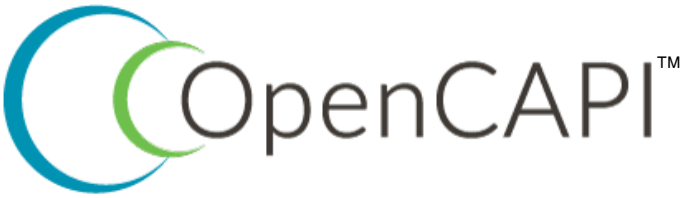

www.opencapi.org

OpenCAPI Discovery and Configuration Architecture Specification

Version 2.01
27 April 2020

Approved

Approved for Distribution to OpenCAPI Members
Approved for Distribution to Non-Members for Learning Purposes Only

OpenCAPI Discovery and Configuration Architectural Specification

OpenCAPI Enablement Work Group
OpenCAPI Consortium

Version 2.01 (27 April 2020)

Abstract

This document provides the architectural specification of the OpenCAPI Device Discovery and Configuration. It is the work product of the OpenCAPI Consortium Enablement Work Group.

This document is handled in compliance with the requirements outlined in the OpenCAPI Consortium Work Group (WG) process document. Comments, questions, etc. can be submitted to membership@opencapi.org.

Contents

List of figures

List of tables

Revision log

Each release of this document supersedes all previously released versions. The revision log lists all significant changes made to the document since its initial release. The use of change bars and mark up notation may be included and are noted in the revision log; change bars in the margin indicate that the adjacent text was modified from the previous release of this document.

Revision date	Summary of changes
27 April 2020	Version 2.01 • Approved change: Remove Application Specific information from Vendor Specific DVSEC. • Approved change: Clarify the term “Vendor ID” is a “PCI Vendor ID”.
18 March 2020	Version 2.0 • Approved change: Added Non-Power of 2 System Memory Size. • Approved change: Added AFU Template 0, Template Version Major/Minor values. • Approved change: Added TL 3.1 and TL 4.0 capabilities to AFU descriptor template 0 and AFU control DVSEC.
3 October 2018	Version 1.1. • Updated for the OpenCAPI Work Group template.
17 August 2017	Version 1.0. Initial release.

About this document

This document contains some terminology, general format information (for example, notes), and document conventions.

Architecture compliance terminology

In architecture descriptions, certain terms carry meaning in addition to their normal use in English. The following terms are used within this architecture specification to describe the requirements an implementation must meet to be considered compliant.

Table 1. Architecture terms

Term	Description
invalid	Used for multi-bit fields where the contents are not reliable. The field or bus shall not be examined for any functional or error checking actions.
may	An architectural option indicating that an implementation is allowed to have this behavior or characteristic.
reserved	With respect to a field of a register or bus: • A reserved field shall be set to 0 by an implementation. • A reserved field shall not be examined by an implementation. With respect to a code point: • A reserved code point shall not be issued by a compliant implementation • A reserved code point shall cause a bounded undefined response (that is, it won't hang the system). • A reserved code point may be used in future revisions of the architecture. The architecture may specify that the use of a reserved code point is an error condition.
shall	An architectural requirement indicating a required behavior or characteristic.
uncertain	Used for single-bit fields where the contents are not reliable. The field or bus shall not be examined for any functional or error checking actions.
undefined	When the value of a field or a bus is undefined, the value may vary between implementations and may vary for a particular implementation for different actions. An implementation shall not examine a field when its value is undefined for functional purposes. However, the field may be checked for errors in those cases where an implementation includes error checking (that is, parity, ECC and so on).

Conventions

The OpenCAPI Consortium documentation uses several typesetting conventions.

Bit and byte numbering

Throughout this document, little-endian notation is used, which means that bits and bytes are numbered in descending order from left to right.

Thus, in the description of a 4-byte field, bit 31 is the most significant bit (MSb) and bit 0 is the least significant bit (LSb). The corresponding byte numbering is also shown.

MSb																													LSb
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
byte 3								byte 2								byte 1								byte 0							

Big-endian and little-endian byte ordering are shown in *Figure 1* and described in the *POWER ISA, version 3.0, Book I.*

Figure 1. Big- and little-endian comparisons

LE	7	6	5	4	3	2	1	0
	Bit numbering within a byte							
BE	0	1	2	3	4	5	6	7

4-byte field with character data shown

LE	3	2	1	0
Content:	M	I	K	E
BE	0	1	2	3

Illustrating the difference between little endian and big endian storing to memory of the 4-byte field shown to the left.

Memory offset	LE stored	BE stored	
0	E	M	
1	K	I	
2	I	K	
3	M	E	

Representation of numbers

The notation for bit encoding is as follows:

- Hexadecimal values are preceded by x and enclosed in single quotation marks. For example x‘0A00’. Bit numbering is little endian and, in this example, is 15 to 0.
- Binary values in sentences are shown in single quotation marks. For example ‘1010’. Bit numbering in is little endian and, in this example, is 3 to 0.
- ^{n}x means the replication of x, n times. That is, x is concatenated to itself n-1 times. $^{n}0$ and $^{n}1$ are special cases:
  - $^{n}0$ means a field of n bits with each bit equal to 0. For example, $^{5}0$ is equivalent to ‘00000’.
  - $^{n}1$ means a field of n bits with each bit equal to 1. For example, $^{5}1$ is equivalent to ‘11111’.

RTL notation

RTL notations are used to specify the architectural transformation performed by execution of a command.

Notation	Meaning
←	Assignment.
||	Concatenation.
=, ≠	Equal, not equal relations.
≥, ≤	Greater than or equal to, less than or equal to relations.
+	Two's complement addition.
-	Two's complement subtraction, unary minus
∨	Bitwise logical OR
∧	Bitwise logical AND
⊕	Bitwise logical exclusive OR
Max(x,y)	Returns x when x ≥ y; otherwise, returns y.
Min(x,y)	Returns x when x ≤ y; otherwise, returns y.
{x...y}	All integer values from x through y.
A = {x...y}	Returns true when A is a member of the set of integer values in the range of x through y.

Reference documents

The following documents can be helpful when reading this specification.

- OpenCAPI Consortium (http://opencapi.org)
  - OpenCAPI Transaction Layer Specification
  - OpenCAPI Data Link Layer Architecture Specification
- PCI Special Interest Group (http://pcisig.com)

  Note: To obtain a valid PCI-SIG® Vendor ID, contact administration@pcisig.com. Use of PCI-SIG's Vendor IDs are subject to terms of use, restrictions and policies established by PCI-SIG. PCI-SIG is the registered trademark of PCI-SIG.
  - PCI Local Bus Specification
  - PCI Express Base Specification
  - PCI Code and ID Assignment Specification
- Other web sources:
  - https://en.wikipedia.org/wiki/PCI_configuration_space
  - http://wiki.osdev.org/PCI
  - http://wiki.osdev.org/PCI_Express
  - http://openpowerfoundation.org/wp-content/uploads/resources/hwarch-caia-spec/content/ch_preface.html

Notes

This section describes engineering, developer, and editor notes.

Engineering notes

Engineering notes provide additional implementation details and recommendations not found elsewhere. The notes might include architectural compliance requirements. That is, the text might include *Architecture compliance terminology*. These notes should be read by all implementation and verification teams to ensure architectural compliance.

Engineering note

This is an example of an Engineering note. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin cursus hendrerit enim, vel tempus nibh ornare ut. Quisque ac augue eu augue convallis hendrerit. Mauris iaculis viverra ipsum nec dapibus. Nunc at porta libero. Curabitur luctus ultrices augue non pulvinar. Vestibulum mattis non ipsum at venenatis. Suspendisse euismod, neque et suscipit luctus, odio metus semper lectus, quis volutpat est libero quis nunc. Vivamus rutrum mauris sed tristique malesuada. Vivamus at augue vitae nisl cursus feugiat.

Developer notes

Developer notes are used to document the reasoning and discussions that led to the current version of the architecture. These notes might also include recommended changes for future versions of the architecture, or warnings of approaches that have failed in the past. These notes should be read by verification teams and contributors to the architecture.

Developer note

This is an example of a Developer note. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin cursus hendrerit enim, vel tempus nibh ornare ut. Quisque ac augue eu augue convallis hendrerit. Mauris iaculis viverra ipsum nec dapibus. Nunc at porta libero. Curabitur luctus ultrices augue non pulvinar. Vestibulum mattis non ipsum at venenatis. Suspendisse euismod, neque et suscipit luctus, odio metus semper lectus, quis volutpat est libero quis nunc.

Editor notes

Editor notes are reminders to the editors and other contributors of additional work that is required. Editor notes may appear as red text in the body of any section or may include an entire section. This is to indicate text that is either out of date or is speculative (unapproved) in nature. The red text might also include directions to the editor for changes to be applied to a future revision of the document. Approved versions of a document are not expected to be released with red text unless approved by the owners of the document.

Terms

The following terms are used in this document.

acTag	Address context tag. The address context tag is managed by the AFU. The acTag is used as an index into a host table that contains the BDF and PASID associated with the acTag.
AFU	Attached functional unit. Architecturally, this term refers to an endpoint unit or resource. Communication from the processor to the AFU goes through a protocol stack, transaction layer (TL), data link layer (DL), and physical medium layer (PHY). Command and data packets at the AFU interface are specified by the AFU command/data interface, which is the interface between the AFU protocol stack and the AFU.
AFUTag	Unique handle specifying the AFU and command instance. Provided by the AFU.
AFU Unique Protocol	The commands, data, and registers defined by the AFU to control its operation. This is outside the scope of this document.
ATC	Address Translation Cache.
BAR	Base Address Register.
BDF	Bus Device Function. Addressing mechanism to bind an OpenCAPI entity to the host. The OpenCAPI device learns its BDF during a `config_write` operation.
cmd_flag	Platform defined special command tags.
default value	Value following a Reset.
DL	OpenCAPI data link layer found on the host processor. See *OpenCAPI Data Link Layer Architecture Specification*.
DLX	OpenCAPI data link layer found on the OpenCAPI device. See *OpenCAPI Data Link Layer Architecture Specification*.
DVSEC	Designated Vendor Specific Extended Capability.
Flit	An acronym for flow control digits. See *OpenCAPI Data Link Layer Architecture Specification*.
Function	Third level selection hierarchy in Bus, Device, Function (BDF) addressing.
Gbps	Gigabits per second.
GHz	Gigahertz.
LSb	Least significant bit.
MEM	The memory-mapped owner of the line. The owner could be the memory controller or the owner of a memory-mapped I/O space. Some coherency protocols refer to this as a point of coherency (POC). This is also referred to as non-BAR memory as the address range is not defined by a Base Address Register.

MMIO	Memory-mapped input/output. Refers to the mapping of the address space required by an I/O device for load or store operations into the system's address space.
MSb	Most-significant bit.
non-BAR Memory	Memory space whose address range is not defined by a Base Address Register. This is also referred to as MEM.
OCDE	OpenCAPI device enable. Out of band signal used for Warm Reset function. See *OpenCAPI Data Link Layer Architecture Specification*.
Obj_handle	Object Handle. Platform unique identifier.
PASID	Process Address Space ID.
PCI	Peripheral Component Interface.
PCIe	PCI Express.
PCI SIG	PCI Special Interest Group.
pL	Partial length. Specifies the number of data bytes specified for a partial write command.
PHY	Physical medium layer.
POC	Point of coherency.
RO	Read Only.
RW	Read Write.
TL	OpenCAPI transaction layer found on the host processor. See *OpenCAPI Transaction Layer Specification*.
TLX	OpenCAPI transaction layer found on the OpenCAPI device. See *OpenCAPI Transaction Layer Specification*.
VPD	Vital Product Data.

1. Overview

This document describes the discovery and configuration mechanisms for an OpenCAPI-compliant device. These mechanisms are compatible with PCI Configuration Space and are operating system, processor, and platform agnostic. This document is not intended to reproduce the PCIe standard for the configuration space but rather to define how the various configuration fields are used by an OpenCAPI-compliant device.

1.1 OpenCAPI model

OpenCAPI is defined to support multiple different classes of device types. Examples are shown in *Figure 1-1* and *Figure 1-2* on page 14.

Figure 1-1. OpenCAPI system model

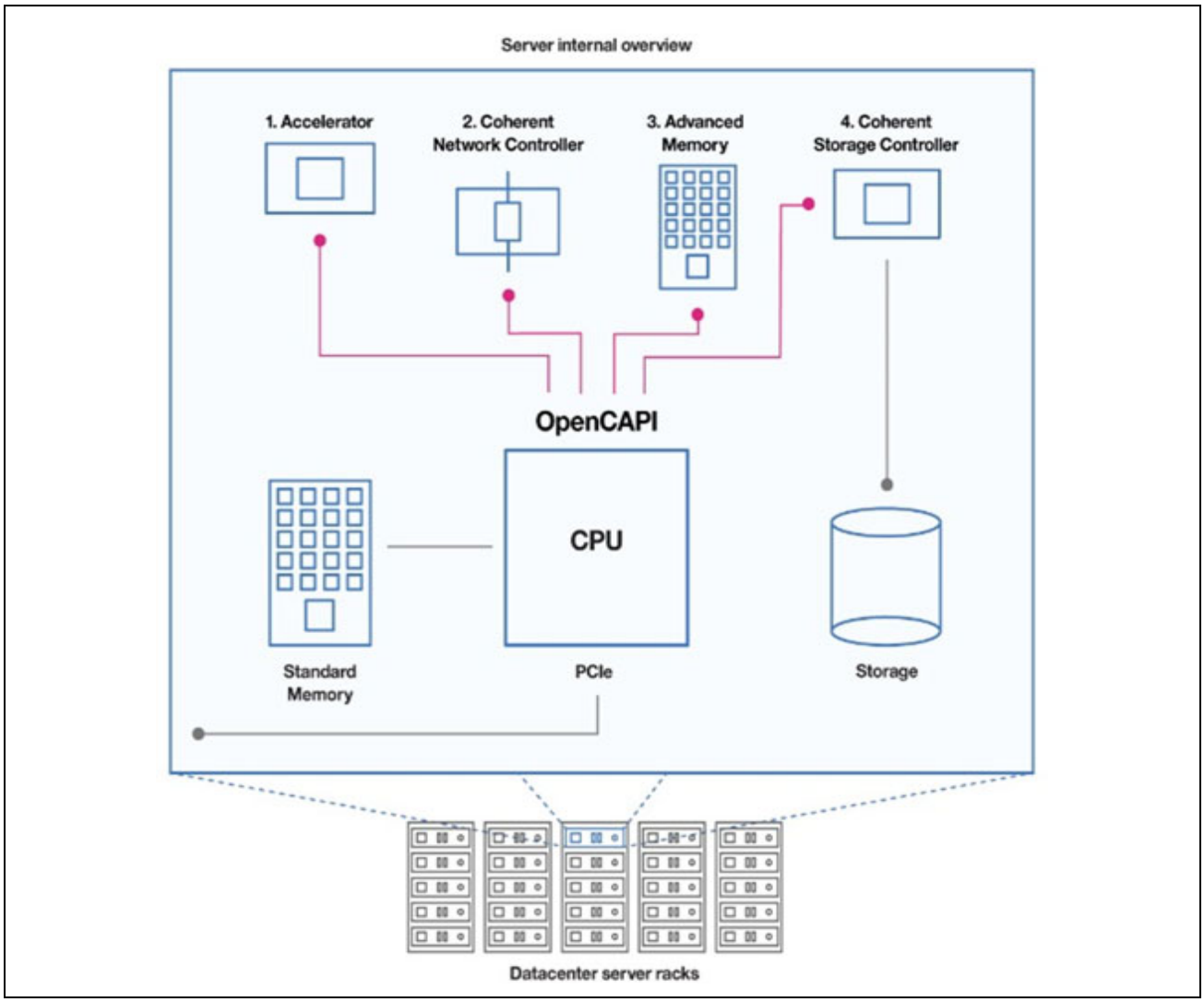

Figure 1-2. OpenCAPI device model **Note:** Each Function has a separate Configuration Space

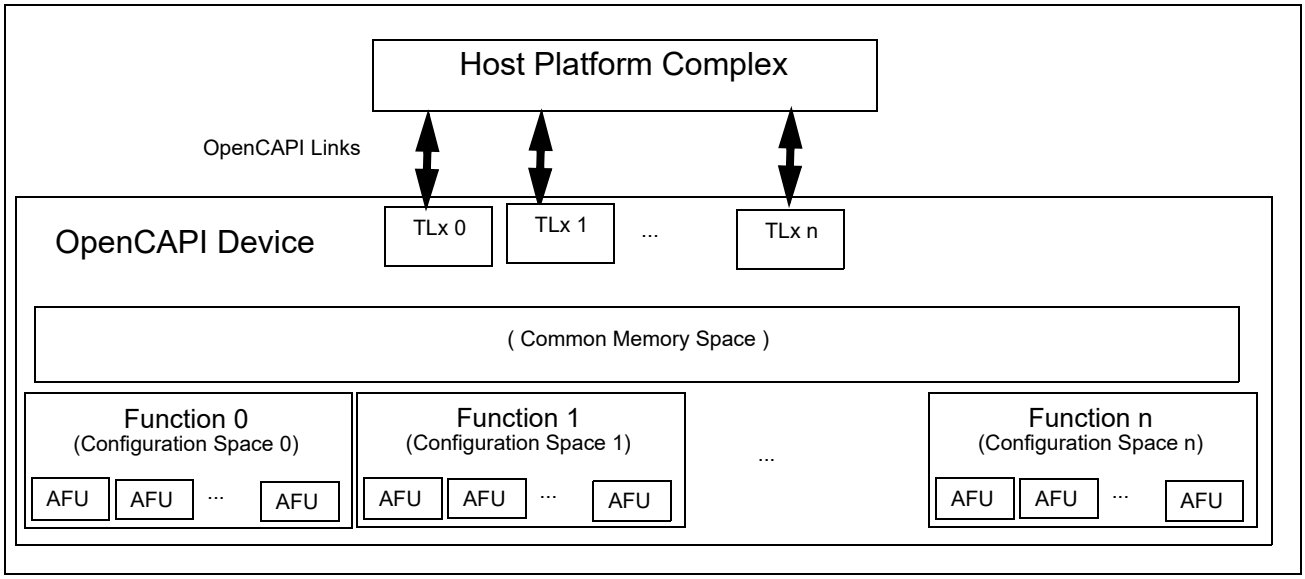

Table 1-1. OpenCAPI device model summary

Attribute	Maximum Number	AFU implications
Number of OpenCAPI Links.	256	1 → 256 independent links allowed. **Note:** Each OpenCAPI Link is treated as associated with a separate TLx stack. There is no ordering implied between the two stacks.
Number of Bus, Device, Function (BDF) as described in *OpenCAPI Transaction Layer Specification*.	Buses: 256 Devices: 32 Functions: 8	A Function (and all AFUs associated with that Function) is shared by all OpenCAPI Links.
Configuration Space (*Table 1-2* on page 15).	8	One per Function, shared by all AFUs associated with a Function. **Notes:** • Separate TLx Configuration Registers (see *OpenCAPI Transport Layer DVSEC* on page 26) exist for each TLx. • A TLx 's Configuration Registers can only be accessed via that TLx. • Function 0 (Configuration Space 0) contains the TLx Configuration Registers.
Number of Attached Functional Units (AFUs). The definition of these AFUs are outside the scope of this document.	64	0 - 64 AFUs per Function. These AFUs may be homogeneous or heterogeneous in any combination and multiple. An instantiation of an AFU is associated with a single Function.
Process Address Space ID (PASID) range.	1,048,576	One range shared by all AFUs associated with a specific Function.
acTag range.	4096	One range shared by all Functions associated with the device.

1.2 Configuration space

The configuration space is a separate address space accessed via configuration read and configuration write packets sent from the Host to the OpenCAPI Bus:Device:Function (BDF). There is a separate configuration space for each Function, see *Figure 1-2* on page 14. Legal lengths of configuration reads or writes are 1, 2 or 4 bytes. The configuration space is only used for discovery of the device, Function(s), AFU(s) and their capabilities, and to program certain settings that are needed for correct operation with a host. The bit and byte numbering for the registers in the configuration space is little-endian.

Note: It is recommended that only operating system trusted processes directly access the configuration space.

Table 1-2. Configuration space contents

Description	Offset
Configuration Space Header (*Section 2 Configuration space header* on page 19).	x‘00’ →
Capabilities (*Section 3 Capabilities* on page 22).	x‘FF’
Extended Capabilities (*Section 4 Extended Capabilities* on page 23).	x‘100’ → x‘FFF’

1.3 Memory space

The memory space is a separate address space accessed via read memory and write memory packets (see *OpenCAPI Transaction Layer Specification*) sent from the Host to the OpenCAPI Device (see *Figure 1-2* on page 14).

- Memory space has 64-bit addressing.
- Memory space address range is shared by all Functions associated with a Device (see *Figure 1-2* on page 14).
- The address ranges used by each Function are assigned using the Base Address Registers (BARs). See *OpenCAPI configuration header* on page 20.
- Exception: Non-BAR memory is defined by the *OpenCAPI Transaction Layer Specification* to begin at address x'0000_0000_0000_0000'. This may be distributed among multiple AFUs (see *Table 1-4* on page 16).

Table 1-3. Example memory space contents

Description	Address range
Non-BAR memory. (For example, Extended System Memory. Also referred to as MEM). See *Table 1-4*.	x‘0000_0000_0000_0000’ → size of Memory.
Expansion ROM.	Expansion ROM Base Address (Expansion ROM Base Address + Expansion ROM Size). **Note:** This must be in the low 4 GB address range. The upper address bits are x‘0000_0000’. **Note:** Expansion ROM is mutually exclusive with non-BAR memory > 2 GB.
Function n, BAR n Window. (For example, AFU Global Regs, AFU PASID Regs). **Note:** Each Function has 0 - 3 BAR windows. See *Table 4-16* on page 36.	Function n, BAR n Address → (Function n, BAR n Address + Function n, BAR n Size).

Table 1-4. Example distributed MEM

Address	Contents
x‘0000_0000_0000_0000’ → x‘mmmm_mmmm_mmmm_mmmm’ - 1	Function a, AFU w contained MEM.
x‘mmmm_mmmm_mmmm_mmmm’ → x‘nnnn_nnnn_nnnn_nnnn’ - 1	Function a, AFU x contained MEM.
x‘nnnn_nnnn_nnnn_nnnn’ → x‘pppp_pppp_pppp_pppp’ - 1	Function b, AFU y contained MEM.
x‘pppp_pppp_pppp_pppp’ → x‘qqqq_qqqq_qqqq_qqqq’ -1	Function c, AFU z contained MEM.

1.4 Resets

The following Reset types are defined in OpenCAPI.

Table 1-5. Reset Types

Reset Type	Scope	OpenCAPI Mechanism	Applicability		
			No AFU per Function	Single AFU per Function	Multiple AFU per Function
Cold Reset	Device + all Functions + all AFUs	Power on of the device.	Required	Required	Required
Warm Reset	Device + all Functions + all AFUs	Out-of-band signal “OpenCAPI Device Enable” is de-asserted. This is described in the *OpenCAPI Data Link Layer Architecture Specification*.	Required	Required	Required
Hot Reset	Device + all Functions + all AFUs	None. None defined by either *OpenCAPI Transaction Layer Specification* or *OpenCAPI Data Link Layer Architecture Specification*.	N/A	N/A	N/A
Function Reset	Function + all associated AFUs	Bit in Configuration Space Function DVSEC Capability (*Function DVSEC* on page 29).	Required	Required	Required
AFU Reset	AFU	Bit in Configuration Space AFU Control Extended Capability (*AFU control DVSEC* on page 37).	Allowed	Required	Required
		AFU Unique bit in MMIO BAR Space assigned to the AFU (*Table 4-16* on page 36).	N/A	Recommended	Recommended

1.5 Interrupts

Interrupts are reported using either the `intrp_req` or `intrp_req.d` commands as defined in *OpenCAPI Transaction Layer Specification.* The source of the content of these commands is shown in *Table 1-6*.

Table 1-6. Interrupt command parameter summary

Note: The system specifies which interrupt command an AFU sends in the AFU Unique Protocol.

Command Field	Size	Applicability		Source
		intrp_req	intrp_req.d	
`stream_id`	4 bits	X	X	AFU.
`cmd_flag`	4 bits	X	X	Specified by the system in the AFU Unique Protocol.
`acTag`	12 bits	X	X	AFU. **Note:** An AFU must send an assign_actag command to the system before using that acTag in an Interrupt.
`Obj_handle`	64 bits	X	X	Specified by the system in the AFU Unique Protocol.
`AFUTag`	16 bits	X	X	AFU.
`pL`	3 bits		X	AFU.
`Data`	pL		X	Specified by the system in the AFU Unique Protocol.

1.6 Thread wakeup

Platforms may support the `wake_host_thread` command as defined in *OpenCAPI Transaction Layer Specification*. The source of the content of this command is shown in *Table 1-7*.

Table 1-7. Wake Host Thread command parameter summary

Note: The system specifies when a `wake_host_thread` command is to be sent by an AFU in the AFU Unique Protocol.

Command Field	size	Applicability	Source
		wake_host_thread	
`stream_id`	4 bits	X	AFU.
`acTag`	12 bits	X	AFU.
`cmd_flag`	4 bits	X	Specified by the system in the AFU Unique Protocol.
`Obj_handle`	68 bits	X	Specified by the system in the AFU Unique Protocol.
`AFUTag`	16 bits	X	AFU.

2. Configuration space header

A configuration space header is required for each Function supported by an OpenCAPI device. This begins at offset x‘00’ in the configuration address space. Definition of this header and comprising fields is in *PCI Local Bus Specification* and *PCI Express Base Specification.*

Table 2-1. OpenCAPI configuration header format

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	Offset
Device ID		PCI Vendor ID		0x00
Reserved	C	Reserved	M rsv	0x04
Class Code			Revision ID	0x08
Reserved	MF Reserved	Reserved		0x0C
BAR 0 Low				0x10
BAR 0 High				0x14
BAR 1 Low				0x18
BAR 1 High				0x1C
BAR 2 Low				0x20
BAR 2 High				0x24
Reserved				0x28
Subsystem ID		Subsystem PCI Vendor ID		0x2C
Expansion ROM BAR				0x30
Reserved			Capabilities Pointer	0x34
Reserved				0x38
Reserved				0x3C

Table 2-2. OpenCAPI configuration header

Offset	Bits	Field Name	Description	Attributes
x‘00’	31:16	Device ID	The Device ID is an implementation-specific field.	RO
	15:0	PCI Vendor ID	The PCI Vendor ID is an implementation-specific field.	RO
x‘04’	31: 21	Reserved	Reserved = ‘0000_0000_000’.	RO
	20	Capabilities List	Capabilities list pointer supported = ‘1’.	RO
	19:2	Reserved	Reserved = ‘0000_0000_0000_0000_00’.	RO
	1	Memory Space	0 = Disable response to BAR memory space accesses. Non-BAR memory can still be accessed. All memory space accesses should be treated as if they are targeting non-BAR Memory. 1 = Enable response to BAR memory space accesses.	RW
	0	Reserved	Reserved = ‘0’.	RO
x‘08’	31:8	Class Code	The Class Code ID is an implementation-specific field.	RO
	7:0	Revision ID	The Revision ID is an implementation-specific field.	RO
x‘0C’	31:24	Reserved	Reserved = x‘00’.	RO
	23	MF	0 = Single-function device. 1 = Multi-function device.	RO
	22:16	Reserved	Reserved = ‘000_0000’.	RO
	15:0	Reserved	Reserved = x‘0000’.	RO
x‘10’	31:0	BAR 0 Low	Implementation-Specific Function MMIO Base Address Register 0 Low.	RW
x‘14’	31:0	BAR 0 High	Implementation-Specific Function MMIO Base Address Register 0 High.	RW
x‘18’	31:0	BAR 1 Low	Implementation-Specific Function MMIO Base Address Register 1 Low.	RW
x‘1C’	31:0	BAR 1 High	Implementation-Specific Function MMIO Base Address Register 1 High.	RW
x‘20’	31:0	BAR 2 Low	Implementation-Specific Function MMIO Base Address Register 2 Low.	RW
x‘24’	31:0	BAR 2 High	Implementation-Specific Function MMIO Base Address Register 2 High.	RW
x‘28’	31:0	Reserved	Reserved = x‘0000_0000’.	RO
x‘2C’	31:16	Subsystem ID	The Subsystem ID is an implementation-specific field.	RO
	15:0	Subsystem PCI Vendor ID	The Subsystem PCI Vendor ID is an implementation-specific field.	RO
x‘30’	31:11	Expansion ROM BAR	Implementation-Specific Function Expansion ROM Base Address Register Low. **Note:** High 32 bits are assumed to be x‘0000_0000’.	RW
	10:1	Reserved	Reserved = ‘000_0000_000’.	RO
	0	Expansion ROM Enable	0 = Disable Expansion ROM BAR. 1 = Enable Expansion ROM BAR.	RW
x‘34’	31:8	Reserved	Reserved = x‘0000_00’.	RO
	7:0	Capabilities Pointer	Configuration space offset of the first item in the capabilities list.	RO
x‘38’	31:0	Reserved	Reserved = x‘0000_0000’.	RO
x‘3C’	31:0	Reserved	Reserved = x‘0000_0000’.	RO

2.1 Base Address Registers

Base Address Registers (BARs) define the Memory Space addresses that the Function recognizes. The “Memory Space” bit in the Configuration header is used to enable/disable BAR defined Memory Space accesses for a Function.

The BAR window size for each BAR is a power of 2 and can be determined by writing all ‘1’s to the Address fields in the BAR register and then reading back the value contained in the Address fields. Zeros in the low bits identify offsets within the window. Thus, the first ‘1’ bit combined with the ‘0’ bits specifies the size of the window. If all ‘0’s are returned then a window is not implemented for that BAR.

After each window size is determined, the ‘1’s in the BAR address fields must be set to the base address the window is to be mapped to. After all BARs have been processed, the Memory Space bit must be set to ‘1’ to enable the decoding of the BAR windows.

Note: Example MMIO BAR usage is shown in *Table 4-16* on page 36.

Table 2-3. 64-bit Base Address Register format

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	Offset
BAR Address Low			P Type A	0x00
BAR Address High				0x04

Table 2-4. 64-bit Base Address Register

Offset	Bits	Field Name	Description	Attributes
x‘00’	31:4	BAR Address Low	Address Low.	RW
	3	Prefetchable	0 = Side effects on reads (for example, clear on read) or write has ordering dependencies. 1 = No side effects on reads and writes do not have ordering dependencies.	RO
	2:1	Type	Reserved = ‘10’ (all addresses are 64 bits).	RO
	0	Address Space	Reserved = ‘0’ (Memory Address Space).	RO
x‘04’	31:0	BAR Address High	Address high.	RW

3. Capabilities

The Capabilities are a linked list format that reside in Configuration address space. The Configuration header points to the first capability in the list. The format of these Capabilities is defined by the *PCI Local Bus Specification* and *PCI Express Base Specification*.

Table 3-1. Capability usage summary

ID	Capability	Applicability					
		No AFU per Function		Single AFU per Function		Multiple AFU per Function	
		Function 0	Function 1 - n	Function 0	Function 1 - n	Function 0	Function 1 - n
x'03'	Vital Product Data (VPD)	Read: Recommended Write: Allowed	Allowed	Read: Recommended Write: Allowed	Allowed	Read: Recommended Write: Allowed	Allowed

3.1 Vital Product Data capability

Encoding of Vital Product Data (VPD) is defined by the *PCI Local Bus Specification*.

Table 3-2. VPD capability format

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	Offset
F	VPD Address	Next Pointer	Capability ID	0x00
VPD Data				0x04

Table 3-3. Vital Product Data capability

Offset	Bits	Field Name	Description	Attributes
x'00'	31	Flag	'0' = Read VPD '1' = Write VPD **Read VPD Process:** 1. Write VPD address to desired VPD offset and Flag to 0. 2. Poll until Flag = 1. VPD data is now valid. **Write VPD Process:** 1. Write VPD data to desired data out. 2. Write VPD address to desired VPD offset and Flag to 1. 3. Poll until Flag = 0. VPD data has been written.	RW
	30:16	VPD Address	Offset of VPD data to be accessed	RW
	15:8	Next Pointer	Offset of next capability	RO
	7:0	Capability ID	VPD Capability = x'03'.	RO
x'04'	31:0	VPD Data	VPD data returned on a read or to be written on a write.	RW

4. Extended Capabilities

The Extended Capabilities are a linked list format that reside in the Configuration address space beginning at offset x‘100’. The format of the general Extended Capabilities is defined by the PCI SIG. The format of the OpenCAPI Designated Vendor-Specific Extended Capabilities (DVSEC) are defined in this document (see *Section 4.3 OpenCAPI DVSEC* on page 25).

Table 4-1. Extended Capability usage summary

Ext Cap ID	Extended Capability	Applicability					
		No AFU per Function		Single AFU per Function		Multiple AFU per Function	
		Function 0	Function 1 - n	Function 0	Function 1 - n	Function 0	Function 1 - n
x‘0003’	Device Serial Number	Single TLx Device - Recommended Multiple TLx Device - Required	Allowed	Single TLx Device - Recommended Multiple TLx Device - Required	Allowed	Single TLx Device - Recommended Multiple TLx Device - Required	Allowed
x‘001B’	Process Address Space ID (PASID)	Allowed		Required			
x‘0023’	Designated Vendor Specific Extended Capability (DVSEC)	See *Table 4-6 OpenCAPI DVSEC usage summary* on page 25.					

4.1 Device serial number Extended Capability

Table 4-2. Device serial number Extended Capability format

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	Offset
Next Capability Offset	Capability Version	Extended Capability ID		x‘00’
Serial Number Register Low				x‘04’
Serial Number Register High				x‘08’

Table 4-3. Device serial number Extended Capability

Offset	Bits	Field Name	Description	Attributes
x‘00’	31:20	Next Capability Offset	Offset of next Extended Capability	RO
	19:16	Capability Version	Version = x‘1’	RO
	15:0	Extended Capability ID	Device Serial Number Extended Capability = x‘0003’	RO
x‘04’	31:0	Serial Number Register Low	IEEE defined 64 bit extended unique identifier (EUI-64)	RO
x‘08’	31:0	Serial Number Register High		RO

4.2 Process address space ID Extended Capability

Table 4-4. Process address space ID (PASID) Extended Capability format

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	Offset
Next Capability Offset	Capability Version	Extended Capability ID		x‘00’
Reserved		Max PASID Width	Reserved	x‘04’

Table 4-5. Process address space ID Extended Capability

Offset	Bits	Field Name	Description	Attributes
x‘00’	31:20	Next Capability Offset	Offset of next Extended Capability	RO
	19:16	Capability Version	Version = x‘1’	RO
	15:0	Extended Capability ID	Process Address Space ID Extended Capability = x’001B’	RO
x‘04’	31:16	Reserved	Reserved = x‘0000’	RO
	15:13	Reserved	Reserved = ‘000’	RO
	12:8	Max PASID Width	PASIDs 0 through (2^n - 1) supported This is the total for all AFUs associated with this Function	RO
	7:0	Reserved	Reserved = x‘00’	RO

4.3 OpenCAPI DVSEC

Table 4-6. OpenCAPI DVSEC usage summary

Ext Cap ID	DVSEC ID	Capability	Applicability					
			No AFU per Function		Single AFU per Function		Multiple AFU per Function	
			Function 0	Function 1 - n	Function 0	Function 1 - n	Function 0	Function 1 - n
x‘0023’	x‘F000’	*OpenCAPI Transport Layer DVSEC* on page 26	Required	Prohibited	Required	Prohibited	Required	Prohibited
	x‘F001’	*Function DVSEC* on page 29	Required					
	x‘F003’	*AFU information DVSEC* on page 30 1 per Function	Allowed		Required			
	x‘F004’	*AFU control DVSEC* on page 37 **Note:** 1 per AFU	Allowed		Required			
	x’F005’ - x’F0BF’	Reserved for this Specification	Reserved					
	x‘F0C0’ - x’F0FF’	*Vendor-specific DVSEC* on page 40 (Not defined in this document)	Allowed					
	x’F100’ - x’FFFF’	Reserved for this Specification	Reserved					

4.3.1 OpenCAPI Transport Layer DVSEC

Note: This DVSEC only exists on Function 0.

Note: The rate limit for a particular template should be programmed before the template is configured as being enabled.

Table 4-7. OpenCAPI Transport Layer DVSEC format

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	Offset
Next Capability Offset	Capability Version	Extended Capability ID		x'00'
DVSEC Length	DVSEC Rev	DVSEC PCI Vendor ID		x'04'
Reserved		DVSEC ID		x'08'
TL Major Version Capability	TL Minor Version Capability	TLx Index	Reserved	x'0C'
TL Major Version Configuration	TL Minor Version Configuration	Reserved	Long Timer / Short Timer	x'10'
Reserved				x'14'
TLX Receive Template Capabilities(63:32)				x'18'
TLX Receive Template Capabilities(31:0)				x'1C'
TLX Transmit Template Configuration(63:32)				x'20'
TLX Transmit Template Configuration(31:0)				x'24'
Reserved				x'28'
Reserved				x'2C'
TLX Flit Receive Rate per Template Capabilities (Templates 63-56)				x'30'
TLX Flit Receive Rate per Template Capabilities (Templates 55-48)				x'34'
TLX Flit Receive Rate per Template Capabilities (Templates 47-40)				x'38'
TLX Flit Receive Rate per Template Capabilities (Templates 39-32)				x'3C'
TLX Flit Receive Rate per Template Capabilities (Templates 31-24)				x'40'
TLX Flit Receive Rate per Template Capabilities (Templates 23-16)				x'44'
TLX Flit Receive Rate per Template Capabilities (Templates 15-8)				x'48'
TLX Flit Receive Rate per Template Capabilities (Templates 7-0)				x'4C'
TLX Transmit Rate per Template Configuration (Templates 63-55)				x'50'
TLX Transmit Rate per Template Configuration (Templates 55-48)				x'54'
TLX Transmit Rate per Template Configuration (Templates 47-40)				x'58'
TLX Transmit Rate per Template Configuration (Templates 39-32)				x'5C'
TLX Transmit Rate per Template Configuration (Templates 31-24)				x'60'
TLX Transmit Rate per Template Configuration (Templates 23-16)				x'64'
TLX Transmit Rate per Template Configuration (Templates 15-8)				x'68'
TLX Transmit Rate per Template Configuration (Templates 7-0)				x'6C'
Reserved				x'70': x'8C'

Table 4-8. OpenCAPI Transport Layer DVSEC (Page 1 of 2)

Offset	Bits	Field Name	Description	Attributes
x‘00’	31:20	Next Capability Offset	Offset of next Extended Capability.	RO
	19:16	Capability Version	Version = x‘1’.	RO
	15:0	Extended Capability ID	DVSEC = x‘0023’.	RO
x‘04’	31:20	DVSEC Length	Length of this capability beginning at offset x‘00’.	RO
	19:16	DVSEC Revision	DVSEC structure revision level = x‘0’.	RO
	15:0	DVSEC PCI Vendor ID	DVSEC PCI Vendor ID = x‘1014’.	RO
x‘08’	31:16	Reserved	Reserved = x‘0000’.	RO
	15:0	DVSEC ID	DVSEC ID = x‘F000’.	RO
x‘0C’	31:24	TL Major Version Capability	This binary encoded field identifies the *OpenCAPI Transaction Layer Specification* Major version that the device implements. (for example, Version 3.0, Major = 3, Minor = 0).	RO
	23:16	TL Minor Version Capability	This binary encoded field identifies the *OpenCAPI Transaction Layer Specification* Minor version that the device implements.	RO
	15:8	TLx Index	Internal device TLx number that the Configuration read came in on. **Note:** The system can determine which CAPI links are connected to the same multi-ported device by comparing the Device Serial Number Extended Capabilities read on the Links (see *Device serial number Extended Capability* on page 23).	RO
	7:0	Reserved	Reserved = ‘000’.	RO
x‘10’	31:24	TL Major Version Configuration	This binary encoded field identifies the *OpenCAPI Transaction Layer Specification* Major version that the device is allowed to send. **Note:** It is recommended that this field be read back after it is written to verify that the device supports the requested version.	RW
	23:16	TL Minor Version Configuration	This binary-encoded field identifies the *OpenCAPI Transaction Layer Specification* Minor version that the device is allowed to send. **Note:** It is recommended that this field be read back after it is written to verify the device supports the requested version.	RW
	15:8	Reserved	Reserved = x’00’.	RO
	7:4	Long Back-off Timer	Long back-off timer for long back-off retry responses. Back-off timer value = 100ns * $2^{(2 * \text{Long Back-off Timer})}$ Range = 100 ns $\rightarrow$ 107.4 sec.	RW
	3:0	Short Back-off Timer	Short back-off timer for short back-off retry response. Back-off timer value = 100ns * $2^{\text{Short Back-off Timer}}$ Range = 100 ns $\rightarrow$ 3.28 ms.	RW
x‘14’	31:0	Reserved	Reserved = x‘0000_0000’	RO
x‘18’	31:0	TLX Receive Template Capabilities (63:32)	Each bit corresponds to the one of the 64 template definitions in the *OpenCAPI Transaction Layer Specification* that TLx supports receiving. **Note:** Template 0 must be supported. Template 0 default value = ‘1’.	RO
x‘1C’	31:0	TLX Receive Template Capabilities (31:0)		
x‘20’	31:0	TLX Transmit Template Configuration (63:32)	Each bit corresponds to the one of the 64 template definitions in the *OpenCAPI Transaction Layer Specification* that TL supports receiving. **Note:** Template 0 must be supported. Template 0 default value = ‘1’. **Note:** It is recommended that this field be read back after it is written to verify the Device supports the requested templates.	RW
x‘24’	31:0	TLX Transmit Template Configuration (31:0)		
x‘28’	31:0	Reserved	Reserved = x’0000_0000’.	RO

Table 4-8. OpenCAPI Transport Layer DVSEC (Page 2 of 2)

Offset	Bits	Field Name	Description	Attributes
x‘2C’	31:0	Reserved	Reserved = x‘0000_0000’.	RO
x‘30’	31:0	Rcv Rate (Templates 63-56)	TLX Flit Receive Rate per Template Capabilities. Each template that the TLX is capable of receiving has a corresponding 4-bit receive rate defined in the *OpenCAPI Transaction Layer Specification*.	RO
x‘34’	31:0	Rcv Rate (Templates 55-48)		
x‘38’	31:0	Rcv Rate (Templates 47-40)		
x‘3C’	31:0	Rcv Rate (Templates 39-32		
x‘40’	31:0	Rcv Rate (Templates 31-24)		
x‘44’	31:0	Rcv Rate (Templates 23-16)		
x‘48’	31:0	Rcv Rate (Templates 15-8)		
x‘4C’	31:0	Rcv Rate (Templates 7-0)		
x‘50’	31:0	Xmit Rate (Templates 63-56)	TLX Transmit Rate per Template Configuration. Each template that the TL is capable of receiving has a corresponding 4-bit receive rate defined in the *OpenCAPI Transaction Layer Specification*. **Note:** Template 0 must be supported. Template 0 default value = ‘1111’. **Note:** All values for all supported templates must be supported.	RW
x‘54’	31:0	Xmit Rate (Templates 55-48)		
x‘58’	31:0	Xmit Rate (Templates 47-40)		
x‘5C’	31:0	Xmit Rate (Templates 39-32)		
x‘60’	31:0	Xmit Rate (Templates 31-24)		
x‘64’	31:0	Xmit Rate (Templates 23-16)		
x‘68’	31:0	Xmit Rate (Templates 15-8)		
x‘6C’	31:0	Xmit Rate (Templates 7-0)		
x‘70’ : x‘8C’	31:0	Reserved	Reserved = x‘0000_0000’.	RO

4.3.2 Function DVSEC

This capability contains attributes scoped at the Function level and is required for all Functions.

Table 4-9. Function DVSEC format

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	Offset
Next Capability Offset	Capability Version	Extended Capability ID		x‘00’
DVSEC Length	DVSEC Rev	DVSEC PCI Vendor ID		x‘04’
A rsv Max AFU Index	R Reserved	DVSEC ID		x‘08’
Reserved	Function acTag Base	Reserved	Function acTag Length Enabled	x‘0C’

Table 4-10. Function DVSEC

Offset	Bits	Field Name	Description	Attributes
x‘00’	31:20	Next Capability Offset	Offset of next Extended Capability.	RO
	19:16	Capability Version	Version = x‘1’.	RO
	15:0	Extended Capability ID	DVSEC = x‘0023’.	RO
x‘04’	31:20	DVSEC Length	Length of this capability beginning at offset x‘00’.	RO
	19:16	DVSEC Revision	DVSEC structure revision level = x‘0’.	RO
	15:0	DVSEC PCI Vendor ID	DVSEC PCI Vendor ID = x‘1014’.	RO
x‘08’	31	AFU Present	0 = No AFUs are associated with this Function. 1 = One or more AFUs are associated with this Function.	RO
	30	Reserved	Reserved = ‘0’.	RO
	29:24	Max AFU Index	Largest value of any Index that references an AFU. **Note:** AFUs associated with the Function are not required to be referenced by consecutive indexes. Sparsely indexed AFUs are permitted.	RO
	23	Function Reset	1 = Reset Function. Resets the Function and all AFUs associated with the Function. Does not affect other Functions or AFUs.	W
	22:16	Reserved	Reserved = ‘000_0000’.	RO
	15:0	DVSEC ID	DVSEC ID = x‘F001’.	RO
x‘0C’	31:28	Reserved	Reserved = ‘0000’.	RO
	27:16	Function acTag Base	First acTag this Function is allowed to use.	RW
	15:12	Reserved	Reserved = ‘0000’.	RO
	11:0	Function acTag Length Enabled	Number of consecutive acTags this Function and all AFUs associated with this Function are allowed to use. x‘000’: no acTags are valid. x‘001’ - x‘FFF’: Valid Function acTag range = Function acTag Base → (Function acTag Base + Function acTag Length Enabled - 1). This range is shared by all AFUs associated with this Function.	RW

4.3.3 AFU information DVSEC

This capability is a means to extract common, general information about all of the AFUs associated with a Function independent of the specific functionality that each AFU provides.

Note: There is one AFU Information DVSEC per Function that has AFUs associated with it.

Table 4-11. AFU information DVSEC format

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	Offset
Next Capability Offset		Capability Version	Extended Capability ID	x‘00’
DVSEC Length		DVSEC Rev	DVSEC PCI Vendor ID	x‘04’
Reserved		AFU Info Index	DVSEC ID	x‘08’
V	AFU Descriptor Offset			x‘0C’
AFU Descriptor Data				x‘10’

Table 4-12. AFU information DVSEC

Offset	Bits	Field Name	Description	Attributes
x‘00’	31:20	Next Capability Offset	Offset of next Extended Capability.	RO
	19:16	Capability Version	Version = x‘1’.	RO
	15:0	Extended Capability ID	DVSEC = x‘0023’.	RO
x‘04’	31:20	DVSEC Length	Length of this capability beginning at offset x‘00’.	RO
	19:16	DVSEC Revision	DVSEC structure revision level = x‘0’.	RO
	15:0	DVSEC PCI Vendor ID	DVSEC PCI Vendor ID = x‘1014’.	RO
x‘08’	31:22	Reserved	Reserved = ‘0000_0000_00’.	RO
	21:16	AFU Info Index	AFU associated with the fields in this DVSEC.	RW
	15:0	DVSEC ID	DVSEC ID = x‘F003’.	RO
x‘0C’	31	Data Valid	AFU Descriptor Data field is Valid.	RW
	30:0	AFU Descriptor Offset	AFU descriptor 4-byte template offset to read. This register is written with the 4-Byte offset of Descriptor Template data that is to be read. The AFU Descriptor Data Valid bit is set to ‘0’ to initiate the read at the desired offset. The Data Valid bit will change to ‘1’ when the AFU Descriptor Data Register is valid.	RW
x‘10’	31:0	AFU Descriptor Data	AFU descriptor template data. See *Section 4.3.3.1 AFU descriptor template 0* on page 31. **Note:** x’0000_0000’ will be returned if data does not exist for the specified AFU info index or AFU descriptor offset.	RO

4.3.3.1 AFU descriptor template 0

Table 4-13. AFU descriptor template 0 format

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	Offset
Template Length		Template Version Major	Template Version Minor	x‘00’
Name Space[3]	Name Space[2]	Name Space[1]	Name Space[0]	x‘04’
Name Space[7]	Name Space[6]	Name Space[5]	Name Space[4]	x‘08’
Name Space[11]	Name Space[10]	Name Space[9]	Name Space[8]	x‘0C’
Name Space[15]	Name Space[14]	Name Space[13]	Name Space[12]	x‘10’
Name Space[19]	Name Space[18]	Name Space[17]	Name Space[16]	x‘14’
Name Space[23]	Name Space[22]	Name Space[21]	Name Space[20]	x‘18’
AFU Version Major	AFU Version Minor	AFU_c | AFU_m | Rsvd	Profile	x‘1C’
Global MMIO Offset Low		Reserved	Global MMIO Bar	x‘20’
Global MMIO Offset High				x‘24’
Global MMIO Size				x‘28’
C1 | C3 | B2 | PM | MC | Reserved	AM | P2 | P1 | host_tag Size	Reserved		x‘2C’
Per Process MMIO Offset Low		Reserved	Per Process MMIO Bar	x‘30’
Per Process MMIO Offset High				x‘34’
Per Process MMIO Stride		Reserved		x‘38’
Reserved			MEM Size	x‘3C’
MEM Start Address Low				x‘40’
MEM Start Address High				x‘44’
NAA WWID[3]	NAA WWID[2]	NAA WWID[1]	NAA WWID[0]	x‘48’
NAA WWID[7]	NAA WWID[6]	NAA WWID[5]	NAA WWID[4]	x‘4C’
NAA WWID[11]	NAA WWID[10]	NAA WWID[9]	NAA WWID[8]	x‘50’
NAA WWID[15]	NAA WWID[14]	NAA WWID[13]	NAA WWID[12]	x‘54’
System Memory Length Low				x‘58’
System Memory Length High				x‘5C’

Table 4-14. AFU descriptor template 0 (Page 1 of 4)

Offset	Bits	Field Name	Description	Attributes
x‘00’	31:16	Template Length	Length of this template beginning at offset x‘00’.	RO
	15:8	Template Version Major	This binary encoded field identifies the Major version of this document that the AFU implements. (For example, Version 1.0, Major = 1, Minor = 0.)	RO
	7:0	Template Version Minor	This binary encoded field identifies the Minor version of this document that the AFU implements. Major = x‘01’, Minor = x‘00’ : Minimum template length = 0x58 Major = x‘01’, Minor = x‘01’ : Minimum template length = 0x60 (System Memory length fields added)	RO
x‘04’	31:0	Name Space	24-byte ASCII character string of descriptive name space for the AFU. Format: <Vendor>, <AFU Name> Low offset aligned, padded with x’00’ bytes Permitted Characters: Alphanumeric, hyphen, underscore, comma	RO
x‘08’	31:0			
x‘0C’	31:0			
x‘10’	31:0			
x‘14’	31:0			
x‘18’	31:0			

Table 4-14. AFU descriptor template 0 (Page 2 of 4)

Offset	Bits	Field Name	Description	Attributes
x‘1C’	31:24	AFU Version Major	This binary-encoded field identifies the Major version of the AFU.	RO
	23:16	AFU Version Minor	This binary-encoded field identifies the Minor version of the AFU.	RO
	15:13	AFU_c Type	AFU_c Subclass Type (see *OpenCAPI Transaction Layer Specification*). ‘000’: AFU_{c0} ‘001’: AFU_{c1} ‘010’: AFU_{c2} ‘011’ → 111’: Reserved	RO
	12:10	AFU_m Type	AFU_m Subclass Type (see *OpenCAPI Transaction Layer Specification*). ‘000’: AFU_{m0} ‘001’: AFU_{m1} ‘010’: AFU_{m2} ‘011’ → ‘111’: Reserved	RO
	9:8	Reserved	Reserved =’00’	RO
	7:0	Profile	OpenCAPI Profile (see *OpenCAPI Transaction Layer Specification*). x‘00’: None x‘01’: OpenCAPI 3.0 Device Interface Class x‘02’: OpenCAPI 3.1 Memory Interface Class x‘03’ → x‘FF’: Reserved	RO
x‘20’	31:16	Global MMIO Offset Low	Bits 31:16 of the 64-bit offset from the MMIO BAR to the start of the Global MMIO space for this AFU. Bits 15:0 of the 64-bit offset are assumed to be x’0000’ as the offset must be 64 KB aligned.	RO
	15:3	Reserved	Reserved = ‘0000_0000_0000_0’.	RO
	2:0	Global MMIO Bar	Identifies BAR register which contains the Global MMIO space. 0: 64-bit BAR 0 2: 64-bit BAR 1 4: 64-bit BAR 2	RO
x‘24’	31:0	Global MMIO Offset High	High 32 bits of the offset from MMIO BAR to the start of the Global MMIO space for this AFU.	RO
x‘28’	31:0	Global MMIO Size	Size of the Global MMIO space. (For example, 1 MB = x‘0010_0000’.)	RO

Table 4-14. AFU descriptor template 0 (Page 3 of 4)

Offset	Bits	Field Name	Description	Attributes
x‘2C’	31	C1	[OpenCAPI 4.0] cmd_flag x‘1’ support on disable_cache, enable_cache, disable_atc, and enable_atc TL commands. ’0’: cmd_flag x‘1’ is not supported. ’1’: cmd_flag x‘1’ is supported.	RO
	30	C3	[OpenCAPI 4.0] cmd_flag x‘3’ support on disable_cache, enable_cache, disable_atc, and enable_atc TL commands. ’0’: cmd_flag x‘3’ is not supported. ’1’: cmd_flag x‘3’ is supported.	RO
	29	B2	256-byte TL operations supported. ‘0’: 256 byte TL operations are not supported. ’1’: 256 byte TL operations are supported.	RO
	28	PM	[OpenCAPI 3.1] TL Pad memory command supported ‘0’: Pad memory command is not supported. ’1’: Pad memory command is supported.	RO
	27	MC	[OpenCAPI 3.1] TL Memory control command supported ‘0’: Memory control command is not supported. ’1’: Memory control command is supported.	RO
	26:24	Reserved	Reserved = ‘000’.	RO
	23	AM	[OpenCAPI 4.0] TL AMO Read, AMO Read Write, and AMO Write commands supported. ‘0’: AMO commands are not supported. ’1’: AMO commands are supported.	RO
	22	P2	[OpenCAPI 4.0] Address Translation Cache (ATC) 2M Page Size Supported. ’0’: 2 MB page size is not supported. ’1’: 2 MB page size is supported. **Note:** 4 KB page size must always be supported.	RO
	21	P1	[OpenCAPI 4.0] Address Translation Cache (ATC) 64K Page Size Supported. ’0’: 64 KB page size is not supported. ’1’: 64 KB page size is supported. **Note:** 4 KB page size must always be supported.	RO
	20:16	host_tag Size	[OpenCAPI 4.0] Maximum size (in bits) supported of the host_tag for caching. x‘00’ : Caching not supported. x‘01’- x‘05’ : Reserved. x‘06’ - x‘18’ : Maximum size of host_tag. x‘19’ - x‘1F’ : Reserved.	RO
	15:0	Reserved	Reserved = x‘0000’.	RO
x‘30’	31:16	Per PASID MMIO Offset Low	Bits 31:16 of the 64-bit offset from the MMIO BAR to the start of the per PASID MMIO space for this AFU. Bits 15:0 of the 64-bit offset are assumed to be x‘0000’ as the offset must be 64 KB aligned.	RO
	15:3	Reserved	Reserved = ‘0000_0000_0000_0’	RO
	2:0	Per PASID MMIO Bar	Identifies the BAR register which contains the per PASID MMIO space. 0: 64-bit BAR 0. 2: 64-bit BAR 1. 4: 64-bit BAR 2.	RO
x‘34’	31:0	Per PASID MMIO Offset High	High 32 bits of the offset from MMIO BAR to the start of the per PASID MMIO space for this AFU.	RO

Table 4-14. AFU descriptor template 0 (Page 4 of 4)

Offset	Bits	Field Name	Description	Attributes
x‘38’	31:16	Per PASID MMIO Stride	Bits 31:16 of the 32-bit size of the MMIO space associated with each PASID. Bits 15:0 of the 32-bit Stride are assumed to be x‘0000’ as the Stride must be 64 KB aligned. The process MMIO space resides at Per PASID MMIO Bar + Per PASID MMIO Offset + (PASID index * Per PASID MMIO Stride).	RO
	15:0	Reserved	Reserved = x‘0000’.	RO
x‘3C’	31:8	Reserved	Reserved = x‘0000_00’.	RO
	7:0	MEM Size	MEM space size. x‘00’: MEM Space does not exist OR descriptor is being read on a TLx that does not have access to MEM. x‘01’ - x’FF’ : MEM Space Size = $2^{\text{MEM Size}}$ (for example, x’1E’ = 1 GB). **Note:** MEM memory is also referred to as non-BAR memory. **Note:** The AFU is not required to contain backing memory to cover the entire MEM Size. See *Table 4-15 Example AFU MEM Space Contents* on page 35.	RO
x‘40’	31:0	MEM Start Address Low	Low 32 bits of the Memory Space Start address of the MEM segment contained within this AFU. This must be aligned on a multiple of MEM Space size.	RO
x‘44’	31:0	MEM Start Address High	High 32 bits of the Memory Space Start address of the MEM segment contained within this AFU. This must be aligned on a multiple of MEM Space size.	RO
x‘48’	31:0	NAA WWID	NAA Formatted World Wide Unique ID. x’0000_0000_0000_0000_0000_0000_0000_0000’ = AFU does not support a WWID	RO
x‘4C’	31:0			
x‘50’	31:0			
x‘54’	31:0			
x‘58’	31:0	System Memory Length Low	Low 32 bits of the length of the General Purpose System Memory in bytes. This is not required to be a power of 2. **Note:** This must be a multiple of 64 KB.	RO
x‘5C’	31:0	System Memory Length High	High 32 bits of the length of the General Purpose System Memory in bytes. This is not required to be a power of 2.	RO

Table 4-15. Example AFU MEM Space Contents

<table>
<tr><td colspan="4">Contents</td></tr>
<tr><td>Example 1</td><td>Example 2</td><td>Example 3</td><td>Example 4</td></tr>
<tr><td>Power of 2 System Memory only

(System Memory Length = MEM Space Size)</td><td>Non-Power of 2 System Memory only
(0 < System Memory Length < MEM Space Size)</td><td>System Memory + AFU Special Purpose Memory
(0 < System Memory Length < MEM Space Size)</td><td>AFU Special Purpose Memory only
(System Memory Length = 0)</td></tr>
<tr><td colspan="4">Address: MEM Start Address High || MEM Start Address Low</td></tr>
<tr><td rowspan="3">General Purpose System Memory</td><td colspan="2">General Purpose System Memory</td><td rowspan="2">Address: (MEM Start Address High || MEM Start Address Low) +
(System Memory Length High || System Memory Length Low)</td></tr>
<tr><td colspan="2">Address: (MEM Start Address High || MEM Start Address Low) +
(System Memory Length High || System Memory Length Low)</td></tr>
<tr><td>(empty address space)</td><td colspan="2">AFU Special Purpose Memory
Specified by the AFU in the AFU unique protocol</td></tr>
<tr><td colspan="4">Address: (MEM Start Address High || MEM Start Address Low) + MEM Space size</td></tr>
</table>

Table 4-16. Example MMIO BAR space contents

AFU	Offset	Description
	...	
AFU w	Global MMIO Offset	Global MMIO Registers
	Global MMIO Offset + Global MMIO Size - 1	
	...	
	Per PASID MMIO Offset	PASID 0 Registers
	Per PASID MMIO Offset + Per PASID MMIO Stride - 1	
	Per PASID MMIO Offset + Per PASID MMIO Stride	PASID 1 Registers
	Per PASID MMIO Offset + (2 * Per PASID MMIO Stride) - 1	
	...	
	Per PASID MMIO Offset + (n* Per PASID MMIO Stride)	PASID n Registers
	Per PASID MMIO Offset + ((n+1) * Per PASID MMIO Stride) - 1	
	...	
AFU x	Global MMIO Offset	Global MMIO Registers
	Global MMIO Offset + Global MMIO Size - 1	
	...	
	Per PASID MMIO Offset	PASID 0 Registers
	Per PASID MMIO Offset + Per PASID MMIO Stride - 1	
	Per PASID MMIO Offset + Per PASID MMIO Stride	PASID 1 Registers
	Per PASID MMIO Offset + (2 * Per PASID MMIO Stride) - 1	
	...	
	Per PASID MMIO Offset + (n* Per PASID MMIO Stride)	PASID n Registers
	Per PASID MMIO Offset + ((n+1) * Per PASID MMIO Stride) - 1	
	...	
AFU y	Global MMIO Offset	Global MMIO Registers
	Global MMIO Offset + Global MMIO Size - 1	
	...	
	Per PASID MMIO Offset	PASID 0 Registers
	Per PASID MMIO Offset + Per PASID MMIO Stride - 1	
	Per PASID MMIO Offset + Per PASID MMIO Stride	PASID 1 Registers
	Per PASID MMIO Offset + (2 * Per PASID MMIO Stride) - 1	
	...	
	Per PASID MMIO Offset + (n* Per PASID MMIO Stride)	PASID n Registers
	Per PASID MMIO Offset + ((n+1) * Per PASID MMIO Stride) - 1	
	...	

4.3.4 AFU control DVSEC

This capability is a means to assign common, general information for an AFU associated with a Function, independent of the specific functionality that AFU provides. It is also a means to control an individual AFU without affecting other AFUs associated with the same Function.

There is separate AFU control DVSEC for each AFU associated with a Function. The AFU is identified by the AFU Control Index field in the DVSEC. The absence of an AFU control DVSEC for a specific AFU Control Index signifies that an AFU for that specific index does not exist.

Table 4-17. AFU control DVSEC format

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	Offset
Next Capability Offset	Capability Version	Extended Capability ID		x‘00’
DVSEC Length	DVSEC Rev	DVSEC PCI Vendor ID		x‘04’
Reserved	AFU Control Index	DVSEC ID		x‘08’
AFU Unique (24:21)	Rsvd (20:19)	F (18)	E (17)	R (16)	Rsvd (15:14)	T (13)	PASID Termination (12:0)		x‘0C’
Reserved			PASID Len Enbl (11:10)	Rsvd (9:8)	PASID Len Sup (7:0)			x‘10’
MS (31)	ME (30)	HTRL (29:27)	ES (26)	EE (25)	Reserved (24:16)	PASID Base (15:0)			x‘14’
Reserved (31:29)	AFU acTag Length Enabled (28:16)	Reserved (15:13)	AFU acTag Length Supported (12:0)						x‘18’
Reserved			acTag Base						x‘1C’

Table 4-18. AFU control DVSEC (Page 1 of 2)

Offset	Bits	Field Name	Description	Attributes
x‘00’	31:20	Next Capability Offset	Offset of next Extended Capability.	RO
	19:16	Capability Version	Version = x’1’.	RO
	15:0	Extended Capability ID	DVSEC = x‘0023’.	RO
x‘04’	31:20	DVSEC Length	Length of this capability beginning at offset x’00’.	RO
	19:16	DVSEC Revision	DVSEC structure revision level = x’0’.	RO
	15:0	DVSEC PCI Vendor ID	DVSEC PCI Vendor ID = x’1014’.	RO
x‘08’	31:22	Reserved	Reserved = ‘0000_0000_00’.	RO
	21:16	AFU Control Index	AFU associated with the fields in this Extended Capability.	RO
	15:0	DVSEC ID	DVSEC ID = x‘F004’.	RO
x’0C’	31:28	AFU Unique	These bits are defined by each specific AFU implementation.	RW
	27:26	Reserved	Reserved = ‘00’.	RO
	25	Fence AFU	Block all electrical signals between the Function and the AFU. 0: Disable Fence between Function and AFU. 1: Enable Fence between Function and AFU.	RW
	24	Enable AFU	Allow the AFU to send Commands and Data on the OpenCAPI link. Does not affect receiving Commands or sending Responses. 0: Disable AFU. 1: Enable AFU.	RW
	23	Reset AFU	1 = Reset the AFU identified by the AFU Index.	W
	22:21	Reserved	Reserved = ‘00’.	RO
	20	Terminate Valid	PASID Termination Valid. Set to ‘1’ to direct AFU to terminate PASID in PASID Termination Value field. Software must first poll to ensure this bit is ‘0’ before setting to ‘1’ along with PASID Termination Value.	RW
	19:0	PASID Termination Value	PASID to terminate.	RW
x‘10’	31:16	Reserved	Reserved = x‘0000’.	RO
	15:13	Reserved	Reserved = ‘000’.	RO
	12:8	PASID Length Enabled	Number of consecutive PASID’s this AFU is allowed to use = $2^{\text{PASID Length Enabled}}$ Valid PASID range = PASID Base -> PASID Base + $(2^{\text{PASID Length Enabled}} - 1)$.	RW
	7:5	Reserved	Reserved = ‘000’	RO
	4:0	PASID Length Supported	Number of consecutive PASID’s this AFU supports = $2^{\text{PASID Length Supported}}$	RO

Table 4-18. AFU control DVSEC (Page 2 of 2)

Offset	Bits	Field Name	Description	Attributes
x‘14’	31	MS	[OpenCAPI 3.1] MetaData Supported. See *OpenCAPI Transaction Layer Specification*. 0 = MetaData is not Supported 1 = MetaData is Supported	RO
	30	ME	[OpenCAPI 3.1] MetaData Enabled. 0 = MetaData is not Enabled 1 = MetaData is Enabled	RW
	29:27	HTRL	[OpenCAPI 4.0] host_tag run length maximum host_tag run length = 2^{HTRL} **Note:** The AFU can use or ignore this field but must not violate the maximum specified by the host. When not used, the AFU is limited in its ability to issue unsolicited cast out commands with a dLength greater than 64 bytes. When a cast out is due to a `force_evict` the run length is implied by the dLength field used by the `force_evict` command and this can be used by the AFU.	RW
	26	ES	[OpenCAPI 3.1] Extended MetaData Supported. See *OpenCAPI Transaction Layer Specification*. 0 = Extended MetaData is not Supported 1 = Extended MetaData is Supported	RO
	25	EE	[OpenCAPI 3.1] Extended MetaData Enabled. 0 = Extended MetaData is not Enabled 1 = Extended MetaData is Enabled	RW
	24:20	Reserved	Reserved =’0_0000’.	RO
	19:0	PASID Base	First PASID this AFU is allowed to use.	RW
x‘18’	31:28	Reserved	Reserved =’0000’.	RO
	27:16	AFU acTag Length Enabled	Number of consecutive acTags this AFU is allowed to use. x‘000’: no acTags are valid x‘001’ - x‘FFF’ : Valid AFU acTag range = AFU acTag Base → (AFU acTag Base + AFU acTag Length Enabled - 1).	RW
	15:12	Reserved	Reserved = ‘0000’.	RO
	11:0	AFU acTag Length Supported	Number of consecutive acTags this AFU supports	RO
x‘1C’	31:12	Reserved	Reserved = x‘0000_0’.	RO
	11:0	AFU acTag Base	First acTag this AFU is allowed to use.	RW

4.3.5 Vendor-specific DVSEC

Table 4-19. Vendor-specific DVSEC format

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	Offset
Next Capability Offset	Capability Version	Extended Capability ID		x‘00’
DVSEC Length	DVSEC Rev	DVSEC PCI Vendor ID		x‘04’
Vendor Unique		DVSEC ID		x‘08’
Vendor Unique				x‘0C’ - x‘nn’

Table 4-20. Vendor-specific DVSEC

Offset	Bits	Field Name	Description	Attributes
x‘00’	31:20	Next Capability Offset	Offset of next Extended Capability.	RO
	19:16	Capability Version	Version = x‘1’.	RO
	15:0	Extended Capability ID	DVSEC = x‘0023’.	RO
x‘04’	31:20	DVSEC Length	Length of this capability beginning at offset x’00’.	RO
	19:16	DVSEC Revision	Vendor Defined.	RO
	15:0	DVSEC PCI Vendor ID	PCI Vendor ID for this implementation.	RO
x‘08’	31:16	Vendor Unique	Vendor Defined.	VU
	15:0	DVSEC ID	Vendor Defined from Vendor Specific usage range.	RO
x‘0C’ - x‘nn’	31:0	Vendor Unique	Vendor Defined.	VU