Universal Serial Bus Type-C Bridge Class Specification

Revision 1.1
October 10, 2017

INTELLECTUAL PROPERTY DISCLAIMER

THIS SPECIFICATION IS PROVIDED TO YOU "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE. THE AUTHORS OF THIS SPECIFICATION DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY PROPRIETARY RIGHTS, RELATING TO USE OR IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. THE PROVISION OF THIS SPECIFICATION TO YOU DOES NOT PROVIDE YOU WITH ANY LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS.

Please send comments via electronic mail to techsup@usb.org

For industry information, refer to the USB Implementers Forum web page at http://www.usb.org

CONTENTS

FIGURES

TABLES

Specification Work Group Contributors

AMD	Jason Hawken		
Apple	Colin Whitby-Strevens (Editor)	Dave Conroy	Robert Walsh
Cypress	Subramanyam Sankaran		
Displaylink	Dan Ellis	Richard Petrie	
Fresco Logic	Bob McVay		
Google	David Schneider		
Intel	Abdul (Rahman) Ismail (Chair/Editor)	Brad Saunders	Bob Dunstan
	Dave Hines	Sophia Kuo	Stephanie Wallick (Editor)
	Tin Cheung (Willis) Kung		
Microchip	Mark Bohm		
Microsoft	Anthony Chen	Rahul Ramadas	Vivek Gupta
NXP	Vijendra Kuroodi		
Renesas	Kiichi Muto	Philip Leung	
Specwerkz	Amanda Hosler	Diane Lenox	Søren Petersen
Via	Terrance Shih		

1 Introduction

1.1 Scope

The USB Type-C Bridge specification describes the following:

- A methodology for a USB Host to communicate with PD Products attached to Connectors of a PDUSB Hub or attached to the Connector on a charge-through Alternate Mode Adapter.
- An interface to expose and configure the USB Type-C capabilities of Connectors on USB Hubs or Alternate Mode Adapters.
- The ability to expose the Billboard functionality.

This specification is intended for USB hub developers and device driver (System Software) developers. The term System Software is used to describe the software running on the PDUSB Host. Where possible the component of the System Software is called out, for example, the OS Policy Manager (OSPM) or the OS Authentication Manager (OSAM).

The reader is expected to be familiar with [USBTYPEC], [USBPD], [USBAUTH], [USBPDFU] and [USBBB]. There may exist conflicts between this specification and one or more of the above mentioned specifications. In such cases [USBTYPEC], [USBPD], [USBAUTH], [USBPDFU] and [USBBB] take precedence.

1.2 Purpose

The purpose of this specification is to describe the data structures, requests and notification mechanisms that a USB Type-C Bridge is required to support. These data structures, requests and notifications are used in communications between the PDUSB Host and the USB Type-C Bridge. They are used by the PDUSB Host to send requests to, and retrieve capabilities and status from the USB Type-C Bridge.

1.3 Related Documents

- [USB2.0] – Universal Serial Bus Specification, Revision 2.0, (including errata and ECNs through June 27, 2017) (referred to in this document as the USB 2.0 Specification) (available at: http://www.usb.org/developers/docs).
- [USB3.2] – Universal Serial Bus 3.2 Specification, Revision 1.0, September 22, 2017 (available at: http://www.usb.org/developers/docs).
- [USBPD] – USB Power Delivery Specification Rev. 3.0, Version 1.1, January 12, 2017 (including errata and ECNs through June 12, 2017) (referred to in this document as the USB PD Specification) (available at: http://www.usb.org/developers/docs).
- [USBTYPEC] – USB Type-C™ Cable and Connector Specification Revision 1.3, July 14, 2017 (available at: http://www.usb.org/developers/docs)
- [USBBB] – USB Billboard Specification Revision 1.21, September 8, 2016(referred to in this document as the USB Billboard Specification) (available at: http://www.usb.org/developers/docs).
- [USBAUTH] – USB Type-C Authentication Specification Revision 1.0 with ECN and Errata through February 2, 2017(available at: http://www.usb.org/developers/docs).
- [USBPDFU] – USB Power Delivery Firmware Update Specification Revision 1.0, September 15, 2016 (available at: http://www.usb.org/developers/powerdelivery/)
- [UCSI] – USB Type-C Connector System Software Interface Requirements Specification Revision 1.1, August 2017 (available at: https://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/usb-type-c-ucsi-spec.pdf)

1.4 Terms and Abbreviations

This section defines terms and abbreviations used throughout this document. For additional terms and abbreviations that pertain to the Universal Serial Bus, see Chapter 2, "Terms and Abbreviations," in [USB2.0] and [USB3.2], Section 1.5 in [USBTYPEC], Section 1.6 in [USBPD] and Section 1.4 in [USBBB], and [USBAUTH].

Table 1-1: Terms and Abbreviations

Term	Description
Asynchronous Event	An event that occurs as the result of a condition occurring in or initiated by the PPM.
Billboard Only USB Type-C Bridge	A USB Type-C Bridge in a Device Container that only supports the Billboard functionality and has no Connectors.
Connector	A downstream or charge-through USB Type-C receptacle.
Device Container	A group of one or more USB functions originating from the same physical device.
Device Policy Manager (DPM)	Module running in a Source or Sink that applies Local Policy to each Connector on a Device Container via the Policy Engine. See [USBPD] for more information.
Operating System Authentication Manager (OSAM)	Operating Software that interfaces with the PAM via the USB Type-C Bridge.
Operating System Firmware Update Manager (OSFUM)	Operating Software that interfaces with the PFUM via the USB Type-C Bridge.
Operating System Policy Manager (OSPM)	Operating Software that interfaces with the PPM via the USB Type-C Bridge. The OSPM includes the System Policy Manager (SPM) as defined in [USBPD].
PDO/PDOS	Power Data Object/Power Data Objects. See [USBPD] for definition.
PD Product	Source, Sink, or Cable Plug as defined in [USBPD].
Platform Authentication Manager (PAM)	Hardware/firmware that manages all the USB Type-C Authentication functionality. See [USBAUTH] for more information.
Platform Firmware Update Manager (PFUM)	Hardware/firmware that manages all the USB Type-C Firmware Update functionality for devices attached to Connectors on a Device Container. See [USBPDFU] for more information.
Platform Policy Manager (PPM)	Hardware/firmware that manages all the Connectors on a Device Container.
PPM Asynchronous Notification	A notification that is generated by the PPM as a result of an Asynchronous Event.
PPM Command Response	A notification that is generated by the PPM as a response to a PPM Command.

1.5 Conventions and Notations

1.5.1 Precedence

If there is a conflict between text, figures, and tables, the precedence shall be tables, figures, and then text.

1.5.2 Keywords

The following keywords differentiate between the levels of requirements and options.

1.5.2.1 Informative

Informative is a keyword that describes information with this specification that intends to discuss and clarify requirements and features as opposed to mandating them.

1.5.2.2 May

May is a keyword that indicates a choice with no implied preference.

1.5.2.3 N/A

N/A is a keyword that indicates that a field or value is not applicable and has no defined value and shall not be checked or used by the recipient.

1.5.2.4 Normative

Normative is a keyword that describes features that are mandated by this specification.

1.5.2.5 Optional

Optional is a keyword that describes features not mandated by this specification. However, if an optional feature is implemented, the feature shall be implemented as defined by this specification (optional normative).

1.5.2.6 Reserved

Reserved is a keyword indicating reserved bits, bytes, words, fields, and code values that are set-aside for future standardization. Their use and interpretation may be specified by future extensions to this specification and, unless otherwise stated, shall not be utilized or adapted by vendor implementation. A reserved bit, byte, word, or field shall be set to zero by the sender and shall be ignored by the receiver. Reserved field values shall not be sent by the sender and, if received, shall be ignored by the receiver.

1.5.2.7 Shall

Shall is a keyword indicating a mandatory (normative) requirement. Designers are mandated to implement all such requirements to ensure interoperability with other compliant Devices.

1.5.2.8 Should

Should is a keyword indicating flexibility of choice with a preferred alternative. Equivalent to the phrase “it is recommended that”.

1.5.3 Numbering

Numbers that are immediately followed by a lowercase “b” (e.g., 01b) are binary values. Numbers that are immediately followed by an uppercase “B” are byte values. Numbers that are immediately followed by a lowercase “h” (e.g., 3Ah) are hexadecimal values. Numbers not immediately followed by either a “b”, “B”, or “h” are decimal values.

1.5.4 Byte Ordering

All multiple byte fields in this specification are interpreted as and moved over the bus in little-endian order, i.e., LSB to MSB unless otherwise specified.

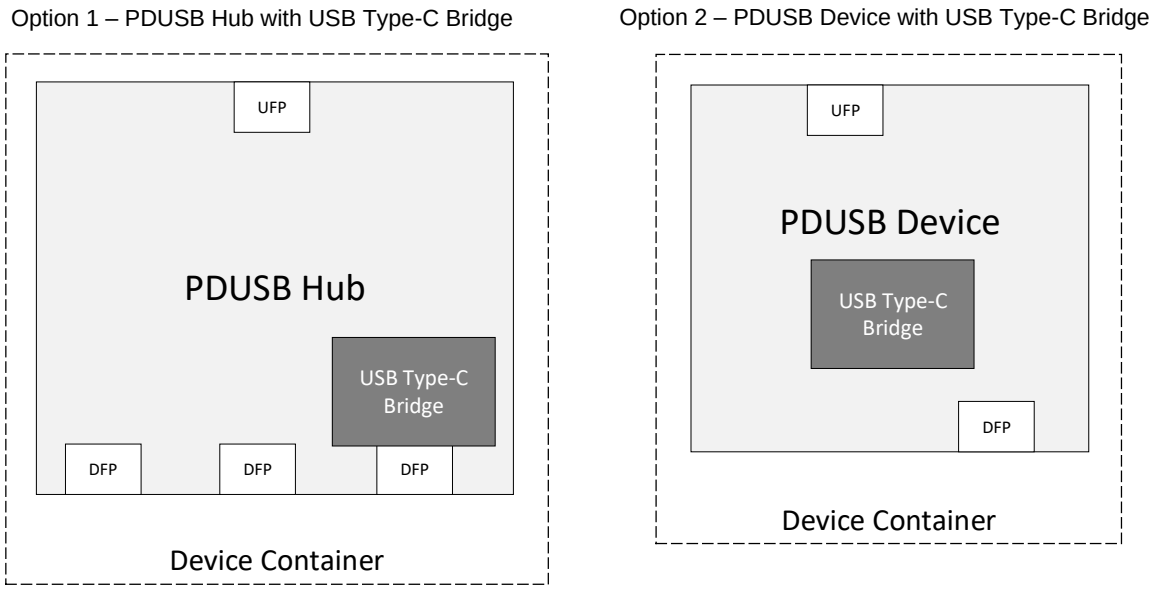
2 Management Overview

The USB Type-C Bridge defined by this specification can only be implemented as part of either:

- An integrated USB 2.0 device behind a PDUSB Hub
- A PDUSB Device with multiple independent Type-C Ports that support USBPD (for example an Alternate Mode Adapter that has two USB Type-C Ports that perform PD negotiations independently of each other, supports charge-through, includes a Connector for connection to a charger, and supports Billboard functionality).

The term “Device Container” is used in this specification to refer to a PDUSB Hub or PDUSB Device that includes a USB Type-C Bridge.

Figure 2-1 USB Type-C Bridge Device Containers



This specification defines the various requests and notifications to enable communication between a USB Host and the Connectors on a Device Container via the USB Type-C Bridge. It details the initialization sequence to determine the number of Connectors that are exposed via the USB Type-C Bridge, and how system software in the USB Host can determine the USB PD and USB Type-C capabilities of both the Device Container and the Connectors on that Device Container that are exposed via the USB Type-C Bridge.

The USB Descriptors and Requests used to communicate between a USB Type-C Bridge Class Driver in a USB Host and a USB Type-C Bridge in a Device Container are specified using a reference architecture, illustrated in Figure 2-2. In this reference architecture, the USB Host contains an OS Policy Manager (OSPM), an OS Authentication Manager (OSAM) and an OS Firmware Update Manager (OSFUM). The Device Container contains a USB Type-C Bridge that contains a Platform Policy Manager (PPM), a Platform Authentication Manager (PAM), a Platform Firmware Update Manager (PFUM) and the USB PD Logic. The USB PD Logic contains the USB PD Device Policy Manager, a PD Policy Engine for each Connector, a PD Protocol Layer for each Connector, and a PD PHY for each Connector.

The OSPM, OSAM and OSFUM communicate with the USB Type-C Bridge Class driver. The PPM, PAM and PFUM communicate with the Device Policy Manager that in turn manages the transmission and reception of appropriate messages on the Connector via the Connector's PD Policy Engine and, indirectly, the Connector's PD Protocol Layer and PD PHY.

The USB Descriptors and Requests used to communicate between a USB Type-C Bridge Class Driver and a USB Type-C Bridge are defined in Section 3. The PPM Commands, PPM Command Responses and PPM Asynchronous Notifications are defined in Section 4. The Operational Model of the USB Type-C Bridge is defined Section 5. This specification does not define or require any particular System Software policies.

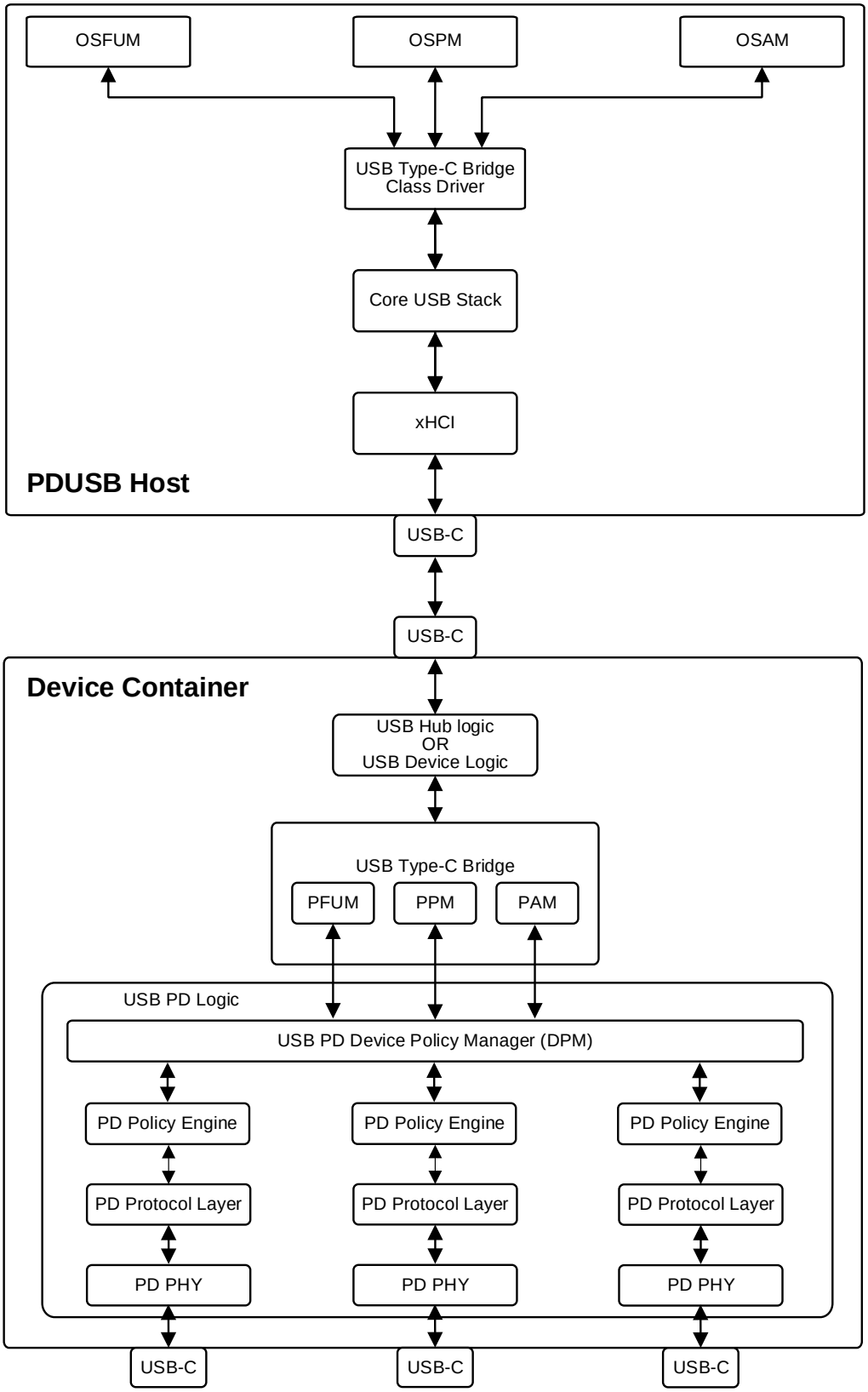
Figure 2-2 depicts:

- The relationship between the OS Policy Manager (OSPM) and Platform Policy Manager (PPM) in the USB Type-C Bridge, and in turn, the relationship between the PPM and the USB PD Device Policy Manager (DPM) defined in [USBPD].

- The relationship between the OS Authentication Manager (OSAM) and the Platform Authentication Manager (PAM), and, in turn, the relationship between the PAM and the USB PD Device Policy Manager (DPM). The PAM in the Device Container acts as a pass-thru for USB PD Authentication commands, allowing the OSAM in the PDUSB Host to exchange USB PD Authentication commands with a PD Product attached to a Connector on the Device Container.

- The relationship between the OS Firmware Update Manager (OSFUM) and the Platform Firmware Update Manager (PFUM), and, in turn, the relationship between the PFUM and the USB PD Device Policy Manager (DPM). The PFUM in the Device Container acts as a pass-thru for USB PD Firmware Update commands, allowing the OSFUM in the PDUSB Host to exchange USB PD Firmware Update commands with a PD Product attached to a Connector on the Device Container.

Figure 2-2 USB Type-C Bridge Reference Architecture



3 USB Descriptors and Requests

This section defines the data structures, descriptors and requests used to communicate between a USB Type-C Bridge Class Driver in a PDUSB Host and a USB Type-C Bridge in a Device Container.

3.1 Descriptors

A USB Type-C Bridge shall return all the descriptors and respond to all standard USB commands as defined in Chapter 9 of [USB2.0] or Chapter 9 of [USB3.2]. USB Type-C Bridge function shall be encapsulated in a single interface. The interface shall not be part of any other function. The interface consists of:

- One Interrupt IN endpoint (Notification endpoint).
- A class-specific Bridge Capability descriptor that shall immediately follow the Interface descriptor in the list of descriptors returned by the USB Type-C Bridge.

A USB Type-C Bridge shall return the Class, Subclass and Protocol values as defined in Appendix A in its Interface descriptor.

The Bridge Capability descriptor shall have the structure defined in Table 3-1.

Table 3-1: Bridge Capability Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this Descriptor in Bytes (7)
1	*bDescriptorType*	1	Constant	BRIDGE_CAPABILITY Descriptor Type
2	*bcdVersion*	2	BCD	Version of the USB Type-C Bridge Specification that the USB Type-C Bridge is compliant to. Shall be set to 0100h.
4	*bmConnectorMask*	2	Bitmap	This field indicates which Connectors on the Device Container the USB Type-C Bridge can communicate with. If a bit is set to one, then the USB Type-C Bridge shall be able communicate with that Connector. The maximum number of bits set to one in this field shall be equal to the value retuned in the **bNumConnectors** field in response to a Get Capability (See Section 4.2.7) PPM Command. Bit 0 is Reserved and shall be set to zero.
6	*bmAttributes*	1	Bitmap	**Bit** / **Description** 0 – If this bit is set to one then the USB Type-C Bridge supports PPM functionality. 1 – If this bit is set to one then the USB Type-C Bridge supports a PAM functionality. 2 – If this bit is set to one then the USB Type-C Bridge supports PFUM functionality. 3 – If this bit is set to one then the USB Type-C Bridge supports Billboard functionality. 4-7 – Reserved and shall be set to zero.

The Notification endpoint shall have the characteristics defined in Table 3-2.

Table 3-2: Notification Endpoint

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this Descriptor in Bytes (9)
1	*bDescriptorType*	1	Constant	ENDPOINT Descriptor Type
2	*bEndpointAddress*	1	Endpoint	Address of the Endpoint
3	*bmAttributes*	1	Bitmap	03h (Interrupt Endpoint)
4	*wMaxPacketSize*	2	Number	40h
6	*bInterval*	1	Number	04h (1ms Intervals)

The data structure of the notifications that a USB Type-C Bridge sends on the Interrupt IN endpoint shall be as defined in Table 3-3.

Table 3-3: Notification Data Structure

Offset	Field	Size	Value	Description
0	*bNotificationType*	1	Number	Identifies the Notification Type. The various notifications a USB TYPE-C Bridge may send are listed in Appendix A.
1	*bParam1*	1	Number	Notification Type-specific Parameter 1.
2	*bParam2*	1	Number	Notification Type-specific Parameter 2.
3	*bParam3*	1	Number	Notification Type-specific Parameter 3.
4	*Payload*	N	Data	Notification Type-specific Data.

3.2 USB Class-Specific Requests

3.2.1 PPM Functionality

This section defines the following:

- Class-specific USB Requests used by the USB Type-C Bridge Class Driver to encapsulate PPM Commands originated by the OSPM.
- USB Type-C Bridge PPM Notification structure that shall be used to encapsulate the PPM Command Responses and PPM Asynchronous Notifications that a PPM (in the USB Type-C Bridge) may send to an OSPM via the USB Type-C Bridge Class Driver.

Section 4 defines the structure of the PPM Commands and PPM Command Responses/PPM Asynchronous Notifications exchanged between the OSPM and the PPM. A USB Type-C Bridge shall only use the PPM Commands and PPM Command Responses/PPM Asynchronous Notifications as defined in Section 4.

A USB Type-C Bridge that is not a Billboard Only USB Type-C Bridge shall support PPM functionality (see Table 3-1) so as to enable a PDUSB Host to query the capabilities of Connectors on the Device Container. A Billboard Only USB Type-C Bridge shall not support PPM functionality.

3.2.1.1 Send PPM Command Request

The Send PPM Command Request shall be used by the USB Type-C Bridge Class Driver to send PPM Commands to the PPM in a USB Type-C Bridge.

Table 3-4: Send PPM Command Request Fields

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100001B	SEND_PPM_COMMAND	00h	Interface	Varies	Varies

The OSPM sends a PPM Command and any related data to the USB Type-C Bridge Class Driver which shall encapsulate the PPM Command and the related data in the **Data** field of the SEND_PPM_COMMAND Request. The USB Type-C Bridge Class Driver shall set the **wLength** field to the number of bytes in the **Data** field. The lower byte of **wIndex** identifies the interface to which the Request is to be sent. The upper byte of **wIndex** shall be set to 00h.

3.2.1.2 USB Type-C Bridge PPM Notification

The PPM in a USB Type-C Bridge shall use the notification data structure in Table 3-3 to send a PPM Command Response or a PPM Asynchronous Notification back to the OSPM via the USB Type-C Bridge Class Driver. The contents of the PPM Notification are given in Table 3-5.

Table 3-5: USB Type-C Bridge PPM Notification

Offset	Field	Size	Value	Description
0	*bNotificationType*	1	Number	Set to PPM_NOTIFICATION.
1	*bParam1*	1	Number	Shall be set to 00h.
2	*bParam2*	1	Number	Shall be set to 00h.
3	*bParam3*	1	Number	Shall be set to 00h.
4	*Payload*	N	Data	Contains the PPM Command Response or PPM Asynchronous Notification. If there is any additional data associated with the PPM Command Response, that data shall be appended to the end of the PPM Command Response.

3.2.2 PAM Functionality

This section defines the following:

- Class-specific USB Requests used by the USB Type-C Bridge Class Driver to encapsulate Authentication Requests or Authentication Responses originated by the OSAM
- USB Type-C Bridge PAM Notification structure that shall be used to encapsulate the Authentication Notifications/Authentication Command Responses that a PAM (in the USB Type-C Bridge) may send to an OSAM via the USB Type-C Bridge Class Driver

Note: Authentication Requests and Responses are defined in [USBAUTH].

A Billboard Only USB Type-C Bridge shall not support PAM functionality. A USB Type-C Bridge that is not a Billboard Only USB Type-C Bridge may support PAM functionality, but is not required to do so.

3.2.2.1 Send Authentication Data Request

The Send Authentication Data Request shall be used by the USB Type-C Bridge Class Driver to transmit an Authentication Request or Response to a Connector on a Device Container via the USB Type-C Bridge.

Table 3-6: Send Authentication Data Request Fields

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100001B	SEND_PD_DATA	Target | AUTH	Connector Number | Interface	Varies	Varies

The OSAM sends either an Authentication Request or Authentication Response to the USB Type-C Bridge Class Driver, which shall encapsulate it in the **Data** of the SEND_PD_DATA request.

The USB Type-C Bridge Class Driver shall set the low byte of the **wValue** field to AUTH when it is sending Authentication Data. It shall set the Target, in the high byte of **wValue**, to one of SOP (0), SOP' (1) or SOP" (2). It shall set the low byte of **wIndex** to the interface to which the Request is to be sent. It shall set the high byte of **wIndex** to the Connector number of the attached PD Product which is to be authenticated. The USB Type-C Bridge Class Driver shall set the **wLength** field to the number of bytes in the **Data** field.

3.2.2.2 Get Authentication Data Request

The Get Authentication Data Request shall be used by the USB Type-C Bridge Class driver to fetch Authentication-related data from a Connector on a Device Container via the USB Type-C Bridge.

Table 3-7: Get Authentication Data Request Fields

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100001B	GET_PD_DATA	Target | AUTH	Connector Number | Interface	Varies	Varies

The USB Type-C Bridge shall send any received Authentication Request or Authentication Response in the **Data** that is sent in response to the GET_PD_DATA request.

The USB Type-C Bridge Class Driver shall set the low byte of the **wValue** field to AUTH when it is requesting Authentication Data. It shall set the Target, in the high byte of **wValue**, to one of SOP (0), SOP' (1) or SOP" (2). It shall set the low byte of **wIndex** to the interface to which the Request is to be sent. It shall set the high byte of **wIndex** to the Connector number of the attached PD Product which is to be authenticated. The USB Type-C Bridge Class Driver shall set the **wLength** field to the maximum number of bytes the USB Type-C Bridge may return in response to this Request.

3.2.2.3 Authentication Notification

The Authentication Notification is used by the USB Type-C Bridge to indicate to the OSAM that it has either:

- Received Authentication-related data from a PD Product attached to a Connector on the Device Container
- Failed to transmit an Authentication Request or Response to a PD Product attached to a Connector on the Device Container

The USB Type-C Bridge shall use the notification data structure in Table 3-3. The contents of the Authentication Notification shall be as detailed in Table 3-8.

Table 3-8: Authentication Notification

Offset	Field	Size	Value	Description
0	*bNotificationType*	1	Number	Set to PD_NOTIFICATION.
1	*bParam1*	1	Number	Set to the Connector Number.
2	*bParam2*	1	Number	Set to AUTH.
3	*bParam3*	1	Number	Set to either Error (80h), Success (00h), or Not_Supported (01h). A Success value indicates that the Authentication Request/Response was transmitted successfully and there is associated data for the OSAM to retrieve. An Error value indicates that the USB Type-C Bridge encountered an error in transmitting the Authentication Request/Response and there is no data for the OSAM to retrieve. A Not_Supported value indicates that an Authentication Request was transmitted successfully and a Not_Supported PD Message was received in response.
4	*Payload*	0	Data	No payload.

3.2.3 PFUM Functionality

This section defines the following:

- Class-specific USB Requests used by the USB Type-C Bridge Class Driver to encapsulate Firmware Update Requests originated by the OSFUM.
- USB Type-C Bridge PFUM Notification structure that shall be used to encapsulate the Firmware Update Command Responses that a PFUM (in the USB Type-C Bridge) may send to an OSFUM via the USB Type-C Bridge Class Driver.

Note: Firmware Update Requests and Responses are defined in [USBPDFU].

A Billboard Only USB Type-C Bridge shall not support PFUM functionality. A USB Type-C Bridge that is not a Billboard Only USB Type-C Bridge may support PFUM functionality, but is not required to do so.

3.2.3.1 Send Firmware Update Data Request

The Send Firmware Update Data Request shall be used by the USB Type-C Bridge Class Driver to transmit a FW Update Request to a PD Product attached to a Connector on a Device Container via the USB Type-C Bridge.

Table 3-9: Send Firmware Update Data Request Fields

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100001B	SEND_PD_DATA	Target | FWU	Connector Number | Interface	Varies	Varies

The OSFUM sends a Firmware Update Request to the USB Type-C Bridge Class Driver, which shall encapsulate it in the **Data** of the SEND_PD_DATA request.

The USB Type-C Bridge Class Driver shall set the low byte of **wValue** field to FWU when it is sending Firmware Update Data. It shall set the Target in the high byte of **wValue** to one of SOP (0), SOP' (1) or SOP" (2). It shall set the low byte of **wIndex** to the interface to which the Request is to be sent. It shall set the high byte of **wIndex** to the Connector number of the Connector to which the PD Product to be updated is attached. The USB Type-C Bridge Class Driver shall set the **wLength** field to the number of bytes in the **Data** field.

3.2.3.2 Firmware Update Notification

The Firmware Update Notification is used by the USB Type-C Bridge to either:

- Transmit a Firmware Update Response to the OSFUM from a PD Product attached to a Connector on a Device Container.
- Notify the OSFUM that a Firmware Update Request could not be transmitted to a PD Product attached to a Connector on a Device Container.

The USB Type-C Bridge shall use the notification data structure in Table 3-3. The contents of the Firmware Update Notification shall be as defined in Table 3-10.

Table 3-10: Firmware Update Notification

Offset	Field	Size	Value	Description
0	*bNotificationType*	1	Number	Set to PD_NOTIFICATION.
1	*bParam1*	1	Number	Set to the Connector Number.
2	*bParam2*	1	Number	Set to FWU.
3	*bParam3*	1	Number	Set to either Error (80h), Success (00h), or Not_Supported (01h). A Success value indicates that the Firmware Update Request was transmitted successfully and the associated Firmware Update Response is in the *Payload* of the Notification. An Error value indicates that the USB Type-C Bridge encountered an error in transmitting the Firmware Update Request. A Not_Supported value indicates that a Firmware Update Request was transmitted successfully and a Not_Supported PD Message was received in response.
4	*Payload*	N	Data	If *bParam3* = Success, shall contain a Firmware Update Response (see [USBPDFU]) If *bParam3* = Error or Not_Supported, notification shall not contain payload.

3.2.4 Billboard Functionality

This section defines the following:

- Class-specific USB Requests used by the USB Type-C Bridge Class Driver to enable Billboard Notifications from the USB Type-C Bridge.
- USB Type-C Bridge Billboard Notification structure that shall be used to indicate changes to the Billboard Capability to the USB Type-C Bridge Class Driver.

The USB Type-C Bridge shall optionally support the Billboard functionality (see Table 3-1) using the methodology described in this specification.

3.2.4.1 Enable/Disable Billboard Notification Request

The Enable/Disable Billboard Notification Request shall be used by the USB Type-C Bridge Class Driver to enable or disable Billboard Notifications from the USB Type-C Bridge.

Table 3-11: Enable/Disable Billboard Notification Request Fields

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100001B	ENABLE_BB_NOTIFICATION	Enable or Disable	Interface	00h	No data

The USB Type-C Bridge Class Driver shall set the **wValue** field to Enable (1) when it wants to enable Billboard Notifications from the USB Type-C Bridge. It shall set the **wValue** field to Disable (0) when it wants to disable Billboard Notifications from the USB Type-C Bridge. The lower byte of **wIndex** identifies the interface that the Request is sent to. The upper byte of **wIndex** shall be set to 00h.

3.2.4.2 Billboard Notification

The Billboard Notification shall be sent by the USB Type-C Bridge to indicate that there is a change to the Billboard Capability in the Device Container. The USB Type-C Bridge shall use the notification data structure in Table 3-3. The contents of the Billboard Notification shall be as detailed in Table 3-12.

Table 3-12: Billboard Notification

Offset	Field	Size	Value	Description
0	*bNotificationType*	1	Number	Set to BILLBOARD_NOTIFICATION.
1	*bParam1*	1	Number	Shall be set to 00h.
2	*bParam2*	1	Number	Shall be set to 00h.
3	*bParam3*	1	Number	Shall be set to 00h.
4	*Payload*	0	Data	No Payload.

4 PPM Commands, PPM Command Responses and PPM Asynchronous Notifications

This section describes the only data structures and PPM Commands that an OSPM shall use in order to communicate with the PPM in the USB Type-C Bridge and the requirements of the PPM in the USB Type-C Bridge when it receives a PPM Command. It also defines the structure of the PPM Command Responses and the PPM Asynchronous Notification that a PPM in the USB Type-C Bridge shall send to the OSPM in response to a PPM Command or due to an Asynchronous Event.

4.1 Data Structures

4.1.1 COMMAND

The COMMAND Data Structure is sent as the **Data** of a SEND_PPM_COMMAND Request, and defines the PPM Command that shall be executed by the PPM in the USB Type-C Bridge. Depending on the Command Type, some fields in this Data Structure need to be interpreted differently. This section defines the high level structure of both the fields that are static and the fields that change based on the PPM Command. Subsequent sections define the structure of the COMMAND Data Structure for each type of PPM Command.

Table 4-1: COMMAND Data Structure

Offset (Bits)	Field	Size (Bits)	Description
0	*Command*	8	The value in this field determines the Command that OSPM wants the PPM to execute. Bits 16-63 shall contain any parameters specific to this Command.
8	*Data Length*	8	Defined by Command.
16	*Command Specific*	48	The definition of these bits is different for each Command that can be sent to the PPM.

Section 4.2 describes the definition of the **Command Specific** field for each PPM Command that can be sent to a PPM.

4.1.2 PPM Command Response or PPM Asynchronous Notification

When the PPM completes a PPM Command or when it detects an Asynchronous Event on one of its Connectors it shall send a PPM Command Response or a PPM Asynchronous Notification respectively to the OSPM. A PPM Command Response or PPM Asynchronous Notification shall be as detailed in Table 4-2.

Table 4-2: PPM Command Response or PPM Asynchronous Notification Data Structure

Offset (Bits)	Field	Size (Bits)	Description
0	*Reserved*	1	Reserved and shall be set to zero.
1	*Connector Change Indicator*	7	The PPM shall use this field to indicate the Connector Number that a change occurred on. The only valid values in this field shall be 0 to the maximum number of Connectors supported on the platform. Connectors are indexed from 1. If this field is set to zero, then no change occurred on any of the Connectors.

Offset (Bits)	Field	Size (Bits)	Description
8	*Data Length*	8	This field shall indicate the number of additional bytes of data that follow a PPM Command Response. The format and contents of the data is dependent on the PPM Command Response. The value in this field shall be less than or equal to MAX_DATA_LENGTH bytes for a PPM Command Response. This field shall be set to zero if this is a PPM Asynchronous Notification.
16	*Reserved*	9	Reserved and shall be set to zero.
25	*Not Supported Indicator*	1	The PPM shall set this field to one when it wants to indicate that it does not currently support a PPM Command. This field shall only be valid when the **Command Completed Indicator** field is set to one.
26	*Cancel Completed Indicator*	1	The PPM shall set this field to one when it has completed a CANCEL Command. This field shall only be valid when the **Command Completed Indicator** field is set to one.
27	*Reset Completed Indicator*	1	The PPM shall set this field to one when it has completed a PPM_RESET Command. If this field is set to one, then no other bits in this Data Structure shall be set by the PPM. The PPM shall clear this field on reception of the next PPM Command (not PPM_RESET) from the OSPM.
28	*Busy Indicator*	1	This value in this field shall be ignored as the field is not supported in this version of the specification.
29	*Acknowledge Command Indicator*	1	The PPM shall set this field to one when it completes the ACK_PCR_PAN (Acknowledge PPM Command Response and/or PPM Asynchronous Notification) Command. The PPM shall automatically reset this bit when it receives the next PPM Command from the OSPM. If this field is set to one, then the only other field that can be set is the **Connector Change Indicator** field.
30	*Error Indicator*	1	The PPM shall set this field to one when it encounters an error when executing the PPM Command sent to it by the OSPM. This field shall only be valid when the **Command Completed Indicator** field is set to one.
31	*Command Completed Indicator*	1	The PPM shall set this field to one when it wants to indicate that it completed the PPM Command sent to it by the OSPM.

4.2 PPM Commands

4.2.1 Command Codes

Table 4-3 details the only Command Codes that shall be used in PPM Commands as well as Reserved values.

Table 4-3: Command Code

Command	Value
RESERVED (shall not be used)	00h
PPM_RESET	01h
CANCEL	02h
CONNECTOR_RESET	03h
ACK_PCR_PAN	04h
SET_NOTIFICATION_ENABLE	05h
GET_CAPABILITY	06h
GET_CONNECTOR_CAPABILITY	07h
RESERVED (shall not be used)	08h – 09h
GET_PDOS	10h
RESERVED (shall not be used)	11h
GET_CONNECTOR_STATUS	12h
GET_ERROR_STATUS	13h
RESERVED (shall not be used)	14h – FFh

4.2.2 PPM RESET

The PPM_RESET Command shall be used to reset the PPM. It may be sent at any time by the OSPM to the PPM. The format of the COMMAND Data Structure shall be as detailed in Table 4-4.

Table 4-4: PPM_RESET Command

Offset (Bits)	Field	Size (Bits)	Description
0	*Command*	8	This field shall be set to PPM_RESET.
8	*Data Length*	8	Shall be set to 00h.
16	*Reserved*	48	Reserved and shall be set to zero.

On successful completion of the PPM_RESET Command the PPM shall return a PPM Command Response as described in Table 4-5.

Table 4-5: PPM_RESET Response

Offset (Bits)	Field	Size (Bits)	Description
0	*Reserved*	1	Reserved and shall be set to zero.
1	*Connector Change Indicator*	7	Shall be set to zero.

Offset (Bits)	Field	Size (Bits)	Description
8	*Data Length*	8	Shall be set to 0b.
16	*Reserved*	9	Reserved and shall be set to zero.
25	*Not Supported Indicator*	1	Shall be set to 0b.
26	*Cancel Completed Indicator*	1	Shall be set to 0b.
27	*Reset Completed Indicator*	1	Shall be set to 1b. The PPM shall clear this field on reception of the next PPM Command from the OSPM.
28	*Busy Indicator*	1	Shall be set to 0b.
29	*Acknowledge Command Indicator*	1	Shall be set to 0b.
30	*Error Indicator*	1	Shall be set to 0b.
31	*Command Completed Indicator*	1	Shall be set to 0b.

4.2.3 CANCEL

The CANCEL Command shall be used to cancel a PPM Command previously sent to the PPM. The format of the COMMAND Data Structure shall be as detailed in Table 4-6.

Table 4-6: CANCEL Command

Offset (Bits)	Field	Size (Bits)	Description
0	*Command*	8	This field shall be set to CANCEL.
8	*Data Length*	8	Shall be set to 00h.
16	*Reserved*	48	Reserved and shall be set to zero.

On successful completion of the CANCEL Command the PPM shall return a PPM Command Response as described in Table 4-7.

Table 4-7: CANCEL Response

Offset (Bits)	Field	Size (Bits)	Description
0	*Reserved*	1	Reserved and shall be set to zero.
1	*Connector Change Indicator*	7	If an Asynchronous Event occurred on a Connector, then the PPM shall set this field to the Connector Number on which the change occurred.
8	*Data Length*	8	Shall be set to 00h.
16	*Reserved*	9	Reserved and shall be set to zero.
25	*Not Supported Indicator*	1	Shall be set to 0b.

Offset (Bits)	Field	Size (Bits)	Description
26	*Cancel Completed Indicator*	1	Set to 1b
27	*Reset Completed Indicator*	1	Shall be set to 0b.
28	*Busy Indicator*	1	Shall be set to 0b.
29	*Acknowledge Command Indicator*	1	Shall be set to 0b.
30	*Error Indicator*	1	Shall be set to 0b.
31	*Command Completed Indicator*	1	Shall be set to 1b.

4.2.4 Connector Reset

The CONNECTOR_RESET Command is reserved and shall not be used.. It is anticipated that this command will be defined in a future revision of this specification.

4.2.5 Acknowledge PPM Command Response and/or PPM Asynchronous Notification

The ACK_PCR_PAN Command shall be used to acknowledge to the PPM that the OSPM received and processed a PPM Command Response and/or PPM Asynchronous Notification. The OSPM acknowledges either the Command Completion or the Connector Change or both by setting the appropriate fields in the COMMAND Data Structure. The format of the COMMAND Data Structure shall be as detailed in Table 4-8.

Table 4-8: ACK_PCR_PAN Command

Offset (Bits)	Field	Size (Bits)	Description
0	*Command*	8	This field shall be set to ACK_PCR_PAN.
8	*Data Length*	8	Shall be set to 00h.
16	*Connector Change Acknowledge*	1	The OSPM shall this field to a one to acknowledge a connector change that occurred on the Connector indicated by the PPM in the CCI Data Structure.
17	*Command Completed Acknowledge*	1	The OSPM shall set this field to a one to acknowledge that a PPM Command completed.
18	*Reserved*	46	Reserved and shall be set to zero.

On successful completion of the ACK_PCR_PAN Command the PPM shall return a PPM Command Response as described in Table 4-9.

Table 4-9: ACK_PCR_PAN Response

Offset (Bits)	Field	Size (Bits)	Description
0	*Reserved*	1	Reserved and shall be set to zero.
1	*Connector Change Indicator*	7	Shall be set to zero.
8	*Data Length*	8	Shall be set to 00h.
16	*Reserved*	9	Reserved and shall be set to zero.
25	*Not Supported Indicator*	1	Shall be set to 0b.
26	*Cancel Completed Indicator*	1	Shall be set to 0b.
27	*Reset Completed Indicator*	1	Shall be set to 0b.
28	*Busy Indicator*	1	Shall be set to 0b.
29	*Acknowledge Command Indicator*	1	Shall be set to 1b.
30	*Error Indicator*	1	Shall be set to 0b.
31	*Command Completed Indicator*	1	Shall be set to 0b.

4.2.6 Set Notification Enable

The SET_NOTIFICATION_ENABLE Command shall be used to set the list of Events that the PPM may send PPM Notifications about to the OSPM. The OSPM may update the list at any time. The values in the **Notification Enable** field overwrite any notifications enabled/disabled by a prior SET_NOTIFICATION_ENABLE Command.

The format of the COMMAND Data Structure shall be as detailed in Table 4-10.

If the OSPM enables any PPM Asynchronous Notification, it shall also enable the **Command Completed Notification Enable**.

Table 4-10: SET_NOTIFICATION_ENABLE Command

Offset (Bits)	Field	Size (Bits)	Description
0	*Command*	8	This field shall be set to SET_NOTIFICATION_ENABLE.
8	*Data Length*	8	Shall be set to 00h.

<table>
<tr><th>Offset (Bits)</th><th>Field</th><th>Size (Bits)</th><th>Description</th></tr>
<tr><td>16</td><td>Notification Enable</td><td>16</td><td>A bitmap indicting which notifications shall be enabled or disabled
<table>
<tr><th>Bit</th><th>Notification enabled when set to 1</th></tr>
<tr><td>0</td><td>Command Completed</td></tr>
<tr><td>1</td><td>Reserved and shall be set to zero</td></tr>
<tr><td>2</td><td>Power Operation Mode Change</td></tr>
<tr><td>3</td><td>Reserved and shall be set to zero</td></tr>
<tr><td>4</td><td>Reserved and shall be set to zero</td></tr>
<tr><td>5</td><td>Supported Provider Capabilities Change</td></tr>
<tr><td>6</td><td>Negotiated Power Level Change</td></tr>
<tr><td>7</td><td>PD Reset Complete</td></tr>
<tr><td>8</td><td>Reserved and shall be set to zero</td></tr>
<tr><td>9</td><td>Reserved and shall be set to zero</td></tr>
<tr><td>10</td><td>Reserved and shall be set to zero</td></tr>
<tr><td>11</td><td>Connector Partner Change</td></tr>
<tr><td>12</td><td>Power Direction Change</td></tr>
<tr><td>13</td><td>Reserved and shall be set to zero</td></tr>
<tr><td>14</td><td>Connect Change</td></tr>
<tr><td>15</td><td>Error</td></tr>
</table></td></tr>
<tr><td>32</td><td>Reserved</td><td>32</td><td>Reserved and shall be set to zero.</td></tr>
</table>

On successful completion of the SET_NOTIFICATION_ENABLE Command the PPM shall return a PPM Command Response as described in Table 4-11.

Table 4-11: SET_NOTIFICATION_ENABLE Response

Offset (Bits)	Field	Size (Bits)	Description
0	*Reserved*	1	Reserved and shall be set to zero.
1	*Connector Change Indicator*	7	If an Asynchronous Event occurred on a Connector, then the PPM shall set this field to the Connector Number on which the change occurred.
8	*Data Length*	8	Shall be set to 00h.
16	*Reserved*	9	Reserved and shall be set to zero.
25	*Not Supported Indicator*	1	Shall be set to 0b.
26	*Cancel Completed Indicator*	1	Shall be set to 0b.
27	*Reset Completed Indicator*	1	Shall be set to 0b.
28	*Busy Indicator*	1	Shall be set to 0b.

Offset (Bits)	Field	Size (Bits)	Description
29	*Acknowledge Command Indicator*	1	Shall be set to 0b.
30	*Error Indicator*	1	If the PPM Command was not successfully completed the PPM shall set this field to 1b.
31	*Command Completed Indicator*	1	Shall be set to 1b.

4.2.7 Get Capability

The GET_CAPABILITY Command shall be used to get the PPM capabilities. The format of the COMMAND Data Structure shall be as detailed in Table 4-12.

Table 4-12: GET_CAPABILITY Command

Offset (Bits)	Field	Size (Bits)	Description
0	*Command*	8	This field shall be set to GET_CAPABILITY.
8	*Data Length*	8	Shall be set to 00h.
16	*Reserved*	48	Reserved and shall be set to zero.

On successful completion of the GET_CAPABILITY Command the PPM shall return a PPM Command Response as described in Table 4-13.

Table 4-13: GET_CAPABILITY Response

Offset (Bits)	Field	Size (Bits)	Description
0	*Reserved*	1	Reserved and shall be set to zero.
1	*Connector Change Indicator*	7	If an Asynchronous Event occurred on a Connector, then the PPM shall set this field to the Connector Number on which the change occurred.
8	*Data Length*	8	If successful shall be set to 10h else shall be set to 00h.
16	*Reserved*	9	Reserved and shall be set to zero.
25	*Not Supported Indicator*	1	Shall be set to 0b.
26	*Cancel Completed Indicator*	1	Shall be set to 0b.
27	*Reset Completed Indicator*	1	Shall be set to 0b.
28	*Busy Indicator*	1	Shall be set to 0b.
29	*Acknowledge Command Indicator*	1	Shall be set to 0b.

Offset (Bits)	Field	Size (Bits)	Description
30	*Error Indicator*	1	If the command was not successfully completed the PPM shall set this field to 1b.
31	*Command Completed Indicator*	1	Shall be set to 1b.

If the GET_CAPABILITY Command completed successfully then the PPM shall return additional Data following the GET_CAPABILITY Response as described in Table 4-14.

Table 4-14: GET_CAPABILITY Data

Offset (Bits)	Field	Size (Bits)	Description
0	*bmAttributes*	32	Bitmap encoding of supported PPM features. See Table 4-15 for a description of each bit.
32	*bNumConnectors*	7	This field indicates the number of Connectors that this PPM supports. This field shall not be set to zero..
39	*Reserved*	1	Reserved and shall be set to zero.
40	*bmOptionalFeatures*	24	No optional features are supported in this version of the specification. This field shall be set to zero.
64	*bNumAltModes*	8	This value in this field shall be ignored as the field is not supported in this version of the specification. This field shall be set to zero.
72	*Reserved*	8	Reserved and shall be set to zero.
80	*bcdBCVersion*	16	Battery Charging Specification Release Number in Binary-Coded Decimal (e.g., V1.20 is 120H). This field shall only be valid if the device indicates that it supports BC in the *bmAttributes* field otherwise the field shall be set to zero.
96	*bcdPDVersion*	16	USB Power Delivery Specification Revision Number in Binary-Coded Decimal (e.g. 0200h for Revision 2.0, 0300h for Revision 3.0). This field shall only be valid if the device indicates that it supports PD in the *bmAttributes* field otherwise the field shall be set to zero.
112	*bcdUSBTypeCVersion*	16	USB Type-C Specification Revision Number in Binary-Coded Decimal (e.g. 0120h for Revision 1.2).

Table 4-15: bmAttributes Field Description

Bit	Description
0	Disabled State Support This bit shall be set to one to indicate this platform supports the Disabled State as defined in Section 4.5.2.2.1 in the [USBTYPEC].
1	Battery Charging This bit shall be set to one to indicate this platform supports the Battery Charging Specification as per the value reported in the **bcdBCVersion** field.
2	USB Power Delivery This bit shall be set to one to indicate this platform supports the USB Power Delivery Specification as per the value reported in the **bcdPDVersion** field.
7:3	Reserved and shall be set to zero.
15:8	bmPowerSource At least one of the following bits 8, 10 and 14 shall be set to indicate which power sources are supported.
31:16	Reserved and shall be set to zero.

Bit	Description
8	AC Supply
9	Reserved and shall be set to zero.
10	Other
13:11	Reserved and shall be set to zero.
14	Uses VBUS
15	Charge-through Connector

4.2.8 Get Connector Capability

The GET_CONNECTOR_CAPABILITY Command is used to get the capabilities of a Connector. The format of the COMMAND Data Structure shall be as detailed in Table 4-16.

Table 4-16: GET_CONNECTOR_CAPABILITY Command

Offset (Bits)	Field	Size (Bits)	Description
0	*Command*	8	This field shall be set to GET_CONNECTOR_CAPABILITY.
8	*Data Length*	8	Shall be set to 00h.
16	*Connector Number*	7	This field indicates the Connector whose capabilities are to be retrieved. This field shall not be set to zero.
23	*Reserved*	41	Reserved and shall be set to zero.

On successful completion of the GET_CONNECTOR_CAPABILITY Command the PPM shall return a PPM Command Response as described in Table 4-17.

Table 4-17: GET_CONNECTOR_CAPABILITY Response

Offset (Bits)	Field	Size (Bits)	Description
0	*Reserved*	1	Reserved and shall be set to zero.
1	*Connector Change Indicator*	7	If an Asynchronous Event occurred on a Connector, then the PPM shall set this field to the Connector Number on which the change occurred.
8	*Data Length*	8	If successful shall be set to 02h else shall be set to 00h.
16	*Reserved*	9	Reserved and shall be set to zero.
25	*Not Supported Indicator*	1	Shall be set to 0b.
26	*Cancel Completed Indicator*	1	Shall be set to 0b.
27	*Reset Completed Indicator*	1	Shall be set to 0b.
28	*Busy Indicator*	1	Shall be set to 0b.
29	*Acknowledge Command Indicator*	1	Shall be set to 0b.
30	*Error Indicator*	1	If the PPM Command was not successfully completed the PPM shall set this field to 1b.
31	*Command Completed Indicator*	1	Shall be set to 1b.

If the GET_CONNECTOR_CAPABILITY Command completed successfully then the PPM shall return additional Data following the GET_CONNECTOR_CAPABILITY Response as described in Table 4-18.

Table 4-18: GET_CONNECTOR_CAPABILITY Data

Offset (Bits)	Field	Size (Bits)	Description
0	*Operation Mode Capability*	8	If the Connector supports operation in a particular mode, the PPM shall set the bit to a one, else it shall set the bit to a zero.

Bit	Meaning
0	Supports Rp Only
1	Supports Rd Only
2	Supports (Rp/Rd)
3	Supports Analog Audio Accessory Mode
4	Supports Debug Accessory Mode
5	Supports USB 2.0
6	Supports USB 3.2
7	Supports Alternate Mode

Offset (Bits)	Field	Size (Bits)	Description
8	*Provider*	1	This bit is only valid when **Operation Mode Capability** field has bits 0 or 2 set to 1b. This bit shall be set to one if the Connector is capable of providing power on this Connector (either BC 1.2, USB Type-C Current or PD) otherwise the bit shall be set to zero.
9	*Consumer*	1	This bit is only valid when **Operation Mode Capability** field has bits 1 or 2 set to 1b. This bit shall be set to one if the Connector is capable of consuming power on this Connector (either BC 1.2, USB Type-C Current or PD) otherwise the bit shall be set to zero.
10	*Swap to DFP*	1	This bit is only valid when **Operation Mode Capability** field has bits 0, 1, or 2 set to 1b. This bit shall be set to one if the connector is capable of accepting swap to DFP.
11	*Swap to UFP*	1	This bit is only valid when **Operation Mode Capability** field has bits 0, 1, or 2 set to 1b. This bit shall be set to one if the connector is capable of accepting swap to UFP.
12	*Swap to SRC*	1	This bit is only valid when **Operation Mode Capability** field has bit 2 set to 1b. This bit shall be set to one if the connector is capable of accepting swap to SRC.
13	*Swap to SNK*	1	This bit is only valid when **Operation Mode Capability** field has bit 2 set to 1b. This bit shall be set to one if the connector is capable of accepting swap to SNK.
14	*Reserved*	2	Reserved and shall be set to zero.

4.2.9 Get PDOs

This command is used to get the Sink or Source PDOs associated with the connector identified with the command. For the connector, this command can be used to get the Source PDOs/Capabilities as defined below:

- Maximum Supported Source Capabilities
- The Maximum Provider Capabilities that the Source can support. These do not change for a connector.
- Current Supported Source capabilities
- The Provider Capabilities that the Source currently supports. These can change dynamically and could be lower than the Maximum Source Capabilities if the system is Reaching Power Budget Limit due to multiple connected Sinks or if the Power Budget has been lowered due to it being unplugged from external power supply.
- Advertised Source Capabilities
- The Provider Capabilities that are advertised by the Source during PD contract negotiation. These could be lower due to the Cable's current carrying capabilities. This is only valid when a port partner is present.

In addition, this command can be used to return the Sink or Source PDOs of the device that is connected to this connector. The format of the CONTROL Data Structure for this command is given in Table 4-19.

Table 4-19: GET_PDOS Command

Offset (Bits)	Field	Size (Bits)	Description
0	*Command*	8	This field shall be set to GET_PDOS.
8	*Data Length*	8	Set to 0x00.
16	*Connector Number*	7	This field shall be set to the connector being queried.
23	*Partner PDO*	1	This field shall be set to one if the OSPM wants to retrieve the PDOS of the device attached to the connector.
24	*PDO Offset*	8	Starting offset of the first PDO to be returned.
32	*Reserved*	2	Reserved and shall be set to zero.
34	*Source or Sink PDOs*	1	This field shall be set to one if the OSPM wants to retrieve the Source PDOs otherwise it wants to retrieve the Sink PDOs.
35	*Source Capabilities Type*	2	This field indicates the type of Source Capabilities requested. This field is valid only if OSPM sets Partner PDO to 0 and Source or Sink PDOs to 1. **Value** / **Meaning** 0 – Current Supported Source Capabilities 1 – Advertised Capabilities 2 – Maximum Supported Source Capabilities 3 – Not Used
37	*Number of PDOs*	8	Number of PDOs to return starting from the PDO Offset. The number of PDOs to return is the value in this field plus 1.
45	*Reserved*	19	Reserved and shall be set to zero.

On successful completion of the GET_PDOS Command the PPM shall return a PPM Command Response as described in Table 4-20.

Table 4-20: GET_PDOS Response

Offset (Bits)	Field	Size (Bits)	Description
0	*Reserved*	1	Reserved and shall be set to zero.
1	*Connector Change Indicator*	7	If an asynchronous event occurred on a connector then the PPM shall set this field to the connector number on which the change occurred.
8	*Data Length*	8	If successful set to four times the number of PDOs returned up to a maximum of MAX_DATA_LENGTH bytes. Else set to 0x00.
16	*Reserved*	9	Reserved and shall be set to zero.
25	*Not Supported Indicator*	1	Set to 0b.
26	*Cancel Completed Indicator*	1	Set to 0b.
27	*Reset Completed Indicator*	1	Set to 0b.
28	*Busy Indicator*	1	This value in this field shall be ignored as the field is not supported in this version of the specification.
29	*Acknowledge Command Indicator*	1	Set to 0b.
30	*Error Indicator*	1	If the command was not successfully completed the PPM shall set this field to 1b.
31	*Command Completed Indicator*	1	Set this field to a 1b.

If the GET_PDOS Command completed successfully then the PPM shall return additional Data following the GET_PDOS Response as described in Table 4-21.

Table 4-21: GET_PDO Data

Offset (Bits)	Field	Size (Bits)	Description
0	*PDO[0]*	32	First PDO at PDO Offset
32	*PDO[1]*	32	Second PDO (if present)
64	…	varies	Third through second to last PDOs (if present)
32 * n	*PDO[n]*	32	Last PDO (if present)

4.2.10 Get Connector Status

This GET_CONNECTOR_STATUS Command shall be used to get the current status of a Connector. The format of the COMMAND Data Structure shall be as detailed in Table 4-22.

Table 4-22: GET_CONNECTOR_STATUS Command

Offset (Bits)	Field	Size (Bits)	Description
0	*Command*	8	This field shall be set to GET_CONNECTOR_STATUS.
8	*Data Length*	8	Shall be set to 00h.
16	*Connector Number*	7	This field indicates the Connector whose status is being queried. This field shall not be set to zero.
23	*Reserved*	41	Reserved and shall be set to zero.

On successful completion of the GET_CONNECTOR_STATUS Command the PPM shall return a PPM Command Response as described in Table 4-23.

Table 4-23: GET_CONNECTOR_STATUS Response

Offset (Bits)	Field	Size (Bits)	Description
0	*Reserved*	1	Reserved and shall be set to zero.
1	*Connector Change Indicator*	7	If an Asynchronous Event occurred on a Connector, then the PPM shall set this field to the Connector Number on which the change occurred.
8	*Data Length*	8	If successful shall be set to 09h else shall be set to 00h.
16	*Reserved*	9	Reserved and shall be set to zero.
25	*Not Supported Indicator*	1	Shall be set to 0b.
26	*Cancel Completed Indicator*	1	Shall be set to 0b.
27	*Reset Completed Indicator*	1	Shall be set to 0b.
28	*Busy Indicator*	1	Shall be set to 0b.
29	*Acknowledge Command Indicator*	1	Shall be set to 0b.
30	*Error Indicator*	1	If the PPM Command was not successfully completed the PPM shall set this field to 1b.
31	*Command Completed Indicator*	1	Shall be set to 1b.

If the GET_CONNECTOR_STATUS Command completed successfully then the PPM shall return additional Data following the GET_CONNECTOR_STATUS Response as described in Table 4-24.

Table 4-24: GET_CONNECTOR_STATUS Data

Offset (Bits)	Field	Size (Bits)	Description
0	*Connector Status Change*	16	A bitmap indicating the types of status changes that have occurred on the Connector. See Table 4-25 for a description of each bit.
16	*Power Operation Mode*	3	This field shall only be valid when the **Connection Status** field is set to one otherwise it is Reserved and shall be set to zero. This field shall indicate the current power operation mode of the Connector. **Value** / **Meaning** 0 – No Consumer 1 – USB Default Operation/Type-C Current – default 2 – BC 3 – PD 4 – Type-C Current – 1.5A 5 – Type-C Current – 3A 6-7 – Reserved
19	*Connection Status*	1	This field indicates the current connection status of the Connector. This field shall be set to one when a device is attached to this Connector.
20	*Power Direction*	1	This field shall only be valid when the **Connection Status** field is set to one. The field shall indicate whether the Connector is operating as a consumer or provider. **Value** / **Meaning** 0 – Connector is operating as a consumer 1 – Connector is operating as a provider
21	*Connector Partner Flags*	8	This field shall only be valid when the **Connection Status** field is set to one otherwise it is Reserved and shall be set to zero. This field indicates the current mode or modes the Connector is operating in. **Bit** / **Meaning** 0 – USB 1 – Alternate Mode 3-7 – Reserved

<table>
<tr><th>Offset (Bits)</th><th>Field</th><th>Size (Bits)</th><th>Description</th></tr>
<tr><td>29</td><td>Connector Partner Type</td><td>3</td><td>This field shall only be valid when the Connection Status field is set to one otherwise it is Reserved and shall be set to zero. This field indicates the type of connector partner detected on this Connector.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>DFP attached</td></tr>
<tr><td>2</td><td>UFP attached</td></tr>
<tr><td>3</td><td>Powered cable/No UFP attached</td></tr>
<tr><td>4</td><td>Powered cable/UFP attached</td></tr>
<tr><td>5</td><td>Debug Accessory attached</td></tr>
<tr><td>6</td><td>Audio Adapter Accessory attached</td></tr>
<tr><td>7</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>32</td><td>Request Data Object</td><td>32</td><td>This field shall only be valid when the Connection Status field is set to one and the Power Operation Mode field is set to PD otherwise it is Reserved and shall be set to zero.
This field shall return the currently negotiated power level as specified by the Request Message during power negotiation. See the specification of Power Request Data Objects in the [USBPD] for additional information on the contents of this data structure.</td></tr>
<tr><td>64</td><td>Reserved</td><td>2</td><td>Reserved and shall be set to zero.</td></tr>
<tr><td>66</td><td>Provider Capabilities Limited Reason</td><td>4</td><td>A bitmap indicating the reasons why the Provider capabilities of the Connector have been limited. This field shall only be valid if the Connector is operating as a provider otherwise it is Reserved and shall be set to zero.
If the PPM has lowered the capabilities but the reason doesn’t fall into any of the predefined categories, it can choose to not set any of these bits. Furthermore, if the Provider Capabilities change and are no longer limited, the PPM shall clear these bits.
See Table 4-26 for description of each bit.</td></tr>
<tr><td>70</td><td>Reserved</td><td>2</td><td>Reserved and shall be set to zero.</td></tr>
</table>

Table 4-25: Connector Status Change Field Description

Bit	Description
0	Reserved and shall be set to zero.
1	Reserved and shall be set to zero.
2	*Power Operation Mode Change* When set the **Power Operation Mode** field in GET_CONNECTOR_STATUS Data shall indicate the current power operational mode of the Connector.
3	Reserved and shall be set to zero.
4	Reserved and shall be set to zero.
5	*Supported Provider Capabilities Change* When set, the OSPM shall get the updated Power Data Objects by using the GET_PDOS Command. The **Supported Provider Capabilities Limited Reason** field shall indicate the reason if the provider capabilities are limited.
6	*Negotiated Power Level Change* When set, the **Request Data Object** field in GET_CONNECTOR_STATUS Data shall indicate the newly negotiated power level. Note that this bit shall be set by the PPM whenever a Power contract is established or renegotiated.
7	*PD Reset Complete* This field shall be set when the PPM completes a PD Hard Reset requested by the connector partner.
8	Reserved and shall be set to zero.
9	Reserved and shall be set to zero.
10	Reserved and shall be set to zero.
11	*Connector Partner Changed* This shall be set when either the **Connector Partner Type** field or the **Connector Partner Flags** field changes.
12	*Power Direction Changed* This shall be set when the PPM completes a Power Role Swap requested by the connector partner or due to the PPM autonomously performing a Power Role Swap. The **Power Direction** field in the GET_CONNECTOR_STATUS Data shall indicate the new Power Role.
13	Reserved and shall be set to zero.
14	*Connect Change* When set, the **Connection Status** field in GET_CONNECTOR_STATUS Data shall indicate whether a device is attached to the Connector. In addition, the **Operation Mode** field in GET_CONNECTOR_STATUS Data shall indicate the current operational mode of the Connector.
15	*Error* When set, this field shall indicate that an unknown error has occurred on the Connector.

Table 4-26: Provider Capabilities Limited Reason Field Description

Bit	Description
0	*Power Budget Lowered* When set, indicates that the Power Budget for the PPM has been lowered due to it being unplugged from an External Supply.
1	*Reaching Power Budget Limit* When set, indicates that the PPM is reaching the Power Budget Limit due to too many attached PD Sinks.
2	Reserved and shall bet set to zero.
3	Reserved and shall bet set to zero.

4.2.11 Get Error Status

The GET_ERROR_STATUS Command shall be used to get details about an error, if one is reported by the PPM. The OSPM may send this PPM Command to get additional details on why a PPM Command failed, which is indicated by the PPM by sending a PPM Command Response with the **Error Indicator** set. The OSPM may also send this PPM Command to get additional details on the error associated with a PPM Asynchronous Notification for an Asynchronous Event (e.g. connector change) with the **Error Indicator** set. In either case, the PPM shall clear the Error Status Data only after the OSPM has acknowledged the Get Error Status Command Response or after receiving a PPM_RESET Command. The format of the COMMAND Data Structure shall be as detailed in Table 4-27.

Table 4-27: GET_ERROR_STATUS Command

Offset (Bits)	Field	Size (Bits)	Description
0	*Command*	8	This field shall be set to GET_ERROR_STATUS.
8	*Data Length*	8	Shall be set to 00h.
16	*Reserved*	48	Reserved and shall be set to zero.

On successful completion of the GET_ERROR_STATUS Command the PPM shall return a PPM Command Response as described in Table 4-28.

Table 4-28: GET_ERROR_STATUS Response

Offset (Bits)	Field	Size (Bits)	Description
0	*Reserved*	1	Reserved and shall be set to zero.
1	*Connector Change Indicator*	7	If an Asynchronous Event occurred on a Connector, then the PPM shall set this field to the Connector Number on which the change occurred.
8	*Data Length*	8	Shall be set to 10h.
16	*Reserved*	9	Reserved and shall be set to zero.
25	*Not Supported Indicator*	1	Shall be set to 0b.
26	*Cancel Completed Indicator*	1	Shall be set to 0b.

Offset (Bits)	Field	Size (Bits)	Description
27	*Reset Completed Indicator*	1	Shall be set to 0b.
28	*Busy Indicator*	1	Shall be set to 0b.
29	*Acknowledge Command Indicator*	1	Shall be set to 0b.
30	*Error Indicator*	1	Shall be set to 0b.
31	*Command Completed Indicator*	1	Shall be set to 1b.

If the GET_ERROR_STATUS Command completed successfully then the PPM shall return additional Data following the GET_ERROR_STATUS Response as described in Table 4-29.

Table 4-29: GET_ERROR_STATUS Data

<table>
<tr><th>Offset (Bits)</th><th>Field</th><th>Size (Bits)</th><th>Description</th></tr>
<tr><td>0</td><td>Error Information</td><td>16</td><td>This field indicates the reason(s) for the error reported by the PPM.
<table>
<tr><th>Bit</th><th>Meaning</th></tr>
<tr><td>0</td><td>Unrecognized PPM Command</td></tr>
<tr><td>1</td><td>Non-existent Connector Number</td></tr>
<tr><td>2</td><td>Invalid PPM Command specific parameters</td></tr>
<tr><td>3</td><td>Incompatible connector partner</td></tr>
<tr><td>4</td><td>CC communication error</td></tr>
<tr><td>5</td><td>PPM Command unsuccessful due to dead battery condition</td></tr>
<tr><td>6</td><td>Contract negotiation failure</td></tr>
<tr><td>7</td><td>Overcurrent</td></tr>
<tr><td>8</td><td>Undefined/unknown error</td></tr>
<tr><td>9</td><td>Port partner rejected swap</td></tr>
<tr><td>10</td><td>Partner sent Hard Reset</td></tr>
<tr><td>11</td><td>PPM Policy Conflict (OSPM sends a command which violates the PPM policy)</td></tr>
<tr><td>12</td><td>Swap Rejected</td></tr>
<tr><td>15: 13</td><td>Reserved and shall be set to zero.</td></tr>
</table></td></tr>
<tr><td>16</td><td>Vendor Defined</td><td>112</td><td>The contents of this field are vendor specific.</td></tr>
</table>

5 Operational Model

There shall only be one USB Type-C Bridge per Device Container unless the Device Container contains multiple PDUSB hubs. When a Device Container contains multiple PDUSB hubs, each PDUSB hub shall have its own Bridge.

If a USB Type-C Bridge Device has multiple Configurations, then the USB Type-C Bridge function shall be exposed in all of them. A Device Container that supports USB Type-C Bridge shall have at least two independent PD Ports.

In addition to the default Control endpoint, a USB Type-C Bridge shall support a Notification endpoint. The default Control and Notification endpoints shall be used by a USB Type-C Bridge Class Driver in a PDUSB Host to send and receive PPM, Authentication and Firmware Update Related Requests and Notifications/Responses respectively. The Notification endpoint in the USB Type-C Bridge may be also used to send Billboard Notifications back to a USB Type-C Bridge Class Driver. A USB Type-C Bridge shall not send any notifications or responses that are not defined in this specification. A USB Type-C Bridge shall ignore any received requests or responses that are not defined in this specification.

The USB Type-C Bridge shall respond to USB Type-C Bridge class-specific USB Requests with a request error if it is not in the **Configured** USB State. The USB Type-C Bridge Class Driver shall only have one outstanding USB Type-C Bridge class-specific USB Request for a given USB Type-C Bridge active at any given time. The USB Type-C Bridge having sent a USB Type-C Bridge notification shall not send another notification until the previous notification has been acknowledged.

All requests to or notifications from the USB Type-C Bridge shall reference the Connector Number of a Connector on the Device Container, if a Connector Number is part of that request or notification. The Connector Number used to address the USB Type-C Connectors that the USB Type-C Bridge Class Driver can communicate with shall be determined by the position of the bits set to one in the *bmConnectorMask* in Table 3-1. For example, bit 1 corresponds to Connector Number 1 and bit 3 corresponds to Connector Number 3 and so on and so forth.

The general operational models that describe the PAM, Billboard, and Firmware Update units are defined in their respective specifications.

5.1 USB Operating Speed

A USB Type-C Bridge Device shall operate at a USB 2.0 speed. A High Speed Bridge Device shall also support Bridge functionality at Full Speed.

If the Device Container includes a PDUSB Hub, then the USB Type-C Bridge shall be connected to a USB 2.0 downstream port on the PDUSB Hub. The port shall be marked as an Internal/Non-removable port.

If the Device Container does not include a PDUSB Hub but includes some other form of USB functionality, then that Device Container shall expose the USB Type-C Bridge as a separate function within that Device Container. The Device Container shall support Bridge functionality at all speeds the other USB Device functionality supports.

5.2 USB TYPE-C Bridge and PPM

A PPM shall always respond to a PPM_RESET Command. The PPM shall not respond to any PPM Command other than PPM_RESET, unless Command Completed Notifications have been enabled by a SET_NOTIFICATION_ENABLE Command. The PPM shall process PPM Commands regardless of whether or not Command Completed Notifications are enabled. The PPM shall

not send the OSPM any asynchronous notifications unless the corresponding notification type has been enabled by a SET_NOTIFICATION_ENABLE Command.

The USB Type-C Bridge Class Driver shall use the default Control endpoint to send PPM Commands to the USB Type-C Bridge and shall receive the PPM Asynchronous Notifications and PPM Command Responses via the Notification endpoint.

The OSPM shall send at most one PPM Command at a time to the PPM. With the exception of the CANCEL and PPM_RESET Commands the OSPM shall wait until the current PPM Command is complete before sending the next PPM Command. A PPM Command is complete to the OSPM in the following circumstances:

- If the PPM Command is a PPM_RESET, it is complete when the OSPM receives a PPM Response with the **Reset Complete Indicator** bit set to 1b.
- If the OSPM attempts to cancel a PPM Command, the cancelled PPM Command is complete when the OSPM receives a PPM Response with the **Cancel Command Indicator** bit set to 1b
- If the PPM Command is not cancelled and is not a PPM_RESET, it is complete when the OSPM receives a PPM Response with the **Acknowledge Command Indicator** bit set to 1b.

A PPM_RESET Command may be sent by the OSPM at any time. If the OSPM receives a PPM Command Response or an Asynchronous Notification after sending a PPM_RESET Command, but before receiving a PPM Response with the **Reset Complete Indicator** bit set to 1b, it shall not acknowledge it.

An OSPM may cancel a PPM command by sending a CANCEL Command before acknowledging the Command Response. An OSPM shall not cancel a PPM Command after it has acknowledged the Command Response. An OSPM shall not cancel a PPM_RESET Command, a CANCEL Command, or an ACK_PCR_PAN Command.

After the OSPM sends a CANCEL Command, it shall not initiate a new PPM Command (other than a PPM_RESET) until it receives a PPM Command Response for the CANCEL Command. Once the OSPM receives a PPM Command Response for the CANCEL Command, it shall consider the previously pending PPM Command to be complete regardless of whether or not it received a PPM Command Response for the cancelled PPM Command. An OSPM shall not send an ACK_PCR_PAN Command for a cancelled PPM Command, even if it received a PPM Command Response for the cancelled PPM Command.

The PPM shall send at most one PPM Asynchronous Notification at a time to the OSPM. The PPM shall wait until the OSPM Acknowledges the PPM Asynchronous Notification before sending the next PPM Asynchronous Notification. A PPM shall not set the **Connector Change Indicator** if the change on the Connector occurred as a direct result of a PPM Command sent to that Connector unless the PPM will require more time to process the command.

5.2.1 PPM Command (other than CANCEL or PPM_RESET) Reception

On reception of a PPM Command that is neither CANCEL nor PPM_RESET, the PPM shall:

1. If the PPM is currently processing a different PPM Command, complete the newly received PPM Command USB Request with a Request Error and exit without performing any additional steps.
2. Execute the PPM Command.

3. If the PPM Command is an ACK_PCR_PAN that is acknowledging a Get Error Status Command, then clear any Error Status Data that have been set.

4. Set the CCI Data Structure and any Data associated with the PPM Command Response as detailed in the sections for each command.

5. If the **Command Completed** notification is enabled, send a PPM Command Response to the OSPM.

5.2.2 CANCEL Command Reception

On reception of a CANCEL command, the PPM shall:

1. If the last received PPM Command was an ACK_PCR_PAN Command, PPM_RESET Command, or a CANCEL Command, complete the newly received CANCEL Command USB Request with a Request Error and exit without performing any additional steps.

2. If the last received PPM Command hasn't started executing yet, the PPM shall not execute the PPM Command. The PPM shall not send a PPM Command Response for the cancelled PPM Command.

3. If the last received PPM Command has started executing but hasn't finished, the PPM shall stop performing the PPM Command. The PPM shall not send a PPM Command Response for the cancelled PPM Command.

4. If the last received PPM Command has completed but PPM has not yet sent a PPM Command Response, it shall drop the Response. The PPM shall not send a PPM Command Response for the cancelled PPM Command.

5. The PPM shall update the CCI Data Structure with the **Cancel Completed Indicator** set to one.

6. If the **Command Completed** notification is enabled, the PPM shall send a PPM Command Response to the OSPM.

5.2.3 PPM_RESET Command Reception

On reception of a PPM_RESET command, the PPM shall:

1. Disable all notifications

2. Reset itself.

3. Set the **Reset Completed Indicator** in the CCI Data Structure.

4. Send a PPM Command Response to the OSPM.

5.2.4 Asynchronous Events

When an Asynchronous Event occurs on one or more of the Connectors, then the PPM shall:

1. Set the CCI Data Structure and optionally any Data associated with the PPM Asynchronous Notification.

2. If the corresponding notification was enabled by the OSPM send a PPM Asynchronous Notification the OSPM.

5.2.5 PPM Command Response or PPM Asynchronous Notification Reception

Once the OSPM is notified of either a PPM Command Response and/or a PPM Asynchronous Notification the OSPM shall:

1. Decode the CCI and any additional Data.

2. If this was a PPM Asynchronous Notification, send any other PPM Commands the OSPM needs to get details of the PPM Asynchronous Notification.

3. Acknowledge the PPM Command Response or PPM Asynchronous Notification via the ACK_PCR_PAN Command. The only PPM Command Response that is not acknowledged by the OSPM is the PPM Command Response for the ACK_PCR_PAN or the PPM_RESET Command.

5.3 USB TYPE-C Bridge and PAM

The USB Type-C Bridge Class Driver shall use a Send Authentication Data Request to send an Authentication Message from the OSAM to the USB Type-C Bridge.

A USB Type-C Bridge shall only send an Authentication Notification when:

- The USB Type-C Bridge receives an Authentication Message from a PD Product attached to a Connector on a Device Container. The Notification shall have *bParam3* set to Success.

- The USB Type-C Bridge receives a Not_Supported PD Message from a PD Product attached to a Connector on a Device Container. The Notification shall have *bParam3* set to Not_Supported.

- The USB Type-C Bridge is unable to send an Authentication Message to a PD Product attached to a Connector. The notification shall have *bParam3* set to Error.

After receiving an Authentication Notification with *bParam3* set to Success, the USB Type-C Bridge Class Driver shall use the Get Authentication Data Request to retrieve the Authentication Message from the USB Type-C Bridge.

All Authentication related requests are deemed complete when either the USB Type-C Bridge sends a USB Type-C Bridge Authentication Notification for that request or the OSAM times out the request per [USBAUTH].

The USB Type-C Bridge Authentication Notifications are acknowledged implicitly by successfully transferring the Authentication Notification to the USB Host over the USB Notification endpoint.

5.3.1 Host-Initiated Authentication

To initiate Authentication of a PD Product attached to a Connector on the Device Container:

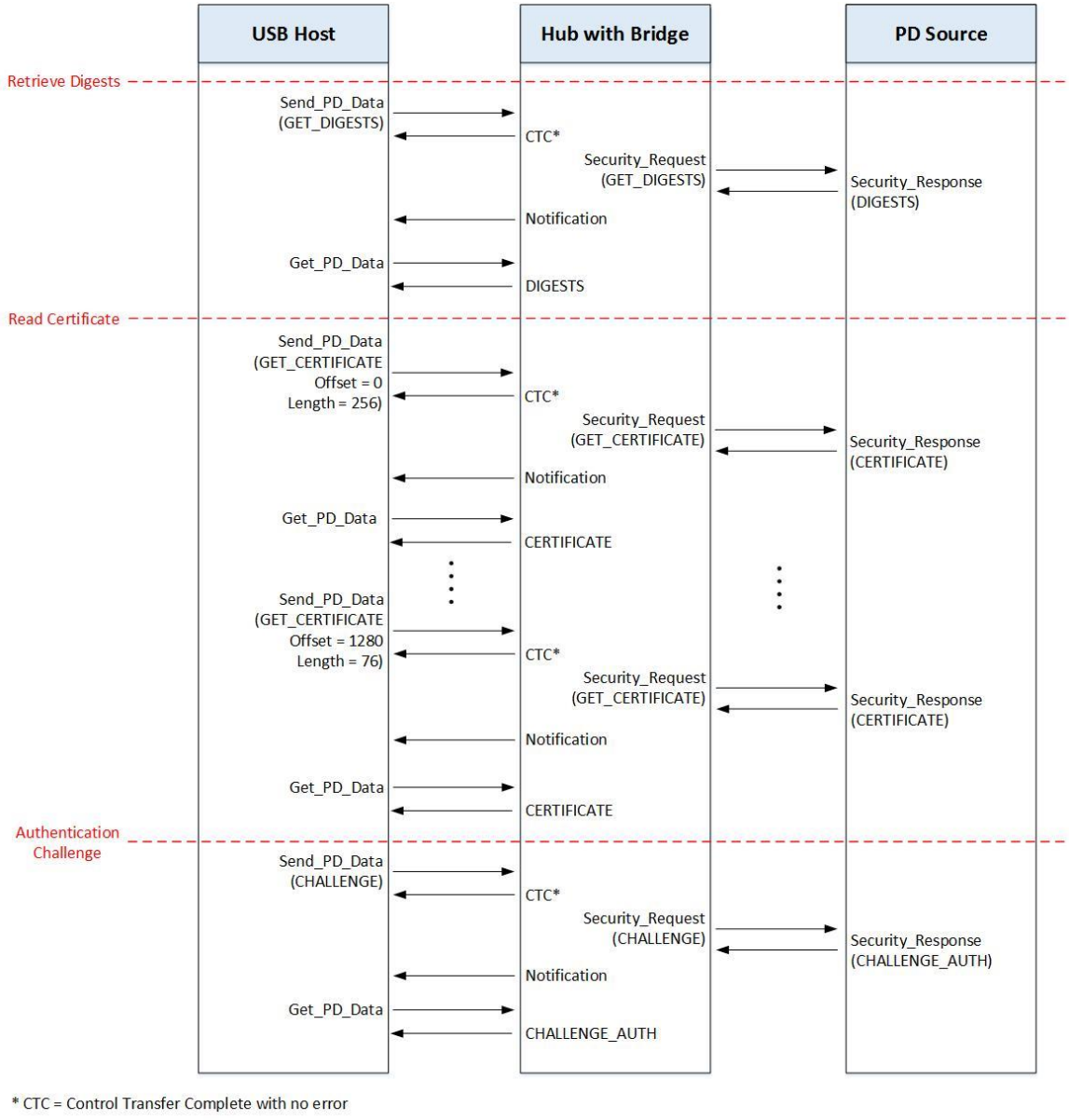
1. A USB Host Sends an Authentication Request in the **Data** field of a Send Authentication Data Request. The USB Type-C Bridge sends and receives Authentication Messages to/from a PD Product attached to a Connector on the Device Container as described in [USBPD].

2. When the USB Type-C Bridge receives an Authentication Response from the PD Product, it notifies the Host using an Authentication Notification.

3. Upon receiving the Authentication Notification, a Host uses a Get Authentication Data Request to retrieve the Authentication Message from the Bridge.

Figure 5-1 shows an example of a Host-initiated Authentication Sequence when a Connector on the Device Container is attached to a PD Source.

Figure 5-1 Example Authentication Sequence Initiated by the USB Host



* CTC = Control Transfer Complete with no error

5.3.2 PD Product-Initiated Authentication

Upon receiving an Authentication Request from a PD Product attached to a Connector on a Device Container:

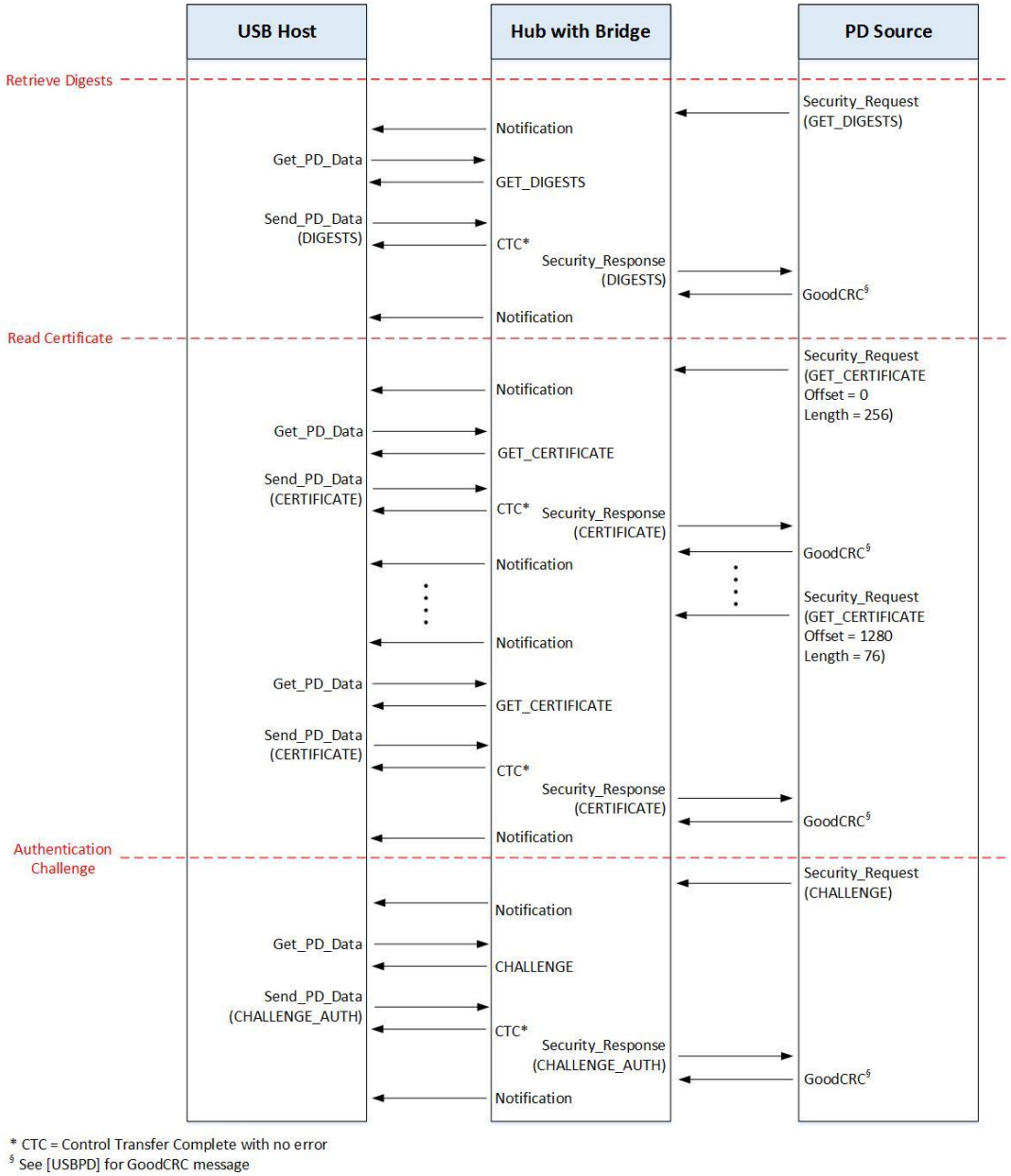
1. A USB Type-C Bridge notifies a USB Host by sending an Authentication Notification.

2. Upon receiving the Authentication Notification, a Host uses a Get Authentication Data Request to retrieve the Authentication Request from the Bridge.

3. When the Bridge receives a Send Authentication Data Request from the Host, it extracts the Authentication Message from the **Data** field of the received Send Authentication Data Request and sends it to the PD Product as described in [USBPD].

Figure 5-2 shows an example of a Host-initiated Authentication Sequence.

Figure 5-2 Example Authentication Sequence Initiated by a PD Product



* CTC = Control Transfer Complete with no error
§ See [USBPD] for GoodCRC message

5.4 USB TYPE-C Bridge and PFUM

The USB Type-C Bridge Class Driver uses the default Control endpoint to Send Firmware Update Requests. The USB Type-C Bridge notifies the USB Type-C Bridge Class Driver of the

completion of the same by sending back a Firmware Update Notification with the **bParam3** field set to either Success or Error. On reception of a Firmware Update Response from a PD Product attached to a Connector on the Device Container, the USB Type-C Bridge shall send a Firmware Update Notification containing the response data.

All Firmware Update related requests are deemed completed when the USB Type-C Bridge sends a USB Type-C Bridge Firmware Update Notification for that command.

The USB Type-C Bridge Firmware Update Notifications are acknowledged implicitly by successfully transferring the Firmware Update Notification to the System over the USB Notification endpoint.

5.4.1 Host-Initiated Firmware Update

To initiate Firmware Update of a PD Product attached to a Connector on a Device Container, a USB Host Sends a Firmware Update Request in the **Data** field of a Send Firmware Update Data Request.

5.5 USB TYPE-C Bridge and Billboard

When a USB Type-C Bridge supports the Billboard functionality, the Billboard Capability shall be as defined in [USBBB]. The USB Type-C Bridge shall use the mechanisms defined in [USBBB] to notify System Software of a change in its Billboard Capability descriptor. If a USB Type-C Bridge supports the Billboard functionality it shall also support the Billboard Notification. Using the Billboard Notification method allows the USB Type-C Bridge to avoid unnecessary USB Disconnect/Connect whenever there is a change in the Billboard Capability descriptor.

On Power up or after a USB Reset the USB Type-C Bridge shall not send any Billboard Notifications and it shall use the mechanisms defined in [USBBB] to notify System Software of any Billboard Capability changes.

If the USB Type-C Bridge supports the Billboard functionality (See Table 3-1) and System Software Enables the Billboard Notification via the Enable/Disable Billboard Notification Request, then the USB Type-C Bridge shall transition to notifying System Software of changes in its Billboard Capability descriptor via the Notification endpoint. The USB Type-C Bridge Class Driver shall use the standard Get BOS Descriptor Request, to retrieve the Billboard Capability descriptor, when the USB Type-C Bridge sends a Billboard Notification to the System Software.

The USB Type-C Bridge Billboard Notifications are acknowledged implicitly by successfully transferring the Billboard Notification to the System over the USB Notification endpoint.

5.6 USB TYPE-C Bridge Initialization

The USB Type-C Bridge shall enumerate and initialize as a standard USB Device. During Type-C Bridge initialization, System Software should scan all the Connectors on the Device Container.

5.7 USB TYPE-C Bridge Operational State Diagram

The figures in this section describe the operational state diagram for a USB Type-C Bridge. The following variables are used to track state:

NotificationCount	Number of Responses and Notifications that the Bridge has queued
PendingCount	Number of queued PPM events
BCRequest	Pointer to the current pending Bridge Class USB Request
PendingPPM	Pointer to the current pending PPM Command
WaitForCI	Indicates whether or not Bridge is waiting for an ACK_PCR_PAN for an Asynchronous Notification
ACKSuccess	Indicates whether or not a PPM_ACK_PCR_PAN Command is valid
SendNotification	Indicates whether or not the PPM should send a Notification or Response after receiving an ACK_PCR_PAN
PPMState	PPM state as defined in [UCSI] *Note: PPM State only applies for PPM Commands, PPM Responses, and PPM Asynchronous Notifications*

Figure 5-3 Request/Notification Processing

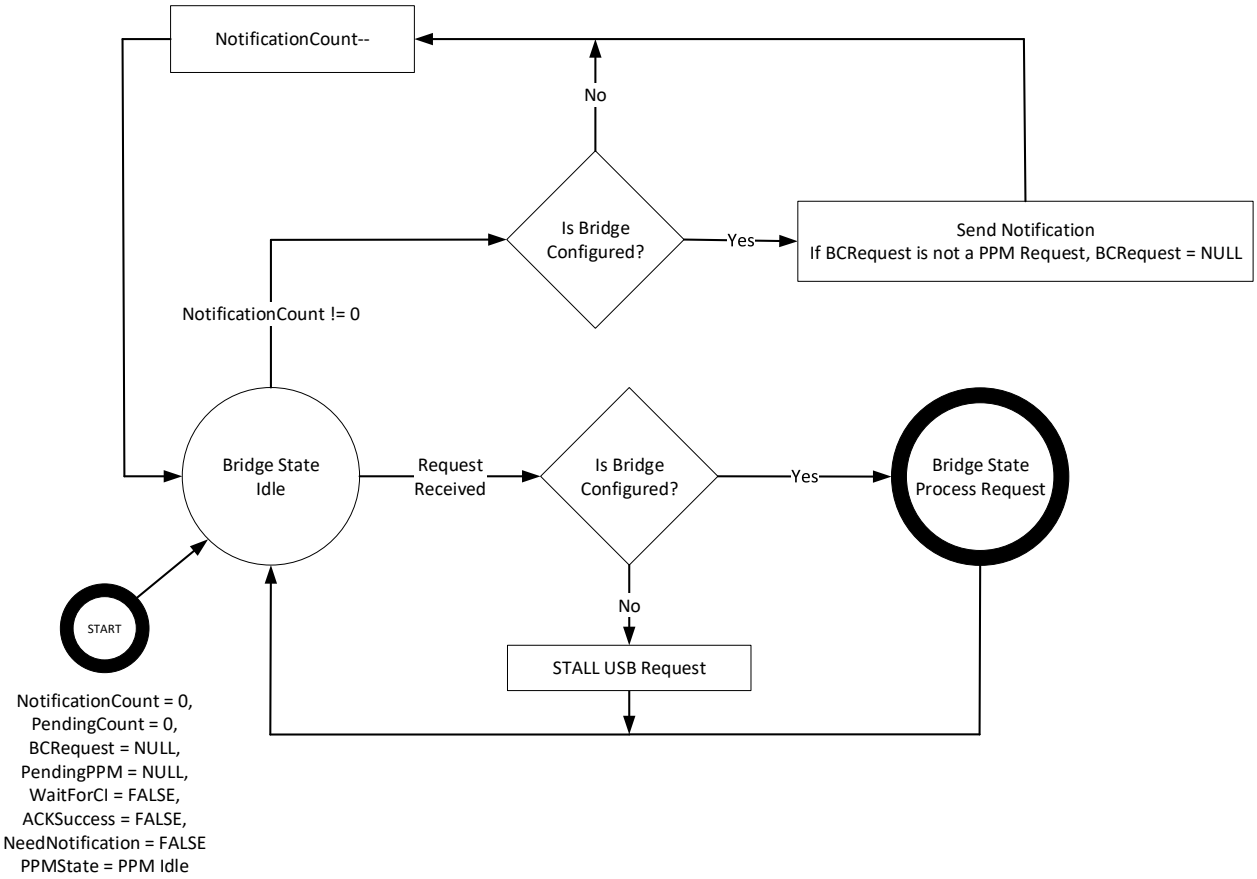


Figure 5-4 Asynchronous Event Handling

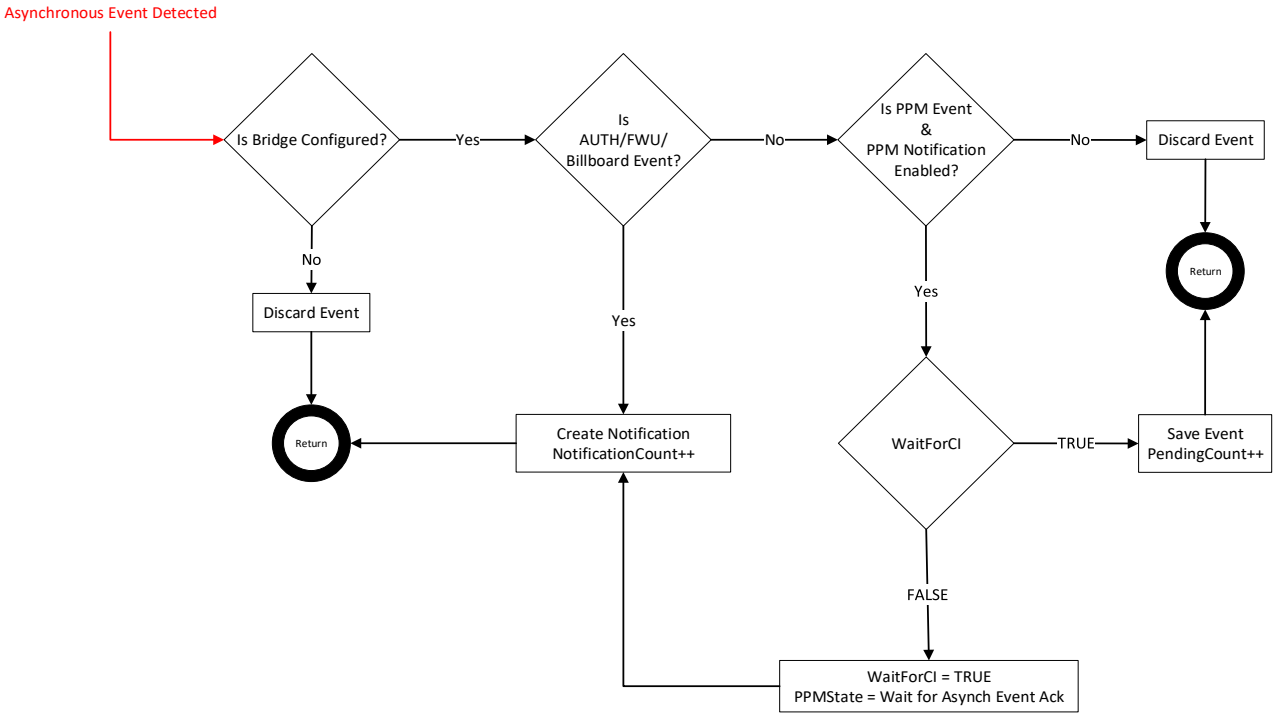


Figure 5-5 Process Request

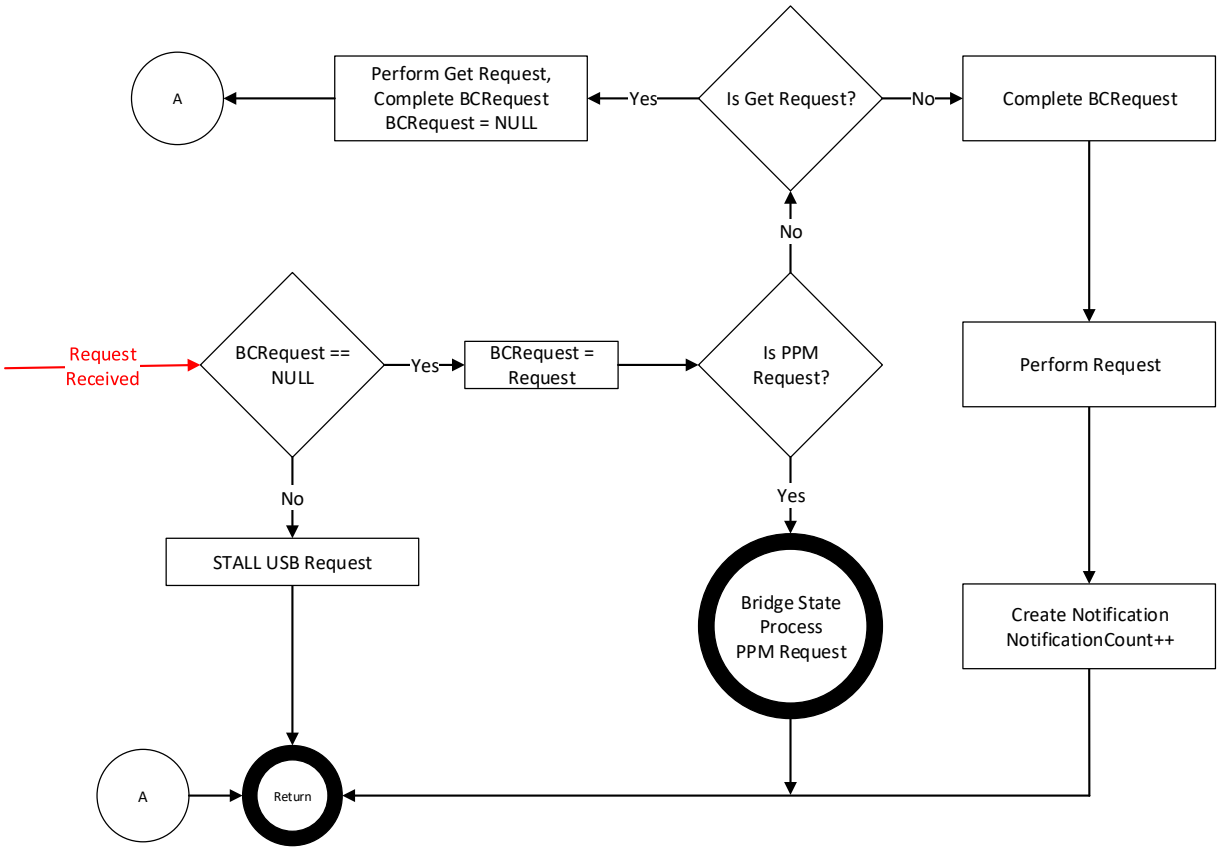


Figure 5-6 Process PPM Request

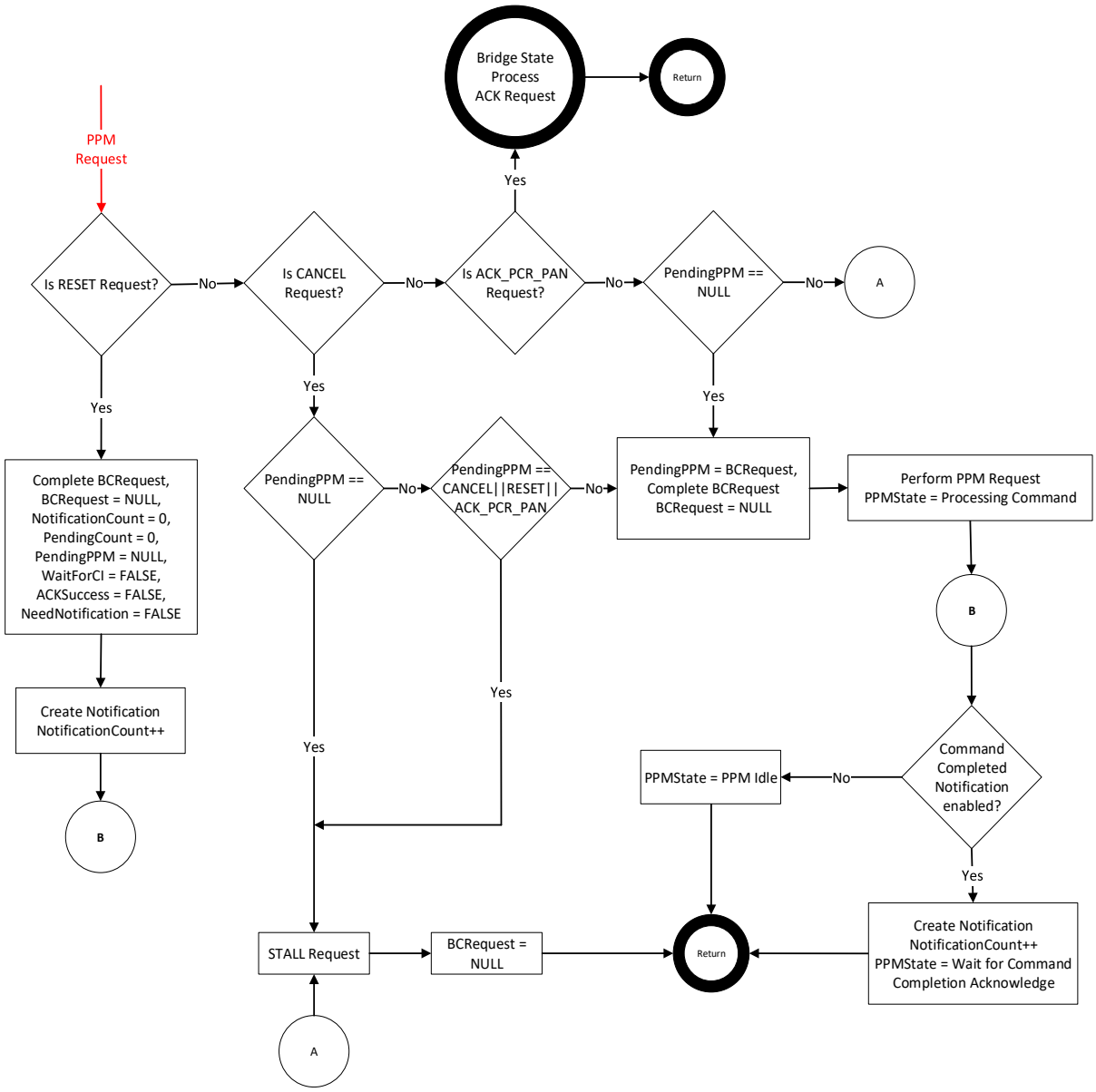
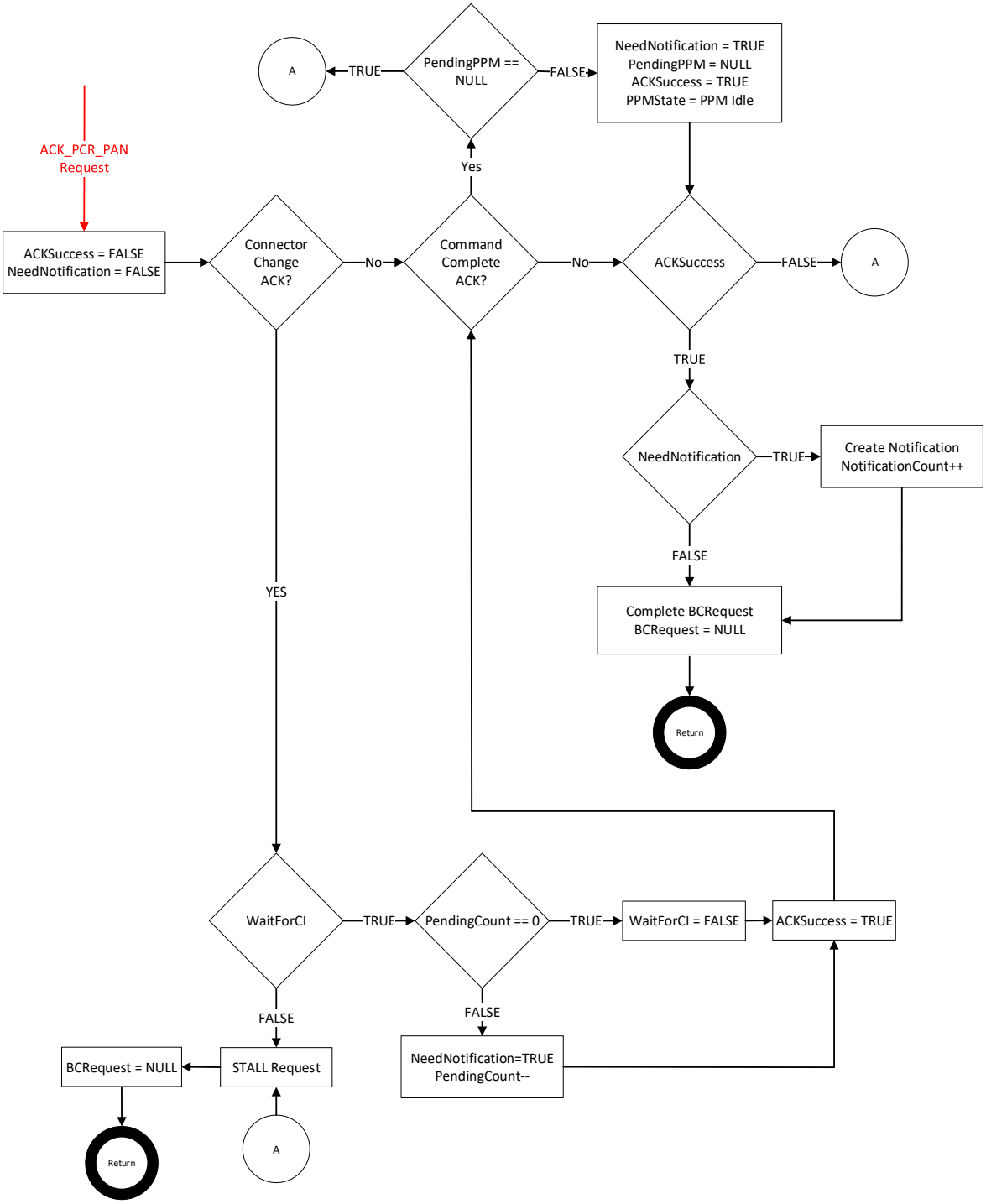


Figure 5-7 Process ACK Request



5.8 USB Type-C Bridge Suspend/Resume

This section details how System Software may suspend a Device Container with a USB Type-C Bridge and the expected behavior of System Software when it wants to Resume the Device Container or it receives Resume signaling from the Device Container.

5.8.1 USB Type-C Bridge Suspend

The USB Type-C Bridge shall support Remote Wake. When enabled for Remote Wake, the USB Type-C Bridge shall attempt to wake the System up whenever it detects an Asynchronous Event that would have caused it to send a notification.

System Software shall enable the appropriate notifications (e.g. OSPM is expected to enable Connect Change Notification on the USB Type-C Bridge) such that an Asynchronous Event is translated into a Resume Signal. Only PPM Notifications can cause a Remote Wake event to be generated by the USB Type-C Bridge. The steps to properly enable Remote Wake shall be the following:

1. System Software enables the appropriate PPM Asynchronous Notifications.
2. System Software enables Remote Wake on the USB Type-C Bridge.
3. System Software suspends the port on which the USB Type-C Bridge is connected.
4. If the Device Container contains a PDUSB Hub:
    a. System Software enables Remote Wake on the PDUSB Hub.
    b. System Software suspends the PDUSB Hub.

5.8.2 USB Type-C Bridge Resumed by Host

If the Device Container is resumed by Host-Initiated Wake, then System Software shall scan all the Connectors on the Device Container for current connector status and capabilities. The steps to properly resume after Host-Initiated Wake shall be the following:

1. System Software drives Resume signaling on Device Container upstream port.
2. If the Device Container contains a PDUSB Hub, System Software resumes the port on which the USB Type-C Bridge is connected.
3. System Software scans all the Connectors on the Device Container.

5.8.3 USB Type-C Bridge Resumed by USB Type-C Bridge

If the Device Container initiates the Wake, then once resumed, the USB Type-C Bridge shall send the PPM Notification that caused the USB Type-C Bridge to resume the System. The steps to properly resume after Device-Initiated Wake shall be the following:

1. USB Type-C Bridge initiates resume signaling.
2. System Software acknowledges resume signaling and clears the port suspend on the downstream port on which the Device Container is connected.
3. If the Device Container contains a PDUSB Hub, System Software resumes the downstream port on which the USB Type-C Bridge is attached.
4. USB Type-C Bridge sends the PPM Notification to System Software.
5. System Software scans all the Connectors on the Device Container.

A Values of Constants

A.1. Class Codes

Table A-1: Class Codes

Code	Value
Class	12h
Subclass	00h
Protocol	00h

A.2. Request Types

Table A-2: Request Types

Request	Value
SEND_PPM_COMMAND	01h
SEND_PD_DATA	02h
GET_PD_DATA	03h
ENABLE_BB_NOTIFICATION	04h

A.3. Notification Types

Table A-3: Notification Types

Notification	Value
PPM_NOTIFICATION	01h
PD_NOTIFICATION	02h
BILLBOARD_NOTIFICATION	03h
Reserved	04h - FFh

A.4. Constants

Table A-4: Constants

Parameter	Value
MAX_DATA_LENGTH	38h
AUTH	01h
FWU	02h
BRIDGE_CAPABILITY	21h