrated in *Establishing a Shared TIK using Key Distribution Server*. This figure illustrates:
- o Component 1 (C1) containing a 256-bit AES Key K1 shared with the KDS.
  - o Component 2 (C2) containing a 256-bit AES Key K2 shared with the KDS.
  - o K1 and K2 are established with the KDS using a key exchange protocol through *Out-of-band Management* or through in-band challenge / response protocol *Mitigating In Situ Insertion*.
- 10

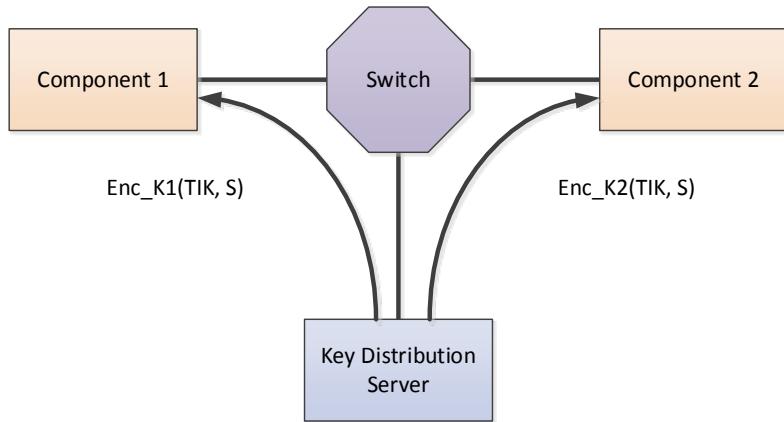


Figure 16-2: Establishing a Shared TIK using Key Distribution Server

To establish a shared TIK using AES in the Counter mode, the following steps are taken:

- 15
1. KDS generates a TIK. Multiple mechanisms exist including the use of a Key Derivation Function (KDF) which requires the KDS to have a master key for the KDF.
  2. The KDS distributes the TIK to C1 and C2 by encrypting it with the shared AES keys. Let  $K_i$  be the 256-bit AES key shared between the KDS and  $C_i$  for  $i = 1$  or  $2$ . Encryption protects the TIK during transmission and while stored within the *Component Security Structure*. For performance reasons, a TIK may be decrypted and with the result stored in an inaccessible resource, i.e., no in-band or out-of-band access.
    - a. Let  $k$  be the length of a TIK in bits and  $h$  be  $k/128$  when  $k = 256, 384$  or  $512$  or  $k/112$  when  $k = 224$ ; as a result  $h = 2, 3$  or  $4$ . Split the  $k$ -bit TIK to  $h$  blocks, writing as  $T_1 \dots T_h$ , such that each block has  $j = k/h$  bits.
    - b. Let  $S$  be a 32-bit unsigned integer (sequence number) managed by the KDS. The KDS shall increment  $S$  by  $1 \bmod (2^{32} - 1)$  each time it replaces the TIK. This value is used to create  $h$  counter values, denoted by  $CTR_1, \dots, CTR_h$ , in the encryption operation below.
    - c. The cipher text  $Enc_{K_i}(TIK, S) = C_1 || \dots || C_h$  is computed by performing the following steps:
      - i.  $CTR_1 = S || PAD$ ;  $PAD$  is 96 bits in length with all bits cleared.
      - ii. For  $m = 1$  to  $h$ ,
- 20
- 25
- 30

- 5
1.  $Em = j \sim ENC\_AES(Ki, CTRm)$ . Here  $j \sim X$  is the string obtained from the string  $X$  by taking the leftmost  $j$  bits of  $X$  (standard ISO notation). The operation of  $j \sim X$  is only needed when  $k = 224$  and  $j = 112$ . In the other cases ( $k = 256, 384$  or  $512$ ),  $j = 128$  which is the same as the length of the AES block.
  2.  $Ym = Tm \oplus Em$ ,
  3.  $CTR(m+1) = (CTRm + 1) \bmod 2^{128}$ . This step is not performed if  $m = h$ .
- 10
- iii.  $Enc\_Ki(TIK, S) = Y1 || \dots || Yh$ .
  - d.  $Enc\_K1(TIK, S)$  is written to the TIK table entry within C1's *Component Security Structure* corresponding to Component 2's DCID.
  - e.  $Enc\_K2(TIK, S)$  is written to the TIK table entry within C2's *Component Security Structure* corresponding to Component 1's DCID.
- 15
3. For authentication of  $Enc\_K1(TIK, S)$  and  $Enc\_K2(TIK, S)$  the KDS writes  $S$  into Next Header[31:0] starting with  $S[0]$  into Next Header[0] and progressing through  $S[31]$  to Next Header[31]. Next Header[63:32] are set to 0. The HMAC is calculated following the algorithm in *HMAC Code Calculation*, the payload of the packet is  $Enc\_Ki(TIK, S)$ . The HMAC key is the TIK value being distributed to C1 and C2. HMAC[63:0] shall be placed in Next Header[127:64] starting with HMAC[0] to Next Header[64] and progressing through HMAC[63] to Next Header[127] (inclusive).

20

After receiving the packet, the component  $Ci$  for  $i = 1$  or  $2$  decrypts the TIK from  $Enc\_Ki(TIK, S)$  and then uses it to check HMAC validation. Since using the Counter mode, the decryption process is identical to the encryption one.

25

A TIK may be shared between two or more components. HMAC verification validates that the packet originated from one of the components that is in possession of the TIK, but not a specific component.

- o If shared between two components, then the identified threats are fully mitigated.
- o If shared between more than two components, then a destination component can validate if the source component is a member of the component group sharing the key, but it cannot ascertain which of the components within the group. The identified threats are partially mitigated.
- o Other key sharing schemes are possible which partially mitigate the identified threats.

30

A KDS may distribute keys during different component lifecycle phases.

1. Key distribution may occur during the component manufacturing process and are not replaced during normal use. The KDS resides within the factory and all components are under factory control. As a result, the factory determines what can be attached to a given subnet.
2. Key distribution may occur during the component installation process. The KDS is not subnet-attached and the keys are not replaced during normal use. The KDS is 'owned' and operated by the installer who configures the keys prior to attaching a component to a subnet. As a result, the installer determines what can be attached to a given subnet.
3. Key distribution may occur during normal component operation. The KDS is subnet-attached and the keys may be replaced during normal use. As a result, whoever controls the KDS determines what components can be attached to a given subnet.

## 17. Strong Ordering Domain (SOD)

A strong ordering domain (SOD) is a logical construct used to delineate individual packets flows, provide Reliable Delivery, and to enforce strong packet ordering. *Single SOD Example* illustrates an example SOD between a Requester and a Responder. Conceptually, a SOD is a pipe through which datagram request and response packets are exchanged in the order they are posted. Each SOD uses a separate sequence number space in place of a Tag field to identify request packets and to correlate response packets.

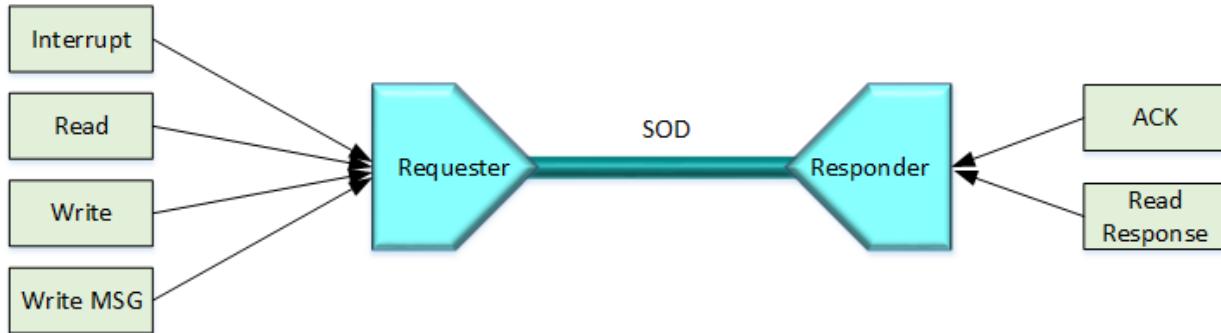


Figure 17-1: Single SOD Example

SODs may be established between multiple components as illustrated in *Multiple SOD Example (A)*. Further, multiple SODs may be established between each component pair as illustrated in (B). SODs are delineated by a combination of the source and destination CIDs / GIDs and source and destination SOD identifiers, i.e., [SCID | SGCID, SSODID, DCID | DGCID, DSODID].

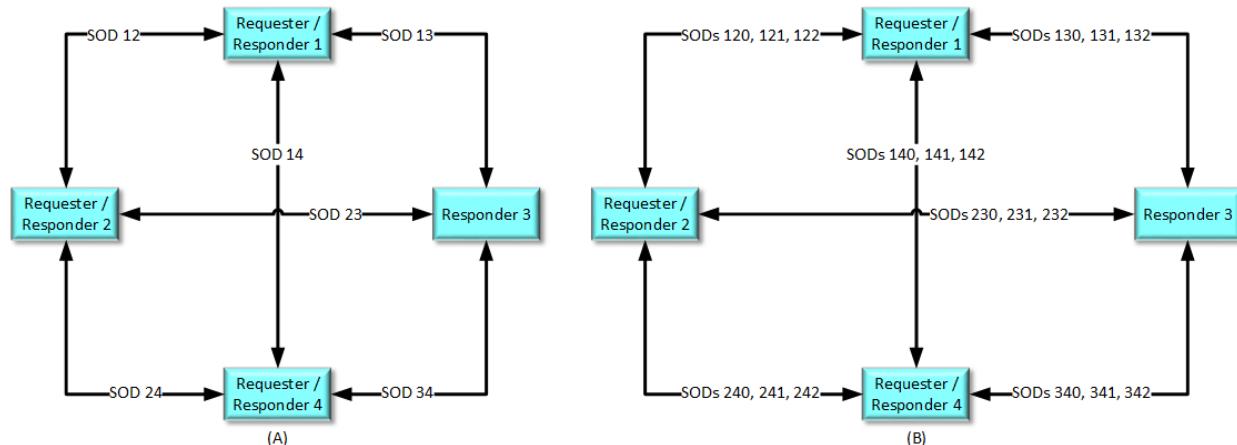


Figure 17-2: Multiple SOD Example

Applications that support request and response contexts may use SODs as illustrated in *Request and Response Context IDs using SODs*. Contexts may communicate through a single SOD (A) or across multiple SODs (B) between the same or different component pairs. Software is responsible for enforcing ordering when communicating across multiple SODs, e.g., by use a fence bit to inform hardware to stall until all prior request packets have successfully completed prior to transmitting the present request packet.

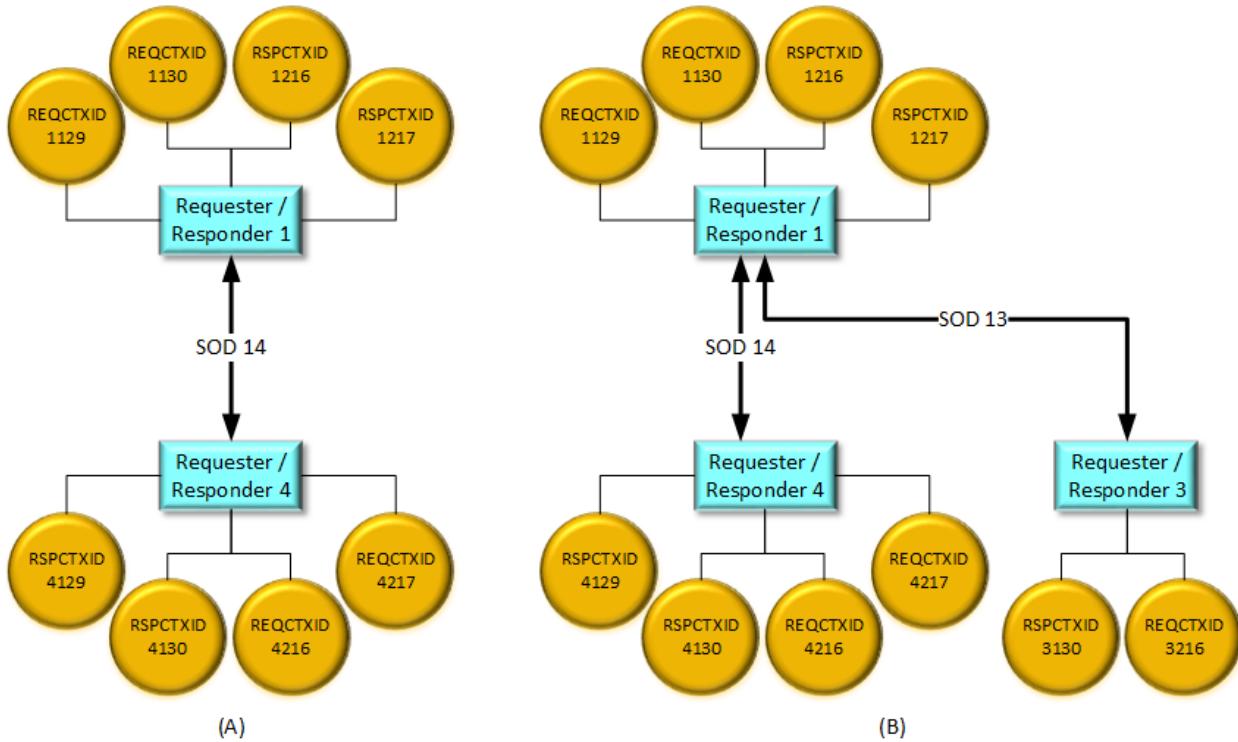


Figure 17-3: Request and Response Context IDs using SODs

Applications generate SOD request packets using the same or similar techniques used for non-SOD communications, e.g., by accessing Responder resources through a Requester ZMMU or by generating contexts-based request packets as illustrated in *Generating SOD Request Packets*.

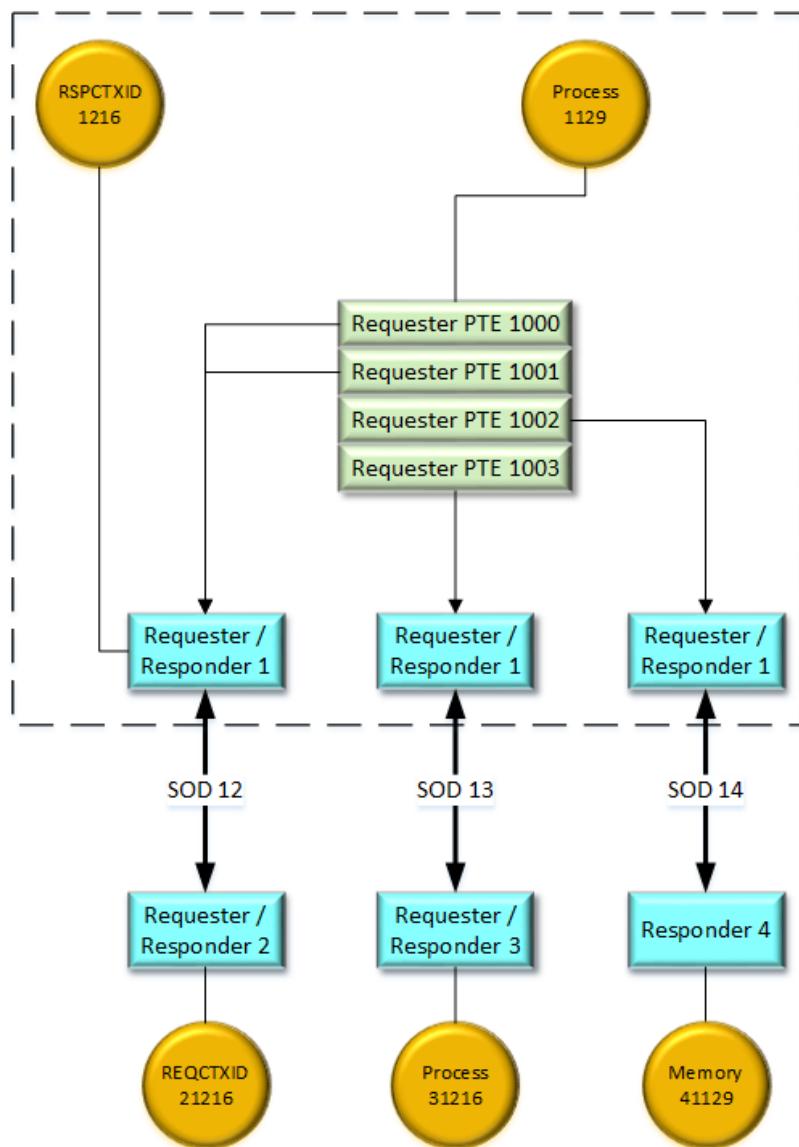


Figure 17-4: Generating SOD Request Packets

SOD request packets are transmitted in the order posted using a transmit (Tx) sequence number to uniquely identify each request packet. *SOD Request and Response Packet Flow* illustrates three request packets posted to a SOD. The first write is assigned transmit sequence number 0 (Tx 0), the read Tx 1, and the last write Tx 2. The Responder generates a unique response packet that acknowledges all request packets successfully executed at the time the response packet was generated. In this example, the Responder transmits ACK Rx 0 that corresponds to write Tx 0, a read response Rx 1 that corresponds to read Tx 1, and ACK Rx 2 that corresponds to write Tx 2.

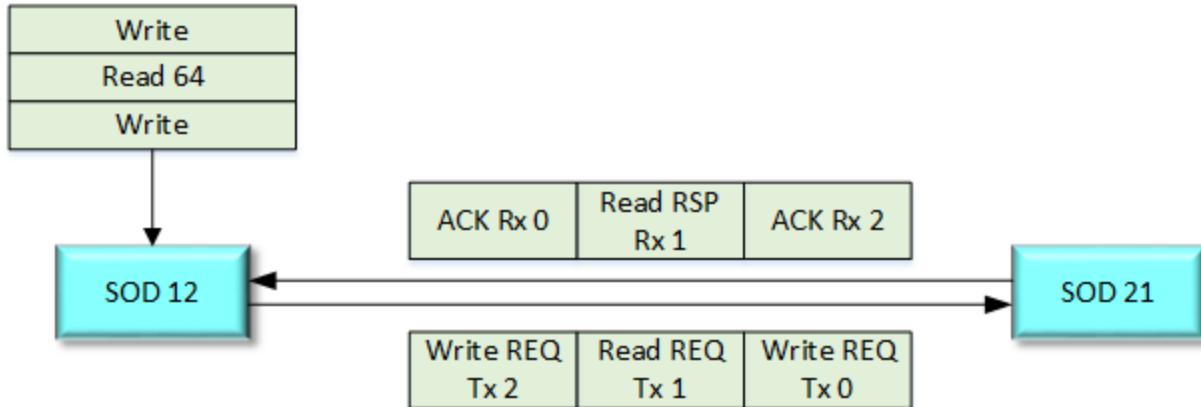


Figure 17-5: SOD Request and Response Packet Flow

Multi-packet operations such as a Write MSG are transmitted as a contiguous sequence of packets across a SOD as illustrated in *SOD Write MSG Packet Sequence*. This simplifies payload placement and determining when the operation as completed. This example also illustrates that response packets are cumulative acknowledgements, i.e., ACK Rx 3 acknowledges all request packets Tx 0 through Tx 3, enabling a Responder to opportunistically reduce the number of acknowledgment packets generated.

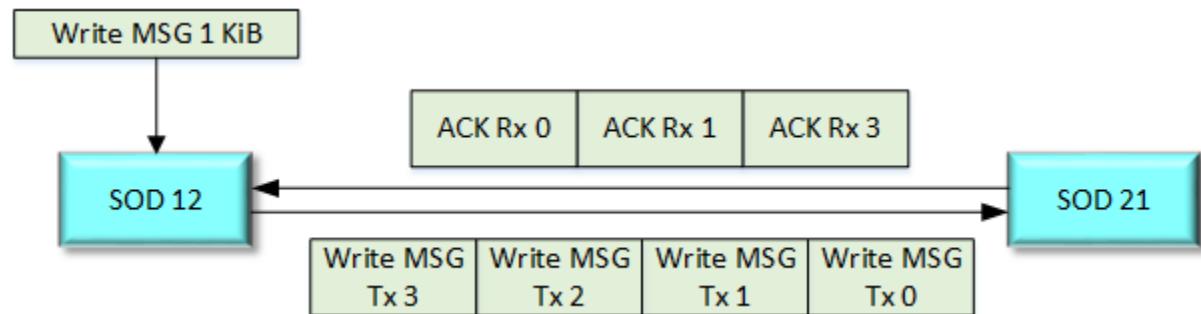


Figure 17-6: SOD Write MSG Packet Sequence

- 10 If a read request is larger than `Max_Packet_Payload`, then the next transmit sequence number is incremented by  $N$  to account for the number of corresponding read response packets. *SOD Read Request* illustrates a 1 KiB read request followed by a write request. Assuming a `Max_Packet_Payload` of 256 bytes, the Responder generates 4 read response packets. The Requester assigns the read request packet Tx 0 and the subsequent write request packet Tx 4. The Responder validates the SOD Read request packet, calculates the number of requisite response packets, and sets its next expected sequence number to Tx 4. Advancing the Tx sequence number simplifies response packet generation and response packet payload placement.
- 15

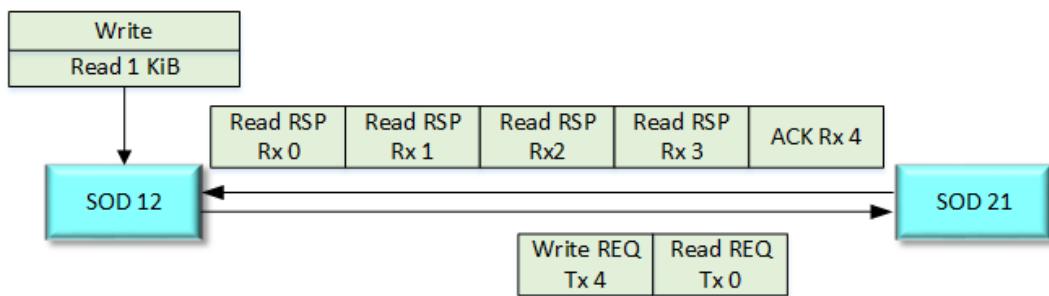


Figure 17-7: SOD Read Request

The following summarizes the unique features and requirements associated with SOD:

- Any component type may support a SOD.
- If supported, then the component shall support the SOD OpClass.
- SODs shall operate independently of one another. Protocol errors and associated error recovery in one SOD shall not impact the operation of another SOD.
- A SOD ID (SODID) shall be an integer value [0 to SODE] (see *Component SOD Structure*). A source SODID is a SSODID. A destination SODID is a DSODID. Management of SODIDs is outside of this specification's scope.
- A SOD shall be uniquely identified by [SCID | SGCID, SSODID, DCID | DGCID, DSODID].
- At any given time, each SOD shall transmit and receive packets through a single interface. If a component supports multiple interfaces, then only component interfaces 0 through 7 may be used to transmit and receive SOD packets. The *Component SOD Structure* is used to configure the primary and secondary interfaces that may be used by a given SOD.
- A Requester multiplexes one or more data flows across a given SOD. Since a Requester is responsible for request ordering, a Requester may load balance or migrate data flows across multiple SODs at its discretion (e.g., for aggregate bandwidth or to perform planned / unplanned data flow migrations with minimal application disruption).
- A sequence number is associated with a single Sequence Number Space.
  - Each SOD shall be associated with a unique Sequence Number Space.
  - Each Sequence Number Space represents  $2^{16}$  Sequence Numbers.
  - A component shall support the entire Sequence Number Space—0 to  $(2^{16} - 1)$ .
  - Sequence numbers shall be assigned to SOD packets in monotonically increasing order (modulo  $2^{16}$ ).
  - A sequence number shall be associated only with a single Outstanding Request Packet. If no sequence numbers are available, a SOD shall not transmit new request packets until Outstanding Request Packets are successfully acknowledged and the associated sequence numbers are released for reuse.
  - In order to distinguish duplicate request packets from out-of-order packets, a SOD shall never have more than  $((2^{16} - 1) / 2)$  Outstanding Request Packets.
  - SOD communication uses two types of sequence numbers:
    - A Tx Seq number is associated with each SOD Outstanding Request Packet transmitted by the source component.
    - An Rx Seq is associated with the last SOD request packet successfully received and executed by this component, i.e., it acts as a positive acknowledgment. This sequence number is cumulative, i.e., sequence number N indicates all request packets up to and including N have been successfully received and executed.
    - When a SOD is initialized (see *Component SOD Structure*), the first SOD request packet transmitted on the SOD by a Requester shall set Tx Seq = 0x1 and the Responder shall set its next expected sequence number for the SOD to 0x1. If a request packet arrives with Tx Seq ≠ 0x1, then the Responder shall transmit a *SOD Standalone Acknowledgment* (Reason = SOD Transient Error) and Rx Seq = 0x0. Upon receiving the *SOD Standalone Acknowledgment*, the Requester shall retransmit all request packets starting with the one identified by Tx Seq = 0x1.
- Within a SOD, all request packets shall be Reliably Delivered between a source and a destination interface. For a SOD, Reliable Delivery means the packets are received exactly once and in the order transmitted.

- Until a SOD is established, a Responder shall silently discard request packets that target the SOD. If *Component Error and Signal Event Structure* is supported and so configured, a Responder may signal a SOD UP error.
- If a SOD request packet's Tx Seq matches the expected Tx Seq, then this is the next expected SOD packet.
- If a SOD request packet's Tx Seq is less than the expected Tx Seq, then this is a duplicate SOD request packet. The Responder shall take the following actions:
  - A Responder shall handle duplicate SOD request packets in the order they are received, i.e., a new corresponding SOD response packet shall be transmitted in the same order as the duplicate SOD request packet was received.
  - If a *SOD Write*, a *SOD Write MSG*, or a *SOD Interrupt* request packet, then the Responder shall transmit a new *SOD Standalone Acknowledgment* without executing the request packet.
  - If a *SOD Read Request* packet, then the Responder shall execute the request packet and shall transmit a new *SOD Read Response* packet.
  - If a *SOD Encapsulation* request packet, then the Responder shall decapsulate the request packet. If a *Standalone Acknowledgment* is to be returned for the encapsulated request packet, then shall transmit a new *SOD Standalone Acknowledgment* without executing the request packet. If request-specific response packet is returned, then the Responder shall take request-specific actions which may or may not require executing the duplicate encapsulated request packet, e.g., if a non-idempotent request packet, then the results of the prior request packet's execution shall be returned without executing the duplicate request.
  - The Responder shall not update its expected Tx Seq tracking logic.
- If a request packet's Tx Seq is greater than the expected Tx Seq, then the Responder shall discard the request packet and shall transmit a *SOD Standalone Acknowledgment* indicating a SOD Transient Error was detected. The Rx Seq field shall contain the sequence number of the last successfully received and executed request packet associated with this SOD.
- If a response packet's Rx Seq is greater than the expected Rx Seq (an intermediate response packet was lost), then the Requester shall silently discard the response packet and initiate error recovery.
- SOD ordering:
  - A Requester shall transmit SOD request packets in the order that they were posted.
  - A Responder shall validate and execute SOD request packets in monotonically-increasing order (modulo  $2^{16}$ ) as indicated by the Tx Seq number.

**Developer Note:** *SOD implementations are strongly encouraged to support Link-level Reliability (LLR) along the entire SOD path to reduce the probability of packets arriving out of order or being silently discarded due to transient errors.*

- All response packets shall be strongly ordered, i.e., if request packet 1 is received prior to request packet 2, then response packet 1 shall be transmitted prior to response packet 2.
- If a Responder does not successfully execute a SOD request packet, then:
  - The Responder shall silently discard all new SOD request packets until it receives a new request packet with the next expected Tx Seq value or a *SOD Sync* packet which modifies the next expected Tx Seq value.

- The Responder shall silently discard any response packet awaiting transmission that corresponds to a subsequently received SOD request packet.
  - The Responder shall not modify the underlying resources associated with any subsequently received SOD request packet that had begun execution prior to error detection.
- A SOD may be dynamically migrated to flow across a different physical path:
  - Management and / or packet relay components may dynamically update packet relay tables to reflect a new communication path between the source and destination component interfaces.
  - Management and / or a component may dynamically migrate a SOD to a new source and / or destination interface on the same respective components.
  - Due to path changes, packets may be incorrectly relayed, lost, or arrive out-of-order. These SOD packets are handled as transient errors and recovered through the SOD retransmission process.
- A *SOD Standalone Acknowledgment* or a request-specific response packet shall cumulatively acknowledge all packets received up to and including the packet identified by the Rx Seq.
  - A *SOD Standalone Acknowledgment* or a request-specific SOD response packet shall be scheduled as soon the corresponding SOD request packet has been successfully validated and executed. The Responder may opportunistically update the Rx Seq field to cumulatively acknowledge multiple SOD request packets that use a *SOD Standalone Acknowledgment* as the corresponding response packet.
- If a SOD request packet suffers a validation or execution error, then the Responder shall generate a *SOD Standalone Acknowledgment* with a non-zero Reason field indicating the highest precedence error. This packet is referred to as a SOD NAK (negative acknowledgment).
  - A SOD NAK shall apply to only a single SOD, i.e., it shall have no impact on unrelated SODs.
  - A SOD NAK shall be transmitted as a *SOD Standalone Acknowledgment*.
  - A SOD NAK shall set the Reason field to the highest-precedence detected error—see *Reason Field Encodings*.
    - If Reason indicates an error that can be recovered by packet retransmission, e.g., a transient error, then the Requester shall retransmit one or more SOD request packets in their original sequential order starting with the first packet at the head of the SOD’s retransmission queue. If not immediately retransmitted, then the Requester shall retransmit all request packets in their original sequential order.
    - If Reason indicates an error that cannot be recovered by packet retransmission, then the Requester shall initiate component-local error handling. If the *Component Error and Signal Event Structure* is supported, then the error shall be handled as configured. Once component-local error handling has completed, software shall disable this SOD’s operation (see *Component SOD Structure*) or the Requester shall transmit a *SOD Sync* packet which will abort any outstanding request operations and synchronize sequence numbers to enable the Requester and Responder to continue to use the SOD.
  - During request packet retransmission, the Requester shall maintain three pointers relative to the SOD’s retransmission queue:
    - The oldest unacknowledged SOD request packet—this represents the logical head of the SOD’s retransmission queue.
    - The most recent unacknowledged SOD request packet—this represents the logical tail of the SOD’s retransmission queue.
    - An intermediate pointer to the current SOD request packet being retransmitted.

- A Responder shall transmit a new SOD NAK for each request packet until it detects a request packet with the next expected TX Seq number.
- To enable a SOD to support any Gen-Z operation without specifying a SOD analog, a SOD may encapsulate any non-SOD OpClass packet using a *SOD Encapsulation* packet. Though encapsulated, these operations shall comply with the specified semantics as when not encapsulated. For example, *Atomic Request and Response Packets* use the same request-response-clear-acknowledgment exchange albeit via *SOD Encapsulation* request and response packets. Similarly, *Forward Progress Screen (FPS)* information shall be communicated within the encapsulated packet.
- At any given time, a SOD shall be associated with a single interface on each component. If the component is unable to transmit packets to the destination component's interface (e.g., due to exceeding the maximum request packet retransmission or due to egress interface failure), then the component may set the MI (Migrate Interface) field within any SOD packet to automatically trigger interface migration. Transmitting a packet with MI == 1b is referred to as triggering SOD interface migration.
  - Upon receipt of a packet with MI == 1b, if the SSOD / MSOD SSODID table entry was configured with an alternative interface and interface migration has been armed, then the component shall exchange all subsequent packets associated with this SOD on the alternative interface.
  - If interface migration was not armed, then the SOD shall not be migrated.
  - Software may force interface migration from the primary to the secondary egress interface at its discretion. Software shall be responsible for configuring the primary and secondary egress interface fields corresponding to the SOD within the component's *Component SOD Structure* prior to initiating interface migration.

## 17.1. SOD Operations and Packet Formats

This section specifies the operations and packet formats used to support SOD communications. Unless explicitly stated otherwise by this specification, protocol fields that share the same name shall be governed by the same requirements and semantics as non-SOD explicit OpClass packets. *Common SOD Packet Protocol Fields* specifies protocol fields that are unique to SOD packets.

Table 17-1: Common SOD Packet Protocol Fields

Field Name	Size (bits)	Description
<b>SSODID</b>	6	Source SOD Identifier
<b>DSODID</b>	6	Destination SOD Identifier
<b>Tx Seq</b>	16	A transmitter-generated sequence number associated with this packet.
<b>Rx Seq</b>	16	The sequence number of the last successfully received and executed packet.
<b>RSPCTXID</b>	24	Identifies the Responder context used to locate an anonymous destination receive buffer at the Responder. RSPCTXID 0x0 shall be Reserved.

Field Name	Size (bits)	Description
REQCTXID	24	Identifies the Requester context used to generate the SOD Write MSG request packet at the Requester. REQCTXID 0x0 shall be Reserved.
	1	Migration Interface—Indicates if the SOD is to migrate to the alternative egress interface (see the ARM field within the <i>Component SOD Structure</i> ). The scope of the MI field shall be only the SOD upon which the packet was received. 0b—Do Not Migrate 1b—Migrate Interface

### 17.1.1. SOD Read Request

The following are the features and requirements of the SOD Read request packet:

- A SOD Read request packet shall use the *SOD Read Request Packet Format*.
- SOD Read request packets shall operate per a Core 64 Read Request as specified in *Read and Read Response*.

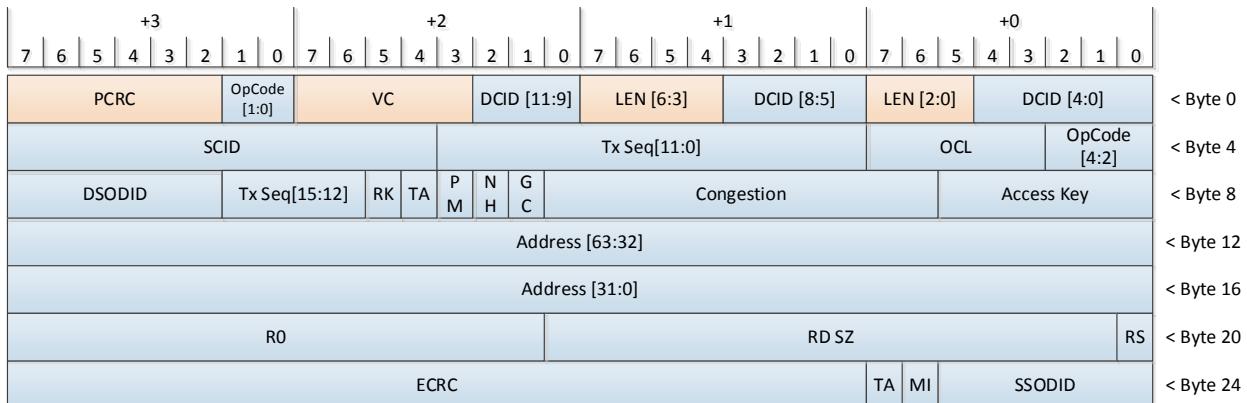


Figure 17-8: SOD Read Request Packet Format

Table 17-2: SOD Read Request-specific Protocol Fields

Field Name	Size (bits)	Description
RS	1	Indicates how to interpret the RD SZ field. 0b—RD SZ represents a less than or equal to Max_Packet_Payload request 1b—RD SZ represents a multi-Max_Packet_Payload request
	16	Indicates the number of bytes to be read. If RS == 0b, then RD SZ shall be interpreted as follows:

Field Name	Size (bits)	Description
RS		<p>RD SZ = an integer 4-byte multiple <math>\leq</math> Max_Packet_Payload</p> <p>If RS == 1b, then RD SZ shall be interpreted as follows:</p> <p>If RD SZ == 0x0, then</p> <p style="padding-left: 20px;">read (<math>2^{16} * \text{Max\_Packet\_Payload}</math>) bytes</p> <p>else</p> <p style="padding-left: 20px;">read (RD SZ * Max_Packet_Payload) bytes</p>
RS	1	<p>If RS == 0b, then the Responder shall generate a single SOD Read Response packet upon successfully executing this SOD Read request packet.</p> <p>If RS == 1b, then the Responder shall generate multiple SOD Read Response packets upon successfully executing this SOD Read request packet.</p>
RO	15	Reserved

### 17.1.2. SOD Read Response

The following are the features and requirements of the SOD Read Response packet:

- A SOD Read Response packet shall use the *SOD Read Response Packet Format*.
- SOD Read Response shall operate per a Core 64 Read Response as specified in *Read and Read Response*.
- If a SOD Read Response packet is one of multiple response packets required to complete a given SOD Read request packet, then:
  - The Rx Seq number in the first SOD Read Response packet is equal to the Tx Seq in the corresponding *SOD Read Request* packet. The payload in this response packet is placed at byte 0 of the corresponding Requester-specific buffer.
  - The Rx Seq number of each subsequent SOD Read Response packet is incremented by one and the payload is placed at  $((\text{Rx Seq} - \text{Tx Seq}) \bmod 2^{16}) * \text{Max\_Packet\_Payload}$  bytes from byte 0 of the corresponding Requester-specific buffer.
  - All SOD Read Response packets shall carry a payload that equals Max\_Packet\_Payload bytes.
  - If a Responder is unable to transmit all SOD Read Response packets, then the Responder shall transmit a *SOD Standalone Acknowledgment* (Reason = highest precedence error) and set the Rx Seq equal to the last expected sequence number required to complete the *SOD Read Request*. Further, for each subsequently received non-*SOD Sync* packet the Responder shall transmit a *SOD Standalone Acknowledgment* (Reason = previously detected error) and Rx Seq equal to the last expected sequence number required to complete the *SOD Read Request* that failed. Upon receipt and successful execution of a *SOD Sync* packet, the Responder shall resume normal operation.

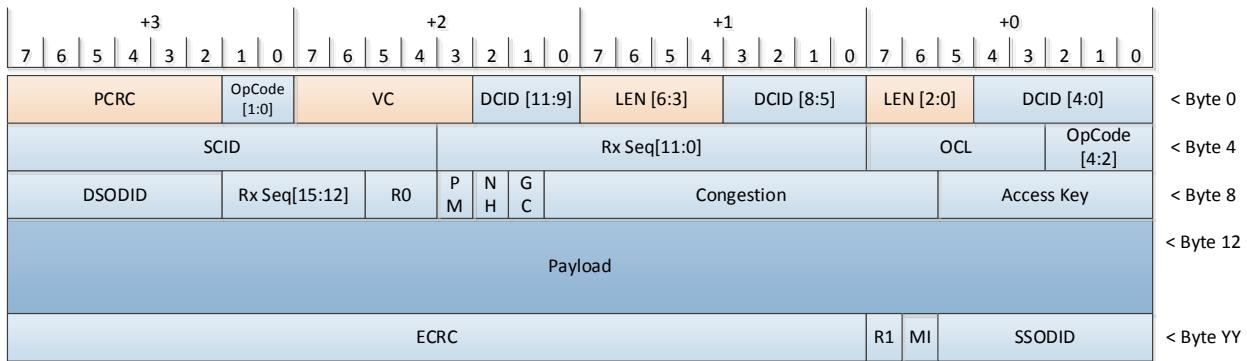


Table 17-3: SOD Read Response-specific Protocol Fields

Field Name	Size (bits)	Description
<b>R0</b>	2	Reserved
<b>R1</b>	1	Reserved

### 17.1.3. SOD Write

The following are the features and requirements of the SOD Write packet:

- 5
- A SOD Write request packet shall use the *SOD Write Request Packet Format*.
  - SOD Write and Write Persistent request packets shall operate per a Core 64 Write or Core 64 Write Persistent request as specified in *Write*.

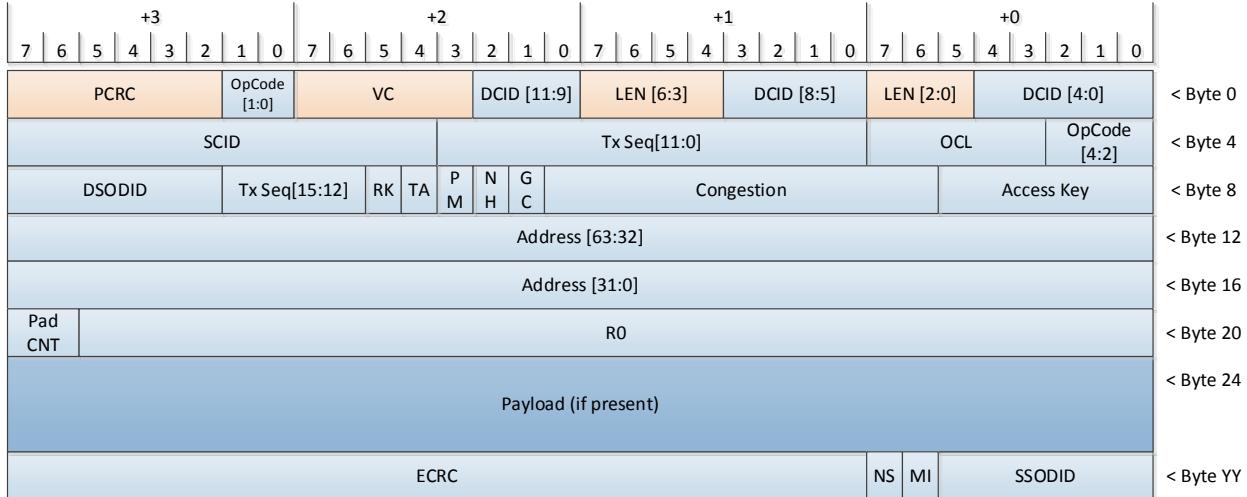


Table 17-4: SOD Write Request-specific Protocol Fields

Field Name	Size (bits)	Description
<b>R0</b>	30	Reserved

### 17.1.4. SOD Write MSG

The following are the features and requirements of the SOD Write MSG:

- Any component type may support SOD Write MSG.
- A SOD Write MSG request packet shall use the *SOD Write MSG Packet Format*.
- A SOD Write MSG request packet shall be acknowledged by a *SOD Standalone Acknowledgment*.
- Unless explicitly stated otherwise by this specification, a SOD Write MSG shall be semantically equivalent to a reliable Write MSG as specified in *Write MSG*.
- Since SOD Write MSG request packets are executed in the order transmitted, a Responder places the payload of the first Write MSG packet at byte zero of the anonymous buffer. The payload of each subsequent SOD Write MSG request packet in a packet series is sequentially placed after the payload of the prior SOD Write MSG request packet.
- Each RSPCTXID may be independently targeted by multiple REQCTXIDs. A Responder delineates SOD Write MSG operations by [SCID | SGCID, REQCTXID, RSPCTXID].
- If ER == 1b, then the payload shall be interpreted as an embedded read as specified in *Write MSG*.

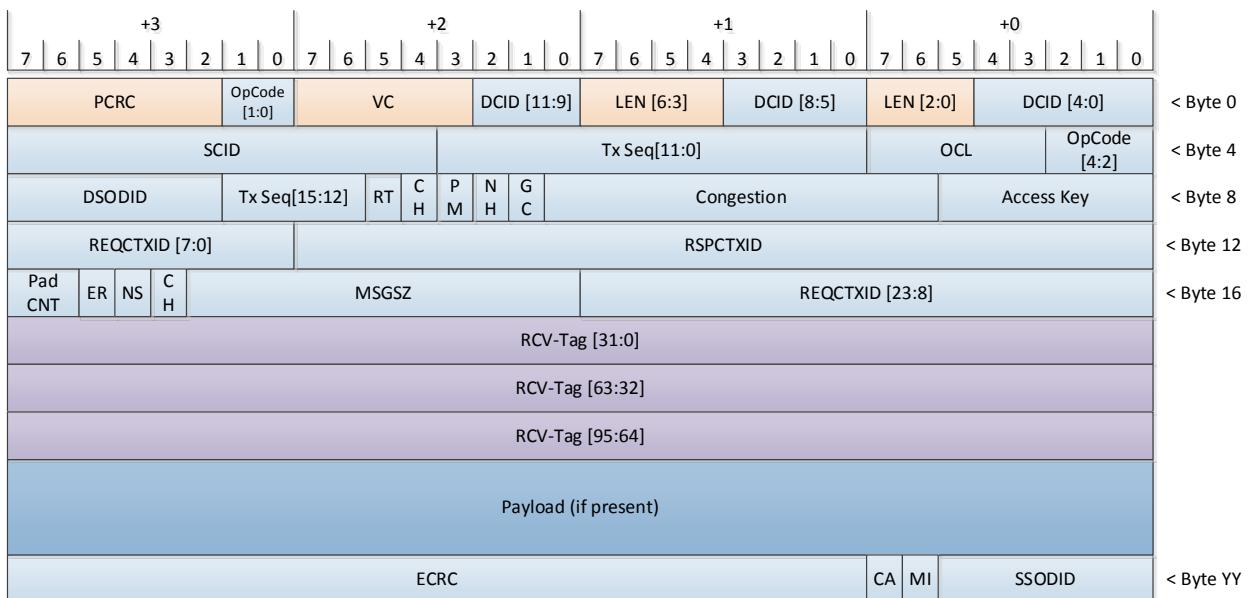


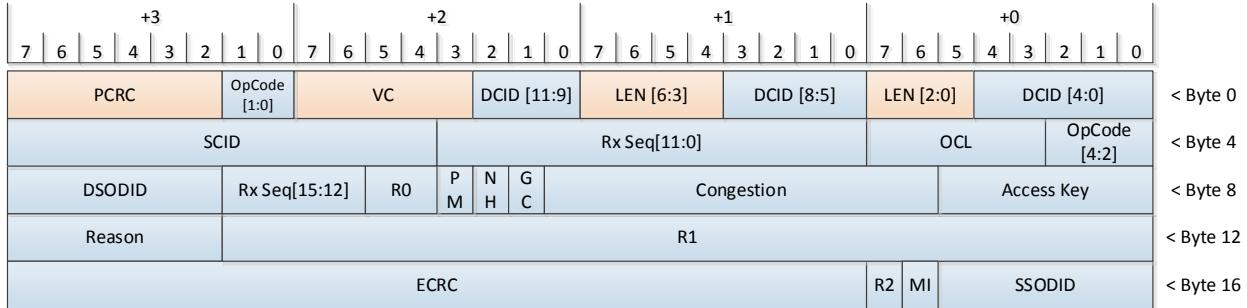
Figure 17-11: SOD Write MSG Packet Format

### 17.1.5. SOD Standalone Acknowledgment

The following are the features and requirements of the SOD Standalone Acknowledgment:

- A SOD Standalone Acknowledgment shall use the *SOD Standalone Acknowledgement Packet Format*.

- Except where explicitly stated otherwise, SOD Standalone Acknowledgments shall operate per *Standalone Acknowledgment*.
- A SOD Standalone Acknowledgment cumulatively acknowledges prior request packets. If a Responder detects an error while processing a request packet, it shall immediately schedule a SOD Standalone Acknowledgment to communicate the last successfully received request packet and the corresponding error. If a Responder successfully validates and executes a request packet and it wants to cumulatively acknowledge multiple request packets, then it may delay scheduling a SOD Standalone Acknowledgement for no more than the *Core Structure Responder Deadline* time.



- If the encapsulated packet is a response packet, then the Seq field shall be semantically treated as the Rx Seq field used in SOD response packets.
- Packet validation shall be performed in two steps:
  - The first step shall validate the SOD Encapsulation packet's protocol fields—the header and tail fields that surround the encapsulated packet. These fields shall be validated as specified in *Destination Component Packet Processing*.
    - The SOD Encapsulation packet's ECRC shall protect the entire encapsulated packet.
  - The second step validates the actual encapsulated packet. The encapsulated packet shall be independently validated using encapsulated packet-specific protocol validation.
    - If present, then the encapsulated packet's VC, DCID, PCRC, SCID, Congestion, Access Key, and ECRC fields shall be Reserved.

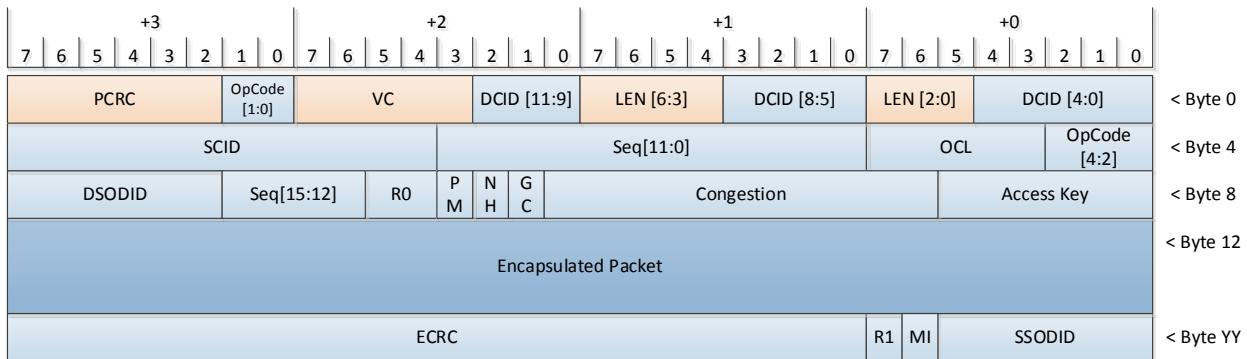


Figure 17-13: SOD Encapsulation Packet Format

Table 17-6: SOD Encapsulation Protocol Fields

Field Name	Size (bits)	Description
Seq	16	If the packet encapsulates a request packet, then Seq = TX Seq If the packet encapsulates a response packet, then Seq = Rx Seq
R0	2	Reserved
R1	1	Reserved

### 17.1.7. SOD Interrupt

A SOD Interrupt request packet is used to transmit a native interrupt to the Interrupt Target.

The following are the features and requirements of the SOD Interrupts:

- Any component type may support SOD Interrupts.
- A SOD Interrupt packet shall use the *SOD Interrupt Packet Format*.
- Except where explicitly stated otherwise in this specification, a SOD Interrupt request shall be semantically operate as a native interrupt as specified in *Interrupts*.
- A SOD Interrupt request packet shall be acknowledged by a *SOD Standalone Acknowledgment*.

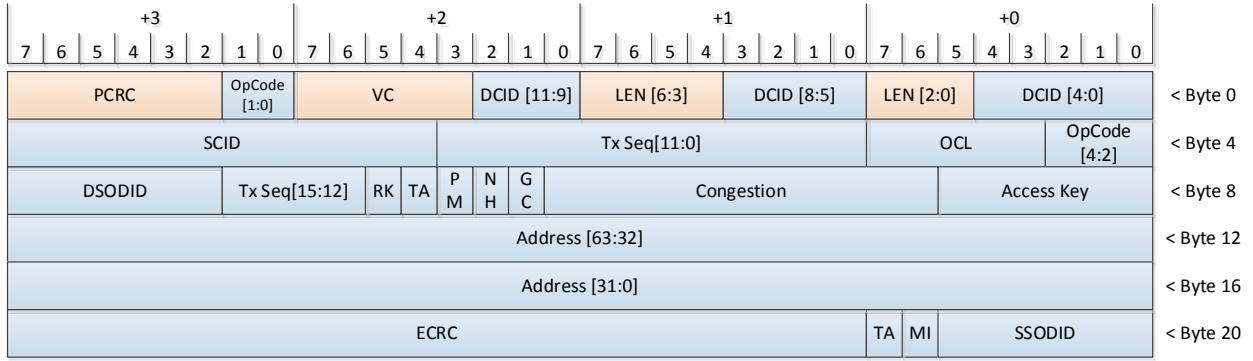


Figure 17-14: SOD Interrupt Packet Format

### 17.1.8. SOD Sync

The SOD Sync is used to abort any request packet whose execution has not completed and to synchronize sequence numbers. The reason to abort a request packet is outside of this specification's scope, e.g., due to the termination of the application that initiated the request packet or a Requester received a SOD NAK indicating a non-recoverable error.

The following are the features and requirements of the SOD Sync:

- Any component type may support SOD Sync.
- A SOD Sync packet shall use the *SOD Sync Packet Format*.
- A SOD Sync packet shall impact only the SOD that received the packet.
- The Requester shall increment the Tx Seq number of the last transmitted request packet for this SOD by  $((Tx\ Seq + 2^{15}) \bmod 2^{16})$ . If the request packet consumed multiple sequence numbers, e.g., a large *SOD Read Request* that progresses the sequence number by N to account for N *SOD Read Response packets*, then the Tx Seq =  $((Tx\ Seq + N + 2^{15}) \bmod 2^{16})$ .
- The Requester shall set the SPD Sync request packet's Tx Seq field to the updated Tx Seq.
- The Requester shall set its next expected response sequence number for this SOD to the updated Tx Seq.
- Upon receipt of a SOD Sync packet:
  - The Responder shall abort the execution of any request packet and shall not schedule any new response packets whose Rx Seq number is less than the SOD Sync packet's Tx Seq.
  - The Responder shall set the SOD's next expected sequence number to the SOD Sync packet's  $((Tx\ Seq + 1) \bmod 2^{16})$ .
  - The Responder shall transmit a *SOD Standalone Acknowledgment* with Rx Seq = SOD Sync Tx Seq.

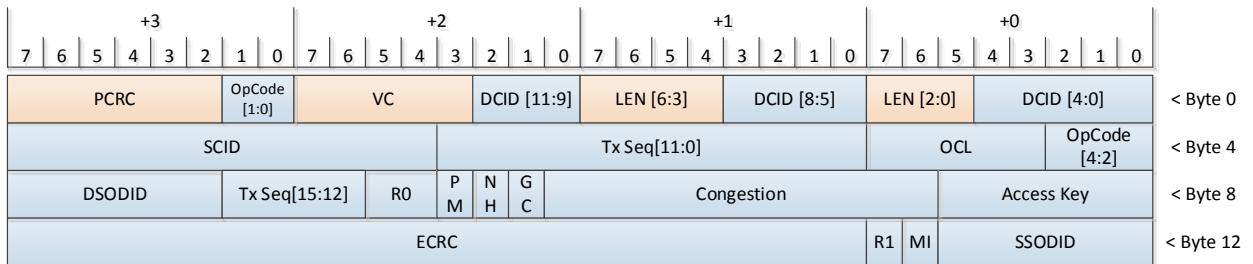


Figure 17-15: SOD Sync Packet Format

Table 17-7: SOD Sync Protocol Fields

Field Name	Size (bits)	Description
<b>R0</b>	2	Reserved
<b>R1</b>	1	Reserved

## 17.2. SOD OpClass Packet Processing

SOD OpClass packets shall be validated and processed as illustrated in the following figures.

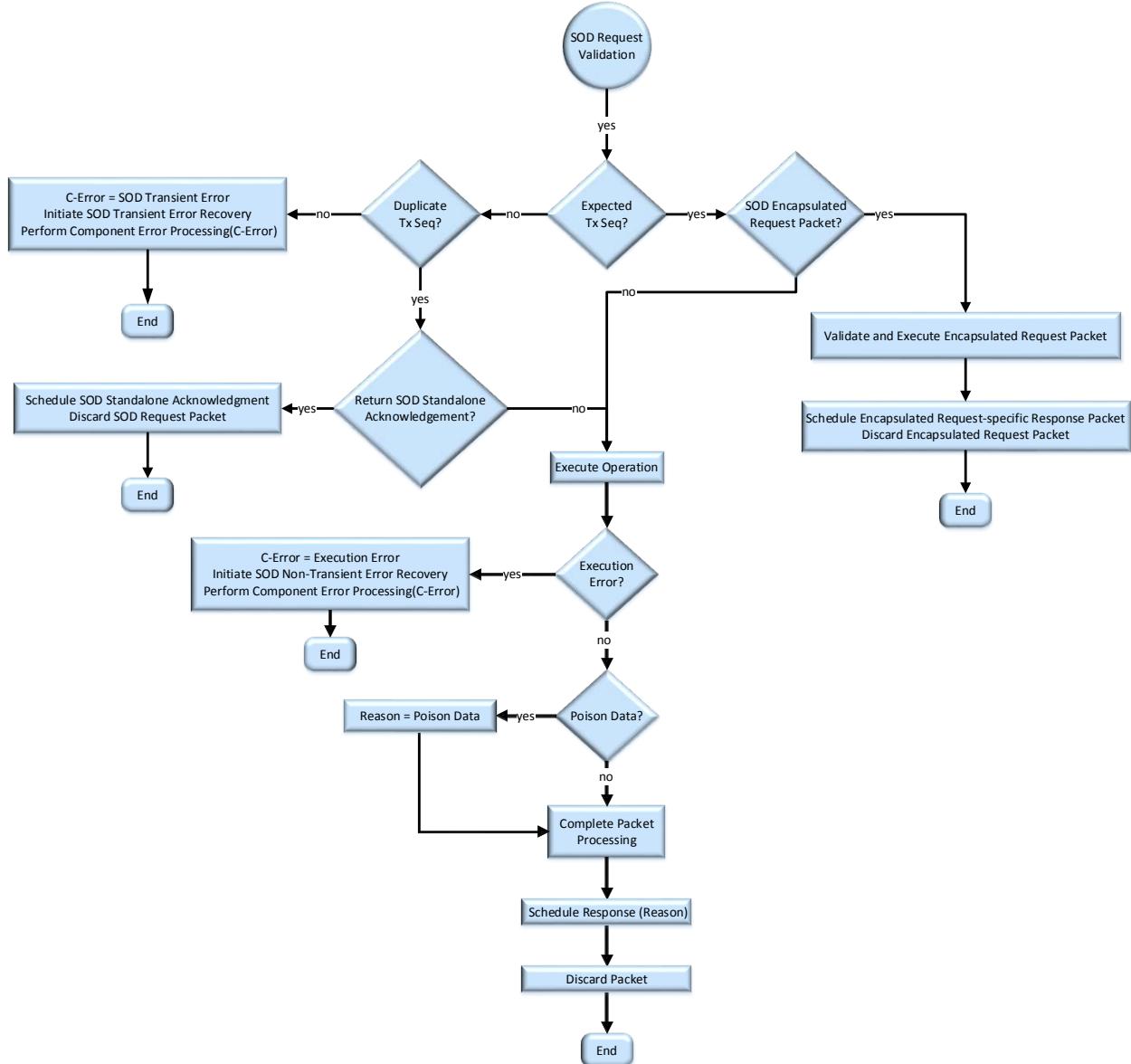


Figure 17-16: SOD OpClass Request Packet Processing

Table 17-8: SOD OpClass Request Packet Processing Details

Validation Step	Explanation
<b>SOD Operation: Expected Tx Seq?</b>	Determine if the packet is the next expected request packet.
<b>Unexpected packet: Duplicate Tx Seq?</b>	Determine if a duplicate packet.
<b>Unexpected packet: Non-duplicate</b>	Non-duplicate packet; generate a <i>SOD Standalone Acknowledgment</i> (Reason = SOD Transient Error)
<b>Duplicate packet: Returns SOD Standalone Acknowledgment?</b>	If yes, then schedule a <i>SOD Standalone Acknowledgment</i> , and discard this packet. If no, then re-validate and execute the SOD operation if required.
<b>Expected packet: SOD Encapsulated Packet?</b>	If a SOD encapsulated packet, then validate and execute the encapsulated packet. Generate a <i>SOD Standalone Acknowledgment</i> or an encapsulated request -specific response packet.
<b>Execution Error?</b>	Was an error detected during the execution of the packet? There are numerous potential causes depending upon the OpCode, component state, etc. If an error was detected, then the packet proceeds to component error processing.
<b>Poison Data?</b>	If poison data was encountered while executing the packet, then set temporary variables to (Reason = Poison Data (PD) Detected).
<b>Request Packet?</b>	If this was a request packet, then complete packet processing, and generate the corresponding response packet.

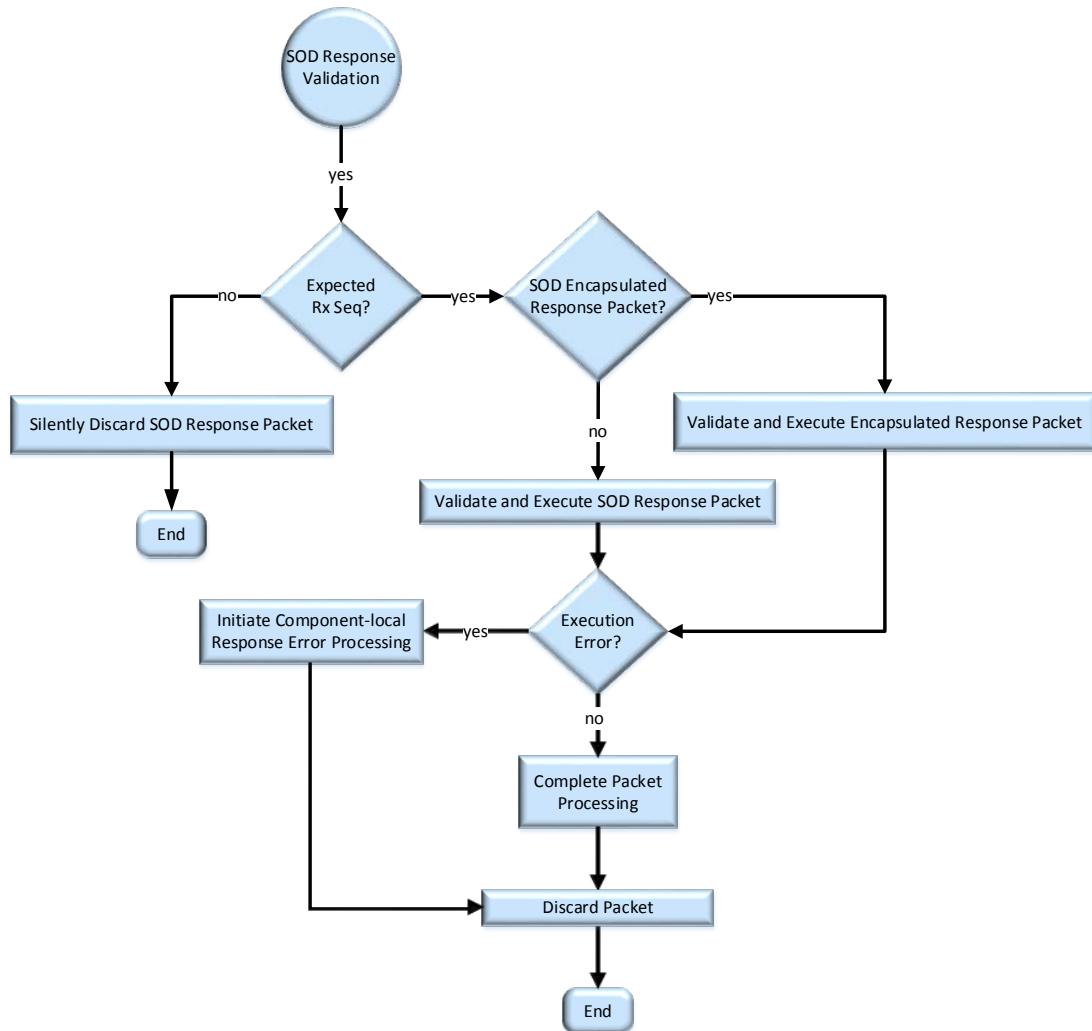


Figure 17-17: SOD OpClass Response Packet Processing

Table 17-9: SOD OpClass Response Packet Processing Details

Validation Step	Explanation
<b>SOD Operation: Expected Rx Seq?</b>	Determine if the packet is the next expected response packet. If not, silently discard the response packet.
<b>SOD Encapsulated Response Packet?</b>	If yes, then validate and execute the encapsulated response packet. If no, then validate and execute the SOD response packet.
<b>Execution Error?</b>	If an execution error was detected, then initiate component-local response error processing, else, complete response packet processing and discard the SOD response packet.

# 18. Logical PCI Device (LPD)

Unless explicitly stated otherwise by this specification, “PCI” refers to both “Conventional PCI” and PCI Express functionality as defined by the *PCI Express Base Specification*.

## 18.1. LPD Operational Overview

5 A system (e.g., a processor) may support a mix of Gen-Z I/O components and PCIe-attached I/O devices as illustrated in *Example System with Gen-Z I/O Components and PCI I/O Devices*. If the associated operating system supports a PCI-based software infrastructure to manage I/O devices, then the Gen-Z I/O components need to be presented to the operating system as LPDs as illustrated in *Example System with Logical and Physical PCI I/O Devices*. To present these LPDs requires the following hardware

10 support:

- The PCI Express ECAM functionality, as specified in the *PCI Express Base Specification*, is emulated to map LPDs to Gen-Z I/O components. This functionality is referred to as the PCI Enhanced Configuration Access Mechanism (PECAM). A Requester that supports PECAM logic is referred to as a PECAM Requester.
- To be mapped as a LPD, a Gen-Z I/O component needs to support a new Control Space structure (see *Component LPD Structure*). At a minimum, this structure contains one PCI Type 0 Configuration Space Header per supported LPD function number.
- Control OpClass support for *Control Read and Control Write Packets* and *the Unsolicited Event (UE) Packet*.

20

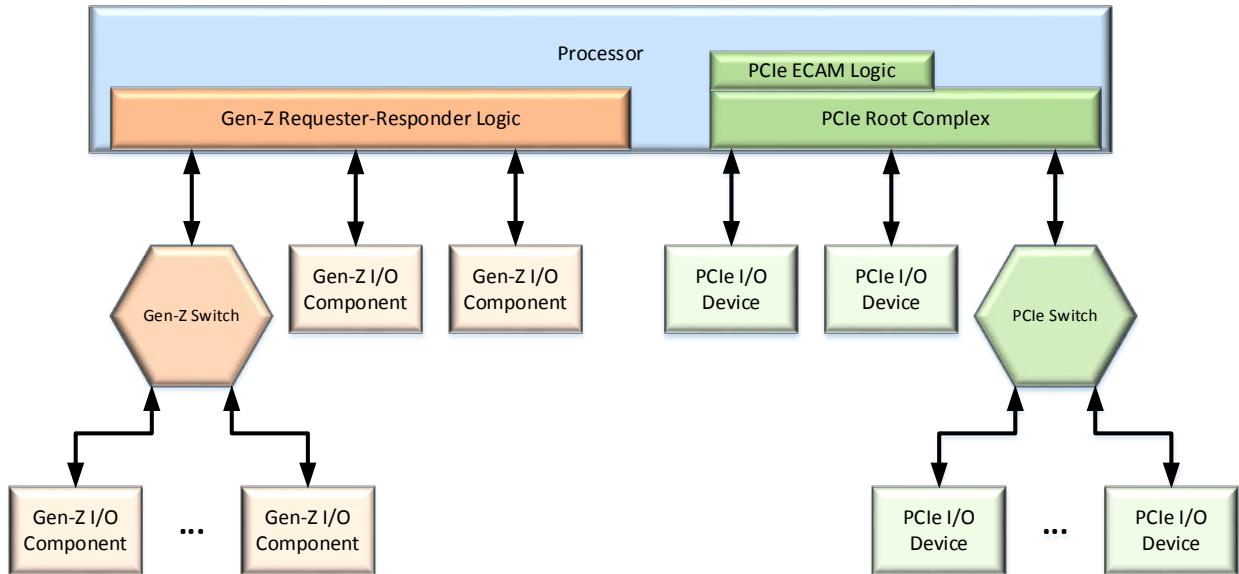


Figure 18-1: Example System with Gen-Z I/O Components and PCI I/O Devices

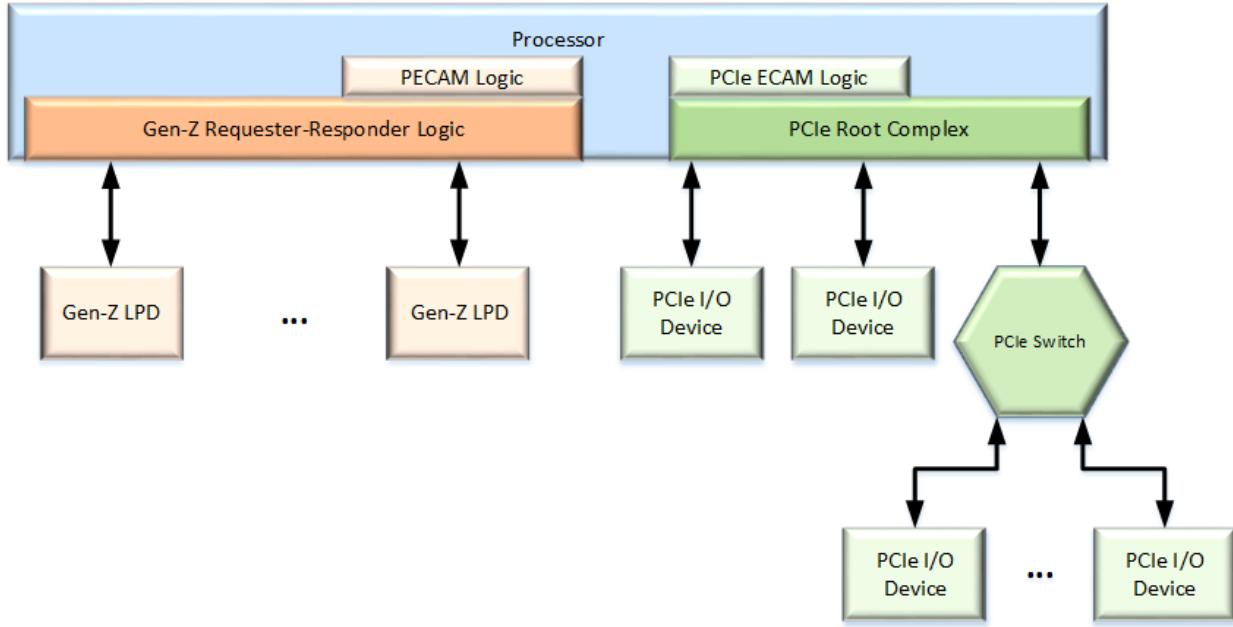


Figure 18-2: Example System with Logical and Physical PCI I/O Devices

## 18.2. PCIe-Compatible Ordering (PCO)

PCI and PCIe transaction ordering is highly reliant on the use of tree-based fabric topologies. Gen-Z protocol and fabrics support topologies far more flexible than PCI's tree-based topologies, and offer increased scaling, resilience, performance, and error recovery capability. To enable this, Gen-Z uses fundamentally different transaction ordering models. PCIe-Compatible Ordering (PCO) is a special mechanism in Gen-Z that enables drivers and I/O infrastructure software written for PCI and PCIe platforms to be ported more easily to Gen-Z platforms, by removing the need to change such software in order to accommodate ordering model differences. PCO is compatible with both PCI and PCIe ordering, but has "PCIe" in its name since it is based on following PCIe ordering rules.

*Simplified Summary of PCIe Ordering Rules* is a simplified summary of PCIe ordering rules, ignoring special cases for Relaxed Ordering (RO), ID-Based Ordering (IDO), PCI/PCI-X bridges, and multi-packet Completions. The "No" entries indicate where passing is prohibited to satisfy the producer/consumer ordering model. The "Yes" entries indicate where passing shall occur in certain situations to avoid deadlock. The "Y/N" entries indicate where it doesn't matter from an ordering perspective whether passing occurs.

Table 18-1: Simplified Summary of PCIe Ordering Rules

Row Pass Column?	Posted Request	Non-Posted Request	Completion
Posted Request	No	Yes	Y/N
Non-Posted Request	No	Y/N	Y/N
Completion	No	Yes	Y/N

PCO satisfies basic PCIe ordering requirements by supporting only one path for each logical channel carrying PCO traffic, and not allowing any end-to-end (EE) packet passing along that path. On each link along the path, software enables *Link-level Reliability (LLR)* to deliver the EE packets reliably and in order. Within each packet-relay component along the path, software configures VCs carrying PCO traffic to do the same. Special rules within PCO Requesters and Responders satisfy the remaining PCIe ordering requirements, including ones for protocol deadlock avoidance.

With LPDs, PCO is supported only between an LPD and its host component or other LPDs.

PCO is also supported between a *Logical PCI Hierarchy (LPH)* and its host component.

**Developer Note:** Though PCO offers the key benefit of not requiring ported PCI/PCIe software to change due to ordering model differences, there are significant benefits for I/O solutions that do not use PCO:

- Multi-path can be configured, supporting higher bandwidth, dynamic load balancing, increased availability, and less fabric congestion
- Automatic path failover can avoid the need to trigger containment and resync drivers for recovery
- Single-path configurations can also achieve increased link utilization, efficiency, and handling of congestion
- Can avoid fabric topology restrictions associated with PCO

### 18.2.1. PCO Operation and Requirements

#### PCO terminology

- A *PCO endpoint* is either a host (e.g., a SoC component), an LPD function that is configured to use PCO communications, or an LPH that is configured to use PCO communications.
- A *PCO stream* is a logical channel between a pair of PCO endpoints.
- A *PCO packet* is an end-to-end packet that is delivered via a PCO stream.
- A *PCO RNR NAK* is a *Responder Not Ready (RNR) NAK* that is delivered via a PCO stream.
- A *PCO traffic VC* is a Virtual Channel with PCO enabled, which carries all PCO packets except PCO RNR NAKs.
- A *PCO control VC* is a Virtual Channel with PCO enabled, which carries only PCO RNR NAKs.

All EE packets in a PCO stream are classified according to PCIe ordering rule classes:

- Posted Requests (P): these request packets have no response or acknowledgement packets, and are used for:
  - Host-to-LPD / LPH write requests, aka memory-mapped I/O (MMIO) writes
  - LPD / LPH-to-host write requests, aka direct memory access (DMA) writes
  - LPD-to-LPD write requests, aka peer writes
  - LPD / LPH-to-host interrupt requests, aka message-signaled-interrupts (MSIs)
- Non-Posted Requests (N): these request packets have response or acknowledgment packets, and are used for:
  - Host-to-LPD / LPH read requests, aka memory-mapped I/O (MMIO) reads
  - LPD / LPH-to-host read requests, aka direct memory access (DMA) reads
  - LPD-to-LPD reads requests, aka peer reads
  - Atomic requests, aka AtomicOp requests
- Completions (C): these response or *Standalone Acknowledgment* packets complete Non-Posted Requests.

Rules for PCIe's producer/consumer ordering model:

- P shall not pass another P
- N shall not pass P
- C shall not pass P

**Developer Note:** Recommendations for components that support PCO communications:

- If a component implements multiple LPD functions capable of using different software device drivers, then it should implement at least five VCs. This specifically applies if the component implements multiple LPDs, or if it implements a single LPD with multiple functions that use different software device driver models.
- There are significant benefits for driver/function combos when PCO is not enabled. The recommendation for at least five VCs is to enable some driver/function pairs not to use PCO while other driver/function pairs do. For the case with five implemented VCs, three of the VCs may be configured for non-PCO use, and two of the VCs may be configured as a PCO stream.

**Developer Note:** Gen-Z VCs are not as resource intensive as PCIe VCs. Each PCIe VC requires six sets of flow-control credits and associated buffers – a pair each for Posted Requests, Non-Posted Requests, and Completions. Each Gen-Z VC requires only one set of flow-control credits and associated buffers.

PCO semantic rules for the Gen-Z fabric and PCO endpoints:

- Each PCO stream shall have only one path between its associated pair of PCO endpoints.
  - If any component along that path does not support PCO communications, then the PCO stream shall not be configured.
- Multiple PCO streams may be configured between a given pair of PCO endpoints, provided that sufficient PCO resources are available. There is no ordering between different PCO streams.
- PCO streams shall use RNR NAKs only for protocol deadlock avoidance, and these are referred to as PCO RNR NAKs. See *Protocol Deadlock Avoidance for PCO*.
- Each PCO stream shall be provisioned with two VCs, each of which is enabled for PCO communications in all interfaces along the path:
  - The PCO traffic VC shall be used for all end-to-end packets on the PCO stream except PCO RNR NAKs.
  - The PCO control VC shall be used only for PCO RNR NAKs.
  - PCO traffic VCs for multiple PCO streams may share the same VC on common links.
  - PCO control VCs for multiple PCO streams may share the same VC on common links.
  - A PCO traffic VC and a PCO control VC shall never share the same VC on a common link, regardless of them being on the same PCO stream or not.
  - PCO VCs shall never carry non-PCO traffic.
  - If the path for a PCO stream includes one or more packet-relay components that support VC remapping, then the VC contained in a PCO packet's VC field may be remapped to different VCs along the path.
- Each PCO endpoint shall be enabled for PCO communications.
  - One or more Traffic Classes shall be established for PCO communications by configuring appropriate tables in the *Component Destination Table Structure* such that for a given PCO Traffic Class:
    - A PCO Requester using that Traffic Class shall always result in the same PCO traffic VC being selected.
    - For all response packets except PCO RNR NAKs, a PCO Responder shall use the same PCO traffic VC as the one in the associated request packet.

- For PCO RNR NAKs, a PCO Responder shall use the PCO control VC, which is obtained by using the PCO traffic VC from the request packet to index into the RESP-VCAT, which shall be configured along with any associated tables such that the PCO control VC corresponding to that PCO traffic VC is always selected.
- 5     ○ For an LPD function or LPH, PCO communications shall be enabled and its PCO Traffic Class shall be specified using its associated *Component LPD Structure* or *Component LPH Structure*.
  - Such functions may send all traffic on PCO streams.
  - Such functions may send selected traffic on non-PCO VCs, as determined by implementation-specific mechanisms. Such traffic will not have PCIe-compatible ordering or guaranteed single-delivery semantics.
- 10    ○ For a host generating PCO requests using a Requester ZMMU, the PTEs used for those requests shall each specify a PCO Traffic Class.
- 15    ● PCO streams may include requests either with or without LPD fields.
- Non-PCO streams may include requests either with or without LPD fields.

PCO semantic rules for a PCO endpoint receiving PCO packets:

- Except as noted below, received PCO packets shall be processed in the order received.
- Ps may pass Ns or Cs, either queued or being serviced.
- Cs may pass Ns, either queued or being serviced.
- Ns may pass each other, either queued or being serviced.
- Received Ps & Cs shall be processed (sunk) by the PCO endpoint without being contingent on the endpoint transmitting other packets.
- The receiver may implement PCIe's Relaxed Ordering (RO) or ID-Based Ordering (IDO) optimizations, as defined in the *PCI Express Base Specification*.
- When required for protocol deadlock avoidance, the receiver shall handle certain Ns with RNR NAK. See *Protocol Deadlock Avoidance for PCO*.

PCO semantic rules for a PCO endpoint transmitting PCO packets.

- Except as noted below, PCO packets queued for transmission shall be sent in the order queued.
- Ps may pass Ns or Cs.
- Cs may pass Ns.
- Cs may pass other Cs that are not associated with the same N.
- The transmitter may implement PCIe's Relaxed Ordering (RO) or ID-Based Ordering (IDO) optimizations, as defined in the *PCI Express Base Specification*.

**Developer Note:** Though these exceptions for outbound PCO packets are permitted, implementing them might not provide significant benefit.

## 18.2.2. Deadlock Avoidance for PCO

### 18.2.2.1. *Protocol Deadlock Avoidance for PCO*

Unlike PCIe, Gen-Z doesn't manage flow-control credits independently for Posted Requests, Non-Posted Requests, and Completions. However, because Non-Posted Requests have the basic property that their processing can't be completed until their associated Completions are queued for transmission, PCO has a potential protocol deadlock case similar to ones associated with PCIe fabrics.

5 *Potential Protocol Deadlock Case for PCO* illustrates a potential PCO protocol deadlock case. Both the host & LPD are doing heavy bi-directional N traffic. Each is servicing as many Ns as it has resources for, and each has all of its receive buffers filled with Ns. The PCO traffic VC connecting both components is filled with other PCO packets, whose specific PCIe ordering class doesn't matter. Both components are unable to transmit any Cs, and thus neither is able to complete processing the Ns they are servicing. Without a protocol deadlock avoidance mechanism, the PCO stream would be deadlocked.

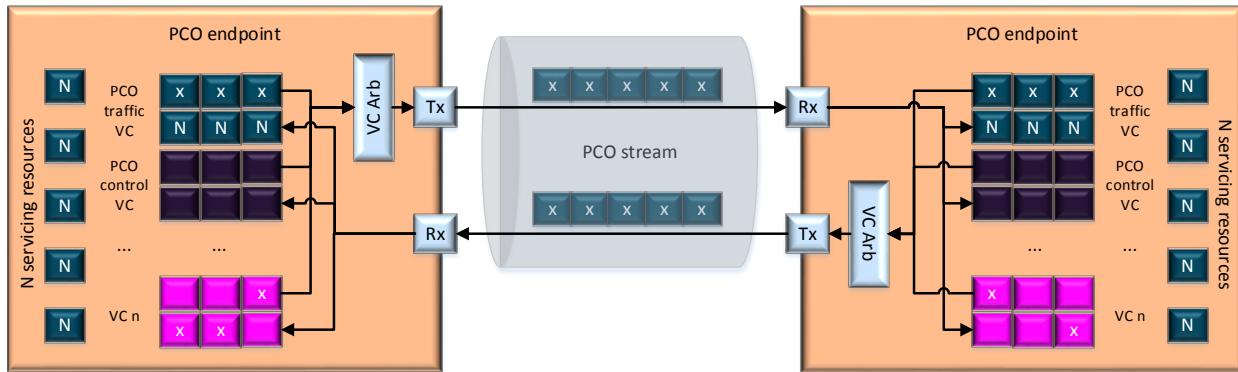


Figure 18-3: Potential Protocol Deadlock Case for PCO

Rules for PCO protocol deadlock avoidance:

- 10 • A PCO Responder shall reserve one or more request resources for servicing PCO Ns. The Responder shall implement a *Forward Progress Screen (FPS)* dedicated for these *reserved* PCO request resources and shall not associate this PCO FPS with any other type of request resources. The Responder shall support one or more *unreserved* request resources suitable for servicing PCO Ns, but these may also be used to service non-PCO requests.
- 15 • In this context, inbound PCO traffic is blocked when Ns fill all receive buffers suitable for Ns, PCO traffic VC receive flow-control credits <  $VC_k$  Min, and the PCO FPS determines that an RNR NAK needs to be transmitted.
- 20 • In this context, outbound PCO traffic is blocked if all Ns being serviced have Cs ready to transmit, but there are insufficient PCO traffic VC flow-control credits to transmit Cs available for transmission.
- 25 • If both inbound and outbound PCO traffic is blocked, then the PCO Responder shall transmit the PCO RNR NAK on the PCO control VC, silently discard the N packet, and schedule a flow-control credit update.
  - The RNR NAK Time Interval shall indicate that immediate retransmission is permitted (see *RNR NAK Time Interval Encoding*).
  - If PCO traffic VC receive flow-control credits <  $VC_k$  Min, then the PCO endpoint shall wait an implementation-specific time period for the peer interface to process flow-control credit updates and transmit the next PCO traffic packet if sufficient flow-control credits are then available.
  - If both inbound and outbound PCO traffic is still blocked, then the PCO Responder shall repeat this process until inbound or outbound PCO traffic becomes unblocked.
- 30 • If inbound PCO traffic is blocked, but outbound PCO traffic is unblocked, then the PCO Responder shall not transmit the PCO RNR NAK and discard the N packet. If outbound PCO traffic remains unblocked, Cs will eventually be transmitted, freeing up request resources for servicing Ns.

**Developer Note:** An RNR NAKed N is effectively passed by all inbound PCO packets until the Requester retransmits the N. This does not violate PCIe ordering rules, which permit all packet types to pass Ns.

### 18.2.2.2. Routing Deadlock Avoidance for PCO

- 5 For routing deadlock avoidance in topologies containing packet-relay components, VCs with PCO enabled shall be configured to avoid cyclic resource dependencies in topologies and routing algorithms. This is no different from VCs with PCO disabled, other than the specific routing algorithms usually being different. See *Switches*.

### 18.2.3. Rules for PCO Requesters

- 10 When PCO is enabled on a PCO endpoint, the following special rules apply to Requesters issuing request packets on a PCO stream:
- When sending any Data Space write request packet, the Requester shall use a version of the request packet where the Responder will never send a response packet, ACK or NAK. For example, a Core 64 Write request packet shall have UR = 1b.
  - When sending any interrupt request packet, the Requester shall use a version of the request packet where the Responder will never send an ACK or NAK. For example, a Core 64 Interrupt request packet shall have UR = 1b.
  - If the Requester sends an atomic request packet and receives a successful response packet, the Requester shall consider the request packet complete. The Requester shall not initiate a corresponding *Non-Idempotent Request Release (NIRR)* packet.

**Developer Note:** PCO's reliable delivery semantics make the NIRR packet for atomics and a Forward Progress Screen (FPS) for atomic resources at the Responder unnecessary.

PCO streams inherently deliver Posted Requests and Non-Posted Requests reliably, so Requester retransmission timers for PCO requests are used to determine when a Non-Posted Request fails, not when it needs to be retransmitted.

- Requester retransmission timers in VCs with PCO enabled should be configured with a value higher than the maximum round-trip latency for a Non-Posted Request and its Completions.
- Developer Note:** Determining suitable timeout values for PCO VCs can be challenging, since one-way latencies for PCO streams are not well bounded, given the no-drop policy for PCO packet relay. Among other things, the maximum latency will be a function of fan-in, VC arbitration policies, and the amount of PCO VC buffering. However, since these timeouts only determine when a Non-Posted Request is deemed to have failed, they usually do not need to be aggressively minimized.
- Requester retransmission timers shall never be used for Posted Requests.
  - If a Requester retransmission timer expires, then the Requester shall not retransmit the associated Non-Posted Request. The Requester shall fail the request packet, and shall take the actions specified in *Unicast Reliable Delivery* for the case when the retry counter is zero, noting that PCO streams never have alternative paths.

## 18.2.4. Rules for PCO Responders

When PCO is enabled on a PCO endpoint, the following special rules apply to Responders receiving request packets on a PCO stream:

- A Responder shall determine if a received request packet is a PCO packet based on the VC of the received packet being enabled for PCO communications.
- As described in *Protocol Deadlock Avoidance for PCO*, a Responder shall implement a *Forward Progress Screen (FPS)* dedicated for reserved PCO request resources.
- A Responder that supports atomics shall reserve one or more atomic resources for PCO atomic use. When processing a received PCO atomic request packet, the Responder shall wait until an atomic resource becomes available, and shall never send an RNR NAK because of no atomic resources being available. The Responder shall consider the PCO atomic operation to be complete as soon as it transmits the associated response packet.
- PCO shall not support request types that require *Forward Progress Screen (FPS)* other than the one for PCO protocol deadlock avoidance. For example, Buffer operations are not supported.
- If a Responder that supports atomics is capable of receiving non-PCO atomic requests concurrently with PCO atomic requests, then the Responder shall reserve one or more atomic resources for non-PCO atomic use in order to guarantee forward progress.

## 18.2.5. Topology Considerations for PCO

### 18.2.5.1. Single-Host Topologies

*Simple Single-Host Topologies* illustrates some example single-host topologies envisioned for common use by PCO communications. The one on the left contains an arbitrary number of LPD components, each directly connected to the LPD host. The one on the right adds an arbitrary number of fan-out switches.

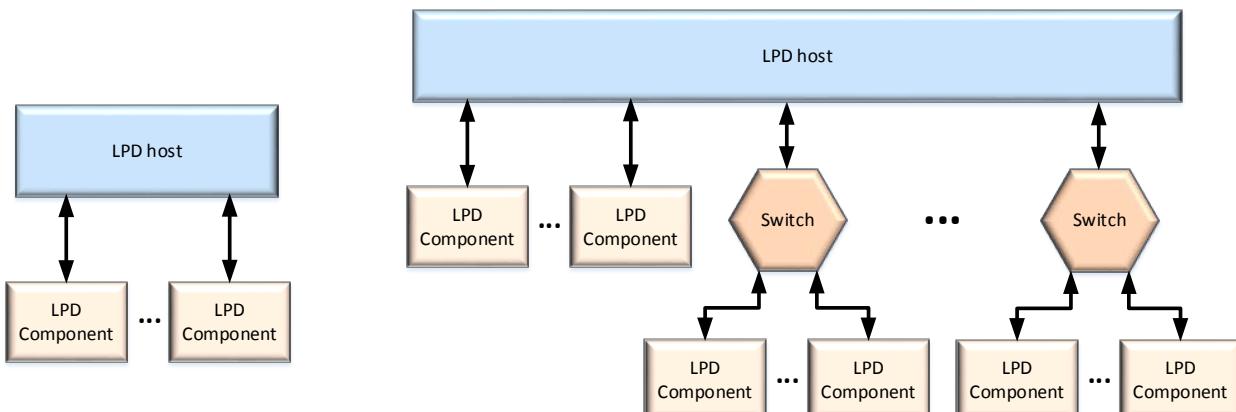


Figure 18-4: Simple Single-Host Topologies

*Single-Host Non-Tree Topologies* illustrates some example single-host non-tree topologies. Though each individual PCO stream can take only a single path, different PCO streams can take different paths, enabling higher aggregate bandwidth and more flexible static load balancing. For LPD hosts and switches doing aggregation, this has the additional benefit of not requiring wider link widths with complex bifurcation options, which are harder to implement and configure.

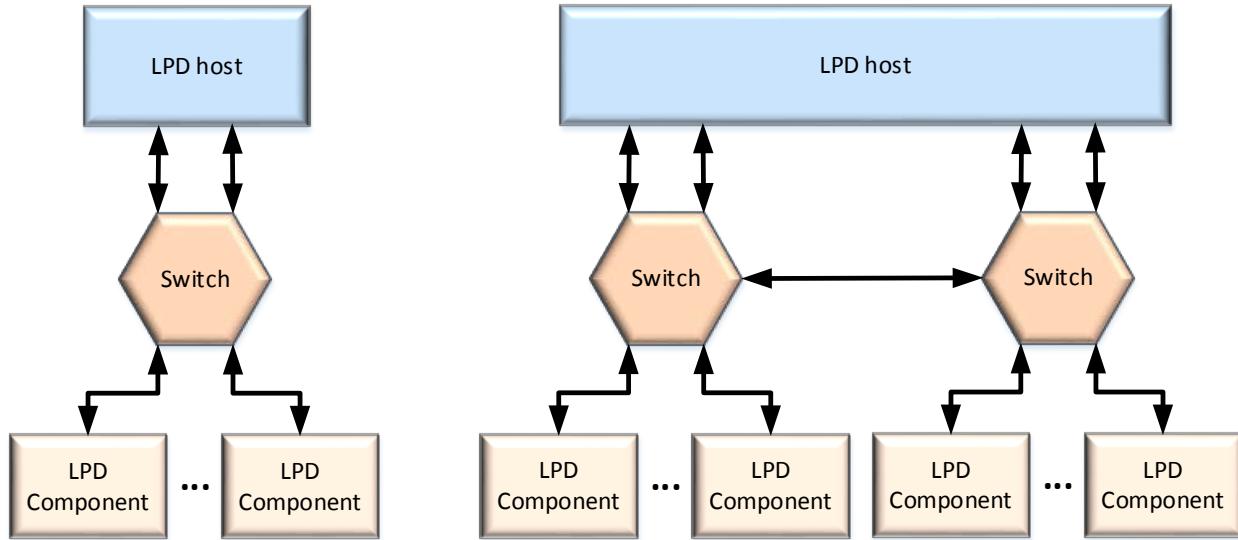


Figure 18-5: Single-Host Non-Tree Topologies

**Developer Note:** Since PCO packets cannot be dropped as a means to manage fabric congestion, PCO streams in certain topologies can incur severe congestion and non-deterministic latency, especially topologies with high fan-in traffic. PCIe fabrics have these same issues. To help mitigate these concerns for PCO topologies, the following approaches are recommended:

- Limit link oversubscription; e.g., reduce the number of LPDs sharing a single link to the LPD host by provisioning additional links to the host using non-tree topologies.
- Limit the number of switch hops for PCO stream paths, especially for PCO streams where more deterministic latency is required.

### 18.2.5.2. Multi-Host Topologies

*Simple Multi-Host Topologies* illustrates some example multi-host topologies envisioned for common use by PCO communications. The one on the left illustrates an arbitrary number of LPD hosts and LPD components connected by a single switch. The one on the right illustrates them connected by multiple switches. Though this figure illustrates a tree topology for each host, non-tree topologies similar to those illustrated in *Single-Host Non-Tree Topologies* are possible as well, and offer the same advantages over tree topologies.

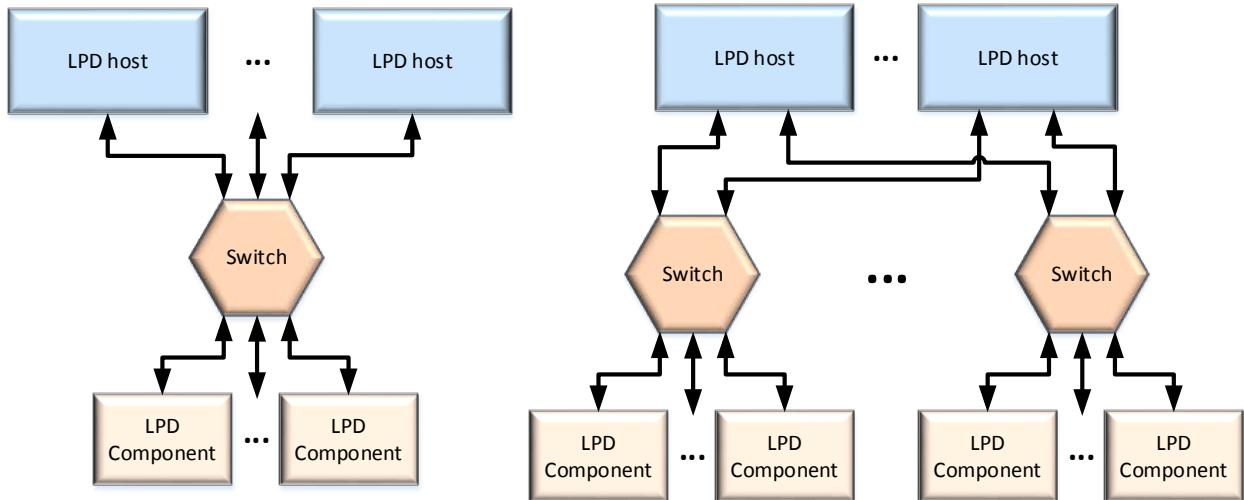


Figure 18-6: Simple Multi-Host Topologies

#### 18.2.5.3. PCO Stream Ordering Domains

**Developer Note:** PCO supports the producer-consumer ordering model only on a per-PCO-stream basis; that is, only between pairs of PCO endpoints. The producer, the consumer, the flag, the status, and the data are required to reside on those two PCO endpoints. Architecturally, PCO could be made to work with more than 2 PCO endpoints in a single producer-consumer ordering domain, but this would place restrictions on switch internal architecture, basically requiring each switch port to behave as having a single ingress queue and a single egress queue, as illustrated in Single-Queue Switch Architecture.

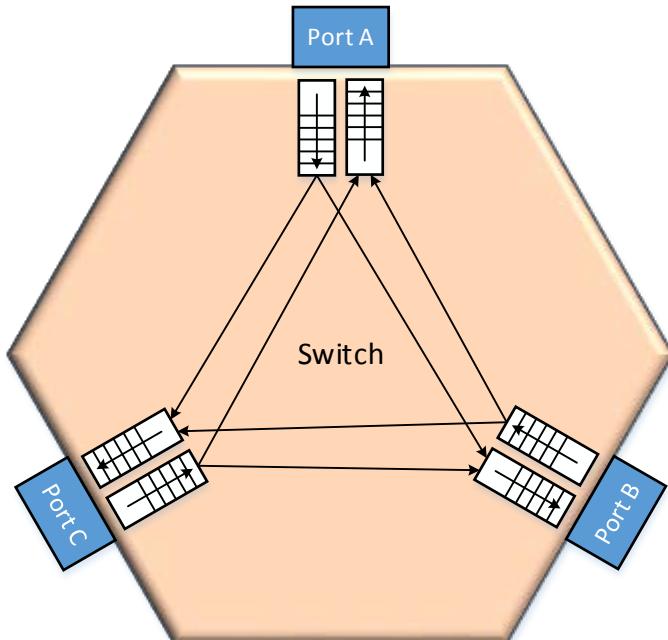


Figure 18-7: Single-Queue Switch Architecture

However, modern high-performance switch designs commonly use multiple ingress queues and/or multiple egress queues at each port, in order to avoid head-of-line blocking (HOLB) and enable other

optimizations. Virtual Output Queue (VOQ) architectures like the one illustrated in Example VOQ Switch Architecture are a common example.

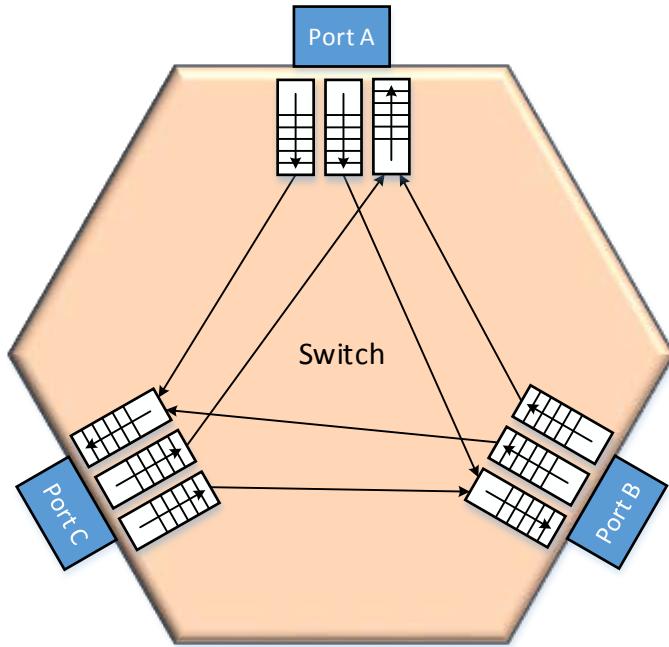


Figure 18-8: Example VOQ Switch Architecture

- 5 *PCIe-compatible ordering when the ordering domain involves 3 or more switch ports requires HOLB in order to work correctly. This is why PCO supports the producer-consumer ordering model only between pairs of PCO endpoints. Supporting more than 2 PCO endpoints in a single producer-consumer ordering domain is outside of this specification's scope.*

### 18.3. PECAM Operation and Requirements

- 10 The PECAM uses a flat memory-mapped address space to access the PCI Configuration Space of a LPD as illustrated in *Conceptual PECAM Access to Local PCI Device Configuration Space*. In this figure, each assigned PCI BDF (bus, device, and function number) is mapped to a destination component and a Control Space address that points to a PCI Type 0 Configuration Space Header.

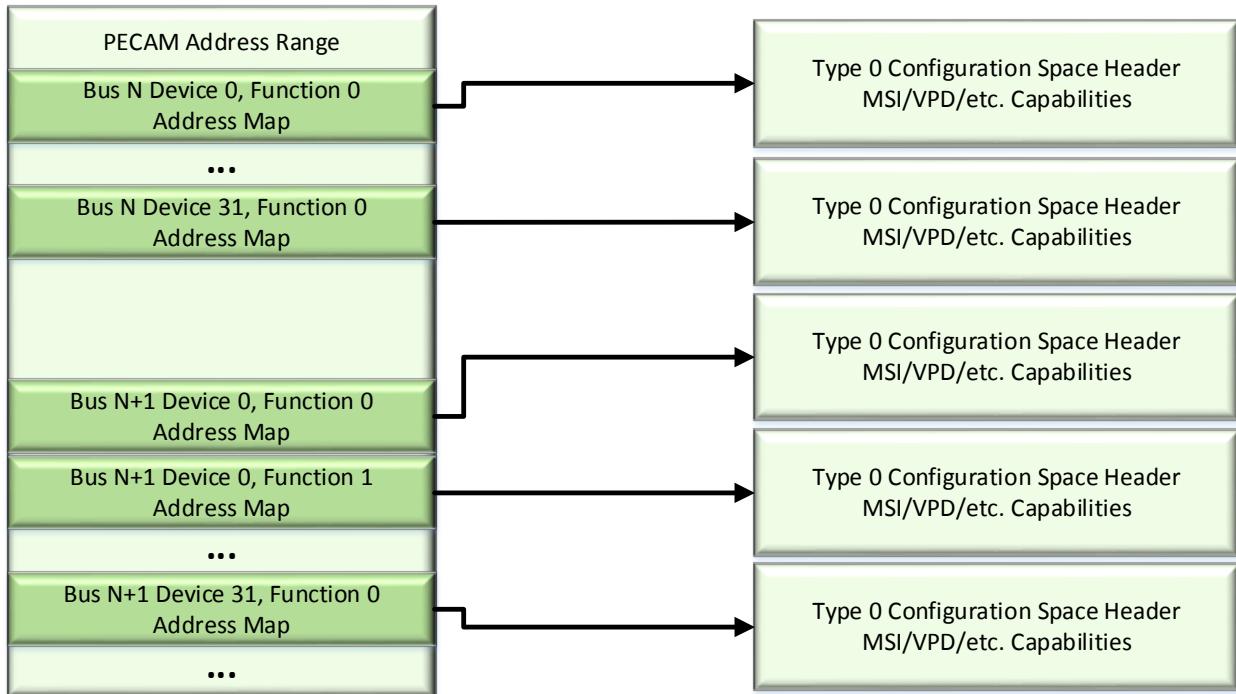


Figure 18-9: Conceptual PECAM Access to Local PCI Device Configuration Space

Systems may implement multiple PCI Segment Groups (PSGs) as defined in the *PCI Firmware Specification*.

5 **Developer Note:** For many systems, a single PSG, which supports up to 256 PCI bus numbers, can be sufficient.

Systems may implement multiple PECAMs, and this might be necessary in order to support multiple PSGs and/or multiple ACPI Host Bridges within a single PSG.

10 A single PECAM may map up to all 256 bus numbers within a PSG, or may map a smaller range of bus numbers. The base address of each PECAM shall be aligned to a minimum of a  $2^{20}$  byte address boundary, but may require alignment to a higher power-of-2 byte address boundary, depending upon the design of the PECAM and system firmware.

15 System firmware shall provide mechanisms to report the existence of PSGs and PECAMs, along with their associated parameters, to the OS. This should be done using mechanisms defined in the PCI Firmware and ACPI Specifications, but may be done by platform-specific means.

To translate a component read or write access request, a memory address is mapped to a PCI Configuration Space as illustrated in *PECAM Address Mapping*.

Table 18-2: PECAM Address Mapping

Memory Address	PCI Configuration Space
Address[27:20]	bus number
Address[19:15]	device number
Address[14:12]	function number
Address[11:8]	Extended Register Number

Memory Address	PCI Configuration Space
Address[7:2]	Register Number
Address[1:0]	DWORD Offset

PECAM functionality should be implemented using the Requester ZMMU, but may be implemented using a platform-specific mechanism. *Conceptual Extension of PCI Express ECAM Logic to Support PECAM Logic* illustrates how BDF number mappings performed by existing PCI Express ECAMs can be handled by a Requester ZMMU, and how the remaining fields translate into a Control Space offset within the targeted component. The Requester ZMMU maps a 4 KiB page to Gen-Z Control Space for each BDF.

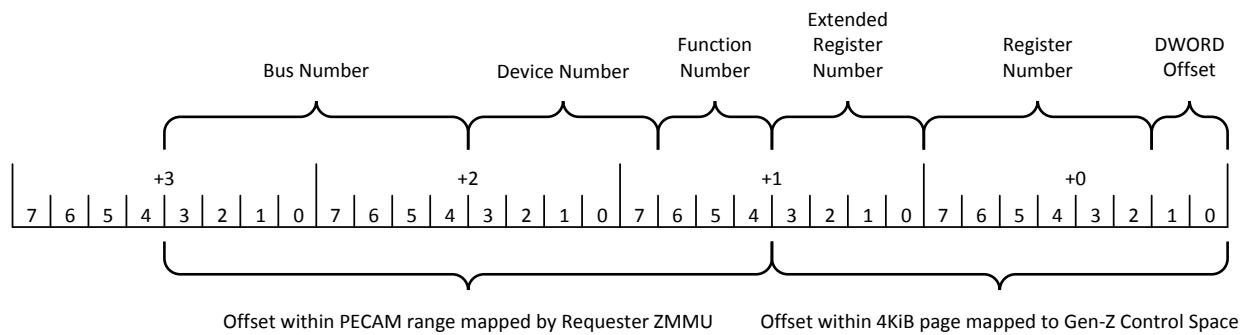


Figure 18-10: Conceptual Extension of PCI Express ECAM Logic to Support PECAM Logic

If any given BDF number tuple is not mapped to a LPD, then the PECAM logic shall not generate a Control Read or Control Write request packet. Instead, the PECAM logic shall handle software accesses with PCI Master Abort semantics, indicating that no LPD is present. I.e., reads shall return a value of all 1s, and writes shall be dropped silently. If using a Requester ZMMU, then this can be achieved by configuring the D-ATTR field in the Requester PTE to indicate “Invalid Entry” (see *Requester PTE Fields*).

*PECAM Mapping to a Component and PCI Configuration Space in Control Space* illustrates a Gen-Z I/O component that supports the *Component LPD Structure*. This structure is accessed using a Control Structure Pointer in the *Core Structure* or the *Component Extension Structure*. A PECAM is configured such that a load or store operation (e.g., issued by a processor) to a mapped address results in the generation of a Control Read or Control Write request packet (see *Control Read and Control Write Packets*) to the component’s Control Space address that corresponds to an address within the LPD’s Configuration Space.

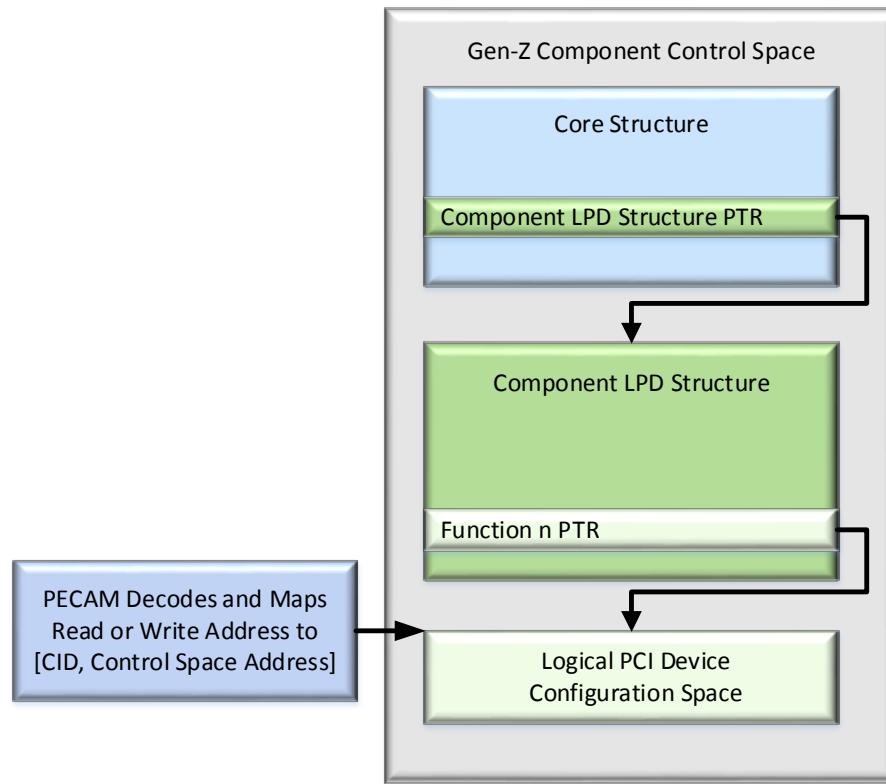


Figure 18-11: PECAM Mapping to a Component and PCI Configuration Space

*Example PECAM Enumeration and Configuration Steps* illustrates the conceptual steps taken by software (e.g., firmware) when it detects a Gen-Z processor component that supports one or more PECAMs.

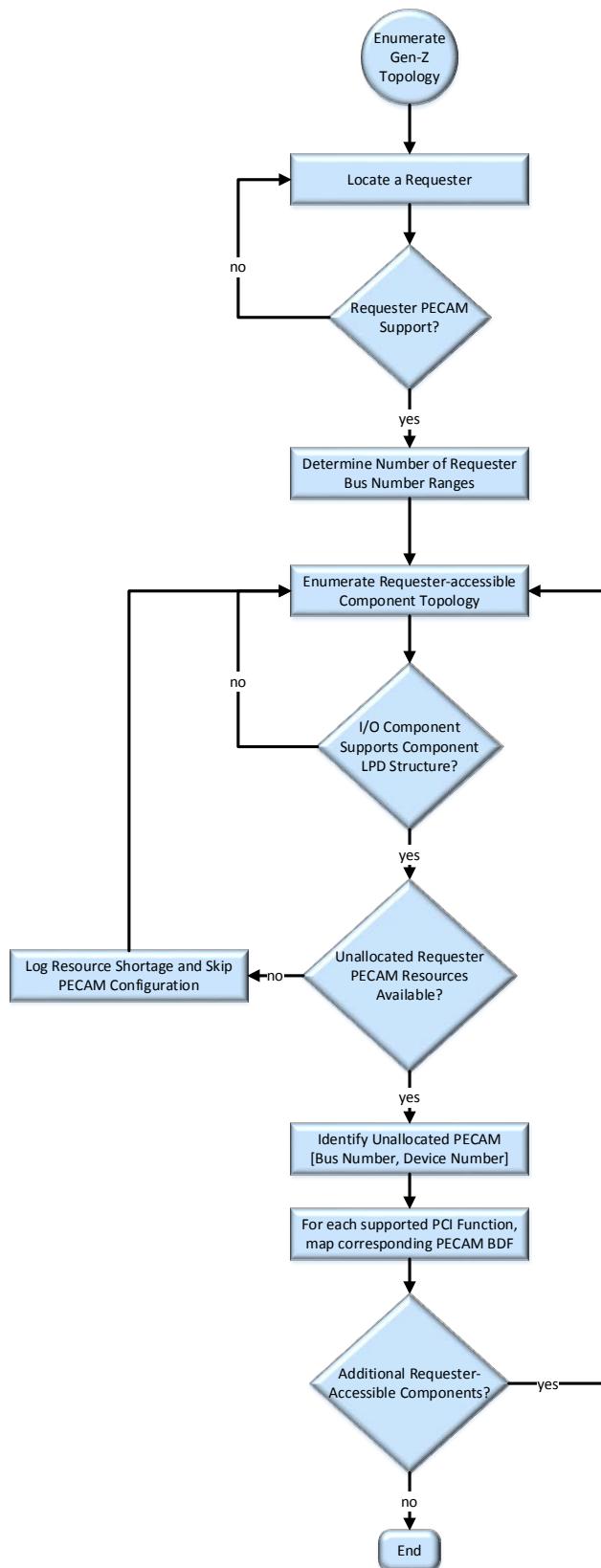


Figure 18-12: Example PECAM Enumeration and Configuration Steps

Table 18-3: PECAM Enumeration and Configuration Details

Processing Step	Explanation
<b>Locate a Requester</b>	<i>In-band Discovery and Initialization</i> covers how new components are detected. As new components are detected, software determines if a component is a Requester (e.g., a SoC) or a Responder.
<b>Requester PECAM Supported?</b>	System firmware discovers and configures PECAMs using platform-specific mechanisms. If one or more PECAM bus number ranges are supported, then determine the number of bus number ranges and proceed to enumerate additional Gen-Z components, else proceed to the next component.
<b>Enumerate Requester-accessible Topology</b>	Locate each component that is accessible by this Requester. If software has configured accessible peer components using the <i>Component Destination Table Structure</i> , then access each accessible component.  If the Component Destination Table structure is unsupported, then software should follow the steps described in <i>Out-of-band Management</i> or <i>In-band Discovery and Initialization</i> to identify accessible components.
<b>I/O Component Supports Component LPD Structure?</b>	If the component does not support this structure, then it isn't a candidate for PECAM configuration.
<b>Unallocated Requester PECAM Resources Available?</b>	If available PECAM resources are consumed, then log the resource shortage and continue enumerating Gen-Z components.
<b>Identify Unallocated PECAM [bus number, device number]</b>	Identify a free LPD mapping entry set—an unallocated mapping entry set consists of a bus number-device number pair (8 entries) if ARI is not supported, and an entire bus number (256 entries) if ARI is supported.
<b>For each supported LPD Function, map a corresponding BDF entry</b>	The <i>Component LPD Structure</i> contains one PCI Configuration Space instance per supported LPD Function. For each supported LPD Function, map the corresponding BDF PECAM entry with component's CID and a pointer to the Control Space address corresponding to byte zero of PCI Configuration Space.
<b>Additional Gen-Z Components?</b>	Continue to enumerate Gen-Z components until all additional components have been evaluated and configured.

If software is informed that a new component has been detected, then software initiates component configuration. During component configuration, software identifies the component type, whether it supports PECAM. If the component is a Requester that supports PECAM, then software configures the component's PECAM with the mapped LPDs. If the component supports the *Component LPD Structure*, then software configures the component and presents the component to the corresponding operating system as a LPD.

I/O components that support a *Component LPD Structure* shall decode all address bits used to access its PCI Configuration Space.

## 18.4. LPD Configuration Space

*PECAM Mapping to Control Space and Data Space* illustrates how a PECAM maps the PCI Configuration Space of a Function in a LPD, which is referred to as a LPD Function in this section.

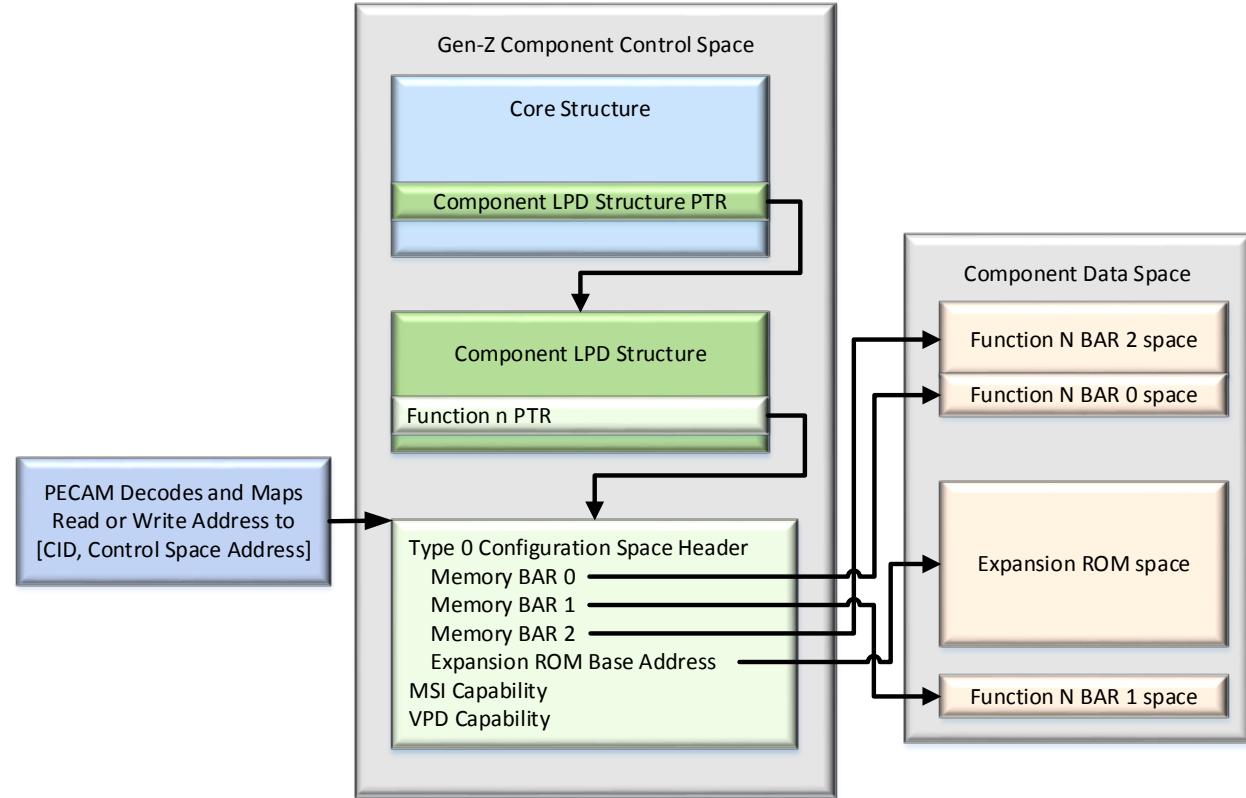


Figure 18-13: PECAM Mapping to Control Space and Data Space

In configuring a LPD Function's Configuration Space, system firmware or Gen-Z-aware OSs may modify certain HwInit fields by first Setting the Function's associated HwInit Write Enable bit in the *Component LPD Structure*. System firmware should clear all HwInit Write Enable bits before OS handoff, so that OSs unaware of Gen-Z will not be able to modify these HwInit fields.

### 18.4.1. PCI Type 0 Configuration Space Header

The PCI Type 0 Configuration Space Header shall be implemented and configured as specified in the *PCI Express Base Specification* except as follows:

- 15 • Within the Command Register:
  - I/O Space Enable shall be hardwired to 0b.
  - Parity Error Response shall be hardwired to 0b.
  - SERR# Enable shall be hardwired to 0b.
  - Interrupt Disable shall be hardwired to 0b. INTx shall not be supported.
- 20 • Within the Status Register:

- Capabilities List shall be hardwired to 1b.
- All other fields shall be hardwired to 0b.
- The following fields shall be hardwired to 0x0:
  - Revision ID Register
  - Latency Timer Register
  - BIST Register
  - Interrupt Line Register
  - Interrupt Pin Register
  - MIN\_GNT
  - MAX\_LAT
- The following field shall be implemented as specified in *PCI Code and ID Assignment Specification*:
  - Class Code Register
- The Cache Line Size Register shall be configured by system firmware, though its setting shall not affect the operation of the component.
- A LPD Function shall support 1-6 PCI Memory Base Address registers (BARs) that enable Memory-Mapped I/O (MMIO) into Component Data Space mapped by the Requester ZMMU. Those ranges are used for the LPD Function's run-time CSRs and the MSI-X Capability if present and configured. If the LPD Function supports an Expansion ROM, then the Expansion ROM Base Address register shall enable mapping it in a similar manner. See *LPD MMIO Space* for extensive additional details and requirements.
- There shall be no Base Address registers for PCI I/O Space.

#### 18.4.2. PCI Type 1 Configuration Space Header

LPD Function support for Type 1 Configuration Space Headers is outside the scope of this specification.

**Developer Note:** *PCI bridges and PCIe Switches implemented as LPDs is not architected since Gen-Z fabrics use a totally different paradigm for routing, and there's not a 1:1 mapping for a bridge to convert Gen-Z packets to PCIe packets or vice-versa.*

The *Logical PCI Hierarchy (LPH)* mechanism enables a Gen-Z component to present a small set of native PCIe devices to a host. Within an LPH hierarchy, native PCIe functions with Type 1 Configuration Space headers (i.e., PCIe Root Ports and PCIe switch ports) are exposed, and peer-to-peer communication between PCIe devices within the hierarchy can be supported.

#### 18.4.3. PCI Capability Structures

Supported PCI Capability structures are specified in the *PCI Express Base Specification*. *PCI Capability Structure Support Levels* lists those capability structures along with their PCI-Compatible Capability ID and level of support in LPDs.

- 35 Each LPD Function shall implement the following PCI Capability structures:
  - MSI or MSI-X

Each LPD Function may implement PCI Capability structures that have a support level other than "None".

LPD Function support for PCI Capability structures having a support level of “None” is outside the scope of this specification.

Table 18-4: PCI Capability Structure Support Levels

Support Level	Capability ID	PCI Capability Name
Full	00h	Null Capability
Near-Full	01h	PCI Power Management
None	02h	AGP
Full	03h	VPD
None	04h	Slot Identification
Full	05h	Message Signaled Interrupts (MSI)
None	06h	CompactPCI Hot Swap
None	07h	PCI-X
None	08h	HyperTransport
Full	09h	Vendor Specific
None	0Ah	Debug port
None	0Bh	CompactPCI central resource control
None	0Ch	PCI Hot-Plug
None	0Dh	PCI Bridge Subsystem Vendor ID
None	0Eh	AGP 8x
None	0Fh	Secure Device
Partial	10h	PCI Express
Full	11h	MSI-X
None	12h	Serial ATA Data/Index Configuration
Full	13h	Advanced Features (AF)
None	14h	Enhanced Allocation (EA)
None	15h	Flattening Portal Bridge (FPB)

**Developer Note:**

- The support level for the PCI Power Management Capability is categorized as “Near-Full” since the Aux\_Current field is not supported.
- The support level for the PCI Express Capability is categorized as “Partial” since only functionality applicable to Type 0 Header Functions (Endpoints) is supported.

Additional requirements:

- Each LPD Function shall support an MSI or an MSI-X capability.
- Function 0 of a LPD may support the VPD capability (if present, Functions 1-7 may support the VPD capability). The VPD capability shall be as specified in *PCI Express Base Specification*.
- Other PCI capabilities may be present; e.g., a Vendor-Specific Capability, for managing vendor-specific functionality.

5

#### 18.4.4. PCI Express Extended Capability Structures

Supported PCI Express Extended Capability structures are specified in the *PCI Express Base Specification*. *PCI Express Extended Capability Structure Support Levels* lists those capability structures along with their Extended Capability ID and level of support in LPDs.

10

Each LPD Function may implement PCI Express Extended Capability structures that have a support level other than “None”.

LPD Function support for PCI Express Extended Capability structures having a support level of “None” is outside the scope of this specification.

Table 18-5: PCI Express Extended Capability Structure Support Levels

Support Level	Extended Capability ID	PCI Express Extended Capability Name
Full	0000h	Null Capability
Partial	0001h	Advanced Error Reporting (AER)
None	0002h	Virtual Channel (VC)
Full	0003h	Device Serial Number
Full	0004h	Power Budgeting
None	0005h	Root Complex Link Declaration
None	0006h	Root Complex Internal Link Control
None	0007h	Root Complex Event Collector Endpoint Association
None	0008h	Multi-Function Virtual Channel (MFVC)
None	0009h	Virtual Channel (VC)
None	000Ah	Root Complex Register Block (RCRB) Header
Full	000Bh	Vendor-Specific Extended Capability (VSEC)
None	000Ch	Configuration Access Correlation (CAC)
None	000Dh	Access Control Services (ACS)
None	000Eh	Alternative Routing-ID Interpretation (ARI)
Full	000Fh	Address Translation Services (ATS)

Near-Full	0010h	Single Root I/O Virtualization (SR-IOV)
None	0011h	Multi-Root I/O Virtualization (MR-IOV)
None	0012h	Multicast
Full	0013h	Page Request Interface (PRI)
None	0014h	Reserved for AMD
None	0015h	Resizable BAR
Full	0016h	Dynamic Power Allocation (DPA)
Near-Full	0017h	TLP Processing Hints (TPH)
None	0018h	Latency Tolerance Reporting (LTR)
None	0019h	Secondary PCI Express
None	001Ah	Protocol Multiplexing (PMUX)
Full	001Bh	Process Address Space ID (PASID)
None	001Ch	LN Requester (LNR)
None	001Dh	Downstream Port Containment (DPC)
None	001Eh	L1 PM Substates
None	001Fh	Precision Time Measurement (PTM)
None	0020h	PCI Express over M-PHY (M-PCIe)
None	0021h	FRS Queueing
None	0022h	Readiness Time Reporting
Full	0023h	Designated Vendor-Specific Extended Capability
None	0024h	VF Resizable BAR
None	0025h	Data Link Feature
None	0026h	Physical Layer 16.0 GT/s
None	0027h	Lane Margining at the Receiver
None	0028h	Hierarchy ID
Partial	0029h	Native PCIe Enclosure Management (NPEM)

**Developer Note:**

- The support level for the Advanced Error Reporting (AER) Capability is categorized as “Partial” since only functionality applicable to Type 0 Header Functions (Endpoints) is supported.

- The support level for the Single Root I/O Virtualization (SR-IOV) Capability is categorized as “Near-Full” since multiple bus numbers and MR-IOV-related functionality is not supported.
- The support level for the TLP Processing Hints (TPH) Capability is categorized as “Near-Full” since Extended Steering Tags functionality is not supported.
- The support level for the Native PCIe Enclosure Management (NPEM) Capability is categorized as “Partial” since only functionality applicable to Type 0 Header Functions (Endpoints) is supported.

**Developer Note:** When leveraging a PCIe hardware design to implement an LPD, capability structures in the PCIe design that will not be supported in the LPD can be “nullified” simply by changing their Extended Capability ID to that of the Null Capability (0000h).

## 18.5. LPD MMIO Space

Each LPD has Memory-Mapped I/O (MMIO) Space that is mapped by one or more Base Address registers (BARs) in the *PCI Type 0 Configuration Space Header* for each of its functions. The Expansion ROM Base Address register also counts as a BAR. LPD BARs support only PCI Memory Space, not PCI I/O Space. System firmware initially configures LPD BARs, and all addresses written to LPD BARs are in PCI Memory Space.

Host physical address (PA) space is managed by host system firmware and the OS. With native PCI/PCIe, the PCI/PCIe Host Bridge maps MMIO space into host PA space. With LPDs, the Requester ZMMU indirectly maps MMIO space into host PA space.

**Developer Note:** Device-side and host-side MMIO mappings configured by system firmware need to remain consistent with each other. Without special accommodations, if an OS that is unaware of Gen-Z reprograms LPD BARs, then this could lead to inconsistencies between the Requester ZMMU mappings and the BAR mappings, resulting in undefined behavior. This section details LPD MMIO apertures, how Gen-Z-aware software configures MMIO mapping mechanisms for LPDs, and how this accommodates an OS that is unaware of Gen-Z modifying LPD BARs without leading to inconsistencies.

PCI bridges, including the virtual ones in PCIe Root Ports and PCIe Switches, support 2 MMIO apertures in PCI Memory Space: a 32-bit aperture with non-prefetchable semantics referred to here as MMIOl (“low”), and a 64-bit aperture with prefetchable semantics referred to here as MMIOh (“high”). While LPDs contain no physical or virtual PCI bridges, they take advantage of PCI/PCIe software infrastructure support for these two apertures.

As illustrated in *MMIOl and MMIOh Apertures in Data Space*, the registers or address ranges in a LPD mapped by its BARs into MMIO space natively reside in 64-bit Gen-Z Data Space, which is unique and zero-based for each Gen-Z component. Non-prefetchable LPD BARs map their resources into the MMIOl aperture, whose base in Gen-Z Data Space (GZ-DS) and size are reported by read-only registers in the LPD’s *Component LPD Structure*. Similarly, prefetchable LPD BARs map their resources into the MMIOh aperture, whose base in GZ-DS and size are also reported by read-only registers in the LPD’s *Component LPD Structure*. The figure illustrates example BAR spaces that could be associated with a single LPD function. If the LPD supports multiple functions, their BARs’ spaces share these same apertures.

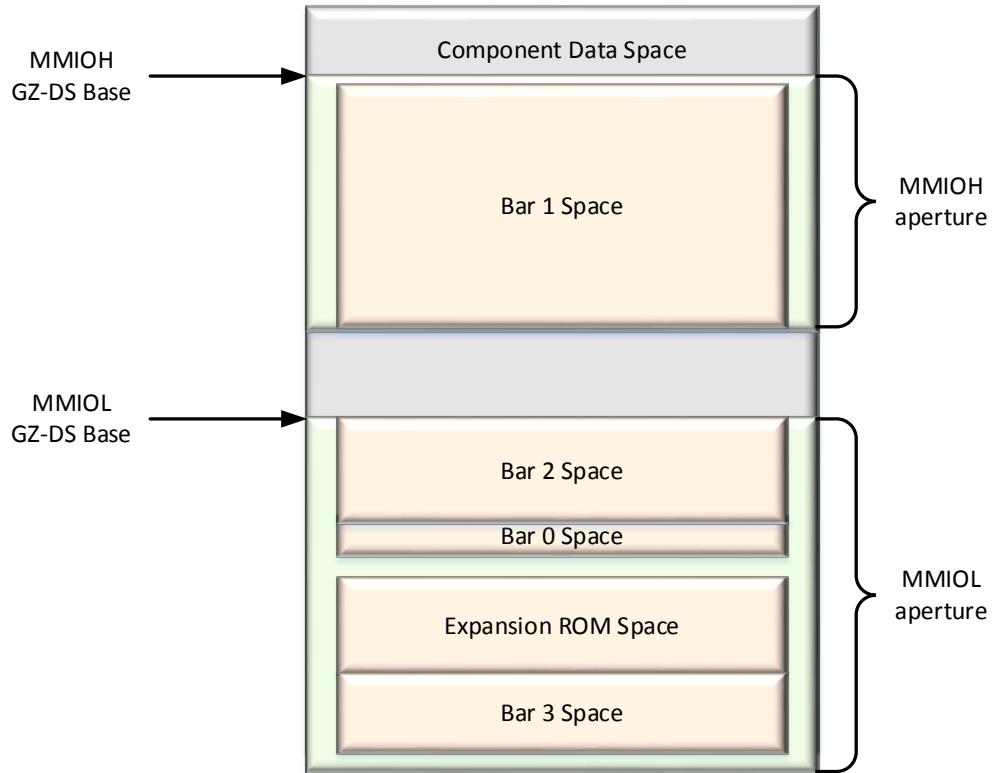


Figure 18-14: MMIOL and MMIOH Apertures in Data Space

The following are hardware requirements for the MMIOL and MMIOH apertures:

- If a component implements multiple LPDs, each LPD shall have its own unique set of MMIOL and MMIOH apertures. This enables different LPDs to be assigned to different hosts.
- The MMIOL size and GZ-DS Base alignment shall be sufficient to accommodate the entire set of non-prefetchable BAR and Expansion ROM spaces associated with all of the LPD's functions.
- The MMIOH size and GZ-DS Base alignment shall be sufficient to accommodate the entire set of prefetchable BAR spaces associated with all of the LPD's functions.
- If a given aperture has no associated BARs, then the aperture's size and GZ-DS Base shall be 0.
- Consistent with PCI Bridge requirements, each aperture's GZ-DS Base shall be aligned to a 1-MiB boundary.
- Consistent with PCI Bridge requirements, each aperture's size shall be an integer multiple of 1 MiB.
- To enable compatibility with software BAR configuration algorithms that configure BARs with the largest size first, the GZ-DS Base for each aperture shall be aligned to a boundary equal to its largest BAR size.
- To enable efficient use of Requester ZMMU mapping resources, if an aperture's size is 32 MiB or greater, then the GZ-DS Base for it should be aligned to a 32-MiB boundary, matching the largest mandatory page size for Requester ZMMUs.

The following are hardware requirements for LPD BARs:

- BARs shall request only Memory Space, not I/O Space.
- Prefetchable BARs shall be 64-bit, consistent with PCI Express Endpoint requirements.
- Each BAR shall be capable of mapping any properly-aligned space within its associated aperture.

- The Expansion ROM BAR shall be associated with the MMIO aperture. If the Expansion ROM BARs in multiple LPD functions are programmed with overlapping address ranges, then only one Expansion ROM BAR with the an overlapping address range shall be enabled at a time.

5 The following are requirements for system firmware or Gen-Z-aware OS software, collectively referred to here as “software”.

- Software shall assign one LPD per PCI bus number, and shall associate a single ACPI Host Bridge with that PCI bus number.
- If sufficient PCI Memory Space is available, then software shall allocate it for the MMIO and MMIOH apertures, matching their size requests as reported in the LPD’s *Component LPD Structure*.
  - For each aperture, software shall not allocate more PCI Memory Space than requested, except to accommodate a Requester ZMMU page size.
  - If insufficient PCI Memory Space is available, then software may allocate less than requested.
  - If software does not allocate the requested Memory Space for every BAR in a given LPD function, then software should clear the function’s Memory Space Enable bit.
- System firmware shall report all LPD MMIO mappings, including any offsets, to the OS. This should be done using mechanisms defined in the PCI Firmware and ACPI Specifications, but may be done by platform-specific means. With ACPI, offsets between host PA addresses and PCI Memory Space addresses with LPD MMIO mappings may be reported using the \_TRA resource descriptor field, enabling OS software to do the necessary translation.
- System firmware shall direct the OS to honor the MMIO and MMIOH apertures, so that an OS that is unaware of Gen-Z will not attempt to configure the LPD’s BAR spaces outside of them. This should be done using mechanisms defined in the PCI Firmware and ACPI Specifications, but may be done by platform-specific means. With ACPI, this may be done using the \_CRS control method within the namespace of the associated ACPI Host Bridge.
- For each aperture, software shall program the PCI-MS Base register in the *Component LPD Structure* with the address assigned in PCI Memory Space.
- Software shall configure non-prefetchable BAR space only into the MMIO aperture, and prefetchable BAR space only into the MMIOH aperture.
- Software shall configure Expansion ROM BAR space only into the MMIO aperture.
- For each aperture, if software changes the value in the aperture’s PCI-MS Base register, then software shall reconfigure every BAR in that aperture before further use. This supports hardware implementations that use the “BAR GZ-DS Base” register approach described below.
- For each aperture, software shall map the entire amount of PCI Memory Space allocated for the aperture into host PA space using the Requester ZMMU, starting with the Gen-Z Data Space address reported by GZ-DS Base.

40 **Developer Note:** Each configured LPD BAR contains an address in PCI Memory Space. However, the BAR needs to match against incoming Gen-Z Requests whose addresses are in Gen-Z Data Space. While a specific implementation is not mandated, here is one possible approach. A hidden “BAR GZ-DS Base” register is implemented for each BAR. When software writes a PCI-MS address to the BAR, hardware subtracts the value from the aperture’s PCI-MS Base register (written earlier by software), yielding an offset within the aperture. Hardware adds this offset to the aperture’s GZ-DS Base address, and writes the result to the hidden BAR GZ-DS Base register, which is then used for matching against incoming Gen-Z Requests.

**Developer Note:** The use of the MMIO apertures as described here enables LPDs to handle the case where system firmware configures LPD BARs initially, but an OS unaware of Gen-Z reconfigures the LPD BARs afterwards. The OS is directed to configure each BAR space within its associated aperture. Since the entire aperture is mapped by the Requester ZMMU, moving or shuffling BAR spaces around within that aperture does not cause inconsistencies with the Requester ZMMU mappings.

5

## 18.6. LPD Error Handling

A LPD needs to surface a limited number of errors to the operating system, e.g., the component has failed. Gen-Z communicates errors through the *Core Structure* or, if supported, the *Component Error and Signal Event Structure*. The *Core C-Status* communicates the class and severity of a given error event, but does not provide error-specific details.

10

If an I/O component does not support the *Component Error and Signal Event Structure*, then:

- If *Core C-Status* indicates a Non-Fatal Internal Error was detected, then the component shall continue to perform component-specific operations and may generate an *Unsolicited Event (UE) Packet* to inform management (firmware) of the problem.
- If *Core C-Status* indicates a Fatal Internal Error or a Non-Transient Protocol Error was detected, then the component shall stop all component-specific operations and generate an *Unsolicited Event (UE) Packet* to inform management (e.g., firmware) of the problem.

15

If an I/O component supports the *Component Error and Signal Event Structure*, then:

20

- Management shall be responsible for configuring the *Component Error and Signal Event Structure* to generate an *Unsolicited Event (UE) Packet* when conditions warrant.

A PECAM Requester informs management upon receipt of an *Unsolicited Event (UE) Packet*.

## 18.7. Component LPD Structure

To enable an operating system to configure LPD components using the existing PCI software infrastructure, Gen-Z I/O components need to provide conventional PCI Configuration and Memory Spaces. These spaces are reported and managed via the Component LPD structure, and accessed by an operating system using the PECAM for Configuration Space and Requester ZMMU for Memory Space. A similar structure exists for the *Logical PCI Hierarchy (LPH)* mechanism.

25

The following are the features and requirements of the Component LPD Structure:

30

- A component may support one or more Component LPD structures.
  - Each Component LPD structure and associated LPD Functions shall be associated with a single PECAM Requester, considered to be the LPD's host.
    - A structure shall be associated with a unique [bus number, device number] as viewed by the PECAM Requester if the structure supports up to 8 functions or shall be associated with a unique bus number as viewed by the PECAM Requester if the structure supports more than 8 functions.
  - If multiple Component LPD structures are supported, then:
    - Each additional structure shall be accessed through the Next Component LPD Structure PTR.
    - Multiple structures may be associated with a single PECAM Requester, up to  $2^8 * 2^5$  structures if device numbers are used, and up to  $2^8$  structures if device

35

40

numbers are not used. *PECAM Mapping to Multiple Type 0 Configuration Space Headers* (A) illustrates a single I/O component that supports three Component LPD structures that use device numbers, enabling the component to support 3-24 LPD Functions.

- Depending upon the physical Gen-Z topology, each structure may be associated with a different PECAM Requester. This enables a Gen-Z I/O component to be simultaneously shared by multiple PECAM Requesters. *PECAM Mapping to Multiple Type 0 Configuration Space Headers* (B) illustrates a single I/O component that supports three Component LPD structures. Each structure is associated with a different PECAM Requester. Since [bus number, device number] uniqueness is on a per PECAM Requester basis, global [bus number, device number] uniqueness is not required.
- The Component LPD structure shall use the *Component LPD Structure Format*.
- Each Component LPD structure shall support PCI Function 0, and may support multiple PCI Functions (up to a maximum of 256 functions).
- Management may configure the Default HCID and the Default HSID with the respective CID and SID used to identify the host component to target for LPD read, write, interrupt, etc. operations. If configured, these default values apply to all LPD functions within the LPD.
  - A component may communicate with multiple host or peer components. Configuration and management of these components are outside of this specification's scope.

**Developer Note:** Each LPD Function's Configuration Space is mapped by the PECAM to an integer 4096-byte address multiple. Such mapping does not require that 4096 bytes of Control Space be provisioned for each PCI Configuration Space. Instead, management (e.g., system firmware) configures the PECAM such that the operating system will see an integer 4096-byte multiple address that is mapped to the appropriate Control Space address range.

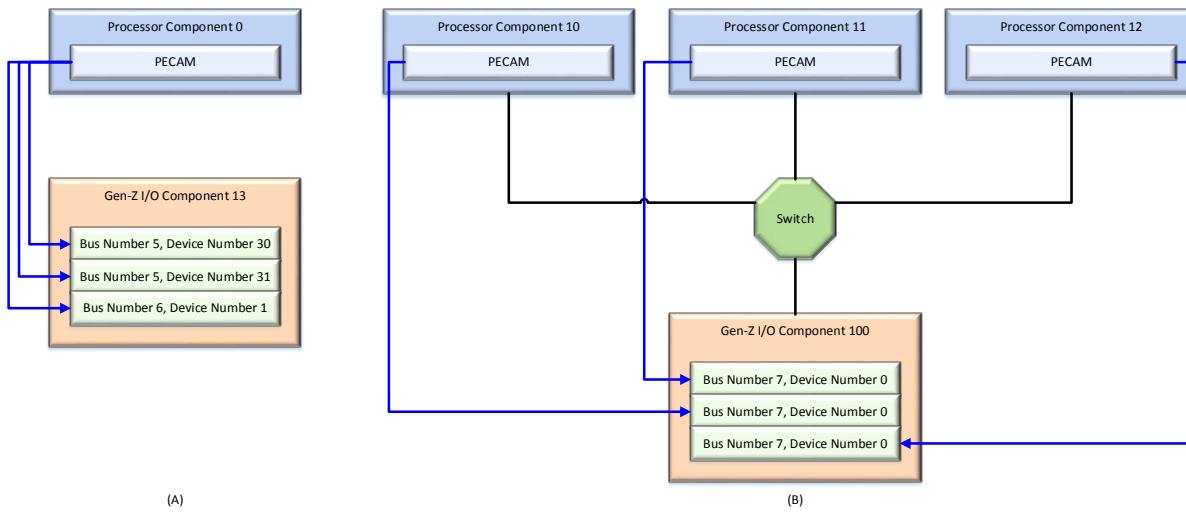


Figure 18-15: PECAM Mapping to Multiple Type 0 Configuration Space Headers

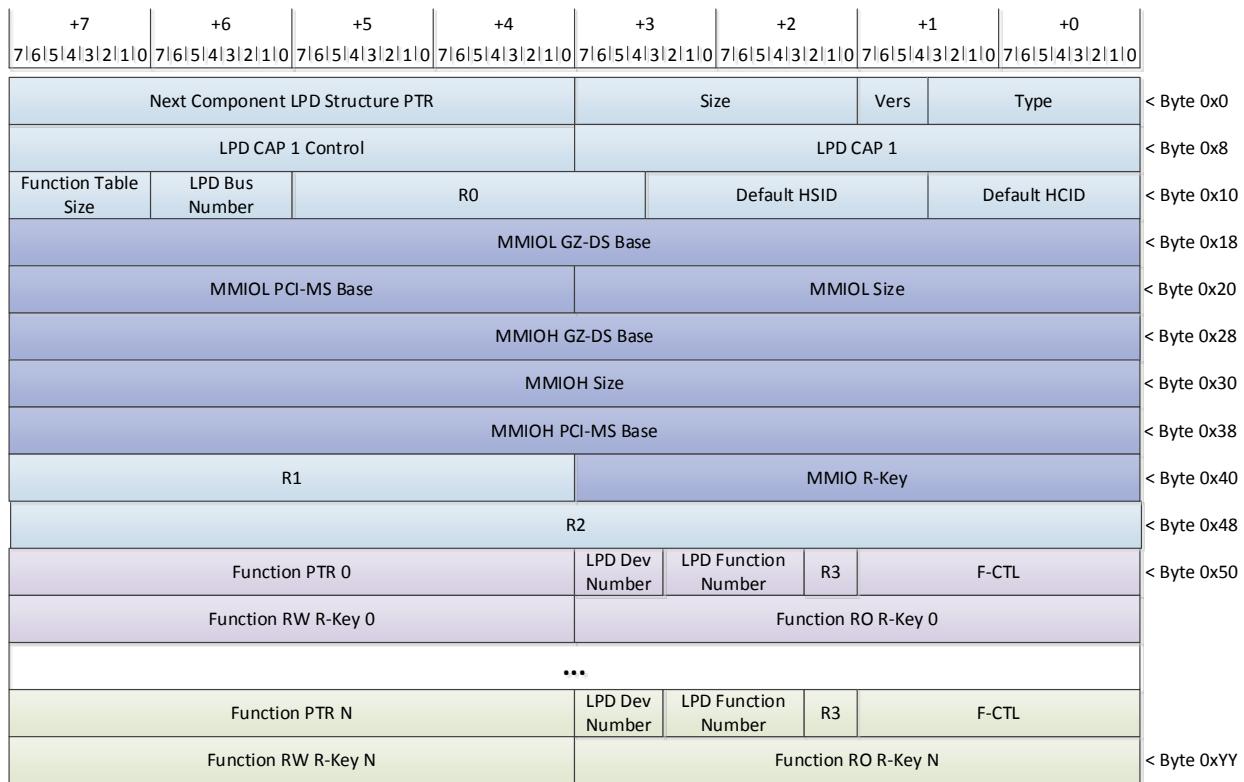


Figure 18-16: Component LPD Structure Format

Table 18-6: Component LPD Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>Version (Vers)</b>	4	0x1	M	RO	Structure Version
<b>Next Component LPD Structure PTR</b>	32	-	O	RO	Indicates if a Component LPD structure is linked to this structure. If Null, then a Component LPD structure was not provisioned.
<b>LPD CAP 1</b>	32	-	M	RO	LPD Capabilities 1
		Bit 0			Multi-subnet Support—Indicates if the component supports multi-subnet LPD operations. If supported, then the Default HSID field may be configured. 0b—Unsupported 1b—Supported
		Bit 1			Non-Default Host ID Support—Indicates if the component supports communication with non-default host components. If supported, then non-default host components are identified and accessed

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
LPD CAP 1 Control	32				through means outside of this specification's scope.  0b—Unsupported 1b—Supported
					PCO Support—Indicates if the LPD supports PCIe Compatible Ordering semantics. PCO support applies to all LPD functions.  0b—Unsupported 1b—Supported
					Device Number Support—Indicates if the LPD Dev Number field is supported, i.e., may be configured with a valid device number value.  0b—Unsupported 1b—Supported
					0b—Unsupported 1b—Supported
		Bits 31:4			RsvdP
LPD CAP 1 Control	32	-	M	RW	LPD Control
		Bit 0			Non-Default Target Enable—if enabled, then any LPD functions associated with this structure may target a component identified and selected through means outside of this specification's scope.  If disabled, then the LPD shall use only the Default HCID and optionally the Default HSID for targeting its host.  0b—Disabled 1b—Enabled
		Bit 1			LPD Reset—if modified, then the contents of this Component LPD structure and all related PCI / PCIe configuration structures shall be returned to their power-on initialized state.  1b—Reset  Reads of this bit shall return 0b.
		Bit 2			Valid Default HSID field—Determines if the Default HSID field has been configured with a valid value.  0b—Not Configured

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Default HCID		Bit 3			1b—Configured LPD Communications Enable—If enabled, then any LPD functions associated with this structure may exchange LPD request and response packets. If disabled, then no LPD functions associated with this structure may exchange LPD request and response packets. 0b—Disabled 1b—Enabled
					RsvdP
	12	-	M	RW	Identifies the CID of the default host component to target LPD interrupt, read, write, and etc. operations.  If enabled via the Non-Default Target Enable bit, an LPD may target components identified through means outside of this specification's scope.
Default HSID	16	-	O	RW	Identifies the SID of the default host component to target LPD interrupt, read, write, and etc. operations.  If enabled via the Non-Default Target Enable bit, an LPD may target non-default components identified through means outside of this specification's scope.  If the component does not support multi-subnet LPD operations, then this field shall be Reserved.
LPD Bus Number	8	-	M	RW	Configured with the bus number associated with all LPD functions provisioned through this structure.
Function Table Size	8	-	M	RO	Indicates the number of provisioned Function Table entries. If Function Table Size == 0x0, then the number of provisioned Function Table entries shall be 256.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
					Functions shall be associated Function Table entries in ascending order, i.e., if a LPD supports functions 0-7, then function 0 is associated with Function Table entry 0, function 1 is associated with Function Table entry 1, and so forth.
<b>MMIOL GZ-DS Base</b>	64	-	M	RO	Indicates the base address of the MMIOL aperture in Gen-Z Data Space
<b>MMIOL Size</b>	32	-	M	RO	Indicates the size of MMIOL in bytes
<b>MMIOL PCI-MS Base</b>	32	-	M	RW	Determines the base address of the MMIOL aperture in PCI Memory Space
<b>MMIOH GZ-DS Base</b>	64	-	M	RO	Indicates the base address of the MMIOH aperture in Gen-Z Data Space
<b>MMIOH Size</b>	64	-	M	RO	Indicates the size of MMIOH in bytes
<b>MMIOH PCI-MS Base</b>	64	-	M	RW	Determines the base address of the MMIOH aperture in PCI Memory Space
<b>MMIO R-Key</b>	32	-	O	RW	If the LPD supports <i>Region Key (R-Key)</i> , then this field contains the R-Key to use for validating incoming request packets that target the MMIOL or MMIOH apertures.
<b>F-CTL</b>	16	-  Bit 0  Bits 4:1	M	RW	Used to control aspects of the corresponding function that are not covered by its PCI / PCIe configuration space.
					HwInit Write Enable—Determines if software is enabled to modify specific HwInit fields within the corresponding function's PCI / PCIe configuration space.  0b—Disabled 1b—Enabled
					Request Traffic Class—Traffic Class to use for VC selection for transmitted request packets. See <i>Component Destination Table Structure</i>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
LPD Function Number		Bit 5			Interrupt R-Key Enable—Determines if an interrupt request packet shall include the RW R-Key associated with this function.  0b—No R-Key Present 1b—R-Key Present
		Bit 6			R-Key Non-Interrupt Enable—Determines if non-interrupt request packets may include the RW R-Key / RO R-Key associated with this function.  0b—Disabled 1b—Enabled
		Bit 7			PCO Enabled—if the LPD supports PCO, then this is used to enable / disable PCO semantics for this function.  0b—Disabled 1b—Enabled
		Bits 15:8			RsvdP
		8	M	RO	This field contains the function number of the corresponding function.  If Device Number Support == 0b, then the maximum LPD Function Number shall be 255.  If Device Number Support == 1b, then the maximum LPD Function Number shall be 7.
LPD Dev Number	5	-	M	RW	If Device Number Support == 1b, then this field shall be configured with the LPD's Device Number.  If Device Number Support == 0b, then this field shall be Reserved.
Function PTR n	32	-	M	RO	Pointer to byte 0 of the corresponding function's PCI / PCIe configuration space.
Function RO R-Key n	32	-	O	RW	If the LPD supports <i>Region Key (R-Key)</i> , then this field contains the R-Key to use in read-based request packets.

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>Function RW R-Key n</b>	32	-	O	RW	If the LPD supports <i>Region Key (R-Key)</i> , then this field contains the R-Key to use in write-based and interrupt request packets.
<b>R0</b>	20	-	-	-	RsvdP
<b>R1</b>	32	-	-	-	Reserved
<b>R2</b>	64	-	-	-	Reserved
<b>R3</b>	3	-	-	-	RsvdP

## 18.8. LPD Component Requirements

The following are the PECAM Requester requirements:

- Shall support the PECAM logic as specified in this document.
- Shall support the Control OpClass and the following operations:
  - Read Request
  - Read Response
  - Write Request
  - *Standalone Acknowledgment*
  - *Unsolicited Event (UE) Packet*
  - *Interrupts as a Responder*

The following are the I/O component requirements to support LPD Functions:

- Shall support at least one *Component LPD Structure*
- Shall support the Control OpClass and the following operations:
  - Read Request
  - Read Response
  - Write Request
  - *Standalone Acknowledgment*
  - *Unsolicited Event (UE) Packet*
  - *Interrupts as a Responder*
- Should support the *Component Error and Signal Event Structure*

## 19. Logical PCI Hierarchy (LPH)

An LPH is a Logical PCI mechanism that serves as a Host Bridge for a native PCIe hierarchy, as defined in the *PCI Express Base Specification*. This enables a Gen-Z I/O Component to attach a set of native PCIe devices to a host via a Gen-Z fabric as illustrated in *Example Gen-Z I/O Component with One LPH*, which illustrates an example LPH bridging to a PCIe hierarchy with two PCIe Root Ports (RPs), one Root Complex Integrated Endpoint (RCiEP), an external PCIe switch, and three external PCIe devices. An LPH highly leverages LPD mechanisms and concepts, but it presents a native PCIe hierarchy, not LPD endpoint functions. “Native PCIe” in this context refers either to external PCIe components (switches or devices) or to hardware logic inside the Gen-Z I/O component (RPs or RCiEPs) that requires no hardware modifications to work in this Gen-Z-attached environment.

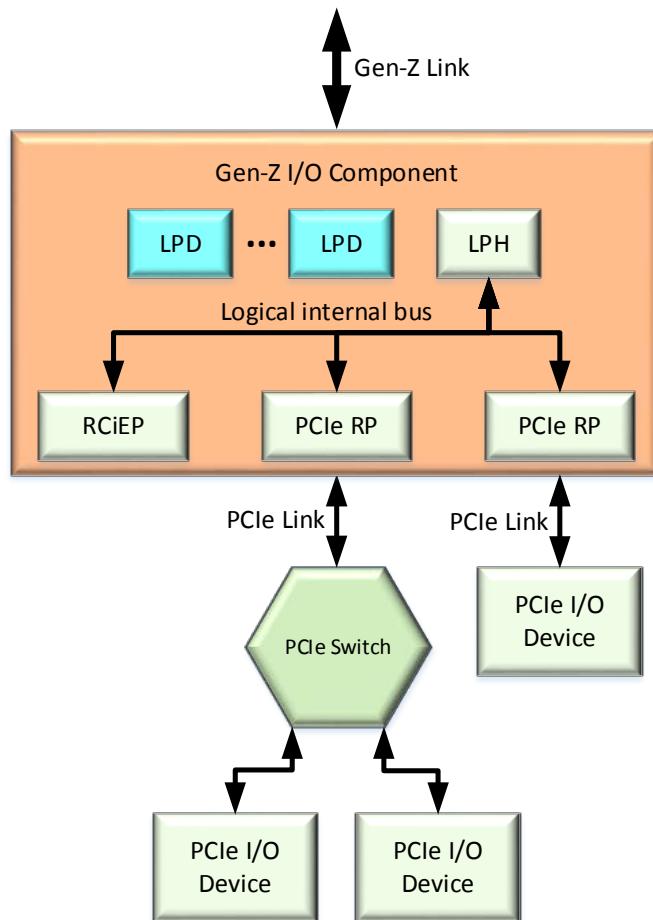


Figure 19-1: Example Gen-Z I/O Component with One LPH

Like an LPD:

- An LPH is implemented by a Gen-Z I/O component that is connected to the host via a Gen-Z fabric.
- No Gen-Z switches connecting the LPH to the host are visible to an unmodified OS.
- For its communications, an LPH uses Gen-Z packets that optionally include the *LPD Field*.

Unlike an LPD, an LPH:

- Exposes an entire PCIe hierarchy, consisting of one or more PCIe devices.
- Exposes a PCI bus number range mapped by a host PECAM
- Exposes any PCIe functions present with a *PCI Type 1 Configuration Space Header*
- Does not present any LPD functions. A PCIe hierarchy does not contain any Gen-Z components.
- May support peer-to-peer traffic across its logical internal bus and within the hierarchy

The logical internal bus below the LPH is the same concept as the logical internal bus inside a PCIe switch. All entities on the logical internal bus may send packets to each other, and all PCIe devices on the logical internal bus have the same bus number. PIO request packets flow across the logical internal bus from the LPH to RPs or RCiEPs. DMA request packets flow across the logical internal bus from RPs or RCiEPs to the LPH. Peer-to-peer request packets flow across the logical internal bus between pairs of RPs or RCiEPs. The logical internal bus supports this logical behavior, but is not required to be implemented as a hardware bus.

## 19.1. LPH Operational Overview

A Gen-Z component that supports LPHs implements a *Component LPH Structure* for each LPH. *Key LPH Control Space and Data Space Structures* illustrates some key LPH Control Space and Data Space structures.

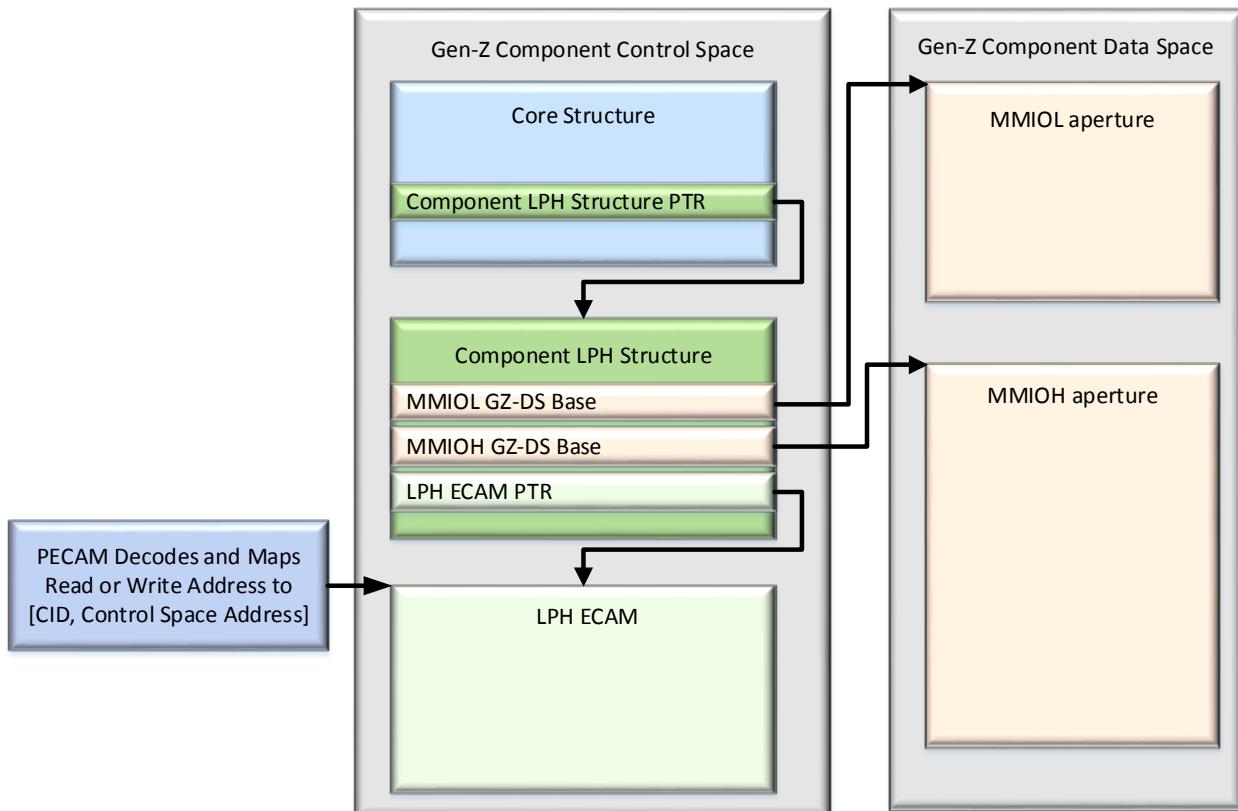


Figure 19-2: Key LPH Control Space and Data Space Structures

Many fields in the *Component LPH Structure* for LPHs serve the same general purpose as with LPDs. However, in contrast to an LPD implementing a 4 KiB PCI Configuration Space region for each LPD function, an LPH implements an ECAM as defined in the *PCI Express Base Specification*.

Gen-Z Control Read/Write accesses to the LPH ECAM range generate PCIe Configuration Read/Write requests, which are transmitted to the PCIe hierarchy. Gen-Z Read/Write accesses to the MMIO apertures generate PCIe Memory Read/Write requests, which are transmitted to the PCIe hierarchy. DMA read/write requests received from the PCIe hierarchy are translated to corresponding Gen-Z Read/Write requests and transmitted to the host. MSI or MSI-X messages received from the PCIe hierarchy are handled the same as DMA write requests, just as they are with PCIe. As with LPDs, PCI I/O Space is not supported by LPHs.

LPHs may implement and use *PCIe-Compatible Ordering (PCO)*. Alternatively, LPHs may use mechanisms outside of the Gen-Z specification scope to handle ordering with native PCIe devices below them.

An LPH may implement a Root Complex Register Block (RCRB), as defined in the *PCI Express Base Specification*. An LPH RCRB may be useful for managing functionality that's common across multiple RPs or RCiEPs, e.g., CRS Software Visibility. If implemented, an LPH RCRB exists in Gen-Z Control Space.

## 19.2. LPH Support for PCI Configuration Space

An LPH ECAM behaves similarly to a native PCIe ECAM in the host, mapping a PCI bus number range so that load/store accesses to that range result in the generation of PCI Configuration Read/Write requests. As illustrated in *Example Gen-Z I/O Component with One LPH*, the LPH ECAM range resides in the Gen-Z I/O component's Control Space. Gen-Z-aware software maps the LPH ECAM range using a Gen-Z PECAM in the host.

Here are the basic steps when a host processor does a successful PCI Configuration Space access:

1. Processor does a load/store access to the associated PECAM range
2. Requester ZMMU generates a Gen-Z Control Read/Write request packet targeting the LPH ECAM
3. Gen-Z fabric delivers the Gen-Z request packet to the LPH
4. LPH ECAM generates a corresponding PCIe Configuration Read/Write request
5. PCIe fabric delivers the PCIe request to the target PCI function
6. Target PCIe function responds with a PCIe completion
7. PCIe fabric delivers the PCIe completion to the LPH
8. LPH generates a corresponding Gen-Z response packet
9. Gen-Z fabric delivers the Gen-Z response packet to the host

An LPH ECAM can be viewed as a proxy for Gen-Z request packets into the PCIe domain, and PCIe completion packets into the Gen-Z domain. An LPH ECAM is Gen-Z and PCIe protocol-aware and maintains state to correlate PCIe completion packets with their associated Gen-Z request packets.

*BDF Number Mappings by an LPH ECAM* illustrates how BDF number mappings are handled by an LPH ECAM, and how the remaining fields are translated into a PCI Configuration Space offset within the targeted PCIe function.

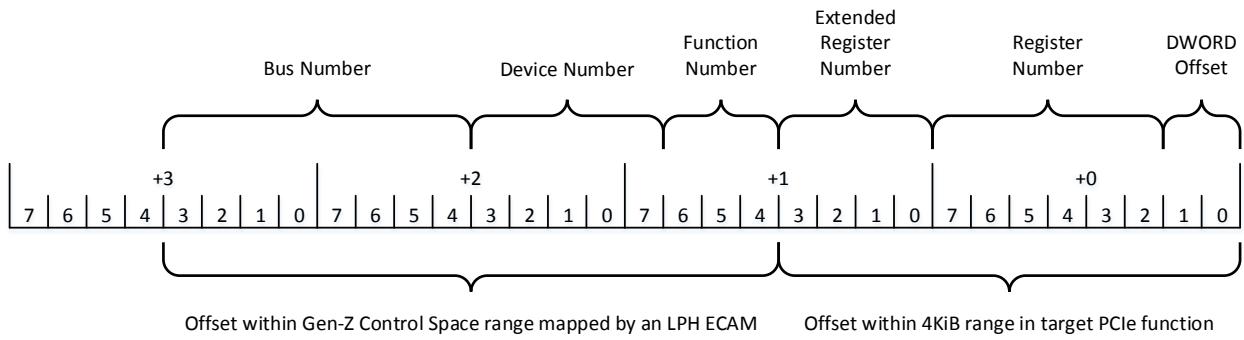


Figure 19-3: BDF Number Mappings by an LPH ECAM

The following are features and requirements associated with an LPH ECAM. All referenced fields are in the *Component LPH Structure*.

- 5 LPH ECAM bus number mappings shall be as illustrated in *BDF Number Mappings by an LPH ECAM*.
- 10 An LPH ECAM shall be capable of mapping an integer number of PCI bus numbers, ranging from 1 to 256, and this shall be indicated by the LPH ECAM Size field. The size of the implemented LPH ECAM range in Control Space shall be (LPH ECAM Size) \* 1 MiB.
- 15 To enable efficient mapping of multiple LPH ECAM ranges by a common PECAM in the host, each LPH ECAM shall support a configurable LPH ECAM BN Offset field. When calculating the bus number used in generated PCIe Configuration Request packets, the LPH ECAM BN Offset field value shall be added to the bus number derived from the Control Space address as illustrated in *BDF Number Mappings by an LPH ECAM*.
- 20 Software that configures a PECAM in the host to map an LPH ECAM range shall not map more bus numbers than the LPH ECAM implements, as indicated by its LPH ECAM Size field. Software may map fewer bus numbers than the full range implemented by the LPH ECAM. If software configures the LPH ECAM BN Offset field to a non-zero value, software shall not map more than  $256 - (\text{LPH ECAM BN Offset})$  bus numbers.
- 25 The LPH ECAM PTR field shall point to the base of the LPH ECAM in Control Space, which shall be aligned to a 1 MiB boundary.
- 30 To enable efficient use of Requester ZMMU mapping resources, if the LPH ECAM range size is 32 MiB or greater, LPH ECAM Base should be aligned to a 32-MiB boundary, matching the largest mandatory page size for Requester ZMMUs.

### 19.3. LPH Support for PCI MMIO

An LPH supports Memory-Mapped I/O (MMIO) apertures conceptually similar to those supported by LPDs, as specified in *LPD MMIO Space*, though an LPH itself has no Base Address registers (BARs) that are mapped into these apertures. Instead, load/store accesses to LPH MMIO apertures by Gen-Z Read/Write requests result in the generation of PCIe Memory Read/Write requests targeting PCIe functions within the PCIe hierarchy. Memory BARs within those PCIe functions map PCI Memory Space within the LPH's MMIO apertures.

PCI I/O Space is not supported by LPHs.

*Key LPH Control Space and Data Space Structures* illustrates the LPH MMIO and MMIOH apertures reside in the Gen-Z I/O component's Data Space. Gen-Z-aware software maps the LPH MMIO aperture ranges using the Gen-Z Requester ZMMU in the host.

MMIO writes on PCIe are posted, so the basic steps for MMIO reads and MMIO writes have some significant differences.

Here are the basic steps when a host processor does a successful PCI MMIO read:

- 5 1. Processor does a load access to the associated MMIO range
2. Requester ZMMU generates a Gen-Z Read request targeting an LPH MMIO aperture
3. Gen-Z fabric delivers the Gen-Z request packet to the LPH
4. LPH MMIO aperture generates a corresponding PCIe Memory Read request
5. PCIe fabric delivers the PCIe request to the target PCI function
6. Target PCIe function responds with a PCIe completion
- 10 7. PCIe fabric delivers the PCIe completion to the LPH
8. LPH generates a corresponding Gen-Z response
9. Gen-Z fabric delivers the Gen-Z response packet to the host

Here are the basic steps when a host processor does a successful PCI MMIO write:

- 15 1. Processor does a store access to the associated MMIO range
2. Requester ZMMU generates a Gen-Z Write request targeting an LPH MMIO aperture
3. Gen-Z fabric delivers the Gen-Z request packet to the LPH
4. LPH MMIO aperture generates a corresponding PCIe Memory Write request, which is posted
5. PCIe fabric delivers the PCIe request to the target PCI function
6. In parallel with previous step: if needed, LPH generates a Gen-Z response
- 20 7. Gen-Z fabric delivers the Gen-Z response (if generated) to the host

Similar to an LPH ECAM, an LPH MMIO aperture can be viewed as a proxy for Gen-Z requests into the PCIe domain, and PCIe completions into the Gen-Z domain. An LPH MMIO aperture is Gen-Z and PCIe protocol aware and maintains state to correlate PCIe completions with their associated Gen-Z requests.

The following are features and requirements associated with LPH MMIO apertures. All referenced fields are in the *Component LPH Structure*.

- 25 • For each of the MMIO and MMIOH apertures, a GZ-DS Base field indicates the base address of the aperture in Gen-Z Data Space, a Size field indicates the size in bytes, and a PCI-MS Base field is configured to determine the starting address in PCI Memory Space.
- 30 • If the LPH does not implement either the MMIO or MMIOH aperture, then the associated aperture's size and GZ-DS Base fields shall be 0.
- Consistent with PCI Bridge requirements, each aperture's GZ-DS Base shall be aligned to a 1-MiB boundary.
- 35 • Consistent with PCI Bridge requirements, each aperture's size shall be an integer multiple of 1 MiB.
- To enable compatibility with software BAR configuration algorithms that configure BARs with the largest size first, the GZ-DS Base for each aperture should be aligned to a boundary equal to the largest BAR size envisioned for PCIe adapters that the LPH will support.
- 40 • To enable efficient use of Requester ZMMU mapping resources, if an aperture's size is 32 MiB or greater, then the GZ-DS Base for it should be aligned to a 32-MiB boundary, matching the largest mandatory page size for Requester ZMMUs.

## 19.4. LPH PIO Common Logic

Host processor access to PCI Configuration and MMIO Space is known as Programmed I/O (PIO). The following are features and requirements associated with LPH PIO common logic. All referenced fields are in the *Component LPH Structure*.

- 5     • With PIO accesses, the LPH operates as a Gen-Z Responder, and shall handle all general requirements for Gen-Z responders.
- 10    • With PIO accesses, the LPH operates as a PCIe requester, and shall handle all general requirements for PCIe requesters as defined in the *PCI Express Base Specification*, including error checking and reporting.
- 15    • The LPH shall implement a set of PIO Request Tracking Resources (RTRs) to support the correlation of PCIe completion packets with Gen-Z request packets, and shall report the number of implemented PIO RTRs in the LPH PIO RTRs field.
- 20    • Each PIO RTR contains a subset of the Gen-Z PIO request packet's protocol fields (e.g., the SCID, 12-bit Gen-Z Tag, Access Key) and a subset of the PCIe PIO request packet's protocol fields (e.g., the 8-bit PCIe Tag).
- 25    • When the LPH receives a Gen-Z PIO request packet and is preparing to generate a corresponding PCIe PIO request packet, it allocates or checks the necessary resources, such as a free PIO RTR, unassigned 8-bit PCIe Tag value, and sufficient PCIe flow-control credits for transmission.
  - o     Note: PCIe Memory Write requests, used for MMIO writes, are posted. Each will receive no PCIe completion, and each requires no PCIe Tag.
  - o     If the necessary resources are available, the LPH shall transmit the PCIe request packet.
  - o     Else if the LPH is using *PCIe-Compatible Ordering (PCO)*, the LPH shall handle this situation as specified in *Deadlock Avoidance for PCO*.
  - o     Else if the LPH is using an implementation-specific mechanism for handling PCIe ordering, it shall use an implementation-specific mechanism for handling this situation.
  - o     Else the LPH shall return a *Responder Not Ready (RNR) NAK* in response to the Gen-Z Control request and release any associated reserved resources.
- 30    • The LPH should implement a *Forward Progress Screen (FPS)* to guarantee forward progress if its resources associated with sending PCIe requests become oversubscribed.
- 35    • If the PCIe PIO request was a Memory Write, there will be no associated PCIe completion.
- 40    • If the PCIe PIO request was a Memory Read, a Configuration Read, or a Configuration Write, when the LPH receives a PCIe completion packet, it correlates it to the associated Gen-Z PIO request packet, first checking for its own PCIe Requester ID and then using the 8-bit PCIe Tag to match a PIO RTR.
- 45    • If the Gen-Z request was for an MMIO write and PCO is enabled, there will be no associated Gen-Z response packet. Otherwise, the LPH generates a corresponding Gen-Z response packet using data from the PCIe completion and the matching PIO RTR. The LPH shall handle PCIe Completion Status values as follows, with most mapping to Gen-Z Reason field encodings:
  - o     PCIe Successful Completion (SC) maps to No Error
  - o     PCIe Unsupported Request (UR) maps to Unsupported Request (UR) Error
  - o     PCIe Configuration Retry Status (CRS) is handled as specified in the *PCIe Base Specification*
  - o     PCIe Completer Abort (CA) maps to Packet Execution Error (EXE) Fatal
  - o     All other PCIe values map to Unsupported Request (UR) Error
- 45    • If a Gen-Z response packet is required and sufficient Gen-Z flow-control credits are available for transmitting the Gen-Z response packet, the LPH transmits the packet. Otherwise, the LPH waits

until sufficient Gen-Z flow-control credits become available, and relies on the PCIe flow-control credit mechanism to back-pressure incoming PCIe packets as necessary.

## 19.5. LPH Support for DMA

If an LPH observes a PCIe Memory Read/Write request on its logical internal bus, and the PCI Memory address falls outside of its MMIO apertures, the LPH will translate and forward the PCIe Memory request to the host for Direct Memory Access (DMA) to host memory. If the PCI Memory address falls within an MMIO aperture, an RP or RCiEP on the logical internal bus may claim the PCIe Memory request, handling it as a peer-to-peer access. See *LPH Support for Peer-to-Peer Traffic*.

PCIe software device drivers specify DMA addresses, which are in PCI Memory Space. No address translation is performed for DMA when an LPH translates PCIe Memory request packets into Gen-Z Data Space request packets. However, an IOMMU in the host may perform address translation on received DMA requests.

DMA writes on PCIe are posted, so the basic steps for DMA reads and DMA writes have some significant differences.

Here are the basic steps when a PCIe device does a successful DMA read:

1. PCIe device generates a PCIe Memory Read request targeting a DMA range
2. PCIe fabric delivers the PCIe request to the LPH
3. LPH DMA logic generates a corresponding Gen-Z Read request packet
4. Gen-Z fabric delivers the Gen-Z request packet to the host
5. Host responds with a Gen-Z response packet
6. Gen-Z fabric delivers the Gen-Z response packet to the LPH
7. LPH generates a corresponding PCIe completion
8. PCIe fabric delivers the PCIe completion to the PCIe device

Here are the basic steps when a host processor does a successful DMA write:

1. PCIe device generates a PCIe Memory Write request packet targeting a DMA range
2. PCIe fabric delivers the PCIe request packet to the LPH
3. LPH DMA logic generates a corresponding Gen-Z Write request packet
4. Gen-Z fabric delivers the Gen-Z request packet to the host
5. If needed, host generates a Gen-Z response packet
6. Gen-Z fabric delivers the Gen-Z response packet (if generated) to the LPH

The following are features and requirements associated with LPH DMA. All referenced fields are in the *Component LPH Structure*.

- With DMA accesses, the LPH operates as a Gen-Z Requester, and shall handle all general requirements for Gen-Z Requesters.
- With DMA accesses, the LPH also operates as a PCIe completer, and shall handle all general requirements for PCIe completers as defined in the *PCI Express Base Specification*, including error checking and reporting.
- If LPH DMA logic observes a PCIe Memory request on its logical internal bus, and the DMA address falls outside of its MMIO apertures, the LPH DMA logic will translate the PCIe Memory request to a Gen-Z Data Space request packet.

- The LPH shall implement DMA Request Tracking Resources (RTRs) to support the correlation of Gen-Z response packets with PCIe request packets, and shall report the number of implemented DMA RTRs in the LPH DMA RTRs field.
- Each DMA RTR contains a subset of the PCIe DMA request packet's protocol fields (e.g., the PCIe Requester ID, 8-bit PCIe Tag) and a subset of the Gen-Z request packet's protocol fields (e.g., the DCID, 12-bit Gen-Z Tag, RD Size).
- When the LPH is generating a Gen-Z request packet, it allocates or checks the necessary resources, such as a free DMA RTR, unassigned 12-bit Gen-Z Tag value, and sufficient Gen-Z flow-control credits for transmission.
  - Note: If PCO is enabled, Gen-Z write request packets are posted. Each has no Gen-Z Tag and requires no Gen-Z response packet.
  - If the necessary resources are available, the LPH shall transmit the Gen-Z request packet.
  - Otherwise, the LPH waits until sufficient resources become available, and relies on the PCIe flow-control credit mechanism to back-pressure incoming PCIe packets as necessary.
- If the Gen-Z request packet was a write and PCO is enabled, there will be no associated Gen-Z response packet. Otherwise, when the LPH receives a Gen-Z response packet, it correlates it to the associated Gen-Z request packet using its 12-bit Gen-Z Tag and SCID to match a DMA RTR. The LPH shall handle conditions like receiving a *Responder Not Ready (RNR) NAK* or a response packet not matching a DMA RTR per general Gen-Z requester requirements.
  - Note: if PCO is enabled, Gen-Z read response packets will be returned in order and be delivered exactly once. Otherwise, LPH logic will have to handle these conditions.
- If a PCIe completion is required and sufficient PCIe flow-control credits are available for transmitting the PCIe Completion packet, the LPH transmits the packet. Otherwise, the LPH waits until sufficient PCIe flow-control credits become available, and relies on the Gen-Z flow-control credit mechanism to back-pressure incoming Gen-Z packets as necessary.

## 19.6. LPH Support for Peer-to-Peer Traffic

An LPH may support peer-to-peer traffic between pairs of RPs or RCIEPs on its logical internal bus. For this case, no PCIe packet proxy or translation is required. As is with PCIe, an LPH may have implementation-specific limitations with peer-to-peer traffic.

Peer-to-peer traffic between different LPHs, or between LPDs and LPHs is beyond the scope of this specification.

## 19.7. LPH Error Handling

The LPH mechanism is highly leveraged from the LPD mechanism, and has the same error handling requirements as specified in *LPD Error Handling*.

## 19.8. Component LPH Structure

To enable an operating system to configure LPHs using the existing PCI software infrastructure, Gen-Z I/O components need to provide conventional PCI Configuration and Memory Spaces. These spaces are reported and managed via the Component LPH structure, and accessed by an operating system using the PECAM for Configuration Space and the Requester ZMMU for Memory Space. The Component LPH

structure is similar to the *Component LPD Structure* used for the *Logical PCI Device (LPD)* mechanism. Many of their fields are similar or identical.

The following are the features and requirements of the Component LPH Structure:

- A component may support one or more Component LPH structures.
  - Each Component LPH structure/hierarchy shall be associated with a single PECAM Requester, considered to be the LPH's host.
    - A Component LPH structure shall be associated with a single contiguous range of bus numbers within a single PCI Segment Group.
  - If multiple Component LPH structures are supported, then:
    - Each additional structure shall be accessed through the Next Component LPH Structure PTR.
    - Multiple LPH structures may be associated with a single PECAM Requester, with the limit being determined by how many bus numbers each LPH structure is assigned and how many PCI Segment Groups the host supports.
    - Depending upon the physical Gen-Z topology, each Component LPH structure may be associated with a different PECAM Requester. This enables a Gen-Z I/O component to be simultaneously shared by multiple PECAM Requesters, as described and illustrated under the *Component LPD Structure* section. Component LPD structures and Component LPH structures may be assigned to PECAM Requesters independently of each other. Configuration and management of these components are outside of this specification's scope.
- The Component LPH structure shall use the *Component LPH Structure Format*.
- Each Component LPH structure shall support at least one bus number, and may support up to 256 bus numbers.
- Management may configure the Default HCID and the Default HSID with the respective CID and SID used to identify the host component to target for LPH read, write, interrupt, etc. operations. If configured, these default values apply to the entire LPH hierarchy.
- An LPH may implement a Root Complex Register Block (RCRB), as defined in the *PCI Express Base Specification*. If implemented, the LPH RCRB shall exist in Gen-Z Control Space, it shall be 4 KiB in size, it shall be aligned to a 4 KiB boundary, the LPH RCRB PTR field shall point to it, and Gen-Z-aware software may map it using the Requester ZMMU.

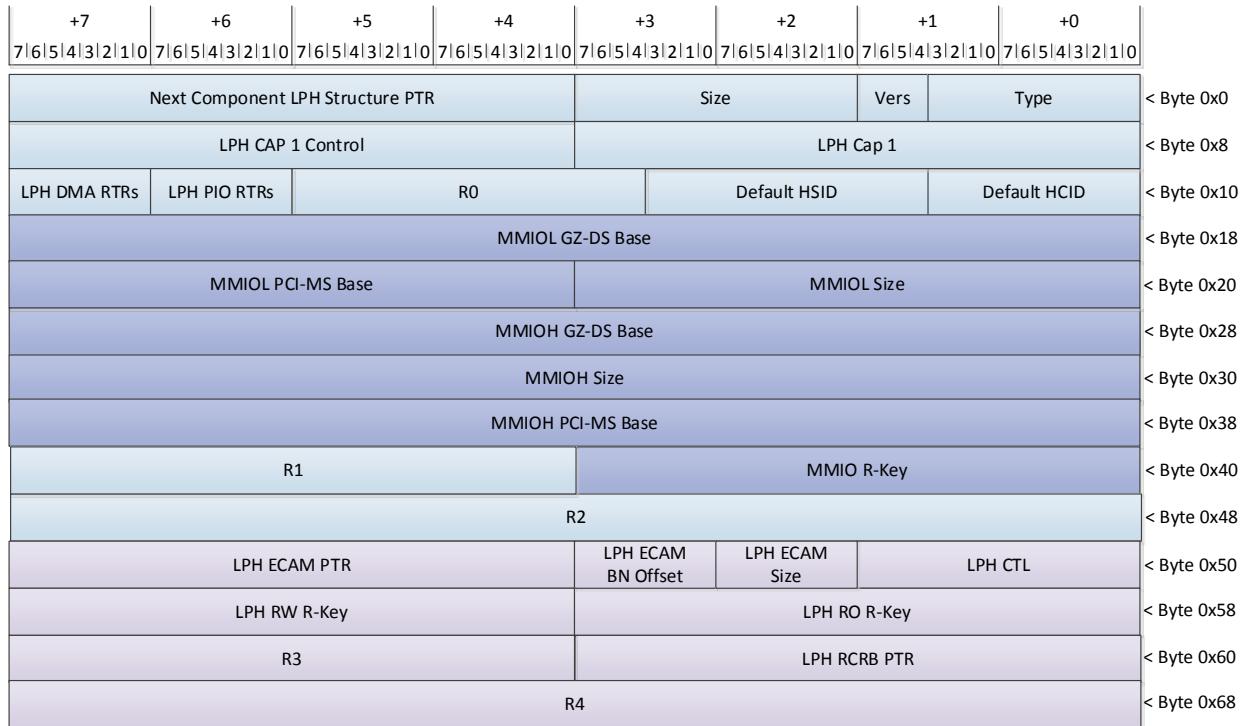


Figure 19-4: Component LPH Structure Format

Table 19-1: Component LPH Structure Fields

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
Version (Vers)	4	0x1	M	RO	Structure Version
Next Component LPH Structure PTR	32	-	O	RO	Indicates if a Component LPH structure is linked to this structure. If Null, then a Component LPH structure was not provisioned.
LPH CAP 1	32	-	M	RO	<p>LPH Capabilities 1</p> <p>Multi-subnet Support—Indicates if the component supports multi-subnet LPH operations. If supported, then the Default HSID field may be configured.</p> <p>0b—Unsupported 1b—Supported</p> <p>Non-Default Host ID Support—Indicates if the component supports LPH communication with non-default host components. If supported, then non-default host components are identified and accessed through means outside of this specification's scope.</p>

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description		
LPH CAP 1 Control	32	Bit 2	M	RW	0b—Unsupported 1b—Supported		
					PCO Support—Indicates if the LPH supports PCIe-Compatible Ordering semantics. PCO support applies to an entire LPH hierarchy.		
					0b—Unsupported 1b—Supported		
		Bits 31:3			RsvdP		
		Bit 0			Non-Default Target Enable—if enabled, then LPH operations associated with this structure may target a component identified and selected through means outside of this specification’s scope.		
					If disabled, then the LPH shall use only the Default HCID and optionally the Default HSID for targeting its host.		
					0b—Disabled 1b—Enabled		
					LPH Reset—When a value of 1b is written to this bit, then the contents of this Component LPH structure and all related PCI / PCIe configuration structures throughout the LPH hierarchy shall be returned to their power-on initialized state.		
		Bit 1			1b—Reset  Reads of this bit shall return 0b.		
					Valid Default HSID field—indicates if the Default HSID field has been configured with a valid value.		
					0b—Not Configured 1b—Configured		
		Bit 2			LPH Communications Enable—Determines if LPH operations associated with this structure are permitted.		
					0b—Disabled 1b—Enabled		
		Bit 3					

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
<b>Default HCID</b>		Bits 31:4			RsvdP
	12	-	M	RW	<p>Identifies the CID of the default host component to target LPH interrupt, read, write, and etc. operations.</p> <p>If enabled via the Non-Default Target Enable bit, an LPH may target components identified through means outside of this specification's scope.</p>
<b>Default HSID</b>	16	-	O	RW	<p>Identifies the SID of the default host component to target LPH interrupt, read, write, and etc. operations.</p> <p>If enabled via the Non-Default Target Enable bit, an LPH may target non-default components identified through means outside of this specification's scope.</p> <p>If the component does not support multi-subnet LPH operations, then this field shall be Reserved.</p>
<b>LPH PIO RTRs</b>	8	-	M	RO	Indicates the number of PIO RTRs implemented, with a value of 0 indicating 256.
<b>LPH DMA RTRs</b>	8	-	M	RO	Indicates the number of DMA RTRs implemented, with a value of 0 indicating 256.
<b>MMIOL GZ-DS Base</b>	64	-	M	RO	Indicates the base address of the MMIOL aperture in Gen-Z Data Space
<b>MMIOL Size</b>	32	-	M	RO	Indicates the size of MMIOL in bytes
<b>MMIOL PCI-MS Base</b>	32	-	M	RW	Determines the base address of the MMIOL aperture in PCI Memory Space
<b>MMIOH GZ-DS Base</b>	64	-	M	RO	Indicates the base address of the MMIOH aperture in Gen-Z Data Space
<b>MMIOH Size</b>	64	-	M	RO	Indicates the size of MMIOH in bytes
<b>MMIOH PCI-MS Base</b>	64	-	M	RW	Determines the base address of the MMIOH aperture in PCI Memory Space

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
MMIO R-Key	32	-	O	RW	If the LPD supports <i>Region Key (R-Key)</i> , then this field contains the R-Key to use for validating incoming request packets that target the MMIO1 or MMIOH apertures.
LPH CTL	16	-	M	RW	Used to control aspects of the LPH hierarchy that are not covered by the PCI / PCIe Configuration Space.
		Bits 3:0			Request Traffic Class—Traffic Class to use for VC selection for transmitted request packets. See <i>Component Destination Table Structure</i> .
		Bit 4			Interrupt R-Key Enable—Determines if an interrupt request packet shall include the RW R-Key associated with this LPH hierarchy. 0b—No R-Key Present 1b—R-Key Present
		Bit 5			R-Key Non-Interrupt Enable—Determines if non-interrupt request packets may include the RW R-Key / RO R-Key associated with this LPH hierarchy. 0b—Disabled 1b—Enabled
		Bit 6			PCO Enabled—If the LPH supports PCO, then this is used to enable / disable PCO semantics for this LPH hierarchy. 0b—Disabled 1b—Enabled
		Bits 15:7			RsvdP
LPH ECAM Size	8	-	M	RO	Indicates the size of the LPH ECAM in terms of bus numbers, with a value of 0 indicating 256.
LPH ECAM BN Offset	8	-	M	RW	For generated PCIe Configuration requests, determines the bus number offset used

Field Name	Size (bits)	Value / Bit Location	M / O	Access	Description
LPH ECAM PTR					relative to the bus number derived from the Gen-Z Control Space address.
LPH RO R-Key	32	-	M	RO	Points to the base of the LPH ECAM.
LPH RW R-Key	32	-	O	RW	If the LPH supports <i>Region Key (R-Key)</i> , then this field contains the R-Key to use in read-based request packets.
LPH RCRB PTR	32	-	O	RO	If not Null, then this points to the base of the LPH RCRB. If the LPH RCRB is not implemented, the LPH RCRB PTR field shall be Null.
R0	20	-	-	-	RsvdP
R1	32	-	-	-	Reserved
R2	64	-	-	-	Reserved
R3	32	-	-	-	Reserved
R4	64	-	-	-	Reserved

## 19.9. LPH Component Requirements

The LPH mechanism is highly leveraged from the LPD mechanism, and has the same requirements as specified in *LPD Component Requirements*, with the following exception:

- Instead of supporting at least one *Component LPD Structure*, an LPH component shall support at least one *Component LPH Structure*

# 20. Address Translation and Page Services

This section specifies the operations used to support address translation and page services. Address translation services enable a Requester to obtain translated addresses that used to optimize subsequent request packet access to a Responder's resources. Page services enable a Requester to request one or more Responder pages to be made available / resident for subsequent request packet access. This eliminates the need to pin or register memory until the application needs access.

Within this section, the statements that apply to a *Logical PCI Device (LPD)* also apply to a native PCIe endpoint that exists within a *Logical PCI Hierarchy (LPH)*.

## 20.1. Address Translation Services Overview

Address translation services involves two components: a component (e.g., a SoC) that contains memory that is accessed by a second component (e.g., I/O). *Translation Request-Translation Response Packet Exchange* illustrates two components. The blue component contains some amount of memory and a Translation Agent (TA) that manages a MMU (memory management unit) / ZMMU. The orange component contains one or more contexts and each context may have a context cache that is contains translated addresses. A translated address is obtained by transmitting a *Translation Request* packet that contains the untranslated address to the TA, to which the TA responds with a *Translation Response* packet that contains the translated address. A *Translation Request* packet may also communicate if the untranslated address is associated with a *PASID* and whether execute or privileged attributes are required.

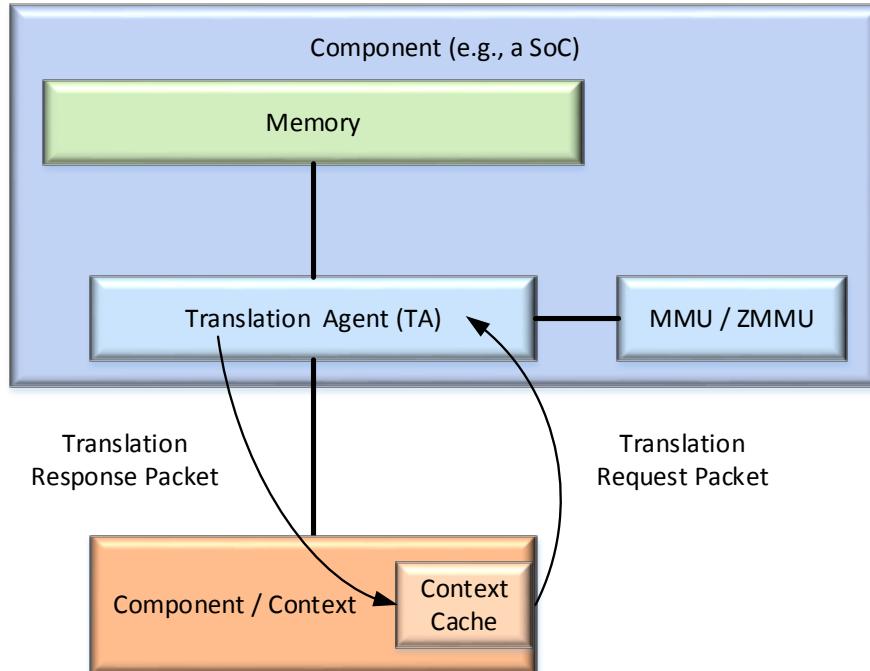


Figure 20-1: Translation Request-Translation Response Packet Exchange

If a TA needs to invalidate a translated address or update the attributes (e.g., execute or privilege), then the TA initiates a *Translation Invalidate Request-Translation Invalidate Response* packet exchange as illustrated in *Translation Invalidate Request-Translation Invalidate Response Packet Exchange*. To

ensure all translations are invalidated, the *Translation Invalidate Request* packet uses the untranslated address. The invalidation may be selective, i.e., on a per PASID basis, or global.

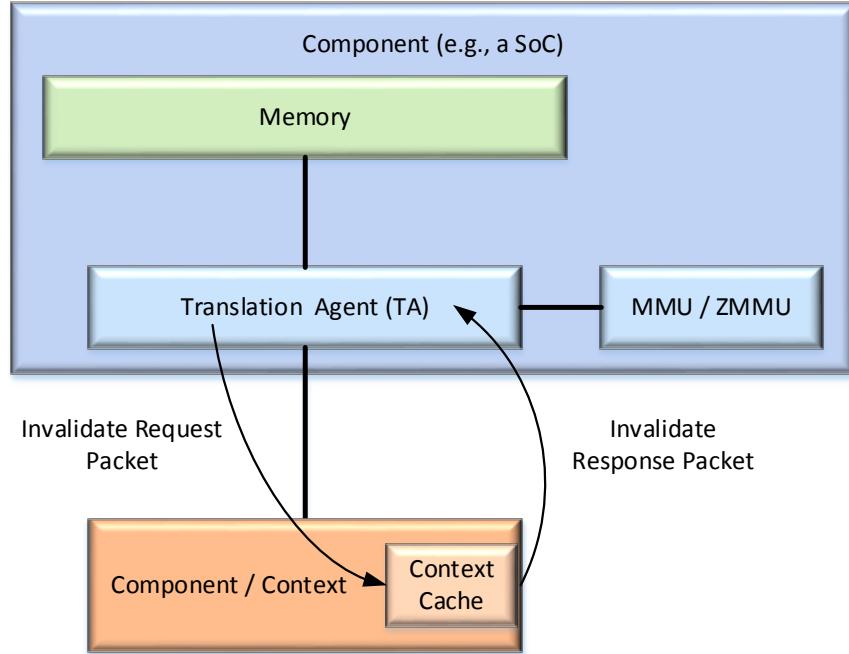


Figure 20-2: Translation Invalidate Request-Translation Invalidate Response Packet Exchange

5 Additional Address Translation Services requirements:

- A context cache shall be updated only through *Translation Request* and *Translation Invalidate Request* packets.
- Each context cache shall be independently managed even if the underlying implementation resources and logic are shared by multiple contexts.
- A component / context shall not set the TA bit to 1b in any request packet unless the address was obtained through a *Translation Request-Translation Response* packet exchange.

10

## 20.2. Page Services Overview

Page services is used to support on-demand page residency management services and to avoid requiring memory to be pinned prior to access. Page services involves two components: a component (e.g., a SOC) that contains memory that is accessed by a component (e.g., I/O). *PRG Request-PRG Response Packet Exchange* illustrates two components. The blue component contains some amount of memory and a Page Management Agent that manages page residency and access control. The orange component contains one or more contexts and each context may have a Page Request Interface (PRI) that issues and processes Page Request Group (PRG) request and response packets. The PRI issues one or more *PRG Request* packets to request one or more pages to be made accessible. The blue component transmits a single *PRG Response Notification* packet to indicate success or failure of the PRG's request packets. *PRG Request* packets may include a PASID to provide more fine-grain page management control (e.g., request a page within a corresponding guest virtual machine).

15

20

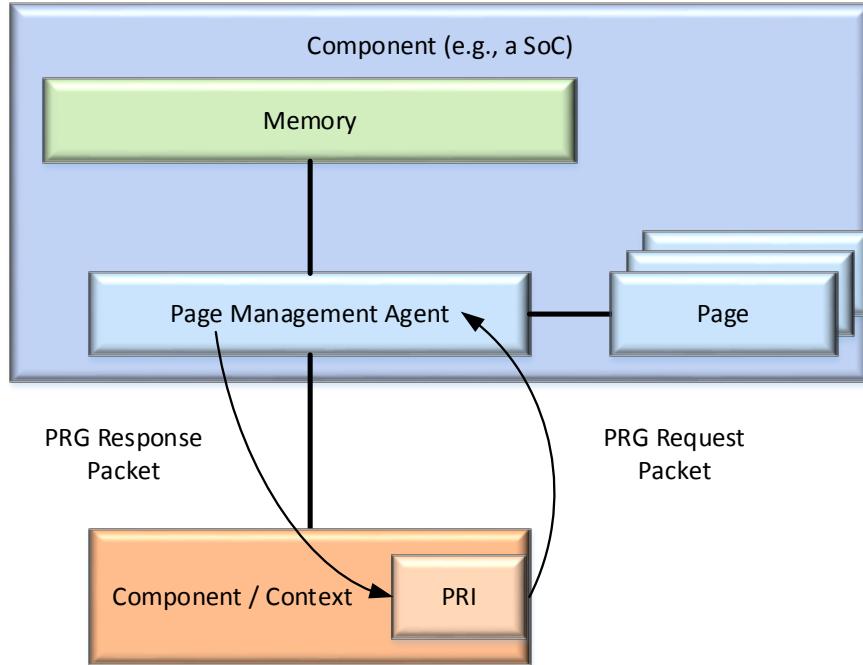


Figure 20-3: PRG Request-PRG Response Packet Exchange

A context may transmit a *PRG Release* packet to indicate when it no longer requires access to the page. Similarly, a Page Management Agent may transmit a *PRG Eviction Notification* packet to indicate when a component should stop using a page. Upon receipt, a component completes all request packet processing and notifies the Requester.

## 20.3. PASID

PASID is a 20-bit value that uniquely identifies a process address space on an application component, e.g., an application process or a guest virtual machine. Using a PASID enables a single component or component context to be simultaneously shared by multiple processes by logically extending each untranslated address to be [untranslated address, PASID]. PASID usage can improve solution performance such as in a virtualization environment by reducing the need for a guest virtual machine to trap to a hypervisor to update address mappings. For example, an untranslated address without a PASID represents a guest physical address (GPA). A hypervisor manages GPA-to-system physical address mappings. In contrast, an untranslated address with a PASID represents a guest virtual address (GVA). A guest virtual machine running on top of a hypervisor directly manages the GVA-to-GPA mappings, i.e., no hypervisor traps.

### 20.3.1. PASID Stop Sequence

In order to stop using a PASID, operating system / application middleware software informs each context using the PASID to stop generating new request packets and complete all request packet processing that use this PASID.

Software takes the following steps to stop using a PASID:

1. Software configures the PASID on the component that contains the memory to be stale.

2. Using a context-specific mechanism, software informs a context to stop using a PASID. The context-specific mechanism responds and indicates whether a *Stop Marker* request packet will be transmitted once the context has stopped issuing *Translation Request* and *PRG Request* packets with the stale PASID.
- 5 3. Upon receipt of a new *PRG Request* packet containing the stale PASID, transmit a *PRG Response Notification* packet with RSP Code = Invalid Request (Invalid PASID).
4. Upon receipt of a new *Translation Request* packet containing the stale PASID, treat the request packet as though it is accessing an unmapped address region.
- 10 5. Subsequent to the receipt of a *Stop Marker* request packet, the PASID in all new *PRG Request* packets is treated as a fresh / new PASID.

Upon being requested to stop using a PASID, a context takes the following steps:

1. Stop queuing new *Translation Request* and *PRG Request* packets with the stale PASID.
2. Complete transmitting any in-progress multi-page *PRG Request* packets with the stale PASID.
- 15 3. Wait for all outstanding non-Address Translation and Page Services request packets using the stale PASID to complete.
4. Wait for all outstanding *Translation Request* packets using the stale PASID to complete.
5. If the context-specific mechanism indicated a *Stop Marker* request packet will be transmitted, then:
  - a. Mark all outstanding and queued *PRG Request* packets with the PASID to be stopped as stale. Upon receipt of a corresponding *PRG Response Notification* packet, update page request allocation and PRG Index tracking logic. Further, the state of any of the requested pages shall be treated as non-deterministic.
  - b. If a LPD that is enabled for PCO, then once all stale *PRG Request* packets have been transmitted, transmit a *Stop Marker* request packet. If not a LPD or a LPD that is enabled for PCO, then wait until all stale *PRG Request* packets have been transmitted and acknowledged and then transmit a *Stop Marker* request packet.
- 20 6. If the context-specific mechanism indicated a *Stop Marker* request packet will not be transmitted, then it waits for all *PRG Request* packets with the stale PASID to be transmitted and for the corresponding *PRG Response Notification* packet to be received. Once all are received, then indicate using a context-specific mechanism that the stale PASID is no longer used.

## 20.4. Common ATP Packet Protocol Fields

*Common Address Translation and Page Protocol Fields* specifies protocol fields used in multiple address translation and page request and response packets.

If RK == 1b in an applicable CTXID ATP request packet, then the RW R-Key shall be copied to the R-Key field.

Table 20-1: Common Address Translation and Page Protocol Fields

Field Name	Size (bits)	Description
PV	1	Indicates if the PASID field contains a valid value 0b—Invalid 1b—Valid

Field Name	Size (bits)	Description
<b>PASID</b>	20	<p>Process Space Identifier—the PASID is combined with a REQCTXID and [SCID, GCID] to uniquely identify the address space associated with the operation (e.g., an application process or thread identifier). Each [REQCTXID, SCID   GCID] has an independent set of PASID values.</p> <p>If PASID Enable == 0b (see <i>Component ATP Structure</i>) or PV == 0b, then the PASID field shall be Reserved.</p> <p>Each PASID shall be independently managed and the actions taken on one PASID shall not impact any other PASID or context, e.g., starting or stopping the use of a given PASID. Specific PASID value management is outside of this specification's scope.</p>
<b>PV</b>	1	<p>Indicates if the PASID field contains a PASID or is Reserved.</p> <p>0b—PASID field is Reserved 1b—PASID field contains a PASID</p>
<b>BDF</b>	1	Indicates if the REQCTXID should be interpreted as a PCI BDF (only the lower 16 bits shall contain the BDF value).
<b>REQCTXID</b>	24	<p>Identifies the Requester context used to generate the request packet at the Requester.</p> <p>Max Context ID (see <i>Component ATP Structure</i>) is used to configure the maximum REQCTXID value.</p> <p>If a component uses only 16-bit identifiers (e.g., those designed to support PCI BDFs), then the identifier shall be placed in REQCTXID[15:0] and REQCTXID[23:16] shall be Reserved.</p>

## 20.5. Address Translation Services

A Translation Agent (TA) is a logical entity that transforms an address to optimize subsequent request packet processing. Though TA implementation is outside of this specification's scope, this document specifies the request and response packets used to interact with any TA.

### 5 20.5.1. Translation Request

A Translation Request packet is used carry address translation requests to a Responder that contains a Translation Agent (TA).

The following are the features and requirements of the Translation Request packet:

- A P2P-Coherency Translation Request packet shall use the *P2P-Coherency Translation Request Packet Format*.
- A CTXID Translation Request packet shall use the *CTXID Translation Request Packet Format*.

- A Translation Request packet has a non-deterministic execution time. To ensure Reliable Delivery, a Translation Request-*Translation Response* packet exchange shall comply with *Non-Deterministic Request Execution*.
- A Translation Request packet may request multiple address translations. The number of requested address translations is indicated by the NT field. Each translated address corresponds to a sequentially-increasing, fix-sized, STU-aligned resource unit (see *Component ATP Structure*) starting at the location indicated by the packet's Address.
- If a CTXID Translation Request packet contains an R-Key field, then the Responder shall validate that requested address range falls within a page(s) whose R-Key matches the packet's R-Key. If the R-Key does not match, then the Responder shall return a Core 64 *Standalone Acknowledgment* (Reason = Access Error (AE)—Invalid Access Permission). If the Requester is a LPD, then this error shall be handled as a Completer Abort (see *PCI Express Base Specification*).
  - If a LPD and the solution requires unmodified software compatibility, then Translation Request packets should not include an R-Key field.
  - A TA shall not translate an additional address (and all subsequent addresses) whose translated address does not fall within the implied range (NT \* STU bytes) that starts at the first Untranslated Address within the Translation Request.
- If a Responder successfully validates and executes a Translation Request packet or execution fails for any reason returned in the *Translation Response* packet's Completion Status field, then the Responder shall transmit a *Translation Response* packet.
  - If packet validation or execution fails for any reason other than those returned in the *Translation Response* packet's Completion Status field, then the Responder shall transmit a *Standalone Acknowledgment* with Reason equal to the highest precedence error detected and shall not transmit a *Translation Response* packet.
- If a Translation Request packet requests multiple translations, then the Responder may transmit multiple *Translation Response* packets.
  - The Requester shall be responsible for determining when all requested translations have been completed and for performing any error recovery.
- If PV == 1b, then the PASID identifies the process address space where the Untranslated Address is located.

Table 20-2: Common Translation Request Packet Protocol Fields

Field Name	Size (bits)	Description
Untranslated Address	52	This is any address within the first STU (see <i>Component ATP Structure</i> ). Address bits 11:0 are not present and have an implicit value of 0x0.
NT	8	The number of translations requested starting at the indicated Address. If a LPD that supports PCO, then NT shall be less than or equal to 16. NT shall be a non-zero value.
NW	1	Indicates if read-only access permission is requested. A TA may ignore the NW bit and return a translation within a <i>Translation Response</i> packet with the W (Write) bit set to 1b.

Field Name	Size (bits)	Description
		<p>If NW == 0b, then a TA may mark the associated pages as modified / dirty. Due to potential negative performance impacts and / or functional impacts (e.g., if the underlying memory within the address range cannot be modified), a context should read-write access permission only if it intends to modify the address range.</p> <p>If a context subsequently determines that it requires write access, then it shall issue a new Translation Request packet with NW = 0b.</p> <p>0b—Read-Write Access Permission 1b—Read-only Access Permission</p>
EX	1	<p>EX (Execute Permission)—Indicates if execute permission is requested for the associated address range. If Execute Permission is granted for the translated address, then this shall be reflected in the <i>Translation Response</i> packet.</p> <p>0b—Non-executable Address Range 1b—Executable Address Range</p>
Priv	1	<p>Priv (Privileged)—Indicates if the Translation Request targets an address range associated with Privileged Mode operation. If Priv does not match the privilege associated with the address range, then the Responder shall handle the Translation Request packet as if the memory was not mapped. If Privileged Mode is granted for the translated address, then this shall be reflected in the <i>Translation Response</i> packet.</p> <p>0b—Non-privileged Mode 1b—Privileged Mode</p>

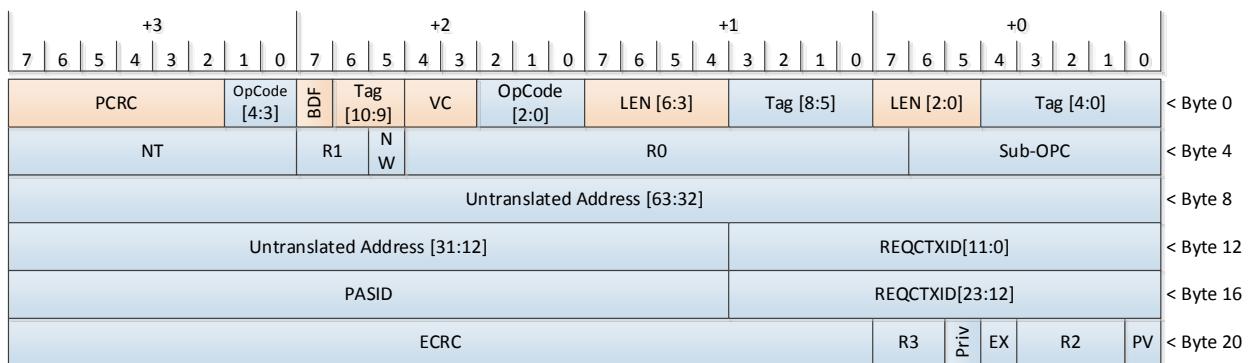


Figure 20-4: P2P-Coherency Translation Request Packet Format

Table 20-3: P2P-Coherency Translation Request Packet-specific Protocol Fields

Field Name	Size (bits)	Description
R0	14	Reserved

Figure 20-5: CTXID Translation Request Packet Format

Table 20-4: CTXID Translation Request Packet-specific Protocol Fields

Field Name	Size (bits)	Description
R0	1	Reserved
	3	Reserved

### 20.5.2. Translation Response

A Translation Response packet is used to communicate the execution results of the corresponding Translation Request packet. If present, then each Translation field indicates how the addressed resource can be accessed.

The following are the features and requirements of the Translation Response packet:

- A P2P-Coherency Translation Response packet shall use the *P2P-Coherency Translation Response Packet Format*.
  - A CTXID Translation Response packet shall use the *CTXID Translation Response Packet Format*.
  - The Completion Status field indicates the success or failure of the Translation Request packet.
    - If Completion Status == 0x0, then the packet shall contain one 8-byte Translation Field per requested translated address. If multiple Translation Fields are present, then they shall be contiguously placed with each representing the next contiguous address range, i.e., Translation Field 0 corresponds to range 0, Translation Field 1 corresponds to range 1, etc.
      - Starting Range field indicates the implicit range number associated with the first Translation Field within the Translation Response packet.

5

10

15

20

- If a single Translation Response packet is transmitted per *Translation Request* packet, then Starting Range shall equal 0x0.
- If multiple Translation Response packets are transmitted per *Translation Request* packet, then Starting Range shall equal 0x0 in the first Translation Response packet and shall be non-zero in the subsequent Translation Response packets.
- Ending Range field indicates the implicit range number associated with the last Translation Field within the Translation Response packet.
- All Translation Fields shall indicate the same Translation Range Size.
- If R = W = 0b, then the TA shall set S = 1b and encode the Translation Range Size in the lower bits of the Translated Address field. The contents of the Translated Address, N, and U bits are undefined.
- If U == 1b, then the indicated address range may be accessed only using untranslated addresses.
- If Completion Status ≠ 0x0, then the packet shall not contain any Translation Fields.
- A TA may return fewer Translation Fields than requested.
- A TA shall not return more Translation Fields than requested.
- A Responder may return Translation Response packets irrespective of the order that the corresponding *Translation Request* packets were received.
- A Requester shall provision sufficient space for all expected Translation Response packets without applying backpressure to the Gen-Z fabric.

Table 20-5: Common Translation Response Packet Protocol Fields

Field Name	Size (bits)	Description
<b>Completion Status</b>	3	Completion Status 0x0—Success. One or more Translation Fields shall be present. 0x1—Unsupported Request (UR). Translation requests from this component or context are not supported. The Translation Field shall not be present. 0x2-0x3—Reserved 0x4—TA Error. The TA was unable to translate the address due to a TA error. The Translation Field shall not be present. If a LPD, then this error may be mapped to a Completer Abort (CA) by the protocol translation logic. 0x5-0x7—Reserved
<b>Starting STU</b>	8	If Completion Status == Success, then Starting STU indicates the implicit STU number of the first Translation Field within this packet, else this field shall be Reserved.
<b>Ending STU</b>	8	If Completion Status == Success, then Ending STU indicates the implicit STU number of the last Translation Field within this packet, else this field shall be Reserved. If a single Translation Field is present, then Ending STU = Starting STU.

Field Name	Size (bits)	Description
<b>Translated Address</b>	52	<p>Address corresponding to the requested address range. If the R = W = 0b or U = 1b, then the Translated Address shall not be used in subsequent applicable request packets.</p> <p>If U == 0b and (R == 1b or W == 1b), then the translated address may be used in subsequent applicable request packets.</p> <p>If enabled and a context caches the Translated Address, then it shall cache the contents of the R and W fields.</p> <p>The Translated Address shall be encoded to indicate the size of the translated or non-translated address range.</p>
<b>N</b>	1	<p>N (Non-snooped accesses)—If N == 1b, then the translation corresponds to an address range that requires NS = 0b in applicable packets. If N == 0b, then other means may be used to determine how to set the NS bit in applicable packets.</p>
<b>R</b>	1	<p>R (Read Access Permission)—Indicates if read access permission was granted. Subsequent applicable read operations may target this address range.</p> <p>If EX == 1b, then R shall equal 1b.</p> <p>0b—Read Access Denied 1b—Read Access Granted</p>
<b>W</b>	1	<p>W (Write Access Permission)—Indicates if write access permission was granted. Granting write access permission also grants zero-length read access permission. Subsequent applicable write and zero-length read operations may target this address range.</p> <p>0b—Write Access Denied 1b—Write Access Granted</p> <p>If R = W = 0b, then the contents of the Translated Address, N, and U bits shall be undefined. The context shall not cache the translation.</p>
<b>U</b>	1	<p>U (Untranslated Access Only)—If U == 1b, then the context shall use untranslated addresses (TA == 0b) in any subsequent applicable request packet accessing the corresponding address range.</p> <p>If U == 0b and (R == 1b or W == 1b), then the Translated Address may be used and a context may cache the translation.</p> <p>If U == 1b and (R == 1b or W == 1b), then the context may use untranslated addresses to access the requested address range with the indicated access permission.</p> <p>If U == 1b and (R == 1b or W == 1b), then the context may cache the contents of the U, R, W, Ex, and Priv fields and the Translation Range Size. The TA shall transmit a <i>Translation Invalidate Request</i></p>

Field Name	Size (bits)	Description
EX		packet if the values associated with the untranslated address range change.
	1	<p>EX (Execute Permission)—Indicates if execute permission is granted for the associated address range. If Execute Permission is granted, then the context is permitted to execute code contained in the associated address range. The meaning of execute is outside of this specification's scope.</p> <p>If Priv == 1b, then the context is granted privileged execute permission. If Priv == 0b, then the context is granted non-privileged execute permission.</p> <p>If EX == 1b, then R shall equal 1b. The context may cache the contents of the EX and R fields.</p> <p>If the TA does not support or Execute Permission is not enabled, then shall set Ex 0b.</p> <p>0b—Execute Permission Not Granted 1b—Execute Permission Granted</p>
Priv	1	<p>Priv (Privileged)—Indicates if Privileged Mode was granted for the associated address range.</p> <p>If Priv == 1b, then R, W, and EX fields refer to permissions associated with Privileged Mode entities.</p> <p>If Priv == 0b, then R, W, EX fields refer to permissions associated with Non-privileged Mode entities.</p> <p>0b—Non-privileged Mode 1b—Privileged Mode</p>
GM	1	<p>G (Global Mapping)—If GM == 1b, then the context may cache the Translation Field contents in all PASIDs.</p> <p>If GM == 0b, then the context may cache the Translation Field contents only in the PASID of the corresponding <i>Translation Request</i> packet.</p> <p>If Global Mapping is unsupported, then shall set to GM = 0b and the context may transmit multiple <i>Translation Request</i> packets for the same untranslated address for independent PASID values.</p>
S	1	<p>S (Translation Range Size)—If S== 0b, then the translation corresponds to a 4096 byte translation range. If S == 1b, then the translation corresponds to a translation range larger than 4096 bytes. <i>Example Translation Range Sizes via Translated Address and S Fields</i> illustrates examples of how the Translated Address field is encoded to indicate the size of the translation range based on the S</p>

Field Name	Size (bits)	Description
		<p>field's encoding. Additional sizes may be supported beyond these examples including larger than 4 GiB.</p> <p>If S == 1b and Translated Address[63:12] == 0xFF FFFF FFFF FFFF, then the behavior is non-deterministic.</p> <p>If S == 1b and Translated Address[63:12] == 0xEF FFFF FFFF FFFF, then the context shall invalidate all translations.</p> <p>If the translation range size is smaller than the context's STU, then the Requester shall handle the Translation Response packet as a UR.</p>

Address Bits																	Translation Range Size
63:32	31	30	29	28	27	26	20	19	18	17	16	15	14	13	12	S	
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	4 KiB
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	1	8 KiB
X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	1	1	16 KiB
X	X	X	X	X	X	X	X	X	X	X	X	X	0	1	1	1	32 KiB
X	X	X	X	X	X	X	0	1	1	1	1	1	1	1	1	1	1 MiB
X	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4 GiB

Figure 20-6: Example Translation Range Sizes via Translated Address and S Fields

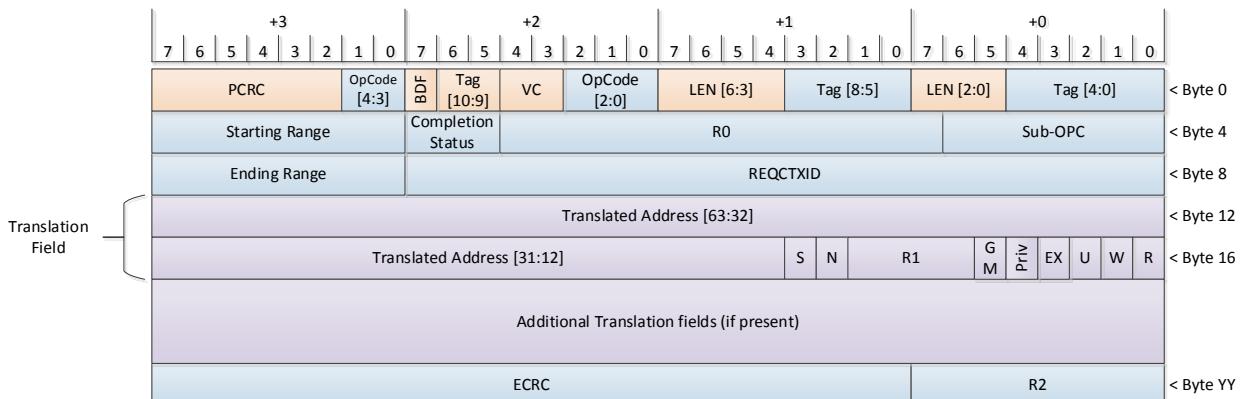


Figure 20-7: P2P-Coherency Translation Response Packet Format

Table 20-6: P2P-Coherency Translation Response Packet-specific Protocol Fields

Field Name	Size (bits)	Description							
<b>R0</b>	14	Reserved							
<b>R1</b>	4	Reserved							
<b>R2</b>	8	Reserved							

Figure 20-8: CTXID Translation Response Packet Format

Table 20-7: CTXID Translation Response Packet-specific Protocol Fields

Field Name	Size (bits)	Description							
<b>R0</b>	4	Reserved							
<b>R1</b>	8	Reserved							

### 20.5.3. Translation Invalidate Request

A Translation Invalidate Request packet is used to invalidate previously translated addresses maintained in the component that requested translation of a given address. A TA transmits Translation Invalidate Request packets to request a context to evict any translations corresponding to an untranslated address.

The following are the features and requirements of Translation Invalidate Request:

- A P2P-Coherency Translation Invalidate Request packet shall use the *P2P-Coherency Translation Invalidate Request Packet Format*.
- A CTXID Translation Invalidate Request packet shall use the *CTXID Translation Invalidate Request Packet Format*.
- A TA shall transmit a Translation Invalidate Request packet if any attribute of a previously translated address is changed. Attributes are only those communicated by a *Translation Response* packet.

- In order to ensure correct delivery and execution of Translation Invalidate Request packets, Requesters and Responders shall take the steps specified in *Non-Deterministic Request Execution*.
- A Requester may have a maximum of 32 outstanding Translation Invalidate Request packets to a given RSPCTXID. Each RSPCTXID shall be capable of receiving 32 Translation Invalidate Request packets without applying backpressure to any Gen-Z link.
- For each Translation Invalidate Request packet that is successfully validated and executed, the Responder shall transmit a single *Translation Invalidate Response* packet.
- If a Translation Invalidation Request packet corresponds to a page that was previously made resident using a *PRG Request* packet, then the Invalidation Request packet also serves as a page eviction notification.
- If the PASID field is Reserved, then the context shall:
  - Invalidate all cached entries within the invalidation range that were requested without a PASID value.
  - Invalidate all cached entries within the invalidation range for all addresses requested with any PASID value.
- Any of the following events shall cause a context to invalidate all cached translation entries (the component shall not transmit *Translation Invalidate Response* packets when these events occur):
  - *Component Reset* (all types)
  - Modification of the Address Translation Services Enable field in the *Component ATP Structure*. If the context is a LPD, then modification of the E (Enable) field to 1b shall cause it to invalidate all cached translation entries.
  - Context-specific content reset (e.g., if a LPD, a Function Level Reset event)
- If a context stops using a PASID or PASID Enable is reset to 0b, then the context shall invalidate all non-Global Mapping entries that were requested using the PASID within it's cache. The component shall not transmit *Translation Invalidate Response* packets when this occurs.
- Upon transmitting an Invalidation Request packet, the TA shall start a timer using *Core Structure* ATSTO. If a TA does not receive an *Translation Invalidate Response* packet before this timer expires, then the TA shall initiate TA-specific error recovery.

Table 20-8: Common Translation Invalidate Request Packet Protocol Fields

Field Name	Size (bits)	Description
Untranslated Address	42	Untranslated Address
S	1	<p>S (Translation Range Size)—If S== 0b, then the Untranslated Address corresponds to a 4096 byte invalidation range. If S == 1b, then the Untranslated Address corresponds to an invalidation range larger than 4096 bytes.</p> <p>If S == 1b and Untranslated Address[63:12] == 0xFF FFFF FFFF FFFF, then the behavior is non-deterministic.</p> <p>If S == 1b and Translated Address[63:12] == 0xEF FFFF FFFF FFFF, then the context shall invalidate all translations.</p>

Field Name	Size (bits)	Description
		If the translation range size is smaller than the context's STU, then the context shall round up the range to be greater than or equal to the context's STU.
GI	1	<p>GI (Global Invalidate)—Indicates if the invalidation is applicable to all supported PASIDs. If Global Invalidate or Global Mapping is unsupported or disabled, then the TA shall set GI == 0b.</p> <p>If GI == 1b, then the TA shall set PV = 0b and the PASID field shall be Reserved.</p> <p>0b—If PASID is valid, then invalidate only non-Global mappings within the invalidation range using the associated PASID. Global mapping entries within the invalidation range may be retained.</p> <p>1b—Invalidate the Global and non-Global mappings within the invalidation range for any PASID.</p>
RSPCTXID	24	Identifies the context that generated the <i>Translation Request</i> packet corresponding to the untranslated address.

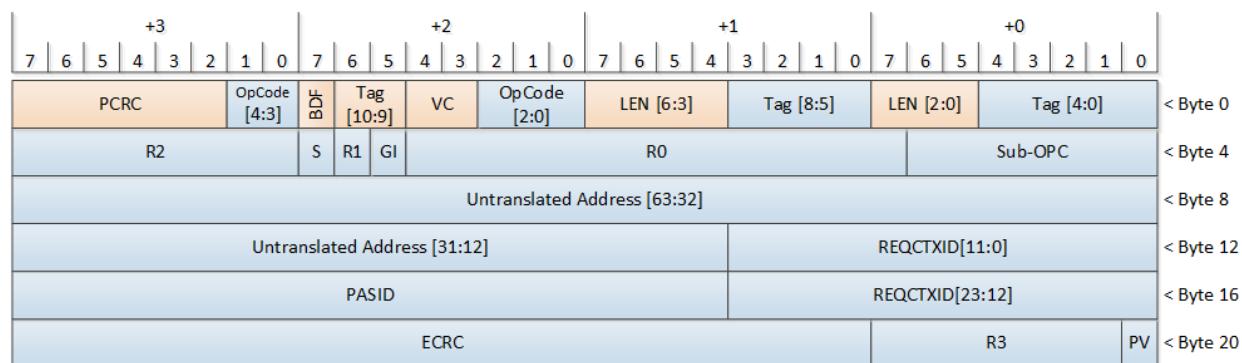


Figure 20-9: P2P-Coherency Translation Invalidate Request Packet Format

Table 20-9: P2P-Coherency Translation Invalidate Request Packet-specific Protocol Fields

Field Name	Size (bits)	Description
R0	14	Reserved
R1	1	Reserved
R2	8	Reserved
R3	7	Reserved

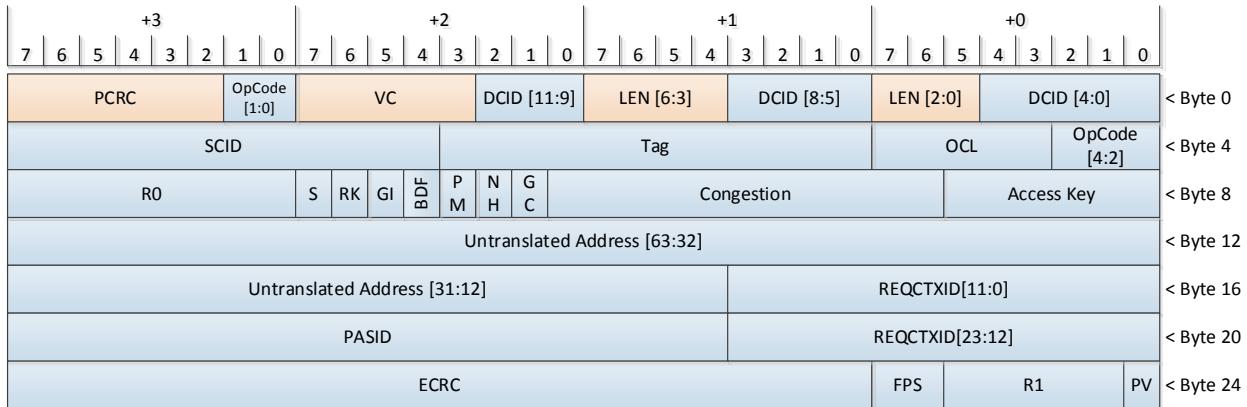


Figure 20-10: CTXID Translation Invalidate Request Packet Format

Table 20-10: CTXID Translation Request Packet-specific Protocol Fields

Field Name	Size (bits)	Description
R0	8	Reserved
R1	5	Reserved

## 20.5.4. Translation Invalidate Response

A Translation Invalidate Response packet is transmitted in response to a Translation Invalidate Request packet once the Responder has discontinued use of the corresponding translated address.

The following are the features and requirements of the Translation Invalidate Response packet:

- A P2P-Coherency Translation Invalidate Response packet shall use the *P2P-Coherency Translation Invalidate Response Packet Format*.
- A CTXID Translation Invalidate Response packet shall use the *CTXID Translation Invalidate Response Packet Format*.
- If Address Translation Services is supported, then a Responder shall transmit a single Translation Invalidate Response packet per received *Translation Invalidate Request* packet independent of the value of *Component ATP Structure* Address Translation Services Enabled. Prior to transmitting a Translation Invalidate Response packet, the context shall ensure:
  - There are no request packets using translated addresses that correspond to the invalidation range.
  - No new request packets shall use any translated address that corresponds to the invalidation range.
- Upon receipt of an *Translation Invalidate Request* packet, the context shall start an invalidation timer using (*Core Structure* ATSTO – 1). If the timer expires prior to transmitting a Translation Invalidate Response packet, then the context shall immediately discontinue use and remove any cache entries corresponding to the translated address and shall transmit a Translation Invalidate Response packet.
- A Requester shall silently discard any Translation Invalidate Response packet that does not correspond to any outstanding *Translation Invalidate Request* packet.

- *Translation Invalidate Request* and *Translation Response* packets from a TA may arrive out of order. Each context's cache is responsible for detecting if an invalidation range overlaps any translation address range within any outstanding *Translation Request* packet. If an overlap is detected, then the outstanding *Translation Request* packet shall be marked as invalid (implementation-specific method) and the context should silently discard the subsequently received *Translation Response* packet. If the subsequently received *Translation Response* packet is not silently discarded, then the Translation Field(s) that overlap the invalidation range shall be discarded.

- If the context receives the *Translation Response* packet prior to scheduling *Translation Invalidate Response* packet transmission, then the context may use the translated addresses in new request packets if these request packets are guaranteed to complete prior to the expiration of the invalidation timer (see above).

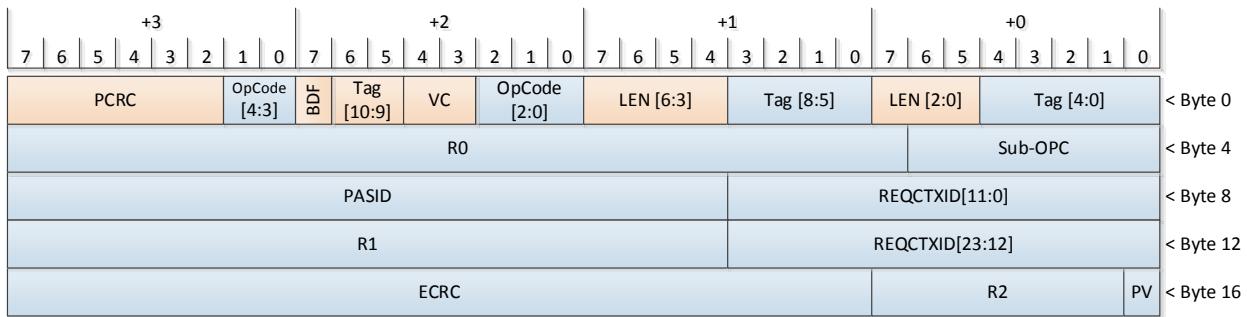


Figure 20-11: P2P-Coherency Translation Invalidate Response Packet Format

Table 20-11: P2P-Coherency Translation Invalidate Response Packet Protocol Fields

Field Name	Size (bits)	Description
RSPCTXID	24	Identifies the originating context that generated the corresponding translation request. If the originating context is a LPD, then the LPD's BDF shall be placed in RSPCTXID[15:0] and RSPCTXID[23:16] shall be Reserved.
R0	25	Reserved
R1	20	Reserved
R2	7	Reserved

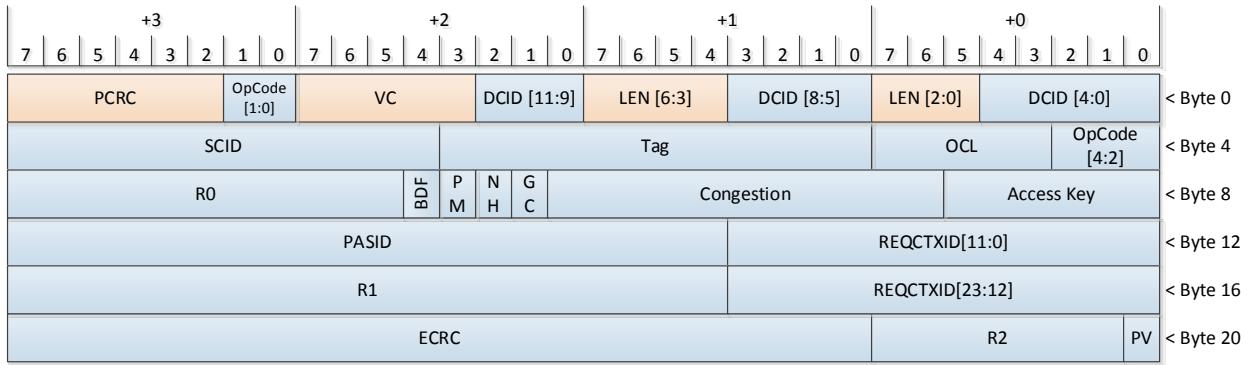


Figure 20-12: CTXID Translation Invalidate Response Packet Format

Table 20-12: CTXID Translation Invalidate Response Packet Protocol Fields

Field Name	Size (bits)	Description
RSPCTXID	24	<p>Identifies the originating context that generated the corresponding translation request.</p> <p>If the originating context is a LPD, then the LPD's BDF shall be placed in RSPCTXID[15:0] and RSPCTXID[23:16] shall be Reserved.</p>
R0	11	Reserved
R1	20	Reserved
R2	7	Reserved

## 20.6. Page Services

Page services are used by a Requester to request memory pages be made resident upon demand. On-demand paging can be used to improve solution performance and alleviate memory pressure due to excessive “pinning” of memory.

### 20.6.1. PRG Request

A PRG Request packet is transmitted by the Page Request Interface (PRI) associated with a Requester context.

The following are the features and requirements of the PRG Request packet:

- A P2P-Coherency PRG Request packet shall use the *P2P-Coherency PRG Request Packet Format*.
  - A CTXID PRG Request packet shall use the *CTXID PRG Request Packet Format*.
  - If a context is a LPD that requires PCO, then it shall transmit PRG Request packets and receive all *PRG Response Notification* packets on a single interface VC<sub>K</sub> on which PCO is enabled. If a context does not require PCO, then it may transmit PRG Request packets and receive *PRG Response Notification* packets on any interface and VC used to communicate with the Responder.
  - A PRG Request packet has a non-deterministic execution time. To ensure Reliable Delivery, a PRG Request-PRG Response packet exchange shall comply with *Non-Deterministic Request Execution*.

- If PCO is required, then the Requester / context shall not start a VET (Variable Execution Timer).
- If PCO is not required, then the *Core Structure* ATSTO shall be used as the VET.
- A component or context may provide a PASID value in a PRG Request packet. The PASID represents the address space of the requested page.
- If multiple pages are requested for a given PRG Index value, then:
  - If a PASID value is provided, then the PASID value in all of the corresponding PRG request packets' PASID fields shall be identical.
    - If the PASID value differs in any PRG request packets associated with the same PRG Index value, then the Responder shall transmit a *PRG Response Notification* packet with Responder Code = "Invalid Request" for this and all subsequent Page Request packets associated with the PRG Index value.
  - If PCO is required, then individual CTXID PRG Request packets associated with a given PRG Index shall not be independently acknowledged unless an error is detected.
  - If PCO is not required or is unsupported, then each PRG request packet shall be independently acknowledged by a *Standalone Acknowledgment*. The Requester shall be responsible for determining when all outstanding PRG Request packets have been acknowledged prior to transmitting the last PRG Request packet.
    - If the VET expires and a *Standalone Acknowledgment* or a *PRG Response Notification* packet was not received, then the Requester shall retransmit the corresponding PRG Request packet.
- Multiple PRG Request packets may request the same page independent of the PRG Index value.
- If all PRG Request packets associated with a PRG Index are successfully validated and executed, then upon receipt of the last PRG Request packet (indicated by L = 1b) the Responder shall transmit a single *PRG Response Notification* packet to acknowledge all PRG Request packets associated with the PRG Index.
- If an error is detected and the detected error is an "Invalid Request" or "Response Failure", then the Responder shall transmit a single *PRG Response Notification* packet with the RSP Code indicating the error; for all other errors, the Responder shall transmit a *Standalone Acknowledgment* with Reason set to the highest precedence error detected and shall not transmit a *PRG Response Notification* packet. Further, the entire sequence of *PRG Request* packets shall be treated as failed, i.e., the state of any of the requested pages is non-deterministic.
- A component or context may have a maximum of Outstanding PRG Allocation (see *Component ATP Structure*) PRG Request packets outstanding at a given time. If per context management is supported, then the maximum outstanding PRG Requests packets across all contexts shall be less than or equal to Outstanding PRG Allocation.
  - If the number of outstanding PRG Request packets a component or context transmits exceeds the allocated value, then the Responder shall transmit a *PRG Response Notification* packet with RSP Code = Response Failure and, upon receipt, the context shall disable the corresponding PRI.

Table 20-13: Common PRG Request Packet Protocol Fields

Field Name	Size (bits)	Description
Page Address	52	The untranslated address of the requested page. The minimum requested page size is 4096 bytes. To request a larger page, the

Field Name	Size (bits)	Description
		least significant bits of the Page Address field shall be set to zero. For example, if page size equals 8192 bytes, then Page Address[12] = 0b.
R	1	R (Read Access Request)—Indicates if the context requests read access to the requested page.  If EX == 1b, then shall set R = 1b.  0b—Read Access Not Requested 1b—Read Access Requested
W	1	W (Write Access Requested)—Indicates if the context requests write access and / or zero-length read access to the requested page. If write access is requested, then the Responder may treat the associated page as modified.  Due to potential negative performance impacts and / or functional impacts (e.g., if the underlying storage cannot be modified), a context shall request write access only if it intends to modify the page and has been granted permission by software or the solution through means outside of this specification’s scope.  A TA may deny write access permission (see RW in <i>PRG</i> ).  0b—Write Access Not Requested 1b—Write Access Requested
L	1	L (Last Request in <i>PRG</i> )—Indicates if this is the last <i>PRG</i> Request packet associated with the <i>PRG</i> Index value.  0b—Additional <i>PRG</i> Request Packets Forthcoming 1b—Last / Only <i>PRG</i> Request Packet
EX	1	EX (Execute Permission)—Indicates if execute permission is requested for the associated page. If Execute Permission is granted for the page, then this shall be reflected in the <i>PRG Response Notification</i> packet.  0b—Non-executable Address Range 1b—Executable Address Range
Priv	1	Priv (Privileged)—Indicates if the <i>PRG</i> Request packet targets a page associated with Privileged Mode operation. If Priv does not match the privilege associated with the page, then the Responder shall return a <i>PRG Response Notification</i> packet with RSP Code = Invalid Request. If Privileged Mode is granted for the page, then this shall be reflected in the <i>PRG Response Notification</i> packet.  0b—Non-privileged Mode 1b—Privileged Mode
PRG Index	9	Identifies the <i>PRG</i> associated with the REQCTXID.

Field Name	Size (bits)	Description
<b>R0</b>	2	Reserved

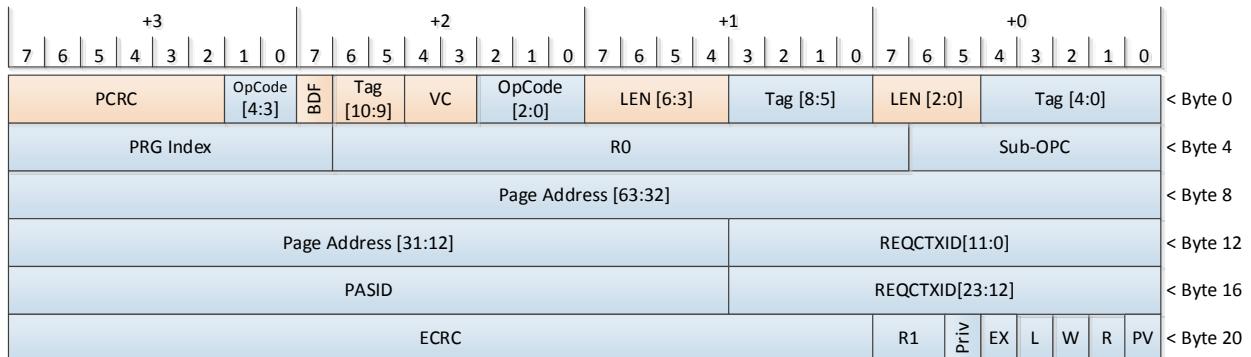


Figure 20-13: P2P-Coherency PRG Request Packet Format

Table 20-14: P2P-Coherency PRG Request Packet-specific Protocol Fields

Field Name	Size (bits)	Description
<b>R0</b>	16	Reserved
<b>R1</b>	2	Reserved

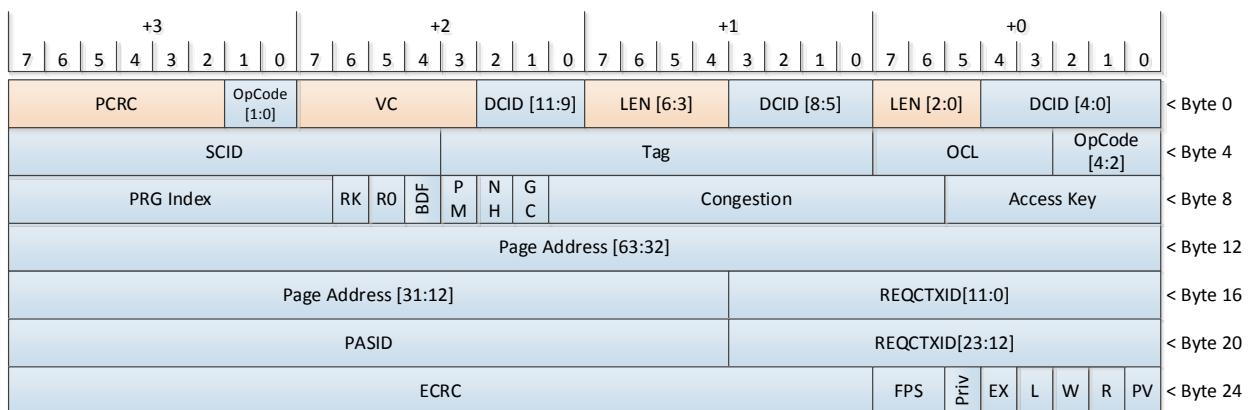


Figure 20-14: CTXID PRG Request Packet Format

Table 20-15: CTXID PRG Request Packet-specific Protocol Fields

Field Name	Size (bits)	Description
<b>R0</b>	1	Reserved

## 20.6.2. PRG Response Notification

A PRG Response Notification is used to indicate the success or failure of all outstanding *PRG Request* packets associated with PRG Index N.

The following are the features and requirements of the PRG Response Notification packet:

- 5 A P2P-Coherency PRG Response Notification packet shall use the *P2P-Coherency PRG Response Notification Packet Format*.
- 10 A CTXID PRG Response Notification packet shall use the *CTXID PRG Response Notification Packet Format*.
- 15 A Response Notification packet shall be acknowledged by a *Standalone Acknowledgment*.
- 20 A PRG Index or a PRG Index combined with a PASID is used to identify a PRG (a set of *PRG Request* packets). If the PRG Index or a PRG Index combined with a PASID does not correspond to any outstanding *PRG Request* packets, then the Responder shall set *Component ATP Structure* ATP Status Unexpected PRG Index = 1b, Failed REQCTXID = REQCTXID, and shall generate a *Standalone Acknowledgment* (Reason = No Error).
- 25 A Requester-Responder shall transmit a single PRG Response Notification packet only upon receipt of a *PRG Request* packet with L == 1b or upon detecting an error in any *PRG Request* packet and the detected error is an “Invalid Request” or “Response Failure”; for all other errors, the Responder shall transmit a *Standalone Acknowledgment* with Reason set to the highest precedence error detected and shall not transmit a PRG Response Notification packet.
- 30 If the PASID field in the corresponding *PRG Request* packet contained a PASID, then the Requester-Responder shall reflect that PASID within the PRG Response Notification packet’s PASID field.
  - o If the PRG Response Notification packet’s PASID field does not contain a value or does not match the expected PASID, then the context’s behavior is non-deterministic.
- 35 A context shall be capable of receiving one PRG Response Notification packet for each outstanding sequence of *PRG Request* packets identified by a given PRG Index. Further, to avoid deadlock, the context shall be capable of processing each PRG Response Notification packet without transmitting or receiving any other packet.

Table 20-16: Common PRG Response Notification Packet Protocol Fields

Field Name	Size (bits)	Description
PRG Index	9	Identifies the PRG associated with the REQCTXID.
RSP Code	2	<p>Response Code—Indicates success or failure and the associated reason.</p> <p>0x0—No Error. All pages associated with the PRG Index are resident.</p> <p>0x1—Invalid <i>PRG Request</i> packet:</p> <ul style="list-style-type: none"> <li>• Invalid Untranslated Address (page does not exist)</li> <li>• Invalid PASID or PASID Is disabled</li> <li>• EX == 1b and R ==0b</li> <li>• Invalid access attributes (R, W, EX, Priv)</li> </ul>

Field Name	Size (bits)	Description
		<ul style="list-style-type: none"> <li>• Disabled context PRI</li> <li>• Unable to fulfill page requests at this time</li> </ul> <p>0x2—Reserved</p> <p>0x3—Response Failure. One or more pages associated with the PRG Index suffered an unrecoverable error. The indicated context (REQCTXID) shall ignore all subsequently received PRG Response Notification packets. Upon receipt of this code, the context PRI shall be disabled and the context shall not transmit new <i>PRG Request</i> packets until the PRI is enabled.</p>
<b>RW</b>	1	<p>Read-Write Access Permission—Indicates if the page has read-only or read-write access permission.</p> <p>0b—Read-only 1b—Read-Write</p>
<b>EX</b>	1	<p>EX (Execute Permission)—Indicates if execute permission is granted for the associated page(s). If Execute Permission is granted, then the context is permitted to execute code contained in the associated page(s). The meaning of execute is outside of this specification's scope.</p> <p>If <b>Priv</b> == 1b, then the context is granted privileged execute permission. If <b>Priv</b> == 0b, then the context is granted non-privileged execute permission.</p> <p>If the TA does not support or Execute Permission is not enabled, then shall set <b>Ex</b> = 0b.</p> <p>0b—Execute Permission Not Granted 1b—Execute Permission Granted</p>
<b>Priv</b>	1	<p>Priv (Privileged)—Indicates if Privileged Mode was granted for the associated page.</p> <p>If <b>Priv</b> == 1b, then <b>RW</b> and <b>EX</b> fields refer to permissions associated with Privileged Mode entities.</p> <p>If <b>Priv</b> == 0b, then <b>RW</b> and <b>EX</b> fields refer to permissions associated with Non-privileged Mode entities.</p> <p>0b—Non-privileged Mode 1b—Privileged Mode</p>

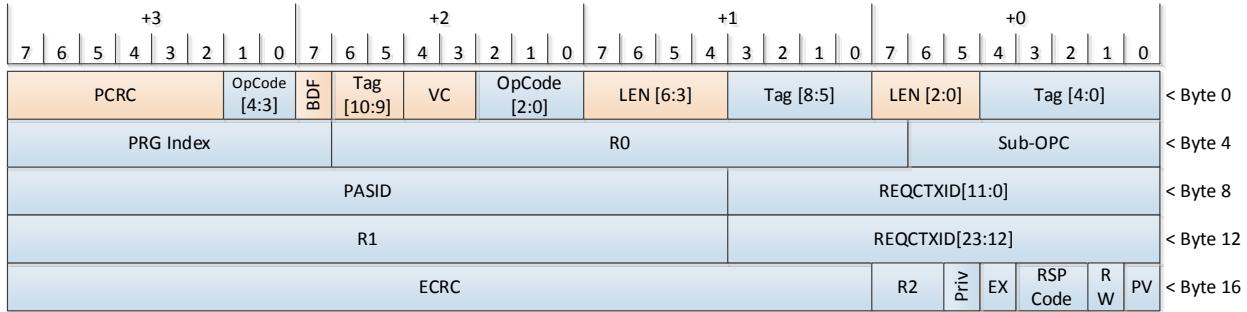


Figure 20-15: P2P-Coherency PRG Response Notification Packet Format

Table 20-17: P2P-Coherency PRG Response Notification Packet-specific Protocol Fields

Field Name	Size (bits)	Description
<b>R0</b>	16	Reserved
<b>R1</b>	20	Reserved
<b>R2</b>	2	Reserved

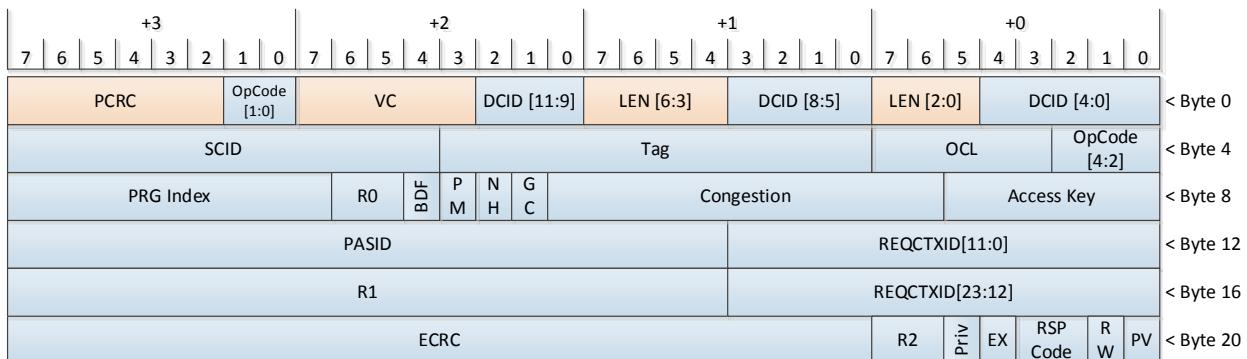


Figure 20-16: CTXID PRG Response Notification Packet Format

Table 20-18: CTXID PRG Response Notification Packet-specific Protocol Fields

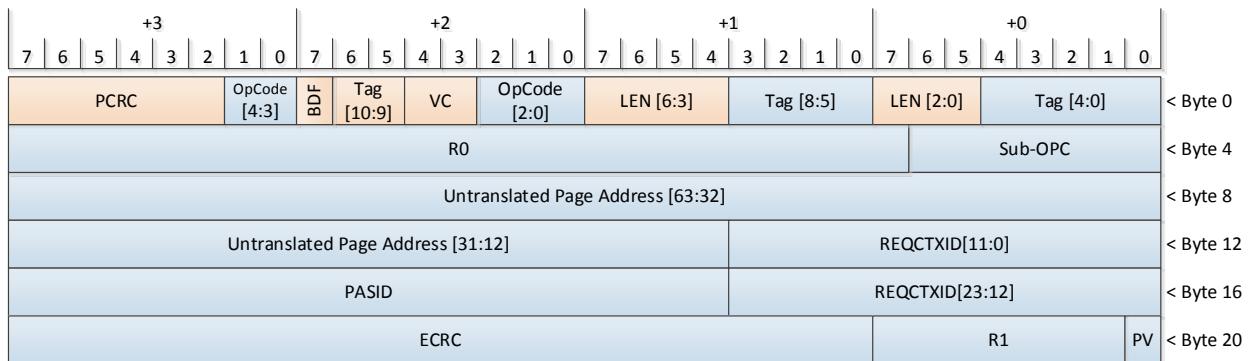
Field Name	Size (bits)	Description
<b>R0</b>	2	Reserved
<b>R1</b>	20	Reserved
<b>R2</b>	2	Reserved

### 5 20.6.3. PRG Release

A PRG Release is used to indicate that a context no longer requires access to the indicated page.

The following are the features and requirements of the PRG Response Notification packet:

- A P2P-Coherency PRG Release packet shall use the *P2P-Coherency PRG Release / PRG Eviction Notification Packet Format*.
- A CTXID PRG Release packet shall use the *CTXID PRG Release / PRG Eviction Notification Packet Format*.
- 5 • A P2P-Coherency PRG Release packet shall be acknowledged by a P2P-Coherency *Standalone Acknowledgment* within *Core Structure* RDLAT time.
- A CTXID PRG Release packet shall be acknowledged by a Core 64 *Standalone Acknowledgment* within the Responder's *Core Structure* Responder Deadline for explicit OpClass request packets RDLAT time.
- 10 • The Responder shall successfully acknowledge a PRG Release packet even if the Untranslated Page Address does not match any resident or tracked page.
- Once transmitted, a context shall not schedule any new request packets that target the released page.



15 Figure 20-17: P2P-Coherency PRG Release / PRG Eviction Notification Packet Format

Table 20-19: P2P-Coherency PRG Release / PRG Eviction Notification Packet Protocol Fields

Field Name	Size (bits)	Description
Untranslated Page Address	52	Untranslated address of the corresponding page
R0	25	Reserved
R1	7	Reserved

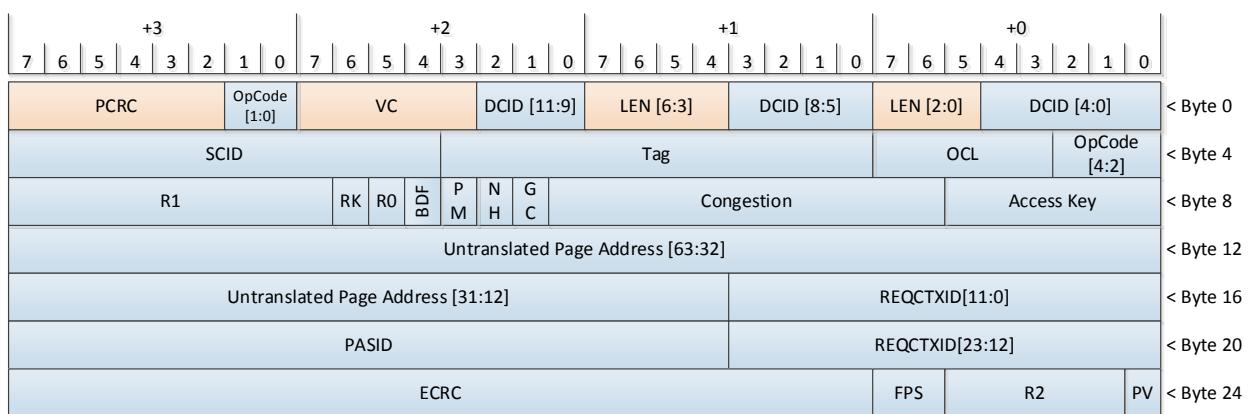


Figure 20-18: CTXID PRG Release / PRG Eviction Notification Packet Format

Table 20-20: CTXID PRG Release / PRG Eviction Notification Packet Protocol Fields

Field Name	Size (bits)	Description
<b>Untranslated Page Address</b>	52	Untranslated address of the corresponding page
<b>R0</b>	1	Reserved
<b>R1</b>	9	Reserved
<b>R2</b>	5	Reserved

## 20.6.4. PRG Eviction Notification

If the address of a page was translated, then a *Translation Invalidate* Request packet is used to inform a context that a page is no longer resident. If the address was not translated, then a PRG Eviction Notification request packet is used to inform a context.

- 5 The following are the features and requirements of the PRG Eviction Notification request packet:
- A P2P-Coherency PRG Eviction Notification request packet shall use the *P2P-Coherency PRG Release / PRG Eviction Notification Packet Format*.
  - A CTXID PRG Eviction Notification request packet shall use the *CTXID PRG Release / PRG Eviction Notification Packet Format*.
  - 10 • A PRG Eviction Notification request packet shall be acknowledged by a *Standalone Acknowledgment*.
    - The Responder shall acknowledge a PRG Eviction Notification request packet even if the untranslated address does not match any page accessed by the context.
  - 15 • In order to ensure correct delivery and execution of PRG Eviction Notification request packets, Requesters and Responders shall take the steps specified in *Non-Deterministic Request Execution*.
  - Upon receipt, a context shall:
    - Stop queuing new request packets that target the evicted page.
    - Wait for all request packets that targeted the evicted page to complete.
    - 20 ○ Once all request packets have completed, then the context shall transmit a *Standalone Acknowledgment*.

## 20.6.5. Stop Marker

Upon being informed to stop using a PASID, a context may indicate that it will transmit a Stop Marker packet as described in *PASID Stop Sequence*.

- 25 The following are the features and requirements of the Stop Marker request packet:
- A P2P-Coherency Stop Marker request packet shall use the *P2P-Coherency Stop Marker Packet Format*.
  - A CTXID Stop Marker request packet shall use the *CTXID Stop Marker Packet Format*.
  - 30 • Prior to transmitting a Stop Marker request packet, a context shall use a context-specific mechanism to inform the Responder that this PASID value will no longer be used.

- Upon receipt of a Stop Marker request packet, any subsequent use of the same PASID value shall be treated as a new version of the PASID, e.g., an independent process address space that reused the same PASID value.
  - The PASID field contain a valid PASID value and shall set PV = 1b. If the PASID field is invalid or contains a value that does not match any outstanding *PRG Request* packets, then the behavior is non-deterministic.
  - A Stop Marker request packet shall be acknowledged by a *Standalone Acknowledgment*.
  - Upon receipt of a Stop Marker request packet, software can safely take any action regarding the page, e.g., swapping the page out to secondary storage.
  - Stop Marker request packets are not counted as an outstanding allocation page services packet (see *Component ATP Structure Outstanding PRG Allocation*).

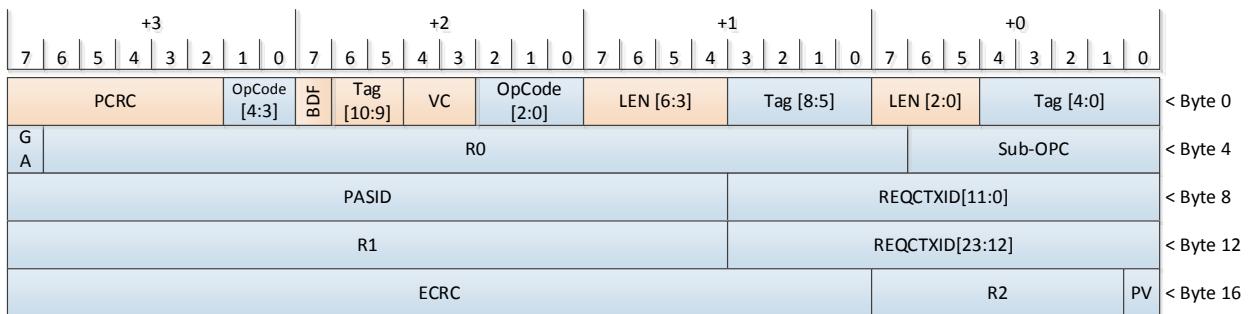


Figure 20-19: P2P-Coherency Stop Marker Packet Format

Table 20-21: P2P-Coherency Stop Marker Packet Protocol Fields

Field Name	Size (bits)	Description
GA	1	Indicates if the Stop Marker request packet is to be acknowledged. If a LPD and the LPD is enabled to use PCO, then shall set GA = 0b. If the REQCTXID is not associated with a LPD or if associated with LPD that is not enabled to use PCO, then shall set GA = 1b. 0b—Unacknowledged 1b—Acknowledged
R0	24	Reserved
R1	20	Reserved
R2	7	Reserved

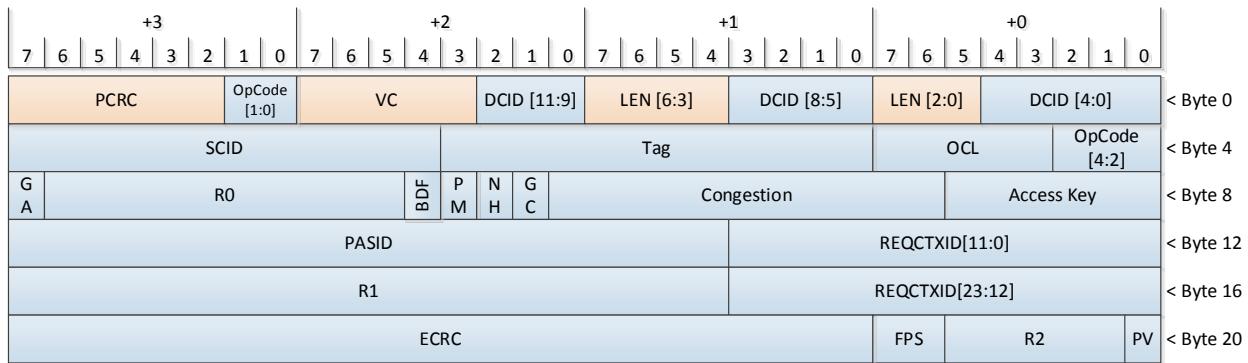


Figure 20-20: CTXID Stop Marker Packet Format

Table 20-22: CTXID Stop Marker Packet Protocol Fields

Field Name	Size (bits)	Description
<b>GA</b>	1	Indicates if the Stop Marker request packet is to be acknowledged. If a LPD and the LPD is enabled to use PCO, then shall set GA = 0b. If the REQCTXID is not associated with a LPD or if associated with LPD that is not enabled to use PCO, then shall set GA = 1b. 0b—Unacknowledged 1b—Acknowledged
<b>R0</b>	10	Reserved
<b>R1</b>	20	Reserved
<b>R2</b>	5	Reserved

# Appendix A Control Space Structure Encodings

This appendix provides a list of Control Space structure Type encodings. All Control Space structures shall have a unique Type encoded within the structure for identity purposes. These encodings are listed in *Control Space structure Type Encodings*.

Table A-1: Control Space structure Type Encodings

Encoded Type	Control Space structure Name
0x0	<i>Core Structure</i>
0x1	<i>OpCode Set Structure</i>
0x2	<i>Interface Structure</i>
0x3	<i>Interface PHY Structure</i>
0x4	<i>Interface Statistics Structure</i>
0x5	<i>Component Error and Signal Event Structure</i>
0x6	<i>Component Media Structure</i>
0x7	<i>Component Switch Structure</i>
0x8	<i>Component Statistics Structure</i>
0x9	<i>Component Extension Structure</i>
0xA	<i>Vendor-defined structure without a UUID field (see Vendor-Defined Structure)</i>
0xB	<i>Vendor-defined with UUID structure (see Vendor-Defined Structure)</i>
0xC	<i>Component Multicast Structure</i>
0xD	<i>Component Security Structure</i>
0xE	<i>Component TR Structure</i>
0xF	<i>Component Image Structure</i>
0x10	<i>Component Precision Time Structure</i>
0x11	<i>Component Mechanical Structure</i>
0x12	<i>Component Destination Table Structure</i>
0x13	<i>Service UUID Structure</i>
0x14	<i>Component C-Access Structure</i>
0x15	<i>Responder Bank Structure</i>
0x16	<i>Requester Bank Structure</i>

Encoded Type	Control Space structure Name
0x17	<i>Component PA (Peer Attribute) Structure</i>
0x18	<i>Component Event Structure</i>
0x19	<i>Component LPD Structure</i>
0x1A	<i>Component SOD Structure</i>
0x1B	<i>Congestion Management Structure</i>
0x1C	<i>Component RKD Structure</i>
0x1D	<i>Component PM Structure</i>
0x1E	<i>Component ATP Structure</i>
0x1F	<i>Component RE Table Structure</i>
0x20	<i>Component LPH Structure</i>
0x21-0xFFFF	Reserved

# Appendix B CRC

The architecture supports packet two CRCs—a 6-bit CRC and a 24-bit CRC. The architecture also supports a 16-bit flit CRC that is selectively enabled based on the enabled physical layer.

## 6-bit CRC

5 The following are the features and requirements associated with 6-bit CRC:

- The 6-bit CRC polynomial shall be 0x37.
- The 6-bit CRC seed value shall be 0x3F.
- The 6-bit CRC covers the following bits within every packet format: byte 0 bits [7:5], byte 1 bits [7:4], and byte 2 bits [7:3]. The 6-bit CRC is calculated as illustrated in *6-bit CRC Calculation* starting at bit 4 of byte 0 and progresses through bit 7, progresses through bit 4 to bit 7 of byte 1, and progresses through bit 3 to bit 7 byte 2.
- The remainder of the 6-bit CRC calculation is complemented and the complemented result bits shall be mapped into the applicable CRC field as illustrated in *Mapping 6-bit CRC Calculated Bits into the CRC Field*.
- If the component supports VC remapping, and the VC field is remapped to a new value, then the 6-bit PCRC field is modified as illustrated in *VC Remapped PCRC Modification*. If the original PCRC was invalid, then the packet shall be handled per transient error recovery. Errors within the original PCRC field's value will be present in the new PCRC value.

10 15 Table B-1: Mapping 6-bit CRC Calculated Bits into the CRC Field

Result Bit	Mapped Bit Position within CRC Field
0	5
1	4
2	3
3	2
4	1
5	0

Polynomial 0x37 (Koopman notation) means:

$$g(x) = x^6 + x^5 + x^3 + x^2 + x + 1$$

Note that the “+ 1” is inferred.

This polynomial is sometimes represented as 0x2f (with the  $x^6$  inferred).

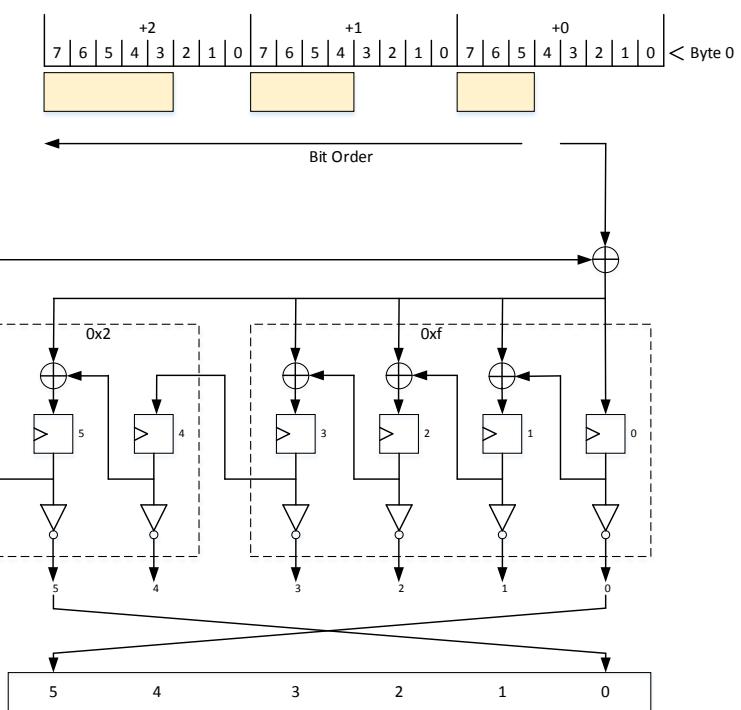


Figure B-1: 6-bit CRC Calculation

Updating PCRC when VC field changes. The following shows how the PCRC is adjusted, in principle. If there are any errors in the “old” PCRC code word, they will be preserved, and are detected when the “new” PCRC is eventually checked.

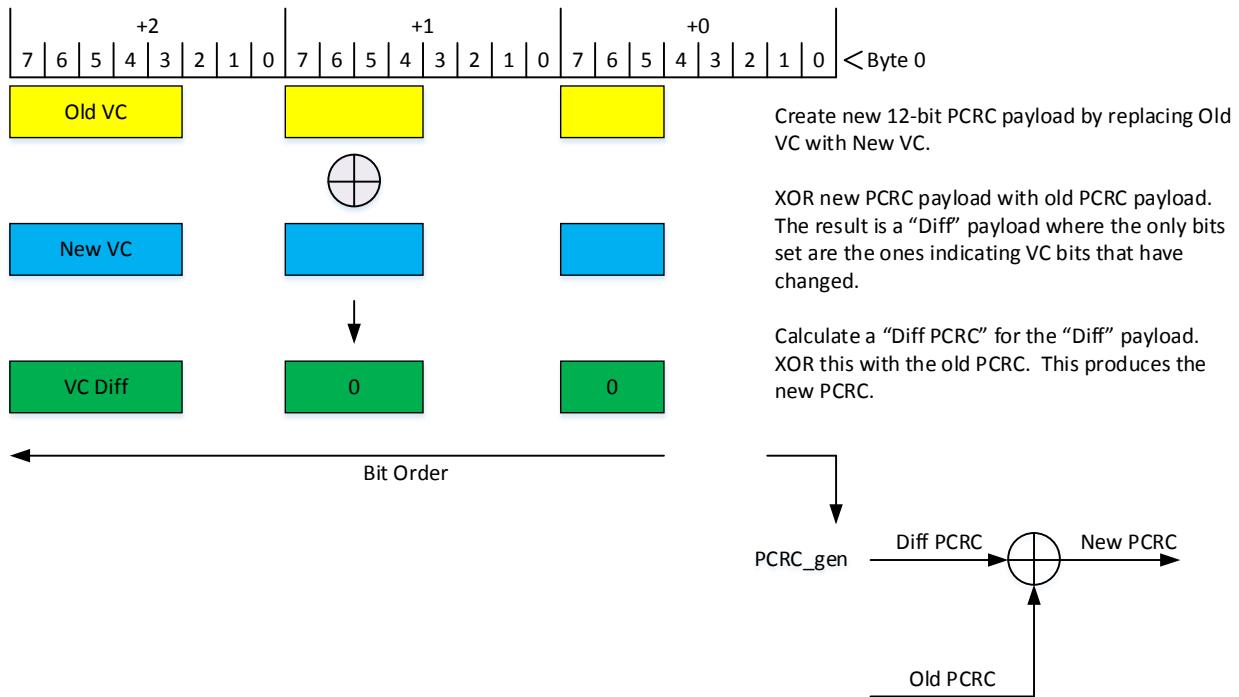


Figure B-2: VC Remapped PCRC Modification

## 24-bit CRC

The 24-bit CRC shall be used only in packets with a 24-bit ECRC field.

- 5 The 24-bit CRC polynomial shall be 0xADE27A.
- 10 The 24-bit CRC seed value shall be 0xFF FFFF.
- 15 All packet protocol fields (with the exception of the 24-bit CRC field) and, if present, payload including any pad bytes shall be included in the 24-bit CRC calculation. As illustrated in *24-bit CRC Calculation*, the calculation starts with bit 0 of byte 0 progresses from bit 0 to bit 7 of each byte of the packet.
- The remainder of the 24-bit CRC calculation is complemented and the complemented result bits shall be mapped into the applicable CRC field as illustrated in *Mapping 24-bit CRC Calculated Bits into the CRC Field*.
- If the component supports VC remapping, and the VC field is remapped to a new value and / or the Congestion field is modified, then the 24-bit ECRC field is modified as illustrated in *Figure B-4* (the PCRC is modified prior to modifying the ECRC). If the original ECRC was invalid, then the packet shall be handled per transient error recovery. Errors within the original ECRC field’s value will be present in the new ECRC value.

Table B-2: Mapping 24-bit CRC Calculated Bits into the CRC Field

Result Bit	Mapped Bit Position within CRC Field	Result Bit	Mapped Bit Position within CRC Field
<b>0</b>	23	12	11
<b>1</b>	22	13	10
<b>2</b>	21	14	9
<b>3</b>	20	15	8
<b>4</b>	19	16	7
<b>5</b>	18	17	6
<b>6</b>	17	18	5
<b>7</b>	16	19	4
<b>8</b>	15	20	3
<b>9</b>	14	21	2
<b>10</b>	13	22	1
<b>11</b>	12	23	0

Polynomial 0xade27a (Koopman notation) means:

$$g(x) = x^{24} + x^{22} + x^{20} + x^{19} + x^{17} + x^{16} + x^{15} + x^{14} + x^{10} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$$

Note that the “+ 1” is inferred.

This polynomial is sometimes represented as 0x5bc4f5 (with the  $x^{24}$  inferred).

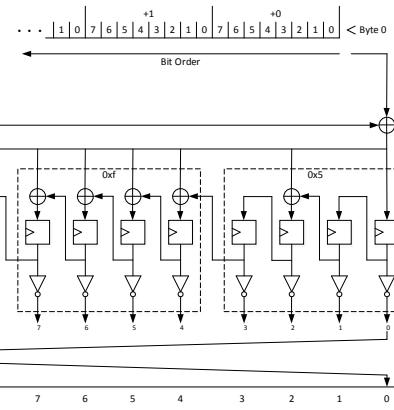


Figure B-3: 24-bit CRC Calculation

Updating ECRC when Congestion and / or VC field changes. The following shows how the ECRC is adjusted, in principle. If there are any errors in the “old” ECRC code word, they will be preserved, and are detected when the “new” ECRC is eventually checked.

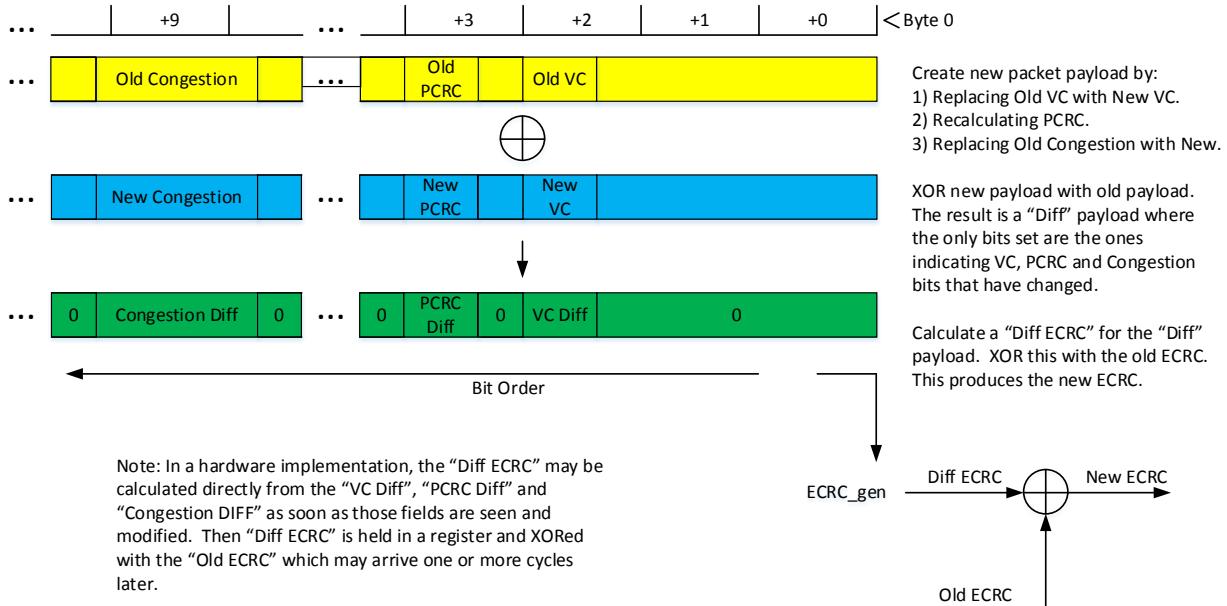


Figure B-4: VC Remapped and Congestion ECRC Modification

## 16-bit CRC

The 16-bit CRC shall be used only to protect only Flits with a 16-bit CRC. Flits are transparently used depending upon the physical layer (see *Physical Layer Abstraction*).

- The 16-bit CRC polynomial shall be 0xBAAD.
- The 16-bit CRC seed value shall be 0xFFFF.

- As illustrated in *16-bit Flit CRC Calculation*, the CRC calculation starts with bit 0 of byte 0 of the payload and progresses from bit 0 to bit 7 of each byte of the packet.
- The remainder of the 16-bit CRC calculation is complemented and the complemented result bits shall be mapped into the flit CRC field as illustrated in *Mapping 16-bit CRC Calculated Bits into the 16-bit CRC Field*.
- If used, Flit CRC is added at the transmitting end of a link, and removed at the receiving end. There is never a need to modify the Flit CRC due to packet field changes.

Table B-3: Mapping 16-bit CRC Calculated Bits into the 16-bit CRC Field

Result Bit	Mapped Bit Position within CRC Field	Result Bit	Mapped Bit Position within CRC Field
0	15	8	7
1	14	9	6
2	13	10	5
3	12	11	4
4	11	12	3
5	10	13	2
6	9	14	1
7	8	15	0

Polynomial 0xBAAD (Koopman notation) means:

$$g(x) = x^{16} + x^{14} + x^{13} + x^{12} + x^{10} + x^8 + x^6 + x^4 + x^3 + x + 1$$

Note that the “+ 1” is inferred.

This polynomial is sometimes represented as 0x755B (with the  $x^{16}$  inferred).

... | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | +1 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | +0 | < Byte 0

Bit Order

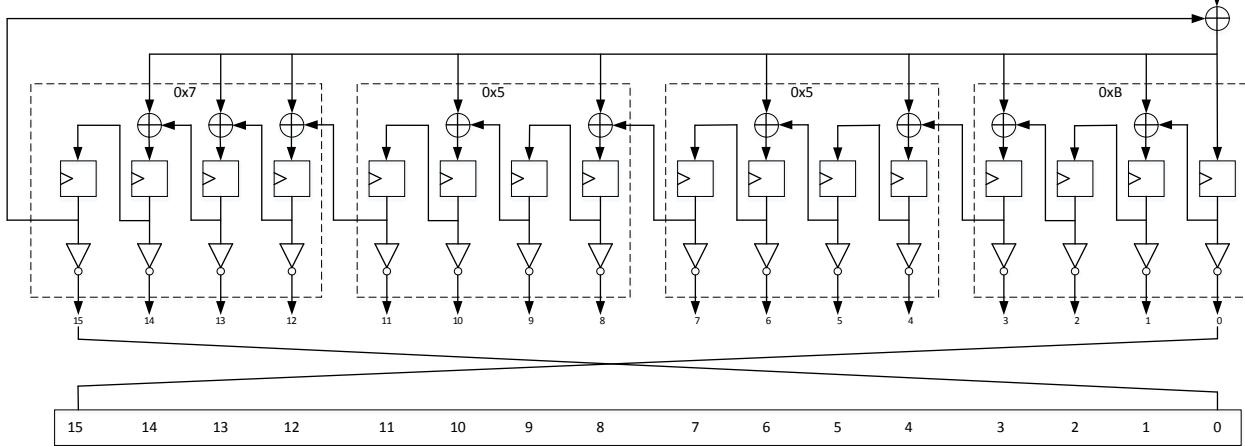


Figure B-5: 16-bit Flit CRC Calculation

# Appendix C Component Class Encodings

This appendix provides a list of Component Class encodings. These encodings are listed in *Component Class Encodings*.

The Base Component Class field in the *Core Structure* or the Class fields in the *Service UUID Structure* are used during power-on / reset initialization to determine how to initially handle a component without comprehending any component UUIDs. The following guidelines are used to select an appropriate class encoding:

- If a component supports multiple service types, then the *Core Structure* Base C-Class field shall be set to Multi-Class Component. For example, a memory module that supports load-store memory access and non-boot block storage access would set Base C-Class = Multi-Class Component, and would support the *Service UUID Structure* with two Class fields and two Service-UUID fields. One Class field would be set to Memory (P2P-Core) or Memory (Explicit OpClass), and the other Class field to I/O (Non-coherent, non-boot).
- A bootable component is one that requires management to supply an executable boot image in order to become operational, e.g., a processor that requires an operating system boot image. Where applicable, a bootable component shall use an encoding with a “bootable” attribute. Where applicable, a component that does not require management to supply an executable boot image shall use an encoding with a “non-boot” attribute.
- An accelerator is a FPGA, DSP, GPU, GPGPU, customized ASIC, etc., i.e., a component that would not be normally classified as a traditional application processor or SoC.
- A *Logical PCI Device (LPD)* shall be identified as an I/O component of the appropriate class name. Upon identification, management determines if the component supports the *Component LPD Structure*. If it does, then management enumerates the supported PCI functions and performs the necessary configuration.
- If a memory component supports the P2P-Core and Core 64 OpClasses, then the corresponding Class field shall be set to Memory (Explicit OpClass). Management uses the *OpCode Set Structure* to select the OpClass to enable based on solution topology.
- A discrete switch shall identify itself as an Enclosure Expansion switch or a Fabric switch. An Enclosure Expansion switch provides packet relay only within a single enclosure. A Fabric switch provides packet relay between enclosures or between enclosure-internal components and external enclosures / components.
- A component that contains an integrated switch shall not be identified as a switch, but by its base functionality, e.g., a Memory component or a Processor / SoC. If switch-specific software is required, then the component shall support the *Service UUID Structure* with one of the Class fields set to Integrated Switch and one of the Service-UUID fields set to the UUID used to identify the switch-specific software.
- A transparent router (TR) shall be identified as its base functionality, e.g., a TR that is attached to a Requester as a P2P-Core memory module identifies itself as a Memory (P2P-Core). If TR-specific software is required, then the component shall support the *Service UUID Structure* with one of the Class fields set to Transparent Router and one of the Service-UUID fields set to the UUID used to identify the TR-specific software.

Table C-1: Component Class Encodings

Encoded Type	Component Class Name
0x0	Reserved—shall not be used
0x1	Memory (P2P-Core)
0x2	Memory (Explicit OpClass)
0x3	Integrated Switch
0x4	Enclosure / Expansion Switch
0x5	Fabric Switch
0x6	Processor (Bootable)
0x7	Processor (Non-boot)
0x8	Accelerator (Non-coherent, non-boot)
0x9	Accelerator (Coherent, non-boot)
0xA	Accelerator (Non-coherent, bootable)
0xB	Accelerator (Coherent, bootable)
0xC	I/O (Non-coherent, non-boot)
0xD	I/O (Coherent, non-boot)
0xE	I/O (Non-coherent, bootable)
0xF	I/O (Coherent, bootable)
0x10	Block Storage (Bootable)
0x11	Block Storage (Non-boot)
0x12	Transparent Router
0x13	Multi-class Component (see <i>Service UUID Structure</i> )
0x14	Discrete Gen-Z Bridge
0x15	Integrated Gen-Z Bridge
0x16-0xFFFF	Reserved

# Acknowledgments

The following individuals contributed to the development of Gen-Z and the associated specifications:

5	Michael Krause	Mike Witkowski	Steve Chalmers	Joe Cowan
	Doug Voigt	Derek Sherlock	David Patrick	Gary Gostin
	Greg Astfalk	John Bockhaus	Alan Goodrum	Hahn Norden
	Chris Brueggen	James Regan	Nic McDonald	James Singer
	Al Davis	Darel Emmot	Gary Piccirillo	Kate Walker
10	Dong Wei	Gregg Lesartre	David Koenen	Ryan Akkerman
	Karl Bois	Pete Maroni	Ben Patella	Jim Hull
	Brian Bolich	Matt Neumann	Kevin Cash	Keith McAuliffe
	Sarah Silverthorn	Ron Noblett	Chris Barnette	Ronald Tsai
	John Norton	Abner Chang	Tom Vaden	Derek Schumacher
	Michael Ruan	Paul Kaler	Jeff Hilland	Arash Behziz
	David Herring	Nathan Tracy	Belgi McClelland	Terry Lee
15	Steve Lyle	Russ Herrell	Bill Scherer	Glenn Moore
	Alexander Jizrawi	Robert Elliott	John Mayfield	Ben Mosser
	Zhineng Fan	Greg Casey	Kevin Mundt	Kangkang Shen
	Alvin Cox	Joe Breher	Tim Symons	John Lynch
	Jon Masters	Lenny Szubowicz	Erich Hanke	Elene Chobanyan
20	Mike Tryson	Bob Frey	Arthur Sainio	Victor Mahran
	Thomas Choi	James Yu	BengSeng Phuah	Kurtis Bowman
	Sergey Blagodurov	Wendy Elsasser	Henry Hu	Barry McAuliffe
	Ben Toby	Ed Cady	Bob Frey	David Miller
	Ed Poh	David Kopp	Robert Winter	Ken Ma
25	George Scholhamer	JS Choi	Paul Hartke	Robert Hormuth
	JY Jung	Nicholas Witkowski	Hayden Lee	Mel Benedict
	Dwight Barron	Mitch Wright	William Walker	Amit Sharma
	Dan Thero	David Binford	Ericka Homer	Tim Pertuit
	Victorio Cargnini	Yogesh Varma	Don Boyd	Patrick Higgins
30	Duk Kim	Jason Jung	Clark Laughlin	Tracy Spitler
	Noah Husek	Duane Voth		