PCI-SIG ENGINEERING CHANGE NOTICE

TITLE:	Atomic Operations
DATE:	January 15, 2008; approved by PWG April 17, 2008
AFFECTED DOCUMENTS:	PCI Express Base Specification version 2.0 Errata to 2.0 Base Spec Section 2.4.1
SPONSORS:	Hewlett-Packard, Intel, Advanced Micro Devices, IBM

Part I

1. Summary of the Functional Changes

This optional normative ECN defines 3 new PCIe transactions, each of which carries out a specific Atomic Operation ("AtomicOp") on a target location in Memory Space. The 3 AtomicOps are FetchAdd (Fetch and Add), Swap (Unconditional Swap), and CAS (Compare and Swap). FetchAdd and Swap support operand sizes of 32 and 64 bits. CAS supports operand sizes of 32, 64, and 128 bits.

Endpoints and Root Ports may serve as Requesters for AtomicOps. PCIe Functions with Memory Space BARs as well as all Root Ports may serve as Completers for AtomicOp Requests. Routing elements (Switches, as well as Root Complexes supporting peer-to-peer access between Root Ports) require modification in order to route AtomicOp Requests. AtomicOps are architected for device-to-host, device-to-device, and host-to-device transactions.

Four new bits in the Device Capabilities 2 register enable software to discover specific AtomicOp Completer capabilities in arbitrary PCIe Functions, and AtomicOp routing capability in routing elements. Software discovery of AtomicOp Requester capabilities is not architected, but a new AtomicOp Requester Enable bit in the Device Control 2 register must be set by software in order for a Function to initiate AtomicOp Requests. Software can set a new AtomicOp Egress Blocking bit in routing element Ports as needed to avoid forwarding ("block") AtomicOp Requests to components that shouldn't receive them.

2. Benefits as a Result of the Changes

AtomicOps enable advanced synchronization mechanisms that are particularly useful when there are multiple producers and/or multiple consumers that need to be synchronized in a non-blocking fashion. AtomicOps also enable lock-free statistics counters, for example where a device atomically increments a counter, and host software atomically reads and clears the counter. Compared to Locked Transactions, AtomicOps provide lower latency, higher scalability, advanced synchronization algorithms, and dramatically less impact to other PCIe traffic.

Direct support for the 3 chosen AtomicOps over PCIe enables easier migration of existing high-performance SMP applications to systems that use PCIe as the interconnect to tightly-coupled accelerators, co-processors, or GP-GPUs.

While applicable across many market segments, AtomicOps are envisioned to enable price/performance breakthroughs for systems targeting HPTC, financial services, business intelligence, telecommunications, real-time data stream processing, and high-performance graphics.

3. Assessment of the Impact

AtomicOps support is optional normative, and is applicable to Root Complexes, Switches, and components with Endpoint Functions. AtomicOps routing support is not applicable to PCIe to PCI/PCI-X Bridges. AtomicOps are architected for device-to-host, device-to-device, and host-to-device transactions. In each case, the Requester, Completer, and all intermediate routing elements must support the associated AtomicOp capabilities.

4. Analysis of the Hardware Implications

Components that implement AtomicOp Requester capability may generate any or all of the 3 new Non-Posted Request types, each supporting one or more operand sizes.

Components that implement AtomicOp Completer capability carry out the new transactions. A Function with AtomicOp Completer capability is not required to support all AtomicOp type/size combinations, and is not required to support all of its Memory Space as a target range for AtomicOps. For increased interoperability, certain AtomicOp Completer capabilities are required to be supported by Root Ports in sets if at all.

Components with routing elements that support AtomicOp Request routing recognize the new Requests, and route them the same as all other Memory Requests. AtomicOp Egress Blocking is required to be supported by all routing elements that have AtomicOp routing capability, and the associated error is managed/reported by new AtomicOp Egress Blocked Status/Mask/Severity bits in AER. A Switch with AtomicOp routing capability is required to support it between all of its Ports. An RC with AtomicOp routing capability is not required to support it on all of its Root Ports. Maximum Non-Posted Data (NPD) flow control credit requirements for AtomicOp Requests vs other Non-Posted Requests is increased from 1 unit to 2. No changes are required for AtomicOp Completion routing.

Ordering requirements for AtomicOp Requests are identical to those for Non-Posted Write Requests, with the exception that the Relaxed Ordering attribute is also permitted. Ordering requirements for AtomicOp Completions are similar to Memory Read Completions.

5. Analysis of the Software Implications

Config Space structure enhancements mentioned earlier enable software to discover the specific AtomicOp Completer and routing capabilities. Software discovery of AtomicOp Requester capability is outside the scope of this specification, but a Function is not permitted to issue AtomicOp Requests unless software sets its AtomicOp Requester Enable bit. AtomicOps are Memory Transactions, so existing standard mechanisms for managing Memory Space access like Bus Master Enable, Memory Space Enable, and Base Address registers apply.

Part II

Detailed Description of the change

Modify Terms and Acronyms as shown

AtomicOp	One of 3 architected atomic operations, where a single PCI Express transaction targeting a location in Memory Space reads the location's value, potentially writes a new value to the location, and returns the original value. This read-modify-write sequence to the location is performed atomically. AtomicOps include FetchAdd, Swap, and CAS.
CAS	Compare and Swap. An AtomicOp where the value of a target location is compared to a specified value, and if they match, another specified value is written back to the location. Regardless, the original value of the location is returned.
FetchAdd	Fetch and Add. An AtomicOp where the value of a target location is incremented by a specified value using two's complement arithmetic ignoring any carry or overflow, and the result is written back to the location. The original value of the location is returned.
Swap	Unconditional Swap. An AtomicOp where a specified value is written to a target location, and the original value of the location is returned.

Modify Section 2.1.1.1. as shown

2.1.1.1. Memory Transactions

Memory Transactions include the following types:

- Read Request/Completion
- Write Request
- AtomicOp Request/Completion

Modify Section 2.2.1. as shown

2.2.1. Common Packet Header Fields

…

- Permitted Fmt[1:0] and Type[4:0] field values are shown in Table 2-3.

…

Table 2-3: Fmt[1:0] and Type[4:0] Field Encodings

TLP Type	Fmt [1:0][1] (b)	Type [4:0] (b)	Description
…	…	…	…
Cpl	00	0 1010	Completion without Data – Used for I/O and Configuration Write Completions with any Completion Status. Also used for AtomicOp Completions and Read Completions (I/O, Configuration, or Memory) with Completion Status other than Successful Completion.
CplD	10	0 1010	Completion with Data – Used for Memory, I/O, and Configuration Read Completions. Also used for AtomicOp Completions.
…	…	…	…
CplDLk	10	0 1011	Completion for Locked Memory Read – otherwise like CplD.
FetchAdd	10 11	0 1100	Fetch and Add AtomicOp Request
Swap	10 11	0 1101	Unconditional Swap AtomicOp Request
CAS	10 11	0 1110	Compare and Swap AtomicOp Request
			All encodings not shown above are Reserved.

[1] Requests with two Fmt[1:0] values shown can use either 32 bits (the first value) or 64 bits (the second value) Addressing Packet formats.

Modify Section 2.2.2. as shown

2.2.2. TLPs with Data Payloads – Rules

...

- When a data payload is included in a TLP other than an AtomicOp Request or an AtomicOp Completion, the first byte of data following the header corresponds to the byte address closest to zero and the succeeding bytes are in increasing byte address sequence.
  - Example: For a 16-byte write to location 100h, the first byte following the header would be the byte to be written to location 100h, and the second byte would be written to location 101h, and so on, with the final byte written to location 10Fh.

- The data payload in AtomicOp Requests and AtomicOp Completions must be formatted such that the first byte of data following the TLP header is the least significant byte of the first data value, and subsequent bytes of data are strictly increasing in significance. With CAS Requests, the second data value immediately follows the first data value, and must be in the same format.
  - The endian format used by AtomicOp Completers to read and write data at the target location is implementation specific, and is permitted to be whatever the Completer determines is appropriate for the target memory (e.g., little endian, big endian, etc). Endian format capability reporting and controls for AtomicOp Completers are outside the scope of this specification.
  - Little endian example: For a 64-bit (8-byte) Swap Request targeting location 100h with the target memory in little endian format, the first byte following the header is written to location 100h, the second byte is written to location 101h, and so on, with the final byte written to location 107h. Note that before performing the writes, the Completer first reads the target memory locations so it can return the original value in the Completion. The byte address correspondence to the data in the Completion is identical to that in the Request.
  - Big endian example: For a 64-bit (8-byte) Swap Request targeting location 100h with the target memory in big endian format, the first byte following the header is written to location 107h, the second byte is written to location 106h, and so on, with the final byte written to location 100h. Note that before performing the writes, the Completer first reads the target memory locations so it can return the original value in the Completion. The byte address correspondence to the data in the Completion is identical to that in the Request.
  - Figure 2-xx shows little endian and big endian examples of Completer target memory access for a 64-bit (8-byte) FetchAdd. The bytes in the operands and results are numbered 0-7, with byte 0 being least significant and byte 7 being most significant. In each case, the Completer fetches the target memory operand using the appropriate endian format. Next, AtomicOp compute logic in the Completer performs the FetchAdd operation using the original target memory value and the "add" value from the FetchAdd Request. Finally, the Completer stores the FetchAdd result back to target memory using the same endian format used for the fetch.

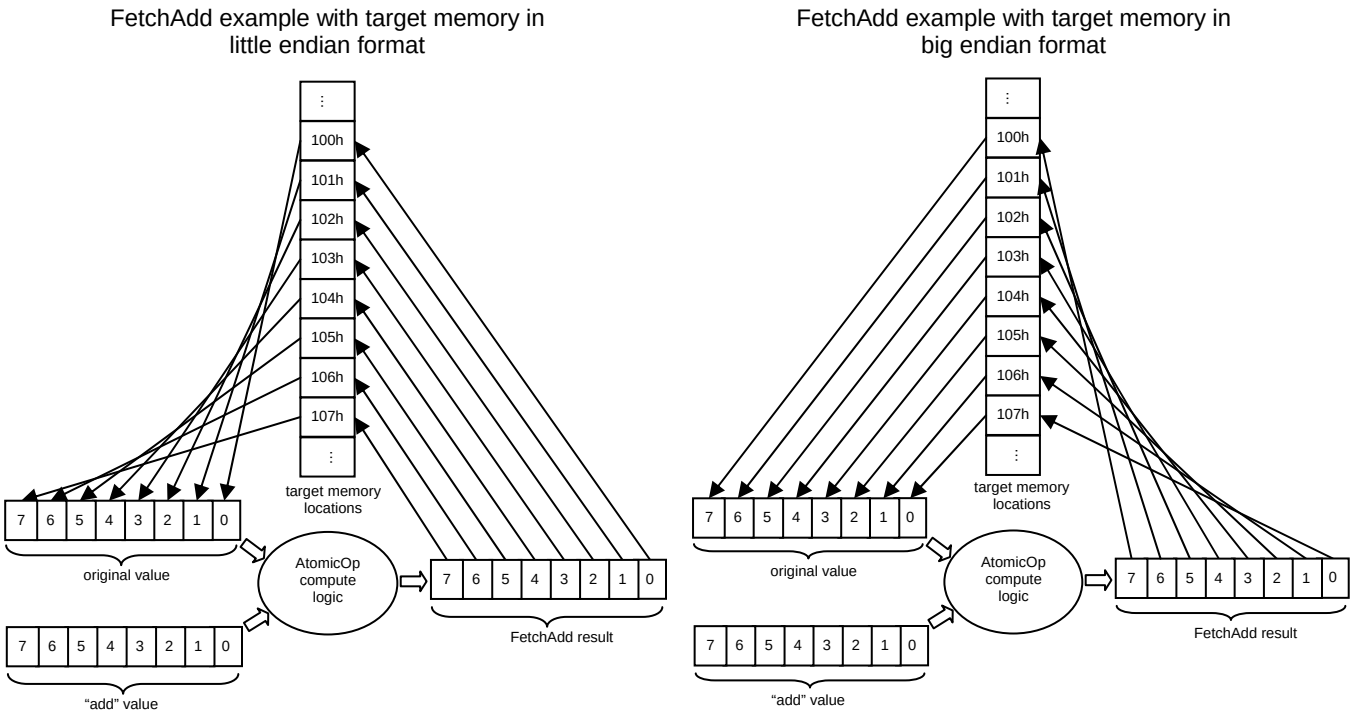

Figure 2-xx: Examples of Completer Target Memory Access for FetchAdd

IMPLEMENTATION NOTE

Endian Format Support by RC AtomicOp Completers

One key reason for permitting an AtomicOp Completer to access target memory using an endian format of its choice is so that PCIe devices targeting host memory with AtomicOps can interoperate with host software that uses atomic operation instructions (or instruction sequences). Some host environments have limited endian format support with atomic operations, and by supporting the "right" endian format(s), an RC AtomicOp Completer may significantly improve interoperability.

For an RC with AtomicOp Completer capability on a platform supporting little-endian-only processors, there is little envisioned benefit for the RC AtomicOp Completer to support any endian format other than little endian. For an RC with AtomicOp Completer capability on a platform supporting bi-endian processors, there may be benefit in supporting both big endian and little endian formats, and perhaps having the endian format configurable for different regions of host memory.

There is no PCIe requirement that an RC AtomicOp Completer support the host processor's "native" format (if there is one), nor is there necessarily significant benefit to doing so. For example, some processors can use load-link/store-conditional or similar instruction sequences to do atomic operations in non-native endian formats and thus not need the RC AtomicOp Completer to support alternative endian formats.

Modify Section 2.2.4.1. as shown

2.2.4.1. Address Based Routing Rules

…

- Two address formats are specified, a 64-bit format used with a 4 DW header (see Figure 2-5) and a 32-bit format used with a 3 DW header (see Figure 2-6).

…

- For Memory Read, ~~Requests and~~ Memory Write, and AtomicOp Requests, the Address Type (AT) field is encoded as shown in Table 2-5, with full descriptions contained in the *Address Translation Services Specification, Revision 1.0*. For all other Requests, the AT field is reserved.

…

- Memory Read, ~~Requests and~~ Memory Write, and AtomicOp Requests can use either format.

Modify Section 2.2.7. as shown

2.2.7. Memory, I/O, and Configuration Request Rules

The following rule applies to all Memory, I/O, and Configuration Requests. Additional rules specific to each type of Request follow.

- All Memory, I/O, and Configuration Requests include the following fields in addition to the common header fields:
  - Requester ID[15:0] and Tag[7:0], forming the Transaction ID
  - Last DW BE[3:0] and 1st DW BE[3:0]. For AtomicOp Requests, the DW BE fields are reserved.

For Memory Requests, the following rules apply:

- Memory Requests route by address, using either 64-bit or 32-bit Addressing (see Figure 2-13 and Figure 2-14)
- For Memory Read Requests, Length must not exceed the value specified by Max_Read_Request_Size (see Section 7.8.4)
- For AtomicOp Requests, architected operand sizes and their associated Length field values are specified in Table 2-xx. The Completer must check the Length field value. If the value doesn't match an architected value, the Completer must handle the TLP as a Malformed TLP. Otherwise, if the value doesn't match an operand size that the Completer supports, the Completer must handle the TLP as an Unsupported Request (UR). This is a reported error associated with the Receiving Port (see Section 6.2).

Table 2-xx: Length Field Values for AtomicOp Requests

AtomicOp Request	Length Field Value for Architected Operand Sizes		
	32 bits	64 bits	128 bits
FetchAdd, Swap	1 DW	2 DW	N/A
CAS	2 DW	4 DW	8 DW

- A FetchAdd Request contains one operand, the "add" value.
- A Swap Request contains one operand, the "swap" value.
- A CAS Request contains two operands. The first in the data area is the "compare" value, and the second is the "swap" value.

❑ For AtomicOp Requests, the Address must be naturally aligned with the operand size. The Completer must check for violations of this rule. If a TLP violates this rule, the TLP is a Malformed TLP. This is a reported error associated with the Receiving Port (see Section 6.2).

Modify Section 2.2.9. as shown

2.2.9. Completion Rules

All Read, ~~Requests and~~ Non-Posted Write, and AtomicOp Requests require Completion. Completions include a Completion header that, for some types of Completions, will be followed by some number of DW of data. The rules for each of the fields of the Completion header are defined in the following sections.

…

❑ In addition to the header fields included in all TLPs and the ID routing fields, Completions contain the following additional fields (see Figure 2-19)

…

- Byte Count[11:0] – The remaining byte count for Request
  - ♦ The Byte Count value is specified as a binary number, with 0000 0000 0001b indicating 1 byte, 1111 1111 1111b indicating 4095 bytes, and 0000 0000 0000b indicating 4096 bytes
  - ♦ For Memory Read Completions, Byte Count[11:0] is set according to the rules in Section 2.3.1.1.
  - ♦ For AtomicOp Completions, the Byte Count value must equal the associated AtomicOp operand size in bytes.
  - ♦ For all other types of Completions, the Byte Count field must be 4.

…

- Lower Address[6:0] – lower byte address for starting byte of Completion
  - For Memory Read Completions, the value in this field is the byte address for the first enabled byte of data returned with the Completion (see the rules in Section 2.3.1.1)
  - For AtomicOp Completions, the Lower Address field is reserved.
  - This field is set to all 0's for all remaining types of Completions ~~other than Memory Read Completions~~.

Modify Section 2.3.2 as shown

2.3.2. Completion Handling Rules

…

- ❑ When a Read Completion or an AtomicOp Completion is received with a Completion Status other than Successful Completion:
  - No data is included with the Completion
    - The Cpl (or CplLk) encoding is used instead of CplD (CplDLk)

Modify Section 2.4.1. as shown. Note: the Base Specification text shown here in Section 2.4.1 already incorporates major errata, and indicated changes are relative to the errata text.

2.4.1. Transaction Ordering Rules

…

Table 2-24: Ordering Rules Summary Table

Row Pass Column?		Posted Request (Col 2)	Non-Posted Request		Completion (Col 5)
			Read Request (Col 3)	NPR with Data (Col 4)	
Posted Request (Row A)		a) No b) Y/N	Yes	Yes	a) Y/N b) Yes
Non-Posted Request	**Read Request** (Row B)	No	Y/N	Y/N	Y/N
	NPR with Data (Row C)	a) No b) Y/N	Y/N	Y/N	Y/N
Completion (Row D)		a) No b) Y/N	Yes	Yes	a) Y/N b) No

Explanation of the row and column headers in Table 2-24:

- A **Posted Request** is a Memory Write Request or a Message Request.
- A **Read Request** is a Configuration Read Request, an I/O Read Request, or a Memory Read Request.
- An **NPR** (Non-Posted Request) **with Data** is a Configuration Write Request ~~or,~~ an I/O Write Request, or an AtomicOp Request.
- A **Non-Posted Request** is a Read Request or an NPR with Data.

Explanation of the entries in Table 2-24:

A2a	A Posted Request must not pass another Posted Request unless A2b applies.
A2b	A Posted Request with RO[2] Set is permitted to pass another Posted Request.
A3, A4	A Posted Request must be able to pass Non-Posted Requests to avoid deadlocks.
A5a	A Posted Request is permitted to pass a Completion, but is not required to be able to pass Completions unless A5b applies.
A5b	Inside a PCI Express to PCI/PCI-X Bridge whose PCI/PCI-X bus segment is operating in conventional PCI mode, for transactions traveling in the PCI Express to PCI direction, a Posted Request must be able to pass Completions to avoid deadlock.
B2	A Read Request must not pass a Posted Request.
C2a	An NPR with Data must not pass a Posted Request unless C2b applies.
C2b	An NPR with Data and with RO Set[3] is permitted to pass Posted Requests.
B3, B4, C3, C4	A Non-Posted Request is permitted to pass another Non-Posted Request.
B5, C5	A Non-Posted Request is permitted to pass a Completion.
D2a	A Completion must not pass a Posted Request unless D2b applies.
D2b	An I/O or Configuration Write Completion[4] is permitted to pass a Posted Request. A Completion with RO Set is permitted to pass a Posted Request.
D3, D4	A Completion must be able to pass Non-Posted Requests to avoid deadlocks.
D5a	Completions with different Transaction IDs are permitted to pass each other.
D5b	Completions with the same Transaction ID must not pass each other. This ensures that multiple Completions associated with a single Memory Read Request will remain in ascending address order.

---

[2] In this section, "RO" is an abbreviation for the Relaxed Ordering Attribute field.

[3] Note: Not all NPR with Data transactions are permitted to have RO Set.

[4] Note: Not all components can distinguish I/O and Configuration Write Completions from other Completions. In particular, routing elements not serving as the associated Requester or Completer generally can't make this distinction. A component must not apply this rule for I/O and Configuration Write Completions unless it is certain of the associated Request type.

Modify Section 2.6.1. as shown

2.6.1. Flow Control Rules

…

- Flow Control distinguishes three types of TLPs (note relationship to ordering rules – see Section 2.4):
  - Posted Requests (P) – Messages and Memory Writes
  - Non-Posted Requests (NP) – All Reads, I/O Writes, ~~and~~ Configuration Writes, and AtomicOps
  - Completions (CPL) – Associated with corresponding NP Requests

…

- TLPs consume Flow Control credits as shown in Table 2-27.

Table 2-27: TLP Flow Control Credit Consumption

TLP	Credit Consumed[5]
Memory, I/O, Configuration Read Request	1 NPH unit
Memory Write Request	1 PH + n PD units[6]
I/O, Configuration Write Request	1 NPH + 1 NPD Note: size of data written is never more than 1 (aligned) DW
AtomicOp Request	1 NPH + n NPD units
Message Requests without data	1 PH unit
Message Requests with data	1 PH + n PD units
Memory Read Completion	1 CPLH + n CPLD units
I/O, Configuration Read Completions	1 CPLH unit + 1 CPLD unit
I/O, Configuration Write Completions	1 CPLH unit
AtomicOp Completion	1 CPLH unit + 1 CPLD unit Note: size of data returned is never more than 4 (aligned) DWs

…

[5] Each header credit implies the ability to accept a TLP Digest along with the corresponding TLP.

[6] For all cases where "n" appears, n = Roundup(Length/FC unit size).

- ❑ During FC initialization for any Virtual Channel, including the default VC initialized as a part of Link initialization, Receivers must initially advertise VC credit values equal to or greater than those shown in Table 2-28.

Table 2-28: Minimum Initial Flow Control Advertisements[7]

Credit Type	Minimum Advertisement
PH	1 unit – credit value of 01h
PD	Largest possible setting of the Max_Payload_Size for the component divided by FC Unit Size. For a multi-Function device, this includes all Functions in the device. Example: If the largest Max_Payload_Size value supported is 1024 bytes, the smallest permitted initial credit value would be 040h.
NPH	1 unit – credit value of 01h
NPD	Receiver that supports AtomicOp routing capability or any AtomicOp Completer capability: 2 units – credit value of 02h All other Receivers: 1 unit – credit value of 01h
…	…

Modify Section 2.6.1.2. as shown

2.6.1.2. FC Information Tracked by Receiver

…

- ❑ For non-infinite NPH, NPD, PH, and CPLH types, an UpdateFC FCP must be scheduled for Transmission each time the following sequence of events occurs:
  - (a) all advertised FC units for a particular type of credit are consumed by TLPs received, or (b) the NPD credit drops below 2 and the Receiver supports AtomicOp routing capability or 128b CAS Completer capability
  - one or more units of that type are made available by TLPs processed

Modify Section 2.7.2.1. as shown

2.7.2.1. Error Forwarding Usage Model

- ❑ Error Forwarding is only used for Read Completion Data, AtomicOp Completion Data, AtomicOp Request Data, or Write Data, never for the cases when the error is in the "header" (request phase, address/command, etc.). Requests/Completions with header errors cannot be forwarded in general since true destination cannot be positively known and, therefore, forwarding may cause direct or side effects such as data corruption, system failures, etc.

[7] PCI Express to PCI/PCI-X Bridge requirements are addressed in the *PCI Express to PCI/PCI-X Bridge Specification, Revision 1.0*.

Modify Section 2.7.2.2. as shown

2.7.2.2. Rules For Use of Data Poisoning

...

- ❑ Data poisoning applies only to the data within a Write Request (Posted or Non-Posted), an AtomicOp Request, ~~or~~ a Read Completion, or an AtomicOp Completion.
  - Poisoning of a TLP is indicated by a 1b value in the EP field
  - Transmitters are permitted to set the EP field to 1b only for TLPs that include a data payload. The behavior of the receiver is not specified if the EP bit is set for any TLP that does not include a data payload.

...

- ❑ A Poisoned Configuration Write Request must be discarded by the Completer, and a Completion with a Completion Status of UR is returned (see Section 2.2.9).
  - A Switch must route a Poisoned Configuration Write Request in the same way it would route a non-Poisoned Request, unless the Request addresses the Configuration Space of the Switch itself, in which case the Switch is the Completer for the Request and must follow the above rule.
- ❑ A Poisoned I/O or Memory Write Request, or a Message with data (except for vendor-defined Messages), that addresses a control register or control structure in the Completer must be handled as an Unsupported Request (UR) by the Completer (see Section 2.2.9).
  - A Switch must route a Poisoned I/O or Memory Write Request or Message with data in the same way it would route the same Request if it were not poisoned, unless the Request addresses a control register or control structure of the Switch itself, in which case the Switch is the Completer for the Request and must follow the above rule.
- ❑ Unless there's a higher precedence error, a Completer must handle a Poisoned AtomicOp Request as a Poisoned TLP Received error, and must also return a Completion with a Completion Status of Unsupported Request (UR). See Sections 6.2.3.2.3, 6.2.3.2.4, and 6.2.5. The value of the target location must remain unchanged.
  - A Switch must route a Poisoned AtomicOp Request the same way it would route the same Request if it were not poisoned, unless the Request targets a Memory Space location in the Switch itself, in which case the Switch is the Completer for the Request and must follow the above rule.

Modify footnote in Section 6.2.3.2.4.1. as shown

6.2.3.2.4.1. Completer Sending a Completion with UR/CA Status

A Completer generally sends a Completion with an Unsupported Request or Completer Abort (UR/CA) Status to signal a uncorrectable error for a Non-Posted Request.[8] If the severity of the UR/CA error[9] is non-fatal, the Completer must handle this case as an Advisory Non-Fatal Error.[10] A Completer with AER signals the non-fatal error (if enabled) by sending an ERR_COR Message. A Completer without AER sends no Error Message for this case.

Modify Section 6.2.3.2.3. as shown. Note: this section already incorporates Base Specification errata adding ACS Violation to the precedence list.

6.2.3.2.3. Error Pollution

…

For errors detected in the Transaction layer, it is permitted and recommended that no more than one error be reported for a single received TLP, and that the following precedence (from highest to lowest) be used:

- Receiver Overflow
- Flow Control Protocol Error
- ECRC Check Failed
- Malformed TLP
- AtomicOp Egress Blocked
- ACS Violation
- Unsupported Request (UR), Completer Abort (CA), or Unexpected Completion[11]
- Poisoned TLP Received

---

[8] If the Completer is returning data in a Completion, and the data is bad or suspect, the Completer is permitted to signal the error using the Error Forwarding (Data Poisoning) mechanism instead of handling it as a UR or CA.

[9] ~~An ACS Violation error~~ Certain other errors (e.g., ACS Violation) with a Non-Posted Request also result~~s~~ in the Completer sending a Completion with UR or CA Status. If the severity of the ~~ACS Violation~~ error (e.g., ACS Violation) is non-fatal, the Completer must also handle this case as an Advisory Non-Fatal Error.

[10] If the severity is fatal, the error is not an Advisory Non-Fatal Error, and must be signaled (if enabled) with ERR_FATAL.

[11] These are mutually exclusive errors, so their relative order does not matter.

Modify Section 6.2.7. as shown

6.2.7. Error Listing and Rules

…

Table 6-4 Transaction Layer Error List

Error Name	Severity	Detecting Agent Action	References
…	Uncorrectable (Non-Fatal)	…	
AtomicOp Egress Blocked		*Egress Port*: Send ERR_COR to Root Complex. This is an Advisory Non-Fatal Error case described in Section 6.2.3.2.4.1. Log the header of the AtomicOp Request that encountered the error.	Section 6.xx.2

Modify Section 6.12.1.1. as shown

6.12.1.1. ACS Downstream Ports

…

- ACS P2P Completion Redirect: must be implemented by Root Ports that implement ACS P2P Request Redirect; must be implemented by Switch Downstream Ports.

  The intent of ACS P2P Completion Redirect is to avoid ordering rule violations between Completions and Requests when Requests are redirected. Refer to Section 6.12.5 for more information.

  ACS P2P Completion Redirect does not interact with ACS controls that govern Requests.

  When ACS P2P Completion Redirect is enabled in a Switch Downstream Port, peer-to-peer ~~Read~~ Completions[12] that do not have the Relaxed Ordering Attribute bit set (1b) must be redirected Upstream towards the RC. Otherwise, peer-to-peer Completions must be routed normally.

  When ACS P2P Completion Redirect is enabled in a Root Port, peer-to-peer ~~Read~~ Completions that do not have the Relaxed Ordering bit set must be handled such that they do not pass Requests that are sent to Redirected Request Validation logic within the RC. Such Completions must eventually be sent Downstream towards their original peer-to-peer targets, without incurring additional ACS access control checks.

  Downstream Ports never redirect Completions that are traveling Downstream.

  Requests are never affected by ACS P2P Completion Redirect.

[12] This includes Read Completions, AtomicOp Completions, and other Completions, either with or without Data.

Modify Section 6.12.5.1. as shown

6.12.5.1. Completions Passing Posted Requests

When a peer-to-peer Posted Request is redirected, a subsequent peer-to-peer non-RO[13] ~~Read~~ Completion that is routed directly can effectively pass the redirected Posted Request, violating the ordering rule that non-RO ~~Read~~ Completions must not pass Posted Requests. Refer to Section 2.4.1 for more information.

ACS P2P Completion Redirect can be used to avoid violating this ordering rule. When ACS P2P Completion Redirect is enabled, all peer-to-peer non-RO ~~Read~~ Completions will be redirected, thus taking the same path as redirected peer-to-peer Posted Requests. Enabling ACS P2P Completion Redirect when some or all peer-to-peer Requests are routed directly will not cause any ordering rule violations, since it is permitted for a given Completion to be passed by any TLP other than another Completion with the same Transaction ID.

As an alternative mechanism to ACS P2P Request Redirect for enforcing peer-to-peer access control, some RCs implement "Request Retargeting", where the RC supports special address ranges for "peer-to-peer" traffic, and the RC will retarget validated Upstream Requests to peer devices. Upon receiving an Upstream Request targeting a special address range, the RC validates the Request, translates the address to target the appropriate peer device, and sends the Request back Downstream. With retargeted Requests that are Non-posted, if the RC does not modify the Requester ID, the resulting Completions will travel "directly" peer-to-peer back to the original Requester, creating the possibility of non-RO ~~Read~~ Completions effectively passing retargeted Posted Requests, violating the same ordering rule as when ACS P2P Request Redirect is being used. ACS P2P Completion Redirect can be used to avoid violating this ordering rule here as well.

If ACS P2P Request Redirect and RC P2P Request Retargeting are not being used, there's no envisioned benefit to enabling ACS P2P Completion Redirect, and it is recommended not to do so because of potential performance impacts.

IMPLEMENTATION NOTE

Performance Impacts with ACS P2P Completion Redirect

While the use of ACS P2P Completion Redirect can avoid ordering violations with Completions passing Posted Requests, it also may impact performance. Specifically, all redirected Completions will have to travel up to the RC from the point of redirection and back, introducing extra latency and possibly increasing Link and RC congestion.

Since peer-to-peer ~~Read~~ Completions with the Relaxed Ordering bit set are never redirected (thus avoiding performance impacts), it is strongly recommended that Requesters be implemented to maximize the proper use of Relaxed Ordering, and that software enable Requesters to utilize Relaxed Ordering by setting the Enable Relaxed Ordering bit in the Device Control register.

---

[13] In this section, "non-RO" is an abbreviation characterizing TLPs whose Relaxed Ordering Attribute field is not set.

If software enables ACS P2P Request Redirect, RC P2P Request Retargeting, or both, and software is certain that proper operation is not compromised by peer-to-peer non-RO ~~Read~~ Completions passing peer-to-peer[14] Posted Requests, it is recommended that software leave ACS P2P Completion Redirect disabled as a way to avoid its performance impacts.

Add new Section 6.xx.

6.xx. Atomic Operations (AtomicOps)

An Atomic Operation (AtomicOp) is a single PCI Express transaction that targets a location in Memory Space, reads the location's value, potentially writes a new value back to the location, and returns the original value. This "read-modify-write" sequence to the location is performed atomically. AtomicOps include the following:

- ❑ FetchAdd (Fetch and Add): Request contains a single operand, the "add" value
  - Read the value of the target location.
  - Add the "add" value to it using two's complement arithmetic ignoring any carry or overflow.
  - Write the sum back to the target location.
  - Return the original value of the target location.
- ❑ Swap (Unconditional Swap): Request contains a single operand, the "swap" value
  - Read the value of the target location.
  - Write the "swap" value back to the target location.
  - Return the original value of the target location.
- ❑ CAS (Compare and Swap): Request contains 2 operands, a "compare" value and a "swap" value
  - Read the value of the target location.
  - Compare that value to the "compare" value.
  - If equal, write the "swap" value back to the target location.
  - Return the original value of the target location.

A given AtomicOp transaction has an associated operand size, and the same size is used for the target location accesses and the returned value. FetchAdd and Swap support operand sizes of 32 and 64 bits. CAS supports operand sizes of 32, 64, and 128 bits.

AtomicOp capabilities are optional normative. Endpoints and Root Ports are permitted to implement AtomicOp Requester capabilities. PCIe Functions with Memory Space BARs as well as all Root Ports are permitted to implement AtomicOp Completer capabilities. Routing elements (Switches, as well as Root Complexes supporting peer-to-peer access between Root Ports) require

---

[14] These include true peer-to-peer Requests that are redirected by the ACS P2P Request Redirect mechanism, as well as "logically peer-to-peer" Requests routed to the Root Complex that the Root Complex then retargets to the peer device.

AtomicOp routing capability in order to route AtomicOp Requests. AtomicOps are architected for device-to-host, device-to-device, and host-to-device transactions. In each case, the Requester, Completer, and all intermediate routing elements must support the associated AtomicOp capabilities.

AtomicOp capabilities are not supported on PCI Express to PCI/PCI-X Bridges. If need be, Locked Transactions can be used for devices below such Bridges. AtomicOps and Locked Transactions can operate concurrently on the same hierarchy.

Software discovers specific AtomicOp Completer capabilities via 3 new bits in the Device Capabilities 2 register (see Section 7.8.15). For increased interoperability, Root Ports are required to implement certain AtomicOp Completer capabilities in sets if at all (see Section 6.xx.3.1). Software discovers AtomicOp routing capability via the AtomicOp Routing Supported bit in the Device Capabilities 2 register. Software discovery of AtomicOp Requester capabilities is outside the scope of this specification, but software must set the AtomicOp Requester Enable bit in a Function's Device Control 2 register before the Function can initiate AtomicOp Requests (see Section 7.8.16).

With routing elements, software can set an AtomicOp Egress Blocking bit (see Section 7.8.16) on a Port-by-Port basis to avoid AtomicOp Requests being forwarded to components that shouldn't receive them, and might handle each as a Malformed TLP, which by default is a Fatal Error. Each blocked Request is handled as an AtomicOp Egress Blocked error, which by default is an Advisory Non-Fatal Error.

AtomicOps are Memory Transactions, so existing standard mechanisms for managing Memory Space access like Bus Master Enable, Memory Space Enable, and Base Address registers apply.

6.xx.1. AtomicOp Use Models and Benefits

AtomicOps enable advanced synchronization mechanisms that are particularly useful when there are multiple producers and/or multiple consumers that need to be synchronized in a non-blocking fashion. For example, multiple producers can safely enqueue to a common queue without any explicit locking.

AtomicOps also enable lock-free statistics counters, for example where a device can atomically increment a counter, and host software can atomically read and clear the counter.

Direct support for the 3 chosen AtomicOps over PCIe enables easier migration of existing high-performance SMP applications to systems that use PCIe as the interconnect to tightly-coupled accelerators, co-processors, or GP-GPUs. For example, a ported application that uses PCIe-attached accelerators may be able to use the same synchronization algorithms and data structures as the earlier SMP application.

An AtomicOp to a given target generally incurs latency comparable to a Memory Read to the same target. Within a single hierarchy, multiple AtomicOps can be "in flight" concurrently. AtomicOps generally create negligible disruption to other PCIe traffic.

Compared to Locked Transactions, AtomicOps provide lower latency, higher scalability, advanced synchronization algorithms, and dramatically less impact to other PCIe traffic.

6.xx.2. AtomicOp Transaction Protocol Summary

Detailed protocol rules and requirements for AtomicOps are distributed throughout the rest of this specification, but here is a brief summary plus some unique requirements.

- ❑ AtomicOps are Non-Posted Memory Transactions, supporting 32- and 64-bit address formats.
- ❑ FetchAdd, Swap, and CAS each use a distinct type code.
- ❑ The Completer infers the operand size from the Length field value and type code in the AtomicOp Request.
- ❑ The endian format used by AtomicOp Completers to read and write data at the target location is implementation specific, and permitted to be whatever the Completer determines to be appropriate for the target memory (e.g. little-endian, big-endian, etc). See Section 2.2.2.
- ❑ If an AtomicOp Requester supports Address Translation Services (ATS), the Requester is permitted to use a Translated address in an AtomicOp Request only if the Translated address has appropriate access permissions. Specifically, the Read (R) and Write (W) fields must both be Set, and the Untranslated access only (U) field must be Clear. See Section 2.2.4.1.
- ❑ If a component supporting Access Control Services (ACS) supports AtomicOp routing or AtomicOp Requester capability, it handles AtomicOp Requests and Completions the same as with other Memory Requests and Completions with respect to ACS functionality.
- ❑ The No Snoop attribute is applicable and permitted to be Set with AtomicOp Requests, but atomicity must be guaranteed regardless of the No Snoop attribute value.
- ❑ The Relaxed Ordering attribute is applicable and permitted to be Set with AtomicOp Requests, where it affects the ordering of both the Requests and their associated Completions.
- ❑ Ordering requirements for AtomicOp Requests are similar to those for Non-Posted Write Requests. Thus, if a Requester wants to ensure that an AtomicOp Request is observed by the Completer before a subsequent Posted or Non-Posted Request, the Requester must wait for the AtomicOp Completion before issuing the subsequent Request.
- ❑ Ordering requirements for AtomicOp Completions are similar to those for Read Completions.
- ❑ Unless there's a higher precedence error, a Completer must handle a Poisoned AtomicOp Request as a Poisoned TLP Received error, and must also return a Completion with a Completion Status of Unsupported Request (UR). See Section 2.7.2.2. The value of the target location must remain unchanged.
- ❑ If the Completer of an AtomicOp Request encounters an uncorrectable error accessing the target location or carrying out the Atomic operation, the Completer must handle it as a Completer Abort (CA). The subsequent state of the target location is implementation specific.
- ❑ Completers are required to handle any properly formed AtomicOp Requests with types or operand sizes they don't support as an Unsupported Request (UR). If the Length field in an AtomicOp Request contains an unarchitected value, the Request must be handled as a Malformed TLP. See Section 2.2.7.
- ❑ If any Function in a multi-Function device supports AtomicOp Completer or AtomicOp routing capability, all Functions with Memory Space BARs in that device must decode properly formed AtomicOp Requests and handle any they don't support as an Unsupported Request (UR). Note

that in such devices, Functions lacking AtomicOp Completer capability are forbidden to handle properly formed AtomicOp Requests as Malformed TLPs.

- ❑ If an RC has any Root Ports that support AtomicOp routing capability, all Root Complex Integrated Endpoints in the RC reachable by forwarded AtomicOp Requests must decode properly formed AtomicOp Requests and handle any they don't support as an Unsupported Request (UR).
- ❑ With an AtomicOp Request having a supported type and operand size, the Completer is required either to carry out the Request or handle it as Completer Abort (CA) for any location in its target Memory Space. Completers are permitted to support AtomicOp Requests on a subset of their target Memory Space as needed by their programming model (see Section 2.3.1). Memory Space structures defined or inherited by PCI Express (e.g., the MSI-X Table structure) are not required to be supported as AtomicOp targets unless explicitly stated in the description of the structure.
- ❑ For a Switch or an RC, when AtomicOp Egress Blocking is enabled in an Egress Port, and an AtomicOp Request targets going out that Egress Port, the Egress Port must handle the Request as an AtomicOp Egress Blocked error[15] (see Figure 6-2) and must also return a Completion with a Completion Status of UR. If the severity of the AtomicOp Egress Blocked error is non-fatal, this case must be handled as an Advisory Non-Fatal Error as described in Section 6.2.3.2.4.1.

6.xx.3. Root Complex Support for AtomicOps

RCs have unique requirements and considerations with respect to AtomicOp capabilities.

6.xx.3.1 Root Ports with AtomicOp Completer Capabilities

AtomicOp Completer capability for a Root Port indicates that the Root Port supports receiving on its Ingress Port AtomicOp Requests that target host memory or Memory Space allocated by a Root Port BAR. This is independent of any Root Complex Integrated Endpoints that have AtomicOp Completer capabilities.

If a Root Port implements any AtomicOp Completer capability for host memory access, it must implement all 32-bit and 64-bit AtomicOp Completer capabilities. Implementing 128-bit CAS Completer capability is optional.

If an RC has one or more Root Ports that implement AtomicOp Completer capability, the RC must ensure that host memory accesses to a target location on behalf of a given AtomicOp Request are performed atomically with respect to each host processor or device access to that target location range.

If a host processor supports atomic operations via its instruction set architecture, the RC must also ensure that host memory accesses on behalf of a given AtomicOp Request preserve the atomicity of any host processor atomic operations.

---

[15] Though an AtomicOp Egress Blocked error is handled by returning a Completion with UR Status, the error is not otherwise handled as an Unsupported Request. For example, it does not set the Unsupported Request Detected bit in the Device Status register.

6.xx.3.2 Root Ports with AtomicOp Routing Capability

As with other PCIe Transactions, the support for peer-to-peer routing of AtomicOp Requests and Completions between Root Ports is optional and implementation dependent. If an RC supports AtomicOp routing capability between 2 or more Root Ports, it must indicate that capability in each associated Root Port via the AtomicOp Routing Supported bit in the Device Capabilities 2 register.

An RC is not required to support AtomicOp routing between all pairs of Root Ports that have the AtomicOp Routing Supported bit Set. An AtomicOp Request that would require routing between unsupported pairs of Root Ports must be handled as an Unsupported Request (UR), and reported by the "sending" Port.

The AtomicOp Routing Supported bit must be Set for any Root Port that supports forwarding of AtomicOp Requests initiated by host software or Root Complex Internal Endpoints. The AtomicOp Routing Supported bit must be Set for any Root Ports that support forwarding of AtomicOp Requests received on their Ingress Port to Root Complex Integrated Endpoints.

6.xx.3.3 RCs with AtomicOp Requester Capabilities

An RC is permitted to implement the capability for either host software or Root Complex Integrated Endpoints to initiate AtomicOp Requests.

Software discovery of AtomicOp Requester capabilities is outside the scope of this specification.

If an RC supports software-initiated AtomicOp Requester capabilities, the specific mechanisms for how software running on a host processor causes the RC to generate AtomicOp Requests is outside the scope of this specification.

IMPLEMENTATION NOTE

Generating AtomicOp Requests via Host Processor Software

If a host processor instruction set architecture (ISA) supports atomic operation instructions that directly correspond to one or more PCIe AtomicOps, an RC might process the associated internal atomic transaction that targets PCIe Memory Space much like it processes the internal read transaction resulting from a processor load instruction. However, instead of "exporting" the internal read transaction as a PCIe Memory Read Request, the RC would export the internal atomic transaction as a PCIe AtomicOp Request. Even if an RC uses the "export" approach for some AtomicOp types and operand sizes, it would not need to use this approach for all.

For AtomicOp types and operand sizes where the RC does not use the "export" approach, the RC might use an RC register-based mechanism similar to one where some PCI host bridges use CONFIG_ADDRESS and CONFIG_DATA registers to generate Configuration Requests. Refer to the PCI Local Bus Specification, Revision 3.0, for details.

The "export" approach may permit a large number of concurrent AtomicOp Requests without becoming RC register limited. It may also be easier to support AtomicOp Request generation from user space software using this approach.

The RC register-based mechanism offers the advantage of working for all AtomicOp types and operand sizes even if the host processor ISA doesn't support the corresponding atomic instructions. It might also support a polling mode for waiting on AtomicOp Completions as opposed to stalling the processor while waiting for a Completion.

6.xx.4. Switch Support for AtomicOps

If a Switch supports AtomicOp routing capability for any of its Ports, it must do so for all of them.

Modify Section 7.5.1.1. as shown

7.5.1.1. Command Register (Offset 04h)

Table 7-3 establishes the mapping between PCI 3.0 and PCI Express for PCI 3.0 Configuration Space Command register.

Table 7-3: Command Register

Bit Location	Register Description	Attributes
2	**Bus Master Enable** – Controls the ability of a PCI Express Endpoint to issue Memory[16] and I/O Read/Write Requests, and the ability of a Root or Switch Port to forward Memory and I/O Read/Write Requests in the Upstream direction …	RW

[16] The AtomicOp Requester Enable bit in the Device Control 2 register must also be Set in order for an AtomicOp Requester to initiate AtomicOp Requests, which are Memory Requests.

Modify Section 7.8.15. as shown

7.8.15. Device Capabilities 2 Register (Offset 24h)

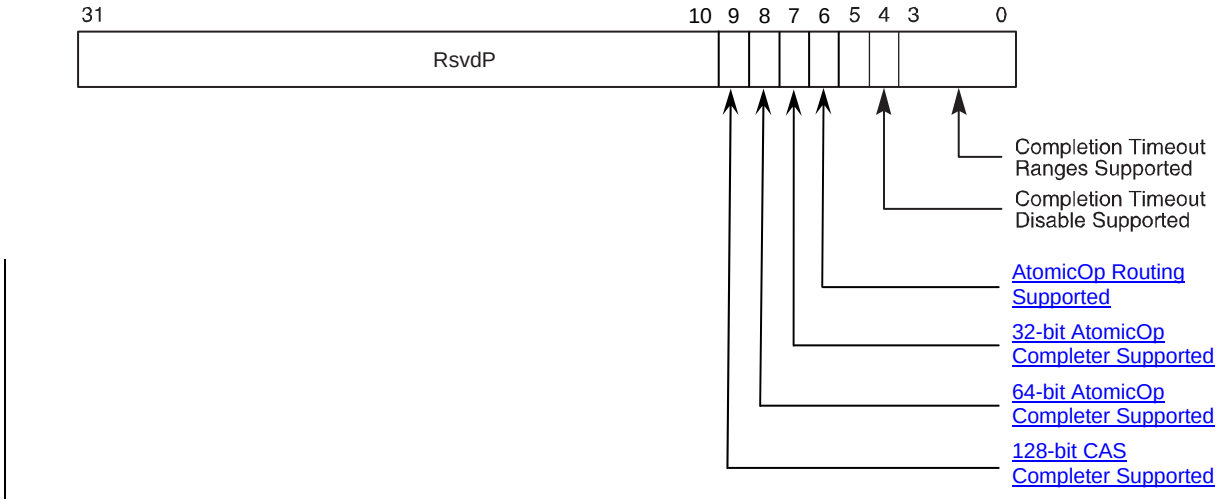

Figure 7-24 Device Capabilities 2 Register

Table 7-23 Device Capabilities 2 Register

Bit Location	Register Description	Attributes
…	…	…
6	**AtomicOp Routing Supported –** Applicable only to Switch Upstream Ports, Switch Downstream Ports, and Root Ports; must be 0b for other Function types. This bit must be set to 1b if the Port supports this optional capability. See Section 6.xx. for additional details.	RO
7	**32-bit AtomicOp Completer Supported –** Applicable to Functions with Memory Space BARs as well as all Root Ports; must be 0b otherwise. Includes FetchAdd, Swap, and CAS AtomicOps. This bit must be set to 1b if the Function supports this optional capability. See Section 6.xx.3.1. for additional RC requirements.	RO
8	**64-bit AtomicOp Completer Supported –** Applicable to Functions with Memory Space BARs as well as all Root Ports; must be 0b otherwise. Includes FetchAdd, Swap, and CAS AtomicOps. This bit must be set to 1b if the Function supports this optional capability. See Section 6.xx.3.1. for additional RC requirements.	RO
9	**128-bit CAS Completer Supported –** Applicable to Functions with Memory Space BARs as well as all Root Ports; must be 0b otherwise. This bit must be set to 1b if the Function supports this optional capability. See Section 6.xx. for additional details.	RO

Modify Section 7.8.16. as shown

7.8.16. Device Control 2 Register (Offset 28h)

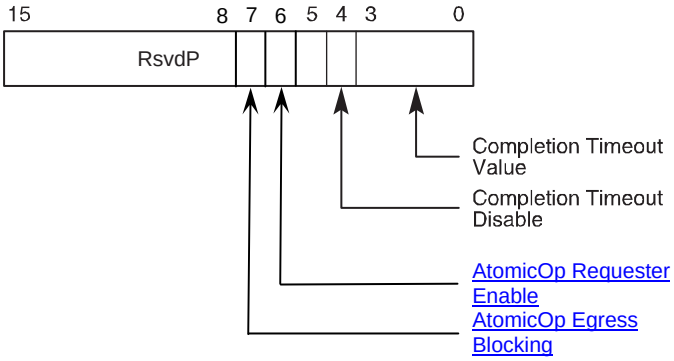

Figure 7-25 Device Control 2 Register

Table 7-24 Device Control 2 Register

Bit Location	Register Description	Attributes
…	…	…
6	**AtomicOp Requester Enable** – Applicable only to Endpoints and Root Ports; must be hardwired to 0b for other Function types. The Function is allowed to initiate AtomicOp Requests only if this bit and the Bus Master Enable bit in the Command register are both Set. This bit is required to be RW if the Endpoint or Root Port is capable of initiating AtomicOp Requests, but otherwise is permitted to be hardwired to 0b. This bit does not serve as a capability bit. This bit is permitted to be RW even if no AtomicOp Requester capabilities are supported by the Endpoint or Root Port. Default value of this bit is 0b.	RW
7	**AtomicOp Egress Blocking** – Applicable and mandatory for Switch Upstream Ports, Switch Downstream Ports, and Root Ports that implement AtomicOp routing capability; otherwise must be hardwired to 0b. When this bit is Set, AtomicOp Requests that target going out this Egress Port must be blocked. See Section 6.xx.2. Default value of this bit is 0b.	RW

Modify Figure 7-32 and Table 7-29

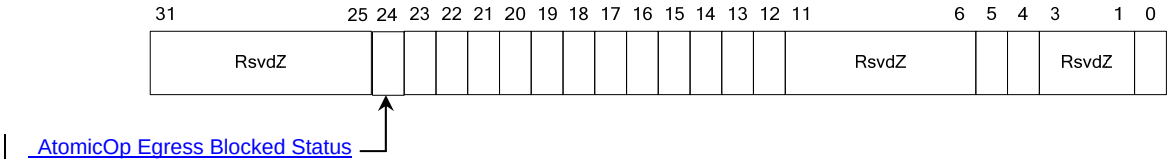

Figure 7-32 Uncorrectable Error Status Register

Table 7-29: Uncorrectable Error Status Register

Bit Location	Register Description	Attributes	Default
…	…	…	…
24	**AtomicOp Egress Blocked Status (Optional)**	RW1CS	0b

Modify Figure 7-33 and Table 7-30

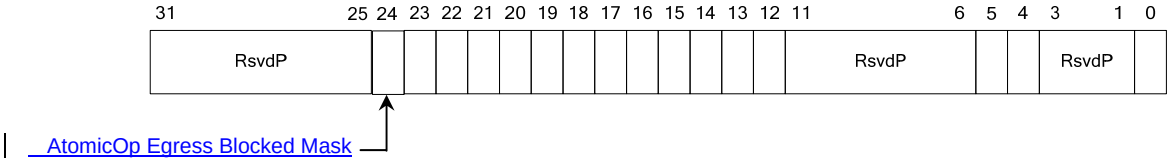

Figure 7-33: Uncorrectable Error Mask Register

Table 7-30: Uncorrectable Error Mask Register

Bit Location	Register Description	Attributes	Default
…	…	…	…
24	**AtomicOp Egress Blocked Mask (Optional)**	RWS	0b

Modify Figure 7-34 and Table 7-31

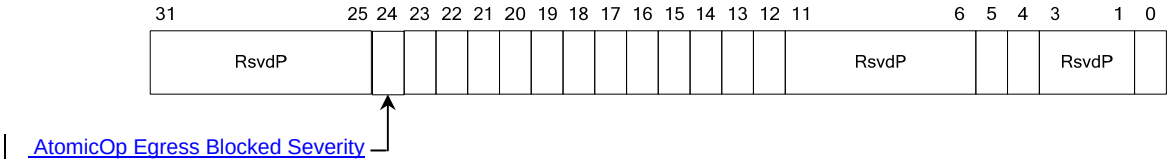

Figure 7-34: Uncorrectable Error Severity Register

Table 7-31: Uncorrectable Error Severity Register

Bit Location	Register Description	Attributes	Default
…	…	…	…
24	**AtomicOp Egress Blocked Severity (Optional)**	RWS	0b

Modify Section 7.16.3. as shown

7.16.3. ACS Control Register (Offset 06h)

…

Table 7-65: ACS Control Register

Bit Location	Register Description	Attributes
…	…	…
3	**ACS P2P Completion Redirect Enable (C)** – Determines when the component redirects peer-to-peer Completions Upstream; applicable only to ~~Read~~ Completions[17] whose Relaxed Ordering Attribute is clear. Default value of this bit is 0b. Must be hardwired to 0b if the ACS P2P Completion Redirect functionality is not implemented.	RW

[17] This includes Read Completions, AtomicOp Completions, and other Completions, either with or without Data.