## A1: Use of either/and, plural of TLP

*Page 13, Section 1.1 "Address Translation Services(ATS) Overview"*

As one can surmise, ATS Translation Request and Translation Completion processing is conceptually similar and, in many respects, identical to PCIe Read Request and Read Completion processing. This is intentional to reduce design complexity and to simplify integration of ATS into existing and new PCIe-based solutions. Keeping this in mind, ATS requires the following:

- ATS capable components must interoperate with *PCI Express Base Specification, Revision 1.1* compliant components.
- ATS is enabled through a new Capability and associated configuration structure. To enable ATS, software must detect this Capability and enable the Function to issue ATS TLPs. If a Function is not enabled, the Function is required not to issue ATS Translation Requests and is required to issue all DMA Read and Write Requests with the TLP AT field set to "untranslated."
- ATS TLPs are routed using either address-based ~~and~~or Requester ID (RID) routing.
- ATS TLPs are required to use the same ordering rules as specified within the *PCI Express Base Specification.*
- ATS TLPs are required to flow unmodified through PCIe 1.1-compliant Switches.
- A Function is permitted to intermix translated and untranslated requests.

## A2: Use of term "flush"

*Page 15, Section 1.1 "Address Translation Services(ATS) Overview"*

As Figure 1-4 illustrates, there are essentially three steps in the ATS Invalidation protocol:

1. The system software updates an entry in the tables used by the TA. After the table is changed, the TA determines that a translation should be invalidated in an ATC and initiates an Invalidation Request TLP which is transmitted from the RP to the example single-Function Device. The Invalidate Request communicates an untranslated address range, the TC, and an RP unique tag which is used to correlate Invalidate Completions with the Invalidation Request.
2. The Function receives the Invalidate Request and ~~flushes~~ invalidates all matching ATC entries. A Function is not required to immediately flush all pending requests upon receipt of an Invalidate Request. If transactions are in a queue waiting to be sent, it is not necessary for the Function to expunge requests from the queue even if those transactions use an address that is being invalidated.
    a. A Function is required not to indicate the invalidation has completed until all outstanding Read Requests or Translation Requests that reference the associated translated address have been retired or nullified.

b. A Function is required to ensure that the Invalidate Completion indication to the RC will arrive at the RC after any previously posted writes that use the "stale" address.

## A3: Identical Invalidation Responses

*Page 16, Section 1.1 "Address Translation Services(ATS) Overview"*

3. When a Function has ascertained that all uses of the translated address are complete, it issues one or more ATS Invalidate Completions.
   a. An Invalidate Completion is issued for each TC that may have referenced the range invalidated. These completions act as a flush mechanism to ensure the hierarchy is cleansed of any in-flight transactions which may contain references to the translated address.
      i. The number of Completions required is communicated within each Invalidate Completion. A TA or RC implementation can maintain a counter to ensure that all Invalidate Completions are received before considering the translation to no longer be in use.
      ii. If more than one Invalidation Complete is sent, the Invalidate Completion sent in each TC must be identical in the fields detailed in Section 3.2.

## A4: UR Clarification

*Page 18, Section 2.1 "Memory Request with Address Type"*

**Table 0-1: Address Type (AT) Field Encodings**

| AT Coding | Mnemonic | Meaning |
|---|---|---|
| 00b | Untranslated | A TA may treat the address as either virtual or physical. |
| 01b | Translation Request | The TA will return the translation of the address contained in the address field of the request as a read completion. This value only has meaning for an explicit Translation Request (see Section 2.2) ~~and will result in an UR if not in a Translation Request~~. A UR will result if the AT is set to 01b in a cycle other than Memory Read. |
| 10b | Translated | The address in the transaction has been translated by an ATC. If the Function associated with the SourceID is allowed to present physical addresses to the system memory, then the TA might not translate this address. If the Function is not allowed to present physical addresses, then the TA may treat this as an UR. |
| 11b | reserved | Results in an UR if used in a Memory Read or Memory Write Request. |

## A5: UR and ATS disable

**IOV Approved ??? -- Release Date???**

*Page 21, Section 2.3 "Translation Completion"*

**Table 0-1: Translation Completion with No Data Status Codes**

| Value | Status | Meaning |
|---|---|---|
| 000b | Success | This completion status has a nominal meaning of "success." The TA will not return this value in a Cpl. |
| 001b | Unsupported Request (UR) | Translation Requests from this Function are not supported by the TA. If a~~n ATC~~ Function receives this Completion code, it must disable its ATC and not send requests using translated addresses until the ATC is re-enabled. For transactions the Function may internally have in flight, the Function may either terminate or complete them. This may also cause an Error message to be sent if Advanced Error Reporting is supported. See AER within the *PCI Express Base Specification.* |
| 010b | CRS | This value is not allowed in any Completion to a request initiated by a PCI Express Function. If received by a Function, it shall be treated as a malformed packet. |
| 100b | Completer Abort (CA) | The TA was not able to translate the address because of an error in the TA. This nominally causes an error to be reported to the device driver associated with the ATC. See AER within the *PCI Express Base Specification.* |
| All others | Reserved | A Translation Completion with a Reserved Completion Status value is treated as if the Completion Status was Unsupported Request (001b). |

Note: Return values other than *Success* indicate an error.

## A6: Invalidation Completion Combining

*Page 29, Section 3.2 "Invalidation Completion"*

The ITag Vector field is used to indicate which Invalidate Request has been completed. Each of the 32 possible ITag field values from the Invalidation Request is represented by a single bit in the ITag Vector field. The least significant bit (bit 0; i.e., the right-most bit in the schematic representation of the Invalidate Completion message shown in Figure 3-3) of the ITag Vector field corresponds to the ITag field value of 0. The most significant bit (bit 31) of the ITag Vector field corresponds to the ITag field value of 31. Implementations are allowed to coalesce multiple Invalidate Completions by setting multiple ITag Vector bits in a single message provided the following conditions are met:

- The Invalidate Completions flow in the same TC.
- The Invalidate Completions have the same CC value.
- All fragments of an Invalidate Completion must have identical Request ID, CC, and ITag Vector fields.

~~When combining Invalidate Completions that are replicated in multiple TC (CC not equal to 1), each message of the replicated set may be combined independent of the others in the set.~~

## A7: Invalidation Completion Handling

**IOV Approved ??? -- Release Date???**

*Page 30, Section 3.2 "Invalidation Completion"*

Functions that do not support ATS ~~or that have ATS services disabled~~ will treat an Invalidate Request as UR. See the *PCI Express Base Specification* for further details.

Functions supporting ATS are required to send an Invalidate Completion in response to a Invalidate Request independent of whether the ~~BME~~ Bus Master Enable bit is Set or not. Note that the above conditions must be satisfied even when ~~BME~~ Bus Master Enable bit is Cleared. The method for a device to achieve this is implementation dependent.

---

IMPLEMENTATION NOTE

### ~~BME~~ Bus Master Enable Change

When ~~BME~~ Bus Master Enable changes from Set to Clear, no further memory requests should be queued. It is possible that queued write requests are present when ~~BME~~ Bus Master Enable is Cleared. These requests could block an Invalidate Completion. These requests must be either sent or dropped. This will ensure that all outstanding write transactions that are potentially dependent upon the outstanding invalidation are complete.

---

**A8: Terminology Clarification**

*Page 33, Section 3.6 "Invalidate Ordering Semantics"*

IMPLEMENTATION NOTE

## Request Range Overlap in Invalidations

In the description above, N is the number of STU sized translations that were requested in the Translation Request. This is equal to (Length field in Translation Request)/2.

As an example:

STU is 00 0010b indicating 16384-byte pages.

An outstanding Translation Request has a Length field of 00 0000 0100b indicating two translations covering a range of 32768 bytes.

The high-order 48 bits of the Translation Request are 0000 0FFF FFFFh.

The low-order 16 bits of the address in the request are 11xx xxxx xxxx xxxxb indicating that the translation request covers a range that overlaps a 32768-byte boundary (in fact, the request crosses a 16-TB boundary).

If two translations are returned, they would cover the two STU sized regions at 0000 0FFF FFFF C000h and 0000 1000 0000 0000h.

An Invalidate Request is received with the high-order 48 bits of 0000 1000 0000h and the low-order 16 bits of 0001 1xxx xxxx xxxxb.

The ATC must detect that a translation associated with a portion of the Translation Request is now invalidated and the Translation Completion associated with the invalidated region must be discarded (for simplification, the ATC is allowed to discard all of the Translation Completion).

It should be noted that, processing of the Invalidate~~s~~ Requests is simplified if Translation Requests do not cross alignment boundaries of the request. The Translation Request from the above example is not aligned to a 32768-byte boundary. If it were broken into two requests, it would be simpler to associate the range of the Invalidate Request with the address in the Translation Request. Breaking the Translation Requests into aligned requests is not a requirement.

**A9: Name correction**

*Page 37, "Acknowledgements"*

**Acknowledgements**

The following persons were instrumental in the development of the ATS Specification:[1]

Antonio ~~Ansaro~~ Asaro, Advanced Micro Devices, Inc.

Lucian Gozu, Neterion Corporation

Andrew Gruber, Advanced Micro Devices, Inc.

Mark Hummel, Advanced Micro Devices, Inc.

Michael Krause, Hewlett-Packard Corporation

Brian Langendorf, Nvidia Corporation

Renato Recio, IBM Corporation

Rajesh Sankaran, Intel Corporation

David Wooten, Microsoft Corporation

William Wu, Broadcom Corporation

## A10: Translation Completion padding clarification

*Page 26, "Completions With Multiple Translations"*

If a Translation Completion CplD has a Byte Count that is equal to four times the Length field, then the packet completes the request. For such a CplD, if the sum of Byte Count and Lower Address is not a multiple of RCB, then the CplD is the last of a sequence and it is an error if no previous CplD has been received, and all translation values should be discarded.

> Note: There are multiple reasons that the TA may truncate the results of the completion. For example, the request might ask for a range of addresses, not all of which are defined. This could occur if the first translation is valid but located at the end of a page of translations. The TA, in looking up the next page of translations, may find that the page is not valid so the addresses are not valid. The range of addresses that are valid would be returned and no error indicated. When truncating a Translation Completion the TA is not allowed to pad the response with invalid entries (R=0b,W=0b).

> Note: There are multiple reasons that the TA may break a Translation Completion into multiple TLPs. As an example, if the virtual address of the Translation Completion resolves to a table access that crosses an RCB boundary of the memory system, the completion to the TA may be broken into multiple completions by the memory. Rather than require that the TA accumulate the results, it is allowed to send each portion of the Translation Completion to a Function when it is received from the system memory.

**A11: ATS capability should be in extended configuration space**

*Page 35, section 4.1 "ATS Capability Structure"*

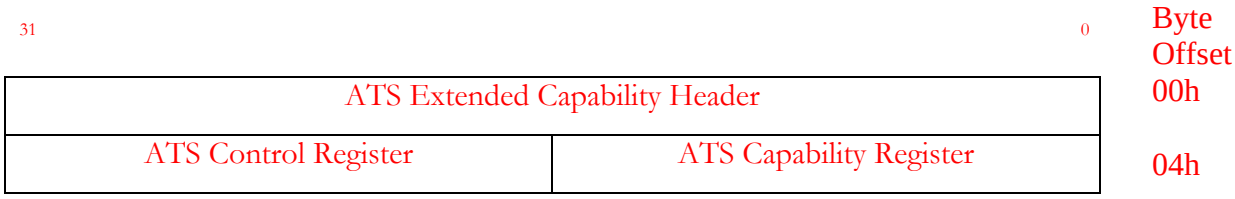
***Note: This section has been rewritten in entirety.***

## 4.1. ATS Capability Structure

Each Function that supports ATS must have the ATS Extended Capability structure in its extended configuration space.

Figure 4-1 details allocation of the register fields in the ATS Capability structure.

**Figure 4-1: ATS Extended Capability Header**




### 4.1.1 ATS Extended Capability Header

Figure 4-2 details the allocation of register fields of an ATS Capability Header, Table 4-1 provides the respective bit definitions.

**Figure 4-2: ATS Extended Capability Header**

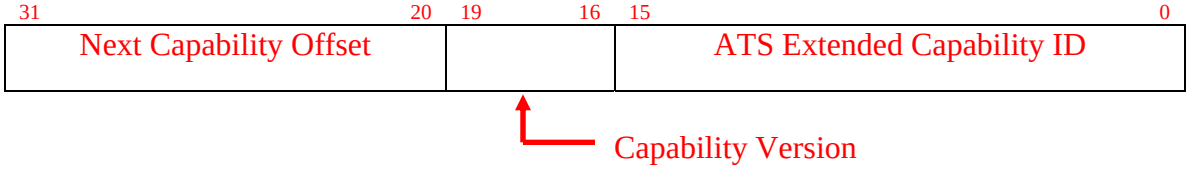



**Table 0-1: ATS Extended Capability Header**

| Bit Location | Register Description | Attributes |
|---|---|---|
| 15:0 | **ATS Extended Capability ID** – Indicates the ATS Extended Capability structure. This field must return a Capability ID of 000Fh indicating that this is an ATS Extended Capability structure. | RO |
| 19:16 | Capability Version – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present.<br><br>Must be 1h for this version of the specification | RO |

| Bit Location | Register Description | Attributes |
|---|---|---|
| 31:20 | **Next Capability Pointer** – The offset to the next PCI Extended Capability structure or 000h if no other items exist in the linked list of capabilities. | RO |

### 4.1.2 ATS Capability Register

Figure 4-3 details the allocation of register fields of an ATS Capability Register; Table 4-2 provides the respective bit definitions.

**Figure 4-3: ATS Capability Register**

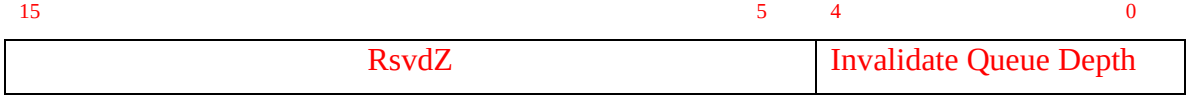



**Table 4-2: ATS Capability Structure**

| Bit Location | Register Description | Attributes |
|---|---|---|
| 4:0 | **Invalidate Queue Depth** – The number of Invalidate Requests that the Function can accept before putting backpressure on the upstream connection. If 0 0000b, the Function can accept 32 Invalidate Requests. | RO |

### *4.1.3* ATS Control Register

Figure 4-4 details the allocation of register fields of an ATS Control Register, Table 4-3 provides the respective bit definitions.

**Figure 4-4: ATS Control Register**

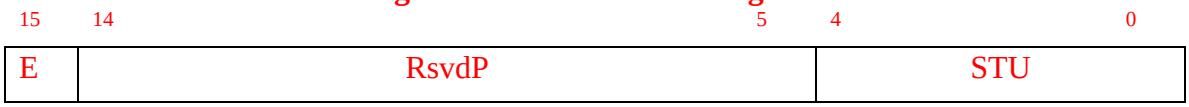



**Table 0-3: ATS Control Register**

| Bit Location | Register Description | Attributes |
|---|---|---|

| Bit Location | Register Description | Attributes |
|---|---|---|
| 4:0 | **Smallest Translation Unit (STU)** – This value indicates to the Function the minimum number of 4096-byte blocks that is indicated in a Translation Completions or Invalidate Requests. This is a power of 2 multiplier and the number of blocks is $2^{STU}$. A value of 0 0000b indicates one block and a value of 1 1111b indicates $2^{31}$ blocks (or 8-TB total) Default value is 0 0000b. | RW |
| 15 | **Enable (E)** – When Set, the Function is enabled to cache translations. Default value is 0b. | RW |

--- end of errata ---