31/10/2023 18:58:45

Compare Results

Old File:		New File:
USB_PD_R3_1 V1.8 2023-04_Ch6.pdf	versus	**USB_PD_R3_2 V1.0 2023-10_Ch 6.pdf**
165 pages (5.82 MB)		**224 pages (6.47 MB)**
13/10/2023 19:35:54		31/10/2023 18:08:46

Total Changes

3054

Text only comparison

Content

1723 Replacements

700 Insertions

631 Deletions

Styling and Annotations

0 Styling

0 Annotations

Go to First Change (page 1)

6. Protocol Layer

6.1 Overview

This chapter describes the requirements of the USB Power Delivery Specification's protocol layer including:

- Details of how Messages are constructed and used.
- Use of timers and timeout values.
- Use of Message and retry counters.
- Reset operation.
- Error handling.
- State behavior.

Refer to *Section 2.6 "Architectural Overview"* for an overview of the theory of operation of USB Power Delivery.

6.2 Messages

This specification defines three types of Messages:

- Control Messages that are short and used to manage the Message flow between Port Partners or to exchange Messages that require no additional data. Control Messages are 16 bits in length.
- Data Messages that are used to exchange information between a pair of Port Partners. Data Messages range from 48 to 240 bits in length.
  - There are three types of Data Messages:
    - Those used to expose capabilities and negotiate power.
    - Those used for the BIST.
    - Those that are Vendor Defined.
- Extended Messages that are used to exchange information between a pair of Port Partners. Extended Messages are up to ***MaxExtendedMsgLen*** bytes.
  - There are several types of Extended Messages:
    - Those used for Source and Battery information.
    - Those used for Security.
    - Those used for Firmware Update.
    - Those that are vendor defined.

6.2.1 Message Construction

All Messages ***Shall*** be composed of a Message Header and a variable length (including zero) data portion. A Message either originates in the Protocol Layer and is passed to the Physical Layer, or it is received by the Physical Layer and is passed to the Protocol Layer.

Figure 6-1 "USB Power Delivery Packet Format including Control Message Payload" illustrates a Control Message as part of a Packet showing the parts are provided by the Protocol and PHY Layers.

Figure 6-1 "USB Power Delivery Packet Format including Control Message Payload"

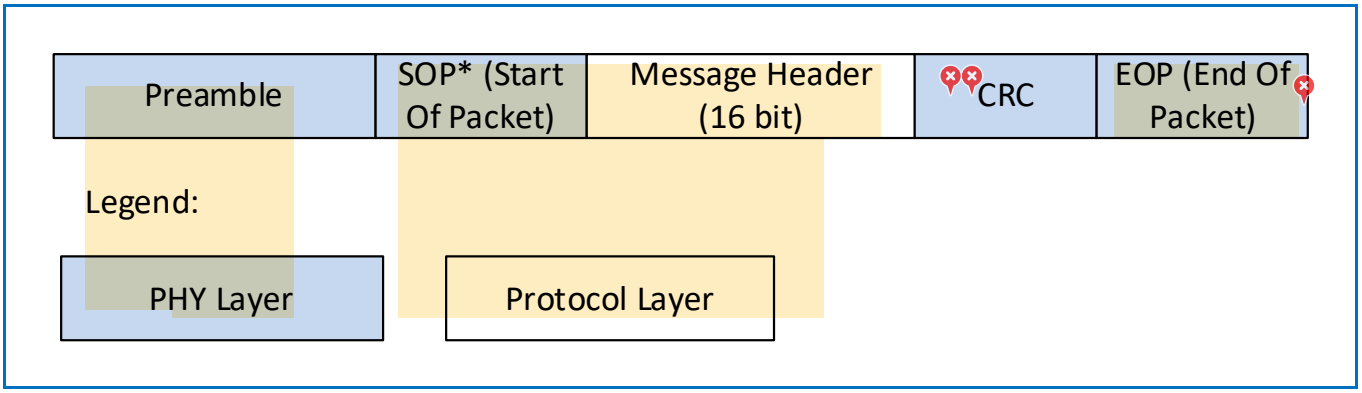


Figure 6-2 "USB Power Delivery Packet Format including Data Message Payload" illustrates a Data Message as part of a Packet showing the parts are provided by the Protocol and PHY Layers.

Figure 6-2 "USB Power Delivery Packet Format including Data Message Payload"

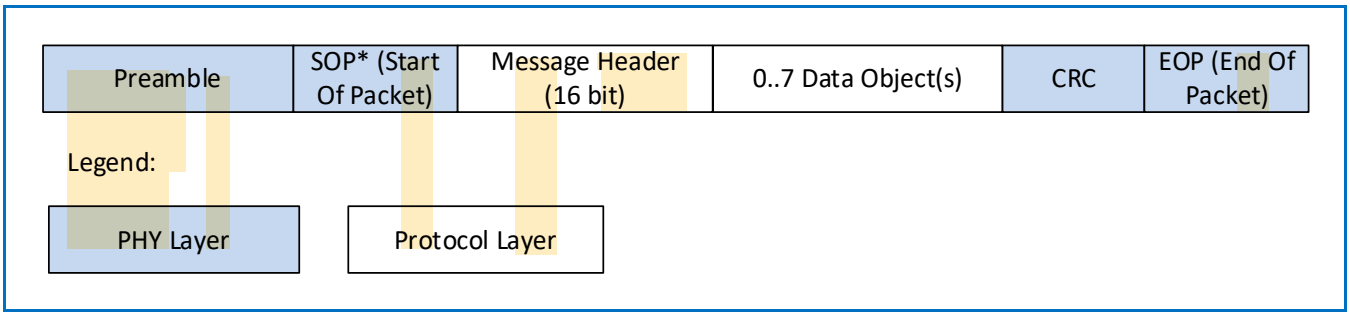
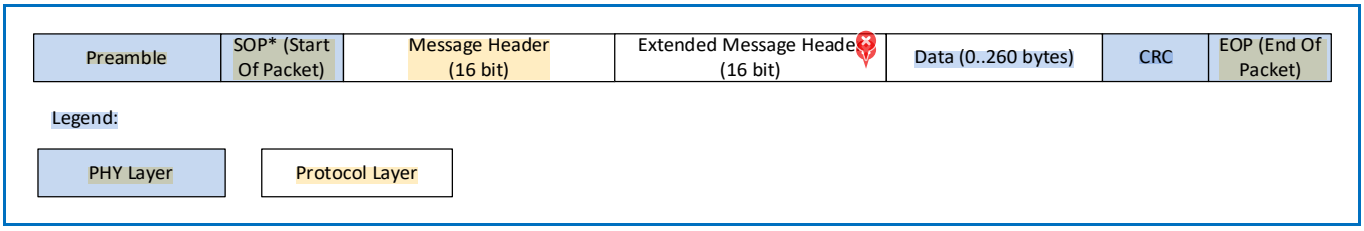


Figure 6-3 "USB Power Delivery Packet Format including an Extended Message Header and Payload" illustrates an Extended Message as part of a Packet showing the parts are provided by the Protocol and PHY Layers.

Figure 6-3 "USB Power Delivery Packet Format including an Extended Message Header and Payload"



6.2.1.1 Message Header

Every Message ***Shall*** start with a Message Header as shown in:

- *Figure 6-1 "USB Power Delivery Packet Format including Control Message Payload"*
- *Figure 6-2 "USB Power Delivery Packet Format including Data Message Payload"*
- *Figure 6-3 "USB Power Delivery Packet Format including an Extended Message Header and Payload"*

and as defined in *Table 6.1 "Message Header"*. The Message Header contains basic information about the Message and the PD Port Capabilities.

The Message Header ***May*** be used standalone as a Control Message when the Number of Data Objects field is zero or as the first part of a Data Message when the ***Number of Data Objects*** field is non-zero.

Table 6.1 "Message Header"

Bit(s)	Start of Packet	Field Name	Reference
15	SOP*	***Extended***	***Section 6.2.1.1.1***
14…12	SOP*	***Number of Data Objects***	***Section 6.2.1.1.2***
11…9	SOP*	***MessageID***	***Section 6.2.1.1.3***
8	SOP only	***Port Power Role***	***Section 6.2.1.1.4***
	SOP'/SOP''	***Cable Plug***	***Section 6.2.1.1.7***
7…6	SOP*	***Specification Revision***	***Section 6.2.1.1.5***
5	SOP only	***Port Data Role***	***Section 6.2.1.1.6***
	SOP'/SOP''	***Reserved***	***Section 1.4.2.10***
4…0	SOP*	***Message Type***	***Section 6.2.1.1.8***

6.2.1.1.1 Extended

The 1-bit ***Extended*** field ***Shall*** be set to zero to indicate a Control Message or Data Message and set to one to indicate an Extended Message.

The ***Extended*** field ***Shall*** apply to all SOP* Packet types.

6.2.1.1.2 Number of Data Objects

When the ***Extended*** field is set to zero the 3-bit ***Number of Data Objects*** field ***Shall*** indicate the number of 32-bit Data Objects that follow the Message Header. When this field is zero the Message is a Control Message and when it is non-zero, the Message is a Data Message.

The ***Number of Data Objects*** field ***Shall*** apply to all SOP* Packet types.

When both the ***Extended*** bit and ***Chunked*** bit are set to one, the ***Number of Data Objects*** field ***Shall*** indicate the number of Data Objects in the Message padded to the 4-byte boundary including the Extended Header as part of the first Data Object.

When the ***Extended*** bit is set to one and ***Chunked*** bit is set to zero, the ***Number of Data Objects*** field ***Shall*** be ***Reserved***. Note that in this case, the message length is determined solely by the ***Data Size*** field in the Extended Message Header.

6.2.1.1.3 MessageID

The 3-bit ***MessageID*** field is the value generated by a rolling counter maintained by the originator of the Message. The ***MessageIDCounter*** ***Shall*** be initialized to zero at power-on as a result of a Soft Reset, or a Hard Reset. The ***MessageIDCounter*** ***Shall*** be incremented when a Message is successfully received as indicated by receipt of a ***GoodCRC*** Message.

Note: the usage of ***MessageID*** during testing with BIST Messages is defined in ***[USBPDCompliance]***.

The ***MessageID*** field ***Shall*** apply to all SOP* Packet types.

6.2.1.1.4 Port Power Role

The 1-bit ***Port Power Role*** field ***Shall*** indicate the Port's present power role:

- 0b Sink
- 1b Source

Messages, such as ***Ping***, and ***GotoMin***, that are only ever sent by a Source, ***Shall*** always have the ***Port Power Role*** field set to Source. Similarly, Messages such as the ***Request*** Message that are only ever sent by a Sink ***Shall*** always have the ***Port Power Role*** field set to Sink.

During the Power Role Swap Sequence, for the initial Source Port, the ***Port Power Role*** field ***Shall*** be set to Sink in the ***PS_RDY*** Message indicating that the initial Source's power supply is turned off (see ***Table 8.63 "Steps for a Successful Source Initiated Power Role Swap Sequence"*** and ***Table 8.66 "Steps for a Successful Sink Initiated Power Role Swap Sequence"***).

During the Power Role Swap Sequence, for the initial Sink Port, the ***Port Power Role*** field ***Shall*** be set to Source for Messages initiated by the Policy Engine after receiving the ***PS_RDY*** Message from the initial Source (see ***Table 8.63 "Steps for a Successful Source Initiated Power Role Swap Sequence"*** and ***Table 8.66 "Steps for a Successful Sink Initiated Power Role Swap Sequence"***).

During the Fast Role Swap Sequence, for the initial Source Port, the ***Port Power Role*** field ***Shall*** be set to Sink in the ***PS_RDY*** Message indicating that V_{BUS} is not being driven by the initial Source and is within ***vSafe5V*** (see ***Figure 8-41 "Successful Fast Role Swap Sequence"***).

During the Fast Role Swap Sequence, for the initial Sink Port, the ***Port Power Role*** field ***Shall*** be set to Source for Messages initiated by the Policy Engine after receiving the ***PS_RDY*** Message from the initial Source (see ***Figure 8-41 "Successful Fast Role Swap Sequence"***).

Note that the ***GoodCRC*** Message sent by the initial Sink in response to the ***PS_RDY*** Message from the initial Source will have its ***Port Power Role*** field set to Sink since this is initiated by the Protocol Layer. Subsequent Messages initiated by the Policy Engine, such as the ***PS_RDY*** Message sent to indicate that V_{BUS} is ready, will have the ***Port Power Role*** field set to Source.

The ***Port Power Role*** field of a received Message ***Shall Not*** be verified by the receiver and ***Shall Not*** lead to Soft Reset, Hard Reset or Error Recovery if it is incorrect.

The ***Port Power Role*** field ***Shall*** only be defined for SOP Packets.

6.2.1.1.5 Specification Revision

The ***Specification Revision*** field ***Shall*** be one of the following values (except 11b):

- 00b –Revision 1.0
- 01b –Revision 2.0
- 10b – Revision 3.0
- 11b – ***Reserved, Shall Not*** be used.

To ensure interoperability with existing USBPD Products, USBPD Products ***Shall*** support every PD Specification Revision starting from ***[USBPD 2.0]*** for ***SOP****; the only exception to this is a VPD which ***Shall Ignore*** Messages sent with PD Specification Revision 2.0 and earlier.

After a physical or logical (USB Type-C® Error Recovery) Attach, a Port discovers the common Specification Revision level between itself and its Port Partner and/or the Cable Plug(s), and uses this Specification Revision level until a Detach, Hard Reset or Error Recovery happens.

After detection of the Specification Revision to be used, all PD communications ***Shall*** comply completely with the relevant revision of the PD specification.

The 2-bit ***Specification Revision*** field of a ***GoodCRC*** Message does not carry any meaning and ***Shall*** be considered as don't care by the recipient of the Message. The sender of a ***GoodCRC*** Message ***Shall*** set the Specification Revision field to 01b when responding to a Message that contains 01b in the Specification Revision field of the Message Header. The sender of a ***GoodCRC*** Message ***May*** set the Specification Revision field to 00b or 01b or 10b when responding to a Message that contains 10b in the Specification Revision field of the Message Header.

The ***Specification Revision*** field ***Shall*** apply to all SOP* Packet types.

An Attach event or a Hard Reset ***Shall*** cause the detection of the applicable Specification Revision to be performed for both Ports and Cable Plugs according to the rules stated below:

When the Source Port first communicates with the Sink Port the ***Specification Revision*** field ***Shall*** be used as described by the following steps:

- The Source Port sends a ***Source_Capabilities*** Message to the Sink Port setting the ***Specification Revision*** field to the highest Revision of the Power Delivery Specification the Source Port supports.
- The Sink Port responds with a ***Request*** Message setting the ***Specification Revision*** field to the highest Revision of the Power Delivery Specification the Sink Port supports that is equal to or lower than the ***Specification Revision*** received from the Source Port.
- The Source and Sink Ports Shall use the ***Specification Revision*** in the ***Request*** Message from the Sink in step 2 in all subsequent communications until a Detach, Hard Reset, or Error Recovery happens.

Prior to entering an explicit contract, the VCONN Source ***Shall*** use the following steps to establish a Specification Revision level:

- The VCONN Source sends a ***Discover Identity*** REQ to the Cable Plug (SOP') setting the ***Specification Revision*** field in the Message to the highest Revision of the Power Delivery Specification the VCONN Source supports. After a VCONN Swap the required ***Soft_Reset*** / ***Accept*** message exchange is used for the same purpose (see ***Section 6.3.13 "Soft Reset Message"***).
- The Cable Plug responds with a ***Discover Identity*** ACK setting the ***Specification Revision*** field in the Message to the highest Revision of the Power Delivery Specification the VCONN Source supports that is equal to or lower than the ***Specification Revision*** it received from the Source Port.
- The Cable Plug and VCONN Source Shall communicate using the lower of the two revisions until an Explicit Contract has been established.
- ***Table 6.2 "Revision Interoperability during an Explicit Contract"*** shows the ***Specification Revision*** that Shall be used between the Port Partners and the Cable Plugs when the ***Specification Revision*** has been discovered and an Explicit Contract is in place.

Notes:

- A VCONN Source that does not communicate with the Cable Plug(s) ***May*** skip the above procedure.
- When a Cable Plug does not respond to a Revision 3.0 ***Discover Identity*** REQ with a ***Discover Identity*** ACK or BUSY the VCONN Source ***May*** repeat steps 1-4 using a Revision 2.0 ***Discover Identity*** REQ in step 1 before establishing that there is no Cable Plug to communicate with.

A VCONN Source that supports Revision 3.0 of the Power Delivery Specification ***May*** communicate with a Cable Plug also supporting Revision 3.0 using Revision 3.0 Compliant Communications regardless of the ***Specification Revision*** of its Port Partner while no Explicit Contract exists. After an Explicit Contract has been established the Port Partners and Cable Plug(s) ***Shall*** use *Table 6.2 "Revision Interoperability during an Explicit Contract"* to determine the Revision to be used.

All data in all Messages ***Shall*** be consistent with the ***Specification Revision*** field in the Message Header for that particular Message.

A Cable Plug ***Shall Not*** save the state of the agreed ***Specification Revision***. A Cable Plug ***Shall*** respond with the highest ***Specification Revision*** it supports that is equal to or lower than the ***Specification Revision*** contained in the Message received from the VCONN Source.

Cable Plugs ***Shall*** operate using the same Specification Revision for both SOP' and SOP''. Cable assemblies with two Cable Plugs ***Shall*** operate using the same Specification Revision for both Cable Plugs.

See *Table 6.2 "Revision Interoperability during an Explicit Contract"* for details of how various Revisions ***Shall*** interoperate.

Table 6.2 "Revision Interoperability during an Explicit Contract"

Port 1 Revision	Cable Plug Revision	Port 2 Revision	Port to Port Operating Revision	Port to Cable Plug Operating Revision
2	2	2	2	2
2	2	3	2	2
2	3	2	2	2
2	3	3	2	2
3	2	2	2	2
3	2	3	3	2
3	3	2	2	2
3	3	3	3	3

6.2.1.1.6 Port Data Role

The 1-bit ***Port Data Role*** field ***Shall*** indicate the Port's present data role:

- 0b UFP
- 1b DFP

The ***Port Data Role*** field ***Shall*** only be defined for SOP Packets. For all other SOP* Packets the ***Port Data Role*** field is ***Reserved*** and ***Shall*** be set to zero.

If a USB Type-C® Port receives a Message with the ***Port Data Role*** field set to the same Data Role as its current Data Role, except for the ***GoodCRC*** Message, USB Type-C® Error Recovery actions as defined in ***[USB Type-C 2.3]*** ***Shall*** be performed.

For a USB Type-C® Port the ***Port Data Role*** field ***Shall*** be set to the default value at Attachment after a Hard Reset: 0b for a Port with R_d asserted and 1b for a Port with R_p asserted.

In the case that a Port is not USB Communications Capable, at Attachment a Source Port ***Shall*** default to DFP and a Sink Port ***Shall*** default to UFP.

6.2.1.1.7 Cable Plug

The 1-bit ***Cable Plug*** field ***Shall*** indicate whether this Message originated from a Cable Plug or VPD:

- 0b Message originated from a DFP or UFP.
- 1b Message originated from a Cable Plug or VPD

The ***Cable Plug*** field ***Shall*** only apply to SOP' and SOP'' Packet types.

6.2.1.1.8 Message Type

The 5-bit ***Message Type*** field ***Shall*** indicate the type of Message being sent. To fully decode the ***Message Type***, the ***Number of Data Objects*** field is first examined to determine whether the Message is a Control Message or a Data Message. Then the specific ***Message Type*** can be found in ***Table 6.5 "Control Message Types"*** or ***Table 6.6 "Data Message Types"***.

The ***Message Type*** field ***Shall*** apply to all SOP* Packet types.

6.2.1.2 Extended Message Header

Every Extended Message (indicated by the ***Extended*** field being set in the Message Header) ***Shall*** contain an Extended Message Header following the Message Header as shown in ***Figure 6-3 "USB Power Delivery Packet Format including an Extended Message Header and Payload"*** and defined in ***Table 6.3 "Extended Message Header"***.

The Extended Message Header is used to support Extended Messages containing Data Blocks of ***Data Size*** either sent in a single Message or as a series of Chunks. When the Data Block is sent as a series of Chunks, each Chunk in the series, except for the last Chunk, ***Shall*** contain ***MaxExtendedMsgChunkLen*** bytes. The last Chunk in the series ***Shall*** contain the remainder of the Data Block and so could be less than ***MaxExtendedMsgChunkLen*** bytes and ***Shall*** be padded to the next 4-byte Data Object boundary.

Table 6.3 "Extended Message Header"

Bit(s)	Start of Packet	Field Name	Reference
15	SOP*	***Chunked***	***Section 6.2.1.2.1***
14...11	SOP*	***Chunk Number***	***Section 6.2.1.2.2***
10	SOP*	***Request Chunk***	***Section 6.2.1.2.3***
9	SOP*	Reserved	***Section 1.4.2.10***
8...0	SOP*	***Data Size***	***Section 6.2.1.2.4***

6.2.1.2.1 Chunked

The Port Partners ***Shall*** use the Unchunked Extended Messages Supported fields in the ***Source_Capabilities*** Message and the ***Request*** Message to determine whether to send Messages of Data Size > ***MaxExtendedMsgLegacyLen*** bytes in a single Unchunked Extended Message (see ***Section 6.4.1.2.2.6 "Unchunked Extended Messages Supported"*** and ***Section 6.4.2.6 "Unchunked Extended Messages Supported"***).

When either Port Partner only supports Chunked Extended Messages:

1) The ***Chunked*** bit in every Extended Message ***Shall*** be set to one.

2) Every Extended Message of Data Size > ***MaxExtendedMsgLegacyLen*** ***Shall*** be transmitted between the Port Partners in Chunks

3) The ***Number of Data Objects*** in the Message Header ***Shall*** indicate the number of Data Objects in the Message padded to the 4-byte boundary including the Extended Header as part of the first Data Object.

Point 1, Point 2 and Point 3 above ***Shall*** apply until the Port Pair is Detached, there is a Hard Reset or the Source removes power (except during a Power Role Swap or Fast Role Swap when the initial Source removes power in order to for the new Source to apply power).

When both Port Partners support Unchunked Extended Messages:

1) The ***Chunked*** bit in every Extended Message ***Shall*** be set to zero.

2) Every Extended Message ***Shall*** be transmitted between the Port Partners Unchunked.

3) The ***Number of Data Objects*** in the Message Header is Reserved.

Point 1, Point 2 and Point 3 above ***Shall*** apply until the Port Pair is Detached, there is a Hard Reset or the Source removes power (except during a Power Role Swap or Fast Role Swap when the initial Source removes power in order to for the new Source to apply power).

When sending Extended Messages to the Cable Plug the VCONN Source ***Shall*** only send Chunked Messages. Cable Plugs ***Shall*** always send Extended Messages of Data Size > ***MaxExtendedMsgLegacyLen*** Chunked and ***Shall*** set the ***Chunked*** bit in every Extended Message to one.

When Extended Messages are supported Chunking ***Shall*** be supported.

6.2.1.2.2 Chunk Number

The ***Chunk Number*** field ***Shall*** only be ***Valid*** in a Message if the ***Chunked*** flag is set to one. if the ***Chunked*** flag is set to zero the ***Chunk Number*** field ***Shall*** also be set to zero.

The ***Chunk Number*** field is used differently depending on whether the Message is a request for Data, or a requested Data Block being returned:

In a request for data the ***Chunk Number*** field indicates the number of the Chunk being requested. The requestor ***Shall*** only set this field to the number of the next Chunk in the series (the next Chunk after the last received Chunk).

In the requested Data Block the ***Chunk Number*** field indicates the number of the Chunk being returned. The Chunk number for each Chunk in the series ***Shall*** start at zero and ***Shall*** increment for each Chunk by one up to a maximum of 9 corresponding to 10 Chunks in total.

6.2.1.2.3 Request Chunk

The ***Request Chunk*** bit ***Shall*** only be used for the Chunked transfer of an Extended Message when the ***Chunked*** bit is set to 1 (see ***Figure 6-7 "Example Security_Request sequence Chunked (Chunked bit = 1)"***). For Unchunked Extended Message transfers, Messages ***Shall*** be sent and received without the request/response mechanism (see ***Figure 6-4 "Example Security_Request sequence Unchunked (Chunked bit = 0)"***).

The ***Request Chunk*** bit ***Shall*** be set to one to indicate that this is a request for a Chunk of a Data Block and ***Shall*** be set to zero to indicate that this is a Chunk response containing a Chunk. Except for Chunk zero, a requested Chunk of a Data Block ***Shall*** only be returned as a Chunk response to a corresponding request for that Chunk. Both the Chunk request and the Chunk response ***Shall*** contain the same value in the ***Message Type*** field. When the ***Request Chunk*** bit is set to one the ***Data Size*** field ***Shall*** be zero.

6.2.1.2.4 Data Size

The *Data Size* field ***Shall*** indicate how many bytes of data in total are in Data Block being returned. The total number of data bytes in the Message ***Shall Not*** exceed *MaxExtendedMsgLen*.

If the *Data Size* field is less than *MaxExtendedMsgLegacyLen* and the *Chunked* bit is set then the Packet payload Shall be padded to the next 4-byte Data Object boundary with zeros (0x00).

If the *Data Size* field is greater than expected for a given Extended Message but less than or equal to *MaxExtendedMsgLen* then the expected fields in the Message ***Shall*** be processed appropriately and the additional fields ***Shall*** be ***Ignored***.

6.2.1.2.5 Extended Message Examples

The following examples illustrate the transmission of Extended Messages both Chunked (*Chunked* bit is one) and Unchunked (*Chunked* bit is zero). The examples use a *Security_Request* Message of *Data Size* 7 bytes which is responded to by a *Security_Response* Message of *Data Size* 30 bytes. The sizes of these Messages are arbitrary and are used to illustrate Message transmission; they are not intended to correspond to genuine security related Messages.

During negotiation of the Explicit Contract after connection, the Port Partners use the Unchunked Extended Messages Supported fields in the *Source_Capabilities* Message and the *Request* Message to determine the value of the *Chunked* bit (see *Table 6.4 "Use of Unchunked Message Supported bit"*). When both Port Partners support Unchunked Messages then the *Chunked* bit is zero otherwise the *Chunked* bit is one.

- The *Chunked* bit is used to determine whether:
  - The Chunk request/response mechanism is used.
  - Extended Messages are Chunked.
  - Padding is applied.
  - The *Number of Data Objects* field is used.

The following examples illustrate the expected usage in each case.

Table 6.4 "Use of Unchunked Message Supported bit"

		Source: *Source_Capabilities* Message	
		Unchunked Message Supported bit = 0	Unchunked Message Supported bit = 1
Sink: *Request* Message	Unchunked Message Supported bit = 0	*Chunked* bit = 1	*Chunked* bit = 1
	Unchunked Message Supported bit = 1	*Chunked* bit = 1	*Chunked* bit = 0

6.2.1.2.5.1 Security_Request/Security_Response Unchunked Example

Figure 6-4 "Example Security_Request sequence Unchunked (Chunked bit = 0)" illustrates a typical sequence for a *Security_Request* Message responded to by a *Security_Response* Message using Unchunked Extended Messages (*Chunked* bit is zero) between a USB Host and a power brick. The entire Data Block is returned in one Message. The Chunk request/response mechanism is not used.

Figure 6-4 "Example Security_Request sequence Unchunked (Chunked bit = 0)"

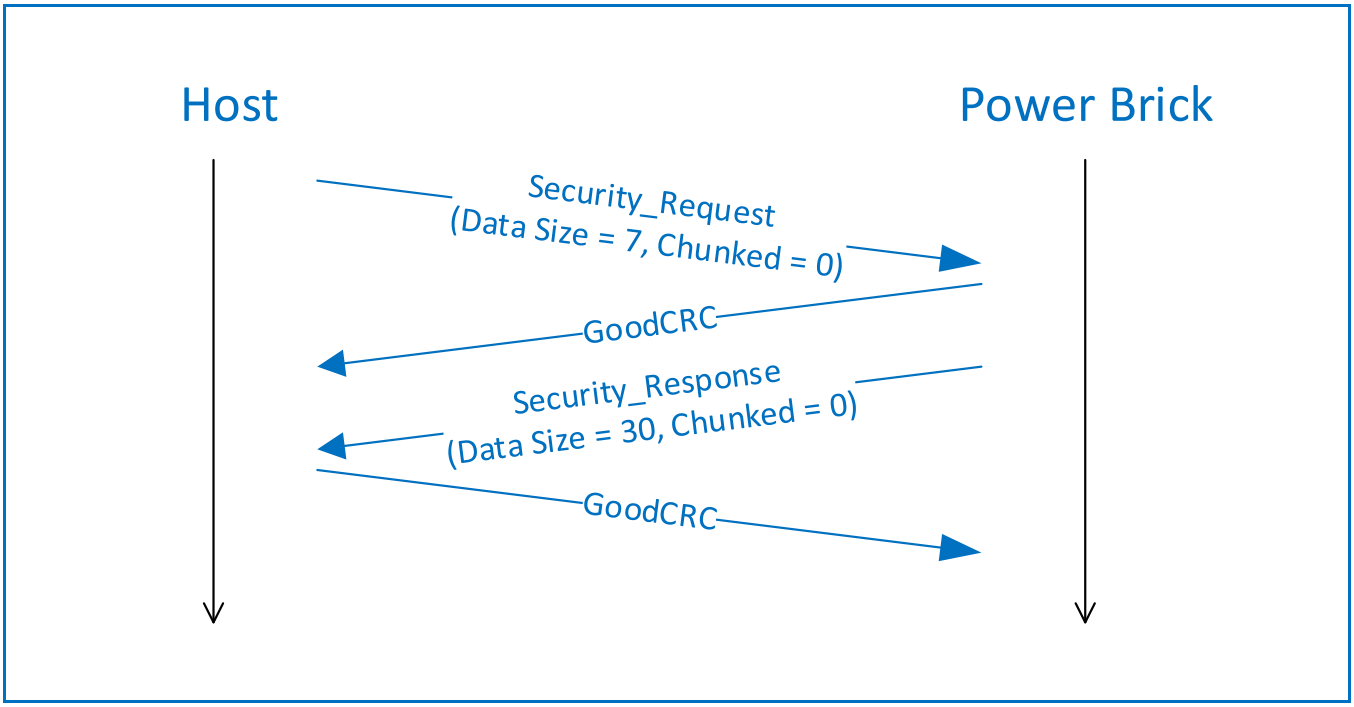


Figure 6-5 "Example byte transmission for Security_Request Message of Data Size 7 (Chunked bit is set to zero)" details the ***Security_Request*** Message shown in *Figure 6-4 "Example Security_Request sequence Unchunked (Chunked bit = 0)"*. The figure shows the byte ordering on the bus as well as the fact that there is no padding in this case. The ***Number of Data Objects*** field has a value of 0 since it is ***Reserved*** when the ***Chunked*** bit is zero. The ***Data Size field*** indicates the length of the Extended Message when the ***Chunked*** bit is set to 0, which in this case is 7 bytes.

Figure 6-5 "Example byte transmission for Security_Request Message of Data Size 7 (Chunked bit is set to zero)"

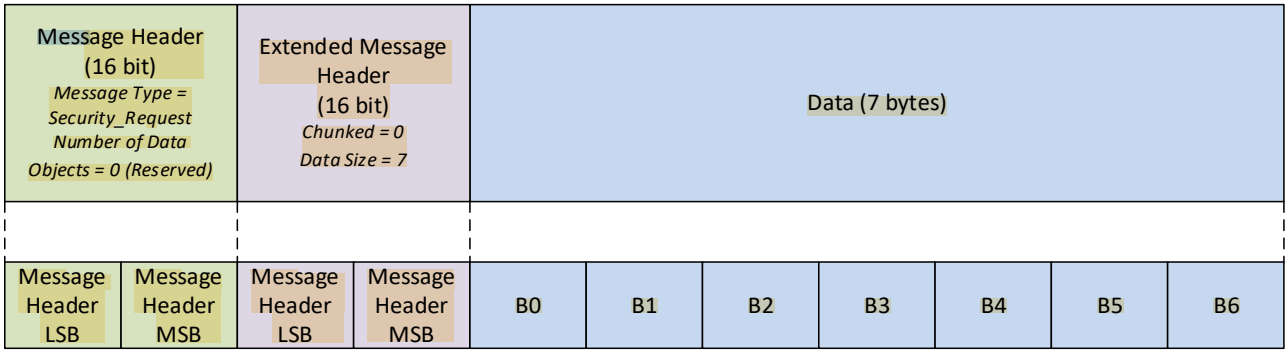
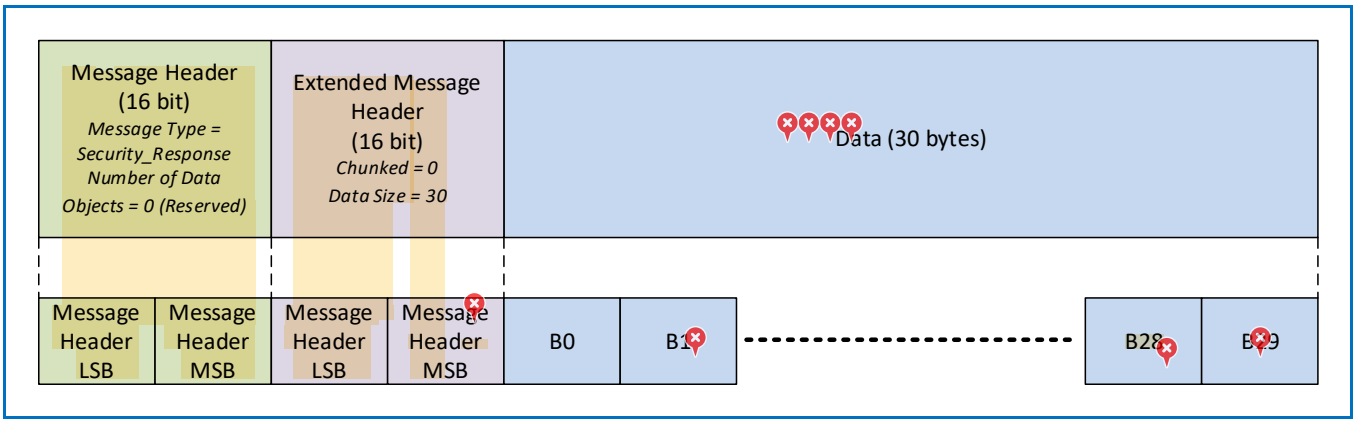


Figure 6-6 "Example byte transmission for Security_Response Message of Data Size 7 (Chunked bit is set to zero)" details the ***Security_Response*** Message shown in *Figure 6-4 "Example Security_Request sequence Unchunked (Chunked bit = 0)"*. The figure shows the byte ordering on the bus as well as the fact that there is no padding in this case. The ***Number of Data Objects*** field has a value of 0 since it is ***Reserved*** when the ***Chunked*** bit is zero. The ***Data Size field*** indicates the length of the Extended Message when the ***Chunked*** bit is set to zero, which in this case is 30 bytes.

Figure 6-6 "Example byte transmission for Security_Response Message of Data Size 7 (Chunked bit is set to zero)"



6.2.1.2.5.2 Security_Request/Security_Response Chunked Example

Figure 6-7 "Example Security_Request sequence Chunked (Chunked bit = 1)" illustrates a typical sequence for a ***Security_Request*** Message responded to by a ***Security_Response*** Message using Chunked Extended Messages (***Chunked*** bit is one) between a USB Host and a power brick. Note that ***Chunk Number*** zero in every Extended Message is sent without the need for a Chunk Request, but ***Chunk Number*** one and following need to be requested with a Chunk request.

Figure 6-7 "Example Security_Request sequence Chunked (Chunked bit = 1)"

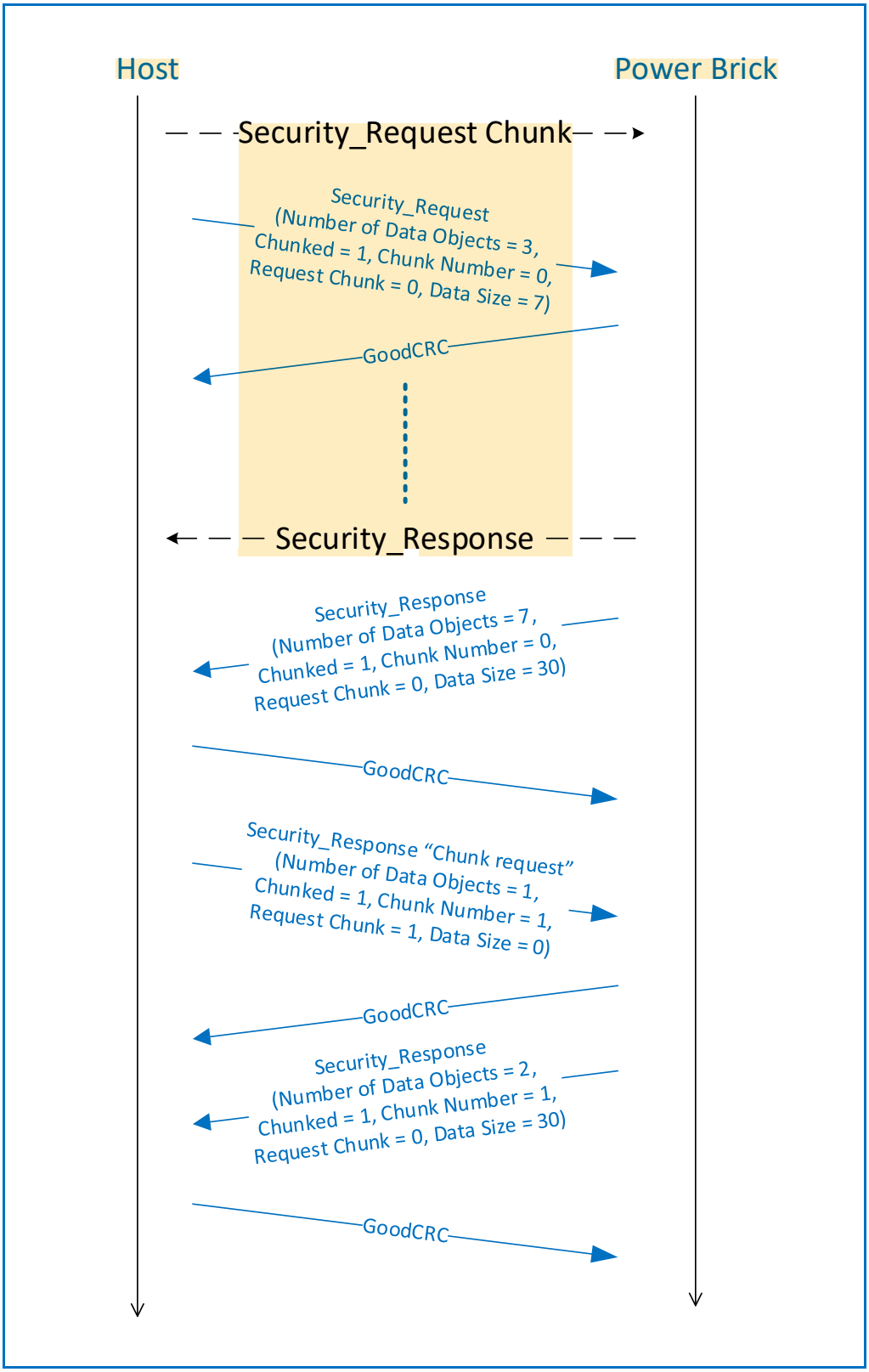


Figure 6-8 "Example Security_Request Message of Data Size 7 (Chunked bit set to 1)" shows the ***Security_Request*** Message shown in *Figure 6-7 "Example Security_Request sequence Chunked (Chunked bit = 1)"* in more detail including the byte ordering on the bus and padding. Three bytes of padding have been added to the Message so that the total number of bytes is a multiple of 32-bits, corresponding to 3 Data Objects. The ***Number of Data Objects*** field is set to 3 to indicate the length of this Chunk. The ***Chunk Number*** is set to zero and the ***Data Size*** field is set to 7 to indicate the length of the whole Extended Message.

Figure 6-8 "Example Security_Request Message of Data Size 7 (Chunked bit set to 1)"

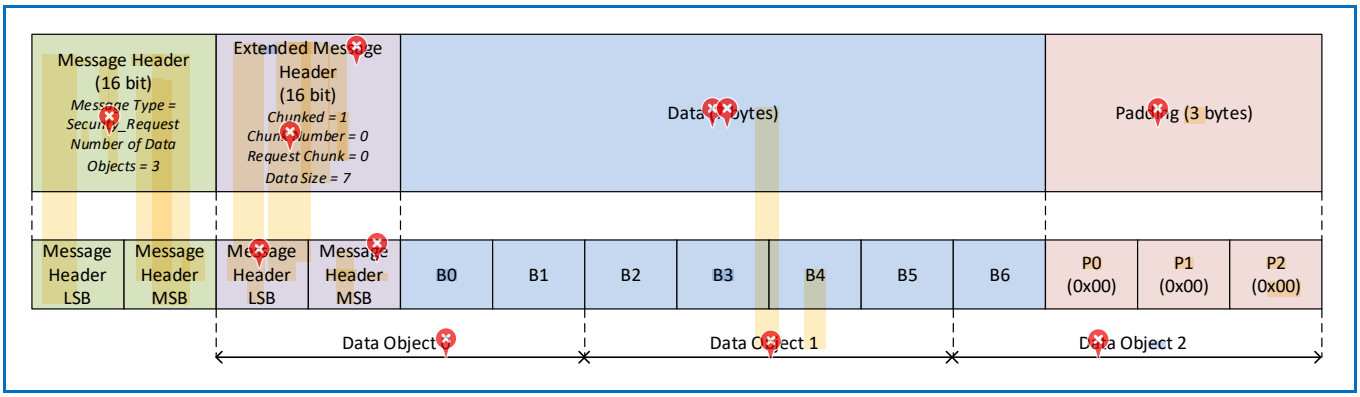


Figure 6-9 "Example Chunk 0 of Security_Response Message of Data Size 30 (Chunked bit set to 1)" shows ***Chunk Number*** zero of the ***Security_Response*** Message shown in *Figure 6-7 "Example Security_Request sequence Chunked (Chunked bit = 1)"* in more detail including the byte ordering on the bus and padding. No padding is need for this Chunk since the full 26-byte payload plus 2-byte Extended Message Header is a multiple of 32-bits, corresponding to 7 Data Objects. The ***Number of Data Objects*** field is set to 7 to indicate the length of this Chunk and the ***Data Size*** field is set to 30 to indicate the length of the whole Extended Message.

Figure 6-9 "Example Chunk 0 of Security_Response Message of Data Size 30 (Chunked bit set to 1)"

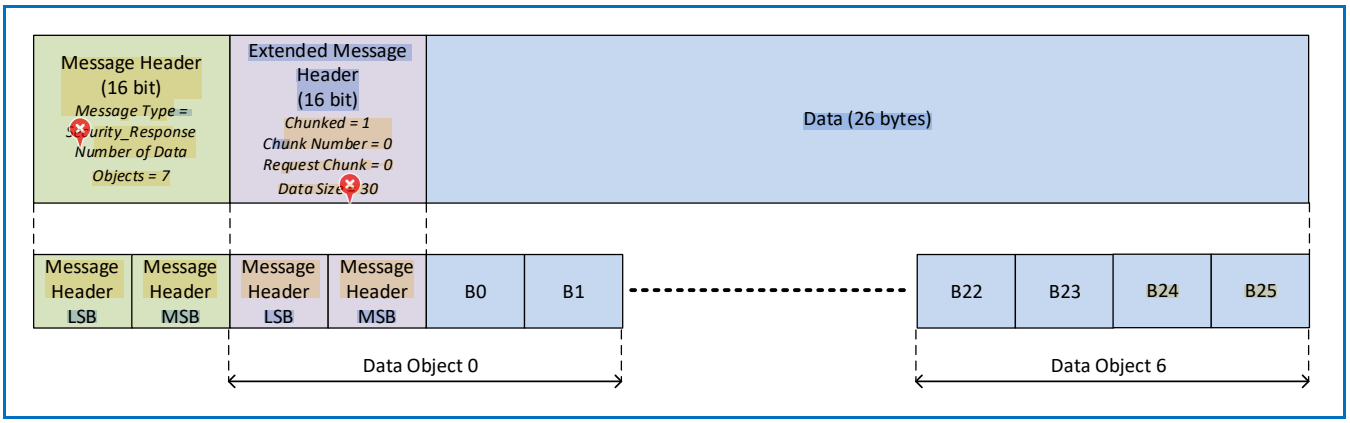


Figure 6-10 "Example byte transmission for a Security_Response Message Chunk request (Chunked bit is set to 1)" shows an example of the Message format, byte ordering and padding for the ***Security_Response*** Message Chunk request for ***Chunk Number*** one shown in *Figure 6-7 "Example Security_Request sequence Chunked (Chunked bit = 1)"*. In the Chunk request the ***Number of Data Objects*** field in the Message is set to 1 to indicate that the payload is 32 bits equivalent to 1 data object. Since the ***Chunked*** bit is set to 1 the Chunk request/Chunk response mechanism is used. The Message is a Chunk request so the ***Request Chunk*** bit is set to one, and in this case Chunk one is being requested so ***Chunk Number*** is set to one. ***Data Size*** is set to zero indicating the length of the Data Block being transferred. Two bytes of padding are added to ensure that the payload is a multiple of 32 bits.

Figure 6-10 "Example byte transmission for a Security_Response Message Chunk request (Chunked bit is set to 1)"

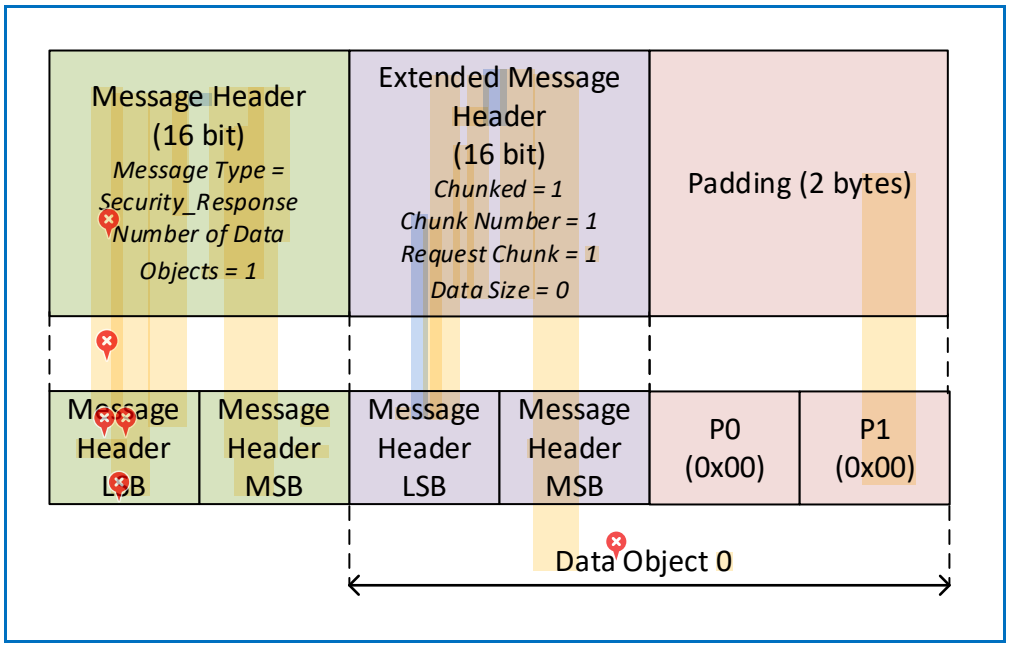
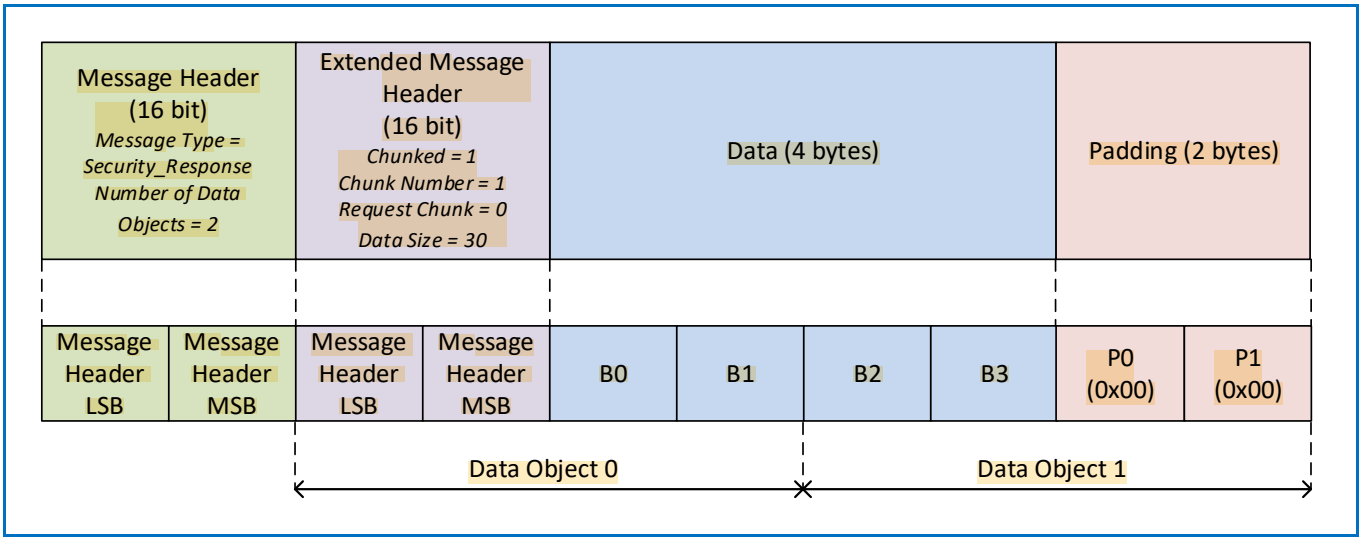


Figure 6-11 "Example Chunk 1 of Security_Response Message of Data Size 30 (Chunked bit set to 1)" shows ***Chunk Number*** one of the ***Security_Response*** Message shown in *Figure 6-7 "Example Security_Request sequence Chunked (Chunked bit = 1)"* in more detail including the byte ordering on the bus and padding. Two bytes of padding are added to ensure that the payload is a multiple of 32 bits, corresponding to 2 Data Objects. The ***Number of Data Objects*** field is set to 2 to indicate the length of this Chunk and the ***Data Size*** field is set to 30 to indicate the length of the whole Extended Message.

Figure 6-11 "Example Chunk 1 of Security_Response Message of Data Size 30 (Chunked bit set to 1)"



6.3 Control Message

A Message is defined as a Control Message when the ***Number of Data Objects*** field in the Message Header is set to zero. The Control Message consists only of a Message Header and a CRC. The Protocol Layer originates the Control Messages (i.e., ***Accept*** Message, ***Reject*** Message etc.).

The Control Message types are specified in the Message Header's ***Message Type*** field (bits 4...0) and are summarized in ***Table 6.5 "Control Message Types".*** The Sent by column indicates entities which ***May*** send the given Message (Source, Sink or Cable Plug); entities not listed ***Shall Not*** issue the corresponding Message. The "***Valid*** Start of Packet" column indicates the Messages which ***Shall*** only be issued in SOP Packets and the Messages which ***May*** be issued in SOP* Packets.

Table 6.5 "Control Message Types"

Bits 4...0	Message Type	Sent by	Description	Valid Start of Packet
0_0000	***Reserved***	***N/A***	All values not explicitly defined are ***Reserved*** and ***Shall Not*** be used.	
0_0001	***GoodCRC***	Source, Sink or Cable Plug	***Section 6.3.1.***	SOP*
0_0010	***GotoMin***	Source only	***Section 6.3.2.***	SOP only
0_0011	***Accept***	Source, Sink or Cable Plug	***Section 6.3.3.***	SOP*
0_0100	***Reject***	Source, Sink or Cable Plug	***Section 6.3.4.***	SOP*
0_0101	***Ping***	Source only	***Section 6.3.5.***	SOP only
0_0110	***PS_RDY***	Source or Sink	***Section 6.3.6.***	SOP only
0_0111	***Get_Source_Cap***	Sink or DRP	***Section 6.3.7.***	SOP only
0_1000	***Get_Sink_Cap***	Source or DRP	***Section 6.3.8.***	SOP only
0_1001	***DR_Swap***	Source or Sink	***Section 6.3.9***	SOP only
0_1010	***PR_Swap***	Source or Sink	***Section 6.3.10***	SOP only
0_1011	***VCONN_Swap***	Source or Sink	***Section 6.3.11***	SOP only
0_1100	***Wait***	Source or Sink	***Section 6.3.12***	SOP only
0_1101	***Soft_Reset***	Source or Sink	***Section 6.3.13***	SOP*
0_1110	***Data_Reset***	Source or Sink	***Section 6.3.14***	SOP only
0_1111	***Data_Reset_Complete***	Source or Sink	***Section 6.3.15***	SOP only
1_0000	***Not_Supported***	Source, Sink or Cable Plug	***Section 6.3.16***	SOP*
1_0001	***Get_Source_Cap_Extended***	Sink or DRP	***Section 6.3.17***	SOP only
1_0010	***Get_Status***	Source or Sink	***Section 6.3.18***	SOP*
1_0011	***FR_Swap***	Sink[1]	***Section 6.3.19***	SOP only
1_0100	***Get_PPS_Status***	Sink	***Section 6.3.20***	SOP only
1_0101	***Get_Country_Codes***	Source or Sink	***Section 6.3.21***	SOP only

1_0110	*Get_Sink_Cap_Extended*	Source or DRP	*Section 6.3.22*	SOP only
1_0111	*Get_Source_Info*	Sink or DRP	*Section 6.3.23*	SOP Only
1_1000	*Get_Revision*	Source or Sink	*Section 6.3.24*	SOP*
1_1001… 1_1111	***Reserved***	***N/A***	All values not explicitly defined are ***Reserved*** and ***Shall Not*** be used.	

1) In this case the Port is providing *vSafe5V* however it will have R_d asserted rather than R_p and sets the *Port Power Role* field to Sink, until the Fast Role Swap AMS has completed.

6.3.1 GoodCRC Message

The *GoodCRC* Message ***Shall*** be sent by the receiver to acknowledge that the previous Message was correctly received (i.e., had a good CRC). The *GoodCRC* Message ***Shall*** return the Message's *MessageID* so the transmitter can determine that the correct Message is being acknowledged. The first bit of the *GoodCRC* Message ***Shall*** be returned within *tTransmit* after receipt of the last bit of the previous Message.

BIST does not send the *GoodCRC* Message while in a Continuous BIST Mode (see *Section 6.4.3 "BIST Message"*).

The retry mechanism is triggered when the Message sender fails to receive a *GoodCRC* Message before the *CRCReceiveTimer* expires. It is used by the Message sender to detect that the Message was not correctly received by the Message recipient due to noise or other disturbance on the Configuration Channel (CC). The retry mechanism ***Shall Not*** be used for any other purpose such as a means of gaining time for processing the required response to the received Message.

6.3.2 GotoMin Message

The *GotoMin* Message applies only to those Sinks that have requested power with the GiveBack capable flag set in the Sink Request Data Object.

It is a directive to the Sink Port to reduce its operating power level to the amount specified in the Minimum Operating Current field of its latest Sink Request Data Object.

The GotoMin process is designed to allow the Source to temporarily reallocate power to meet a short-term requirement. For example, a Source can reduce a Sink's power consumption for 10-20 seconds to allow another Sink (e.g., an HDD to spin up).

The Source sends this Message to harvest power to meet a request for power that it cannot otherwise meet. The Device Policy Manager determines which Port or ports will receive the Message.

The Sink ***Shall*** respond to a *GotoMin* Message by reducing its power consumption to less than or equal to the pre-negotiated value (Minimum Operating Current) within *tSnkNewPower* time.

The Source sends a *GotoMin* Message as a shortcut in the power negotiation process since the Source and Sink have already made a Contract with respect to the power to be returned. The Source does not have to Advertise its Capabilities and the Sink does not have to make a Request based on them. The Source simply sends the *GotoMin* Message in place of the *Accept* Message normally sent during the power negotiation process (see step 19 in *Figure 8-5 "Successful Fixed, Variable or Battery SPR Power Negotiation"*). The power negotiation process then completes from this point in the normal manner with the Source sending a *PS_RDY* Message once the power supply transition is complete. The steps of the GotoMin process are fully described in *Figure 8-8 "Successful GotoMin operation"*.

The Source ***Shall*** return power to the Sink(s) it has 'borrowed' from using the GotoMin mechanism before it can allocate any 'new' power to other devices.

6.3.3 Accept Message

The ***Accept*** Message is a ***Valid*** response in the following cases:

- It ***Shall*** be sent by the Source, in SPR Mode, to signal the Sink that the Source is willing to meet the ***Request*** Message.
- It ***Shall*** be sent by the Source, in EPR Mode, to signal the Sink that the Source is willing to meet the ***EPR_Request*** Message.
- It ***Shall*** be sent by the recipient of the ***PR_Swap*** Message to signal that it is willing to do a Power Role Swap and has begun the Power Role Swap sequence.
- It ***Shall*** be sent by the recipient of the ***DR_Swap*** Message to signal that it is willing to do a Data Role Swap and has begun the Data Role Swap sequence.
- It ***Shall*** be sent by the recipient of the ***VCONN_Swap*** Message to signal that it is willing to do a VCONN Swap and has begun the VCONN Swap sequence.
- It ***Shall*** be sent by the recipient of the ***FR_Swap*** Message to indicate that it has begun the Fast Role Swap sequence.
- It ***Shall*** be sent by the recipient of the ***Soft_Reset*** Message to indicate that it has completed its Soft Reset.
- It ***Shall*** be sent by the recipient of the ***Enter_USB*** Message to indicate that it has begun the Enter USB Sequence.
- It ***Shall*** be sent by the recipient of the ***Data_Reset*** Message to indicate that it has begun the Data Reset Sequence.

The ***Accept*** Message ***Shall*** be sent within ***tReceiverResponse*** of the receipt of the last bit of the Message (see ***Section 6.6.2 "SenderResponseTimer"***).

6.3.4 Reject Message

The ***Reject*** Message is a ***Valid*** response in the following cases:

- It ***Shall*** be sent to signal the Sink, in SPR Mode, that the Source is unable to meet the ***Request*** Message. This ***May*** be due an ***Invalid*** request or because the Source can no longer provide what it previously Advertised.
- It ***Shall*** be sent to signal the Sink, in EPR Mode, that the Source is unable to meet the ***EPR_Request*** Message. This ***May*** be due an ***Invalid*** request or because the Source can no longer provide what it previously Advertised.
- It ***Shall*** be sent by the recipient of a ***PR_Swap*** Message to indicate it is unable to do a Power Role Swap.
- It ***Shall*** be sent by the recipient of a ***DR_Swap*** Message to indicate it is unable to do a Data Role Swap.
- It ***Shall*** be sent by the recipient of a ***VCONN_Swap*** Message that is not presently the VCONN Source, to indicate it is unable to do a VCONN Swap.
- It ***Shall*** be sent by UFP on receiving an ***Enter_USB*** Message to indicate it is unable to enter the requested USB Mode.

The sender of a ***Request***, ***EPR_Request***, ***PR_Swap***, ***DR_Swap***, ***VCONN_Swap***, or ***Enter_USB*** Message, on receiving a ***Reject*** Message response, ***Shall Not*** send this same Message to the recipient until one of the following has occurred:

- A New Explicit Contract negotiation as a result of the Source sending a ***Source_Capabilities*** Message or ***EPR_Source_Capabilities*** Message. This can be triggered by:
- The Source's Device Policy Manager.
- A ***Get_Source_Cap*** Message sent from the Sink to the Source in SPR Mode.
- An ***EPR_Get_Source_Cap*** Message sent from the Sink to the Source in EPR Mode.
- A Power Role swap.
- A Soft Reset.
- A Hard Reset.
- A Disconnect/Re-connect.
- A Data Role Swap.
- A Data Reset.

The Sink ***May*** send a different ***Request*** Message to the one which was rejected but ***Shall Not*** repeat the same ***Request*** Message, using the same RDO, unless there has been a New Explicit Contract negotiation, Data Role Swap or Data Reset as described above.

The ***Reject*** Message ***Shall*** be sent within ***tReceiverResponse*** of the receipt of the last bit of Message (see ***Section 6.6.2 "SenderResponseTimer"***).

Note: the ***Reject*** Message is not a ***Valid*** response when a Message is not supported. In this case the ***Not_Supported*** Message is returned (see ***Section 6.3.16 "Not_Supported Message"***).

6.3.5 Ping Message

The ***Ping*** Message was previously used on USB Type-A and USB Type-B connectors to determine the continued presence of the Sink when no other messaging was taking place. USB Type-C® connectors have a mechanism to determine Sink presence so when the Port Partners are both connected using USB Type-C® connectors the ***Ping*** Message is not necessary but ***May*** be sent by a Source if desired. A Sink using a USB Type-C® connector cannot expect to receive ***Ping*** Messages but ***Shall Not*** treat ***Ping*** Messages as an error if they are received.

6.3.6 PS_RDY Message

The ***PS_RDY*** Message ***Shall*** be sent by the Source (or by both the new Sink and new Source during the Power Role Swap sequence or Fast Role Swap sequence) to indicate its power supply has reached the desired operating condition (see ***Section 8.3.2.2 "Power Negotiation"***).

6.3.7 Get_Source_Cap Message

The ***Get_Source_Cap*** (Get Source Capabilities) Message ***May*** be sent by a Port to request the Source Capabilities and Dual-Role Power capability of its Port Partner (e.g., Dual-Role Power capable). The Port ***Shall*** respond by returning a ***Source_Capabilities*** Message (see ***Section 6.4.1.1.1 "Use by Sources"***).

6.3.8 Get_Sink_Cap Message

The ***Get_Sink_Cap*** (Get Sink Capabilities) Message ***May*** be sent by a Port to request the Sink Capabilities and Dual-Role Power capability of its Port Partner (e.g., Dual-Role Power capable). The Port ***Shall*** respond by returning a ***Sink_Capabilities*** Message (see ***Section 6.4.1.1.2 "Use by Sinks"***).

6.3.9 DR_Swap Message

The ***DR_Swap*** Message is used to exchange DFP and UFP operation between Port Partners while maintaining the direction of power flow over V_{BUS}. The DR_Swap process can be used by Port Partners whether or not they support USB Communications capability. A DFP that supports USB Communication Capability starts as the USB Host on Attachment. A UFP that supports USB Communication Capability starts as the USB Device on Attachment.

[USB Type-C 2.3] DRDs ***Shall*** have the capability to perform a Data Role Swap from the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** states. DFPs and UFPs ***May*** have the capability to perform a Data Role Swap from the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** states. A Data Role Swap ***Shall*** be regarded in the same way as a cable Detach/re-Attach in relation to any USB communication which is ongoing between the Port Partners. If there are any *Active Mode*s between the Port Partners when a ***DR_Swap*** Message is a received, then a Hard Reset ***Shall*** be performed (see ***Section 6.4.4.3.4 "Enter Mode Command"***). If the Cable Plug has any *Active Mode*s then the DFP ***Shall Not*** issue a ***DR_Swap*** Message and ***Shall*** cause all *Active Mode*s in the Cable Plug to be exited before accepting a DR Swap request.

The Source of V_{BUS} and V_{CONN} Source ***Shall*** remain unchanged as well as the R_p/R_d resistors on the CC wire during the Data Role Swap process.

The ***DR_Swap*** Message ***May*** be sent by either Port Partner. The recipient of the ***DR_Swap*** Message ***Shall*** respond by sending an ***Accept*** Message, a ***Wait*** Message or a ***Reject*** Message (see ***Section 6.9 "Accept, Reject and Wait"***).

- If an ***Accept*** Message is sent, the Source and Sink ***Shall*** exchange operational roles.
- If a ***Reject*** Message is sent, the requester is informed that the recipient is unable, or unwilling, to do a Data Role Swap and no action ***Shall*** be taken.
- If a ***Wait*** Message is sent, the requester is informed that a Data Role Swap might be possible in the future but that no immediate action ***Shall*** be taken.

Before a Data Role Swap the initial DFP ***Shall*** have its ***Port Data Role*** bit set to DFP, and the initial UFP ***Shall*** have its ***Port Data Role*** bit set to UFP.

After a successful Data Role Swap the DFP/Host ***Shall*** become the UFP/Device and vice-versa; the new DFP ***Shall*** have its ***Port Data Role*** bit set to DFP, and the new UFP ***Shall*** have its ***Port Data Role*** bit set to UFP. Where USB Communication is supported by both Port Partners a USB data connection ***Should*** be established according to the new data roles.

If the Data Role Swap, after having been accepted by the Port Partner, is subsequently not successful, in order to attempt a re-establishment of the connection on the CC Wire, USB Type-C® Error Recovery actions, such as disconnect, as defined in ***[USB Type-C 2.3]*** will be necessary.

See ***Section 8.3.2.10 "Data Role Swap"***.

6.3.10 PR_Swap Message

The ***PR_Swap*** Message ***May*** be sent by either Port Partner to request an exchange of power roles. The recipient of the Message ***Shall*** respond by sending an ***Accept*** Message, a ***Wait*** Message or a ***Reject*** Message (see ***Section 6.9 "Accept, Reject and Wait"***).

- If an ***Accept*** Message is sent, the Source and Sink ***Shall*** do a Power Role Swap.
- If a ***Reject*** Message is sent, the requester is informed that the recipient is unable, or unwilling, to do a Power Role Swap and no action ***Shall*** be taken.
- If a ***Wait*** Message is sent, the requester is informed that a Power Role Swap might be possible in the future but that no immediate action ***Shall*** be taken.

After a successful Power Role Swap the Port Partners ***Shall*** reset their respective Protocol Layers (equivalent to a Soft Reset): resetting their ***MessageIDCounter***, ***RetryCounter*** and Protocol Layer state machines before attempting to establish an Explicit Contract. At this point the Source ***Shall*** also reset its ***CapsCounter***.

The Source ***Shall*** have R_p asserted on the CC wire and the Sink ***Shall*** have R_d asserted on the CC wire as defined in ***[USB Type-C 2.3]***. When performing a Power Role Swap from Source to Sink, the Port ***Shall*** change its CC Wire resistor from R_p to R_d. When performing a Power Role Swap from Sink to Source, the Port ***Shall*** change its CC Wire resistor from R_d to R_p. The DFP (Host), UFP (Device) roles and VCONN Source ***Shall*** remain unchanged during the Power Role Swap process.

Note: during the Power Role Swap process the initial Sink does not disconnect even though V_{BUS} drops below ***vSafe5V***.

For more information regarding the Power Role Swap, refer to:

- ***Section 7.3.2 "Transitions Caused by Power Role Swap"*** in the Power Supply chapter
- ***Section 8.3.2.6 "Data Reset"***
- ***Section 8.3.3.20.3 "Policy Engine in Source to Sink Power Role Swap State Diagram"***
- ***Section 8.3.3.20.4 "Policy Engine in Sink to Source Power Role Swap State Diagram"***
- ***Section 9.1.2 "Mapping to USB Device States"*** for V_{BUS} mapping to USB states.

6.3.11 VCONN_Swap Message

The ***VCONN_Swap*** Message ***Shall*** be supported by any Port that can operate as a VCONN Source.

The ***VCONN_Swap*** Message ***May*** be sent by either Port Partner to request an exchange of VCONN Source. The recipient of the Message ***Shall*** respond by sending an ***Accept*** Message, ***Reject*** Message, ***Wait*** Message (see ***Section 6.9 "Accept, Reject and Wait"***) or ***Not_Supported*** Message.

- If an ***Accept*** Message is sent, the Port Partners ***Shall*** perform a VCONN Swap. The new VCONN Source ***Shall*** send a ***PS_RDY*** Message within ***tVCONNSourceOn*** to indicate that it is now sourcing VCONN. The initial VCONN Source ***Shall*** cease sourcing VCONN within ***tVCONNSourceOff*** of receipt of the last bit of the ***EOP*** of the ***PS_RDY*** Message.
- If a ***Reject*** Message is sent, the requester is informed that the recipient is unable, or unwilling, to do a VCONN Swap and no action ***Shall*** be taken. A ***Reject*** Message ***Shall*** only be sent by the Port that is not presently the Vconn Source in response to a ***VCONN_Swap*** Message. The Port that is presently the Vconn Source ***Shall Not*** send a ***Reject*** Message in response to ***VCONN_Swap*** Message.
- If a ***Wait*** Message is sent, the requester is informed that a VCONN Swap might be possible in the future but that no immediate action ***Shall*** be taken. A ***Wait*** Message ***Shall*** only be sent by the Port that is not presently the Vconn Source in response to a ***VCONN_Swap*** Message. The Port that is presently the Vconn Source ***Shall Not*** send a ***Wait*** Message in response to ***VCONN_Swap*** Message.
- If a ***Not_Supported*** Message is sent, the requester is informed that VCONN Swap is not supported. The Port that is not presently the Vconn Source ***May*** turn on VCONN when a ***Not_Supported*** Message is received in response to a ***VCONN_Swap*** Message.

The DFP (Host), UFP (Device) roles and Source of V_{BUS} ***Shall*** remain unchanged as well as the R_p/R_d resistors on the CC wire during the VCONN Swap process.

VCONN ***Shall*** be continually sourced during the VCONN Swap process to maintain power to the Cable Plug(s) i.e., make before break.

Before communicating with a Cable Plug a Port ***Shall*** ensure that it is the VCONN Source and that the Cable Plugs are powered, by performing a VCONN swap if necessary. Since it cannot be guaranteed that the present VCONN Source is supplying VCONN, the only means to ensure that the Cable Plugs are powered is for a Port wishing to communicate with a Cable Plug to become the VCONN Source. If a ***Not_Supported*** Message is returned in response to the ***VCONN_Swap*** Message, then the Port is allowed to become the VCONN Source until a Hard Reset or Detach.

A VCONN Source that is also a Source can attempt to send a ***Discover Identity*** Command using SOP' to a Cable Plug prior to the establishment of an Explicit Contract.

Note: even when it is presently the VCONN Source, the Sink is not permitted to initiate an AMS with a Cable Plug unless R_p is set to ***SinkTxOk*** (see ***Section 6.9 "Accept, Reject and Wait"***).

6.3.12 Wait Message

The ***Wait*** Message is a ***Valid*** response to one of the following Messages:

- It ***Shall*** be sent to signal the Sink, in response to a ***Request*** Message in SPR Mode during Power Negotiation, to indicate that the Source is currently unable to meet the request.
- It ***Shall*** be sent to signal the Sink, in response to a ***EPR_Request*** Message in EPR Mode during Power Negotiation, to indicate that the Source is currently unable to meet the request.
- It ***Shall*** be sent by the recipient of a ***PR_Swap*** Message to indicate it is currently unable to do a Power Role Swap.
- It ***Shall*** be sent by the recipient of a ***DR_Swap*** Message to indicate it is currently unable to do a Data Role Swap.
- It ***Shall*** be sent by the recipient of a ***VCONN_Swap*** Message that is not presently the VCONN Source to indicate it is currently unable to do a VCONN Swap.
- It ***Shall*** be sent by the recipient of an ***Enter_USB*** Message to indicate it is currently unable to enter the requested USB Mode.

The ***Wait*** Message ***Shall*** be sent within ***tReceiverResponse*** of the receipt of the last bit of the Message (see ***Section 6.9 "Accept, Reject and Wait"***).

6.3.12.1 Wait in response to a Request Message

The ***Wait*** Message is used by the Source when a Sink that has ***Reserved*** power, requests it. The ***Wait*** Message allows the Source time to recover the power it requires to meet the request through the GotoMin process. A Source ***Should*** only send a ***Wait*** Message in response to a ***Request*** Message when an Explicit Contract exists between the Port Partners.

The Sink is allowed to repeat the ***Request*** Message using the ***SinkRequestTimer*** and ***Shall*** ensure that there is ***tSinkRequest*** after receiving the ***Wait*** Message before sending another ***Request*** Message.

6.3.12.2 Wait in response to a PR_Swap Message

The ***Wait*** Message is used when responding to a ***PR_Swap*** Message to indicate that a Power Role Swap might be possible in the future. This can occur in any case where the device receiving the ***PR_Swap*** Message needs to evaluate the request further e.g., by requesting Capabilities from the originator of the ***PR_Swap*** Message. Once it has completed this evaluation one of the Port Partners ***Should*** initiate the Power Role Swap process again by sending a ***PR_Swap*** Message.

The ***Wait*** Message is also used where a Hub is operating in hybrid mode when a request cannot be satisfied (see ***[UCSI]***).

A Port that receives a ***Wait*** Message in response to a ***PR_Swap*** Message ***Shall*** wait ***tPRSwapWait*** after receiving the ***Wait*** Message before sending another ***PR_Swap*** Message.

6.3.12.3 Wait in response to a DR_Swap Message

The ***Wait*** Message is used when responding to a ***DR_Swap*** Message to indicate that a Date Role Swap might be possible in the future. This can occur in any case where the device receiving the ***DR_Swap*** Message needs to evaluate the request further. Once it has completed this evaluation one of the Port Partners ***Should*** initiate the Data Role Swap process again by sending a ***DR_Swap*** Message.

A Port that receives a ***Wait*** Message in response to a ***DR_Swap*** Message ***Shall*** wait ***tDRSwapWait*** after receiving the ***Wait*** Message before sending another ***DR_Swap*** Message.

6.3.12.4 Wait in response to a VCONN_Swap Message

The ***Wait*** Message is used when responding to a ***VCONN_Swap*** Message to indicate that a ***VCONN_Swap*** might be possible in the future. This can occur in any case where the device receiving the ***VCONN_Swap*** Message needs to evaluate the request further. A ***Wait*** Message ***Shall*** only be sent by the Port that is not presently the VCONN Source in response to a ***VCONN_Swap*** Message. The Port that is presently the VCONN Source ***Shall Not*** send a ***Wait*** Message in response to ***VCONN_Swap*** Message. Once it has completed this evaluation one of the Port Partners ***Should*** initiate the VCONN Swap process again by sending a ***VCONN_Swap*** Message.

A Port that receives a ***Wait*** Message in response to a ***VCONN_Swap*** Message ***Shall*** wait ***tVCONNSwapWait*** after receiving the ***Wait*** Message before sending another ***VCONN_Swap*** Message.

6.3.12.5 Wait in response to an Enter_USB Message

The ***Wait*** Message is used, by the UFP, when responding to an ***Enter_USB*** Message to indicate that entering the requested USB Mode might be possible in the future. This can occur, for example, in any case where the UFP needs to negotiate more power to enter the mode. Once the UFP has completed this the DFP ***Should*** initiate the Enter USB process again by sending an ***Enter_USB*** Message.

A DFP that receives a ***Wait*** Message in response to an ***Enter_USB*** Message ***Shall*** wait ***tEnterUSBWait*** after receiving the ***Wait*** Message before sending another ***Enter_USB*** Message.

6.3.13 Soft Reset Message

A ***Soft_Reset*** Message ***May*** be initiated by either the Source or Sink to its Port Partner requesting a Soft Reset. The ***Soft_Reset*** Message ***Shall*** cause a Soft Reset of the connected Port Pair (see ***Section 6.8.1 "Soft Reset and Protocol Error"***). If the ***Soft_Reset*** Message fails a Hard Reset ***Shall*** be initiated within ***tHardReset*** of the last ***CRCReceiveTimer*** expiring after ***nRetryCount*** retries have been completed.

A ***Soft_Reset*** Message is used to recover from Protocol Layer errors; putting the Message counters to a known state to regain Message synchronization. The ***Soft_Reset*** Message has no effect on the Source or Sink; that is the previously negotiated direction. Voltage and current remain unchanged. Modal Operation is unaffected by Soft Reset. However after a Soft Reset has completed, an Explicit Contract negotiation occurs, in order to re-establish PD Communication and to bring state operation for both Port Partners back to either the ***PE_SNK_Ready*** or ***PE_SRC_Ready*** states as appropriate (see ***Section 8.3.3.4 "SOP Soft Reset and Protocol Error State Diagrams"***).

A ***Soft_Reset*** Message ***May*** be sent by either the Source or Sink when there is a Message synchronization error. If the error is not corrected by the Soft Reset, ***Hard Reset*** Signaling ***Shall*** be issued (see ***Section 6.8.3 "Hard Reset"***).

A ***Soft_Reset*** Message ***Shall*** be targeted at a specific entity depending on the type of SOP* Packet used. ***Soft_Reset*** Messages sent using SOP Packets ***Shall*** Soft Reset the Port Partner only. ***Soft_Reset*** Messages sent using SOP'/SOP'' Packets ***Shall*** Soft Reset the corresponding Cable Plug only.

After a VCONN Swap the VCONN Source needs to reset the Cable Plug's Protocol Layer to ensure ***MessageID*** synchronization. If after a VCONN Swap the VCONN Source wants to communicate with a Cable Plug using SOP' Packets, it ***Shall*** issue a ***Soft_Reset*** Message using a SOP' Packet in order to reset the Cable Plug's Protocol Layer. If the VCONN Source wants to communicate with a Cable Plug using SOP'' Packets, it ***Shall*** issue a ***Soft_Reset*** Message using a SOP'' Packet in order to reset the Cable Plug's Protocol Layer.

6.3.14 Data_Reset Message

The ***Data_Reset*** Message ***May*** be sent by either the DFP or UFP and ***Shall*** reset the USB data connection and exit all Alternate Modes with its Port Partner while preserving the power on V_{BUS}. USB4® capable ports ***Shall*** support the ***Data_Reset*** Message and other ports ***May*** support the ***Data_Reset*** Message.

The ***Data_Reset*** Message ***Shall Not*** change the existing:

- Power Contract
- Data Roles (i.e., which port is the DFP or UFP)

The receiver of the ***Data_Reset*** Message ***Shall*** respond by sending an ***Accept*** Message and then follow the process outlined in the following steps. Neither the sender nor receiver ***Shall*** initiate a VCONN Swap until the Data Reset process is complete, and the ***Data_Reset_Complete*** Message has been sent. Following receipt of the ***Accept*** Message, or ***GoodCRC*** following the ***Accept***, depending which port sends the ***Data_Reset*** Message:

- The DFP ***Shall***:
  - Disconnect the Port's ***[USB 2.0]*** D+/D- signals.
  - If operating in ***[USB 3.2]*** remove the port's Rx Terminations (see ***[USB 3.2]***).
  - If operating in ***[USB4]*** drive the port's SBTX to a logic low (see ***[USB4]***).
- Both the DFP and UFP ***Shall*** exit all Alternate Modes if any.
- Reset the cable:
  - If the VCONN source port is also the UFP, then it ***Shall*** run the UFP VCONN Power Cycle process described in ***Section 7.1.15.1 "UFP Vconn Power Cycle"***.
  - If the VCONN source port is also the DFP, then it ***Shall*** run the DFP VCONN Power Cycle process described in ***Section 7.1.15.2 "DFP Vconn Power Cycle"***.
  - The DFP ***Shall*** exit the VCONN Power Cycle process as the VCONN Source and be sourcing VCONN.
- After ***tDataReset*** the DFP Shall:
  - Reconnect the ***[USB 2.0]*** D+/D- signals.
  - If the Port was operating in ***[USB 3.2]*** or ***[USB4]*** reapply the port's Rx Terminations (see ***[USB 3.2]***).
- The Data Reset process is complete; the DFP ***Shall*** send a ***Data_Reset_Complete*** Message and enter the USB4® Discovery and Entry Flow (See ***[USB Type-C 2.3]***).

If the initiator of the ***Data_Reset*** Message does not receive a ***Valid*** response within ***tSenderResponse*** it ***Shall*** enter the ***ErrorRecovery*** State.

6.3.15 Data_Reset_Complete Message

The ***Data_Reset_Complete*** Message ***Shall*** be sent by the DFP to the UFP to indicate the completion of the Data Reset process (see ***Section 6.3.14 "Data_Reset Message"***).

6.3.16 Not_Supported Message

The ***Not_Supported*** Message ***Shall*** be sent by a Port or Cable Plug in response to any Message it does not support. Returning a ***Not_Supported*** Message is assumed in this specification and has not been called out explicitly except in ***Section 6.13 "Message Applicability"*** which defines cases where the ***Not_Supported*** Message is returned.

6.3.17 Get_Source_Cap_Extended Message

The ***Get_Source_Cap_Extended*** (Get Source Capabilities Extended) Message is sent by a Port to request additional information about a Port's Source Capabilities. The Port ***Should*** respond by returning a ***Source_Capabilities_Extended*** Message (see ***Section 6.5.1" Source_Capabilities_Extended Message"***).

6.3.18 Get_Status Message

The ***Get_Status*** Message is sent by a Port using ***SOP*** to request the Port Partner's present status.

The Port Partner ***Shall*** respond by returning a ***Status*** Message (see ***Section 6.5.2 "Status Message"***). A Port that receives an ***Alert*** Message (see ***Section 6.4.6 "Alert Message"***) indicates that the Source or Sink's Status has changed and ***Should*** be re-read using a ***Get_Status*** Message.

The ***Get_Status*** Message ***May*** also be sent to an *Active Cable* to get its present status using ***SOP'***/***SOP"***.

The *Active Cable* ***Shall*** respond by returning a ***Status*** Message (see ***Section 6.5.2 "Status Message"***).

6.3.19 FR_Swap Message

The ***FR_Swap*** Message ***Shall*** be sent by the new Source within ***tFRSwapInit*** after it has detected a Fast Role Swap signal (see ***Section 5.8.6.3 "Fast Role Swap Detection"*** and ***Section 6.6.17.3 "tFRSwapInit"***). The Fast Role Swap AMS is necessary to apply R_p to the new Source and R_d to the new Sink and to re-synchronize the state machines. The ***tFRSwapInit*** time ***Shall*** be measured from the time the FRS signal has been sent for ***tFRSwapRx*** (max) until the last bit of the ***EOP*** of the ***FR_Swap*** Message has been transmitted by the Physical Layer.

The recipient of the ***FR_Swap*** Message ***Shall*** respond by sending an ***Accept*** Message.

After a successful Fast Role Swap the Port Partners ***Shall*** reset their respective Protocol Layers (equivalent to a Soft Reset): resetting their ***MessageIDCounter***, ***RetryCounter*** and Protocol Layer state machines before attempting to establish an Explicit Contract. At this point the Source ***Shall*** also reset its ***CapsCounter***.

This ensures that only the Cable Plug responds with a ***GoodCRC*** Message to the ***Discover Identity*** Command.

Prior to the Fast Role Swap AMS, the new Source ***Shall*** have R_d asserted on the CC wire and the new Sink ***Shall*** have R_p asserted on the CC wire. Note that this is an incorrect assignment of R_p/R_d (since R_p follows the Source and R_d follows the Sink as defined in ***[USB Type-C 2.3]***) that is corrected by the Fast Role Swap AMS.

During the Fast Role Swap AMS, the new Source ***Shall*** change its CC Wire resistor from R_d to R_p and the new Sink ***Shall*** change its CC Wire resistor from R_p to R_d. The DFP (Host), UFP (Device) roles and VCONN Source ***Shall*** remain unchanged during the Fast Role Swap process.

The initial Source ***Should*** avoid being the VCONN source (by using the VCONN Swap process) whenever not actively communicating with the cable, since it is difficult for the initial Source to maintain VCONN power during the Fast Role Swap process.

Note: A Fast Role Swap is a "best effort" solution to a situation where a PDUSB Device has lost its external power. This process can occur at any time, even during an AMS in which case error handling such as Hard Reset or ***[USB Type-C 2.3]*** Error Recovery will be triggered.

Note: during the Fast Role Swap process the initial Sink does not disconnect even though V_{BUS} drops below ***vSafe5V***.

For more information regarding the Fast Role Swap process, refer to:

- ***Section 7.1.13 "Fast Role Swap"***
- ***Section 7.2.10 "Fast Role Swap"***
- ***Section 8.3.3.20.5 "Policy Engine in Source to Sink Fast Role Swap State Diagram"***
- ***Section 8.3.3.20.6 "Policy Engine in Sink to Source Fast Role Swap State Diagram"***
- ***Section 9.1.2 "Mapping to USB Device States"*** for V_{BUS} mapping to USB states.

6.3.20 Get_PPS_Status

The ***Get_PPS_Status*** Message is sent by the Sink to request additional information about a Source's status. The Port ***Shall*** respond by returning a ***PPS_Status*** Message (see ***Section 6.5.10 "PPS_Status Message"***).

6.3.21 Get_Country_Codes

The ***Get_Country_Codes*** Message is sent by a Port to request the alpha-2 country codes its Port Partner supports as defined in ***[ISO 3166]***. The Port Partner ***Shall*** respond by returning a ***Country_Codes*** Message (see ***Section 6.5.11 "Country_Codes Message"***).

6.3.22 Get_Sink_Cap_Extended Message

The ***Get_Sink_Cap_Extended*** (Get Sink Capabilities Extended) Message is sent by a Port to request additional information about a Port's Sink Capabilities. The Port ***Shall*** respond by returning a ***Sink_Capabilities_Extended*** Message (see ***Section 6.5.13 "Sink_Capabilities_Extended Message"***).

6.3.23 Get_Source_Info Message

The ***Get_Source_Info*** Message is sent by a Port to request the type, maximum capabilities and present capabilities of the port when it is operating as a Source. The port ***Shall*** respond by returning the ***Source_Info*** Message (See ***Section 6.4.11 "Source_Info Message"***).

6.3.24 Get_Revision Message

The ***Get_Revision*** Message is sent by a Port using ***SOP*** to request the Revision and Version of the Power Delivery Specification its Port Partner supports.

The Port Partner ***Shall*** respond by returning a ***Revision*** Message (See ***Section 6.4.12 "Revision Message"***).

The ***Get_Revision*** Message ***May*** also be sent to a Cable Plug to request the Revision and Version of the Power Delivery Specification it supports using ***SOP'/SOP''***.

The *Active Cable* ***Shall*** respond by returning a ***Revision*** Message (see ***Section 6.4.12 "Revision Message"***).

6.4 Data Message

A Data Message ***Shall*** consist of a Message Header and be followed by one or more Data Objects. Data Messages are easily identifiable because the ***Number of Data Objects*** field in the Message Header is a non-zero value.

There are many types of Data Objects used to compose Data Messages. Some examples are:

- Power Data Object (PDO) used to expose a Source Port's power capabilities or a Sink's power requirements.
- Request Data Object (RDO) used by a Sink Port to negotiate a Contract.
- Vendor Defined Data Object (VDO) used to convey vendor specific information.
- BIST Data Object (BDO) used for PHY Layer compliance testing.
- Battery Status Data Object (BSDO) used to convey Battery status information.
- Alert Data Object (ADO) used to indicate events occurring on the Source or Sink.

The type of Data Object being used in a Data Message is defined by the Message Header's ***Message Type*** field and is summarized in ***Table 6.6 "Data Message Types"***. The Sent by column indicates entities which ***May*** send the given Message (Source, Sink or Cable Plug); entities not listed ***Shall Not*** issue the corresponding Message. The "***Valid*** Start of Packet" column indicates the Messages which ***Shall*** only be issued in SOP Packets and the Messages which ***May*** be issued in SOP* Packets.

Table 6.6 "Data Message Types"

Bits 4...0	Type	Sent by	Description	Valid Start of Packet
0_0000	***Reserved***	***N/A***	All values not explicitly defined are ***Reserved*** and ***Shall Not*** be used.	***N/A***
0_0001	***Source_Capabilities***	Source or Dual-Role Power	See ***Section 6.4.1.2***	SOP only
0_0010	***Request***	Sink only	See ***Section 6.4.2***	SOP only
0_0011	***BIST***	Tester, Source or Sink	See ***Section 6.4.3***	SOP*
0_0100	***Sink_Capabilities***	Sink or Dual-Role Power	See ***Section 6.4.1.2.5.3***	SOP only
0_0101	***Battery_Status***	Source or Sink	See ***Section 6.4.5***	SOP only
0_0110	***Alert***	Source or Sink	See ***Section 6.4.6***	SOP only
0_0111	***Get_Country_Info***	Source or Sink	See ***Section 6.4.7***	SOP only
0_1000	***Enter_USB***	DFP	See ***Section 6.4.8***	SOP*
0_1001	***EPR_Request***	Sink	See ***Section 6.4.9***	SOP only
0_1010	***EPR_Mode***	Source or Sink	See ***Section 6.4.10***	SOP only
0_1011	***Source_Info***	Source	See ***Section 6.4.11***	SOP only
0_1100	***Revision***	Source, Sink or Cable Plug	See ***Section 6.4.12***	SOP*
0_1101... 0_1110	***Reserved***	***N/A***	All values not explicitly defined are ***Reserved*** and ***Shall Not*** be used.	***N/A***
0 1111	***Vendor_Defined***	Source, Sink or Cable Plug	See ***Section 6.4.4***	SOP*
1_0000... 1_1111	***Reserved***	***N/A***	All values not explicitly defined are ***Reserved*** and ***Shall Not*** be used.	***N/A***

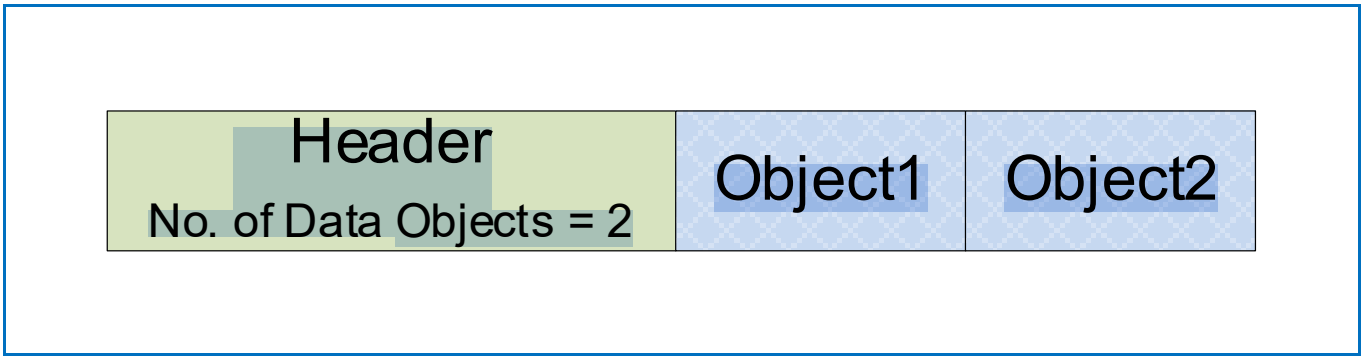
6.4.1 Capabilities Message

A Capabilities Message (***Source_Capabilities*** Message or ***Sink_Capabilities*** Message) ***Shall*** have at least one Power Data Object for ***vSafe5V***. The Capabilities Message ***Shall*** also contain the sending Port's information followed by up to 6 additional Power Data Objects. Power Data Objects in a Capabilities Message ***Shall*** be sent in the following order:

1) The ***vSafe5V*** Fixed Supply Object ***Shall*** always be the first object.
2) The remaining Fixed Supply Objects, if present, ***Shall*** be sent in Voltage order; lowest to highest.
3) The Battery Supply Objects if present ***Shall*** be sent in Minimum Voltage order; lowest to highest.
4) The Variable Supply (non-Battery) Objects, if present, ***Shall*** be sent in Minimum Voltage order; lowest to highest.
5) The SPR Adjustable Voltage Supply Object, if present, ***Shall*** be sent.
6) The Programmable Power Supply Objects, if present, ***Shall*** be sent in Maximum Voltage order, lowest to highest.

Note: The EPR Capabilities message construction is defined in ***Section 6.5.15.1 "EPR Capabilities Message Construction"***.

Figure 6-12 "Example Capabilities Message with 2 Power Data Objects"



In ***Figure 6-12 "Example Capabilities Message with 2 Power Data Objects"***, the ***Number of Data Objects*** field is 2: ***vSafe5V*** plus one other Voltage.

Power Data Objects (PDO) and Augmented Power Data Objects (APDO) are identified by the Message Header's Type field. They are used to form ***Source_Capabilities*** Messages and ***Sink_Capabilities*** Messages.

There are three types of Power Data Objects. They contain additional information beyond that encoded in the Message Header to identify each of the three types of Power Data Objects:

- Fixed Supply is used to expose well-regulated fixed Voltage power supplies.
- Variable power supply is used to expose very poorly regulated power supplies.
- Battery is used to expose batteries that can be directly connected to V_{BUS}.

There are three types of Augmented Power Data Objects:

- SPR Programmable Power Supply is used to expose a power supply whose output Voltage can be programmatically adjusted over the Advertised Voltage range and limited by the Source to a programmable current limit.

- SPR and EPR Adjustable Voltage Supply are used to expose a power supply whose output Voltage can be adjusted over the Advertised Voltage range but otherwise is equivalent to a fixed Voltage power supply (AVS does not support a programmable current limit).

Power Data Objects are also used to expose additional capabilities that ***May*** be utilized, such as in the case of a Power Role Swap.

A list of one or more Power Data Objects ***Shall*** be sent by the Source to convey its capabilities. The Sink ***May*** then request one of these capabilities by returning a Request Data Object that contains an index to a Power Data Object, to negotiate a mutually agreeable Contract.

Where Maximum and Minimum Voltage and Current values are given in PDOs these ***Shall*** be taken to be absolute values.

The Source and Sink ***Shall Not*** negotiate a power level that would allow the current to exceed the maximum current supported by their receptacles or the Attached plug (see ***[USB Type-C 2.3]***). The Source ***Shall*** limit its offered capabilities to the maximum current supported by its receptacle and Attached plug. A Sink ***Shall*** only make a request from any of the capabilities offered by the Source. For further details see ***Section 4.4 "Cable Type Detection"***.

Sources expose their power capabilities by sending a ***Source_Capabilities*** Message. Sinks expose their power requirements by sending a ***Sink_Capabilities*** Message. Both are composed of several 32-bit Power Data Objects (see ***Table 6.7 "Power Data Object"***).

Table 6.7 "Power Data Object"

Bit(s)	Description	
	Value	Parameter
B31…30	00b	Fixed supply (Vmin = Vmax)
	01b	Battery
	10b	Variable Supply (non-Battery)
	11b	Augmented Power Data Object (APDO)
B29…0	Specific Power Capabilities are described by the PDOs in the following sections.	

The Augmented Power Data Object (APDO) is defined to allow support for more than the four PDO types by extending the Power Data Object field from 2 to 4 bits when the B31…B30 are 11b. The generic APDO structure is shown in ***Table 6.8 "Augmented Power Data Object"***.

Table 6.8 "Augmented Power Data Object"

Bit(s)	Description	
	Value	Parameter
B31…30	11b	Augmented Power Data Object (APDO)
B29…28	00b	SPR Programmable Power Supply
	01b	EPR Adjustable Voltage Supply
	10b	SPR Adjustable Voltage Supply
	11b	***Reserved***
B27…0	Specific Power Capabilities are described by the APDOs in the following sections.	

6.4.1.1 Use of the Capabilities Message

6.4.1.1.1 Use by Sources

Sources send a ***Source_Capabilities*** Message (see ***Section 6.4.1.2 "Source_Capabilities Message"***) either as part of advertising Port capabilities, or in response to a ***Get_Source_Cap*** Message. See ***Section 6.5.15.2 "EPR_Source_Capabilities Message"*** for information about EPR Source capabilities messages.

Following a Hard Reset, a power-on event or plug insertion event, a Source Port ***Shall*** send a ***Source_Capabilities*** Message after every ***SourceCapabilityTimer*** timeout as an Advertisement that ***Shall*** be interpreted by the Sink Port on Attachment. The Source ***Shall*** continue sending a minimum of ***nCapsCount Source_Capabilities*** Messages until a ***GoodCRC*** Message is received.

Additionally, a ***Source_Capabilities*** Message ***Shall*** only be sent by a Port in the following cases:

- By the Source Port from the ***PE_SRC_Ready*** state upon a change in its ability to supply power to this Port.
- By a Source Port or Dual-Role Power Port in response to a ***Get_Source_Cap*** Message.
- ***Optionally*** by a Source Port from the ***PE_SRC_Ready*** state when available power in a multi-port system change, even if the source capabilities for this Port have not changed.

6.4.1.1.2 Use by Sinks

Sinks send a ***Sink_Capabilities*** Message (see ***Section 6.4.1.3 "Sink Capabilities Message"***) in response to a ***Get_Sink_Cap*** Message. See ***Section 6.5.15.3 "EPR_Sink_Capabilities Message"*** for more information about EPR Sink capabilities messages.

A USB Power Delivery capable Sink, upon detecting ***vSafe5V*** on V_{BUS} and after a ***SinkWaitCapTimer*** timeout without seeing a ***Source_Capabilities*** Message, ***Shall*** send a Hard Reset. If the Attached Source is USB Power Delivery capable, it responds by sending ***Source_Capabilities*** Messages thus allowing power negotiations to begin.

6.4.1.1.3 Use by Dual-Role Power devices

Dual-Role Power devices send a ***Source_Capabilities*** Message (see ***Section 6.4.1.2 "Source_Capabilities Message"***) as part of advertising Port capabilities when operating in Source role. Dual-Role Power devices send a ***Source_Capabilities*** Message (see ***Section 6.4.1.2 "Source_Capabilities Message"***) in response to a ***Get_Source_Cap*** Message regardless of their present operating role. Similarly Dual-Role Power devices send a ***Sink_Capabilities*** Message (see ***Section 6.4.1.3 "Sink Capabilities Message"***) in response to a ***Get_Sink_Cap*** Message regardless of their present operating role.

6.4.1.2 Source_Capabilities Message

A Source Port ***Shall*** report its capabilities in a series of 32-bit Power Data Objects (see ***Table 6.7 "Power Data Object"***) as part of a ***Source_Capabilities*** Message (see ***Figure 6-12 "Example Capabilities Message with 2 Power Data Objects"***). Power Data Objects are used to convey a Source Port's capabilities to provide power including Dual-Role Power ports presently operating as a Sink.

Each Power Data Object ***Shall*** describe a specific Source capability such as a Battery (e.g., 2.8-4.1V) or a fixed power supply (e.g., 15V) at a maximum allowable current. The ***Number of Data Objects*** field in the Message Header ***Shall*** define the number of Power Data Objects that follow the Message Header in a Data Message. All Sources ***Shall*** minimally offer one Power Data Object that reports ***vSafe5V***. A Source ***Shall Not*** offer multiple Power Data Objects of the same type (fixed, variable, Battery) and the same Voltage but ***Shall*** instead offer one Power Data Object with the highest available current for that Source capability and Voltage.

Sinks with Accessory Support do not source V_{BUS} (see ***[USB Type-C 2.3]***). Sinks with Accessory Support are still considered Sources when sourcing V_{CONN} to an Accessory even though V_{BUS} is not applied; in this case they ***Shall*** Advertise ***vSafe5V*** with the Maximum Current set to 0mA in the first Power Data Object. The main purpose of this is to enable the Sink with Accessory Support to get into the ***PE_SRC_Ready*** State to enter an Alternate Mode.

A Sink in SPR Mode ***Shall*** evaluate every ***Source_Capabilities*** Message it receives and ***Shall*** respond with a ***Request*** Message. If its power consumption exceeds the Source's capabilities it ***Shall*** re-negotiate so as not to exceed the Source's most recently Advertised capabilities.

A Sink, in SPR Mode, in an Explicit Contract with a PPS APDO, ***Shall*** periodically re-request the PPS APDO at least every ***tPPSRequest*** until either:

- The Sink requests something other than PPS APDO.
- There is a Power Role Swap.
- There is a Hard Reset.

A Sink in EPR Mode that receives a ***Source_Capabilities*** Message in response to a ***Get_Source_Cap*** Message ***Shall Not*** respond with a ***Request*** Message. If a Sink in EPR Mode receives a ***Source_Capabilities*** Message, not in response to a ***Get_Source_Cap*** Message, the Sink ***Shall*** initiate a Hard Reset.

A Source that has accepted a ***Request*** Message with a Programmable RDO ***Shall*** issue ***Hard Reset*** Signaling if it has not received a ***Request*** Message with a Programmable RDO within ***tPPSTimeout***. The Source ***Shall*** discontinue this behavior after:

- Receiving a ***Request*** Message with a Fixed, Variable or Battery RDO.
- There is a Power Role Swap.
- There is a Hard Reset.

6.4.1.2.1 Management of the Power Reserve

A Power Reserve ***May*** be allocated to a Sink when it makes a request from Source Capabilities which includes a Maximum Operating Current/Power. The size of the Power Reserve for a particular Sink is calculated as the difference between its Maximum Operating Current/Power field and its Operating Current/Power field. For a Hub with multiple ports this same Power Reserve ***May*** be shared between several Sinks. The Power Reserve ***May*** also be temporarily used by a Sink which has indicated it can give back power by setting the GiveBack flag.

Where a Power Reserve has been allocated to a Sink the Source ***Shall*** indicate the Power Reserve as part of every ***Source_Capabilities*** Message it sends. When the same Power Reserve is shared between several Sinks the Source ***Shall*** indicate the Power Reserve as part of every ***Source_Capabilities*** Message it sends to every Sink. Every time a

Source sends capabilities including the Power Reserve capability and then accepts a request from a Sink including the Power Reserve indicated by its Maximum Operating Current/Power it is confirming that the Power Reserve is part of the Explicit Contract with the Sink.

When the Reserve is being temporarily used by a giveback capable Sink the Source ***Shall*** indicate the Power Reserve as available in every ***Source_Capabilities*** Message it sends. However, in this situation, when the Power Reserve is requested by a Sink, the Source ***Shall*** return a ***Wait*** Message while it retrieves this power using a ***GotoMin*** Message. Once the additional power has been retrieved the Source ***Shall*** send a new ***Source_Capabilities*** Message in order to trigger a new request from the Sink requesting the Power Reserve.

The Power Reserve ***May*** be de-allocated by the Source at any time, but the de-allocation ***Shall*** be indicated to the Sink or Sinks using the Power Reserve by sending a new ***Source_Capabilities*** Message.

6.4.1.2.2 Fixed Supply Power Data Object

Table 6.9 "Fixed Supply PDO – Source" describes the Fixed Supply (00b) PDO. See ***Section 7.1.3 "Types of Sources"*** for the electrical requirements of the power supply.

Since all USB Providers support ***vSafe5V***, the required ***vSafe5V*** Fixed Supply Power Data Object is also used to convey additional information that is returned in bits 29…23. All other Fixed Supply Power Data Objects ***Shall*** set bits 29…23 to zero.

For a Source offering no capabilities, the Voltage (B19…10) ***Shall*** be set to 5V and the Maximum Current ***Shall*** be set to 0mA. This is used in cases such as a Dual-Role Power device which offers no capabilities in its default role or when external power is required to offer power.

When a Source wants a Sink, consuming power from V_{BUS}, to go to its lowest power state, the Voltage (B19…10) ***Shall*** be set to 5V and the Maximum Current ***Shall*** be set to 0mA. This is used in cases where the Source wants the Sink to draw ***pSnkSusp***.

Table 6.9 "Fixed Supply PDO – Source"

Bit(s)	Description
B31…30	Fixed supply
B29	Dual-Role Power
B28	USB Suspend Supported
B27	Unconstrained Power
B26	USB Communications Capable
B25	Dual-Role Data
B24	Unchunked Extended Messages Supported
B23	EPR Mode Capable
B22	***Reserved*** – ***Shall*** be set to zero.
B21…20	Peak Current
B19…10	Voltage in 50mV units
B9…0	Maximum Current in 10mA units

6.4.1.2.2.1 Dual-Role Power

The Dual-Role Power bit ***Shall*** be set when the Port is Dual-Role Power capable i.e., supports the ***PR_Swap*** Message.

This is a static capability which ***Shall*** remain fixed for a given device regardless of the device's present power role. If the Dual-Role Power bit is set to one in the ***Source_Capabilities*** Message the Dual-Role Power bit in the ***Sink_Capabilities*** Message ***Shall*** also be set to one. If the Dual-Role Power bit is set to zero in the ***Source_Capabilities*** Message the Dual-Role Power bit in the ***Sink_Capabilities*** Message ***Shall*** also be set to zero.

6.4.1.2.2.2 USB Suspend Supported

Prior to a Contract or when the USB Communications Capable bit is set to zero, this flag is undefined and Sinks ***Shall*** follow the rules for suspend as defined in ***[USB 2.0]***, ***[USB 3.2]***, ***[USB4]***, ***[USB Type-C 2.3]*** or ***[USBBC 1.2]***. After a Contract has been negotiated:

- If the USB Suspend Supported flag is set, then the Sink ***Shall*** follow the ***[USB 2.0]***, ***[USB 3.2]*** or ***[USB4]*** rules for suspend and resume. A PDUSB Peripheral ***May*** draw up to ***pSnkSusp*** during suspend; a PDUSB Hub ***May*** draw up to ***pHubSusp*** during suspend (see ***Section 7.2.3 "Sink Standby"***).
- If the USB Suspend Supported flag is cleared, then the Sink ***Shall Not*** apply the ***[USB 2.0]***, ***[USB 3.2]*** or ***[USB4]*** rules for suspend and ***May*** continue to draw the negotiated power. Note that when USB is suspended, the USB device state is also suspended.

Sinks ***May*** indicate to the Source that they would prefer to have the USB Suspend Supported flag cleared by setting the No USB Suspend flag in a ***Request*** Message (see ***Section 6.4.2.5 "No USB Suspend"***).

6.4.1.2.2.3 Unconstrained Power

The Unconstrained Power bit ***Shall*** be set when an external source of power is available that is sufficient to adequately power the system while charging external devices, or when the device's primary function is to charge external devices.

To set the Unconstrained Power bit because of an external source, the external source of power ***Should*** be either:

- An AC supply, e.g., a wall wart, directly connected to the Sink.
- Or, in the case of a PDUSB Hub:
  - A PD Source with its Unconstrained Power bit set.
  - Multiple PD Sources all with their Unconstrained Power bits set.

6.4.1.2.2.4 USB Communications Capable

The USB Communications Capable bit ***Shall*** only be set for Sources capable of communication over the USB data lines (e.g., D+/- or SS Tx/Rx).

6.4.1.2.2.5 Dual-Role Data

The Dual-Role Data bit ***Shall*** be set when the Port is Dual-Role data capable i.e., it supports the ***DR_Swap*** Message. This is a static capability which ***Shall*** remain fixed for a given device regardless of the device's present power role or data role. If the Dual-Role Data bit is set to one in the ***Source_Capabilities*** Message the Dual-Role Data bit in the ***Sink_Capabilities*** Message ***Shall*** also be set to one. If the Dual-Role Data bit is set to zero in the ***Source_Capabilities*** Message the Dual-Role Data bit in the ***Sink_Capabilities*** Message ***Shall*** also be set to zero.

6.4.1.2.2.6 Unchunked Extended Messages Supported

The Unchunked Extended Messages Supported bit ***Shall*** be set when the Port can send and receive Extended Messages with ***Data Size*** > ***MaxExtendedMsgLegacyLen*** bytes in a single, Unchunked Message.

6.4.1.2.2.7 EPR Mode Capable

The EPR Mode Capable bit is a static bit that ***Shall*** be set if the Source is designed to supply more than 100W and operate in EPR Mode.

When this bit is set, an EPR Source:

- Operating in SPR Mode Shall only send an *EPR_Source_Capabilities* Message in response to an *EPR_Get_Source_Cap* Message
- ***May*** only enter EPR Mode when the Cable and the Sink also report that they are EPR capable.

6.4.1.2.2.8 Peak Current

The USB Power Delivery Fixed Supply is only required to deliver the amount of current requested in the Operating Current (IOC) field of an RDO. In some usages however, for example computer systems, where there are short bursts of activity, it might be desirable to overload the power source for short periods.

For example, when a computer system tries to maintain average power consumption, the higher the peak current, the longer the low current (see *Section 7.2.8 "Sink Peak Current Operation"*) period needed to maintain such average power. The Peak Current field allows a power source to Advertise this additional capability. This capability is intended for direct Port to Port connections only and ***Shall Not*** be offered to downstream Sinks via a Hub.

Every Fixed Supply PDO ***Shall*** contain a Peak Current field. Supplies that want to offer a set of overload capabilities ***Shall*** Advertise this through the Peak Current field in the corresponding Fixed Supply PDO (see *Table 6.10 "Fixed Power Source Peak Current Capability"*). Supplies that do not support an overload capability ***Shall*** set these bits to 00b in the corresponding Fixed Supply PDO. Supplies that support an extended overload capability specified in the PeakCurrent1…3 fields of the *Source_Capabilities_Extended* Message (see *Section 6.5.1 "Source_Capabilities_Extended Message"*) ***Shall*** also set these bits to 00b. Sinks wishing to utilize these extended capabilities ***Shall*** first send the *Get_Source_Cap_Extended* Message to determine what capabilities, if any are supported by the Source.

Table 6.10 "Fixed Power Source Peak Current Capability"

Bits 21…20	Description
00	Peak current equals IOC (default) or look at extended Source capabilities (send *Get_Source_Cap_Extended* Message)
01	Overload Capabilities: 1. Peak current equals 150% IOC for 1ms @ 5% duty cycle (low current equals 97% IOC for 19ms) 2. Peak current equals 125% IOC for 2ms @ 10% duty cycle (low current equals 97% IOC for 18ms) 3. Peak current equals 110% IOC for 10ms @ 50% duty cycle (low current equals 90% IOC for 10ms)
10	Overload Capabilities: 1. Peak current equals 200% IOC for 1ms @ 5% duty cycle (low current equals 95% IOC for 19ms) 2. Peak current equals 150% IOC for 2ms @ 10% duty cycle (low current equals 94% IOC for 18ms) 3. Peak current equals 125% IOC for 10ms @ 50% duty cycle (low current equals 75% IOC for 10ms)
11	Overload Capabilities: 1. Peak current equals 200% IOC for 1ms @ 5% duty cycle (low current equals 95% IOC for 19ms) 2. Peak current equals 175% IOC for 2ms @ 10% duty cycle (low current equals 92% IOC for 18ms) 3. Peak current equals 150% IOC for 10ms @ 50% duty cycle (low current equals 50% IOC for 10ms)

6.4.1.2.3 Variable Supply (non-Battery) Power Data Object

Table 6.11 "Variable Supply (non-Battery) PDO – Source" describes a Variable Supply (non-Battery) (10b) PDO for a Source. See *Section 7.1.3 "Types of Sources"* for the electrical requirements of the power supply.

The Voltage fields ***Shall*** define the range that output Voltage ***Shall*** fall within. This does not indicate the Voltage that will be supplied, except it ***Shall*** fall within that range. The absolute Voltage, including any Voltage variation, ***Shall Not*** fall below the Minimum Voltage and ***Shall Not*** exceed the Maximum Voltage. The Minimum Voltage ***Shall Not*** be less than 80% of the Maximum Voltage.

Table 6.11 "Variable Supply (non-Battery) PDO – Source"

Bit(s)	Description
B31…30	Variable Supply (non-Battery)
B29…20	Maximum Voltage in 50mV units
B19…10	Minimum Voltage in 50mV units
B9…0	Maximum Current in 10mA units

6.4.1.2.4 Battery Supply Power Data Object

Table 6.12 "Battery Supply PDO – Source" describes a Battery (01b) PDO for a Source. See *Section 7.1.3 "Types of Sources"* for the electrical requirements of the power supply.

The Voltage fields ***Shall*** represent the Battery's Voltage range. The Battery ***Shall*** be capable of supplying the Power value over the entire Voltage range. The absolute Voltage, including any Voltage variation, ***Shall Not*** fall below the Minimum Voltage and ***Shall Not*** exceed the Maximum Voltage.

Note: the Battery PDO uses power instead of current.

The Sink ***May*** monitor the Battery Voltage.

Table 6.12 "Battery Supply PDO – Source"

Bit(s)	Description
B31…30	Battery
B29…20	Maximum Voltage in 50mV units
B19…10	Minimum Voltage in 50mV units
B9…0	Maximum Allowable Power in 250mW units

6.4.1.2.5 Augmented Power Data Object (APDO)

The Voltage fields define the output Voltage range over which the power supply ***Shall*** be adjustable in 20mV steps in SPR PPS Mode and 100mV steps in both SPR AVS Mode and EPR AVS Mode. The Maximum Current field contains the current the Programmable Power Supply ***Shall*** be capable of delivering over the Advertised Voltage range. See *Section 7.1.3 "Types of Sources"* for the electrical requirements of the power supply.

6.4.1.2.5.1 SPR Programmable Power Supply APDO

Table 6.13 "SPR Programmable Power Supply APDO – Source" below describes the SPR Programmable Power Supply (1100b) APDO for a Source operating in SPR Mode and supplying 5V up to 21V.

Table 6.13 "SPR Programmable Power Supply APDO – Source"

Bit(s)	Description
B31…30	11b – Augmented Power Data Object (APDO)
B29…28	00b – SPR Programmable Power Supply
B27	PPS Power Limited
B26…25	***Reserved*** – ***Shall*** be set to zero
B24…17	Maximum Voltage in 100mV increments
B16	***Reserved*** – ***Shall*** be set to zero
B15…8	Minimum Voltage in 100mV increments
B7	***Reserved*** – ***Shall*** be set to zero
B6…0	Maximum Current in 50mA increments

The PPS APDO is used primarily for Sink Directed Charge of a Battery in the Sink. When applying a current to the Battery greater than the cable supports, a high efficiency fixed scaler ***May*** be used in the Sink to reduce the cable current.

6.4.1.2.5.1.1 PPS Power Limited

When the PPS Power Limited bit is set, the SPR PPS Source ***Shall*** operate in the same way as if the PPS Power Limited bit is clear (see *Section 7.1.4.2 "SPR Programmable Power Supply (PPS)"*) with the below exception:

- ***May*** supply power that exceeds the Source's rated PDP within the ***Optional*** operating area in *Figure 7-9 "SPR PPS Constant Power"*.

The SPR PPS Source ***Shall Not*** reject an RDO with an Output Current that is less than or equal to the Maximum Current in the APDO even if the requested Output Current is greater than the Source's PDP/requested Output Voltage.

When the PPS Power Limited bit is cleared, the SPR PPS Source ***Shall*** deliver the Maximum Current up to the Maximum Voltage as Advertised in its APDO.

6.4.1.2.5.2 EPR Adjustable Voltage Supply APDO

Table 6.14 "EPR Adjustable Voltage Supply APDO – Source " below describes the EPR Adjustable Voltage Supply (1101b) APDO for a Source operating in EPR Mode and supplying 15V up to 48V.

Table 6.14 "EPR Adjustable Voltage Supply APDO – Source "

Bit(s)	Description
B31…30	11b – Augmented Power Data Object (APDO)
B29…28	01b – EPR Adjustable Voltage Supply
B27…26	Peak Current (see *Table 6.15 "EPR AVS Power Source Peak Current Capability"*)
B25…17	Maximum Voltage in 100mV increments
B16	***Reserved*** – ***Shall*** be set to zero
B15…8	Minimum Voltage in 100mV increments
B7…0	PDP in 1W increments

6.4.1.2.5.2.1 PDP

The PDP field ***Shall*** contain the AVS Port's PDP.

See *Section 10.2.3.3 "Optional Normative Extended Power Range (EPR)"* and *Figure 10-6 "Valid EPR AVS Operating Region"* for more information regarding how PDP in the AVS APDO relates to maximum available current.

6.4.1.2.5.2.2 Peak Current

The USB Power Delivery EPR Adjustable Voltage Supply is only required to deliver the amount of current requested in the Operating Current (IOC) field of an AVS RDO. In some usages however, for example computer systems, where there are short bursts of activity, it might be desirable to overload the power source for short periods.

For example, when a computer system tries to maintain average power consumption, the higher the peak current, the longer the low current period needed to maintain such average power (see *Section 7.2.8 "Sink Peak Current Operation"*). The Peak Current field allows a power source to Advertise this additional capability. This capability is intended for direct Port to Port connections only and ***Shall Not*** be offered to downstream Sinks via a Hub.

Every EPR Adjustable voltage Supply PDO ***Shall*** contain a Peak Current field. Supplies that want to offer a set of overload capabilities ***Shall*** Advertise this through the Peak Current field in the corresponding EPR AVS PDO (see *Table 6.15 "EPR AVS Power Source Peak Current Capability"*. Supplies that do not support an overload capability ***Shall*** set these bits to 00b in the corresponding EPR AVS PDO. Supplies that support an extended overload capability specified in the PeakCurrent1…3 fields of the *Source_Capabilities_Extended* Message (see *Section 6.5.1 "Source_Capabilities_Extended Message"*) ***Shall*** set these bits to 00b. Sinks wishing to utilize these extended capabilities ***Shall*** first send a *Get_Source_Cap_Extended* Message to determine what capabilities, if any are supported by the Source.

Table 6.15 "EPR AVS Power Source Peak Current Capability"

Bits 21…20	Description
00	Peak current equals IOC (default) or look at extended Source capabilities (send *Get_Source_Cap_Extended* Message)
01	Overload Capabilities: 1. Peak current equals 150% IOC for 1ms @ 5% duty cycle (low current equals 97% IOC for 19ms) 2. Peak current equals 125% IOC for 2ms @ 10% duty cycle (low current equals 97% IOC for 18ms) 3. Peak current equals 110% IOC for 10ms @ 50% duty cycle (low current equals 90% IOC for 10ms)
10	Overload Capabilities: 1. Peak current equals 200% IOC for 1ms @ 5% duty cycle (low current equals 95% IOC for 19ms) 2. Peak current equals 150% IOC for 2ms @ 10% duty cycle (low current equals 94% IOC for 18ms) 3. Peak current equals 125% IOC for 10ms @ 50% duty cycle (low current equals 75% IOC for 10ms)
11	Overload Capabilities: 1. Peak current equals 200% IOC for 1ms @ 5% duty cycle (low current equals 95% IOC for 19ms) 2. Peak current equals 175% IOC for 2ms @ 10% duty cycle (low current equals 92% IOC for 18ms) 3. Peak current equals 150% IOC for 10ms @ 50% duty cycle (low current equals 50% IOC for 10ms)

6.4.1.2.5.3 SPR Adjustable Voltage Supply APDO

Table 6.16 "SPR Adjustable Voltage Supply APDO – Source" below describes the SPR Adjustable Voltage Supply (1110b) APDO for a Source operating in SPR Mode and supplying 9V up to 20V.

Table 6.16 "SPR Adjustable Voltage Supply APDO – Source"

Bits 21…20	Description
B31…30	11b – Augmented Power Data Object (APDO)
B29…28	10b – SPR Adjustable Voltage Supply
B27…26	Peak Current (see *Table 6.10 "Fixed Power Source Peak Current Capability"*)
B25…20	***Reserved*** **–** ***Shall*** be set to zero.
B19…10	For 9V – 15V range: Maximum Current in 10mA units equal to the Maximum Current field of the 15V Fixed Source PDO
B9…0	For 15V – 20V range: Maximum Current in 10mA units equal to the Maximum Current field of the 20V Fixed Source PDO, set to 0 if the Maximum voltage in the SPR AVS range is 15V.

6.4.1.2.5.3.1 Peak Current

Peak Current for SPR AVS APDO follows the same definition for Fixed Supply PDOs (see *Section 6.4.1.2.2.8 "Peak Current"* and *Table 6.10 "Fixed Power Source Peak Current Capability"*).

6.4.1.3 Sink Capabilities Message

A Sink Port ***Shall*** report power levels it is able to operate at in a series of 32-bit Power Data Objects (see *Table 6.7 "Power Data Object"*). These are returned as part of a *Sink_Capabilities* Message in response to a *Get_Sink_Cap* Message (see *Figure 6-12 "Example Capabilities Message with 2 Power Data Objects"*). This is similar to that used for Source Port capabilities with equivalent Power Data Objects for Fixed, Variable and Battery Supplies as defined in this section. Power Data Objects are used to convey the Sink Port's operational power requirements including Dual-Role Power Ports presently operating as a Source.

Each Power Data Object ***Shall*** describe a specific Sink operational power level, such as a Battery (e.g., 2.8-4.1V) or a fixed power supply (e.g., 15V). The *Number of Data Objects* field in the Message Header ***Shall*** define the number of Power Data Objects that follow the Message Header in a Data Message.

All Sinks ***Shall*** minimally offer one Power Data Object with a power level at which the Sink can operate. A Sink ***Shall Not*** offer multiple Power Data Objects of the same type (fixed, variable, Battery) and the same Voltage but ***Shall*** instead offer one Power Data Object with the highest available current for that Sink capability and Voltage.

All Sinks ***Shall*** include one Power Data Object that reports *vSafe5V* even if they require additional power to operate fully. In the case where additional power is required for full operation the Higher Capability bit ***Shall*** be set.

6.4.1.3.1 Sink Fixed Supply Power Data Object

Table 6.17 "Fixed Supply PDO – Sink" describes the Sink Fixed Supply (00b) PDO. See *Section 7.1.3 "Types of Sources"* for the electrical requirements of the power supply. The Sink ***Shall*** set Voltage to its required Voltage and Operational Current to its required operating current. Required operating current is defined as the amount of current a given device needs to be functional. This value could be the maximum current the Sink will ever require or could be sufficient to operate the Sink in one of its modes of operation.

Since all USB Consumers support *vSafe5V*, the required *vSafe5V* Fixed Supply Power Data Object is also used to convey additional information that is returned in bits 29 through 20. All other Fixed Supply Power Data Objects ***Shall*** set bits 29…20 to zero.

For a Sink requiring no power from the Source, the Voltage (B19…10) ***Shall*** be set to 5V and the Operational Current ***Shall*** be set to 0mA.

Table 6.17 "Fixed Supply PDO – Sink"

Bit(s)	Description
B31…30	Fixed supply
B29	Dual-Role Power
B28	Higher Capability
B27	Unconstrained Power
B26	USB Communications Capable
B25	Dual-Role Data
B24…23	Fast Role Swap required USB Type-C® Current (see also *[USB Type-C 2.3]*): Value / Description 00b – Fast Swap not supported (default) 01b – Default USB Power 10b – 1.5A @ 5V 11b – 3.0A @ 5V
B22…20	***Reserved*** – ***Shall*** be set to zero.
B19…10	Voltage in 50mV units
B9…0	Operational Current in 10mA units

6.4.1.3.1.1 Dual-Role Power

The Dual-Role Power bit ***Shall*** be set when the Port is Dual-Role Power capable i.e., supports the ***PR_Swap*** Message. This is a static capability which ***Shall*** remain fixed for a given device regardless of the device's present power role. If the Dual-Role Power bit is set to one in the ***Source_Capabilities*** Message the Dual-Role Power bit in the ***Sink_Capabilities*** Message ***Shall*** also be set to one. If the Dual-Role Power bit is set to zero in the ***Sink_Capabilities*** Message the Dual-Role Power bit in the ***Sink_Capabilities*** Message ***Shall*** also be set to zero.

6.4.1.3.1.2 Higher Capability

In the case that the Sink needs more than ***vSafe5V*** (e.g., 15V) to provide full functionality, then the Higher Capability bit ***Shall*** be set.

6.4.1.3.1.3 Unconstrained Power

The Unconstrained Power bit ***Shall*** be set when an external source of power is available that is sufficient to adequately power the system while charging external devices, or when the device's primary function is to charge external devices.

To set the Unconstrained Power bit because of an external source, the external source of power ***Should*** be either:

- An AC supply, e.g., a wall wart, directly connected to the Sink.
- Or, in the case of a PDUSB Hub:
  - A PD Source with its Unconstrained Power bit set.
  - Multiple PD Sources all with their Unconstrained Power bits set.

6.4.1.3.1.4 USB Communications Capable

The USB Communications Capable bit ***Shall*** only be set for Sinks capable of communication over the USB data lines (e.g., D+/- or SS Tx/Rx).

6.4.1.3.1.5 Dual-Role Data

The Dual-Role Data bit ***Shall*** be set when the Port is Dual-Role data capable i.e., it supports the ***DR_Swap*** Message. This is a static capability which ***Shall*** remain fixed for a given device regardless of the device's present power role or data role. If the Dual-Role Data bit is set to one in the ***Source_Capabilities*** Message the Dual-Role Data bit in the ***Sink_Capabilities*** Message ***Shall*** also be set to one. If the Dual-Role Data bit is set to zero in the ***Source_Capabilities*** Message the Dual-Role Data bit in the ***Sink_Capabilities*** Message ***Shall*** also be set to zero.

6.4.1.3.1.6 Fast Role Swap USB Type-C® Current

The Fast Role Swap USB Type-C® Current field ***Shall*** indicate the current level the Sink will require after a Fast Role Swap has been performed.

The initial Source ***Shall Not*** transmit a Fast Role Swap signal if Fast Role Swap USB Type-C® Current field is set to zero.

Initially when the new Source applies ***vSafe5V*** it will have R_d asserted but ***Shall*** provide the USB Type-C® Current indicated by the new Sink in this field. If the new Source is not able to supply this level of current, it ***Shall Not*** perform a Fast Role Swap. When R_p is asserted by the new Source during the Fast Role Swap AMS (see ***Section 6.3.19 "FR_Swap Message"***), the value of USB Type-C® Current indicated by R_p ***Shall*** be the same or greater than that indicated in the Fast Role Swap USB Type-C® Current field.

6.4.1.3.2 Variable Supply (non-Battery) Power Data Object

Table 6.18 "Variable Supply (non-Battery) PDO – Sink" describes a Variable Supply (non-Battery) (10b) PDO used by a Sink. See ***Section 7.1.3 "Types of Sources"*** for the electrical requirements of the power supply.

The Voltage fields ***Shall*** be set to the output Voltage range that the Sink requires to operate. The Operational Current field ***Shall*** be set to the operational current that the Sink requires at the given Voltage range. The absolute Voltage, including any Voltage variation, ***Shall Not*** fall below the Minimum Voltage and ***Shall Not*** exceed the Maximum Voltage. Required operating current is defined as the amount of current a given device needs to be functional. This value could be the maximum current the Sink will ever require or could be sufficient to operate the Sink in one of its modes of operation.

Table 6.18 "Variable Supply (non-Battery) PDO – Sink"

Bit(s)	Description
B31…30	Variable Supply (non-Battery)
B29…20	Maximum Voltage in 50mV units
B19…10	Minimum Voltage in 50mV units
B9…0	Operational Current in 10mA units

6.4.1.3.3 Battery Supply Power Data Object

Table 6.19 "Battery Supply PDO – Sink" describes a Battery (01b) PDO used by a Sink. See ***Section 7.1.3 "Types of Sources"*** for the electrical requirements of the power supply.

The Voltage fields ***Shall*** be set to the output Voltage range that the Sink requires to operate. The Operational Power field ***Shall*** be set to the operational power that the Sink requires at the given Voltage range. The absolute Voltage,

including any Voltage variation, ***Shall Not*** fall below the Minimum Voltage and ***Shall Not*** exceed the Maximum Voltage. Note, only the Battery PDO uses power instead of current. Required operating power is defined as the amount of power a given device needs to be functional. This value could be the maximum power the Sink will ever require or could be sufficient to operate the Sink in one of its modes of operation.

Table 6.19 "Battery Supply PDO – Sink"

Bit(s)	Description
B31…30	Battery
B29…20	Maximum Voltage in 50mV units
B19…10	Minimum Voltage in 50mV units
B9…0	Operational Power in 250mW units

6.4.1.3.4 Augmented Power Data Objects

See *Section 7.1.3 "Types of Sources"* for the electrical requirements of the power supply.

The Maximum and Minimum Voltage fields ***Shall*** be set to the output Voltage range that the Sink requires to operate.

6.4.1.3.4.1 SPR Programmable Power Supply APDO

Table 6.20 "Programmable Power Supply APDO – Sink" below describes a SPR Programmable Power Supply APDO for a Sink operating in SPR Mode and consuming 21V or less. The Maximum Current field ***Shall*** be set to the maximum current the Sink requires over the Voltage range. The Maximum Current is defined as the maximum amount of current the device needs to fully support its function (e.g., Sink Directed Charge).

Table 6.20 "Programmable Power Supply APDO – Sink"

Bit(s)	Description
B31…30	11b – Augmented Power Data Object (APDO)
B29…28	00b – SPR Programmable Power Supply
B27…25	***Reserved*** – ***Shall*** be set to zero
B24…17	Maximum Voltage in 100mV increments
B16	***Reserved*** – ***Shall*** be set to zero
B15…8	Minimum Voltage in 100mV increments
B7	***Reserved*** – ***Shall*** be set to zero
B6…0	Maximum Current in 50mA increments

6.4.1.3.4.2 EPR Adjustable Voltage Supply APDO

Table 6.21 "EPR Adjustable Voltage Supply APDO – Sink" below describes a EPR Adjustable Voltage Supply APDO for a Sink operating in EPR Mode. The PDP in the EPR AVS APDO for the Sink is defined as the PDP the device needs to fully support its function.

Table 6.21 "EPR Adjustable Voltage Supply APDO – Sink"

Bit(s)	Description
B31…30	11b – Augmented Power Data Object (APDO)
B29…28	01b – EPR Adjustable Voltage Supply
B27…26	***Reserved*** **–** ***Shall*** be set to zero
B25…17	Maximum Voltage in 100mV increments
B16	***Reserved*** **–** ***Shall*** be set to zero
B15…8	Minimum Voltage in 100mV increments
B7…0	PDP in 1W increments

6.4.2 Request Message

A ***Request*** Message ***Shall*** be sent by a Sink to request power, typically during the request phase of an SPR power negotiation. The Request Data Object ***Shall*** be returned by the Sink making a request for power. It ***Shall*** be sent in response to the most recent ***Source_Capabilities*** Message (see ***Section 8.3.2.2 "Power Negotiation"***) when in SPR Mode. A ***Request*** Message ***Shall*** return one and only one Sink Request Data Object that ***Shall*** identify the Power Data Object being requested.

The Source ***Shall*** respond to a ***Request*** Message with an ***Accept*** Message, a ***Wait*** Message or a ***Reject*** Message (see ***Section 6.9 "Accept, Reject and Wait"***).

The ***Request*** Message includes the requested power level. For example, if the ***Source_Capabilities*** Message includes a Fixed Supply PDO that offers 9V @ 1.5A and if the Sink only wants 9V @ 0.5A, it will set the Operating Current field to 50 (i.e., 10mA * 50 = 0.5A). The ***Request*** Message requests the highest current the Sink will ever require in the Maximum Operating Current Field (in this example it would be 100 (100 * 10mA = 1.0A)).

The request uses a different format depending on the kind of power requested.

The Fixed Power Data Object and Variable Power Data Object share a common format shown in:

- ***Table 6.22 "Fixed and Variable Request Data Object"***.
- ***Table 6.23 "Fixed and Variable Request Data Object with GiveBack Support"***.

The Battery Power Data Object uses the format shown in:

- ***Table 6.24 "Battery Request Data Object"***.
- ***Table 6.25 "Battery Request Data Object with GiveBack Support"***.

The PPS Request Data Object's format is shown in ***Table 6.26 "PPS Request Data Object"***.

The AVS Request Data Object's format is shown in ***Table 6.27 "AVS Request Data Object"***.

The Request Data Objects are also used by the ***EPR_Request*** Message when operating in EPR Mode. See ***Section 6.4.9 "EPR_Request Message"*** for information about the use of the ***EPR_Request*** Message.

A Source operating in EPR Mode that receives a ***Request*** Message ***Shall*** initiate a Hard Reset.

Table 6.22 "Fixed and Variable Request Data Object"

Bits	Description
B31…28	Object position (0000b and 1110b…1111b are ***Reserved*** and ***Shall Not*** be used)
B27	GiveBack flag = 0
B26	Capability Mismatch
B25	USB Communications Capable
B24	No USB Suspend
B23	Unchunked Extended Messages Supported
B22	EPR Mode Capable
B21…20	***Reserved*** - ***Shall*** be set to zero.
B19…10	Operating current in 10mA units
B9…0	Maximum Operating Current 10mA units

Table 6.23 "Fixed and Variable Request Data Object with GiveBack Support"

Bits	Description
B31…28	Object position (0000b and 1110b…1111b are ***Reserved*** and ***Shall Not*** be used)
B27	GiveBack flag =1
B26	Capability Mismatch
B25	USB Communications Capable
B24	No USB Suspend
B23	Unchunked Extended Messages Supported
B22	EPR Mode Capable
B21…20	***Reserved*** - ***Shall*** be set to zero.
B19…10	Operating Current in 10mA units
B9…0	Minimum Operating Current 10mA units

Table 6.24 "Battery Request Data Object"

Bits	Description
B31…28	Object position (0000b and 1110b…1111b are ***Reserved*** and ***Shall Not*** be used)
B27	GiveBackFlag = 0
B26	Capability Mismatch
B25	USB Communications Capable
B24	No USB Suspend
B23	Unchunked Extended Messages Supported
B22	EPR Mode Capable
B21…20	***Reserved*** - ***Shall*** be set to zero.
B19…10	Operating Power in 250mW units
B9…0	Maximum Operating Power in 250mW units

Table 6.25 "Battery Request Data Object with GiveBack Support"

Bits	Description
B31…28	Object position (0000b and 1110b…1111b are ***Reserved*** and ***Shall Not*** be used)
B27	GiveBackFlag = 1
B26	Capability Mismatch
B25	USB Communications Capable
B24	No USB Suspend
B23	Unchunked Extended Messages Supported
B22	EPR Mode Capable
B21…20	***Reserved*** - ***Shall*** be set to zero.
B19…10	Operating Power in 250mW units
B9…0	Minimum Operating Power in 250mW units

Table 6.26 "PPS Request Data Object"

Bits	Description
B31…28	Object position (0000b and 1110b…1111b are ***Reserved*** and ***Shall Not*** be used)
B27	***Reserved*** – ***Shall*** be set to zero
B26	Capability Mismatch
B25	USB Communications Capable
B24	No USB Suspend
B23	Unchunked Extended Messages Supported
B22	EPR Mode Capable
B21	***Reserved*** - ***Shall*** be set to zero.
B20…9	Output Voltage in 20mV units.
B8…7	***Reserved*** - ***Shall*** be set to zero.
B6…0	Operating Current 50mA units

Table 6.27 "AVS Request Data Object"

Bits	Description
B31…28	Object position (0000b and 1110b…1111b are ***Reserved*** and ***Shall Not*** be used)
B27	***Reserved*** – ***Shall*** be set to zero
B26	Capability Mismatch
B25	USB Communications Capable
B24	No USB Suspend
B23	Unchunked Extended Messages Supported
B22	EPR Mode Capable
B21	***Reserved*** - ***Shall*** be set to zero.
B20…9	Output Voltage in 25mV units, the least two significant bits ***Shall*** be set to zero making the effective voltage step size 100mV.
B8…7	***Reserved*** - ***Shall*** be set to zero.
B6…0	Operating Current 50mA units

6.4.2.1 Object Position

The value in the Object Position field ***Shall*** indicate which object in the ***Source_Capabilities*** Message or ***EPR_Source_Capabilities*** Message the RDO refers to. The value 0001b always indicates the 5V Fixed Supply PDO as it is the first object following the ***Source_Capabilities*** Message or ***EPR_Source_Capabilities*** Message Header. The number 0010b refers to the next PDO and so forth.

The value in Object positions 0001b…0111b ***Shall*** only be used to refer to SPR PDOs. SPR PDOs ***May*** be requested by either a ***Request*** or an ***EPR_Request*** Message. Object positions 1000b…1011b ***Shall*** only be used to refer to EPR PDOs. EPR PDOs ***Shall*** only be requested by an ***EPR_Request*** Message. If the Object Position field in a ***Request*** message contains a value greater than 0111b, the Source ***Shall*** send ***Hard Reset*** Signaling.

6.4.2.2 GiveBack Flag

The GiveBack flag ***Shall*** be set to indicate that the Sink will respond to a ***GotoMin*** Message by reducing its load to the Minimum Operating Current. It will typically be used by a USB Device while charging its Battery because a short interruption of the charge will have minimal impact on the user and will allow the Source to manage its load better.

6.4.2.3 Capability Mismatch

A Capability Mismatch occurs when the Source cannot satisfy the Sink's power requirements based on the Source Capabilities it has offered. In this case the Sink ***Shall*** make a ***Valid*** request from the offered Source Capabilities and ***Shall*** set the Capability Mismatch bit (see ***Section 8.2.5.2 "Power Capability Mismatch"***). When a Capabilities Mismatch condition does not exist, the Sink ***Shall Not*** set the Capabilities Mismatch bit.

When a Sink returns a Request Data Object with the Capabilities Mismatch bit set in response to a ***Source_Capabilities*** Message, it indicates that it wants more power than the Source is currently offering. This can be due to either a specific Voltage that is not being offered or there is not sufficient current for the Voltages that are being offered.

Sources whose ***Port Reported PDP*** is less than their ***Port Present PDP*** (see ***Section 6.4.11" Source_Info Message"***) ***Shall*** respond to the Requests with the Capabilities Mismatch bit set as follows. The Source within 2 seconds of the ***PS_RDY*** Message ***Shall*** send a new Source Capabilities Message (a ***Source_Capabilities*** Message or an ***EPR_Source_Capabilities*** Message depending on operating mode) that offers either:

- The maximum power the Source can supply at this time as reported by the ***Port Present PDP*** or
- Enough power to satisfy the Sink's requirements based on the power actually required by the Sink for full operation from either the:
  - ***Sink_Capabilities_Extended*** Message (Sink Operational PDP in SPR Mode or EPR Sink Operational PDP in EPR Mode) or
  - ***Sink_Capabilities*** or ***EPR_Sink_Capabilities*** Message if the ***Sink_Capabilities_Extended*** Message is not supported by the Sink.

To prevent looping, Sources ***Should Not*** send a new ***Source_Capabilities*** or ***EPR_Source_Capabilities*** Message in response to subsequent ***Request*** Message or ***EPR_Request*** Message with the Capabilities Mismatch flag set until its Port Present PDP changes.

Once a Guaranteed Capability Source that has responded to a Capabilities Mismatch, it ***Shall Not*** subsequently send out another ***Source_Capabilities*** Message or ***EPR_Source_Capabilities*** Message at a lower PDP unless the power required by the Sink (as indicated in its ***Sink_Capabilities*** Message or ***EPR_Sink_Capabilities*** Message or ***Sink_Capabilities_Extended*** Message) has also been reduced. Sources wishing to manage their power ***May*** periodically check ***Sink_Capabilities*** or ***EPR_Sink_Capabilities*** Message or ***Sink_Capabilities_Extended*** to determine whether these have changed.

Note: a Source Capabilities Message refers to a ***Source_Capabilities*** Message or an ***EPR_Source_Capabilities*** Message, and a Sink Capabilities Message refers to a ***Sink_Capabilities*** Message or ***EPR_Sink_Capabilities*** Message depending on operating mode.

In this context a ***Valid Request*** Message means the following:

- The Object position field ***Shall*** contain a reference to an object in the last received Source Capabilities Message.
- The Operating Current/Power field ***Shall*** contain a value which is less than or equal to the maximum current/power offered in the Source Capabilities Message.

- If the GiveBack flag is set to zero i.e., there is a Maximum Operating Current/Power field:
  - If the Capability Mismatch bit is set to one:
    - The Maximum Operating Current/Power field ***May*** contain a value larger than the maximum current/power offered in the Source Capabilities Message's PDO as referenced by the Object position field. This enables the Sink to indicate that it requires more current/power than is being offered. If the Sink requires a different Voltage this will be indicated by its Sink Capabilities Message.
  - Else if the Capability Mismatch bit is set to zero:
    - The Maximum Operating Current/Power field ***Shall*** contain a value less than or equal to the maximum current/power offered in the Sink Capabilities Message's PDO as referenced by the Object position field.
- Else if the GiveBack flag is set to one i.e., there is a Minimum Operating Current/Power field:
  - The Minimum Operating Current/Power field ***Shall*** contain a value less than the Operating Current/Power field.

6.4.2.4 USB Communications Capable

The USB Communications Capable flag ***Shall*** be set to one when the Sink has USB data lines and is capable of communicating using either ***[USB 2.0]***, ***[USB 3.2]*** or ***[USB4]*** protocols. The USB Communications Capable flag ***Shall*** be set to zero when the Sink does not have USB data lines or is otherwise incapable of communicating using either ***[USB 2.0]***, ***[USB 3.2]*** or ***[USB4]*** protocols. This is used by the Source to determine operation in certain cases such as USB suspend. If the USB Communications Capable flag has been set to zero by a Sink, then the Source needs to be aware that USB Suspend rules cannot be observed by the Sink.

6.4.2.5 No USB Suspend

The No USB Suspend flag ***May*** be set by the Sink to indicate to the Source that this device is requesting to continue its Contract during USB Suspend. Sinks setting this flag typically have functionality that can use power for purposes other than USB communication e.g., for charging a Battery.

The Source uses this flag to evaluate whether it ***Should*** re-issue the ***Source_Capabilities*** Message with the USB Suspend flag cleared.

6.4.2.6 Unchunked Extended Messages Supported

The Unchunked Extended Messages Supported bit ***Shall*** be set when the Port can send and receive Extended Messages with ***Data Size*** > ***MaxExtendedMsgLegacyLen*** bytes in a single, Unchunked Message.

6.4.2.7 EPR Mode Capable

The EPR Mode Capable bit ***Shall*** indicate whether or not the Sink is capable of operating in EPR Mode. When the Sink's ability to operate in EPR Mode changes, it ***Shall*** send a new ***Request*** Message with the updated EPR Mode Capable bit set in the RDO.

6.4.2.8 Operating Current

The Operating Current field in the Request Data Object ***Shall*** be set to the actual amount of current the Sink needs to operate at a given time. A new ***Request*** Message or ***EPR_Request*** Message, with an updated Operating Current value, ***Shall*** be issued whenever the Sink's power needs change e.g., from Maximum Operating Current down to a lower current level. In conjunction with the Maximum Operating Current field or Minimum Operating Current field, it provides the Source with additional information that allows it to better manage the distribution of its power.

The Operating Current field in the SPR Programmable Request Data Object is used in addition by the Sink to request the Source for the Current Limit level it needs. When the request is accepted the Source's output current supplied into any load ***Shall*** be less than or equal to the Operating Current. When the Sink attempts to consume more current, the Source ***Shall*** reduce the output Voltage so as not to exceed the Operating Current value.

The Operating Current field in the EPR AVS Request Data Object ***Shall*** be set to the actual amount of current the Sink needs to operate at a given time. Note a Source in AVS mode, unlike the SPR Source in PPS mode, does not support current limit; the Sink is responsible not to take more current than it requested. A new ***Request*** / ***EPR_Request*** Message, with an updated Operating Current value, ***Shall*** be issued whenever the Sink's power needs change e.g., from Maximum Operating Current down to a lower current level.

The value in the Operating Current field ***Shall Not*** exceed the value in the Maximum Current field. For EPR AVS, the Operating Current field ***Shall Not*** exceed the Source PDP / Output Voltage rounded down to the nearest 50 mA.

This field ***Shall*** apply to the Fixed, Variable, Programmable and AVS RDOs.

6.4.2.9 Maximum Operating Current

The Maximum Operating Current field in the ***Request*** Message or ***EPR_Request*** Message ***Shall*** be set to the highest current the Sink will ever require. The difference between the Operating Current and Maximum Operating Current fields (when the GiveBack Flag is cleared) is used by the Device Policy Manager in the Source to calculate the size of the Power Reserve to be maintained (see ***Section 8.2.5.1 "Managing the Power Reserve"***). The Operating Current value ***Shall*** be less than or equal to the Maximum Operating Current value.

When the Capabilities Mismatch bit is set to zero the requested Maximum Operating Current ***Shall*** be less than or equal to the current in the offered Source Capabilities since the Source will need to reserve this power for future use. The Maximum Operating Current field ***Shall*** continue to be set to the highest current needed in order to maintain the allocation of the Power Reserve. If Maximum Operating Current is requested when the Power Reserve is being used by a GotoMin capable device then the resulting Message will be a ***Wait*** Message to enable the Source to reclaim the additional current (see ***Section 6.3.12.1 "Wait in response to a Request Message"*** and ***Section 8.2.5.1 "Managing the Power Reserve"***).

When the Capabilities Mismatch bit is set to one the requested Maximum Operating Current ***May*** be greater than the current in the offered Source Capabilities since the Source will need this information to ascertain the Sink's actual needs.

See ***Section 6.4.2.3 "Capability Mismatch"*** for more details of the usage of the Capabilities Mismatch bit.

This field ***Shall*** apply to the Fixed and Variable RDO in SPR mode and the Fixed RDO in EPR mode.

6.4.2.10 Minimum Operating Current

The Minimum Operating Current field in the ***Request*** Message or ***EPR_Request*** Message ***Shall*** be set to the lowest current the Sink requires to maintain operation. The difference between the Operating Current and Minimum Operating Current fields (when the GiveBack Flag is set) is used by the Device Policy Manager to calculate the amount of power which can be reclaimed using a ***GotoMin*** Message. The Operating Current value ***Shall*** be greater than the Minimum Operating Current value.

This field ***Shall*** apply to the Fixed and Variable RDO in SPR mode and the Fixed RDO in EPR mode.

6.4.2.11 Operating Power

The Operating Power field in the Request Data Object ***Shall*** be set to the actual amount of power the Sink wants at this time. In conjunction with the Maximum Operating Power field, it provides the Source with additional information that allows it to better manage the distribution of its power.

This field ***Shall*** apply to the Battery RDO.

6.4.2.12 Maximum Operating Power

The Maximum Operating Power field in the ***Request*** Message ***Shall*** be set to the highest power the Sink will ever require. This allows a Source with a power supply shared amongst multiple ports to intelligently distribute power.

When the Capabilities Mismatch bit is set to zero the requested Maximum Operating Power ***Shall*** be less than or equal to the power in the offered Source Capabilities since the Source will need to reserve this power for future use. The Maximum Operating Power field ***Shall*** continue to be set to the highest power needed in order to maintain the allocation of the Power Reserve. If Maximum Operating Power is requested when the Power Reserve is being used by a GotoMin capable device then the resulting Message will be a ***Wait*** Message to enable the Source to reclaim the additional power (see *Section 6.3.12.1 "Wait in response to a Request Message"* and *Section 8.2.5.1 "Managing the Power Reserve"*).

When the Capabilities Mismatch bit is set to one the requested Maximum Operating Power ***May*** be greater than the current in the offered Source Capabilities since the Source will need this information to ascertain the Sink's actual needs

See *Section 6.4.2.3 "Capability Mismatch"* for more details of the usage of the Capabilities Mismatch bit.

This field ***Shall*** apply to the Battery RDO.

6.4.2.13 Minimum Operating Power

The Minimum Operating Power field in the ***Request*** Message ***Shall*** be set to the lowest current the Sink requires to maintain operation. When combined with the Operating Power, it gives a Source with a power supply shared amongst multiple ports information about how much power it can temporarily get back so it can intelligently distribute power.

This field ***Shall*** apply to the Battery RDO.

6.4.2.14 Output Voltage

The Output Voltage field in the Programmable and AVS Request Data Objects ***Shall*** be set by the Sink to the Voltage the Sink requires as measured at the Source's output connector. The Output Voltage field ***Shall*** be greater than or equal to the Minimum Voltage field and less than or equal to the Maximum Voltage field in the Programmable Power Supply and AVS APDOs, respectively.

This field ***Shall*** apply to the Programmable RDO and AVS RDO.

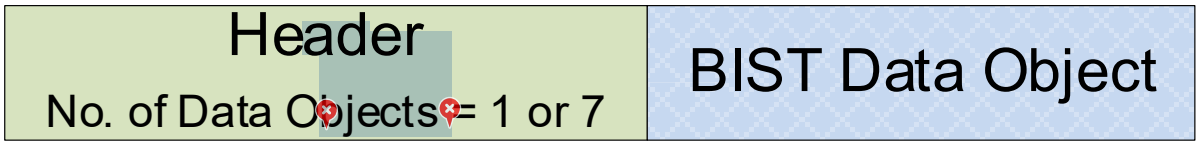
6.4.3 BIST Message

The ***BIST*** Message is sent to request the Port to enter a Physical Layer test mode (see ***Section 5.9 "Built in Self-Test (BIST)"***) that performs one of the following functions:

- Enter a Continuous BIST Mode to send a continuous stream of test data to the Tester.
- Enter and leave a shared capacity group test mode.

The Message format is as shown in ***Figure 6-13 "BIST Message"***.

Figure 6-13 "BIST Message"



All Ports ***Shall*** be able to be a Unit Under Test (UUT) only when operating at ***vSafe5V***. All of the following BIST Modes ***Shall*** be supported:

- Process reception of a ***BIST Carrier Mode*** BIST Data Object that ***Shall*** result in the generation of the appropriate carrier signal.
- Process reception of a ***BIST Test Data*** BIST Data Object that ***Shall*** result in the Message being ***Ignored***.

UUTs with Ports constituting a shared capacity group (see ***[USB Type-C 2.3]***) ***Shall*** support the following BIST Mode:

- Process reception of a ***BIST Shared Test Mode Entry*** BIST Data Object that ***Shall*** cause the UUT to enter BIST Shared Capacity Test Mode; a mode in which the UUT offers its full Source Capabilities on every port in the shared capacity group.
- Process reception of a ***BIST Shared Test Mode Exit*** BIST Data Object that ***Shall*** cause the UUT to exit the Shared Capacity Test Mode.

When a Port receives a ***BIST*** Message BIST Data Object for a BIST Mode when not operating at ***vSafe5V***, the ***BIST*** Message ***Shall*** be ***Ignored***.

When a Port receives a ***BIST*** Message BIST Data Object for a BIST Mode it does not support the ***BIST*** Message ***Shall*** be ***Ignored***.

When a Port or Cable Plug receives a ***BIST*** Message BIST Data Object for a Continuous BIST Mode the Port or Cable Plug enters the requested BIST Mode and ***Shall*** remain in that BIST Mode for ***tBISTContMode*** and then ***Shall*** return to normal operation (see ***Section 6.6.7.2 "BISTContModeTimer"***).

The usage model of the PHY Layer BIST modes generally assumes that some controlling agent will request a test of its Port Partner.

In ***Section 8.3.2.16 "Built in Self-Test (BIST)"*** there is a sequence description of the test sequences used for compliance testing.

The fields in the BIST Data Object are defined in the ***Table 6.28 "BIST Data Object"***.

Table 6.28 "BIST Data Object"

Bit(s)	Value	Parameter	Description	Reference	Applicability
B31...28	0000b...0100b	***Reserved***	***Shall Not*** be used	*Section 1.4.2.10*	-
	0101b	*BIST Carrier Mode*	Request Transmitter to enter BIST Carrier Mode	*Section 6.4.3.1*	Mandatory
	0110b...0111b	***Reserved***	***Shall Not*** be used	*Section 1.4.2.10*	
	1000b	*BIST Test Data*	Sends a Test Data Frame.	*Section 6.4.3.2*	Mandatory
	1001b	*BIST Shared Test Mode Entry*	Requests UUT to enter Shared Capacity Test Mode.		Mandatory for UUTs with shared capacity
	1010b	*BIST Shared Test Mode Exit*	Requests UUT to exit Shared Capacity Test Mode.		Mandatory for UUTs with shared capacity
	1011b...1111b	***Reserved***	***Shall Not*** be used	*Section 1.4.2.10*	-
B27...0		***Reserved***	***Shall*** be set to zero.	*Section 1.4.2.10*	

6.4.3.1 BIST Carrier Mode

Upon receipt of a *BIST* Message, with a *BIST Carrier Mode* BIST Data Object, the UUT ***Shall*** send out a continuous string of BMC encoded alternating "1"s and "0"s.

The UUT ***Shall*** exit the Continuous BIST Mode within *tBISTContMode* of this Continuous BIST Mode being enabled (see *Section 6.6.7.2 "BISTContModeTimer"*).

6.4.3.2 BIST Test Data

Upon receipt of a *BIST* Message, with a *BIST Test Data* BIST Data Object, the UUT ***Shall*** return a *GoodCRC* Message and ***Shall*** enter a test mode in which it sends no further Messages except for *GoodCRC* Messages in response to received Messages. See *Section 5.9.2 "BIST Test Data"* for the definition of the Test Data Frame.

The test ***Shall*** be ended by sending *Hard Reset* Signaling to reset the UUT.

6.4.3.3 BIST Shared Capacity Test Mode

A shared capacity group of Ports share a common power source that is not capable of simultaneously powering all the ports to their full Source Capabilities (see *[USB Type-C 2.3]*). The BIST Shared Capacity Test Mode ***Shall*** only be implemented by ports in a shared capacity group.

The UUT shared capacity group of Ports ***Shall*** contain one or more Ports, designated as Master Ports, that recognize both the *BIST Shared Test Mode Entry* BIST Data Object and the *BIST Shared Test Mode Exit* BIST Data Object..

6.4.3.3.1 BIST Shared Test Mode Entry

When any Master Port in a shared capacity group receives a BIST Message with a *BIST Shared Test Mode Entry* BIST Data Object, while in the *PE_SRC_Ready* State, the UUT ***Shall*** enter a compliance test mode where the maximum source capability is always offered on every port, regardless of the availability of shared power i.e. all shared power management is disabled.

Ports in the shared capacity group that are not Master Ports ***Shall Not*** enter compliance mode on receiving the *BIST Shared Test Mode Entry* BIST Data Object.

Upon receipt of a ***BIST*** Message, with a ***BIST Shared Test Mode Entry*** BIST Data Object, the UUT ***Shall*** return a ***GoodCRC*** Message and ***Shall*** enter the BIST Shared Capacity Test Mode.

On entering this mode, the UUT ***Shall*** send a new ***Source_Capabilities*** Message from each Port in the shared capacity group within ***tBISTSharedTestMode***. The Tester will not exceed the shared capacity during this mode.

6.4.3.3.2 BIST Shared Test Mode Exit

Upon receipt of a ***BIST*** Message, with a ***BIST Shared Test Mode Exit*** BIST Data Object, the UUT ***Shall*** return a ***GoodCRC*** Message and ***Shall*** exit the BIST Shared Capacity Test Mode. If any other Message, aside from a ***BIST*** Message, with a ***BIST Shared Test Mode Exit*** BIST Data Object, is received while in BIST Shared Capacity Test Mode this ***Shall Not*** cause the UUT to exit the BIST Shared Capacity Test Mode

On exiting the mode, the UUT ***May*** send a new ***Source_Capabilities*** Message to each port in the shared capacity group or the UUT ***May*** perform ***ErrorRecovery*** on each port.

Ports in the shared capacity group that are not Master Ports ***Shall Not*** exit compliance mode on receiving the ***BIST Shared Test Mode Entry*** BIST Data Object.

Ports in the shared capacity group that are not Master Ports ***Should Not*** exit compliance mode on receiving the ***BIST Shared Test Mode Exit*** BIST Data Object.

- The UUT ***Shall*** exit BIST Shared Capacity Test Mode when It is powered off.
- The UUT ***Shall*** remain in BIST Shared Capacity Test Mode for any PD event (except when a ***BIST Shared Test Mode Exit*** BIST Data Object, is received); specifically the UUT ***Shall*** remain in BIST Shared Capacity Test Mode when any of the following PD events occurs:
  - Hard Reset
  - Cable Reset
  - Soft Reset
  - Data Role Swap
  - Power Role Swap
  - Fast Role Swap
  - VCONN Swap.
- The UUT ***May*** leave test mode if the tester makes a request that exceeds the capabilities of the UUT.

6.4.4 Vendor Defined Message

The ***Vendor_Defined*** Message (VDM) is provided to allow vendors to exchange information outside of that defined by this specification.

A ***Vendor_Defined*** Message ***Shall*** consist of at least one Vendor Data Object, the VDM Header, and ***May*** contain up to a maximum of six additional VDM Objects (VDO).

To ensure vendor uniqueness of ***Vendor_Defined*** Messages, all ***Vendor_Defined*** Messages ***Shall*** contain a ***Valid*** USB Standard or Vendor ID (SVID) allocated by USB-IF in the VDM Header.

Two types of ***Vendor_Defined*** Messages are defined: Structured VDMs and Unstructured VDMs. A Structured VDM defines an extensible structure designed to support Modal Operation. An Unstructured VDM does not define any structure and Messages ***May*** be created in any manner that the vendor chooses.

Vendor_Defined Messages ***Shall Not*** be used for direct power negotiation. They ***May*** however be used to alter Local Policy, affecting what is offered or consumed via the normal PD Messages.

The Message format ***Shall*** be as shown in ***Figure 6-14 "Vendor Defined Message"***.

Figure 6-14 "Vendor Defined Message"

The VDM Header ***Shall*** be the first 4-byte object in a Vendor Defined Message. The VDM Header provides command space to allow vendors to customize Messages for their own purposes. Additionally, vendors ***May*** make use of the Commands in a Structured VDM.

The fields in the VDM Header for an Unstructured VDM, when the VDM Type Bit is set to zero, ***Shall*** be as defined in ***Table 6.29 "Unstructured VDM Header"***. The fields in the VDM Header for a Structured VDM, when the VDM Type Bit is set to one ***Shall*** be as defined in ***Table 6.30 "Structured VDM Header"***.

Both Unstructured and Structured VDMs ***Shall*** only be sent and received after an Explicit Contract has been established. The only exception to this is the ***Discover Identity*** Command which ***May*** be sent by Source when a Default Contract or an Implicit Contract (in place after Attach, a Power Role Swap or Fast Role Swap) is in place in order to discover Cable capabilities (see ***Section 8.3.3.26.3 "Source Startup Structured VDM Discover Identity of a Cable Plug State Diagram"***).

6.4.4.1 Unstructured VDM

The Unstructured VDM does not define the contents of bits B14…0 in the VDM Header. Their definition and use are the sole responsibility of the vendor indicated by the VID. The Port Partners and Cable Plugs ***Shall*** exit any states entered using an Unstructured VDM when a Hard Reset appears on PD.

The following rules apply to the use of Unstructured VDM Messages:

- Unstructured VDMs ***Shall*** only be used when an Explicit Contract is in place.
- Prior to establishing an Explicit Contract Unstructured VDMs ***Shall Not*** be sent and ***Shall*** be ***Ignored*** if received.
- Only the DFP ***Shall*** be an Initiator of Unstructured VDMs.
- Only the UFP or a Cable Plug ***Shall*** be a Responder to Unstructured VDM.
- Unstructured VDMs ***Shall Not*** be initiated or responded to under any other circumstances.
- Unstructured VDMs ***Shall*** only be used during Modal Operation in the context of an *Active Mode* i.e., only after the UFP has Ack'ed the ***Enter Mode*** Command can Unstructured VDMs be sent or received. The *Active Mode* and the associated Unstructured VDMs ***Shall*** use the same SVID.
- Unstructured VDMs ***May*** be used with SOP* Packets.
- When a DFP or UFP does not support Unstructured VDMs or does not recognize the VID it ***Shall*** return a ***Not_Supported*** Message.

Table 6.29 "Unstructured VDM Header" illustrates the VDM Header bits.

Table 6.29 "Unstructured VDM Header"

Bit(s)	Parameter	Description
B31…16	Vendor ID (VID)	Unique 16-bit unsigned integer. Assigned by the USB-IF to the Vendor.
B15	VDM Type	0 = Unstructured VDM
B14…0	Available for Vendor Use	Content of this field is defined by the vendor.

6.4.4.1.1 USB Vendor ID

The Vendor ID field ***Shall*** contain the 16-bit Vendor ID value assigned to the vendor by the USB-IF (VID). No other value ***Shall*** be present in this field.

6.4.4.1.2 VDM Type

The VDM Type field ***Shall*** be set to zero indicating that this is an Unstructured VDM.

6.4.4.2 Structured VDM

Setting the VDM Type field to 1 (Structured VDM) defines the use of bits B14…0 in the Structured VDM Header. The fields in the Structured VDM Header are defined in ***Table 6.30 "Structured VDM Header"***.

The following rules apply to the use of Structured VDM Messages:

- Structured VDMs ***Shall*** only be used when an Explicit Contract is in place with the following exception:
  - Prior to establishing an Explicit Contract, a Source ***May*** issue ***Discover Identity*** Messages, to a Cable Plug using SOP' Packets, as an Initiator (see ***Section 8.3.3.26.3 "Source Startup Structured VDM Discover Identity of a Cable Plug State Diagram"***).
- Either Port ***May*** be an Initiator of Structured VDMs except for the ***Enter Mode*** and ***Exit Mode*** Commands which ***Shall*** only be initiated by the DFP.
- A Cable Plug ***Shall*** only be a Responder to Structured VDMs.
- Structured VDMs ***Shall Not*** be initiated or responded to under any other circumstances.
- When a DFP or UFP does not support Structured VDMs any Structured VDMs received ***Shall*** return a ***Not_Supported*** Message.
- When using any of the SVID Specific Commands in the Structured VDM Header (VDM Header b4…0 - value 16 - 31) the responder ***Shall*** NAK messages where the SVID in the VDM Header is not recognized as an SVID that uses SVID Specific Commands or the use of SVID Specific Commands is not supported for the SVID.
- When a Cable Plug does not support Structured VDMs any Structured VDMs received ***Shall*** be ***Ignored***.
- A DFP, UFP or Cable Plug which supports Structured VDMs and receiving a Structured VDM for a SVID that it does not recognize ***Shall*** reply with a NAK Command.

Table 6.30 "Structured VDM Header"

Bit(s)	Field	Description
B31…16	Standard or Vendor ID (SVID)	Unique 16-bit unsigned integer, assigned by the USB-IF
B15	VDM Type	1 = Structured VDM
B14…13	Structured VDM Version (Major)	Version Number (Major) of the Structured VDM (not this specification Version): • Version 1.0 = 00b (***Deprecated*** and ***Shall Not*** be used) • Version 2.x = 01b • Values 2-3 are ***Reserved*** and ***Shall Not*** be used
B12…11	Structured VDM Version (Minor)	For Commands 0…15 Version Number (Minor) of the Structure VDM • Version 2.0 = 00b (Used for ports implemented prior to USB PD Revision 3.1, Version 1.6) • Version 2.1 = 01b (Used for ports implemented starting with USB PD Revision 3.1, Version 1.6) • All other Values are ***Reserved*** and ***Shall Not*** be used • SVID Specific Commands (16…31) defined by the SVID.
B10…8	Object Position	For the ***Enter Mode, Exit Mode*** and ***Attention*** Commands (Requests/Responses): • 000b = ***Reserved*** and ***Shall Not*** be used. • 001b…110b = Index into the list of VDOs to identify the desired Mode VDO • 111b = Exit all *Active Modes* (equivalent of a power on reset). ***Shall*** only be used with the ***Exit Mode*** Command. Commands 0…3, 7…15: • 000b • 001b…111b = ***Reserved*** and ***Shall Not*** be used. SVID Specific Commands (16…31) defined by the SVID.
B7…6	Command Type	00b = REQ (Request from Initiator Port) 01b = ACK (Acknowledge Response from Responder Port) 10b = NAK (Negative Acknowledge Response from Responder Port) 11b = BUSY (Busy Response from Responder Port)
B5	Reserved	***Shall*** be set to zero and ***Shall*** be ***Ignored***
B4…0	Command[1]	0 = ***Reserved***, ***Shall Not*** be used 1 = ***Discover Identity*** 2 = ***Discover SVIDs*** 3 = ***Discover Modes*** 4 = ***Enter Mode*** 5 = ***Exit Mode*** 6 = ***Attention*** 7-15 = ***Reserved***, ***Shall Not*** be used 16…31 = SVID Specific Commands

1) In the case where a SID is used the modes are defined by a standard. When a VID is used the modes are defined by the Vendor.

Table 6.31 "Structured VDM Commands" shows the Commands, which SVID to use with each Command and the *SOP** values which ***Shall*** be used.

Table 6.31 "Structured VDM Commands"

Command	VDM Header SVID Field	SOP* used
Discover Identity	***Shall*** only use the *PD SID*.	***Shall*** only use *SOP*/*SOP'*.
Discover SVIDs	***Shall*** only use the *PD SID.*	***Shall*** only use *SOP*/*SOP'*.
Discover Modes	***Valid*** with any SVID.	***Shall*** only use *SOP*/*SOP'*.
Enter Mode	***Valid*** with any SVID.	***Valid*** with *SOP**.
Exit Mode	***Valid*** with any SVID.	***Valid*** with *SOP**.
Attention	***Valid*** with any SVID.	***Valid*** with *SOP*.
SVID Specific Commands	***Valid*** with any SVID.	***Valid*** with *SOP** (defined by SVID).

6.4.4.2.1 SVID

The SVID field ***Shall*** contain either a 16-bit USB Standard ID value (SID) or the 16-bit assigned to the vendor by the USB-IF (VID). No other value ***Shall*** be present in this field.

Table 6.32 "SVID Values" lists specific SVID values referenced by this specification.

Table 6.32 "SVID Values"

Parameter	Value	Description
PD SID	0xFF00	Standard ID allocated to this specification.

6.4.4.2.2 VDM Type

The VDM Type field ***Shall*** be set to one indicating that this is a Structured VDM.

6.4.4.2.3 Structured VDM Version

The Structured VDM Version fields indicate the level of functionality supported in the Structured VDM part of the specification. This is not the same version as the version of this specification. The Structured VDM Version (Major) ***Shall*** be set to 01b to indicate Version 2.x with the Structured VDM Version (Minor) field set as appropriate based on whether the port is implemented to USB PD Revision 3.1, Version 1.6 (or newer) or a prior version.

To ensure interoperability with existing USBPD Products, USBPD Products ***Shall*** support every Structured VDM Version number starting from Version 1.0.

On receipt of a VDM Header with a higher Version number than it supports, a Port or Cable Plug ***Shall*** respond using the highest Version number it supports. On receipt of a VDM Header with a lower Version number than it supports, a Port or Cable Plug ***Shall*** respond using the same Version number it received.A

The Structured VDM Version field of the *Discover Identity* Command sent and received during VDM discovery ***Shall*** be used to determine the lowest common Structured VDM Version supported by the Port Partners or Cable Plug and ***Shall*** continue to operate using this Specification Revision until they are Detached. After discovering the Structure VDM Version, the Structured VDM Version field ***Shall*** match the agreed common Structured VDM Version.

6.4.4.2.4 Object Position

The Object Position field ***Shall*** be used by the ***Enter Mode*** and ***Exit Mode*** Commands. The ***Discover Modes*** Command returns a list of zero to six VDOs, each of which describes a Mode. The value in Object Position field is an index into that list that indicates which VDO (e.g., Mode) in the list the ***Enter Mode*** and ***Exit Mode*** Command refers to. The Object Position ***Shall*** start with one for the first Mode in the list. If the SVID is a VID, the content of the VDO for the Mode ***Shall*** be defined by the vendor. If the SVID is a SID, the content ***Shall*** be defined by the Standard. The VDO's content ***May*** be as simple as a numeric value or as complex as bit mapped description of capabilities of the Mode. In all cases, the Responder is responsible for deciphering the contents to know whether or not it supports the Mode at the Object Position.

This field ***Shall*** be set to zero in the Request or Response (REQ, ACK, NAK or BUSY) when not required by the specification of the individual Command.

6.4.4.2.5 Command Type

6.4.4.2.5.1 Commands other than Attention

This Command Type field ***Shall*** be used to indicate the type of Command request/response being sent.

An Initiator ***Shall*** set the field to REQ to indicate that this is a Command request from an Initiator.

If Structured VDMs are supported, then the responses are as follows:

- "Responder ACK" is the normal return and ***Shall*** be sent to indicate that the Command request was received and handled normally.
- "Responder NAK" ***Shall*** be returned when the Command request:
- Has an ***Invalid*** parameter (e.g., ***Invalid*** SVID or Mode).
- Cannot be acted upon because the configuration is not correct (e.g., a Mode which has a dependency on another Mode or a request to exit a Mode which is not Active).
- Is an Unrecognized Message.
- The handling of "Responder NAK" is left up to the Initiator.
- "Responder BUSY" ***Shall*** be sent in the response to a VDM when the Responder is unable to respond to the Command request immediately, but the Command request ***May*** be retried. The Initiator ***Shall*** wait ***tVDMBusy*** after a "Responder BUSY" response is received before retrying the Command request.

6.4.4.2.5.2 Attention Command

This Command Type field ***Shall*** be used to indicate the type of Command request being sent. An Initiator ***Shall*** set the field to REQ to indicate that this is a Command request from an Initiator. If Structured VDMs are supported, then no response ***Shall*** be made to an ***Attention*** Command.

6.4.4.2.6 Command

6.4.4.2.6.1 Commands other than Attention

This field contains the value for the VDM Command being sent. The Commands explicitly listed in this field are used to identify devices and manage their operational Modes. There is a further range of Command values left for the vendor to use to manage additional extensions.

A Structured VDM Command consists of a Command request and a Command response (ACK, NAK or BUSY). A Structured VDM Command is deemed to be completed (and if applicable, the transition to the requested functionality

is made) when the ***GoodCRC*** Message has been successfully received by the Responder in reply to its Command response.

If Structured VDMs are supported, but the Structured VDM Command request is an Unrecognized Message, it ***Shall*** be NAKed (see ***Table 6.33 "Commands and Responses"***).

6.4.4.2.6.2 Attention Command

This field contains the value for the VDM Command being sent (***Attention***). The ***Attention*** Command ***May*** be used by the Initiator to notify the Responder that it requires service.

A Structured VDM ***Attention*** Command consists of a Command request but no Command response. A Structured VDM ***Attention*** Command is deemed to be completed when the ***GoodCRC*** Message has been successfully received by the Initiator in reply to its ***Attention*** Command request.

If Structured VDMs are supported, but the Structured VDM ***Attention*** Command request is an Unrecognized Message it ***Shall*** be ***Ignored*** (see ***Table 6.33 "Commands and Responses"***).

6.4.4.3 Use of Commands

The VDM Header for a Structured VDM Message defines Commands used to retrieve a list of SVIDs the device supports, to discover the Modes associated with each SVID, and to enter/exit the Modes. The Commands include:

- ***Discover Identity***.
- ***Discover SVIDs***.
- ***Discover Modes***.
- ***Enter Mode***.
- ***Exit Mode***.
- ***Attention***.

Additional Command space is also ***Reserved*** for Standard and Vendor use and for future extensions.

The Command sequences use the terms Initiator and Responder to identify messaging roles the ports are taking on relative to each other. This role is independent of the Port's power capability (Provider, Consumer etc.) or its present power role (Source or Sink). The Initiator is the Port sending the initial Command request and the Responder is the Port replying with the Command response. See *Section 6.4.4.4 "Command Processes"*.

All Ports that support Modes ***Shall*** support the ***Discover Identity***, ***Discover SVIDs***, the ***Discover Modes***, the ***Enter Mode*** and ***Exit Mode*** Commands.

Table 6.33 "Commands and Responses" details the responses a Responder ***May*** issue to each Command request. Responses not listed for a given Command ***Shall Not*** be sent by a Responder. A NAK response ***Should*** be taken as an indication not to retry that particular Command.

Table 6.33 "Commands and Responses"

Command	Allowed Response	Reference
Discover Identity	ACK, NAK, BUSY	*Section 6.4.4.3.1*
Discover SVIDs	ACK, NAK, BUSY	*Section 6.4.4.3.2*
Discover Modes	ACK, NAK, BUSY	*Section 6.4.4.3.3*
Enter Mode	ACK, NAK	*Section 6.4.4.3.4*
Exit Mode	ACK, NAK	*Section 6.4.4.3.5*
Attention	None	*Section 6.4.4.3.6*

Examples of Command usage can be found in *Appendix C "VDM Command Examples"*.

6.4.4.3.1 Discover Identity

The ***Discover Identity*** Command is provided to enable an Initiator to identify its Port Partner and for an Initiator (VCONN Source) to identify the Responder (Cable Plug or VPD). The ***Discover Identity*** Command is also used to determine whether a Cable Plug or VPD is PD-Capable by looking for a ***GoodCRC*** Message Response.

The ***Discover Identity*** Command ***Shall*** only be sent to ***SOP*** when there is an Explicit Contract.

The ***Discover Identity*** Command ***Shall*** be used to determine whether a given Cable Plug or VPD is PD Capable (see ***Section 8.3.3.22.1 "Initiator Structured VDM Discover Identity State Diagram"*** and S***ection 8.3.3.26.3 "Source Startup Structured VDM Discover Identity of a Cable Plug State Diagram"***). In this case a ***Discover Identity*** Command request sent to SOP' ***Shall Not*** cause a Soft Reset if a ***GoodCRC*** Message response is not returned since this can indicate a non-PD Capable cable or VPD. Note that a Cable Plug or VPD will not be ready for PD Communication until tVCONNStable after VCONN has been applied (see ***[USB Type-C 2.3]***). During Cable Plug or VPD discovery, when there is an Explicit Contract, ***Discover Identity*** Commands are sent at a rate defined by the ***DiscoverIdentityTimer*** (see ***Section 6.6.15 "DiscoverIdentityTimer"***) up to a maximum of ***nDiscoverIdentityCount*** times (see ***Section 6.7.5 "Discover Identity Counter"***).

A PD-Capable Cable Plug or VPD ***Shall*** return a ***Discover Identity*** Command ACK in response to a ***Discover Identity*** Command request sent to ***SOP'***.

The ***Discover Identity*** Command ***Shall*** be used to determine the identity and/or capabilities of the Port Partner. The following products ***Shall*** return a ***Discover Identity*** Command ACK in response to a ***Discover Identity*** Command request sent to ***SOP***:

- A PD-Capable UFP that supports Modal Operation.
- A PD-Capable product that has multiple DFPs.
- A PD-Capable ***[USB4]*** product.

The SVID in the ***Discover Identity*** Command request ***Shall*** be set to the ***PD SID*** (see ***Table 6.32 "SVID Values"***).

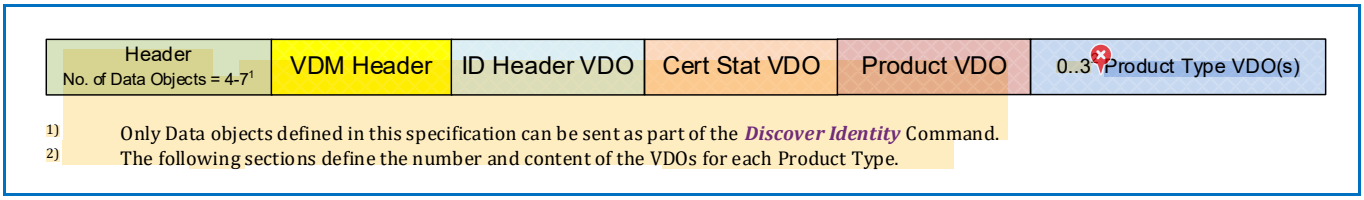
The ***Number of Data Objects*** field in the Message Header in the ***Discover Identity*** Command request ***Shall*** be set to 1 since the ***Discover Identity*** Command request ***Shall Not*** contain any VDOs.

The ***Discover Identity*** Command ACK sent back by the Responder ***Shall*** contain an ID Header VDO, a Cert Stat VDO, a Product VDO and the Product Type VDOs defined by the Product Type as shown in ***Figure 6-15 "Discover Identity Command response"***. This specification defines the following Product Type VDOs:

- Passive Cable VDO (see ***Section 6.4.4.3.1.6 "Passive Cable VDO"***)
- *Active Cable* VDOs (see ***Section 6.4.4.3.1.7 "Active Cable VDOs"***)
- VCONN Powered USB Device (VPD) VDO (see ***Section 6.4.4.3.1.9 "Vconn Powered USB Device VDO"***)
- UFP VDO (see ***Section 6.4.4.3.1.4 "UFP VDO"***)
- DFP VDO (see ***Section 6.4.4.3.1.5 "DFP VDO"***)

No VDOs other than those defined in this specification ***Shall*** be sent as part of the ***Discover Identity*** Command response. Where there is no Product Type VDO defined for a specific Product Type, no VDOs ***Shall*** be sent as part of the ***Discover Identity*** Command response. Any additional VDOs received by the initiator ***Shall*** be ***Ignored***.

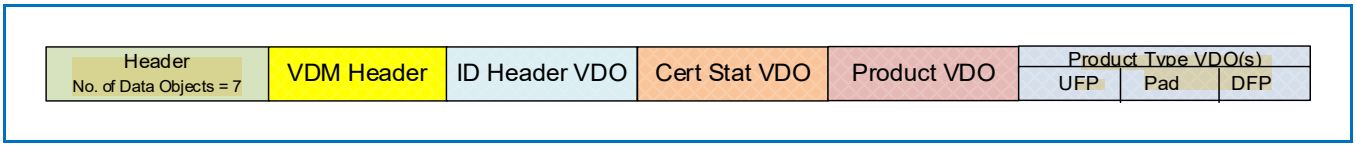
Figure 6-15 "Discover Identity Command response"



The ***Number of Data Objects*** field in the Message Header in the ***Discover Identity*** Command NAK and BUSY responses ***Shall*** be set to 1 since they ***Shall Not*** contain any VDOs.

If the product is a DRD both a Product Type (UFP) and a Product Type (DFP) are declared in the ID Header. These products ***Shall*** return Product Type VDOs for both UFP and DFP beginning with the UFP VDO, then by a 32-bit Pad Object (defined as all '0's), followed by the DFP VDO as shown in ***Figure 6-16 "Discover Identity Command response for a DRD"***.

Figure 6-16 "Discover Identity Command response for a DRD"



6.4.4.3.1.1 ID Header VDO

The ID Header VDO contains information corresponding to the Power Delivery Product. The fields in the ID Header VDO ***Shall*** be as defined in ***Table 6.34 "ID Header VDO"***.

Table 6.34 "ID Header VDO"

Bit(s)	Description	Reference
B31	USB Communications Capable as USB Host: • ***Shall*** be set to one if the product is capable of enumerating USB Devices. • ***Shall*** be set to zero otherwise.	*Section 6.4.4.3.1.1.1*
B30	USB Communications Capable as a USB Device: • ***Shall*** be set to one if the product is capable of being enumerated as a USB Device. • ***Shall*** be set to zero otherwise	*Section 6.4.4.3.1.1.2*
B29…27	SOP Product Type (UFP): • 000b – Not a UFP • 001b – PDUSB Hub • 010b – PDUSB Peripheral • 011b – PSD • 100b…111b – ***Reserved*** , ***Shall Not*** be used. SOP' Product Type (Cable Plug/VPD): • 000b – Not a Cable Plug/VPD • 001b…010b – ***Reserved*** , ***Shall Not*** be used. • 011b – Passive Cable • 100b – *Active Cable* • 101b – ***Reserved*** , ***Shall Not*** be used. • 110b – VCONN-Powered USB Device (VPD) • 111b – ***Reserved*** , ***Shall Not*** be used.	*Section 6.4.4.3.1.1.3*
B26	Modal Operation Supported: • ***Shall*** be set to one if the product (UFP/Cable Plug) is capable of supporting Modal Operation (Alternate Modes). • ***Shall*** be set to zero otherwise	*Section 6.4.4.3.1.1.4*
B25…23	SOP - Product Type (DFP): • 000b – Not a DFP • 001b – PDUSB Hub • 010b – PDUSB Host • 011b – Power Brick • 100b…111b – ***Reserved*** , ***Shall Not*** be used. SOP': ***Reserved*** , ***Shall Not*** be used.	
B22…21	Connector Type: • 00b – Reserved, for compatibility with legacy systems. • 01b – ***Reserved*** , ***Shall Not*** be used. • 10b – USB Type-C® Receptacle • 11b – USB Type-C® Plug	
B20…16	***Reserved***, ***Shall*** be set to zero.	
B15…0	USB Vendor ID.	*[USB 2.0]*/*[USB 3.2]*/*[USB4]*

6.4.4.3.1.1.1 USB Communications Capable as a USB Host

The USB Communications Capable as a USB Host field is used to indicate whether or not the Port has a USB Host Capability.

6.4.4.3.1.1.2 USB Communications Capable as a USB Device

The USB Communications Capable as a USB Device field is used to indicate whether or not the Port has a USB Device Capability.

6.4.4.3.1.1.3 Product Type (UFP)

The Product Type (UFP) field indicates the type of Product when in UFP Data Role, whether a VDO will be returned and if so the type of VDO to be returned. The Product Type indicated in the Product Type (UFP) field ***Shall*** be the closest categorization of the main functionality of the Product in UFP Data Role or "Undefined" when there is no suitable category for the product. For DRD Products this field ***Shall*** always indicate the Product Type when in UFP role regardless of the present Data Role. *Table 6.35 "Product Types (UFP)"* defines the Product Type VDOs which ***Shall*** be returned.

Table 6.35 "Product Types (UFP)"

Product Type	Description	Product Type VDO	Reference
Undefined	***Shall*** be used when this is not a UFP.	None	
PDUSB Hub	***Shall*** be used when the Product is a PDUSB Hub.	UFP VDO	*Section 6.4.4.3.1.4*
PDUSB Peripheral	***Shall*** be used when the Product is a PDUSB Device other than a PDUSB Hub.	UFP VDO	*Section 6.4.4.3.1.4*
PSD	***Shall*** be used when the Product is a PSD, e.g., power bank.	None	

6.4.4.3.1.1.4 Product Type (Cable Plug)

The Product Type (Cable Plug) field indicates the type of Product when the Product is a Cable Plug, whether a VDO will be returned and if so the type of VDO to be returned. *Table 6.36 "Product Types (Cable Plug/VPD)"* defines the Product Type VDOs which ***Shall*** be returned.

Table 6.36 "Product Types (Cable Plug/VPD)"

Product Type	Description	Product Type VDO	Reference
Undefined	***Shall*** be used where no other Product Type value is appropriate.	None	
Active Cable	***Shall*** be used when the Product is a cable that incorporates signal conditioning circuits.	*Active Cable* VDO	*Section 6.4.4.3.1.7*
Passive Cable	***Shall*** be used when the Product is a cable that does not incorporate signal conditioning circuits.	Passive Cable VDO	*Section 6.4.4.3.1.6*
VCONN Powered USB Device	***Shall*** be used when the Product is a PDUSB VCONN Powered USB Device.	VPD VDO	*Section 6.4.4.3.1.9*

6.4.4.3.1.1.5 Modal Operation Supported

The Modal Operation Supported bit is used to indicate whether or not the Product (either a Cable Plug or a device that can operate in the UFP role) is capable of supporting Modes. The Modal Operation Supported bit does not describe a DFP's Alternate Mode Controller functionality.

A product that supports Modal Operation ***Shall*** respond to the ***Discover SVIDs*** Command with a list of SVIDs for all of the Modes it is capable of supporting whether or not those Modes can currently be entered.

6.4.4.3.1.1.6 Product Type (DFP)

The Product Type (DFP) field indicates the type of Product when in DFP Data Role, whether a VDO will be returned and if so the type of VDO to be returned. The Product Type indicated in the Product Type (DFP) field ***Shall*** be the closest categorization of the main functionality of the Product in DFP Data Role or "Undefined" when there is no suitable category for the product. For DRD Products this field ***Shall*** always indicate the Product Type when in DFP role regardless of the present Data Role. *Table 6.37 "Product Types (DFP)"* defines the Product Type VDOs which ***Shall*** be returned.

In SOP' Communication (Cable Plugs and VPDs) this bit field is ***Reserved*** and ***Shall*** be set to zero.

Table 6.37 "Product Types (DFP)"

Product Type	Description	Product Type VDO	Reference
Undefined	***Shall*** be used where no other Product Type value is appropriate.	None	
PDUSB Hub	***Shall*** be used when the Product is a PDUSB Hub.	DFP VDO	*Section 6.4.4.3.1.5*
PDUSB Host	***Shall*** be used when the Product is a PDUSB Host or a PDUSB host that supports one or more alternate modes as an AMC.	DFP VDO	*Section 6.4.4.3.1.5*
Power Brick	***Shall*** be used when the Product is a Power Brick/Wall Wart.	DFP VDO	*Section 6.4.4.3.1.5*

6.4.4.3.1.1.7 Connector Type Field

The Connector Type field (B22…21) ***Shall*** contain a value identifying it as either a USB Type-C® receptacle or a USB Type-C® plug.

6.4.4.3.1.1.8 Vendor ID

Manufacturers ***Shall*** set the Vendor ID field to the value of the Vendor ID assigned to them by USB-IF. For USB Devices or Hubs which support USB communications the Vendor ID field ***Shall*** be identical to the Vendor ID field defined in the product's USB Device Descriptor (see *[USB 2.0]* and *[USB 3.2]*).

6.4.4.3.1.2 Cert Stat VDO

The Cert Stat VDO ***Shall*** contain the XID assigned by USB-IF to the product before certification in binary format. The fields in the Cert Stat VDO ***Shall*** be as defined in *Table 6.38 "Cert Stat VDO"*.

Table 6.38 "Cert Stat VDO"

Bit(s)	Description	Reference
B31…0	32-bit unsigned integer, XID	Assigned by USB-IF

6.4.4.3.1.3 Product VDO

The Product VDO contains identity information relating to the product. The fields in the Product VDO ***Shall*** be as defined in *Table 6.39 "Product VDO"*.

Table 6.39 "Product VDO"

Bit(s)	Description	Reference
B31...16	16-bit unsigned integer. USB Product ID	*[USB 2.0]*/*[USB 3.2]*
B15...0	16-bit unsigned integer. bcdDevice	*[USB 2.0]*/*[USB 3.2]*

Manufacturers ***Should*** set the USB Product ID field to a unique value identifying the product and ***Should*** set the bcdDevice field to a version number relevant to the release version of the product.

6.4.4.3.1.4 UFP VDO

The UFP VDO defined in this section ***Shall*** be returned by Ports capable of operating as a UFP including traditional USB peripherals, USB hub's upstream Port and DRD capable host Ports. The UFP VDO defined in this section ***Shall*** be sent when the Product Type (UFP) field in the ID Header VDO is given as a PDUSB Peripheral or PDUSB Hub. *Table 6.40 "UFP VDO"* defines the UFP VDO that ***Shall*** be sent based on the Product Type.

A *[USB4]* UFP ***Shall*** support the Structured VDM *Discover Identity* Command.

Table 6.40 "UFP VDO"

<table>
<tr><th>Bit(s)</th><th>Field</th><th>Description</th></tr>
<tr><td>B31…29</td><td>UFP VDO Version</td><td>Version Number of the VDO (not this specification Version):
• Version 1.3 = 011b
Values 100b…111b are Reserved and Shall Not be used</td></tr>
<tr><td>B28</td><td>Reserved</td><td>Shall be set to zero.</td></tr>
<tr><td>B27…24</td><td>Device Capability</td><td><table><tr><th>Bit</th><th>Description</th></tr><tr><td>0</td><td>[USB 2.0] Device Capable</td></tr><tr><td>1</td><td>[USB 2.0] Device Capable (Billboard only)</td></tr><tr><td>2</td><td>[USB 3.2] Device Capable</td></tr><tr><td>3</td><td>[USB4] Device Capable</td></tr></table></td></tr>
<tr><td>B23…22</td><td>Connector Type (Legacy)</td><td>Shall be set to 00b</td></tr>
<tr><td>B21…11</td><td>Reserved</td><td>Shall be set to zero.</td></tr>
<tr><td>B10…8</td><td>VCONN Power</td><td>When the VCONN required field is set to "Yes" the VCONN Power Field indicates the VCONN power needed by the AMA for full functionality:
• 000b = 1W
• 001b = 1.5W
• 010b = 2W
• 011b = 3W
• 100b = 4W
• 101b = 5W
• 110b = 6W
• 111b = Reserved, Shall Not be used
When the VCONN required field is set to "No" the VCONN Power Field is Reserved and Shall be set to zero.</td></tr>
<tr><td>B7</td><td>VCONN Required</td><td>Indicates whether the AMA requires VCONN in order to function.
• 0 = No
• 1 = Yes
When the Alternate Modes field indicates no modes are supported, the VCONN Required field is Reserved and Shall be set to zero.</td></tr>
<tr><td>B6</td><td>VBUS Required</td><td>Indicates whether the AMA requires VBUS in order to function.
• 0 = Yes
• 1 = No
When the Alternate Modes field indicates no modes are supported, the VBUS Required field is Reserved and Shall be set to zero.</td></tr>
<tr><td>B5…3</td><td>Alternate Modes</td><td><table><tr><th>Bit</th><th>Description</th></tr><tr><td>0</td><td>Supports [TBT3] Alternate Mode</td></tr><tr><td>1</td><td>Supports Alternate Modes that reconfigure the signals on the [USB Type-C 2.3] connector – except for [TBT3].</td></tr><tr><td>2</td><td>Supports Alternate Modes that do not reconfigure the signals on the [USB Type-C 2.3] connector</td></tr></table></td></tr>
</table>

B2…0	USB Highest Speed	000b = *[USB 2.0]* only, no SuperSpeed support 001b = *[USB 3.2]* Gen1 010b = *[USB 3.2]*/*[USB4]* Gen2 011b = *[USB4]* Gen3 100b = *[USB4]* Gen4 101b…111b = ***Reserved***, ***Shall Not*** be used

6.4.4.3.1.4.1 VDO Version Field

The UFP VDO Version field contains a VDO version for this VDM version number. This field indicates the expected content for the UFP VDOs.

6.4.4.3.1.4.2 Device Capability Field

The Device Capability bit-field describes the UFP's capabilities when operating as either a PDUSB Device or PDUSB Hub.

The bits in the bit-field ***Shall*** be non-zero when the corresponding USB Device speed is supported and ***Shall*** be set to zero when the corresponding USB Device speed is not supported.

[USB 2.0] "Device capable" and "Device capable Billboard only" (bits 0 and 1) ***Shall Not*** be simultaneously set.

6.4.4.3.1.4.3 Connector Type Field

This field was previously used for the UFP VDO's Connector Type. ***Shall*** be set to 00b by the Cable Plug and ***Shall*** be ***Ignored*** by the receiver. The receiver can find this information in the Connector Type Field in the ID Header VDO (6.4.4.3.1.1.7).

6.4.4.3.1.4.4 VCONN Power Field

When the VCONN required field indicates that VCONN is required the VCONN power field ***Shall*** indicate how much power an AMA needs in order to fully operate. When the VCONN required field is set to "No" the VCONN Power Field is ***Reserved*** and ***Shall*** be set to zero.

6.4.4.3.1.4.5 VCONN Required Field

The VCONN required field ***Shall*** indicate whether VCONN is needed for the AMA to operate. The VCONN required field ***Shall*** only be used if the Alternate Modes fields indicates that an Alternate Mode is supported. If no alternate modes are supported, this field is ***Reserved*** and ***Shall*** be set to zero.

6.4.4.3.1.4.6 VBUS Required Field

The VBUS required field ***Shall*** indicate whether VBUS is needed for the AMA to operate. The VBUS required field ***Shall*** only be used if the Alternate Modes fields indicates that an Alternate Mode is supported. If no alternate modes are supported, this field is ***Reserved*** and ***Shall*** be set to zero.

6.4.4.3.1.4.7 Alternate Modes Field

The Alternate Mode field ***Shall*** be used to identify all the types of Alternate Modes, if any, a device supports.

6.4.4.3.1.4.8 USB Highest Speed Field

The USB Highest Speed field ***Shall*** indicate the port's highest signaling capability. The DFP ***Shall*** consider all values indicated in this field that are higher than the highest value that the DFP recognizes as being ***Valid*** and functionally compatible with the highest speed that the DFP supports.

6.4.4.3.1.5 DFP VDO

The DFP VDO ***Shall*** be returned by Ports capable of operating as a DFP; including those implemented by Hosts, Hubs and Power Bricks. The DFP VDO ***Shall*** be returned when the Product Type (DFP) field in the ID Header VDO is given as Power Brick, PDUSB Host or PDUSB Hub. *Table 6.41 "DFP VDO"* defines the DFP VDO that ***Shall*** be sent.

Table 6.41 "DFP VDO"

Bit(s)	Field	Description
B31…29	DFP VDO Version	Version Number of the VDO (not this specification Version): • Version 1.2 = 010b Values 011b…111b are ***Reserved*** and ***Shall Not*** be used
B28…27	***Reserved***	***Shall*** be set to zero.
B26…24	Host Capability	Bit 0: *[USB 2.0]* Host Capable Bit 1: *[USB 3.2]* Host Capable Bit 2: *[USB4]* Host Capable
B23…22	Connector Type (Legacy)	***Shall*** be set to 00b.
B21…5	***Reserved***	***Shall*** be set to zero.
B4…0	Port Number	Unique port number to identify a specific port on a multi-port device.

6.4.4.3.1.5.1 VDO Version Field

The DFP VDO Version field ***Shall*** contain a VDO version for this VDM version number. This field indicates the expected content for the DFP VDO.

6.4.4.3.1.5.2 Host Capability Field

The Host Capability field bit-field ***Shall*** describe whether the DFP can operate as a PDUSB Host and the DFP's capabilities when operating as a PDUSB Host.

Power Bricks and PDHUB Hubs ***Shall*** set the Host Capability bits to zero.

6.4.4.3.1.5.3 Connector Type Field

This field was previously used for the UFP VDO's Connector Type. ***Shall*** be set to 00b by the Cable Plug and ***Shall*** be ***Ignored*** by the receiver. The receiver can find this information in the Connector Type Field in the ID Header VDO (6.4.4.3.1.1.7).

6.4.4.3.1.5.4 Port Number Field

The Port Number field ***Shall*** be a static unique number that unambiguously identifies each *[USB Type-C 2.3]* DFP, including DRPs, on the device. Note that this number is independent of the USB port number.

6.4.4.3.1.6 Passive Cable VDO

The Passive Cable VDO defined in this section ***Shall*** be sent when the Product Type is given as Passive Cable. *Table 6.42 "Passive Cable VDO"* defines the Cable VDO which ***Shall*** be sent.

A Passive Cable has a USB Plug on each end at least one of which is a Cable Plug supporting SOP' Communication. A Passive Cable ***Shall Not*** incorporate data bus signal conditioning circuits and hence has no concept of Super Speed Directionality. A Passive Cable ***Shall*** include a V_{BUS} wire and ***Shall*** only respond to SOP' Communication. Passive

Cables ***Shall*** support the Structured VDM ***Discover Identity*** Command and ***Shall*** return the Passive Cable VDO in a ***Discover Identity*** Command ACK as shown in ***Table 6.42 "Passive Cable VDO"***.

Table 6.42 "Passive Cable VDO"

Bit(s)	Field	Description
B31…28	HW Version	0000b…1111b assigned by the VID owner
B27…24	Firmware Version	0000b…1111b assigned by the VID owner
B23…21	VDO Version	Version Number of the VDO (not this specification Version): • Version 1.0 = 000b Values 001b…111b are ***Reserved*** and ***Shall Not*** be used
B20	***Reserved***	***Shall*** be set to zero.
B19…18	USB Type-C® plug to USB Type-C®/Captive	00b = ***Reserved***, ***Shall Not*** be used 01b = ***Reserved***, ***Shall Not*** be used 10b = USB Type-C® 11b = Captive
B17	EPR Mode Capable	0b – Cable is not EPR Mode Capable 1b = Cable is EPR Mode Capable
B16…13	Cable Latency	0000b – ***Reserved***, ***Shall Not*** be used. 0001b – <10ns (~1m) 0010b – 10ns to 20ns (~2m) 0011b – 20ns to 30ns (~3m) 0100b – 30ns to 40ns (~4m) 0101b – 40ns to 50ns (~5m) 0110b – 50ns to 60ns (~6m) 0111b – 60ns to 70ns (~7m) 1000b – > 70ns (>~7m) 1001b ….1111b ***Reserved***, ***Shall Not*** be used. Includes latency of electronics in *Active Cable.*
B12…11	Cable Termination Type	00b = VCONN not required. Cable Plugs that only support ***Discover Identity*** Commands ***Shall*** set these bits to 00b. 01b = VCONN required 10b…11b = ***Reserved***, ***Shall Not*** be used
B10…9	Maximum VBUS Voltage	Maximum Cable VBUS Voltage: 00b – 20V 01b – 30V[1] (***Deprecated***) 10b – 40V[1] (***Deprecated***) 11b – 50V
B8…7	***Reserved***	***Shall*** be set to zero.
B6…5	VBUS Current Handling Capability	00b = ***Reserved***, ***Shall Not*** be used. 01b = 3A 10b = 5A 11b = ***Reserved***, ***Shall Not*** be used.
B4…3	***Reserved***	***Shall*** be set to zero.

B2…0	USB Highest Speed	000b = ***[USB 2.0]*** only, no SuperSpeed support 001b = ***[USB 3.2]*** Gen1 010b = ***[USB 3.2]***/***[USB4]*** Gen2 011b = ***[USB4]*** Gen3 100b = ***[USB4]*** Gen4 101b…111b = ***Reserved***, ***Shall Not*** be used

1) Values no longer allowed. When present the field ***Shall*** be interpreted as if it was 00b.

2) EPR Sinks with a captive cable ***Shall*** report 50V.

6.4.4.3.1.6.1 HW Version Field

The HW Version field (B31…28) contains a HW Version assigned by the VID owner.

6.4.4.3.1.6.2 FW Version Field

The FW Version field (B27…24) contains a FW Version assigned by the VID owner.

6.4.4.3.1.6.3 VDO Version Field

The VDO Version field (B23…20) contains a VDO version for this VDM version number. This field indicates the expected content for this VDO.

6.4.4.3.1.6.4 USB Type-C® plug to USB Type-C®/Captive Field

The USB Type-C® plug to USB Type-C®/Captive field (B19…18) ***Shall*** contain a value indicating whether the opposite end from the USB Type-C® plug is another USB Type-C® plug (i.e., a detachable Standard USB Type-C® Cable Assembly) or is a Captive Cable Assembly.

6.4.4.3.1.6.5 EPR Mode Capable

A static bit which ***Shall*** only be set when the cable is specifically designed for safe operation when carrying up to 48 volts at 5 amps.

6.4.4.3.1.6.6 Cable Latency Field

The Cable Latency field (B16…13) ***Shall*** contain a value corresponding to the signal latency through the cable which can be used as an approximation for its length.

6.4.4.3.1.6.7 Cable Termination Type Field

The Cable Termination Type field (B12…11) ***Shall*** contain a value indicating whether the Passive Cable needs V_{CONN} only initially in order to support the ***Discover Identity*** Command, after which it can be removed, or the Passive Cable needs V_{CONN} to be continuously applied in order to power some feature of the Cable Plug.

6.4.4.3.1.6.8 Maximum V_{BUS} Voltage Field

The Maximum V_{BUS} Voltage field (B10…9) ***Shall*** contain the maximum Voltage that ***Shall*** be negotiated using a Fixed Supply over the cable as part of an Explicit Contract where the maximum Voltage that ***Shall*** be applied to the cable is ***vSrcNew*** max + ***vSrcValid*** max. For example, when the Maximum SPR V_{BUS} Voltage field is 20V, a Fixed Supply of 20V can be negotiated as part of an Explicit Contract where the absolute maximum Voltage that can be applied to the cable is 21.55V. Similarly, when the Maximum EPR V_{BUS} Voltage field is 50V, a Fixed Supply of 48V can be negotiated as part of an Explicit Contract where the absolute maximum Voltage that can be applied to the cable is 50.9V. Maximum V_{BUS} Voltage field values of 01b and 10b (formerly 30V and 40V) ***Shall*** be treated if they were 00b (20V).

6.4.4.3.1.6.9 V_{BUS} Current Handling Capability Field

The V_{BUS} Current Handling Capability field (B6…5) ***Shall*** indicate whether the cable is capable of carrying 3A or 5A.

6.4.4.3.1.6.10 USB Highest Speed Field

The USB Highest Speed field (B2…0) ***Shall*** indicate the highest signaling rate the cable supports. The DFP ***Shall*** consider all values indicated in this field that are higher than the highest value that the DFP recognizes as being ***Valid*** and functionally compatible with the highest speed that the DFP supports.

6.4.4.3.1.7 Active Cable VDOs

An *Active Cable* has a USB Plug on each end at least one of which is a Cable Plug supporting SOP' Communication. An *Active Cable* ***Shall*** incorporate data bus signal conditioning circuits and ***May*** have a concept of Super Speed Directionality on its Super Speed wires. An *Active Cable* ***May*** include a V_{BUS} wire.

An *Active Cable*:

- ***Shall*** respond to SOP' Communication.
- ***May*** respond to SOP'' Communication.
- ***Shall*** support the Structured VDM ***Discover Identity*** Command.
- In the ***Discover Identity*** Command ACK:

Shall set the Product Type in the ID Header VDO to *Active Cable*.

Shall return the *Active Cable* VDOs defined in ***Table 6.43 "Active Cable VDO 1"*** and ***Table 6.44 "Active Cable VDO 2"***.

Table 6.43 "Active Cable VDO 1"

Bit(s)	Field	Description
B31…28	HW Version	0000b…1111b assigned by the VID owner
B27…24	Firmware Version	0000b…1111b assigned by the VID owner
B23…21	VDO Version	Version Number of the VDO (not this specification Version): • Version 1.3 = 011b Values 000b, 100b…111b are Reserved and ***Shall Not*** be used
B20	***Reserved***	***Shall*** be set to zero.
B19…18	USB Type-C® plug to USB Type-C®/Captive	00b = ***Reserved***, ***Shall Not*** be used 01b = ***Reserved***, ***Shall Not*** be used 10b = USB Type-C® 11b = Captive
B17	EPR Mode Capable	0b – Cable is not EPR Mode Capable 1b = Cable is EPR Mode Capable
B16…13	Cable Latency	0000b – ***Reserved***, ***Shall Not*** be used. 0001b – <10ns (~1m) 0010b – 10ns to 20ns (~2m) 0011b – 20ns to 30ns (~3m) 0100b – 30ns to 40ns (~4m) 0101b – 40ns to 50ns (~5m) 0110b – 50ns to 60ns (~6m) 0111b – 60ns to 70ns (~7m) 1000b –1000ns (~100m) 1001b –2000ns (~200m) 1010b – 3000ns (~300m) 1011b ….1111b ***Reserved***, ***Shall Not*** be used. Includes latency of electronics in *Active Cable*.
B12…11	Cable Termination Type	00b…01b = ***Reserved***, ***Shall Not*** be used 10b = One end Active, one end passive, VCONN required 11b = Both ends Active, VCONN required
B10…9	Maximum V_{BUS} Voltage[2]	Maximum Cable V_{BUS} Voltage: 00b – 20V 01b – 30V[1] (***Deprecated***) 10b – 40V[1] (***Deprecated***) 11b – 50V
B8	SBU Supported	0 = SBUs connections supported 1 = SBU connections are not supported
B7	SBU Type	When SBU Supported = 1 this bit ***Shall*** be ***Ignored*** When SBU Supported = 0: 0 = SBU is passive 1 = SBU is active

B6…5	V_{BUS} Current Handling Capability	When V_{BUS} Through Cable is "No", this field ***Shall*** be ***Ignored***. When V_{BUS} Though Cable is "Yes": 00b = USB Type-C® Default Current 01b = 3A 10b = 5A 11b = ***Reserved***, ***Shall Not*** be used.
B4	V_{BUS} Through Cable	0 = No 1 = Yes
B3	SOP" Controller Present	0 = No SOP" controller present 1 = SOP" controller present
B2…0	USB Highest Speed	000b = ***[USB 2.0]*** only, no SuperSpeed support 001b = ***[USB 3.2]*** Gen1 010b = ***[USB 3.2]***/ ***[USB4]*** Gen2 011b = ***[USB4]*** Gen3 100b = ***[USB4]*** Gen4 101b…111b = ***Reserved***, ***Shall Not*** be used

1) Values no longer allowed. When present the field ***Shall*** be interpreted as if it was 00b.

2) EPR Sinks with a captive cable ***Shall*** report 50V.

Table 6.44 "Active Cable VDO 2"

Bit(s)	Field	Description
B31…24	Maximum Operating Temperature	The maximum internal operating temperature in °C. It might or might not reflect the plug's skin temperature.
B23…16	Shutdown Temperature	The temperature, in °C, at which the cable will go into thermal shutdown so as not to exceed the allowable plug skin temperature.
B15	Reserved	***Shall*** be set to zero.
B14…12	U3/CLd Power	000b: >10mW 001b: 5-10mW 010b: 1-5mW 011b: 0.5-1mW 100b: 0.2-0.5mW 101b: 50-200µW 110b: <50µW 111b: ***Reserved***, ***Shall Not*** be used
B11	U3 to U0 transition mode	0b: U3 to U0 direct 1b: U3 to U0 through U3S
B10	Physical connection	0b = Copper 1b = Optical
B9	Active element	0b = Active Redriver 1b = Active Retimer
B8	USB4® Supported	0b = ***[USB4]*** supported 1b = ***[USB4]*** not supported
B7…6	USB 2.0 Hub Hops Consumed	Number of ***[USB 2.0]*** 'hub hops' cable consumes. ***Shall*** be set to zero if USB 2.0 not supported.
B5	USB 2.0 Supported	0b = ***[USB 2.0]*** supported 1b = ***[USB 2.0]*** not supported
B4	USB 3.2 Supported	0b = ***[USB 3.2]*** SuperSpeed supported 1b = ***[USB 3.2]*** SuperSpeed not supported
B3	USB Lanes Supported	0b = One lane 1b = Two lanes
B2	Optically Isolated *Active Cable*	0b = No 1b = Yes
B1	USB4® Asymmetric Mode Supported	0b = No 1b = Yes ***Shall*** be set to zero if asymmetry is not supported.
B0	USB Gen	0b = Gen 1 1b = Gen 2 or higher **Note:** see VDO1 USB Highest Speed for details of Gen supported.

6.4.4.3.1.7.1 HW Version Field

The HW Version field (B31…28) contains a HW Version assigned by the VID owner.

6.4.4.3.1.7.2 FW Version Field

The FW Version field (B27…24) contains a FW Version assigned by the VID owner.

6.4.4.3.1.7.3 VDO Version Field

The VDO Version field (B23…20) contains a VDO version for this VDM version number. This field indicates the expected content for the *Active Cable* VDOs.

6.4.4.3.1.7.4 Connector Type Field

The USB Type-C® plug to USB Type-C®/Captive field (B19…18) ***Shall*** contain a value indicating whether the opposite end from the USB Type-C® plug is another USB Type-C® plug (i.e., a detachable Standard USB Type-C® Cable Assembly) or is a Captive Cable Assembly.

6.4.4.3.1.7.5 EPR Mode Capable

A static bit which ***Shall*** only be set when the cable is specifically designed for safe operation when carrying up to 48 volts at 5 amps.

6.4.4.3.1.7.6 Cable Latency Field

The Cable Latency field (B16…13) ***Shall*** contain a value corresponding to the signal latency through the cable which can be used as an approximation for its length.

6.4.4.3.1.7.7 Cable Termination Type Field

The Cable Termination Type field (B12…11) ***Shall*** contain a value corresponding to whether the Active Cable has one or two Cable Plugs requiring power from V_{CONN}.

6.4.4.3.1.7.8 Maximum V_{BUS} Voltage Field

The Maximum V_{BUS} Voltage field (B10…9) ***Shall*** contain the maximum Voltage that ***Shall*** be negotiated as part of an Explicit Contract where the maximum Voltage that ***Shall*** be applied to the cable is ***vSrcNew*** max + ***vSrcValid*** max. When this field is set to 20V, the cable will safely carry a Programmable Power Supply APDO of 20V where the absolute maximum Voltage that can be applied to the cable is 21.55V. Similarly, when the Maximum EPR V_{BUS} Voltage field is 50V, a Fixed Supply of 48V can be negotiated as part of an Explicit Contract where the absolute maximum Voltage that can be applied to the cable is 50.9V. Maximum V_{BUS} Voltage field values of 01b and 10b (formerly 30V and 40V) ***Shall*** be treated if they were 00b (20V).

6.4.4.3.1.7.9 SBU Supported Field

The SBU Supported field (B8) ***Shall*** indicate whether the cable supports the SBUs in the cable.

6.4.4.3.1.7.10 SBU Type Field

The SBU Type field (B7) ***Shall*** indicate whether the SBUs are passive or active (e.g., digital).

6.4.4.3.1.7.11 V_{BUS} Current Handling Capability Field

The V_{BUS} Current Handling Capability field (B6…5) ***Shall*** indicate whether the cable is capable of carrying default current (500mA [USB 2.0], 900mA [USB 3.2] x1, 1.5A [USB 3.2] x2), 3A or 5A. The V_{BUS} Current Handling Capability ***Shall*** only be ***Valid*** when the V_{BUS} Through Cable field indicates an end-to-end V_{BUS} wire.

6.4.4.3.1.7.12 V_{BUS} Through Cable Field

The V_{BUS} Through Cable field (B4) ***Shall*** indicate whether the cable contains an end-to-end V_{BUS} wire.

6.4.4.3.1.7.13 SOP'' Controller Present Field

The SOP'' Controller Present field (B3) ***Shall*** indicate whether one of the Cable Plugs is capable of SOP'' Communication in addition to the ***Normative*** SOP' Communication.

6.4.4.3.1.7.14 USB Highest Speed Field

The USB Highest Speed field (B2…0) ***Shall*** indicate the highest signaling rate the cable supports. The DFP ***Shall*** consider all values indicated in this field that are higher than the highest value that the DFP recognizes as being ***Valid*** and functionally compatible with the highest speed that the DFP supports.

6.4.4.3.1.7.15 Maximum Operating Temperature Field

Maximum Operating Temperature field (B31…24) ***Shall*** report the maximum allowable operating temperature inside the plug in °C.

6.4.4.3.1.7.16 Shutdown Temperature Field

Shutdown Temperature field (B23…16) ***Shall*** indicate the temperature inside the plug, in °C, at which the plug will shut down its active signaling components. When this temperature is reached, it will be reported in the Active Cable ***Status*** Message through the Thermal Shutdown bit.

6.4.4.3.1.7.17 U3/CLd Power Field

The U3/CLd Power field (B14…12) ***Shall*** indicate the power the cable consumes while in ***[USB 3.2]*** U3 or ***[USB4]*** CLd.

6.4.4.3.1.7.18 U3 to U0 Transition Mode Field

The U3 to U0 transition mode field (B11) ***Shall*** indicate which U3 to U0 mode the cable supports. This does not include the power in U3S if supported.

6.4.4.3.1.7.19 Physical Connection Field

The Physical Connection field (B10) ***Shall*** indicate the cable's construction, whether the connection between the active elements is copper or optical.

6.4.4.3.1.7.20 Active element Field

The Active Element field (B9) ***Shall*** indicate the cable's active element, whether the active element is a retimer or a redriver.

6.4.4.3.1.7.21 USB4® Supported Field

The USB4® Supported field (B8) ***Shall*** indicate whether or not the cable supports ***[USB4]*** operation.

6.4.4.3.1.7.22 USB 2.0 Hub Hops Consumed field

The USB 2.0 Hub Hops Consumed field (B7…6) ***Shall*** indicate the number of USB 2.0 'hub hops' that are lost due to the transmission time of the cable.

6.4.4.3.1.7.23 USB 2.0 Supported Field

The USB 2.0 Supported field (B5) ***Shall*** indicate whether or not the cable supports ***[USB 2.0]*** only signaling.

6.4.4.3.1.7.24 USB 3.2 Supported Field

The USB3.2 Supported field (B4) ***Shall***, indicate whether or not the cable supports ***[USB 3.2]*** SuperSpeed signaling.

6.4.4.3.1.7.25 USB Lanes Supported Field

The USB Lanes Supported field (B3) ***Shall*** indicate whether the cable supports one or two lanes of ***[USB 3.2]*** SuperSpeed signaling.

6.4.4.3.1.7.26 Optically Isolated Active Cable Field

The Optically Isolated *Active Cable* field (B2) ***Shall*** indicate whether this cable is an optically isolated *Active Cable* or not (as defined in ***[USB Type-C 2.3]***). Optically Isolated *Active Cables* ***Shall*** have a retimer or linear redriver (LRD) as the active element and do not support ***[USB 2.0]*** or carry V_{BUS}.

6.4.4.3.1.7.27 USB4® Asymmetric Mode Supported Field

The USB4® Asymmetric Mode Supported field (B1) ***Shall*** indicate that the *Active Cable* supports asymmetric mode as defined in ***[USB4]*** and ***[USB Type-C 2.3]***.

6.4.4.3.1.7.28 USB Gen Field

The USB Gen field (B0) ***Shall*** indicate the signaling Gen the cable supports. Gen 1 ***Shall*** only be used by ***[USB 3.2]*** cables as indicated by the USB 3.2 Supported field. Gen 2 or higher ***May*** be used by either ***[USB 3.2]*** or ***[USB4]*** cables as indicated by their respective supported fields. When Gen 2 or higher is indicated the USB Highest Speed field in VDO1 ***Shall*** indicate the actual Gen supported.

6.4.4.3.1.8 Alternate Mode Adapter VDO

The Alternate Mode Adapter (AMA) VDO has been ***Deprecated***. PD USB Devices which support one or more Alternate Modes ***Shall*** set an appropriate Product Type (UFP), and ***Shall*** set the Modal Operation Supported bit to '1'.

6.4.4.3.1.9 V_{CONN} Powered USB Device VDO

The V_{CONN} Powered USB Device (VPD) VDO defined in this section ***Shall*** be sent when the Product Type is given as V_{CONN} Powered USB Device. *Table 6.45 "VPD VDO"* defines the VPD VDO which ***Shall*** be sent.

Table 6.45 "VPD VDO"

Bit(s)	Field	Description
B31…28	HW Version	0000b…1111b assigned by the VID owner
B27…24	Firmware Version	0000b…1111b assigned by the VID owner
B23…21	VDO Version	Version Number of the VDO (not this specification Version): Version 1.0 = 000b Values 001b…111b are ***Reserved*** and ***Shall Not*** be used
B20…17	Reserved	***Shall*** be set to zero.
B16…15	Maximum V_{BUS} Voltage	Maximum VPD VBUS Voltage: 00b – 20V 01b – 30V[1] (***Deprecated***) 10b – 40V[1] (***Deprecated***) 11b – 50V[1] (***Deprecated***)
B14	Charge Through Current Support	Charge Through Support bit=1b: 0b - 3A capable. 1b - 5A capable Charge Through Support bit = 0b: ***Reserved***, ***Shall*** be set to zero
B13	Reserved	***Shall*** be set to zero.
B12…7	V_{BUS} Impedance	Charge Through Support bit = 1b: V_{BUS} impedance through the VPD in 2 mΩ increments. Values less than 10 mΩ are ***Reserved*** and ***Shall Not*** be used. Charge Through Support bit = 0b: ***Reserved, Shall*** be set to zero
B6…1	Ground Impedance	Charge Through Support bit = 1b: Ground impedance through the VPD in 1 mΩ increments. Values less than 10 mΩ are ***Reserved*** and ***Shall Not*** be used. Charge Through Support bit = 0b: ***Reserved, Shall*** be set to zero
B0	Charge Through Support	1b – the VPD supports Charge Through 0b – the VPD does not support Charge Through

1) Values no longer allowed. When present the field ***Shall*** be interpreted as if it was 00b.

6.4.4.3.1.9.1 HW Version Field

The HW Version field (B31…28) contains a HW Version assigned by the VID owner.

6.4.4.3.1.9.2 FW Version Field

The FW Version field (B27…24) contains a FW Version assigned by the VID owner.

6.4.4.3.1.9.3 VDO Version Field

The VDO Version field (B23…20) contains a VDO version for this VDM version number. This field indicates the expected content for this VDO.

6.4.4.3.1.9.4 Maximum V_{BUS} Voltage Field

The Maximum V_{BUS} Voltage field (B16…15) ***Shall*** contain the maximum Voltage that a Sink ***Shall*** negotiate through the VPD Charge Through port as part of an Explicit Contract.

Note: the maximum Voltage that will be applied to the cable is ***vSrcNew*** max + ***vSrcValid*** max. For example, when the Maximum V_{BUS} Voltage field is 20V, a Fixed Supply of 20V can be negotiated as part of an Explicit Contract where the absolute maximum Voltage that can be applied to the cable is 21.55V. Maximum V_{BUS} Voltage field values of 01b and 10b (formerly 30V and 40V) ***Shall*** be treated if they were 00b (20V).

6.4.4.3.1.9.5 V_{BUS} Impedance Field

The V_{BUS} Impedance field (B12…7) ***Shall*** contain the impedance the VPD adds in series between the Source and the Sink. The Sink ***Shall*** take this value into account when requesting current so as to not to exceed the V_{BUS} IR drop limit of 0.5V between the Source and itself. If the Sink can tolerate a larger IR drop on V_{BUS} it ***May*** do so.

6.4.4.3.1.9.6 Ground Impedance Field

Ground Impedance field (B6…1) ***Shall*** contain the impedance the VPD adds in series between the Source and the Sink. The Sink ***Shall*** take this value into account when requesting current so as to not to exceed the Ground IR drop limit of 0.25V between the Source and itself.

6.4.4.3.1.9.7 Charge Through Field

The Charge Through field (B0) ***Shall*** be set to 1b when the VPD supports Charge Through and 0b otherwise.

6.4.4.3.2 Discover SVIDs

The ***Discover SVIDs*** Command is used by an Initiator to determine the SVIDs for which a Responder has Modes. The ***Discover SVIDs*** Command is used in conjunction with the ***Discover Modes*** Command in the Discovery Process to determine which Modes a device supports. The list of SVIDs is always terminated with one or two 0x0000 SVIDs.

The SVID in the ***Discover SVIDs*** Command ***Shall*** be set to the ***PD SID*** (see ***Table 6.32 "SVID Values"***) by both the Initiator and the Responder for this Command.

The ***Number of Data Objects*** field in the Message Header in the ***Discover SVIDs*** Command request ***Shall*** be set to 1 since the ***Discover SVIDs*** Command request ***Shall Not*** contain any VDOs.

The ***Discover SVIDs*** Command ACK sent back by the Responder ***Shall*** contain one or more SVIDs. The SVIDs are returned 2 per VDO (see ***Table 6.46 "Discover SVIDs Responder VDO"***). If there are an odd number of supported SVIDs, the ***Discover SVIDs*** Command is returned ending with a SVID value of 0x0000 in the last part of the last VDO. If there are an even number of supported SVIDs, the ***Discover SVIDs*** Command is returned ending with an additional VDO containing two SVIDs with values of 0x0000. A Responder ***Shall*** only return SVIDs for which a ***Discover Modes*** Command request for that SVID will return at least one Mode.

A Responder that does not support any SVIDs ***Shall*** return a NAK.

The ***Number of Data Objects*** field in the Message Header in the ***Discover SVIDs*** Command NAK and BUSY responses ***Shall*** be set to 1 since they ***Shall Not*** contain any VDOs.

If the Responder supports 12 or more SVIDs then the ***Discover SVIDs*** Command ***Shall*** be executed multiple times until a Discover SVIDs VDO is returned ending either with a SVID value of 0x0000 in the last part of the last VDO or with a VDO containing two SVIDs with values of 0x0000. Each Discover SVID ACK Message, other than the one containing the terminating 0x0000 SVID, ***Shall*** convey 12 SVIDs. The Responder ***Shall*** restart the list of SVIDs each time a ***Discover Identity*** Command request is received from the Initiator.

Note: since a Cable Plug does not retry Messages if the ***GoodCRC*** Message from the Initiator becomes corrupted the Cable Plug will consider the ***Discover SVIDs*** Command ACK unsent and will send the same list of SVIDs again.

Figure 6-17 "Example Discover SVIDs response with 3 SVIDs" shows an example response to the ***Discover SVIDs*** Command request with two VDOs containing three SVIDs. ***Figure 6-18 "Example Discover SVIDs response with 4 SVIDs"*** shows an example response with two VDOs containing four SVIDs followed by an empty VDO to terminate the response. ***Figure 6-19 "Example Discover SVIDs response with 12 SVIDs followed by an empty response"*** shows an example response with six VDOs containing twelve SVIDs followed by an additional request that returns an empty VDO indicating there are no more SVIDs to return.

Table 6.46 "Discover SVIDs Responder VDO"

Bit(s)	Field	Description
B31…16	SVID n	16-bit unsigned integer, assigned by the USB-IF or 0x0000 if this is the last VDO and the Responder supports an even number of SVIDs.
B15…0	SVID n+1	16-bit unsigned integer, assigned by the USB-IF or 0x0000 if this is the last VDO and the Responder supports an odd or even number of SVIDs.

Figure 6-17 "Example Discover SVIDs response with 3 SVIDs"

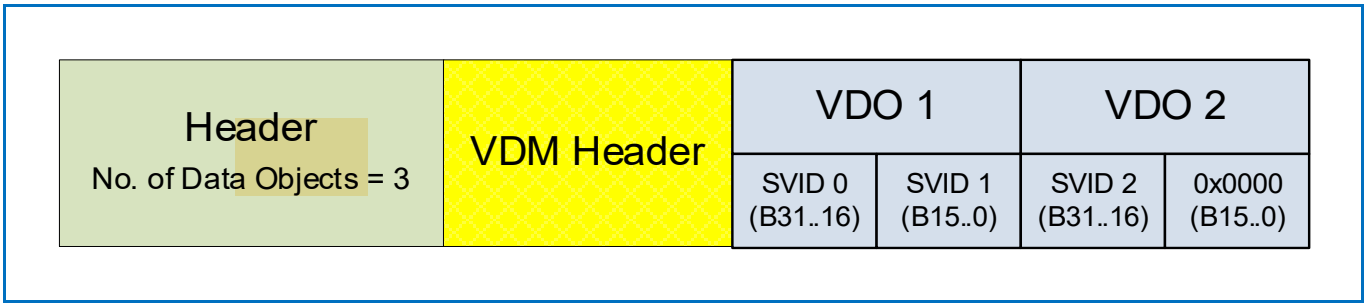


Figure 6-18 "Example Discover SVIDs response with 4 SVIDs"

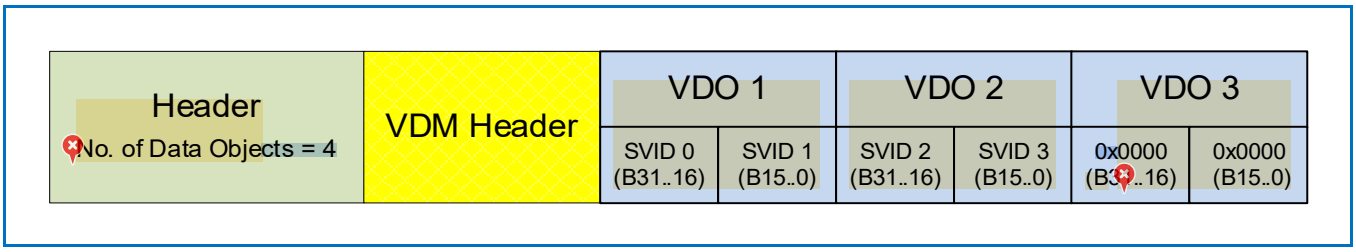
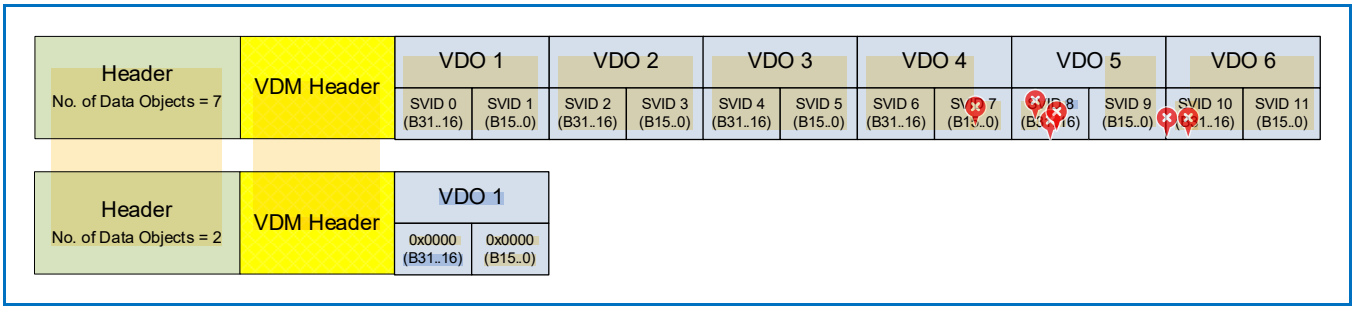


Figure 6-19 "Example Discover SVIDs response with 12 SVIDs followed by an empty response"



6.4.4.3.3 Discover Modes

The ***Discover Modes*** Command is used by an Initiator to determine the Modes a Responder supports for a given SVID.

The SVID in the ***Discover Modes*** Command ***Shall*** be set to the SVID for which Modes are being requested by both the Initiator and the Responder for this Command.

The ***Number of Data Objects*** field in the Message Header in the ***Discover Modes*** Command request ***Shall*** be set to 1 since the ***Discover Modes*** Command request ***Shall Not*** contain any VDOs.

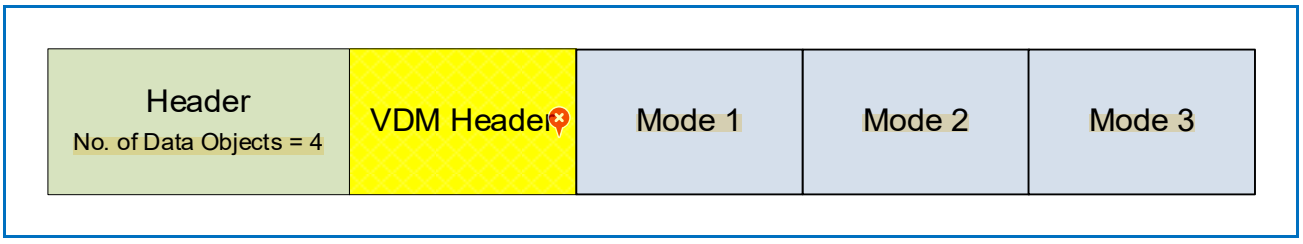
The ***Discover Modes*** Command ACK sent back by the Responder ***Shall*** contain one or more Modes. The ***Discover Modes*** Command ACK ***Shall*** contain a Message Header with the ***Number of Data Objects*** field set to a value of 2 to 7 (the actual value is the number of Mode objects plus one). If the ID is a VID, the structure and content of the VDO is left to the Vendor. If the ID is a SID, the structure and content of the VDO is defined by the relevant Standard.

A Responder that does not support any Modes ***Shall*** return a NAK.

The ***Number of Data Objects*** field in the Message Header in the ***Discover Modes*** Command NAK and BUSY responses ***Shall*** be set to 1 since they ***Shall Not*** contain any VDOs.

Figure 6-20 "Example Discover Modes response for a given SVID with 3 Modes" shows an example of a ***Discover Modes*** Command response from a Responder which supports three Modes for a given SVID.

Figure 6-20 "Example Discover Modes response for a given SVID with 3 Modes"



6.4.4.3.4 Enter Mode Command

The ***Enter Mode*** Command is used by an Initiator (DFP) to command a Responder (UFP or Cable Plug) to enter a specified Mode of operation. Only a DFP ***Shall*** initiate the Enter Mode Process which it starts after it has successfully completed the Discovery Process.

The value in the Object Position field in the VDM Header ***Shall*** indicate to which Mode in the ***Discover Modes*** Command the VDO refers (see ***Figure 6-20 "Example Discover Modes response for a given SVID with 3 Modes"***). The value 1 always indicates the first Mode as it is the first object following the VDM Header. The value 2 refers to the next Mode and so forth.

The ***Number of Data Objects*** field in the Message Header in the Command request ***Shall*** be set to either 1 or 2 since the ***Enter Mode*** Command request ***Shall Not*** contain more than 1 VDO. When a VDO is included in an ***Enter Mode*** Command request the contents of the 32-bit VDO is defined by the Mode.

The ***Number of Data Objects*** field in the Command response ***Shall*** be set to 1 since an ***Enter Mode*** Command response (ACK, NAK) ***Shall Not*** contain any VDOs.

Before entering a Mode, by sending the ***Enter Mode*** Command request that requires the reconfiguring of any pins on entry to that Mode, the Initiator ***Shall*** ensure that those pins being reconfigured are placed into the USB Safe State. Before entering a Mode that requires the reconfiguring of any pins, the Responder ***Shall*** ensure that those pins being reconfigured are placed into either USB operation or the USB Safe State.

A device ***May*** support multiple Modes with one or more active at any point in time. Any interactions between them are the responsibility of the Standard or Vendor. Where there are multiple *Active Mode*s at the same time Modal Operation ***Shall*** start on entry to the first Mode.

On receiving an ***Enter Mode*** Command requests the Responder ***Shall*** respond with either an ACK or a NAK response. The Responder is not allowed to return a BUSY response. The value in the Object Position field of the ***Enter Mode*** Command response ***Shall*** contain the same value as the received ***Enter Mode*** Command request.

If the Responder responds to the ***Enter Mode*** Command request with an ACK, the Responder ***Shall*** enter the Mode before sending the ACK. The Initiator ***Shall*** enter the Mode on reception of the ACK. Successful transmission of the message confirms to the Responder that the Initiator will enter an *Active Mode*.

See ***Figure 8-113 "DFP to UFP Enter Mode"*** for more details.

If the Responder responds to the ***Enter Mode*** Command request with a NAK, the Mode is not entered. If not presently in Modal Operation the Initiator ***Shall*** return to USB operation. If not presently in Modal Operation the Responder ***Shall*** remain in either USB operation or the USB Safe State.

If the Initiator fails to receive a response within ***tVDMWaitModeEntry*** it ***Shall Not*** enter the Mode but return to USB operation.

Figure 6-21 "Successful Enter Mode sequence" shows the sequence of events during the transition between USB operation and entering a Mode. It illustrates when the Responder's Mode changes and when the Initiator's Mode changes. ***Figure 6-22 "Unsuccessful Enter Mode sequence due to NAK"*** illustrates that when the Responder returns a NAK the transition to a Mode do not take place and the Responder and Initiator remain in their default USB roles.

Figure 6-21 “Successful Enter Mode sequence”

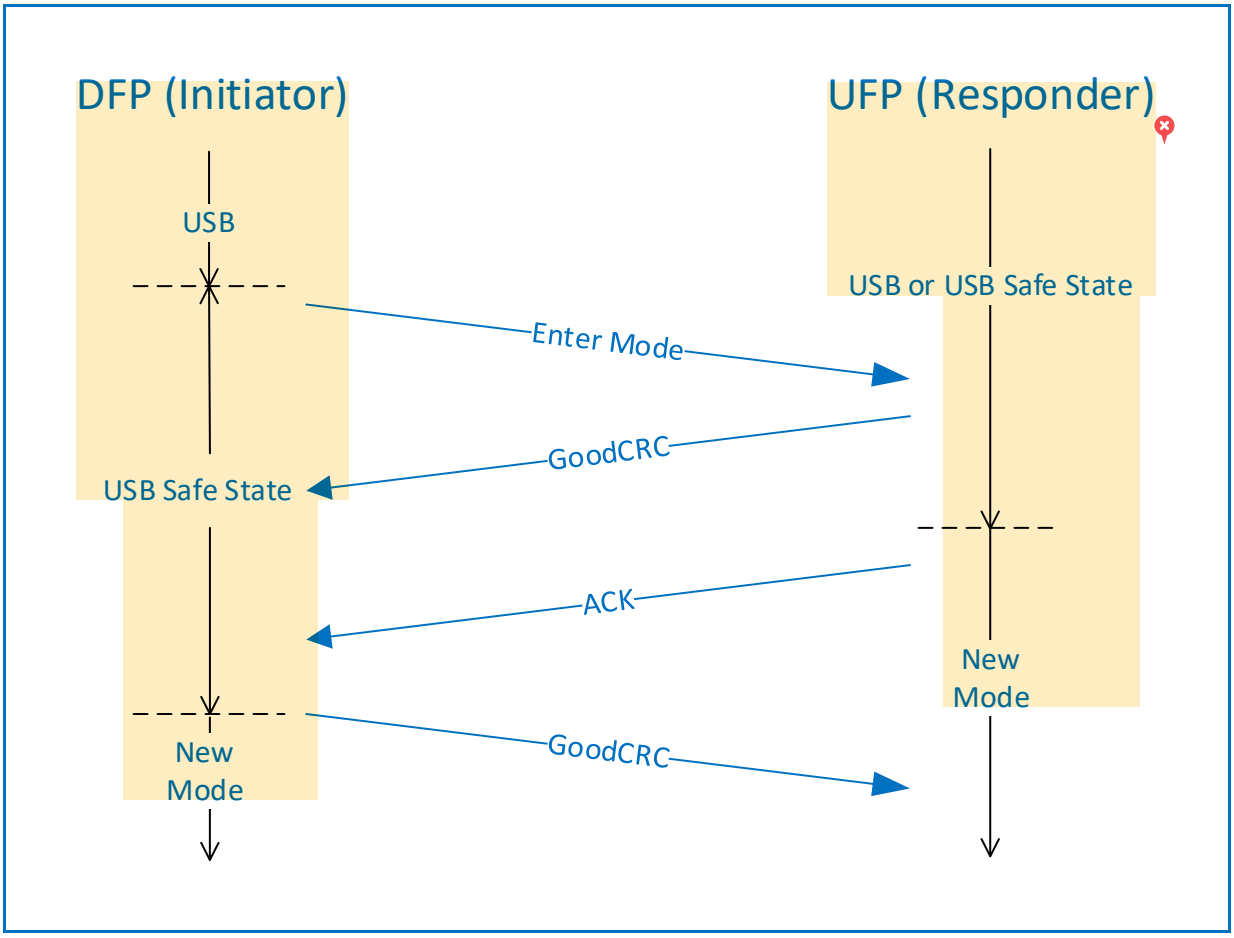
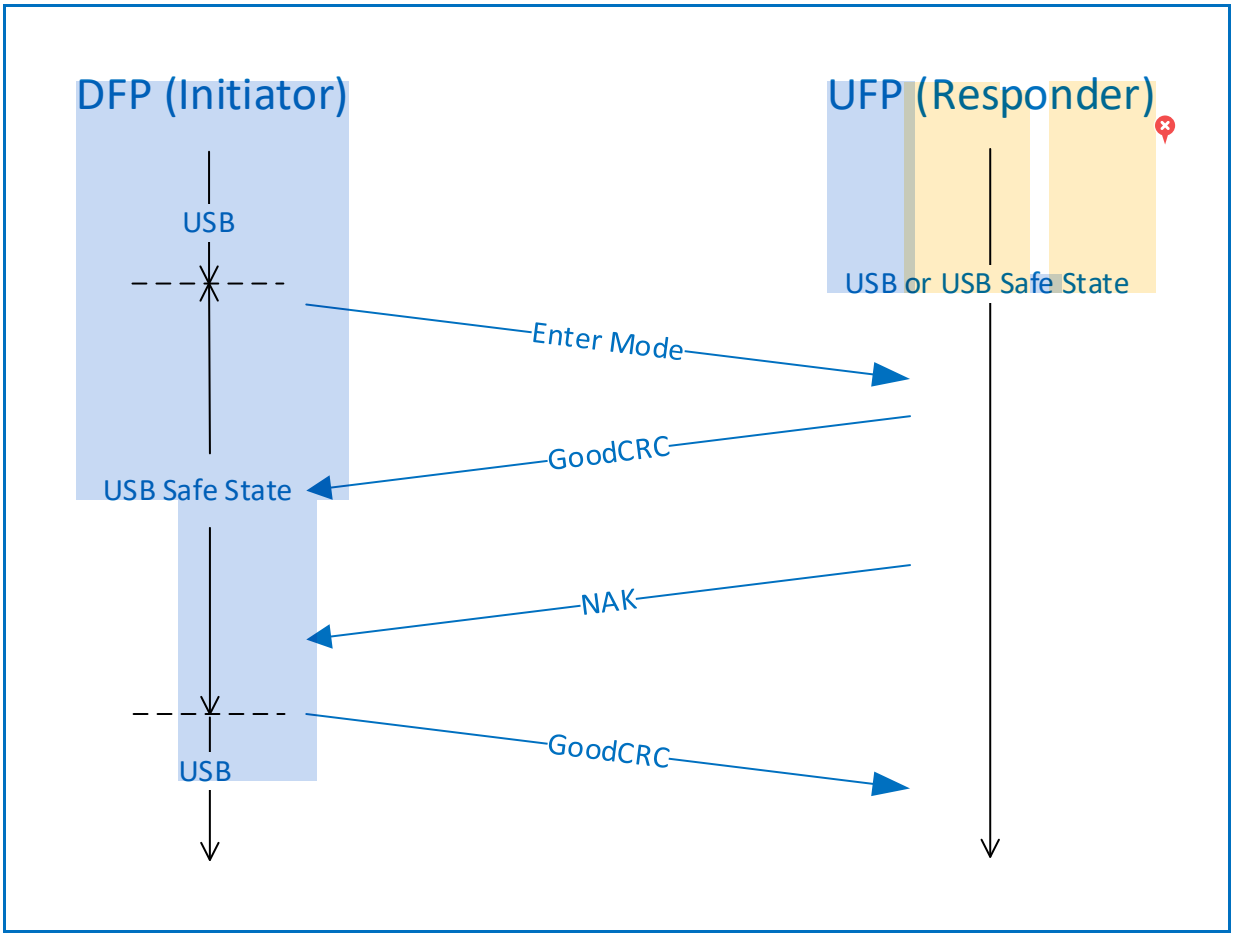


Figure 6-22 "Unsuccessful Enter Mode sequence due to NAK"



Once the Mode is entered, the device ***Shall*** remain in that *Active Mode* until the ***Exit Mode*** Command is successful (see ***Section 6.4.4.3.5 "Exit Mode Command"***).

The following events ***Shall*** also cause the Port Partners and Cable Plug(s) to exit all *Active Modes*:

- A PD Hard Reset.
- The Port Partners or Cable Plug(s) are Detached.
- A Cable Reset (only exits the Cable Plug's Active Modes).
- A Data Reset (removing power briefly resets all the Active Modes in the Cable Plug).

The Initiator ***Shall*** return to USB Operation within ***tVDMExitMode*** of a disconnect or of ***Hard Reset*** Signaling being detected.

The Responder ***Shall*** return to either USB operation or USB Safe State within ***tVDMExitMode*** of a disconnect or of ***Hard Reset*** Signaling being detected.

A ***DR_Swap*** Message ***Shall Not*** be sent during Modal Operation between the Port Partners (see ***Section 6.3.9 "DR_Swap Message"***).

6.4.4.3.5 Exit Mode Command

The ***Exit Mode*** Command is used by an Initiator (DFP) to command a Responder (UFP or Cable Plug) to exit its *Active Mode* and return to normal USB operation. Only the DFP ***Shall*** initiate the Exit Mode Process.

The value in the Object Position field ***Shall*** indicate to which Mode in the ***Discover Modes*** Command the VDO refers (see ***Figure 6-20 "Example Discover Modes response for a given SVID with 3 Modes"***) and ***Shall*** have been used previously in an ***Enter Mode*** Command request for an *Active Mode*. The value 1 always indicates the first Mode as it is the first object following the VDM Header. The value 2 refers to the next Mode and so forth. A value of 111b in the Object Position field ***Shall*** indicate that all *Active Modes* ***Shall*** be exited.

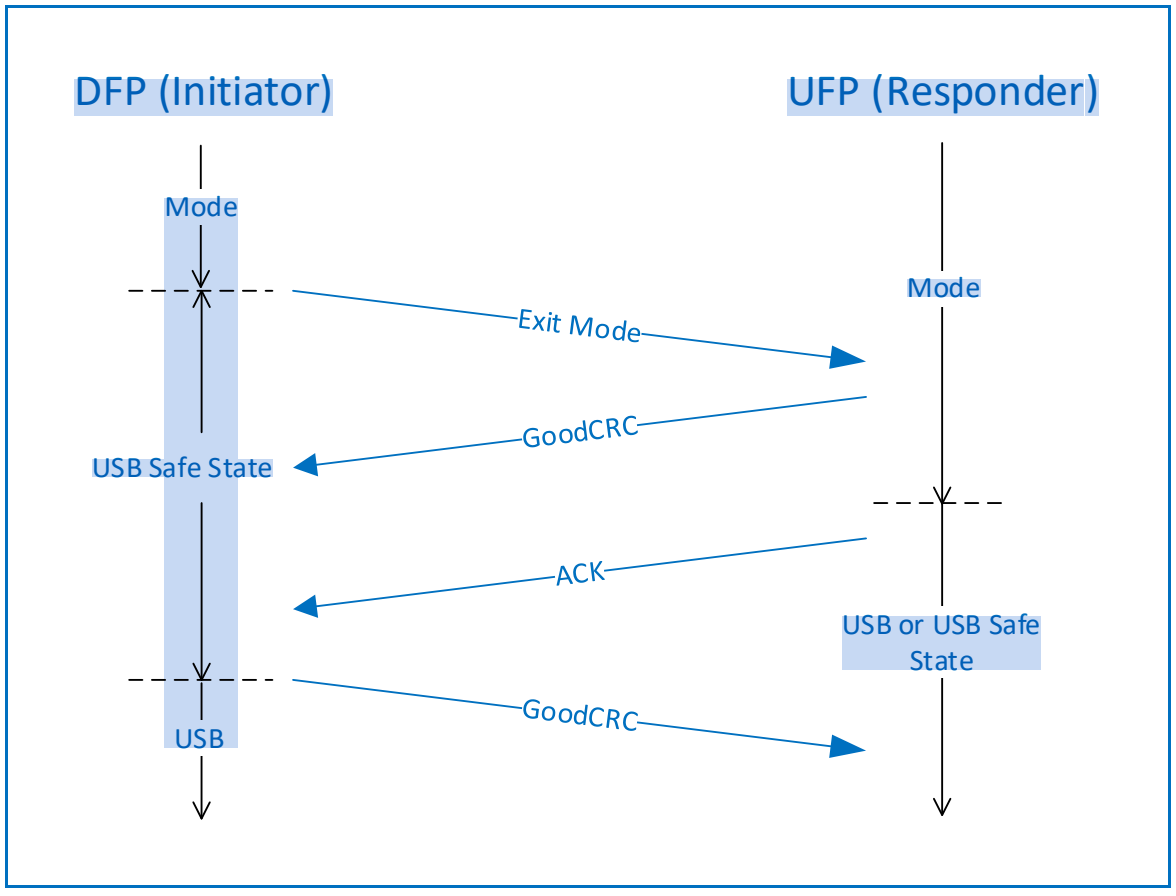
The ***Number of Data Objects*** field in both the Command request and Command response (ACK, NAK) ***Shall*** be set to 1 since an ***Exit Mode*** Command ***Shall Not*** contain any VDOs.

The Responder ***Shall*** exit its *Active Mode* before sending the response Message. The Initiator ***Shall*** exit its *Active Mode* when it receives the ACK. The Responder ***Shall Not*** return a BUSY acknowledgement and ***Shall*** only return a NAK acknowledgement to a request not containing an *Active Mode* (i.e., ***Invalid*** object position). An Initiator which fails to receive an ACK within ***tVDMWaitModeExit*** or receives a NAK or BUSY response ***Shall*** exit its *Active Mode*.

See ***Figure 8-114 "DFP to UFP Exit Mode"*** for more details.

Figure 6-23 "Exit Mode sequence" shows the sequence of events during the transition between exiting an *Active Mode* and USB operation. It illustrates when the Responder's Mode changes and when the Initiator's Mode changes.

Figure 6-23 "Exit Mode sequence"



6.4.4.3.6 Attention

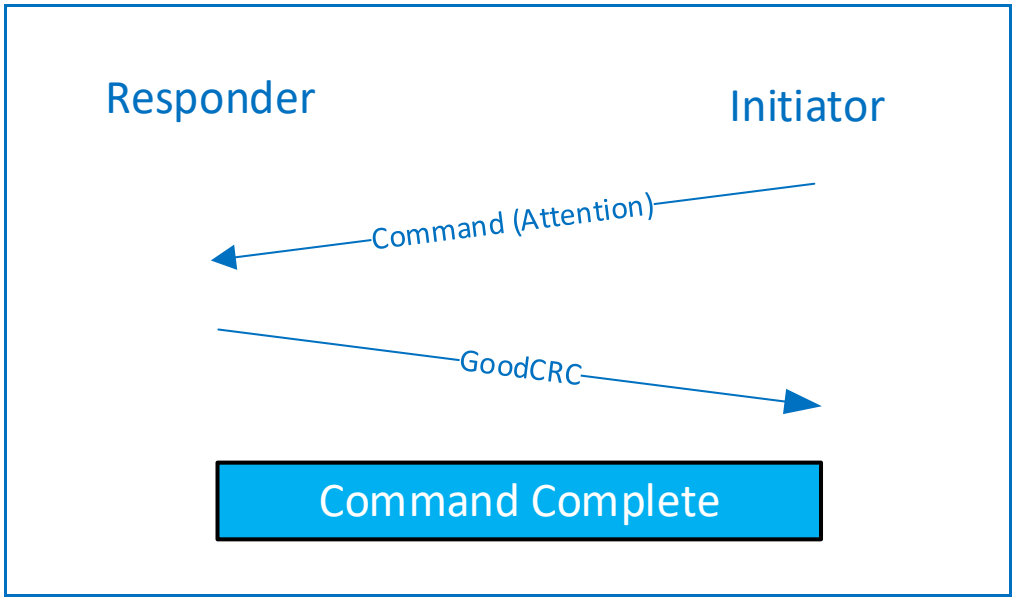
The ***Attention*** Command ***May*** be used by the Initiator to notify the Responder that it requires service.

The value in the Object Position field ***Shall*** indicate to which Mode in the ***Discover Modes*** Command the VDO refers (see ***Figure 6-20 "Example Discover Modes response for a given SVID with 3 Modes"***) and ***Shall*** have been used previously in an ***Enter Mode*** Command request for an *Active Mode*. The value 1 always indicates the first Mode as it is the first object following the VDM Header. The value 2 refers to the next Mode and so forth. A value of 000b or 111b in the Object Position field ***Shall Not*** be used by the ***Attention*** Command.

The ***Number of Data Objects*** field in the Message Header ***Shall*** be set to 1 or 2 since the ***Attention*** Command ***Shall Not*** contain more than 1 VDO. When a VDO is included in an ***Attention*** Command the contents of the 32-bit VDO is defined by the Mode.

Figure 6-23 "Exit Mode sequence" shows the sequence of events when an ***Attention*** Command is received.

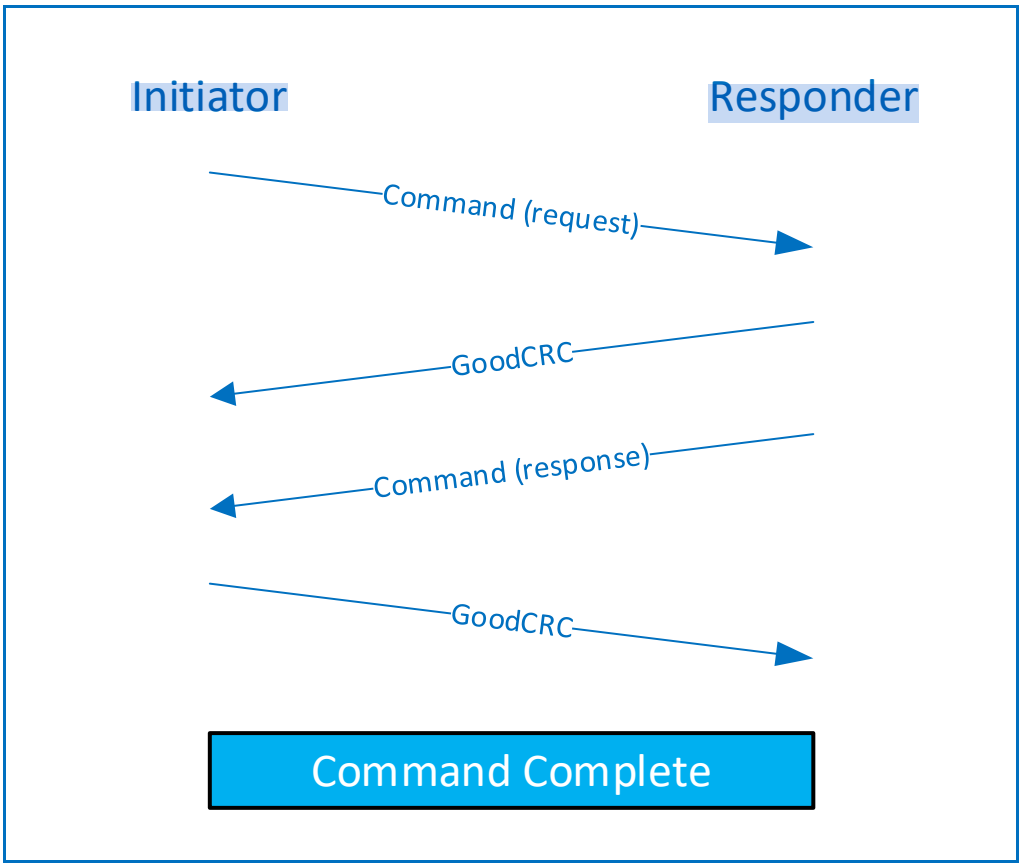
Figure 6-24 "Attention Command request/response sequence"



6.4.4.4 Command Processes

The Message flow of Commands during a Process is a query followed by a response. Every Command request sent has to be responded to with a ***GoodCRC*** Message. The ***GoodCRC*** Message only indicates the Command request was received correctly; it does not mean that the Responder understood or even supports a particular SVID. ***Figure 6-25 "Command request/response sequence"*** shows the request/response sequence including the ***GoodCRC*** Messages.

Figure 6-25 "Command request/response sequence"



In order for the Initiator to know that the Command request was actually consumed, it needs an acknowledgement from the Responder. There are three responses that indicate the Responder received and processed the Command request:

- ACK.
- NAK.
- BUSY.

The Responder ***Shall*** complete:

- Enter Mode requests within ***tVDMEnterMode.***
- Exit Mode requests within ***tVDMExitMode.***
- Other requests within ***tVDMReceiverResponse.***

An Initiator not receiving a response within the following times ***Shall*** timeout and return to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state (as appropriate):

- Enter Mode requests within ***tVDMWaitModeEntry.***
- Exit Mode requests within ***tVDMWaitModeExit.***
- Other requests within ***tVDMSenderResponse.***

The Responder ***Shall*** respond with:

- ACK if it recognizes the SVID and can process it at this time.
- NAK:
  - if it recognizes the SVID but cannot process the Command request
  - or if it does not recognize the SVID
  - or if it does not support the Command
  - or if a VDO contains a field which is ***Invalid.***
- BUSY if it recognizes the SVID and the Command but cannot process the Command request at this time.

The ACK, NAK or BUSY response ***Shall*** contain the same SVID as the Command request.

6.4.4.4.1 Discovery Process

The Initiator (usually the DFP) always begins the Discovery Process. The Discovery Process has two phases. In the first phase, the ***Discover SVIDs*** Command request is sent by the Initiator to get the list of SVIDs the Responder supports. In the second phase, the Initiator sends a ***Discover Modes*** Command request for each SVID supported by both the Initiator and Responder.

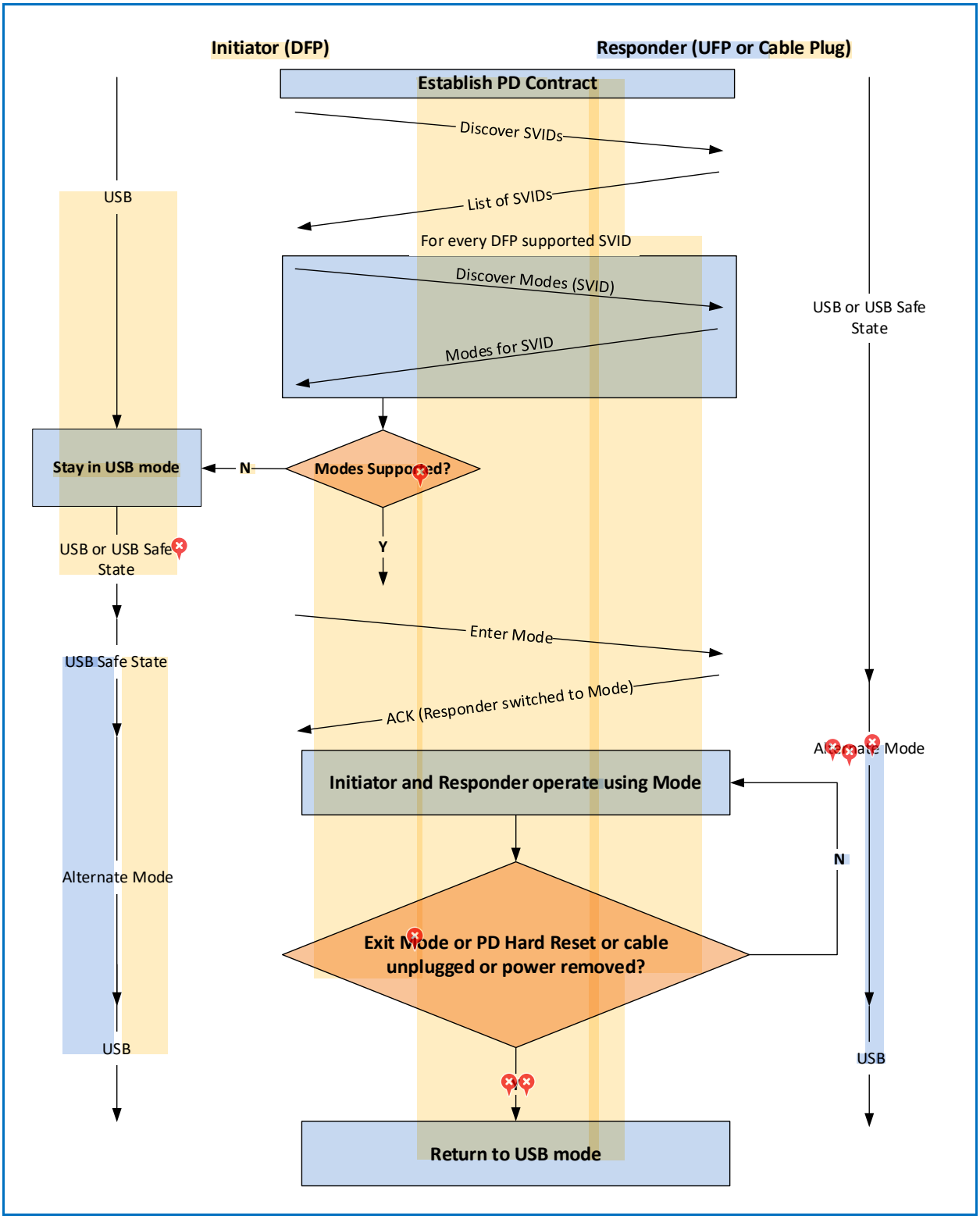
6.4.4.4.2 Enter Vendor Mode / Exit Vendor Mode Processes

The result of the Discovery Process is that both the Initiator and Responder identify the Modes they mutually support. The Initiator (DFP), upon finding a suitable Mode, uses the ***Enter Mode*** Command to enable the Mode.

The Responder (UFP or Cable Plug) and Initiator continue using the *Active Mode* until the *Active Mode* is exited. In a managed termination, using the ***Exit Mode*** Command, the *Active Mode* ***Shall*** be exited in a controlled manner as described in ***Section 6.4.4.3.5 "Exit Mode Command".*** In an unmanaged termination, triggered by a Power Delivery Hard Reset (i.e. ***Hard Reset*** Signaling sent by either Port Partner) or by cable Detach (device unplugged), the *Active Mode* ***Shall*** still be exited but there ***Shall Not*** be a transition through the USB Safe State. In both the managed and unmanaged terminations, the Initiator and Responder return to USB operation as defined in ***[USB Type-C 2.3]*** following an exit from a Mode.

The overall Message flow is illustrated in ***Figure 6-26 "Enter/Exit Mode Process"***.

Figure 6-26 "Enter/Exit Mode Process"



6.4.4.5 VDM Message Timing and Normal PD Messages

The timing and interspersing of VDMs between regular PD Messages ***Shall*** be done without perturbing the PD Message sequences. This requirement ***Shall*** apply to both Unstructured VDMs and Structured VDMs.

The use of Structured VDMs by an Initiator ***Shall Not*** interfere with the normal PD Message timing requirements nor ***Shall*** either the Initiator or Responder interrupt a PD Message sequence (e.g., Power Negotiation, Power Role Swap, Data Role Swap etc.). The use of Unstructured VDMs ***Shall Not*** interfere with normal PD Message timing.

6.4.5 Battery_Status Message

The ***Battery_Status*** Message ***Shall*** be sent in response to a ***Get_Battery_Status*** Message. The ***Battery_Status*** Message contains one Battery Status Data Object (BSDO) for one of the Batteries it supports as reported by Battery field in the ***Source_Capabilities_Extended*** Message. The returned BSDO ***Shall*** correspond to the Battery requested in the ***Battery Status Ref*** field contained in the ***Get_Battery_Status*** Message.

The ***Battery_Status*** Message returns a BSDO whose format ***Shall*** be as shown in ***Figure 6-27 "Battery_Status Message"*** and ***Table 6.47 "Battery Status Data Object (BSDO)"***. The ***Number of Data Objects*** field in the ***Battery_Status*** Message ***Shall*** be set to 1.

Figure 6-27 "Battery_Status Message"

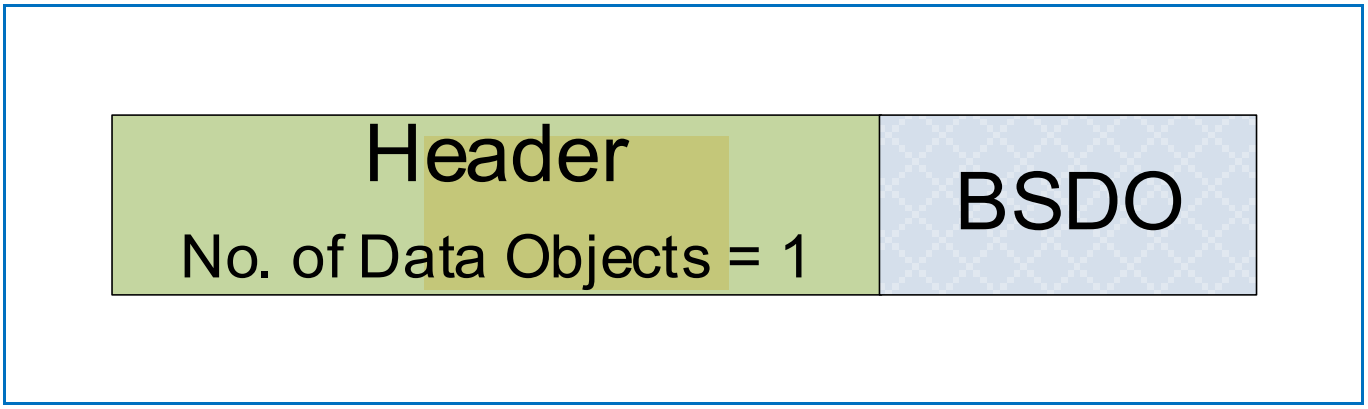


Table 6.47 "Battery Status Data Object (BSDO)"

Bit(s)	Field	Description
B31…16	Battery Present Capacity	Battery's State of Charge (SoC) in 0.1 WH increments **Note:** 0xFFFF = Battery's SOC unknown
B15…8	Battery Info	<table><tr><th>Bit</th><th>Description</th></tr><tr><td>0</td><td>***Invalid*** Battery reference</td></tr><tr><td>1</td><td>Battery is present when set</td></tr><tr><td>3…2</td><td>When Battery is present ***Shall*** contain the Battery charging status: 00b: Battery is Charging. 01b: Battery is Discharging. 10b: Battery is Idle. 11b: ***Reserved, Shall Not*** be used. When Battery is not present: 11b…00b: ***Reserved***, ***Shall Not*** be used</td></tr><tr><td>7…4</td><td>***Reserved*** and ***Shall*** be set to zero</td></tr></table>
B7…0	Reserved	***Shall*** be set to zero

6.4.5.1 Battery Present Capacity

The Battery Present Capacity field ***Shall*** return either the Battery's State of Charge (SoC) in tenths of WH or indicate that the Battery's present State of Charge (SOC) is unknown.

6.4.5.2 Battery Info

The Battery Info field ***Shall*** be used to report additional information about the Battery's present status. The Battery Info field's bits ***Shall*** reflect the present conditions under which the Battery is operating in the systems.

6.4.5.2.1 Invalid Battery Reference

The ***Invalid*** Battery Reference bit ***Shall*** be set when the ***Get_Battery_Status*** Message contains a reference to a Battery or Battery Slot (see ***Section 6.5.1.13 "Number of Batteries/Battery Slots Field"***) that does not exist.

6.4.5.2.2 Battery is Present

The Battery is Present bit ***Shall*** be set whenever the Battery is present. It ***Shall*** always be set for Batteries that are not Hot Swappable Batteries. For Hot Swappable Batteries, Battery is Present bit ***Shall*** indicate whether the Battery is Attached or Detached.

6.4.5.2.3 Battery Charging Status

The Battery charging status bits indicate whether the Battery is being charged, discharged or is idle (neither charging nor discharging). These bits ***Shall*** be set when the Battery is present bit is set. Otherwise, when the Battery is present bit is zero the Battery charging status bits ***Shall*** also be zero.

6.4.6 Alert Message

The ***Alert*** Message is provided to allow Port Partners to inform each other when there is a status change event. Some of the events are critical such as OCP, OVP and OTP, while others are ***Informative*** such as change in a Battery's status from charging to neither charging nor discharging.

The ***Alert*** Message ***Shall*** only be sent when the Source or Sink detects a status change.

The ***Alert*** Message ***Shall*** contain exactly one Alert Data Object (ADO) and the format ***Shall*** be as shown in ***Figure 6-28 "Alert Message"*** and ***Table 6.48 "Alert Data Object*** (ADO)".

Figure 6-28 "Alert Message"

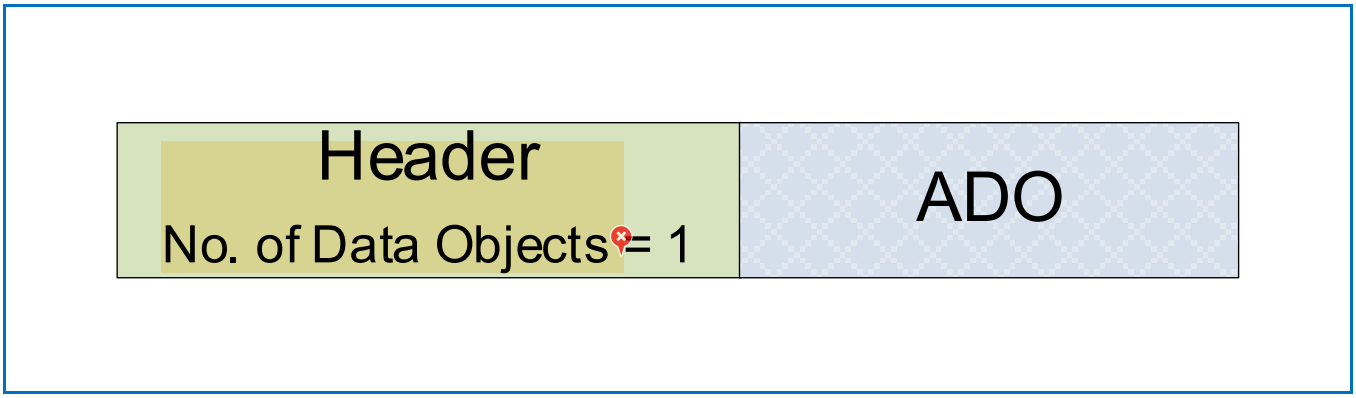


Table 6.48 "Alert Data Object (ADO)"

<table>
<tr><th>Bit(s)</th><th>Field</th><th>Description</th></tr>
<tr><td>B31...24</td><td>Type of Alert</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>Reserved and Shall be set to zero</td></tr>
<tr><td>1</td><td>Battery Status Change Event (Attach/Detach/charging/discharging/idle)</td></tr>
<tr><td>2</td><td>OCP event when set (Source only, for Sink Reserved and Shall be set to zero)</td></tr>
<tr><td>3</td><td>OTP event when set</td></tr>
<tr><td>4</td><td>Operating Condition Change when set</td></tr>
<tr><td>5</td><td>Source Input Change Event when set</td></tr>
<tr><td>6</td><td>OVP event when set</td></tr>
<tr><td>7</td><td>Extended Alert Event</td></tr>
</table>
</td></tr>
<tr><td>B23...20</td><td>Fixed Batteries</td><td>When Battery Status Change bit set indicates which Fixed Batteries have had a status change. B20 corresponds to Battery 0 and B23 corresponds to Battery 3.</td></tr>
<tr><td>B19...16</td><td>Hot Swappable Batteries</td><td>When Battery Status Change bit set indicates which Hot Swappable Batteries have had a status change. B16 corresponds to Battery 4 and B19 corresponds to Battery 7.</td></tr>
<tr><td>B15...4</td><td>Reserved</td><td>Shall be set to zero</td></tr>
<tr><td>B3...0</td><td>Extended Alert Event Type</td><td>When the Extended Alert Event bit in the Type of Alert field equals '1', then the Extended Alert Event Type field indicates the event which has occurred:
0 = Reserved.
1 = Power state change (DFP only)
2 = Power button press (UFP only)
3 = Power button release (UFP only)
4 = Controller initiated wake e.g., Wake on Lan (UFP only)
5-15 = Reserved
When the Extended Alert Event bit in the Type of Alert field equals '0', then the Extended Alert Event Type field is Reserved and Shall be set to zero.</td></tr>
</table>

6.4.6.1 Type of Alert

The *Type of Alert* field ***Shall*** be used to report Source or Sink status changes. Only one *Alert* Message ***Shall*** be generated for each Event or Change; however multiple Type of Alert bits ***May*** be set in one *Alert* Message. Once the *Alert* Message has been sent the *Type of Alert* field ***Shall*** be cleared.

A *Get_Battery_Status* Message ***Should*** be sent in response to a Battery status change in an *Alert* Message to get the details of the change.

A *Get_Status* Message ***Should*** be sent in response to a non-Battery status change in an *Alert* Message from to get the details of the change.

6.4.6.1.1 Battery Status Change

The Battery Status Change bit ***Shall*** be set when any Battery's power state changes between charging, discharging, neither. For Hot Swappable Batteries, it ***Shall*** also be set when a Battery is Attached or Detached.

6.4.6.1.2 Over-Current Protection Event

The Over-Current Protection Event bit ***Shall*** be set when a Source detects its output current exceeds its limits triggering its protection circuitry. This bit is ***Reserved*** for a Sink.

6.4.6.1.3 Over-Temperature Protection Event

The Over-Temperature Protection Event bit ***Shall*** be set when a Source or Sink shuts down due to over-temperature triggering its protection circuitry.

6.4.6.1.4 Operating Condition Change

The Operating Condition Change bit ***Shall*** be set when a Source or Sink detects its Operating Condition enters or exits either the 'warning' or 'over temperature' temperature states.

The Operating Condition Change bit ***Shall*** be set when the Source operating in the Programmable Power Supply mode detects it has changed its operating condition between Constant Voltage (CV) and Current Limit (CL).

6.4.6.1.5 Source Input Change Event

The Source Input Event bit ***Shall*** be set when the Source/Sink's input changes. For example, when the AC input is removed, and the Source/Sink continues to be powered from one or more of its batteries or when AC returns and the Source/Sink transitions from Battery to AC operation or when the Source/Sink changes operation from one (or more) Battery to another (or more) Battery.

6.4.6.1.6 Over-Voltage Protection Event

The Over-Voltage Protection Event bit ***Shall*** be set when the Sink detects its output Voltage exceeds its limits triggering its protection circuitry.

The Over-Voltage Protection Event bit ***May*** be set when the Source detects its output Voltage exceeds its limits triggering its protection circuitry.

6.4.6.1.7 Extended Alert Event

The Extended Alert Event bit ***Shall*** be set when the event is defined as an Extended Alert Type.

6.4.6.2 Fixed Batteries

The ***Fixed Batteries*** field indicates which Fixed Batteries have had a status change. B20 corresponds to Battery 0 and B23 corresponds to Battery 3.

Once the ***Alert*** Message has been sent the ***Fixed Batteries*** field Shall be cleared.

6.4.6.3 Hot Swappable Batteries

The ***Hot Swappable Batteries*** field indicates which Hot Swappable Batteries have had a status change. B16 corresponds to Battery 0 and B19 corresponds to Battery 3.

Once the ***Alert*** Message has been sent the ***Hot Swappable Batteries*** field Shall be cleared.

6.4.6.4 Extended Alert Event Types

The ***Extended Alert Event Type*** field provides extensions to the available types for the ***Alert*** Message. If the Extended Alert Event bit is not set, then the Extended Alert Event Type is ***Reserved*** and ***Shall*** be set to zero.

6.4.6.4.1 Power State Change

The Power state change event value ***May*** be set when the DFP transitions into a new power state. The new power state ***Shall*** be communicated via the Power state change byte in the ***Status*** Message. This message ***Should*** be sent by the host in response to any system power state change.

6.4.6.4.2 Power Button Press

The Power button press event value ***May*** be set when the power button on the UFP is pressed. The press and release events are separated into two different events so that devices that respond differently to a long button press will see a long button press. On the host-side, the power button press event typically initiates the same behavior as a power button press of the host's power button.

6.4.6.4.3 Power Button Release

If a Power button press event was sent, then the Power button release event value ***Shall*** be sent by the UFP following the Power button press event. If a physical power button press initiated the Power button press event, then the Power button release event ***Should*** be sent when the physical button is released.

6.4.6.4.4 Controller initiated wake

The Controller initiated wake is used to communicate a wake event from the UFP to the DPF such as Wake on Lan from a NIC or another controller. This event doesn't need the press/release form of the Power button press, because it only needs to communicate the presence of the event, and not the timing.

6.4.7 Get_Country_Info Message

The *Get_Country_Info* Message ***Shall*** be sent by a port to get country specific information from its port partner using the country's Alpha-2 Country Code defined by *[ISO 3166]*. The port partner responds with a *Country_Info* Message that contains the country specific information. The *Get_Country_Info* Message ***Shall*** be as shown in *Figure 6-29 "Get_Country_Info Message"* and *Table 6.49 "Country Code Data Object (CCDO)"*.

For example, if the request is for China information, then the Country Code Data Object (CCDO) would be CCDO [31:0] = 434E0000h for "CN" country code.

Figure 6-29 "Get_Country_Info Message"

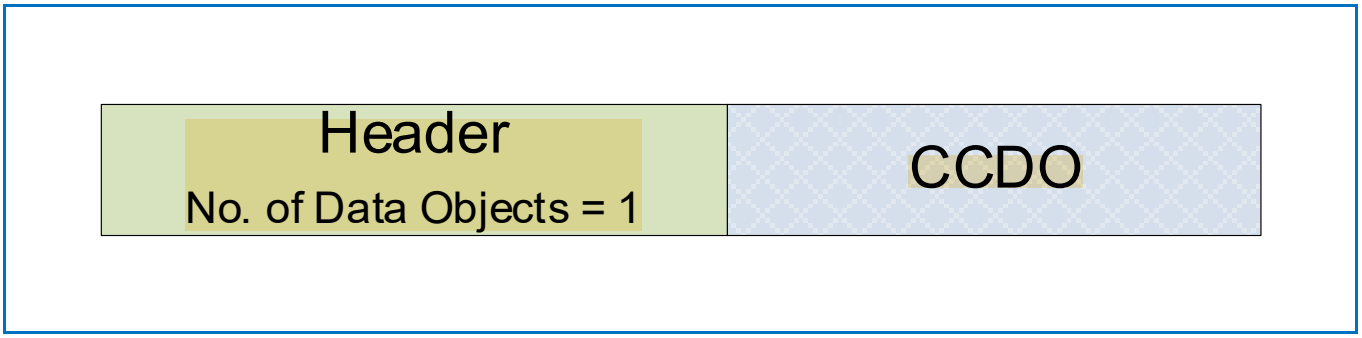


Table 6.49 "Country Code Data Object (CCDO)"

Bit(s)	Description
B31…24	First character of the Alpha-2 Country Code defined by *[ISO 3166]*
B23…16	Second character of the Alpha-2 Country Code defined by *[ISO 3166]*
B15…0	***Reserved***, ***Shall*** be set to zero.

6.4.8 Enter_USB Message

The *Enter_USB* Message ***Shall*** be sent by the DFP to its UFP Port Partner and to the Cable Plug(s) of an Active Cable, when in an Explicit Contract, to enter a specified USB Mode of operation. The recipient of the Message ***Shall*** respond by sending an *Accept* Message, a *Wait* Message or a *Reject* Message (see *Section 6.9 "Accept, Reject and Wait"*).

When entering *[USB4]* operation, the *Enter_USB* Message ***Shall*** be sent by a *[USB4]* PDUSB Hub's DFP(s) or *[USB4]* PDUSB Host's DFP(s) within *tEnterUSB*:

- following a PD Connection.
- after a Data Reset to enter *[USB4]* operation is completed.
- after a DR Swap is completed.

The *Enter_USB* Message ***May*** be sent by a PDUSB Hub's DFP(s) or PDUSB Host's DFP(s) within *tEnterUSB* following a PD Connection or after a Data Reset to enter *[USB 3.2]* or *[USB 2.0]* operation.

The *Enter_USB* Message ***Shall*** be used by a PDUSB Hub's DFP(s) to speculatively train the USB links or enter *[DPTC2.1]* or *[TBT3]* Alternate Modes prior to the presence of a host. In this case, the Host Present bit ***Shall*** be cleared. When the Host is Connected the *Enter_USB* Message ***Shall*** be resent with the Host Present bit set. The *Enter_USB* Message's Enter USB Data Object (EUDO), received from the Root Hub when the USB Host is connected, ***Shall*** be propagated down through the hub tree.

See *[USB Type-C 2.3]* USB4® Hub Connection Requirements.

The *Enter_USB* Message ***Shall*** be as shown in *Figure 6-30 "Enter_USB Message"* and *Table 6.50 "Enter_USB Data Object (EUDO)"*.

Figure 6-30 "Enter_USB Message"

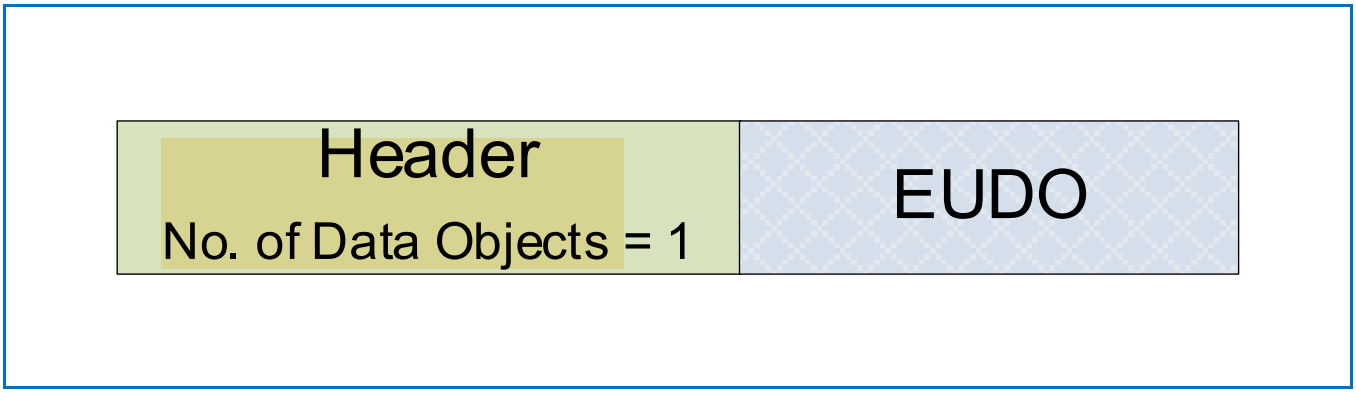


Table 6.50 "Enter_USB Data Object (EUDO)"

Bit(s)	Field	Description
B31	Reserved	***Shall*** be set to zero.
B30…28	***USB Mode***[1]	000b: ***[USB 2.0]*** 001b: ***[USB 3.2]*** 010b: ***[USB4]*** 111b…011b: ***Reserved***, ***Shall*** not be used
B27	***Reserved***	***Shall*** be set to zero.
B26	***USB4 DRD***[2]	0b: Not capable of operating as a ***[USB4]*** Device 1b: Capable of operating as a ***[USB4]*** Device
B25	***USB3 DRD***[2]	0b: Not capable of operating as a ***[USB 3.2]*** Device 1b: Capable of operating as a ***[USB 3.2]*** Device
B24	***Reserved***	***Shall*** be set to zero.
B23…21	***Cable Speed***[2,3]	000b: ***[USB 2.0]*** only, no SuperSpeed support 001b: ***[USB 3.2]*** Gen1 010b: ***[USB 3.2]*** Gen2 and ***[USB4]*** Gen2 011b: ***[USB4]*** Gen3 100b: ***[USB4]*** Gen4 101b…111b: ***Reserved***, ***Shall*** not be used
B20…19	***Cable Type***[2,3]	00b: Passive 01b: Active Re-timer 10b: Active Re-driver 11b: Optically Isolated
B18…17	***Cable Current***[2]	00b = V_{BUS} is not supported 01b = ***Reserved*** 10b = 3A 11b = 5A
B16	***PCIe Support***[2]	***[USB4]*** PCIe tunneling supported by the host
B15	***DP Support***[2]	***[USB4]*** DP tunneling supported by the host
B14	***TBT Support***[2]	***[TBT3]*** is supported by the host's USB4® Connection Manager
B13	***Host Present***[2]	Connected to a Host. When this bit is set ***PCIe Support***, ***DP Support***, and ***TBT Support*** represent the Host's capabilities that Shall be propagated down the Hub tree.
B12…0	***Reserved***	***Shall*** be set to zero.

1) Entry into ***[USB 3.2]*** and ***[USB4]*** include entry into ***[USB 2.0]***.

2) ***Shall*** be ***Ignored*** when received by a Cable Plug (e.g., SOP' or SOP'').

3) The DFP ***Shall*** interpret the Cable Plug's reported capability as defined in ***[USB Type-C 2.3]*** in the USB4 Discovery and Entry Section.

6.4.8.1 USB Mode Field

The ***USB Mode*** field ***Shall*** be used by the DFP to direct the USB Mode the Port Partner is to enter.

6.4.8.2 USB4® DRD Field

The *USB4 DRD* field ***Shall*** be set when the Host DFP is capable of operating as a *[USB4]* Device. A *[USB4]* Host DFP that sets the *USB4 DRD* field ***Shall*** also be capable of operating as a *[USB 2.0]* Device.

6.4.8.3 USB3 DRD Field

The *USB3 DRD* field Shall be set when the Host DFP is capable of operating as a *[USB 3.2]* Device. A *[USB 3.2]* Host DFP that sets the *USB3 DRD* field Shall also be capable of operating as a *[USB 2.0]* Device.

6.4.8.4 Cable Speed Field

The *Cable Speed* field ***Shall*** be used to indicate the cable's maximum speed. The value is read from the Cable Plug and interpreted by the DFP as defined by *[USB Type-C 2.3]* in the USB4 Discovery and Entry Section.

6.4.8.5 Cable Type Field

The *Cable Type* field ***Shall*** be used to indicate whether the cable is passive or active. Further if the cable is active, it indicates the type of active circuits in the cable and if the cable is optically isolated. The value is read from the Cable Plug and interpreted by the DFP as defined by *[USB Type-C 2.3]* in the USB4 Discovery and Entry Section.

6.4.8.6 Cable Current Field

The *Cable Current* field ***Shall*** be used to indicate the cable's current carrying capability. The value is reported by the Cable Plug in the Vbus Current Handling Capability field.

6.4.8.7 PCIe Support Field

The *PCIe Support* field ***Shall*** be set when the Host DFP is capable of tunneling PCIe over *[USB4]*.

The *PCIe Support* field ***May*** be set speculatively when the Hub's DFP is capable of tunneling PCIe over *[USB4]*.

6.4.8.8 DP Support Field

The *DP Support* field ***Shall*** be set when the Host DFP is capable of tunneling DP over *[USB4]*.

The *DP Support* field ***May*** be set speculatively when the Hub's DFP is capable of tunneling DP over *[USB4]*.

6.4.8.9 TBT Support Field

The *TBT Support* field ***Shall*** be set when the Host DFP is capable of tunneling Thunderbolt™ over *[USB4]* and that the Connection Manager (CM) supports discovery and configuration of Thunderbolt™ 3 devices connected to the DFP of *[USB4]* Hubs.

The *TBT Support* field ***May*** be set speculatively when the Hub's DFP is capable of tunneling Thunderbolt over *[USB4]*.

6.4.8.10 Host Present Field

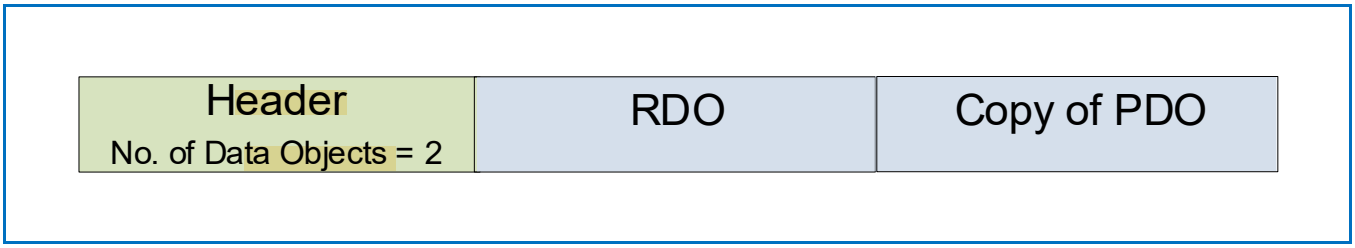
The *Host Present* field ***Shall*** be set to indicate that a Host is present upstream.

6.4.9 EPR_Request Message

An ***EPR_Request*** Message ***Shall*** be sent by a Sink, operating in EPR Mode, to request power, typically during the request phase of a power negotiation. The ***EPR_Request*** Message ***Shall*** be sent in response to the most recent ***EPR_Source_Capabilities*** Message. The ***EPR_Request*** Message ***Shall*** return a Sink Request Data Object (RDO) that ***Shall*** identify the Power Data Object being requested followed by a copy of the Power Data Object being requested. Note the requested Power Data Object ***May*** be either an EPR (A)PDO or SPR (A)PDO.

The ***EPR_Request*** Message ***Shall*** be as shown in ***Figure 6-31 "EPR_Request Message"***.

Figure 6-31 "EPR_Request Message"



The Source ***Shall*** verify the PDO in the ***EPR_Request*** Message exactly matches the PDO in the latest ***EPR_Source_Capabilities*** Message pointed to by the Object Position field in the RDO.

The Source ***Shall*** respond to an ***EPR_Request*** Message in the same manner as it responds to a ***Request*** Message with an ***Accept*** Message, or a ***Reject*** Message (see ***Section 6.9 "Accept, Reject and Wait"***). A Sink receiving a ***Wait*** response ***Shall*** initiate a Hard Reset. The Explicit Contract Negotiation process for EPR is the same as the process for SPR Mode except that the ***Source_Capabilities*** Message is replaced by the ***EPR_Source_Capabilities*** and the ***Request*** message is replaced by the ***EPR_Request*** message.

The RDO takes a different form depending on the kind of power requested. The PDO and APDO formats are detailed in ***Section 6.4.2 "Request Message"***.

6.4.10 EPR_Mode Message

The ***EPR_Mode*** Message is used to enter, acknowledge, and exit the EPR Mode. The Action field is used to describe the action that is to be taken by the recipient of the ***EPR_Mode*** Message. The Data field provides additional information for the Message recipient in the EPR Mode Data Object (ERMDO).

The ***EPR_Mode*** Message ***Shall*** be as shown in *Figure 6-32 "EPR Mode DO Message"* and *Table 6.51 "EPR Mode Data Object (EPRMDO)"*.

Figure 6-32 "EPR Mode DO Message"

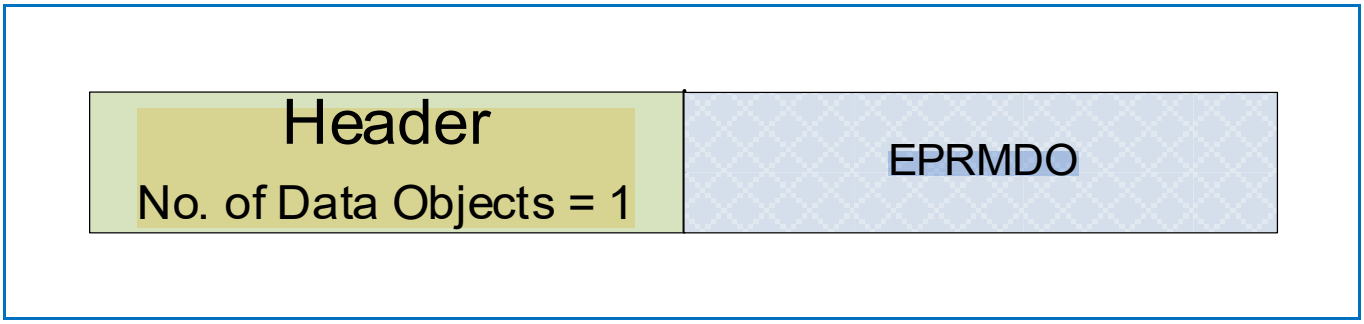


Table 6.51 "EPR Mode Data Object (EPRMDO)"

<table>
<tr><th>Bit(s)</th><th>Field</th><th>Description</th></tr>
<tr><td>B31...24</td><td>Action</td><td>
<table>
<tr><th>Value</th><th>Action</th><th>Sent By</th></tr>
<tr><td>0x00</td><td>Reserved and Shall Not be used</td><td></td></tr>
<tr><td>0x01</td><td>Enter</td><td>Sink only</td></tr>
<tr><td>0x02</td><td>Enter Acknowledged</td><td>Source only</td></tr>
<tr><td>0x03</td><td>Enter Succeeded</td><td>Source only</td></tr>
<tr><td>0x04</td><td>Enter Failed</td><td>Source only</td></tr>
<tr><td>0x05</td><td>Exit</td><td>Sink or Source</td></tr>
<tr><td>0x06...0xFF</td><td>Reserved and Shall Not be used</td><td></td></tr>
</table>
</td></tr>
<tr><td>B23...16</td><td>Data</td><td>
<table>
<tr><th>Action Field</th><th>Data Field Value</th></tr>
<tr><td>Enter</td><td>Shall be set to the EPR Sink Operational PDP</td></tr>
<tr><td>Enter Acknowledged</td><td>Shall be set to zero</td></tr>
<tr><td>Enter Succeeded</td><td>Shall be set to zero</td></tr>
<tr><td>Enter Failed</td><td>Shall be one of the following values:
0x00 - Unknown cause
0x01 - Cable not EPR capable
0x02 –Source failed to become VCONN source.
0x03 – EPR Mode Capable bit not set in RDO.
0x04 – Source unable to enter EPR Mode at this time.
0x05 - EPR Mode Capable bit not set in PDO.
All other values are Reserved and Shall Not be used</td></tr>
<tr><td>Exit</td><td>Shall be set to zero</td></tr>
</table>
</td></tr>
<tr><td>B15...0</td><td colspan="2">Reserved, Shall be set to zero.</td></tr>
</table>

6.4.10.1 Process to enter EPR Mode

For port partners to successfully enter EPR mode, the following conditions must be met:

- The Sink ***Shall*** request entry into the EPR Mode.
- The Source ***Shall*** verify the cable is EPR capable.
- A Sink ***Shall Not*** be Connected to the Source through a Charge Through VPD (CT-VPD).
- The Source and Sink ***Shall*** be in an SPR Explicit Contract.
- The EPR Mode capable bit ***Shall*** be set in the Fixed 5V PDO.
- The EPR Mode capable bit ***Shall*** have been set in the RDO in the last ***Request*** Message received by the Source.

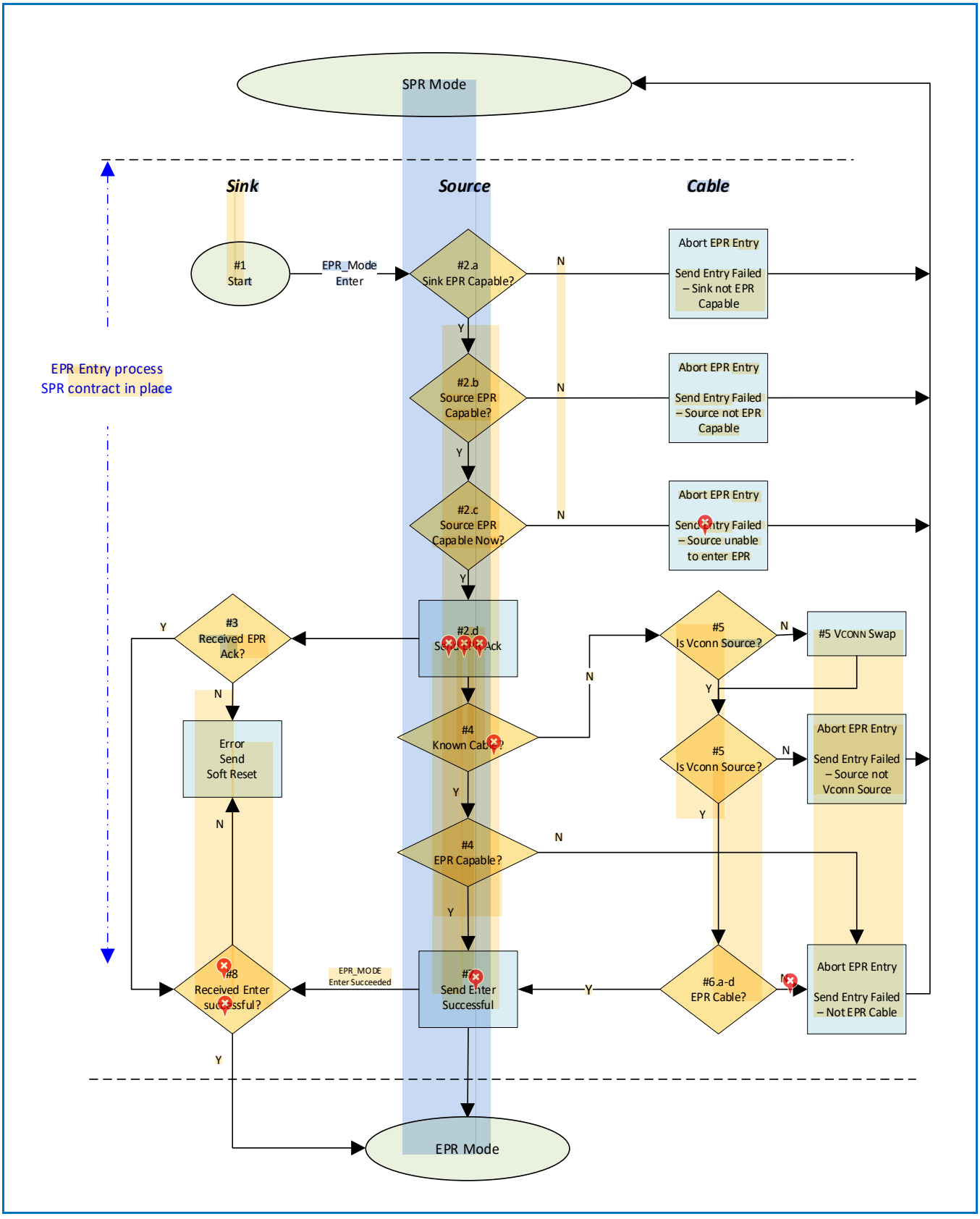
To verify the cable is EPR capable, the EPR Source ***Shall*** have already done the following:

- Discover the cable prior to entering its First Explicit Contract
- Alternatively, within ***tEPRSourceCableDiscovery*** of entry into the First Explicit Contract
  - If it is the VCONN Source, discover the cable.
  - If not the VCONN Source, do a VCONN Swap then discover the cable.

or ***Shall*** verify the cable is EPR capable by completing steps 5 and 6 in the entry process below.

The EPR Mode entry process is a non-interruptible multi-message sequence. An illustration of this sequence is shown in ***Figure 6-33 "Illustration of process to enter EPR Mode"***. Note that ***Figure 6-33 "Illustration of process to enter EPR Mode"*** is not ***Normative*** but is illustrative only.

Figure 6-33 "Illustration of process to enter EPR Mode"



The entry process ***Shall*** follow these steps in order:

1) The Sink ***Shall*** send the ***EPR_Mode*** Message with the Action field set to 1 (Enter) and the Data field set to its Operational PDP. If the EPR Source receives an ***EPR_Mode*** message with the Action field not set to Enter it ***Shall*** initiate a Soft Reset.

2) The Source ***Shall*** do the following:

   a) Verify the EPR Mode Capable bit was set in the most recent RDO. If not set, the Source ***Shall*** do the following:

      i) Send an ***EPR_Mode*** Message with the Action field set to 4 (“Enter Failed”) and the Data field set to 3 (“EPR Mode Capable bit not set in the RDO”).

      ii) Abort the EPR Mode entry process and remain in the existing SPR Explicit Contract.

   b) Verify the EPR Mode Capable bit was set in the most recent 5V fixed PDO. If not set, the Source ***Shall*** do the following:

      i) Send an ***EPR_Mode*** Message with the Action field set to 4 (“Enter Failed”) and the Data field set to 5 (“EPR Mode Capable bit not set in the fixed 5V PDO”).

      ii) Abort the EPR Mode entry process and remain in the existing SPR Explicit Contract.

   c) Verify the Source is still able to support EPR Mode. If not, the Source ***Shall*** do the following:

      i) Send an ***EPR_Mode*** Message with the Action field set to 4 (“Enter Failed”) and Data field set to 4 (“Unable at this time”).

      ii) Abort the EPR Mode entry process and remain in the existing SPR Explicit Contract

   d) Send an ***EPR_Mode*** Message with the Action field set to 2 (“Enter Acknowledged”).

3) If the Sink receives any Message, other than an ***EPR_Mode*** Message with the Action Field set to 2, the Sink ***Shall*** initiate a Soft Reset.

4) When the EPR Source has used the ***Discover Identity*** Command to determine and remembers the cable’s capabilities or the EPR Source is connected with a captive cable:

   a) If the cable is EPR capable it ***Should*** go directly to Step 7, but ***May*** continue to Step 5.

   b) If the cable is not EPR capable it ***Shall*** do the following:

   c) Send an ***EPR_Mode*** Message with the Action field set to 4 (“Enter Failed”) and the Data field set to 1 (“Cable not EPR capable”).

   d) Abort the EPR Mode entry process and remain in the existing SPR Explicit Contract.

5) If the Source is not the VCONN Source, it Shall send a ***VCONN_Swap*** Message

   a) If the Source fails to become the VCONN Source, it ***Shall***:

      i) Send an ***EPR_Mode*** message with the Action field set to 4 (Enter Failed) and the Data field set to 2 (not VCONN source).

      ii) Abort the EPR Mode entry process and remain in the existing SPR Explicit Contract.

6) The Source ***Shall*** use the ***Discover Identity*** Command to read the cable’s e-Marker and verify the following:

   a) Cable VDO - Maximum V_{BUS} Voltage field is 11b (50V)

b) Cable VDO - V_{BUS} Current Handling Capability field is 10b (5A)

c) Cable VDO - EPR Mode Capable field is 1b (EPR Mode Capable)

d) If the cable fails to respond to the ***Discover Identity*** Command or is not EPR capable, the Source ***Shall*** do the following:

i) Send an ***EPR_Mode*** Message with the Action field set to 4 ("Enter Failed") and the Data field to1 ("Cable not EPR capable").

ii) Abort the EPR Mode entry process and remain in the existing SPR Explicit Contract.

7) The Source ***Shall*** send the ***EPR_Mode*** message with the Action field set to 3 ("Enter Succeeded") and ***Shall*** enter EPR Mode.

8) If the Sink receives an ***EPR_Mode*** Message with the Action field set to 3 ("Enter Succeeded) it ***Shall*** enter EPR Mode, otherwise it ***Shall*** initiate a Soft Reset.

If the EPR Mode entry process has not been aborted or does not complete within ***tEnterEPR*** of the last bit of the ***GoodCRC*** Message sent in response to the ***EPR_Mode*** Message with the Action field set to 1 ("Enter"), the Sink ***Shall*** initiate a Soft Reset.

6.4.10.2 Operation in EPR Mode

While operating in EPR Mode, the Source ***Shall*** only send ***EPR_Source_Capabilities*** Messages to Advertise its power capabilities and the Sink ***Shall*** only respond with ***EPR_Request*** Messages to Negotiate Explicit Contracts. The ***EPR_Request*** Message ***May*** be for either an SPR or EPR (A)PDO. The Port Partners ***May*** continue to operate in EPR Mode even if they have Negotiated an SPR Explicit Contract.

If the Source sends a ***Source_Capabilities*** Message, the Sink ***Shall*** initiate a Hard Reset. If the Sink sends a ***Request*** Message, the Source ***Shall*** initiate a Hard Reset.

The Source ***Shall*** monitor the CC communications path to ensure that there is periodic traffic. The Sink ***Shall*** send an ***EPR_KeepAlive*** Message when it has not sent any Messages for more than ***tSinkEPRKeepAlive*** to ensure there is timely periodic traffic. If there is no traffic for more than ***tSourceEPRKeepAlive***, the Source ***Shall*** initiate a Hard Reset.

6.4.10.3 Exiting EPR Mode

6.4.10.3.1 Commanded Exit

While in EPR Mode, either the Source or Sink ***May*** exit EPR Mode by sending an ***EPR_Mode*** message with the Action field set to 5 ("Exit").

The ports ***Shall*** be in a power contract with an SPR (A)PDO prior to the EPR Mode exit process by either:

- The Source sending an ***EPR_Source_Capabilities*** Message with no EPR PDOs (e.g., only SPR PDOs) or
- The Sink negotiating a new Explicit Contract with bit 31 in the RDO set to zero (e.g., not an EPR PDO)

The process to exit EPR Mode is a non-interruptible multi-message sequence and ***Shall*** follow these steps in order:

1) The Port Partners ***Shall*** be in an Explicit Contract with an SPR (A)PDO.

2) Either the Source or Sink ***Shall*** send an ***EPR_Mode*** message with the Action field set to 5 (“Exit”) to exit the EPR Mode

3) The Source Shall send a ***Source_Capabilities*** Message within ***tFirstSourceCap*** of the ***GoodCRC*** Message in response to the ***EPR_Mode*** Message with the Action field set to 5 (“Exit”).

4) If the Sink does not receive a ***Source_Capabilities*** Message within ***tTypeCSinkWaitCap*** of the last bit of the ***GoodCRC*** Message in response to the ***EPR_Mode*** Message with the Action field set to 5 (“Exit”), Sink Shall initiate a Hard Reset.

6.4.10.3.2 Implicit Exit

EPR Mode ***Shall*** be exited as the side-effect of the PR Swap and FR Swap processes. This is because at the end of these processes V_{BUS} will be at ***vSafe5V*** and the Ports will be in an Implicit Contract. The new Source will then send a ***Source_Capabilities*** Message (not an ***EPR_Source_Capabilities*** Message) to begin the process of negotiating an SPR Explicit Contract. Once an SPR Explicit Contract is entered, the Source and Sink can then enter EPR Mode if needed.

6.4.10.3.3 Exits due to errors

Other critical errors can occur while in EPR Mode; these errors ***Shall*** result in Hard Reset being initiated by the Port that detects the error. Some of these errors include:

- An ***EPR_Mode*** Message with the Action field set to 5 (“Exit”) to exit EPR Mode is received by a Port in an Explicit contract with an EPR (A)PDO.
- The Sink receives an ***EPR_Source_Capabilities*** Message with an EPR (A)PDO in any of the first seven object positions.
- The PDO in the ***EPR_Request*** Message does not match the PDO in the latest ***EPR_Source_Capabilities*** Message pointed to by the Object Position field in the RDO.
- The Source receives a ***Request*** Message.
- The Sink receives a ***Source_Capabilities*** Message not in response to a ***Get_Source_Cap*** Message.

6.4.11 Source_Info Message

The *Source_Info* Message ***Shall*** be sent in response to a *Get_Source_Info* Message. The *Source_Info* Message contains one Source Information Data Object (SIDO).

The *Source_Info* Message returns a SIDO whose format ***Shall*** be as shown in *Figure 6-34 "Source_Info Message"* and *Table 6.52 "Source_Info Data Object (SIDO)"*. The *Number of Data Objects* field in the *Source_Info* Message ***Shall*** be set to 1.

The *Port Maximum PDP*, *Port Present PDP*, *Port Reported PDP* and the *Port Type* are used to identify capabilities of a Source port.

Figure 6-34 "Source_Info Message"

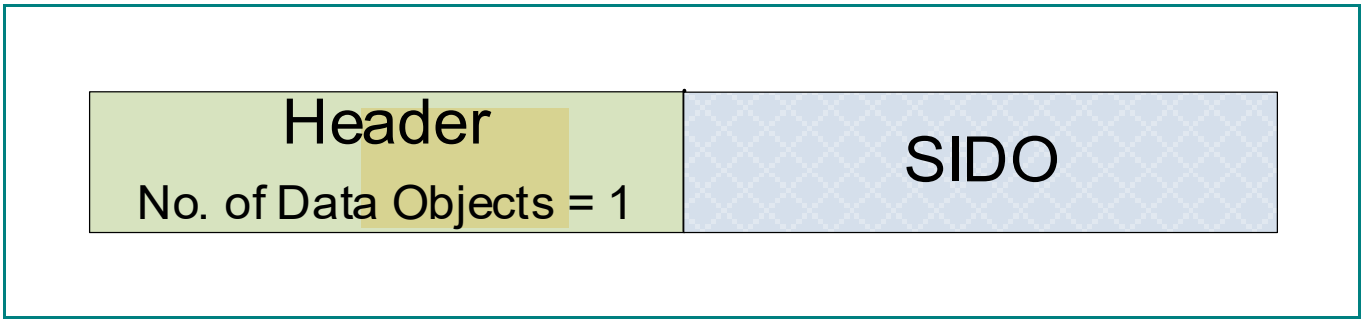


Table 6.52 "Source_Info Data Object (SIDO)"

Bit(s)	Field	Description
B31	*Port Type*	0 = Managed Capability Port 1 = Guaranteed Capability Port
B30…24	***Reserved***	***Shall*** be set to zero
B23…16	*Port Maximum PDP*	Power the port is designed to supply
B15…8	*Port Present PDP*	Power the port is presently capable of supplying
B7…0	*Port Reported PDP*	Power the port is actually advertising

6.4.11.1 Port Type Field

Port Type is a static field that ***Shall*** be used to indicate whether the amount of power the port can provide is fixed or can change dynamically.

A Guaranteed Capability Port ***Shall*** always report its *Port Maximum PDP* equal to its *Port Present PDP* when the correct cable is used (e.g., 5A for Sources with PDPs greater than 60W or EPR Capable for EPR capable Sources). A Managed Capability Port is not required to have its *Port Maximum PDP* equal to its *Port Present PDP*.

6.4.11.2 Port Maximum PDP Field

Port Maximum PDP is a static field that ***Shall*** indicate the maximum amount of power the Port is designed to deliver. A Guaranteed Capability Port (as indicated by the Port Type field being set to '1') ***Shall*** always be capable of supplying this amount of power. A Managed Capability Port (as indicated by the Port Type field being set to '0') ***Shall*** be able to offer this amount of power at some time.

The *Port Maximum PDP* ***Shall*** be the same as the larger of the Source PDP Rating and the EPR Source PDP Rating in the *Source_Capabilities_Extended* Message.

6.4.11.3 Port Present PDP Field

The ***Port Present PDP*** is a Static field when the ***Port Type*** is Guaranteed Capability and is dynamic when the ***Port Type*** field is Managed Capability. It ***Shall*** indicate the amount of power the port is presently capable of supplying. A Guaranteed Capability port ***Shall*** always set its ***Port Present PDP*** to be the same as its ***Port Maximum PDP*** except when limited by the cable's capabilities. A Managed Capability Port ***Shall*** set its ***Port Present PDP*** to the amount of power it has available to offer at this time which might be limited by the cable's capabilities.

6.4.11.4 Port Reported PDP Field

The ***Port Reported PDP*** field Shall track the amount of power the Port is offering in its ***Source_Capabilities*** Message or ***EPR_Source_Capabilities*** Message. The ***Port Reported PDP*** field ***May*** be dynamic or static depending on the Port's other characteristics such as Managed/Guaranteed Capability, SPR/EPR mode, its power policy etc.

Note: The ***Port Reported PDP*** field is computed as the largest of the products of the Voltage times current of the fixed PDOs returned in the ***Source_Capabilities*** Message or ***EPR_Source_Capabilities*** Messages.

6.4.12 Revision Message

The ***Revision*** Message ***Shall*** be sent in response to the ***Get_Revision*** Message sent by the Port Partner. This Message is used to identify the highest revision the port is capable of operating at. The ***Revision*** Message contains one Revision Message Data Object (RMDO).

The ***Revision*** Message returns an RMDO whose format ***Shall*** be as shown in ***Figure 6-36 "Source_Capabilities_Extended Message"*** and ***Table 6.53 "Revision Message Data Object (RMDO)"***. The ***Number of Data Objects*** field in the ***Revision*** Message ***Shall*** be set to 1.

Figure 6-35 "Revision Message Data Object"

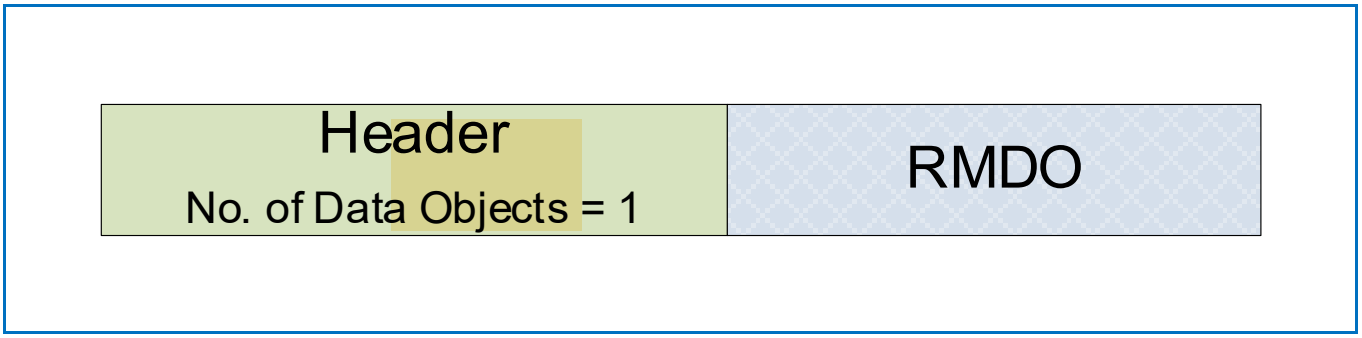


Table 6.53 "Revision Message Data Object (RMDO)"

Bit(s)	Description
B31...28	Revision.major
B27...24	Revision.minor
B23...20	Version.major
B19...16	Version.minor
B15...0	***Reserved***, ***Shall*** be set to zero.

E.g., for Revision 3.2, Version 1.0 the fields would be the following:

- Revision.major = 0011b
- Revision.minor = 0010b
- Version.major = 0001b
- Version.minor – 0000b

6.5 Extended Message

An Extended Message ***Shall*** contain an Extended Message Header (indicated by the ***Extended*** field in the Message Header being set) and be followed by zero or more data bytes. Additional bytes that might be added to existing Messages in future revision of this specification ***Shall*** be ***Ignored***.

The format of the Extended Message is defined by the Message Header's ***Message Type*** field and is summarized in ***Table 6.54 "Extended Message Types"***. The Sent by column indicates entities which ***May*** send the given Message (Source, Sink or Cable Plug); entities not listed ***Shall Not*** issue the corresponding Message. The "***Valid*** Start of Packet" column indicates the Messages which ***Shall*** only be issued in SOP Packets and the Messages which ***May*** be issued in SOP* Packets.

Table 6.54 "Extended Message Types"

Bits 4...0	Type	Sent by	Description	Valid Start of Packet
0 0000	***Reserved***		All values not explicitly defined are ***Reserved*** and ***Shall Not*** be used.	
0 0001	*Source_Capabilities_Extended*	Source or Dual-Role Power	See *Section 6.5.1*	SOP only
0 0010	*Status*	Source, Sink or Cable Plug	See *Section 6.5.2*	SOP*
0 0011	*Get_Battery_Cap*	Source or Sink	See *Section 6.5.3*	SOP only
0 0100	*Get_Battery_Status*	Source or Sink	See *Section 6.5.4*	
0 0101	*Battery_Capabilities*	Source or Sink	See *Section 6.5.5*	SOP only
0 0110	*Get_Manufacturer_Info*	Source or Sink	See *Section6.5.6*	SOP*
0 0111	*Manufacturer_Info*	Source, Sink or Cable Plug	See *Section 6.5.7*	SOP*
0 1000	*Security_Request*	Source or Sink	See *Section 6.5.8.1*	SOP*
0 1001	*Security_Response*	Source, Sink or Cable Plug	See *Section 6.5.8.2*	SOP*
0 1010	*Firmware_Update_Request*	Source or Sink	See *Section 6.5.9.1*	SOP*
0 1011	*Firmware_Update_Response*	Source, Sink or Cable Plug	See *Section 6.5.9.2*	SOP*
0 1100	*PPS_Status*	Source	See *Section 6.5.10*	SOP only
0 1101	*Country_Info*	Source or Sink	See *Section 6.5.12*	SOP only
0 1110	*Country_Codes*	Source or Sink	See *Section 6.5.11*	SOP only
0 1111	*Sink_Capabilities_Extended*	Sink or Dual-Role Power	See *Section 6.5.13*	SOP only
1 0000	*Extended_Control*	Source or Sink	See *Section 6.5.14*	SOP only
1 0001	*EPR_Source_Capabilities*	Source or Dual-Role Power	See *Section 6.5.15*	SOP only
1 0010	*EPR_Sink_Capabilities*	Sink or Dual-Role Power	See *Section 6.5.15*	SOP only
1 0011 - 1 1101	***Reserved***		All values not explicitly defined are ***Reserved*** and ***Shall Not*** be used.	
1 1110	*Vendor_Defined_Extended*	Source, Sink or Cable Plug	See *Section 6.5.16*	SOP*
1 1111	***Reserved***		All values not explicitly defined are ***Reserved*** and ***Shall Not*** be used.	

6.5.1 Source_Capabilities_Extended Message

The *Source_Capabilities_Extended* Message ***Should*** be sent in response to a *Get_Source_Cap_Extended* Message. The *Source_Capabilities_Extended* Message enables a Source or a DRP to inform the Sink about its capabilities as a Source.

The *Source_Capabilities_Extended* Message ***Shall*** return a 25-byte Source Capabilities Extended Data Block (SCEDB) whose format ***Shall*** be as shown in *Figure 6-36 "Source_Capabilities_Extended Message"* and *Table 6.55 "Source Capabilities Extended Data Block (SCEDB)"*.

Figure 6-36 "Source_Capabilities_Extended Message"

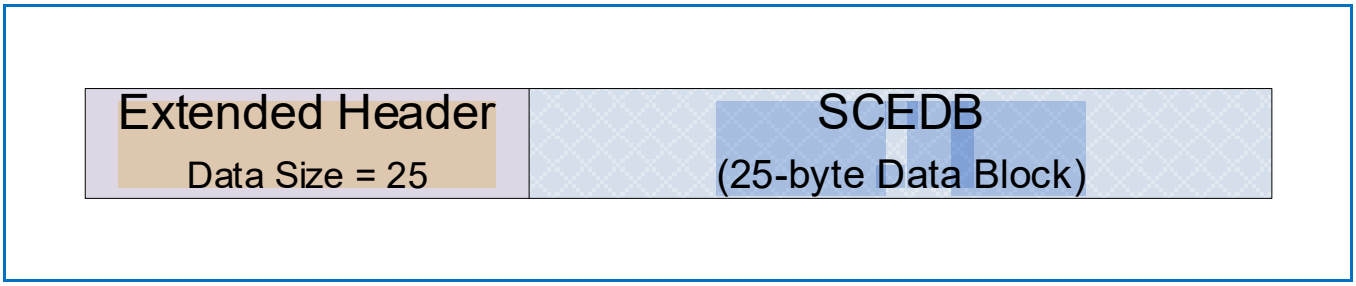


Table 6.55 "Source Capabilities Extended Data Block (SCEDB)"

Offset	Field	Description
0	VID	Vendor ID (assigned by the USB-IF)
2	PID	Product ID (assigned by the manufacturer)
4	XID	Value provided by the USB-IF assigned to the product
8	FW Version	Firmware version number
9	HW Version	Hardware version number
10	Voltage Regulation	<table><tr><th>Bit</th><th>Description</th></tr><tr><td>1...0</td><td>00b: 150mA/µs Load Step (default) 01b: 500mA/µs Load Step 11b...10b: ***Reserved*** and ***Shall Not*** be used</td></tr><tr><td>2</td><td>0b: 25% IoC (default) 1b: 90% IoC</td></tr><tr><td>3...7</td><td>***Reserved*** and ***Shall*** be set to zero</td></tr></table>
11	Holdup Time	Output will stay with regulated limits for this number of milliseconds after removal of the AC from the input. 0x00 = feature not supported **Note:** a value of 3ms ***Should*** be used

<table>
<tr><td>12</td><td>Compliance</td><td>Compliance in SPR Mode:<table><tr><th>Bit</th><th>Description</th></tr><tr><td>0</td><td>LPS compliant when set</td></tr><tr><td>1</td><td>PS1 compliant when set</td></tr><tr><td>2</td><td>PS2 compliant when set</td></tr><tr><td>3…7</td><td>Reserved and Shall be set to zero</td></tr><tr><td></td><td></td></tr></table></td></tr>
<tr><td>13</td><td>Touch Current</td><td><table><tr><th>Bit</th><th>Description</th></tr><tr><td>0</td><td>Low touch Current EPS when set</td></tr><tr><td>1</td><td>Ground pin supported.
when set</td></tr><tr><td>2</td><td>Ground pin intended for protective earth when set</td></tr><tr><td>3…7</td><td>Reserved and Shall be set to zero</td></tr></table></td></tr>
<tr><td>14</td><td>Peak Current1</td><td><table><tr><th>Bit</th><th>Description</th></tr><tr><td>0…4</td><td>Percent overload in 10% increments
Values higher than 25 (11001b) are clipped to 250%.</td></tr><tr><td>5…10</td><td>Overload period in 20ms</td></tr><tr><td>11.14</td><td>Duty cycle in 5% increments</td></tr><tr><td>15</td><td>V_{BUS} Voltage droop</td></tr></table></td></tr>
<tr><td>16</td><td>Peak Current2</td><td><table><tr><th>Bit</th><th>Description</th></tr><tr><td>0…4</td><td>Percent overload in 10% increments
Values higher than 25 (11001b) are clipped to 250%.</td></tr><tr><td>5…10</td><td>Overload period in 20ms</td></tr><tr><td>11.14</td><td>Duty cycle in 5% increments</td></tr><tr><td>15</td><td>V_{BUS} Voltage droop</td></tr></table></td></tr>
<tr><td>18</td><td>Peak Current3</td><td><table><tr><th>Bit</th><th>Description</th></tr><tr><td>0…4</td><td>Percent overload in 10% increments
Values higher than 25 (11001b) are clipped to 250%.</td></tr><tr><td>5…10</td><td>Overload period in 20ms</td></tr><tr><td>11.14</td><td>Duty cycle in 5% increments</td></tr><tr><td>15</td><td>V_{BUS} Voltage droop</td></tr></table></td></tr>
</table>

<table>
<tr><td>20</td><td>Touch Temp</td><td>Temperature conforms to:
0 = [IEC 60950-1] (default)
1 = [IEC 62368-1] TS1
2 = [IEC 62368-1] TS2
Note: All other values Reserved</td></tr>
<tr><td>21</td><td>Source Inputs</td><td><table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>0b: No external supply
1b: External supply present</td></tr>
<tr><td>1</td><td>If bit 0 is set:
0b: External supply is constrained.
1b: External supply is unconstrained.
If bit 0 is not set Reserved and Shall be set to zero</td></tr>
<tr><td>2</td><td>0b: No internal Battery
1b: Internal Battery present</td></tr>
<tr><td>3…7</td><td>Reserved and Shall be set to zero</td></tr>
</table></td></tr>
<tr><td>22</td><td>Number of Batteries/Battery Slots</td><td>Upper Nibble = Number of Hot Swappable Battery Slots (0…4)
Lower Nibble = Number of Fixed Batteries (0…4)</td></tr>
<tr><td>23</td><td>SPR Source PDP Rating</td><td>0…6: Source PDP rating
7: Reserved and Shall be set to zero</td></tr>
<tr><td>24</td><td>EPR Source PDP Rating</td><td>0…7: EPR Source PDP Rating</td></tr>
</table>

6.5.1.1 Vendor ID (VID) Field

The Vendor ID field ***Shall*** contain the 16-bit Vendor ID (VID) assigned to the Source's vendor by the USB-IF. If the vendor does not have a VID, the Vendor ID field ***Shall*** be set to 0xFFFF. Devices that have a USB data interface ***Shall*** report the same VID as the idVendor in the Standard Device Descriptor (see ***[USB 2.0]*** and ***[USB 3.2]***).

6.5.1.2 Product ID (PID) Field

The Product ID field ***Shall*** contain the 16-bit Product ID (PID) assigned by the Source's vendor. Devices that have a USB data interface ***Shall*** report the same PID as the idProduct in the Standard Device Descriptor (see ***[USB 2.0]*** and ***[USB 3.2]***).

6.5.1.3 XID Field

The XID field ***Shall*** contain the 32-bit XID provided by the USB-IF to the vendor who in turns assigns it to a product. If the vendor does not have an XID, then it ***Shall*** return zero in this field (see ***[USB 2.0]*** and ***[USB 3.2]***).

6.5.1.4 Firmware Version Field

The Firmware Version field ***Shall*** contain an 8-bit firmware version number assigned to the device by the vendor.

6.5.1.5 Hardware Version Field

The Hardware Version field ***Shall*** contain an 8-bit hardware version number assigned to the device by the vendor.

6.5.1.6 Voltage Regulation Field

The Voltage Regulation field contains bits covering Load Step Slew Rate and Magnitude.

See *Section 7.1.12.1 "Voltage Regulation Field"* for further details.

6.5.1.6.1 Load Step Slew Rate

The Source ***Shall*** report its load step response capability in bits 0…1 of the Voltage Regulation bit field.

6.5.1.6.2 Load Step Magnitude

The Source ***Shall*** report its load step magnitude rate as a percentage of IoC in bit 2 of the Voltage Regulation field.

6.5.1.7 Holdup Time Field

The Holdup Time field ***Shall*** contain the Source's holdup time (see *Section 7.1.12.2 "Holdup Time Field"*).

6.5.1.8 Compliance Field

The Compliance field is static and ***Shall*** contain the standards the Source is compliant with in SPR (see *Section 7.1.12.3 "Compliance Field"*).

6.5.1.9 Touch Current Field

The Touch Current field reports whether the Source meets certain leakage current levels and if it has a ground pin.

A Source ***Shall*** set the Touch Current bit (bit 0) when their leakage current is less than 65µA rms when Source's maximum capability is less than or equal to 30W, or when their leakage current is less than 100 µA rms when its power capability is between 30W and 100W. The total combined leakage current ***Shall*** be measured in accordance with *[IEC 60950-1]* when tested at 250VAC rms at 50 Hz.

A Source with a ground pin ***Shall*** set the Ground pin bit (bit 1).

A Source whose Ground pin is intended to be connected to a protective earth ***Shall*** set both bit1 and bit 2.

6.5.1.10 Peak Current Field

The Peak Current field ***Shall*** contain the combinations of Peak Current that the Source supports (see *Section 7.1.12.4 "Peak Current"*).

Peak Current provides a means for Source report its ability to provide current in excess of the negotiated amount for short periods. The Peak Current descriptor defines up to three combinations of % overload, duration and duty cycle defined as PeakCurrent1, PeakCurrent2 and PeakCurrent3 that the Source supports. A Source ***May*** offer no Peak Current capability. A Source ***Shall*** populate unused Peak Current bit fields with zero.

The Bit Fields within Peak Current1, Peak Current2, and Peak Current3 contain the following subfields:

- Percentage Overload
  - ***Shall*** be the maximum peak current reported in 10% increments as a percentage of the negotiated operating current (IoC) offered by the Source. Values higher than 25 (11001b) are clipped to 250%.
- Overload Period
  - ***Shall*** be the minimum rolling average time window in 20ms increments, where a value of 20ms is recommended.

- Duty Cycle
  - ***Shall*** be the maximum percentage of overload period reported in 5% increments. The values ***Should*** be 5%, 10% and 50% for PeakCurrent1, PeakCurrent2, and PeakCurrent3, respectively.
- V_{BUS} Droop
  - ***Shall*** be set to one to indicate there is an additional 5% Voltage droop on V_{BUS} when the overload conditions occur as defined by ***vSrcPeak***. However, it is recommended that the Source ***Should*** provide V_{BUS} in the range of ***vSrcNew*** when overload conditions occur and set this bit to zero.

6.5.1.11 Touch Temp Field

The Touch Temp field ***Shall*** report the IEC standard used to determine the surface temperature of the Source's enclosure. Safety limits for the Source's touch temperature are set in applicable product safety standards (e.g., ***[IEC 60950-1]*** or ***[IEC 62368-1]***). The Source ***May*** report when its touch temperature performance conforms to the TS1 or TS2 limits described in ***[IEC 62368-1]***.

6.5.1.12 Source Inputs Field

The Source Inputs field ***Shall*** identify the possible inputs that provide power to the Source. Note some Sources are only powered by a Battery (e.g., an automobile) rather than the more common mains.

- When bit 0 is set, the Source can be sourced by an external power supply.
- When bits 0 and 1 are set, the Source can be sourced by an external power supply which is assumed to be effectively "infinite" i.e., it won't run down over time.
- When bit 2 is set the Source can be sourced by an internal Battery.

Bit 2 ***May*** be set independently of bits 0 and 1.

6.5.1.13 Number of Batteries/Battery Slots Field

The Number of Batteries/Battery Slots field ***Shall*** report the number of Fixed Batteries and Hot Swappable Battery Slots the Source supports. This field ***Shall*** independently report the number of Battery Slots and the number of Fixed Batteries.

A Source ***Shall*** have no more than 4 Fixed Batteries and no more than 4 Battery Slots.

Fixed Batteries ***Shall*** be numbered consecutively from 0 to 3. The number assigned to a given Fixed Battery ***Shall Not*** change between Attach and Detach.

Battery Slots ***Shall*** be numbered consecutively from 4 to 7. The number assigned to a given Battery Slot ***Shall Not*** change between Attach and Detach.

6.5.1.14 SPR Source PDP Rating Field

The SPR Source PDP Rating field ***Shall*** report the integer portion of the Source's Source PDP Rating, when operating in SPR Mode, as defined in ***Table 10.2 "SPR Normative Voltages and Minimum Currents"***, ***Table 10.12 "EPR Source Capabilities based on the Port Maximim*** PDP and using an EPR Capable Cable" and ***Table 10.13 "EPR Source Capabilities when Port Present PDP is less than*** Port Maximum PDP and using an EPR-capable cable".

The Source PDP Rating field that is reported ***Shall*** be invariant and ***Shall*** follow the ***[USB Type-C 2.3]*** requirements for single-port, Multi-port Assured Capacity Chargers, or Multi-port Shared Capacity Chargers.

6.5.1.15 EPR Source PDP Rating Field

The EPR Source PDP Rating field ***Shall*** report the integer portion of the EPR Source's Source PDP Rating as defined in *Table 10.12 "EPR Source Capabilities based on the Port Maximim* PDP and using an EPR Capable Cable". If the Source is not an EPR capable Source, this field ***Shall*** be set to zero.

The EPR Source PDP Rating field that is reported ***Shall*** be invariant and ***Shall*** follow the *[USB Type-C 2.3]* requirements for single-port, Multi-port Assured Capacity Chargers, or Multi-port Shared Capacity Chargers.

6.5.2 Status Message

The ***Status*** Message ***Shall*** be sent in response to a ***Get_Status*** Message. The content of the ***Status*** Message depends on the target of the ***Get_Status*** Message. When sent to ***SOP*** the Status Message returns the status of the Port's Port Partner. When sent to ***SOP'*** or ***SOP''*** the ***Status*** Message returns the status of one of the *Active Cable*'s Cable Plugs.

6.5.2.1 SOP Status Message

A ***Status*** Message, sent in response to ***Get_Status*** Message to ***SOP***, enables a Port to inform its Port Partner about the present status of the Source or Sink. Typically, a ***Get_Status*** Message will be sent by the Port after receipt of an ***Alert*** Message. Some of the reported events are critical such as OCP, OVP and OTP, while others are ***Informative*** such as change in a Battery's status from charging to neither charging nor discharging.

The ***Status*** Message returns a 7-byte Status Data Block (SDB) whose format ***Shall*** be as shown in ***Figure 6-37 "SOP Status Message"*** and ***Table 6.56 "SOP Status Data Block (SDB)"***.

Figure 6-37 "SOP Status Message"

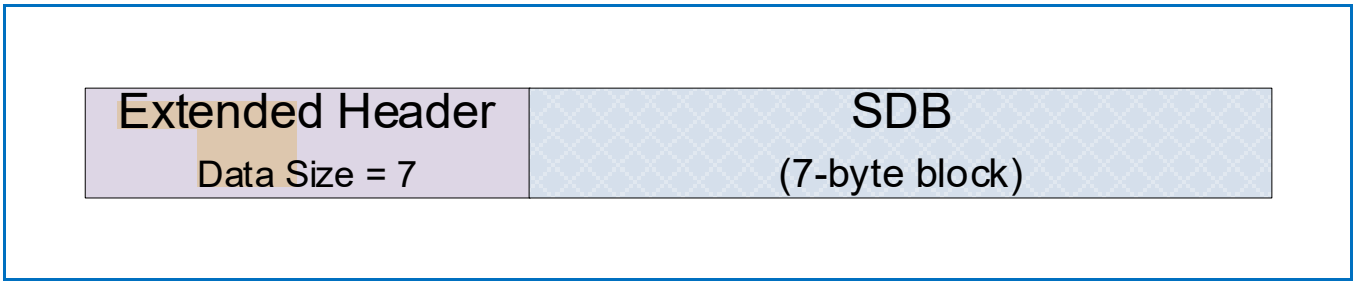


Table 6.56 "SOP Status Data Block (SDB)"

<table>
<tr><th>Offset (Byte)</th><th>Field</th><th>Description</th></tr>
<tr><td>0</td><td>Internal Temp</td><td>Source or Sink's internal temperature in °C
0 = feature not supported
1 = temperature is less than 2°C.
2-255 = temperature in °C.</td></tr>
<tr><td>1</td><td>Present Input</td><td><table><tr><th>Bit</th><th>Description</th></tr><tr><td>0</td><td>Reserved and Shall be set to zero</td></tr><tr><td>1</td><td>External Power when set</td></tr><tr><td>2</td><td>External Power AC/DC (Valid when Bit 1 set)
• 0: DC
• 1: AC
Reserved when Bit 1 is zero</td></tr><tr><td>3</td><td>Internal Power from Battery when set</td></tr><tr><td>4</td><td>Internal Power from non-Battery power source when set</td></tr><tr><td>5…7</td><td>Reserved and Shall be set to zero</td></tr></table></td></tr>
<tr><td>2</td><td>Present Battery Input</td><td>When Present Input field bit 3 set Shall contain the bit corresponding to the Battery or Batteries providing power:

Upper nibble = Hot Swappable Battery (b7…4)
Lower nibble = Fixed Battery (b3…0)

When Present Source Input field bit 3 is not set this field is Reserved and Shall be set to zero.</td></tr>
<tr><td>3</td><td>Event Flags</td><td><table><tr><th>Bit</th><th>Description</th></tr><tr><td>0</td><td>Reserved and Shall be set to zero</td></tr><tr><td>1</td><td>OCP event when set</td></tr><tr><td>2</td><td>OTP event when set</td></tr><tr><td>3</td><td>OVP event when set</td></tr><tr><td>4</td><td>CF mode when set, CV mode when cleared</td></tr><tr><td>5…7</td><td>Reserved and Shall be set to zero</td></tr></table></td></tr>
<tr><td>4</td><td>Temperature Status</td><td><table><tr><th>Bit</th><th>Description</th></tr><tr><td>0</td><td>Reserved and Shall be set to zero</td></tr><tr><td>1…2</td><td>00 – Not Supported.
01 – Normal
10 – Warning
11 – Over temperature</td></tr><tr><td>3…7</td><td>Reserved and Shall be set to zero</td></tr></table></td></tr>
</table>

<table>
<tr><td>5</td><td>Power Status</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>Reserved and Shall be set to zero</td></tr>
<tr><td>1</td><td>Source power limited due to cable supported current</td></tr>
<tr><td>2</td><td>Source power limited due to insufficient power available while sourcing other ports</td></tr>
<tr><td>3</td><td>Source power limited due to insufficient external power</td></tr>
<tr><td>4</td><td>Source power limited due to Event Flags in place (Event Flags must also be set)</td></tr>
<tr><td>5</td><td>Source power limited due to temperature</td></tr>
<tr><td>6…7</td><td>Reserved and Shall be set to zero</td></tr>
</table>
</td></tr>
<tr><td>6</td><td>Power State Change</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0…2</td><td>New Power State
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Status not supported</td></tr>
<tr><td>1</td><td>S0</td></tr>
<tr><td>2</td><td>Modern Standby</td></tr>
<tr><td>3</td><td>S3</td></tr>
<tr><td>4</td><td>S4</td></tr>
<tr><td>5</td><td>S5 (Off with battery, wake events supported)</td></tr>
<tr><td>6</td><td>G3 (Off with no battery, wake events not supported)</td></tr>
<tr><td>7</td><td>Reserved and Shall be set to zero</td></tr>
</table>
</td></tr>
<tr><td>3…5</td><td>New power state indicator
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Off LED</td></tr>
<tr><td>1</td><td>On LED</td></tr>
<tr><td>2</td><td>Blinking LED</td></tr>
<tr><td>3</td><td>Breathing LED</td></tr>
<tr><td>4…7</td><td>Reserved and Shall be set to zero</td></tr>
</table>
</td></tr>
<tr><td>6…7</td><td>Reserved and Shall be set to zero</td></tr>
</table>
</td></tr>
</table>

6.5.2.1.1 Internal Temp Field

The Internal Temp field reports the instantaneous temperature of a portion of the Source or Sink.

6.5.2.1.2 Present Input Field

The Present Input field indicates which supplies are presently powering the Source or Sink.

The following bits are defined:

- Bit 1 indicates that an external Source is present.
- Bit 2 indicates whether the external unconstrained Source is AC or DC.
- Bit 3 indicates that power is being provided from Battery.
- Bit4 indicates an alternative internal source of power that is not a Battery.

6.5.2.1.3 Present Battery Input Field

The Present Battery Input field indicates which Battery or Batteries are presently supplying power to the Source or Sink. The Present Battery Input field is only ***Valid*** when the Present Input field indicates that there is Internal Power from Battery.

The upper nibble of the field indicates which Hot Swappable Battery/Batteries are supplying power with bit 4 in upper nibble corresponding to Battery 4 and bit 7 in the upper nibble corresponding to Battery 7 (see *Section 6.5.3 "Get_Battery_Cap Message"* and *Section 6.5.4 "Get_Battery_Status Message"*).

The lower nibble of the field indicates which Fixed Battery/Batteries are supplying power with bit 0 in lower nibble corresponding to Battery 0 and bit 3 in the lower nibble corresponding to Battery 3 (see *Section 6.5.3 "Get_Battery_Cap Message"* and *Section 6.5.4 "Get_Battery_Status Message"*).

6.5.2.1.4 Event Flags Field

The Event Flags field returns event flags. The OTP, OVP and OCP event flags ***Shall*** be set when there is an event and ***Shall*** only be cleared when read with the *Get_Status* Message.

When the OTP event flag is set the Temperature Status field ***Shall*** also be set to over temperature.

The CL/CV mode bit is only ***Valid*** when operating as a Programmable Power Supply and ***Shall*** be ***Ignored*** otherwise. When the Source is operating as a Programmable Power Supply the CL/CV mode bit ***Shall*** be set when operating in Current Limit mode (CL mode) and ***Shall*** be cleared when operating in Constant Voltage mode (CV mode).

6.5.2.1.5 Temperature Status Field

The Temperature Status field returns the current temperature status of the device either: normal, warning or over temperature. When the Temperature Status field is set to over temperature the OTP event flag ***Shall*** also be set.

6.5.2.1.6 Power Status Field

The Power Status field indicates the current status of a Source. A non-zero return of the field indicates Advertised Source power is being reduced for either: the cable does not support the full Source current, the Source is supplying power to other ports and is unable to provide its full power, the external power to the Source is insufficient to support full power, or an Event has occurred that is causing the Source to reduce it Advertised power.

A Sink ***Shall*** set this field to zero.

6.5.2.1.7 Power state change

6.5.2.1.7.1 New power state

The Power state change status byte indicates a power state change to one of the specified power states. Any device that supports the ACPI standard system power states ***Shall*** use the ACPI states. For devices that do not support the ACPI power states, the following mapping ***Should*** be used:

- High power (on) state -> S0
- Sleep state -> S3

- Low power (off) state -> S5 or G3

6.5.2.1.7.2 New power state indicator

The Power indicator value defines the host's desired indicator for the specified power state. This indicator allows several possibilities for pre-defined behaviors that the host can specify to indicate its system power state to the user via the downstream device. The New power state indicator is a "best effort" indicator. If the device cannot provide the requested indicator, then it provides the best indicator that it can. If a Breathing indicator cannot be provided, then a Blinking indicator ***Should*** be provided. If a Blinking indicator cannot be provided, then a constant on indicator ***Should*** be provided.

New power state indicators in decreasing precedence:

- Breathing
- Blinking
- Constant on
- No indicator

6.5.2.2 SOP'/SOP'' Status Message

A ***Status*** Message, sent in response to a ***Get_Status*** Message to ***SOP'*** or ***SOP''***, enables a Source or Sink to get the present status of the Cable's Cable Plug(s). Typically, a ***Get_Status*** Message will be used by the USB Host and/or USB Device to manage the Cable's Cable Plug(s) temperature. The ***Status*** Message returns a 2-byte Status Data Block (SDB) whose format ***Shall*** be as shown in ***Figure 6-38 "SOP'/SOP'' Status Message"*** and ***Table 6.57 "SOP'/SOP'' Status Data Block (SDB)"***.

Passive Cable Plugs ***Shall Not*** indicate Thermal Shutdown.

Figure 6-38 "SOP'/SOP'' Status Message"

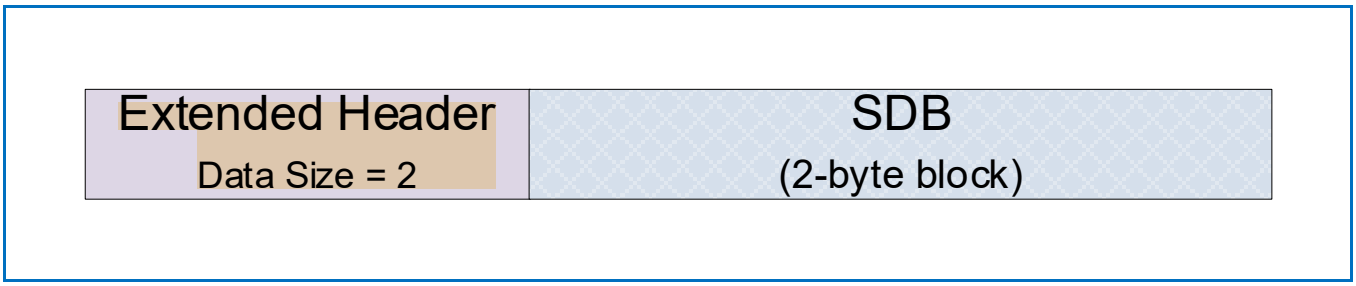


Table 6.57 "SOP'/SOP'' Status Data Block (SDB)"

Offset (Byte)	Field	Value	Description
0	Internal Temp	Unsigned Int	Cable Plug's internal temperature in °C. • 0 = feature not supported • 1 = temperature is less than 2°C. • 2…255 = temperature in °C.
1	Flags	Bit field	**Bit** / **Description** 0 – Thermal Shutdown 1…7 – ***Reserved*** and ***Shall*** be set to zero

6.5.2.2.1 Internal Temp Field

The Internal Temp field reports the instantaneous temperature of the plug in °C. The internal temperature ***Shall*** be monotonic. The Cable Plug ***Shall*** report its internal temperature every ***tACTempUpdate***.

6.5.2.2.2 Thermal Shutdown Field

The Thermal Shutdown flag ***Shall*** also be set when the plug's internal temperature exceeds the Internal Maximum Temperature reported in the *Active Cable* VDO. Once this bit has been set, it ***Shall*** remain set and the plug ***Shall*** remain in Thermal Shutdown until there is a Hard Reset or the *Active Cable*'s power is removed. The Thermal Shutdown flag ***Shall Not*** be cleared by a Cable Reset.

6.5.3 Get_Battery_Cap Message

The ***Get_Battery_Cap*** (Get Battery Capabilities) Message is used to request the capability of a Battery present in its Port Partner. The Port ***Shall*** respond by returning a ***Battery_Capabilities*** Message (see ***Section 6.5.5 "Battery_Capabilities Message"***) containing a Battery Capabilities Data Block (BCDB) for the targeted Battery.

The ***Get_Battery_Cap*** Message contains a 1-byte Get Battery Cap Data Block (GBCDB), whose format ***Shall*** be as shown in ***Figure 6-39 "Get_Battery_Cap Message"*** and ***Table 6.58 "Get Battery Cap Data Block (GBCDB)"***. This block defines for which Battery the request is being made.

The ***Data Size*** field in the ***Get_Battery_Cap*** Message ***Shall*** be set to 1.

Figure 6-39 "Get_Battery_Cap Message"

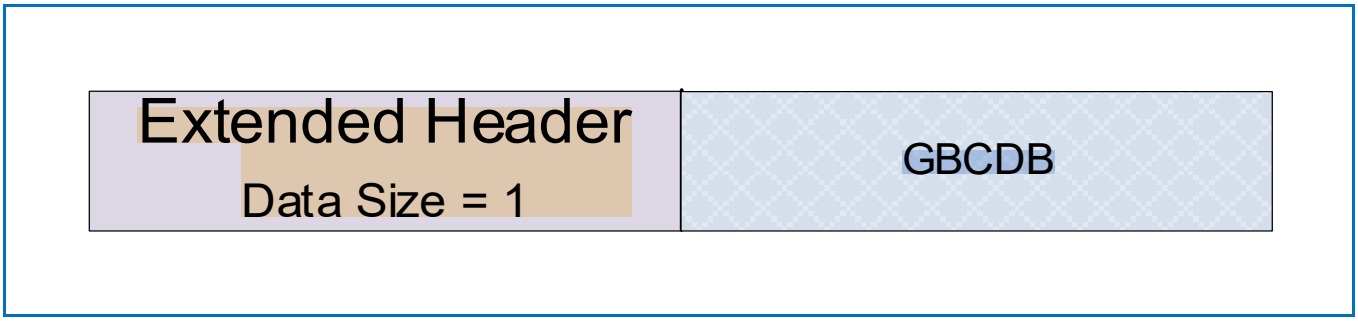


Table 6.58 "Get Battery Cap Data Block (GBCDB)"

Offset	Field	Description
0	***Battery Cap Ref***	Number of the Battery indexed from zero: • Values 0…3 represent the Fixed Batteries. • Values 4…7 represent the Hot Swappable Batteries. • Values 8…255 are ***Reserved*** and ***Shall Not*** be used.

6.5.4 Get_Battery_Status Message

The ***Get_Battery_Status*** (Get Battery Status) Message is used to request the status of a Battery present in its Port Partner. The port ***Shall*** respond by returning a ***Battery_Status*** Message (see ***Section 6.4.5 "Battery_Status Message"***) containing a Battery Status Data Object (BSDO) for the targeted Battery.

The ***Get_Battery_Status*** Message contains a 1-byte Get Battery Status Data Block (GBSDB) whose format ***Shall*** be as shown in ***Figure 6-40 "Get_Battery_Status Message"*** and ***Table 6.59 "Get Battery Status Data Block (GBSDB)"***. This block contains details of the requested Battery. The ***Data Size*** field in the ***Get_Battery_Status*** Message ***Shall*** be set to 1.

Figure 6-40 "Get_Battery_Status Message"

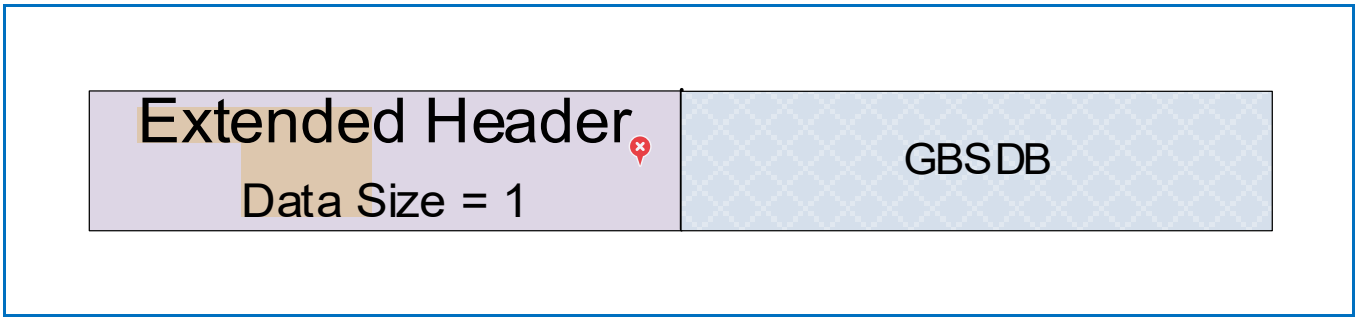


Table 6.59 "Get Battery Status Data Block (GBSDB)"

Offset	Field	Description
0	***Battery Status Ref***	Number of the Battery indexed from zero: • Values 0…3 represent the Fixed Batteries. • Values 4…7 represent the Hot Swappable Batteries. • Values 8…255 are ***Reserved*** and ***Shall Not*** be used.

6.5.5 Battery_Capabilities Message

The ***Battery_Capabilities*** Message is sent in response to a ***Get_Battery_Cap*** Message. The ***Battery_Capabilities*** Message contains one Battery Capability Data Block (BCDB) for one of the Batteries its supports as reported by Battery field in the ***Source_Capabilities_Extended*** Message. The returned BCDB ***Shall*** correspond to the Battery requested in the ***Battery Cap Ref*** field contained in the ***Get_Battery_Cap*** Message.

The ***Battery_Capabilities*** Message returns a 9-byte BCDB whose format ***Shall*** be as shown in ***Figure 6-41 "Battery_Capabilities Message"*** and ***Table 6.60 "Battery Capability Data Block (BCDB)"***.

Figure 6-41 "Battery_Capabilities Message"

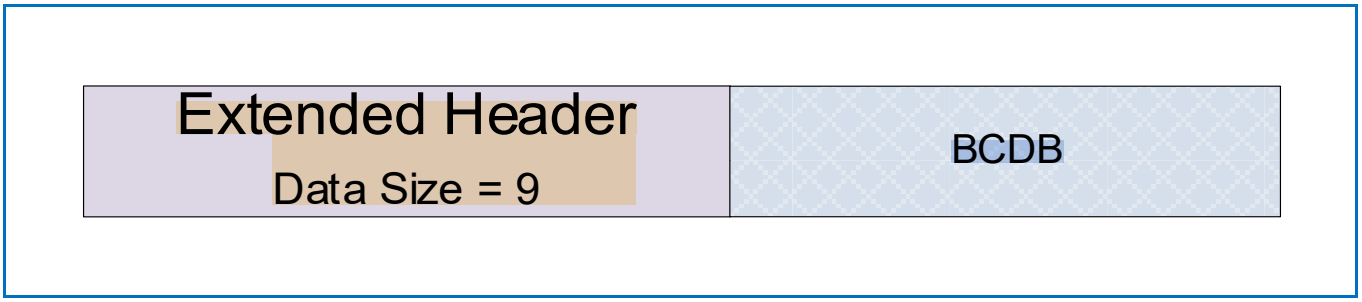


Table 6.60 "Battery Capability Data Block (BCDB)"

Offset (Byte)	Field	Description
0	VID	Vendor ID (assigned by the USB-IF)
2	PID	Product ID (assigned by the manufacturer)
4	Battery Design Capacity	Battery's design capacity in 0.1 WH **Note:** 0x0000 = Battery not present 0xFFFF = design capacity unknown
6	Battery Last Full Charge Capacity	Battery's last full charge capacity in 0.1 WH **Note:** 0x0000 = Battery not present 0xFFFF = last full charge capacity unknown
8	Battery Type	

Bit	Description
0	***Invalid*** Battery reference
1-7	***Reserved***

6.5.5.1 Vendor ID (VID)

The VID field ***Shall*** contain the manufacturer VID associated with the Battery, as assigned by the USB-IF, or 0xFFFF in the case that no such VID exists.

If the Battery Cap Ref field in the ***Get_Battery_Cap*** Message is ***Invalid***, this VID field ***Shall*** be 0xFFFF.

6.5.5.2 Product ID (PID)

The following rules apply to the PID field. When the VID:

- Belongs to the Battery vendor the PID field ***Shall*** contain the Battery's 16-bit product identifier designated by the Battery vendor.
- Belongs to the Device vendor the PID field ***Shall*** contain the Battery's 16-bit product identifier designated by the Device vendor.
- Is 0xFFFF the PID field ***Shall*** be set to 0x0000.

6.5.5.3 Battery Design Capacity Field

The Battery Design Capacity field ***Shall*** return the Battery's design capacity in tenths of WH. If the Battery is Hot Swappable and is not present, the Battery Design Capacity field ***Shall*** be set to zero. If the Battery is unable to report its Design Capacity, it ***Shall*** return 0xFFFF.

6.5.5.4 Battery Last Full Charge Capacity Field

The Battery Last Full Charge Capacity field ***Shall*** return the Battery's last full charge capacity in tenths of WH. If the Battery is Hot Swappable and is not present, the Battery Last Full Charge Capacity field ***Shall*** be set to zero. If the Battery is unable to report its Design Capacity, the Battery Last Full Charge Capacity field ***Shall*** be set to 0xFFFF.

6.5.5.5 Battery Type Field

The Battery Type Field is used to report additional information about the Battery's capabilities.

6.5.5.5.1 Invalid Battery Reference

The ***Invalid*** Battery Reference bit ***Shall*** be set when the ***Get_Battery_Cap*** Message contains a reference to a Battery that does not exist.

6.5.6 Get_Manufacturer_Info Message

The ***Get_Manufacturer_Info*** (Get Manufacturer Info) Message is sent by a Port to request manufacturer specific information relating to its Port Partner, Cable Plug or of a Battery behind a Port. The Port ***Shall*** respond by returning a ***Manufacturer_Info*** Message (*Section 6.5.7 "Manufacturer_Info Message"*) containing a Manufacturer Info Data Block (MIDB). Support for this feature by the Cable Plug is ***Optional Normative.***

The ***Get_Manufacturer_Info*** Message contains a 2-byte Get Manufacturer Info Data Block (GMIDB). This block defines whether it is the Device or Battery manufacturer information being requested and for which Battery the request is being made.

The ***Get_Manufacturer_Info*** Message returns a GMIDB whose format ***Shall*** be as shown in *Figure 6-42 "Get_Manufacturer_Info Message"* and *Table 6.61 "Get Manufacturer Info Data Block (GMIDB)"*.

Figure 6-42 "Get_Manufacturer_Info Message"

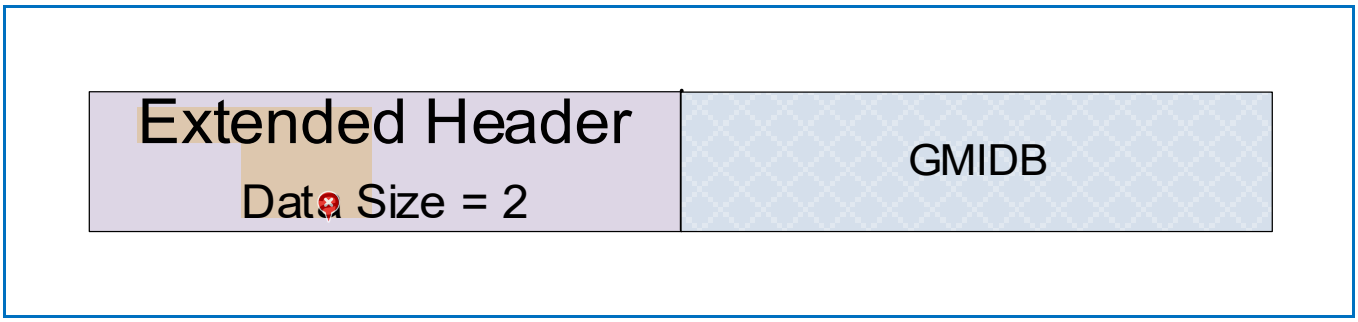


Table 6.61 "Get Manufacturer Info Data Block (GMIDB)"

Offset	Field	Description
0	***Manufacturer Info Target***	0: Port/Cable Plug 1: Battery 255…2: ***Reserved, Shall Not*** be used.
1	***Manufacturer Info Ref***	If ***Manufacturer Info Target*** subfield is Battery (01b) the ***Manufacturer Info Ref*** field Shall contain the Battery number reference which is the number of the Battery indexed from zero: Values 0…3 represent the Fixed Batteries. Values 4…7 represent the Hot Swappable Batteries. Otherwise, this field is ***Reserved*** and ***Shall*** be set to zero.

6.5.7 Manufacturer_Info Message

The *Manufacturer_Info* Message ***Shall*** be sent in response to a *Get_Manufacturer_Info* Message. The *Manufacturer_Info* Message contains the USB VID and the Vendor's PID to identify the device or Battery and the device or Battery's manufacturer byte array in a variable length Data Block of up to *MaxExtendedMsgLegacyLen* .

The *Manufacturer_Info* Message returns a Manufacturer Info Data Block (MIDB) whose format ***Shall*** be as shown in *Figure 6-43 "Manufacturer_Info Message"* and *Table 6.62 "Manufacturer Info Data Block (MIDB)"*.

Figure 6-43 "Manufacturer_Info Message"

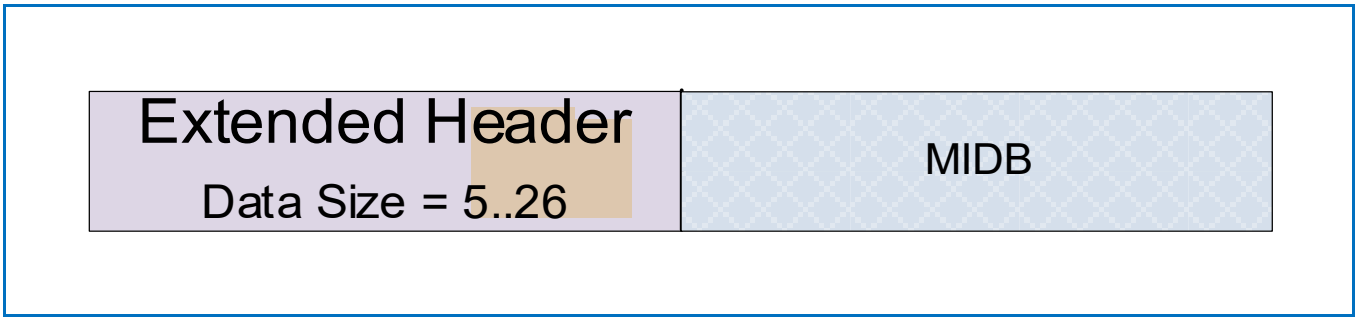


Table 6.62 "Manufacturer Info Data Block (MIDB)"

Offset	Field	Description
0	VID	Vendor ID (assigned by the USB-IF)
2	PID	Product ID (assigned by the manufacturer)
4	Manufacturer String	Vendor defined null terminated string of 0…21 characters. If the Manufacturer Info Target field or Manufacturer Info Ref field in the *Get_Manufacturer_Info* Message is unrecognized the field ***Shall*** return a null terminated ascii text string "Not Supported".

6.5.7.1 Vendor ID (VID)

If the requested Manufacturer Info is associated with the Device, the VID field ***Shall*** contain:

- The manufacturer's VID associated with the Device, as defined by the USB-IF, or
- 0xFFFF in the case that the vendor does not have a VID.

If the requested Manufacturer Info is associated with a Device that has a USB data interface, the Device ***Shall*** report the same VID as the idVendor in the Standard Device Descriptor (see *[USB 2.0]* and *[USB 3.2]*).

If the requested Manufacturer Info is associated with a Battery, the VID field ***Shall*** contain:

- The manufacturer VID associated with the Battery specified, as defined by the USB-IF, or
- 0xFFFF in the case that the vendor does not have a VID.

If the *Manufacturer Info Target* field in the *Get_Manufacturer_Info* Message:

- Is ***Invalid***, this VID field ***Shall*** be 0xFFFF.
- Is Battery (01b) and the *Manufacturer Info Ref* field is ***Invalid***, this VID field ***Shall*** be 0xFFFF.

6.5.7.2 Product ID (PID)

If the VID is 0xFFFF, the PID field ***Shall*** contain 0x0000.

Otherwise:

- If the requested Manufacturer Info is associated with the Device, the PID field ***Shall*** contain the Device's 16-bit product identifier designated by the Device vendor.
- If the requested Manufacturer Info is associated with a Battery:
  - And the VID belongs to the Battery vendor, the PID field ***Shall*** contain the Battery's 16-bit product identifier designated by the Battery vendor.
  - And the VID belongs to the Device vendor, the PID field ***Shall*** contain the Battery's 16-bit product identifier designated by the Device vendor.

6.5.7.3 Manufacturer String

This field ***Shall*** contain the devices or Battery's manufacturer string as defined by the vendor.

If the ***Manufacturer Info Target*** field or ***Manufacturer Info Ref*** field in the ***Get_Manufacturer_Info*** Message is unrecognized the field ***Shall*** return a null terminated ascii text string "Not Supported".

6.5.8 Security Messages

The authentication process between Port Partners or a Port and Cable Plug is fully described in *[USBTypeCAuthentication 1.0]*. This specification describes two Extended Messages used by the authentication process when applied to PD.

In the authentication process described in *[USBTypeCAuthentication 1.0]* there are three basic exchanges that serve to:

- Get the Port or Cable Plug's certificates.
- Get the Port or Cable Plug's digest.
- Challenge the Port Partner or Cable Plug.

Certificates are used to convey information, attested to by a signer, which attests to the Port Partner's or Cable Plug's authenticity. The Port's or Cable Plug's certificates are needed when a Port encounters a Port Partner or Cable Plug it has not been Attached to before. To minimize calculations after the initial Attachment, a Port can also use a digest consisting of hashes of the certificates rather than the certificates themselves. Once the port has the certificates and has calculated the hashes, it stores the hashes and uses the digest in future exchanges. After the port gets the certificates or digest, it challenges its Port Partner or the Cable Plug to detect replay attacks.

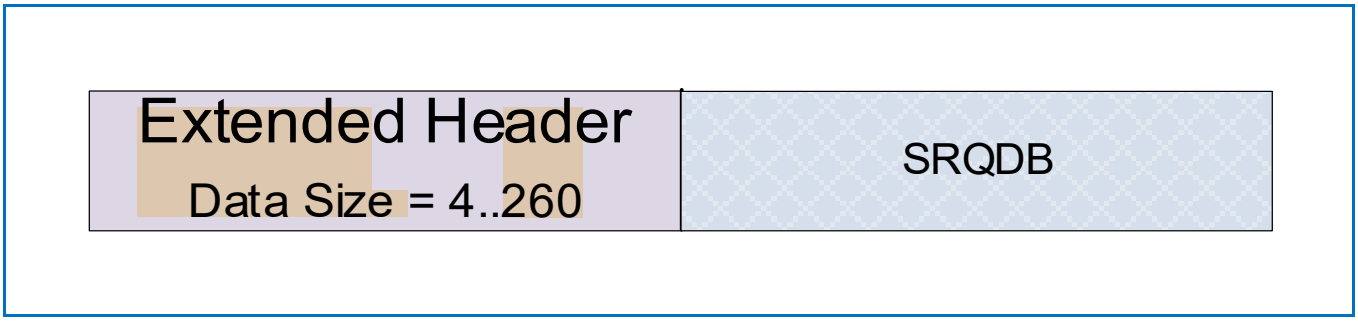
For further details refer to *[USBTypeCAuthentication 1.0]*.

6.5.8.1 Security_Request

The *Security_Request* Message is used by a Port to pass a security data structure to its Port Partner or a Cable Plug.

The *Security_Request* Message contains a Security Request Data Block (SRQDB) whose format ***Shall*** be as shown in *Figure 6-44 "Security_Request Message"*. The contents of the SRQDB and its use are defined in *[USBTypeCAuthentication 1.0]*.

Figure 6-44 "Security_Request Message"

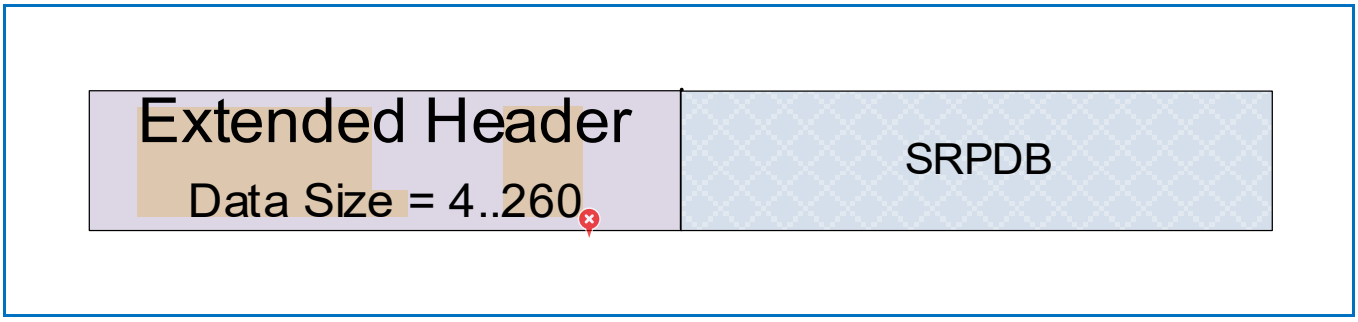


6.5.8.2 Security_Response

The *Security_Response* Message is used by a Port or Cable Plug to pass a security data structure to the Port that sent the *Security_Request* Message.

The *Security_Response* Message contains a Security Response Data Block (SRPDB) whose format ***Shall*** be as shown in *Figure 6-45 "Security_Response Message"*. The contents of the SRPDB and its use are defined in *[USBTypeCAuthentication 1.0]*.

Figure 6-45 “Security_Response Message”



6.5.9 Firmware Update Messages

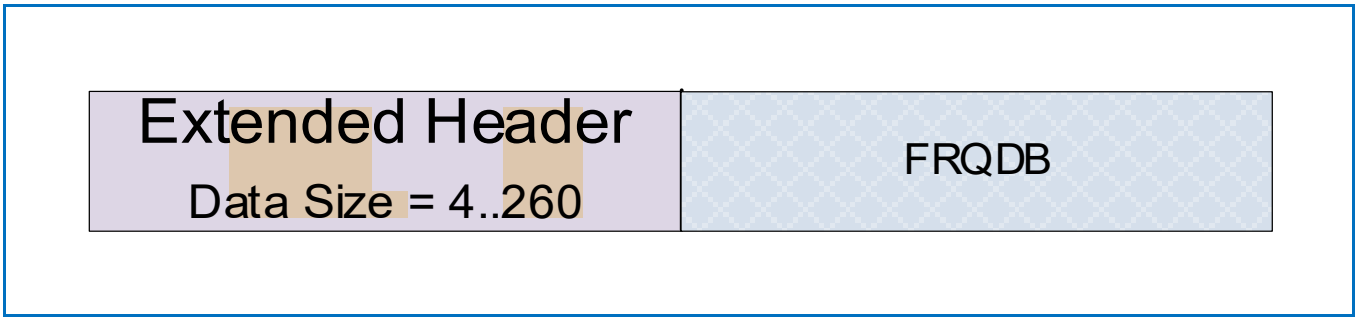
The firmware update process between Port Partners or a Port and Cable Plug is fully described in *[USBPDFirmwareUpdate 1.0]*. This specification describes two Extended Messages used by the firmware update process when applied to PD.

6.5.9.1 Firmware_Update_Request

The ***Firmware_Update_Request*** Message is used by a Port to pass a firmware update data structure to its Port Partner or a Cable Plug.

The ***Firmware_Update_Request*** Message contains a Firmware Update Request Data Block (FRQDB) whose format ***Shall*** be as shown in *Figure 6-46 "Firmware_Update_Request Message"*. The contents of the FRQDB and its use are defined in *[USBPDFirmwareUpdate 1.0]*.

Figure 6-46 "Firmware_Update_Request Message"

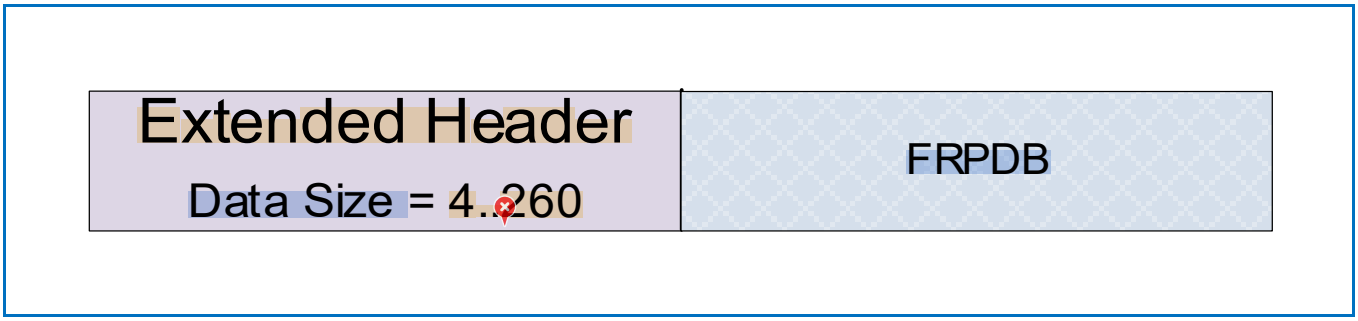


6.5.9.2 Firmware_Update_Response

The ***Firmware_Update_Response*** Message is used by a Port or Cable Plug to pass a firmware update data structure to the Port that sent the ***Firmware_Update_Request*** Message.

The ***Firmware_Update_Response*** Message contains a Firmware Update Response Data Block (FRPDB) whose format ***Shall*** be as shown in *Figure 6-47 "Firmware_Update_Response Message"*. The contents of the FRPDB and its use are defined in *[USBPDFirmwareUpdate 1.0]*.

Figure 6-47 "Firmware_Update_Response Message"



6.5.10 PPS_Status Message

The ***PPS_Status*** Message ***Shall*** be sent in response to a ***Get_PPS_Status*** Message. The ***PPS_Status*** Message enables a Sink to query the Source to get additional information about its operational state. The ***Get_PPS_Status*** Message and the ***PPS_Status*** Message ***Shall*** only be supported when the ***Alert*** Message is also supported.

The ***PPS_Status*** Message ***Shall*** return a 4-byte PPS Status Data Block (PPSSDB) whose format ***Shall*** be as shown in ***Figure 6-48 "PPS_Status Message"*** and ***Table 6.63 "PPS Status Data Block (PPSSDB)"***.

Figure 6-48 "PPS_Status Message"

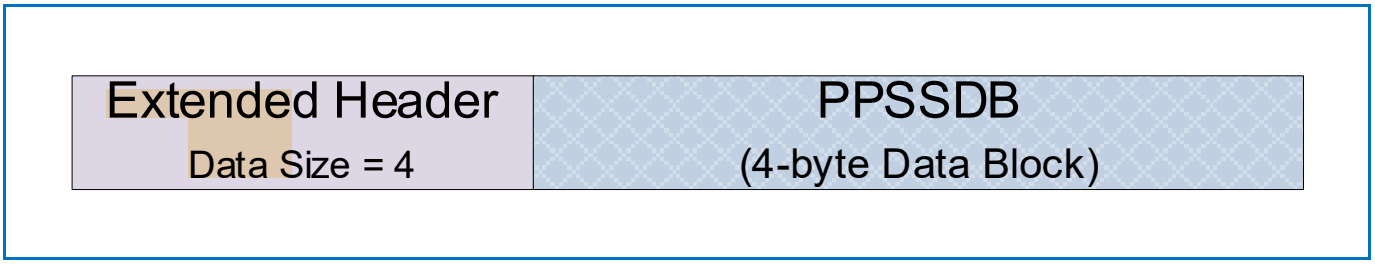


Table 6.63 "PPS Status Data Block (PPSSDB)"

Offset	Field	Size	Description
0	Output Voltage	2	Source's output Voltage in 20mV units. When set to 0xFFFF, the Source does not support this field.
2	Output Current	1	Source's output current in 50mA units. When set to 0xFF, the Source does not support this field.
3	Real Time Flags	1	**Bit** / **Description** 0 – ***Reserved*** and ***Shall*** be set to zero 1…2 – PTF: 00 – Not Supported PTF: 01 – Normal PTF: 10 – Warning PTF: 11 – Over temperature 3 – OMF (Operating Mode Flag) OMF is set when operating in Current Limit mode and cleared when operating in Constant Voltage mode. 4…7 – ***Reserved*** and ***Shall*** be set to zero

6.5.10.1 Output Voltage Field

The Output Voltage field ***Shall*** return the Source's output Voltage at the time of the request. The output Voltage is measured either at the Source's receptacle or, if the Source has a captive cable, where the Voltage is applied to the cable.

The measurement accuracy ***Shall*** be +/-3% rounded to the nearest 20mV in SPR PPS Mode.

If the Source does not support the Output Voltage field, the field ***Shall*** be set to 0xFFFF.

6.5.10.2 Output Current Field

The Output Current field ***Shall*** return the Source's output current at the time of the request measured at the Source's receptacle.

The measurement accuracy ***Shall*** be +/-150mA.

If the Source does not support the Output Current field, the field ***Shall*** be set to 0xFF.

6.5.10.3 Real Time Flags Field

Real Time flags provide a real-time indication of the Source's operating state:

- The PTF (Present Temperature Flag) ***Shall*** provide a real-time indication of the Source's internal thermal status. If the PTF is not supported, it will be set to zero.
- Normal indicates that that the Source is operating within its normal thermal envelope.
- Warning indicates that the Source is over-heating but is not in imminent danger of shutting down.
- Over Temperature indicates that the Source is over heated and will shut down soon or has already shutdown and has sent an OTP in an ***Alert*** Message.
- The OMF (Operating Mode Flag) ***Shall*** provide a real-time indication of the SPR PPS Source's operating mode. When set, the Source is operating in Current Limit mode; when cleared it is operating Constant Voltage mode. This bit ***Shall*** be set to zero when not in SPR PPS Mode.

6.5.11 Country_Codes Message

The *Country_Codes* Message ***Shall*** be sent in response to a *Get_Country_Codes* Message. The *Country_Codes* Message enables a Port to query its Port partner to get a list of alpha-2 country codes as defined in *[ISO 3166]* for which the Port Partner has country specific information.

The Country_Codes Message ***Shall*** contain a 4…26-byte Country Code Data Block (CCDB) whose format ***Shall*** be as shown in *Figure 6-49 "Country_Codes Message"* and *Table 6.64 "Country Codes Data Block (CCDB)"*.

Figure 6-49 "Country_Codes Message"

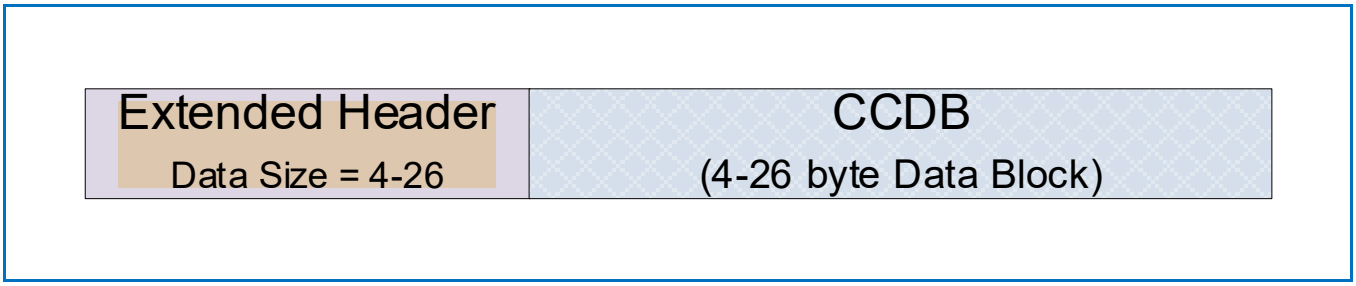


Table 6.64 "Country Codes Data Block (CCDB)"

Offset	Field	Description
0	Length	Number of country codes in the message
1	Reserved	***Shall*** be set to zero.
2	1st Country Code	First character of the Alpha-2 Country Code defined by *[ISO 3166]*
3		Second character of the Alpha-2 Country Code defined by *[ISO 3166]*
4	2nd Country Code	First character of the Alpha-2 Country Code defined by *[ISO 3166]*
5		Second character of the Alpha-2 Country Code defined by *[ISO 3166]*
	…	
Length * 2n	nth Country Code	

6.5.11.1 Country Code Field

The Country Code field ***Shall*** contain the Alpha-2 Country Code defined by *[ISO 3166]*.

6.5.12 Country_Info Message

The *Country_Info* Message ***Shall*** be sent in response to a *Get_Country_Info* Message. The *Country_Info* Message enables a Port to get additional country specific information from its Port Partner.

The *Country_Info* Message ***Shall*** contain a 4…26-byte Country Info Data Block (CIDB) whose format ***Shall*** be as shown in *Figure 6-50 "Country_Info Message"* and *Table 6.65 "Country Info Data Block (CIDB)"*.

Figure 6-50 "Country_Info Message"

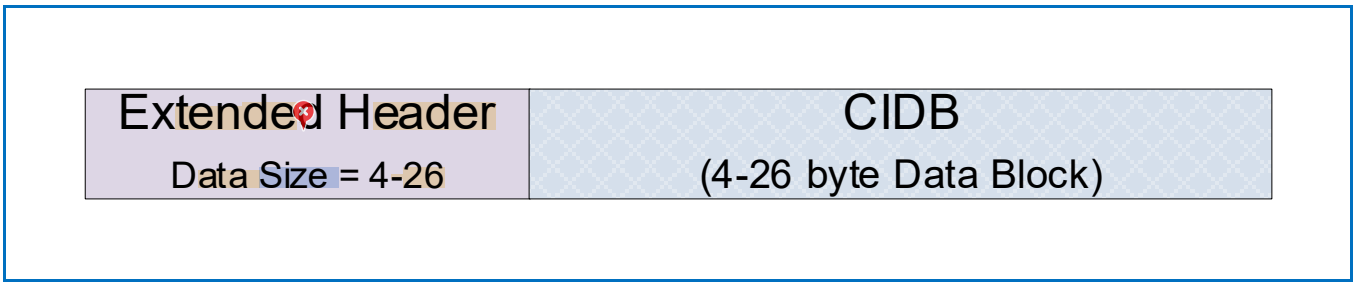


Table 6.65 "Country Info Data Block (CIDB)"

Offset	Field	Size
0	Country Code	First character of the Alpha-2 Country Code received in the corresponding *Get_Country_Info* Message.
1		Second character of the Alpha-2 Country Code received in the corresponding *Get_Country_Info* Message
2…3	Reserved	***Shall*** be set to zero.
4	Country Specific Data	1…22 bytes of content defined by the country's authority.

6.5.12.1 Country Code Field

The Country Code field ***Shall*** contain the Alpha-2 Country Code received in the corresponding *Get_Country_Info* Message.

6.5.12.2 Country Specific Data Field

The Country Specific Data field ***Shall*** contain content defined by and formatted in a manner determined by an official agency of the country indicated in the Country Code field.

If the Country Code field in the *Get_Country_Info* Message is unrecognized then Country Specific Data field ***Shall*** return the null terminated ascii text string "UnsupportedCode".

6.5.13 Sink_Capabilities_Extended Message

The *Sink_Capabilities_Extended* Message ***Shall*** be sent in response to a *Get_Sink_Cap_Extended* Message. The *Sink_Capabilities_Extended* Message enables a Sink or a DRP to inform the Source about its capabilities as a Sink.

The *Sink_Capabilities_Extended* Message ***Shall*** return a 24-byte Sink Capabilities Extended Data Block (SKEDB) whose format ***Shall*** be as shown in *Figure 6-51 "Sink_Capabilities_Extended Message"* and *Table 6.66 "Sink Capabilities Extended Data Block (SKEDB)"*.

Figure 6-51 "Sink_Capabilities_Extended Message"

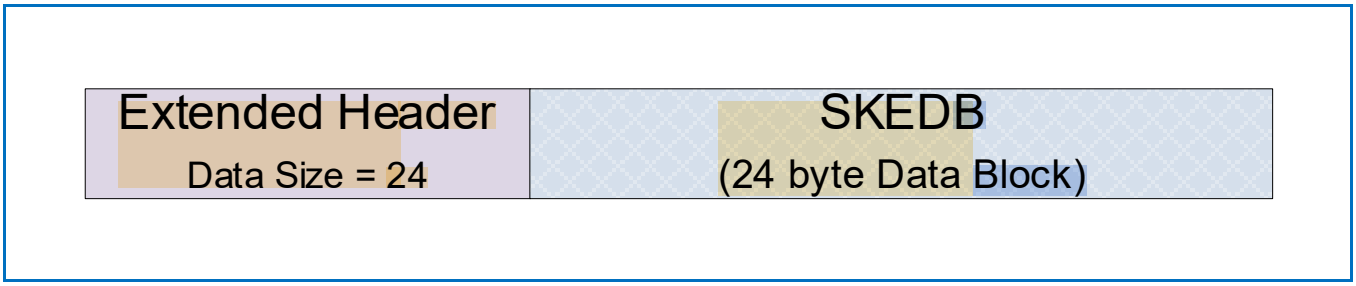


Table 6.66 "Sink Capabilities Extended Data Block (SKEDB)"

<table>
<tr><th>Offset (Byte)</th><th>Field</th><th>Value</th><th>Description</th><th>Offset (Byte)</th></tr>
<tr><td>0</td><td>VID</td><td>2</td><td>Numeric</td><td>Vendor ID (assigned by the USB-IF)</td></tr>
<tr><td>2</td><td>PID</td><td>2</td><td>Numeric</td><td>Product ID (assigned by the manufacturer)</td></tr>
<tr><td>4</td><td>XID</td><td>4</td><td>Numeric</td><td>Value provided by the USB-IF assigned to the product</td></tr>
<tr><td>8</td><td>FW Version</td><td>1</td><td>Numeric</td><td>Firmware version number</td></tr>
<tr><td>9</td><td>HW Version</td><td>1</td><td>Numeric</td><td>Hardware version number</td></tr>
<tr><td>10</td><td>SKEDB Version</td><td>1</td><td>Numeric</td><td>SKEDB Version (not the specification Version):
Version 1.0 = 1
Values 0 and 2-255 are Reserved and Shall Not be used</td></tr>
<tr><td>11</td><td>Load Step</td><td>1</td><td>Bit Field</td><td><table><tr><th>Bit</th><th>Description</th></tr><tr><td>1...0</td><td>00b: 150mA/µs Load Step (default)
01b: 500mA/µs Load Step
11b...10b: Reserved and Shall Not be used</td></tr><tr><td>2...7</td><td>Reserved and Shall be set to zero</td></tr></table></td></tr>
<tr><td>12</td><td>Sink Load Characteristics</td><td>2</td><td>Bit field</td><td><table><tr><th>Bit</th><th>Description</th></tr><tr><td>0...4</td><td>Percent overload in 10% increments
Values higher than 25 (11001b) are clipped to 250%. 00000b is the default.</td></tr><tr><td>5...10</td><td>Overload period in 20ms when bits 0-4 non-zero.</td></tr><tr><td>11.14</td><td>Duty cycle in 5% increments when bits 0-4 are non-zero</td></tr><tr><td>15</td><td>Can tolerate V_{BUS} Voltage droop</td></tr></table></td></tr>
<tr><td>14</td><td>Compliance</td><td>1</td><td>Bit Field</td><td><table><tr><th>Bit</th><th>Description</th></tr><tr><td>0</td><td>Requires LPS Source when set</td></tr><tr><td>1</td><td>Requires PS1 Source when set</td></tr><tr><td>2</td><td>Requires PS2 Source when set</td></tr><tr><td>3...7</td><td>Reserved and Shall be set to zero</td></tr></table></td></tr>
</table>

<table>
<tr><td>15</td><td>Touch Temp</td><td>1</td><td>Value</td><td>Temperature conforms to:
0 = Not applicable
1 = [IEC 60950-1] (default)
2 = [IEC 62368-1] TS1
3 = [IEC 62368-1] TS2
Note: All other values Reserved</td></tr>
<tr><td>16</td><td>Battery Info</td><td>1</td><td>Byte</td><td>Upper Nibble = Number of Hot Swappable Battery Slots (0…4)
Lower Nibble = Number of Fixed Batteries (0…4)</td></tr>
<tr><td>17</td><td>Sink Modes</td><td>1</td><td>Bit field</td><td><table><tr><th>Bit</th><th>Description</th></tr><tr><td>0</td><td>1: PPS charging supported</td></tr><tr><td>1</td><td>1: V_{BUS} powered</td></tr><tr><td>2</td><td>1: Mains powered</td></tr><tr><td>3</td><td>1: Battery powered</td></tr><tr><td>4</td><td>1: Battery essentially unlimited</td></tr><tr><td>5</td><td>1: AVS Supported</td></tr><tr><td>6…7</td><td>Reserved and Shall be set to zero</td></tr></table></td></tr>
<tr><td>18</td><td>Sink Minimum PDP</td><td>1</td><td>Byte</td><td><table><tr><th>Bit</th><th>Description</th></tr><tr><td>0…6</td><td>The Minimum PDP required by the Sink to operate without consuming any power from its Battery(s) if it has one.</td></tr><tr><td>7</td><td>Reserved and Shall be set to zero</td></tr></table></td></tr>
<tr><td>19</td><td>Sink Operational PDP</td><td>1</td><td>Byte</td><td><table><tr><th>Bit</th><th>Description</th></tr><tr><td>0…6</td><td>The PDP the Sink requires to operate normally. For Sinks with a Battery, it is the PDP Rating of the charger supplied with it or recommended for it.</td></tr><tr><td>7</td><td>Reserved and Shall be set to zero</td></tr></table></td></tr>
<tr><td>20</td><td>Sink Maximum PDP</td><td>1</td><td>Byte</td><td><table><tr><th>Bit</th><th>Description</th></tr><tr><td>0…6</td><td>The Maximum PDP the Sink can consume to operate and charge its Battery(s) if it has one.</td></tr><tr><td>7</td><td>Reserved and Shall be set to zero</td></tr></table></td></tr>
<tr><td>21</td><td>EPR Sink Minimum PDP</td><td>1</td><td>Byte</td><td>The Minimum PDP required by the EPR Sink to operate without consuming any power from its Battery(s) if it has one.</td></tr>
<tr><td>22</td><td>EPR Sink Operational PDP</td><td>1</td><td>Byte</td><td>The PDP the EPR Sink requires to operate normally. For Sinks with a Battery, it is the PDP Rating of the charger supplied with it or recommended for it.</td></tr>
<tr><td>23</td><td>EPR Sink Maximum PDP</td><td>1</td><td>Byte</td><td>The Maximum PDP the EPR Sink can consume to operate and charge its Battery(s) if it has one.</td></tr>
</table>

6.5.13.1 Vendor ID (VID) Field

The Vendor ID field ***Shall*** contain the 16-bit Vendor ID (VID) assigned to the Sink's vendor by the USB-IF. If the vendor does not have a VID, the Vendor ID field ***Shall*** be set to 0xFFFF. Devices that have a USB data interface ***Shall*** report the same VID as the idVendor in the Standard Device Descriptor (see ***[USB 2.0]*** and ***[USB 3.2]***).

6.5.13.2 Product ID (PID) Field

The Product ID field ***Shall*** contain the 16-bit Product ID (PID) assigned by the Sink's vendor. Devices that have a USB data interface ***Shall*** report the same PID as the idProduct in the Standard Device Descriptor (see ***[USB 2.0]*** and ***[USB 3.2]***).

6.5.13.3 XID Field

The XID field ***Shall*** contain the 32-bit XID provided by the USB-IF to the vendor who in turns assigns it to a product. If the vendor does not have an XID, then it ***Shall*** return zero in this field (see ***[USB 2.0]*** and ***[USB 3.2]***).

6.5.13.4 Firmware Version Field

The Firmware Version field ***Shall*** contain an 8-bit firmware version number assigned to the device by the vendor.

6.5.13.5 Hardware Version Field

The Hardware Version field ***Shall*** contain an 8-bit hardware version number assigned to the device by the vendor.

6.5.13.6 SKEDB Version Field

The SKEDB version field contains the version level of the SKEDB. Currently only Version 1 is defined.

6.5.13.7 Load Step Field

The Load Step field contains bits indicating the Load Step Slew Rate and Magnitude that this Sink prefers. See ***Section 7.1.12.1 "Voltage Regulation Field"*** for further details.

6.5.13.8 Sink Load Characteristics Field

The Sink ***Shall*** report its preferred load characteristics. Regardless of this value, in operation its load ***Shall Not*** exceed the capabilities reported in the ***Source_Capabilities_Extended*** message.

6.5.13.9 Compliance Field

The Compliance field ***Shall*** contain the types of Sources the Sink has been tested and certified with (see ***Section 7.1.12.3 "Compliance Field"***).

6.5.13.10 Touch Temp

The Touch Temp field ***Shall*** report the IEC standard used to determine the surface temperature of the Sink's enclosure. Safety limits for the Sink's touch temperature are set in applicable product safety standards (e.g., ***[IEC 60950-1]*** or ***[IEC 62368-1]***). The Sink ***May*** report when its touch temperature performance conforms to the TS1 or TS2 limits described in ***[IEC 62368-1]***.

6.5.13.11 Battery Info

The Batteries Info field ***Shall*** report the number of Fixed Batteries and Hot Swappable Battery Slots the Sink supports. This field ***Shall*** independently report the number of Battery Slots and the number of Fixed Batteries. The

information reported in the Battery Info field ***Shall*** match that reported in the Battery Info field of the ***Source_Capabilities_Extended*** Message.

A Sink ***Shall*** have no more than 4 Fixed Batteries and no more than 4 Battery Slots.

Fixed Batteries ***Shall*** be numbered consecutively from 0 to 3. The number assigned to a given Fixed Battery ***Shall Not*** change between Attach and Detach.

Battery Slots ***Shall*** be numbered consecutively from 4 to 7. The number assigned to a given Battery Slot ***Shall Not*** change between Attach and Detach.

6.5.13.12 Sink Modes

The Sink Modes bit field ***Shall*** identify the charging capabilities and the power sources that can be used by the Sink. When bit 0 is set, the Sink has the ability to use a PPS Source for fast charging.

The source of power a Sink can use:

- When bit 1 is set, the Sink has the ability to be sourced by V_{BUS}.
- When bit 2 is set, the Sink has the ability to be sourced by an external mains power supply.
- When bit 3 is set, the Sink has the ability to be sourced by a battery.
- When bit 4 is set, the Sink has the ability to be sourced by a battery with essentially infinite energy (e.g., a car battery).
- When bit 5 is set, the Sink has the ability to support AVS.

Bits 1-5 ***May*** be set independently of one another. The combination indicates what sources of power the Sink can utilize. For example, some Sinks are only powered by a Battery (e.g., an automobile battery) rather than the more common mains and some Sinks are only powered from V_{BUS} or V_{CONN}.

6.5.13.13 Sink Minimum PDP

The Sink Minimum PDP field ***Shall*** contain the minimum power required by the Sink, rounded up to the next integer, to operate all its functional modes except charging its battery if present. The Sink Minimum PDP field ***Shall*** be less than or equal to the Sink Operational PDP. The value is used by the Source to determine whether or not it has sufficient power to minimally support the attached Sink. If the Sink is EPR capable and is unable to operate at PDPs less than 100W, it ***Shall*** set this field to zero.

6.5.13.14 Sink Operational PDP

The Sink Operational PDP field ***Shall*** contain the manufacturer recommended PDP of the Sink, rounded up to the next integer. This corresponds to the PDP Rating of Sources that the Sink is designed to operate with (See ***Section 10.3.2 "Normative Sink Rules"***). The Sink Operational PDP ***Shall*** be sufficient to operate all the Sink's functional modes normally AND charge the Sink's battery if present. For Sinks with a battery(s), it ***Shall*** correspond to the PDP Rating of the charger shipped with the Sink or the recommended charger's PDP Rating. If the Sink is EPR capable and is unable to operate at PDPs less than 100W, it ***Shall*** set this field to zero.

6.5.13.15 Sink Maximum PDP

The Sink Maximum PDP ***Shall*** be highest amount of power the Sink consumes under any operating condition, rounded up to the next integer, including charging its battery if present. The Sink Maximum PDP field ***Shall Not*** be less than the Sink Operational PDP, but ***May*** be the same. The value is used by the Source to determine the maximum amount of power it has to budget for the attached Sink. If the Sink is EPR capable and is unable to operate at PDPs less than 100W, it ***Shall*** set this field to zero.

6.5.13.16 EPR Sink Minimum PDP

The EPR Sink Minimum PDP field ***Shall*** contain the minimum power required by an EPR Sink, rounded up to the next integer, to operate all its functional modes except charging its battery if present. The EPR Sink Minimum PDP field ***Shall*** be less than or equal to the EPR Sink Operational PDP. The value is used by the Source to determine whether or not it has sufficient power to minimally support the attached Sink. If the Sink is not EPR capable, this field ***Shall*** be set to zero.

6.5.13.17 EPR Sink Operational PDP

The EPR Sink Operational PDP field ***Shall*** contain the manufacturer recommended PDP of the Sink, rounded up to the next integer. This corresponds to the PDP Rating of EPR Sources that the Sink is designed to operate with (See *Section 10.3.2 "Normative Sink Rules"*). The EPR Sink Operational PDP ***Shall*** be sufficient to operate all the Sink's functional modes normally AND charge the Sink's battery if present. For Sinks with a battery(s), it ***Shall*** correspond to the PDP Rating of the charger shipped with the EPR Sink or the recommended charger's PDP Rating. If the Sink is not EPR capable, this field ***Shall*** be set to zero.

6.5.13.18 EPR Sink Maximum PDP

The EPR Sink Maximum PDP ***Shall*** be highest amount of power the EPR Sink consumes under any operating condition, rounded up to the next integer, including charging its battery if present. The EPR Sink Maximum PDP field ***Shall Not*** be less than the EPR Sink Operational PDP, but ***May*** be the same. The value is used by the Source to determine the maximum amount of power it has to budget for the attached Sink. If the Sink is not EPR capable, this field ***Shall*** be set to zero.

6.5.14 Extended_Control Message

The *Extended_Control* Message extends the control message space. The *Extended_Control* Message includes one byte of data. The *Extended_Control* Message ***Shall*** be as shown in *Figure 6-52 "Extended_Control Message"* and *Table 6.67 "Extended Control Data Block (ECDB)"*.

Figure 6-52 "Extended_Control Message"

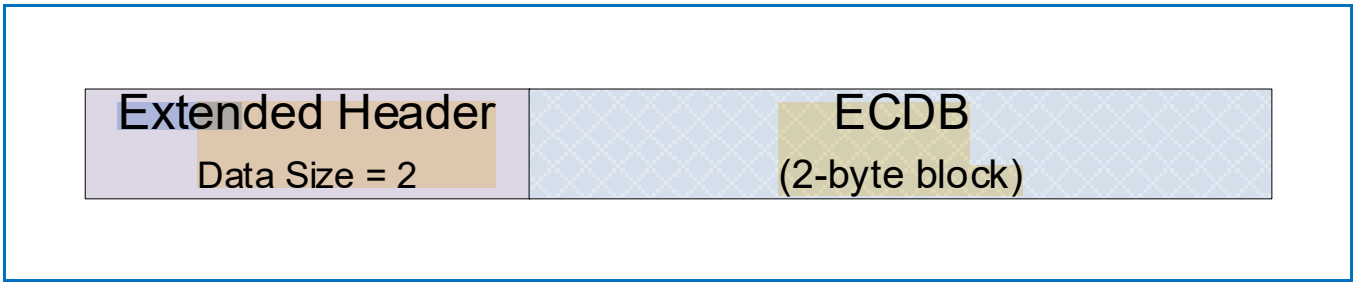


Table 6.67 "Extended Control Data Block (ECDB)"

Offset	Field	Value	Description
0	Type	Unsigned Int	Extended Control Message Type
1	Data	Byte	***Shall*** be set to zero when not used.

The *Extended_Control* Message types are specified in the Type field of the ECDB and are listed in *Table 6.68 "Extended Control Message Types"*. The Sent by column indicates entities which ***May*** send the given Message (Source, Sink or Cable Plug); entities not listed ***Shall Not*** issue the corresponding Message. The "***Valid*** Start of Packet" column indicates the Messages which ***Shall*** only be issued in SOP Packets.

Table 6.68 "Extended Control Message Types"

Type	Data	Message Type	Sent by	Description	Valid Start of Packet
0		***Reserved***	***N/A***	All values not explicitly defined are ***Reserved*** and ***Shall Not*** be used.	
1	Not used	*EPR_Get_Source_Cap*	Sink or DRP	See *Section 6.5.14.1*	SOP only
2	Not used	*EPR_Get_Sink_Cap*	Source or DRP	See *Section 6.5.14.2*	SOP only
3	Not used	*EPR_KeepAlive*	Sink	See *Section 6.5.14.3*	SOP only
4	Not Used	*EPR_KeepAlive_Ack*	Source	See *Section 6.5.14.4*	SOP only
5-255		***Reserved***	***N/A***	All values not explicitly defined are ***Reserved*** and ***Shall Not*** be used.	

6.5.14.1 EPR_Get_Source_Cap Message

The *EPR_Get_Source_Cap* (EPR Get Source Capabilities) Message ***Shall*** only be sent by a Port capable of operating as a Sink and that supports EPR Mode to request the Source Capabilities and Dual-Role Power capability of its Port Partner. A Port that can operate as an EPR Source ***Shall*** respond by returning an *EPR_Source_Capabilities* Message (see *Section 6.5.15.2 "EPR_Source_Capabilities Message"*). A port that does not support EPR Mode as a Source ***Shall*** return the *Not_Supported* Message.

An EPR Mode capable Sink Port that is operating in SPR Mode ***Shall*** treat the ***EPR_Source_Capabilities*** Message as informational only and ***Shall Not*** respond with a ***EPR_Request*** Message.

6.5.14.2 EPR_Get_Sink_Cap Message

The ***EPR_Get_Sink_Cap*** (EPR Get Sink Capabilities) Message ***Shall*** only be sent by a Port capable of operating as a Source and that supports EPR Mode to request the Sink Capabilities and Dual-Role Power capability of its Port Partner. A Port that is EPR Mode capable operating as a Sink ***Shall*** respond by returning an ***EPR_Sink_Capabilities*** Message (see ***Section 6.5.15.3 "EPR_Sink_Capabilities Message"***). A Port that does not support EPR Mode as a Sink ***Shall*** return the ***Not_Supported*** Message.

6.5.14.3 EPR_KeepAlive Message

The ***EPR_KeepAlive*** Message ***May*** be sent by a Sink operating in EPR Mode to meet the requirement for periodic traffic. The Source operating on EPR Mode responds by returning an ***EPR_KeepAlive_Ack*** Message to the Sink. See ***Section 6.4.9 "EPR_Request Message"*** for additional information.

6.5.14.4 EPR_KeepAlive_Ack Message

The ***EPR_KeepAlive_Ack*** Message ***Shall*** be sent by a Source operating in EPR Mode in response to an ***EPR_KeepAlive*** Message. See ***Section 6.4.9 "EPR_Request Message"*** for additional information.

6.5.15 EPR Capabilities Message

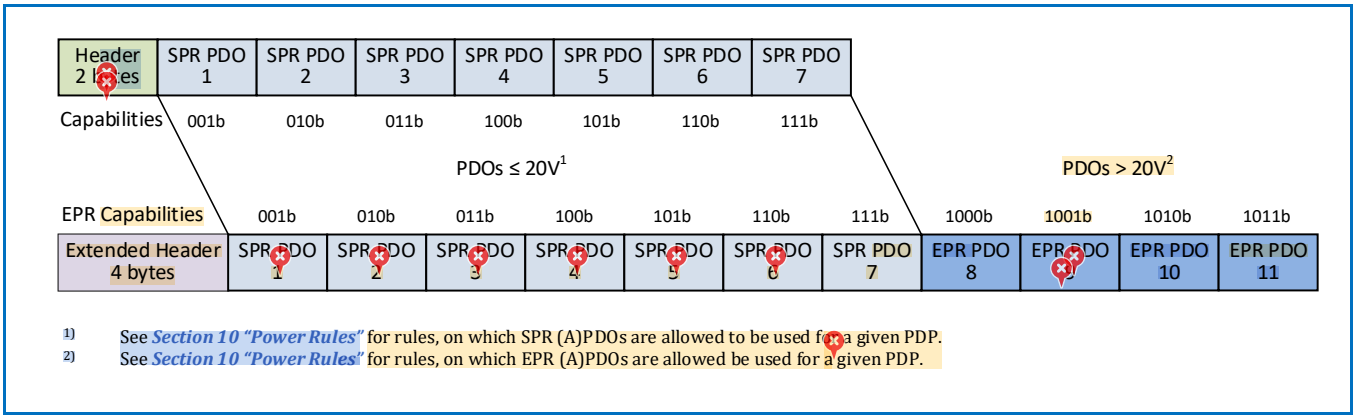
The EPR Capabilities Message is an extended data message made of Power Data Objects (PDO) defined in *Section 6.4.1 "Capabilities Message"*. It is used to form *EPR_Source_Capabilities* Messages and *EPR_Sink_Capabilities* Messages. Sources expose their EPR power capabilities by sending an *EPR_Source_Capabilities* Message. Sinks expose their EPR power requirements by returning an *EPR_Sink_Capabilities* Message when requested. Both are composed of a number of 32-bit Power Data Objects (see *Table 6.7 "Power Data Object"*).

An EPR Capabilities Message ***Shall*** have a 5V Fixed Supply PDO containing the sending Port's information in the first object position followed by up to 10 additional PDOs.

6.5.15.1 EPR Capabilities Message Construction

The EPR Capabilities Messages (*EPR_Source_Capabilities* and *EPR_Sink_Capabilities*) are extended data Messages with the first seven positions filled with the same SPR PDOs returned by the SPR Capabilities Messages (*Source_Capabilities* and *Sink_Capabilities*) followed by the EPR PDOs (see *Section 1.6 "Terms and Abbreviations"*) starting in the eighth position. See *Figure 6-53 "Mapping SPR Capabilities to EPR Capabilities"*.

Figure 6-53 "Mapping SPR Capabilities to EPR Capabilities"



1) See *Section 10 "Power Rules"* for rules, on which SPR (A)PDOs are allowed to be used for a given PDP.
2) See *Section 10 "Power Rules"* for rules, on which EPR (A)PDOs are allowed be used for a given PDP.

Power Data Objects in the EPR Capabilities Messages ***Shall*** be sent in the following order:

1) The SPR PDOs as reported in the SPR Capabilities Message.
2) If the SPR Capabilities Message contains fewer than 7 PDOs, the unused Data Objects ***Shall*** be zero filled.
3) The EPR PDOs as defined in *Section 6.4.1 "Capabilities Message"* ***Shall*** start at object position 8 and ***Shall*** be sent in the following order:
   a) Fixed Supply Objects that offer 28V, 36V or 48V, if present, ***Shall*** be sent in Voltage order; lowest to highest.
   b) One EPR Adjustable Voltage Supply Object ***Shall*** be sent.

6.5.15.2 EPR_Source_Capabilities Message

The *EPR_Source_Capabilities* is an EPR Capabilities message containing a list of Power Data Objects that the EPR Source is capable of supplying. It is sent by an EPR Source in order to convey its capabilities to a Sink. An EPR Source ***Shall*** send the *EPR_Source_Capabilities* message:

- When entering EPR Mode
- While in EPR Modes when its capabilities change

- In response to an ***EPR_Get_Source_Cap*** Message

An EPR Sink operating in EPR Mode ***Shall*** evaluate every ***EPR_Source_Capabilities*** Message it receives and ***Shall*** respond with a ***EPR_Request*** Message. If its power consumption exceeds the Source's capabilities, it ***Shall*** re-negotiate so as not to exceed the Source's most recently Advertised capabilities.

While operating in SPR Mode, an EPR Sink receiving an ***EPR_Source_Capabilities*** message in response to an ***EPR_Get_Source_Cap*** Messages ***Shall Not*** respond with an ***EPR_Request*** Message.

The (A)PDOs in an ***EPR_Source_Capabilities*** Message ***Shall*** only be requested using the ***EPR_Request*** Message and only when in EPR Mode.

6.5.15.3 EPR_Sink_Capabilities Message

The ***EPR_Sink_Capabilities*** is an EPR Capabilities message that contains a list of Power Data Objects that the EPR Sink requires to operate. It is sent by an EPR Sink in order to convey its power requirements to an EPR Source. The EPR Sink ***Shall*** only send the ***EPR_Sink_Capabilities*** message in response to an ***EPR_Get_Sink_Cap*** Message.

6.5.16 Vendor_Defined_Extended Message

The ***Vendor_Defined_Extended*** Message (VDEM) is provided to allow vendors to exchange information outside of that defined by this specification using the extended message format.

A ***Vendor_Defined_Extended*** Message ***Shall*** consist of at least one Vendor Data Object, the VDM Header, and ***May*** contain up to a maximum of 256 additional data bytes.

To ensure vendor uniqueness of ***Vendor_Defined_Extended*** Messages, all ***Vendor_Defined_Extended*** Messages ***Shall*** contain a ***Valid*** USB Standard or Vendor ID (SVID) allocated by USB-IF in the VDM Header.

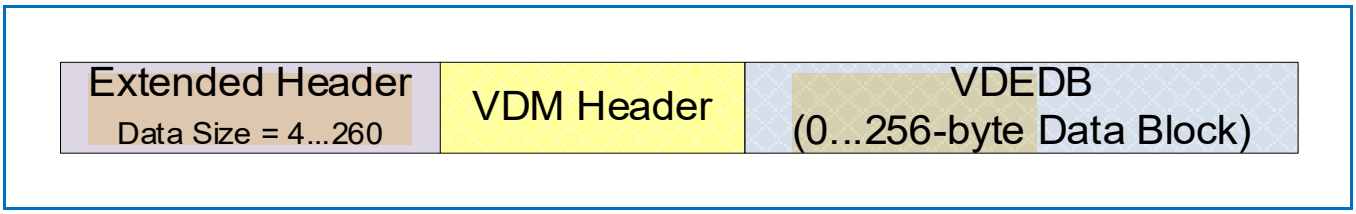
A VDEM does not define any structure and Messages ***May*** be created in any manner that the vendor chooses.

Vendor_Defined_Extended Messages ***Shall Not*** be used for direct power negotiation. They ***May*** however be used to alter Local Policy, affecting what is offered or consumed via the normal PD Messages. For example, a ***Vendor_Defined_Extended*** Message could be used to enable the Source to offer additional power via a ***Source_Capabilities*** Message.

Vendor_Defined_Extended Messages ***Shall Not*** be used where equivalent functionality is contained in the PD Specification e.g., authentication or firmware update.

The Message format ***Shall*** be as shown in ***Figure 6-54 "Vendor_Defined_Extended Message"***.

Figure 6-54 "Vendor_Defined_Extended Message"



The VDM Header ***Shall*** be the first 4-bytes in a Vendor Defined Extended Message. The VDM Header provides command space to allow vendors to customize Messages for their own purposes.

The VDM Header in the VDEM ***Shall*** follow the Unstructured VDM Header format as defined in ***Section 6.4.4.1 "Unstructured VDM"***.

VDEMs ***Shall*** only be sent and received after an Explicit Contract has been established.

A VDEM Message sequence ***Shall Not*** interrupt any other PD Message Sequence.

The VDEM does not define the contents of bits B14…0 in the VDM Header. Their definition and use are the sole responsibility of the vendor indicated by the SVID. The Port Partners and Cable Plugs ***Shall*** exit any states entered using an VDEM when a Hard Reset appears on PD.

The following rules apply to the use of VDEM Messages:

- VDEMs ***Shall*** only be used when an Explicit Contract is in place.
- Prior to establishing an Explicit Contract VDEMs ***Shall Not*** be sent and ***Shall*** be ***Ignored*** if received.
- Cable Plugs ***Shall Not*** initiate VDEMs.
- VDEMs ***Shall Not*** be initiated or responded to under any other circumstances.
- VDEMs ***Shall*** only be used during Modal Operation in the context of an *Active Mode* i.e., only after the UFP has Ack'ed the ***Enter Mode*** Command can VDEMs be sent or received. The *Active Mode* and the associated VDEMs ***Shall*** use the same SVID.

- VDEMs ***May*** be used with SOP* Packets.
- When a DFP or UFP does not support VDEMs or does not recognize the VID it ***Shall*** return a ***Not_Supported*** Message.

Note: Usage of VDEMs with Chunking is not recommended since this is less efficient than using Unstructured VDMs.

6.6 Timers

All the following timers are defined in terms of bits on the bus regardless of where they are implemented in terms of the logical architecture. This is to ensure a fixed reference for the starting and stopping of timers. It is left to the implementer to ensure that this timing is observed in a real system.

6.6.1 CRCReceiveTimer

The *CRCReceiveTimer* ***Shall*** be used by the sender's Protocol Layer to ensure that a Message has not been lost. Failure to receive an acknowledgement of a Message (a *GoodCRC* Message) whether caused by a bad CRC on the receiving end or by a garbled Message within *tReceive* is detected when the *CRCReceiveTimer* expires.

The sender's Protocol Layer response when a *CRCReceiveTimer* expires ***Shall*** be to retry *nRetryCount* times. Note that Cable Plugs do not retry Messages and large Extended Messages that are not Chunked are not retried (see *Section 6.7.2 "Retry Counter"*). Sending of the Preamble corresponding to the retried Message ***Shall*** start within *tRetry* of the *CRCReceiveTimer* expiring.

The *CRCReceiveTimer* ***Shall*** be started when the last bit of the Message *EOP* has been transmitted by the Physical Layer. The *CRCReceiveTimer* ***Shall*** be stopped when the last bit of the *EOP* corresponding to the *GoodCRC* Message has been received by the Physical Layer.

The Protocol Layer receiving a Message ***Shall*** respond with a *GoodCRC* Message within *tTransmit* in order to ensure that the sender's *CRCReceiveTimer* does not expire. The *tTransmit* ***Shall*** be measured from when the last bit of the Message *EOP* has been received by the Physical Layer until the first bit of the Preamble of the *GoodCRC* Message has been transmitted by the Physical Layer.

6.6.2 SenderResponseTimer

The *SenderResponseTimer* ***Shall*** be used by the sender's Policy Engine to ensure that a Message requesting a response (e.g., *Get_Source_Cap* Message) is responded to within a bounded time of *tSenderResponse*. Failure to receive the expected response is detected when the *SenderResponseTimer* expires.

For Extended Messages received as Chunks, the SenderResponseTimer will also be started and stopped by the Chunking Rx State Machine. See *Section 8.3.3.1.1 "SenderResponseTimer State Diagram"* for more details of the *SenderResponseTimer* operation.

The Policy Engine's response when the *SenderResponseTimer* expires ***Shall*** be dependent on the Message sent (see *Section 8.3 "Policy Engine"*).

The *SenderResponseTimer* ***Shall*** be started from the time the last bit of the *GoodCRC* Message *EOP*, corresponding to the Message requesting a response, has been received by the Physical Layer.

The *SenderResponseTimer* ***Shall*** be stopped when the last bit of the *EOP* of the *GoodCRC* Message, corresponding to the expected response Message, has been transmitted by the Physical Layer.

The receiver of a Message requiring a response ***Shall*** respond within *tReceiverResponse* in order to ensure that the sender's *SenderResponseTimer* does not expire.

The *tReceiverResponse* time ***Shall*** be measured from the time the last bit of the *GoodCRC* Message *EOP*, corresponding to the expected request Message, has been transmitted by the Physical Layer until the first bit of the response Message Preamble has been transmitted by the Physical Layer.

6.6.3 Capability Timers

Sources and Sinks use Capability Timers to determine Attachment of a PD Capable device. By periodically sending or requesting capabilities, it is possible to determine PD device Attachment when a response is received.

6.6.3.1 SourceCapabilityTimer

Prior to a successful negotiation a Source ***Shall*** use the ***SourceCapabilityTimer*** to periodically send out a ***Source_Capabilities*** Message every ***tTypeCSendSourceCap*** while:

- The Port is Attached.
- The Source is not in an active connection with a PD Sink Port.

Whenever there is a ***SourceCapabilityTimer*** timeout the Source ***Shall*** send a ***Source_Capabilities*** Message. It ***Shall*** then re-initialize and restart the ***SourceCapabilityTimer***. The ***SourceCapabilityTimer*** ***Shall*** be stopped when the last bit of the ***EOP*** corresponding to the ***GoodCRC*** Message has been received by the Physical Layer since a PD connection has been established. At this point, the Source waits for a ***Request*** Message or a response timeout.

See Section ***8.3.3.2 "Policy Engine Source Port State Diagram"*** more details of when ***Source_Capabilities*** Messages are transmitted.

6.6.3.2 SinkWaitCapTimer

The Sink ***Shall*** support the ***SinkWaitCapTimer***.

While in a Default Contract or an Implicit Contract when a Sink observes an absence of ***Source_Capabilities*** Messages, after V_{BUS} is present, for a duration of ***tTypeCSinkWaitCap*** the Sink ***May*** issue ***Hard Reset*** Signaling in order to restart the sending of ***Source_Capabilities*** Messages by the Source (see ***Section 6.7.4 "Capabilities Counter"***) or continue to operate at USB Type-C Current.

When a Sink, entering EPR Mode, observes an absence of ***EPR_Source_Capabilities*** Messages, after the ***GoodCRC*** Message acknowledging the ***EPR_Mode*** Message with the Action field set to 3 ("Succeeded"), for a duration of ***tTypeCSinkWaitCap*** the Sink ***Shall*** issue ***Hard Reset*** Signaling in order to exit EPR Mode (see ***Section 6.4.10 "EPR_Mode Message"***).

When a Sink, exiting EPR Mode, observes an absence of ***Source_Capabilities*** Messages, after the ***Good CRC*** Message acknowledging the ***EPR_Mode*** Message with the Action field set to 5 ("Exit"), for a duration of ***tTypeCSinkWaitCap*** the Sink ***Shall*** issue ***Hard Reset*** Signaling in order to restart the sending of ***Source_Capabilities*** Messages by the Source (see ***Section 6.7.4 "Capabilities Counter"***).

See ***Section 8.3.3.3 "Policy Engine Sink Port State Diagram"*** for more details of when the ***SinkWaitCapTimer*** are run.

6.6.3.3 tFirstSourceCap

After Port Partners are Attached or after a Hard Reset or after a Power Role Swap or after a Fast Role Swap a Source ***Shall*** send its first ***Source_Capabilities*** Message within ***tFirstSourceCap*** of V_{BUS} reaching ***vSafe5V***.

After Soft Reset, a Source ***Shall*** send its first ***Source_Capabilities*** Message within ***tFirstSourceCap*** after last bit of the ***GoodCRC*** Message ***EOP*** corresponding to ***Accept*** Message.

This ensures that the Sink receives a ***Source_Capabilities*** Message before the Sink's ***SinkWaitCapTimer*** expires.

A Source entering EPR Mode ***Shall*** send its first ***EPR_Source_Capabilities*** Message within ***tFirstSourceCap*** of the ***Good CRC*** Message acknowledging the ***EPR_Mode*** Message with the Action field set to 3 ("Succeeded").

A Source exiting EPR Mode ***Shall*** send its first ***Source_Capabilities*** Message within ***tFirstSourceCap*** of the ***Good CRC*** Message acknowledging the ***EPR_Mode*** Message with the Action field set to 5 (“Exit”).

6.6.4 Wait Timers and Times

6.6.4.1 SinkRequestTimer

The *SinkRequestTimer* is used to ensure that the time before the next Sink *Request* Message, after a *Wait* Message has been received from the Source in response to a Sink *Request* Message, is a minimum of *tSinkRequest* min (see Section *6.3.12 "Wait Message"*).

The *SinkRequestTimer* ***Shall*** be started when the *EOP* of a *Wait* Message has been received and ***Shall*** be stopped if any other Message is received or during a Hard Reset.

The Sink ***Shall*** wait at least *tSinkRequest,* after receiving the *EOP* of a *Wait* Message sent in response to a Sink *Request* Message, before sending a new *Request* Message. Whenever there is a *SinkRequestTimer* timeout the Sink ***May*** send a *Request* Message. It ***Shall*** then re-initialize and restart the *SinkRequestTimer*.

6.6.4.2 tPRSwapWait

The time before the next *PR_Swap* Message, after a *Wait* Message has been received in response to a *PR_Swap* Message is a minimum of *tPRSwapWait* min (see *6.3.12 "Wait Message"*). The Port ***Shall*** wait at least *tPRSwapWait* after receiving the *EOP* of a *Wait* Message sent in response to a *PR_Swap* Message, before sending a new *PR_Swap* Message.

6.6.4.3 tDRSwapWait

The time before the next *DR_Swap* Message, after a *Wait* Message has been received in response to a *DR_Swap* Message is a minimum of *tDRSwapWait* min (see *6.3.12 "Wait Message"*). The Port ***Shall*** wait at least *tDRSwapWait* after receiving the *EOP* of a *Wait* Message sent in response to a *DR_Swap* Message, before sending a new *DR_Swap* Message.

6.6.4.4 tVconnSwapWait

The time before the next *VCONN_Swap* Message, after a *Wait* Message has been received in response to a *VCONN_Swap* Message is a minimum of *tVCONNSwapWait* min (see *6.3.12 "Wait Message"*). The Port ***Shall*** wait at least *tVCONNSwapWait* after receiving the *EOP* of a *Wait* Message sent in response to a *VCONN_Swap* Message, before sending a new *VCONN_Swap* Message.

6.6.4.5 tEnterUSBWait

The time before the next *Enter_USB* Message, after a *Wait* Message has been received in response to a *Enter_USB* Message is a minimum of *tEnterUSBWait* min (see *6.3.12 "Wait Message"*). The DFP ***Shall*** wait at least *tEnterUSBWait* after receiving the *EOP* of a *Wait* Message sent in response to an *Enter_USB* Message, before sending a new *Enter_USB* Message.

6.6.5 Power Supply Timers

See *Section 7.3 "Transitions"* for diagrams showing the usage of the timers in this section.

6.6.5.1 PSTransitionTimer

The *PSTransitionTimer* is used by the Policy Engine to timeout on a *PS_RDY* Message. It is started when a request for a new Capability has been accepted and will timeout after *tPSTransition* if a *PS_RDY* Message has not been received. This condition leads to a Hard Reset and a return to USB Default Operation. The *PSTransitionTimer* relates to the time taken for the Source to transition from one Voltage, or current level, to another (see *Section 7.1 "Source Requirements"*).

The *PSTransitionTimer* ***Shall*** be started when the last bit of the *GoodCRC* Message *EOP*, corresponding to an *Accept* or *GotoMin* Message, has been transmitted by the Physical Layer. The *PSTransitionTimer* ***Shall*** be stopped when the last bit of the *GoodCRC* Message *EOP*, corresponding to the *PS_RDY* Message, has been transmitted by the Physical Layer.

6.6.5.2 PSSourceOffTimer

6.6.5.2.1 Use during Power Role Swap

The *PSSourceOffTimer* is used by the Policy Engine in Dual-Role Power Device that is currently acting as a Sink to timeout on a *PS_RDY* Message during a Power Role Swap sequence. This condition leads to USB Type-C® Error Recovery.

If a *PR_Swap* Message request has been sent by the Dual-Role Power Device currently acting as a Source the Sink can respond with an *Accept* Message. When the last bit of the *GoodCRC* Message *EOP*, corresponding to this transmitted *Accept* Message, is received by the Sink's Physical Layer, then the *PSSourceOffTimer* ***Shall*** be started.

If a *PR_Swap* Message request has been sent by the Dual-Role Power Device currently acting as a Sink the Source can respond with an *Accept* Message. When the last bit of the *GoodCRC* Message *EOP*, corresponding to this received *Accept* Message, is transmitted by the Sink's Physical Layer, then the *PSSourceOffTimer* ***Shall*** be started.

The *PSSourceOffTimer* ***Shall*** be stopped when:

- The last bit of the *GoodCRC* Message *EOP*, corresponding to the received *PS_RDY* Message, is transmitted by the Physical Layer.

The *PSSourceOffTimer* relates to the time taken for the remote Dual-Role Power Device to stop supplying power (see also *Section 7.3.2.1 "Sink Requested Power Role Swap"* and *Section 7.3.2.2 "Source Requested Power Role Swap"*). The timer ***Shall*** time out if a *PS_RDY* Message has not been received from the remote Dual-Role Power Device within *tPSSourceOff* indicating this has occurred.

6.6.5.2.2 Use during Fast Role Swap

The *PSSourceOffTimer* is used by the Policy Engine in Dual-Role Power Device that is the initial Sink (currently providing *vSafe5V*) to timeout on a *PS_RDY* Message during a Fast Role Swap sequence. This condition leads to USB Type-C® Error Recovery.

When the *FR_Swap* Message request has been sent by the initial Sink, the initial Source ***Shall*** respond with an *Accept* Message. When the last bit of the *GoodCRC* Message *EOP*, corresponding to this *Accept* Message is received by the initial Sink's Physical Layer, then the *PSSourceOffTimer* ***Shall*** be started.

The *PSSourceOffTimer* ***Shall*** be stopped when:

- The last bit of the *GoodCRC* Message *EOP*, corresponding to the received *PS_RDY* Message, is transmitted by the Physical Layer.

The *PSSourceOffTimer* relates to the time taken for the initial Source to stop supplying power and for V_{BUS} to revert to *vSafe5V* (see also *Section 7.2.10 "Fast Role Swap"* and *Section 7.3.5 "Transitions Caused by Fast Role Swap"*). The timer ***Shall*** time out if a *PS_RDY* Message has not been received from the initial Source within *tPSSourceOff* indicating this has occurred.

6.6.5.3 PSSourceOnTimer

6.6.5.3.1 Use during Power Role Swap

The *PSSourceOnTimer* is used by the Policy Engine in Dual-Role Power Device that has just stopped sourcing power and is waiting to start sinking power to timeout on a *PS_RDY* Message during a Power Role Swap. This condition leads to USB Type-C® Error Recovery.

The *PSSourceOnTimer* ***Shall*** be started when:

- The last bit of the *GoodCRC* Message *EOP*, corresponding to the transmitted *PS_RDY* Message, is received by the Physical Layer.

The *PSSourceOnTimer* ***Shall*** be stopped when:

- The last bit of the *GoodCRC* Message *EOP*, corresponding to the received *PS_RDY* Message, is transmitted by the Physical Layer.

The *PSSourceOnTimer* relates to the time taken for the remote Dual-Role Power Device to start sourcing power (see also *Section 7.3.2.1 "Sink Requested Power Role Swap"* and *Section 7.3.2.2 "Source Requested Power Role Swap"*) and will time out if a *PS_RDY* Message indicating this has not been received within *tPSSourceOn*.

6.6.5.3.2 Use during Fast Role Swap

The *PSSourceOnTimer* is used by the Policy Engine in Dual-Role Power Device that has just stopped sourcing power and is waiting to start sinking power to timeout on a *PS_RDY* Message during a Fast Role Swap. This condition leads to USB Type-C® Error Recovery.

The *PSSourceOnTimer* ***Shall*** be started when:

- The last bit of the *GoodCRC* Message *EOP*, corresponding to the transmitted *PS_RDY* Message, is received by the Physical Layer.

The *PSSourceOnTimer* ***Shall*** be stopped when:

- The last bit of the *GoodCRC* Message *EOP*, corresponding to the received *PS_RDY* Message, is transmitted by the Physical Layer.

The *PSSourceOnTimer* relates to the time taken for the remote Dual-Role Power Device to start sourcing power (see also *Section 7.2.10 "Fast Role Swap"* and *Section 7.3.5 "Transitions Caused by Fast Role Swap"*) and will time out if a *PS_RDY* Message indicating this has not been received within *tPSSourceOn*.

6.6.6 NoResponseTimer

The ***NoResponseTimer*** is used by the Policy Engine in a Source to determine that its Port Partner is not responding after a Hard Reset. When the ***NoResponseTimer*** times out, the Policy Engine ***Shall*** issue up to ***nHardResetCount*** additional Hard Resets before determining that the Port Partner is non-responsive to USB Power Delivery messaging.

If the Source fails to receive a ***GoodCRC*** Message in response to a ***Source_Capabilities*** Message within ***tNoResponse*** of:

- The last bit of a ***Hard Reset*** Signaling being sent by the PHY Layer if the ***Hard Reset*** Signaling was initiated by the Sink.
- The last bit of a ***Hard Reset*** Signaling being received by the PHY Layer if the ***Hard Reset*** Signaling was initiated by the Source.

Then the Source ***Shall*** issue additional Hard Resets up to ***nHardResetCount*** times (see ***Section 6.8.3 "Hard Reset"***).

For a non-responsive device, the Policy Engine in a Source ***May*** either decide to continue sending ***Source_Capabilities*** Messages or to go to non-USB Power Delivery operation and cease sending ***Source_Capabilities*** Messages.

6.6.7 BIST Timers

6.6.7.1 tBISTCarrierMode

tBISTCarrierMode is used to define the maximum time that a UUT has to enter BIST Carrier Mode when requested by a Tester.

A UUT ***Shall*** enter BIST Carrier Mode within ***tBISTCarrierMode*** of the last bit of the ***GoodCRC*** Message ***EOP***, corresponding to the received the ***BIST*** Message used to initiate the test, being transmitted by the Physical Layer. In ***BIST Carrier Mode*** when transmitting a continuous carrier signal transmission ***Shall*** start as soon as the UUT enters BIST mode.

6.6.7.2 BISTContModeTimer

The ***BISTContModeTimer*** is used by a UUT to ensure that a Continuous BIST Mode (i.e., ***BIST Carrier Mode***) is exited in a timely fashion. A UUT that has been put into a Continuous BIST Mode ***Shall*** return to normal operation (either ***PE_SRC_Transition_to_default***, ***PE_SNK_Transition_to_default***, or ***PE_CBL_Ready***) within ***tBISTContMode*** of starting to transmit a continuous carrier signal.

6.6.7.3 tBISTSharedTestMode

tBISTSharedTestMode is used to define the maximum time that a UUT has to enter BIST Shared Capacity Test Mode when requested by a Tester.

A UUT ***Shall*** enter BIST Shared Capacity Test Mode and send a new ***Source_Capabilities*** Message from all Ports within the shared capacity group within ***tBISTSharedTestMode*** of the last bit of the ***GoodCRC*** Message ***EOP***, corresponding to the received the ***BIST*** Message used to initiate the test, being transmitted by the Physical Layer.

6.6.8 Power Role Swap Timers

6.6.8.1 SwapSourceStartTimer

The ***SwapSourceStartTimer*** ***Shall*** be used by the new Source, after a Power Role Swap or Fast Role Swap, to ensure that it does not send ***Source_Capabilities*** Message before the new Sink is ready to receive the ***Source_Capabilities*** Message. The new Source ***Shall Not*** send the ***Source_Capabilities*** Message earlier than ***tSwapSourceStart*** after the last bit of the ***EOP*** of ***GoodCRC*** Message sent in response to the ***PS_RDY*** Message sent by the new Source indicating that its power supply is ready. The Sink ***Shall*** be ready to receive a ***Source_Capabilities*** Message ***tSwapSinkReady*** after having sent the last bit of the ***EOP*** of ***GoodCRC*** Message sent in response to the ***PS_RDY*** Message sent by the new Source indicating that its power supply is ready.

6.6.9 Soft Reset Timers

6.6.9.1 tSoftReset

A failure to see a ***GoodCRC*** Message in response to any Message within ***tReceive*** (after ***nRetryCount*** retries), when a Port Pair is Connected, is indicative of a communications failure. This ***Shall*** cause the Source or Sink to send a ***Soft_Reset*** Message, transmission of which ***Shall*** be completed within ***tSoftReset*** of the ***CRCReceiveTimer*** expiring.

6.6.9.2 tProtErrSoftReset

If the Protocol Error occurs that causes the Source or Sink to send a ***Soft_Reset*** Message, the transmission of the ***Soft_Reset*** Message ***Shall*** be completed within ***tProtErrSoftReset*** of the ***EOP*** of the ***GoodCRC*** sent in response to the Message that caused the Protocol Error.

6.6.10 Data Reset Timers

6.6.10.1 VCONNDischargeTimer

The ***VCONNDischargeTimer*** is used by the Policy Engine in the DFP to ensure the UFP actively discharges VCONN in a timely manner to ensure the cable will restore Ra. Once the UFP has discharged VCONN below vRaReconnect (see ***[USB Type-C 2.3]***) it sends a ***PS_RDY*** Message (see also ***Section 7.1.15 "Vconn Power Cycle"***).

If the DFP does not receive a ***PS_RDY*** Message from the UFP within ***tVCONNSourceDischarge*** of the last bit of the ***GoodCRC*** acknowledging the ***Accept*** message in response to the ***Data_Reset*** Message, the ***VCONNDischargeTimer*** will time out and the Policy Engine ***Shall*** enter the ***ErrorRecovery*** State.

6.6.10.2 tDataReset

The DFP ***Shall*** complete the Data_Reset process (as defined in ***Section 6.3.14 "Data_Reset Message"***) within ***tDataReset*** of the last bit of the ***GoodCRC*** Message ***EOP***, corresponding to the ***Accept*** Message, being transmitted by the Physical Layer.

6.6.10.3 DataResetFailTimer

The ***DataResetFailTimer*** ***Shall*** be used by the DFP's Policy Engine to ensure the Data Reset process completes within ***tDataResetFail*** of the last bit of the ***GoodCRC*** acknowledging the ***Accept*** Message in response to the ***Data_Reset*** Message. If the DFP's ***DataResetFailTimer*** expires, the DFP ***Shall*** enter the ***ErrorRecovery*** State.

6.6.10.4 DataResetFailUFPTimer

The ***DataResetFailUFPTimer*** ***Shall*** be used by the UFP's Policy Engine to ensure the Data Reset process completes within ***tDataResetFailUFP*** of the last bit of the ***GoodCRC*** acknowledging the ***Accept*** Message in response to the ***Data_Reset*** Message. If the UFP's ***DataResetFailUFPTimer*** expires, the UFP ***Shall*** enter the ***ErrorRecovery*** State.

6.6.11 Hard Reset Timers

6.6.11.1 HardResetCompleteTimer

The ***HardResetCompleteTimer*** is used by the Protocol Layer in the case where it has asked the PHY Layer to send ***Hard Reset*** Signaling and the PHY Layer is unable to send the Signaling within a reasonable time due to a non-idle channel. If the PHY Layer does not indicate that the ***Hard Reset*** Signaling has been sent within ***tHardResetComplete*** of the Protocol Layer requesting transmission, then the Protocol Layer ***Shall*** inform the Policy Engine that the ***Hard Reset*** Signaling has been sent in order to ensure the power supply is reset in a timely fashion.

6.6.11.2 PSHardResetTimer

The ***PSHardResetTimer*** is used by the Policy Engine in a Source to ensure that the Sink has had sufficient time to process ***Hard Reset*** Signaling before turning off its power supply to V_{BUS}.

When a Hard Reset occurs the Source, stops driving V_{CONN}, removes R_p from the V_{CONN} pin and starts to transition the V_{BUS} Voltage to ***vSafe0V*** either:

- ***tPSHardReset*** after the last bit of the ***Hard Reset*** Signaling has been received from the Sink or
- ***tPSHardReset*** after the last bit of the ***Hard Reset*** Signaling has been sent by the Source.

See ***Section 7.1.5 "Response to Hard Resets"***.

6.6.11.3 tDRSwapHardReset

If a ***DR_Swap*** Message is received during Modal Operation then a Hard Reset ***Shall*** be initiated by the recipient of the unexpected ***DR_Swap*** Message; ***Hard Reset*** Signaling ***Shall*** be generated within ***tDRSwapHardReset*** of the EOP of the ***GoodCRC*** sent in response to the ***DR_Swap*** Message.

6.6.11.4 tProtErrHardReset

If a Protocol Error occurs that directly leads to a Hard Reset, the transmission of the ***Hard Reset*** Signaling ***Shall*** be completed within ***tProtErrHardReset*** of the ***EOP*** of the ***GoodCRC*** sent in response to the Message that caused the Protocol Error.

6.6.12 Structured VDM Timers

6.6.12.1 VDMResponseTimer

The ***VDMResponseTimer*** ***Shall*** be used by the Initiator's Policy Engine to ensure that a Structured VDM Command request needing a response (e.g. ***Discover Identity*** Command request) is responded to within a bounded time of ***tVDMSenderResponse***. The ***VDMResponseTimer*** ***Shall*** be applied to all Structured VDM Commands except the ***Enter Mode*** and ***Exit Mode*** Commands which have their own timers (***VDMModeEntryTimer*** and ***VDMModeExitTimer*** respectively). Failure to receive the expected response is detected when the ***VDMResponseTimer*** expires.

The Policy Engine's response when the ***VDMResponseTimer*** expires ***Shall*** be dependent on the Message sent (see ***Section 8.3 "Policy Engine"***).

The ***VDMResponseTimer*** ***Shall*** be started from the time the last bit of the ***GoodCRC*** Message ***EOP***, corresponding to the VDM Command requesting a response, has been received by the Physical Layer. The ***VDMResponseTimer*** ***Shall*** be stopped when the last bit of the ***EOP*** of the ***GoodCRC*** Message, corresponding to the expected VDM Command response, has been transmitted by the Physical Layer.

The receiver of a Message requiring a response ***Shall*** respond within ***tVDMReceiverResponse*** in order to ensure that the sender's ***VDMResponseTimer*** does not expire.

The ***tVDMReceiverResponse*** time ***Shall*** be measured from the time the last bit of the Message ***EOP*** has been transmitted by the Physical Layer until the first bit of the response Message Preamble has been transmitted by the Physical Layer.

6.6.12.2 VDMModeEntryTimer

The ***VDMModeEntryTimer*** ***Shall*** be used by the Initiator's Policy Engine to ensure that the response to a Structured VDM ***Enter Mode*** Command request (ACK or NAK with ACK indicating that the requested Mode has been entered) arrives within a bounded time of ***tVDMWaitModeEntry***. Failure to receive the expected response is detected when the ***VDMModeEntryTimer*** expires.

The Policy Engine's response when the ***VDMModeEntryTimer*** expires is to inform the Device Policy Manager (see ***Section 8.3.3.24.1 "DFP Structured VDM Mode Entry State Diagram"***).

The ***VDMModeEntryTimer*** ***Shall*** be started from the time the last bit of the ***EOP*** of the ***GoodCRC*** Message, corresponding to the VDM Command request, has been received by the Physical Layer. The ***VDMModeEntryTimer*** ***Shall*** be stopped when the last bit of the ***EOP*** of the ***GoodCRC*** Message, corresponding to the expected Structured VDM Command response (ACK, NAK or BUSY), has been transmitted by the Physical Layer.

The receiver of a Message requiring a response ***Shall*** respond within ***tVDMEnterMode*** in order to ensure that the sender's ***VDMModeEntryTimer*** does not expire.

The ***tVDMEnterMode*** time ***Shall*** be measured from the time the last bit of the ***EOP*** of the ***GoodCRC*** Message, corresponding to VDM Command Request, has been transmitted by the Physical Layer until the first bit of the response Message Preamble has been transmitted by the Physical Layer.

6.6.12.3 VDMModeExitTimer

The ***VDMModeExitTimer*** ***Shall*** be used by the Initiator's Policy Engine to ensure that the ACK response to a Structured VDM ***Exit Mode*** Command, indicating that the requested Mode has been exited, arrives within a bounded time of ***tVDMWaitModeExit***. Failure to receive the expected response is detected when the ***VDMModeExitTimer*** expires.

The Policy Engine's response when the ***VDMModeExitTimer*** expires is to inform the Device Policy Manager (see ***Section 8.3.3.24.2 "DFP Structured VDM Mode Exit State Diagram"***).

The ***VDMModeExitTimer*** ***Shall*** be started from the time the last bit of the ***GoodCRC*** Message ***EOP***, corresponding to the VDM Command requesting a response, has been received by the Physical Layer. The ***VDMModeExitTimer*** ***Shall*** be stopped when the last bit of the ***GoodCRC*** Message ***EOP***, corresponding to the expected Structured VDM Command response ACK, has been transmitted by the Physical Layer.

The receiver of a Message requiring a response ***Shall*** respond within ***tVDMExitMode*** in order to ensure that the sender's ***VDMModeExitTimer*** does not expire.

The ***tVDMExitMode*** time ***Shall*** be measured from the time the last bit of the Message ***EOP*** has been received by the Physical Layer until the first bit of the response Message Preamble has been transmitted by the Physical Layer.

6.6.12.4 tVDMBusy

The Initiator ***Shall*** wait at least ***tVDMBusy***, after receiving a BUSY Command response, before repeating the Structured VDM request again.

6.6.13 VCONN Timers

6.6.13.1 VCONNOnTimer

The ***VCONNOnTimer*** is used during a VCONN Swap.

The ***VCONNOnTimer*** ***Shall*** be started when:

- The last bit of ***GoodCRC*** Message ***EOP***, corresponding to the ***Accept*** Message, is transmitted or received by the Physical Layer.

The ***VCONNOnTimer*** ***Shall*** be stopped when:

- The last bit of the ***GoodCRC*** Message ***EOP***, corresponding to the ***PS_RDY*** Message, is transmitted by the Physical Layer.

Prior to sending the ***PS_RDY*** Message, the Port ***Shall*** have turned VCONN On.

6.6.13.2 tVCONNSourceOff

The ***tVCONNSourceOff*** time applies during a VCONN Swap. The initial VCONN Source ***Shall*** cease sourcing VCONN within ***tVCONNSourceOff*** of the last bit of the ***GoodCRC*** Message ***EOP***, corresponding to the ***PS_RDY*** Message, being transmitted by the Physical Layer.

6.6.14 tCableMessage

Ports compliant with this Revision of the specification ***Shall Not*** wait ***tCableMessage*** before sending an SOP' or SOP'' Packet even when communicating using ***[USBPD 2.0]*** with a Cable Plug. This specification defines collision avoidance mechanisms that obviate the need for this time.

Cable Plugs ***Shall*** only wait ***tCableMessage*** before sending an SOP' or SOP'' Packet when operating at ***[USBPD 2.0]***. When operating at Revisions higher than ***[USBPD 2.0]*** Cable Plugs ***Shall Not*** wait ***tCableMessage*** before sending an SOP' or SOP'' Packet.

6.6.15 DiscoverIdentityTimer

The ***DiscoverIdentityTimer*** is used during an Explicit Contract when discovering whether a Cable Plug is PD Capable using SOP'. When performing cable discovery during an Explicit Contract the ***Discover Identity*** Command request ***Shall*** be sent every ***tDiscoverIdentity***. No more than ***nDiscoverIdentityCount Discover Identity*** Messages without a ***GoodCRC*** Message response ***Shall*** be sent. If no ***GoodCRC*** Message response is received after ***nDiscoverIdentityCount Discover Identity*** Command requests have been sent by a Port, the Port ***Shall Not*** send any further SOP'/SOP'' Messages.

6.6.16 Collision Avoidance Timers

6.6.16.1 SinkTxTimer

The ***SinkTxTimer*** is used by the Protocol Layer in a Source to allow the Sink to complete its transmission before initiating an AMS.

The Source ***Shall*** wait a minimum of ***tSinkTx*** after changing R_p from ***SinkTxOk*** to ***SinkTxNG*** before initiating an AMS by sending a Message.

A Sink ***Shall*** only initiate an AMS when it has determined that R_p is set to ***SinkTxOk***.

6.6.16.2 tSrcHoldsBus

If a transition into the ***PE_SRC_Ready*** state will result in an immediate transition out of the ***PE_SRC_Ready*** state within ***tSrcHoldsBus*** e.g. it is due to a Protocol Error that has not resulted in a Soft Reset, then the notifications of the end of AMS and first Message in an AMS ***May Not*** be sent to avoid changing the R_p value unnecessarily.

6.6.17 Fast Role Swap Timers

6.6.17.1 tFRSwap5V

The ***tFRSwap5V*** time ***Shall*** be measured from:

- The later of:
  - The last bit of the ***GoodCRC*** Message ***EOP*** , corresponding to the ***Accept*** message.
  - V_{BUS} being within ***vSafe5V***.
- Until the first bit of the response ***PS_RDY*** Message Preamble has been transmitted by the Physical Layer.

During a Fast Role Swap, the initial Source ***Shall*** start the ***PS_RDY*** Message within ***tFRSwap5V*** after both:

- The initial Source has sent the ***Accept*** Message, and
- V_{BUS} is at or below ***vSafe5V***.

6.6.17.2 tFRSwapComplete

During a fast-role swap, the initial Sink ***Shall*** respond with a the ***PS_RDY*** Message within ***tFRSwapComplete*** after it has received the ***PS_RDY*** Message from the Initial Source. The ***tFRSwapComplete*** time ***Shall*** be measured from the time the last bit of the ***GoodCRC*** Message ***EOP***, corresponding to the ***PS_RDY*** Message, has been transmitted by the Physical Layer until the first bit of the response ***PS_RDY*** Message Preamble has been transmitted by the Physical Layer.

6.6.17.3 tFRSwapInit

That last bit of the ***EOP*** of the ***FR_Swap*** Message ***Shall*** be transmitted by the new Source no later than ***tFRSwapInit*** after the Fast Role Swap Request has been detected (see ***Section 5.8.6.3 "Fast Role Swap Detection"***).

6.6.18 Chunking Timers

6.6.18.1 ChunkingNotSupportedTimer

The ***ChunkingNotSupportedTimer*** is used by a Source or Sink which does not support multi-chunk Chunking but has received a Message Chunk.

The ***ChunkingNotSupportedTimer*** ***Shall*** be started when:

- The last bit of the ***GoodCRC*** Message ***EOP***, corresponding to a Message Chunk of a multi-chunk Message, is transmitted by the Physical Layer. The Policy Engine ***Shall Not*** send its ***Not_Supported*** Message before the ChunkingNotSupportedTimer expires.

6.6.18.2 ChunkSenderRequestTimer

The ***ChunkSenderRequestTimer*** is used during a Chunked Message transmission.

The ***ChunkSenderRequestTimer*** ***Shall*** be used by the sender's Chunking state machine to ensure that a Chunk Response is responded to within a bounded time of ***tChunkSenderRequest***. Failure to receive the expected response is detected when the ***ChunkSenderRequestTimer*** expires.

The ***ChunkSenderRequestTimer*** ***Shall*** be started when:

- The last bit of the ***GoodCRC*** Message ***EOP***, corresponding to the Chunk Response Message, is received by the Physical Layer.

The ***ChunkSenderRequestTimer*** ***Shall*** be stopped when:

- The last bit of the ***EOP*** of the ***GoodCRC*** Message, corresponding to the Chunk Request Message, is transmitted by the Physical Layer.
- A Message other than a Chunk Request is received from the Protocol Layer Rx.

The receiver of a Chunk Response requiring a Chunk Request ***Shall*** respond with a Chunk Request within ***tChunkReceiverRequest*** in order to ensure that the sender's ***ChunkSenderRequestTimer*** does not expire.

The ***tChunkReceiverRequest*** time ***Shall*** be measured from the time the last bit of the ***EOP*** of the ***GoodCRC*** Message, corresponding to the Chunk Response Message, has been transmitted by the Physical Layer until the first bit of the response Message Preamble has been transmitted by the Physical Layer.

6.6.18.3 ChunkSenderResponseTimer

The ***ChunkSenderResponseTimer*** is used during a Chunked Message transmission.

The ***ChunkSenderResponseTimer*** ***Shall*** be used by the sender's Chunking state machine to ensure that a Chunk Request is responded to within a bounded time of ***tChunkSenderResponse***. Failure to receive the expected response is detected when the ChunkSenderResponseTimer expires.

The ***ChunkSenderResponseTimer*** ***Shall*** be started when:

- The last bit of the ***GoodCRC*** Message ***EOP***, corresponding to the Chunk Request Message, is received by the Physical Layer.

The ***ChunkSenderResponseTimer*** ***Shall*** be stopped when:

- The last bit of the ***GoodCRC*** Message ***EOP***, corresponding to the Chunk Response Message, is transmitted by the Physical Layer.
- A Message other than a Chunk is received from the Protocol Layer.

The receiver of a Chunk Request requiring a Chunk Response ***Shall*** respond with a Chunk Response within ***tChunkReceiverResponse*** in order to ensure that the sender's ***ChunkSenderResponseTimer*** does not expire.

The ***tChunkReceiverResponse*** time ***Shall*** be measured from the time the last bit of the ***EOP*** of the ***GoodCRC*** Message, corresponding to the Chunk Request Message, has been transmitted by the Physical Layer until the first bit of the response Message Preamble has been transmitted by the Physical Layer.

6.6.19 Programmable Power Supply Timers

6.6.19.1 SinkPPSPeriodicTimer

The ***SinkPPSPeriodicTimer*** ***Shall*** be used by the Sink's Policy Engine to ensure that communication between the Sink and Source occurs within a bounded time of ***tPPSRequest*** when in SPR PPS operation. In the absence of any other traffic, a ***Request*** Message requesting an SPR PPS APDO is sent periodically as a keep alive mechanism.

SinkPPSPeriodicTimer ***Shall*** be re-initialized and restarted on transmission, by the Physical Layer, of the last bit of the ***GoodCRC*** Message ***EOP***, corresponding to any received Message, that causes the Sink to enter the ***PE_SNK_Ready*** state.

The Sink ***Shall*** stop the ***SinkPPSPeriodicTimer*** on transmission, by the Physical Layer, of the last bit of the ***GoodCRC*** Message ***EOP***, corresponding to any Message, or the last bit of any Signaling is received, by the Physical Layer, from the Source and by the Sink that causes the Sink to leave the ***PE_SNK_Ready*** state.

6.6.19.2 SourcePPSCommTimer

The ***SourcePPSCommTimer*** ***Shall*** be used by the Source's Policy Engine to ensure that communication between the Sink and Source occurs within a bounded time of ***tPPSTimeout*** when in SPR PPS operation. In the absence of any other traffic, a ***Request*** Message requesting an SPR PPS APDO is received periodically as a keep alive mechanism.

SourcePPSCommTimer ***Shall*** be re-initialized and restarted when, after receiving any Message that causes the Source to enter the ***PE_SRC_Ready*** state, the last bit of the corresponding ***GoodCRC*** Message ***EOP*** is transmitted by the Physical Layer.

The Source ***Shall*** stop the ***SourcePPSCommTimer*** when:

- after receiving any message that causes the Source to leave the ***PE_SRC_Ready*** state, the last bit of the of the corresponding GoodCRC Message ***EOP*** is sent by the Physical Layer, or
- the last bit of any Signaling is received by the Physical Layer from the Sink by the Source that causes the Source to leave the ***PE_SRC_Ready*** state.

When the ***SourcePPSCommTimer*** times out the Source ***Shall*** issue ***Hard Reset*** Signaling.

6.6.20 tEnterUSB

The DFP ***Shall*** send the ***Enter_USB*** Message within ***tEnterUSB*** of either:

- The last bit of the ***GoodCRC*** acknowledging the ***Data_Reset_Complete*** Message in response to the ***Data_Reset*** Message or
- A PD Connection , specifically the last bit of the ***GoodCRC*** acknowledging the ***Source_Capabilities*** Message after the initial entry into the ***PE_SRC_Send_Capabilities*** state or
- The last bit of the ***GoodCRC*** acknowledging the ***Accept*** Message in response to the ***DR_Swap*** Message

Failure by the DFP to meet this timeout parameter can result in the ports not transitioning into ***[USB4]*** operation. Any AMS initiated by the UFP prior to receiving the ***Enter_USB*** Message will delay reception of the ***Enter_USB*** Message and ***[USB4]*** operation, therefore a USB4® -capable UFP ***Should Not*** initiate any AMS until the DFP has been given time to send the ***Enter_USB*** Message.

6.6.21 EPR Timers

6.6.21.1 SinkEPREnterTimer Timer

The ***SinkEPREnterTimer*** is used to ensure the EPR Mode entry process completes within ***tEnterEPR***. The Sink ***Shall*** start the timer when it sees the last bit of the ***GoodCRC*** Message in response to the ***EPR_Mode*** Message with the Action field set to 1 ("Enter"). The Sink ***Shall*** stop the timer when the last bit of the corresponding ***GoodCRC*** Message ***EOP***, corresponding to the received ***EPR_Mode*** Message with the Action field set to 3 ("Enter Succeeded"), has been transmitted by the Physical Layer. If the timer expires the Sink ***Shall*** send a ***Soft_Reset*** Message.

6.6.21.2 SinkEPRKeepAlive Timer

The ***SinkEPRKeepAliveTimer*** ***Shall*** be used by the Sink's Policy Engine to ensure that communication between the Sink and Source occurs within a bounded time of ***tSinkEPRKeepAlive***. The Sink ***Shall*** initialize and run this timer upon entry into the ***PE_SNK_Ready*** State when in EPR mode and ***Shall*** stop it upon exit from the ***PE_SNK_Ready*** when in EPR Mode.

While operating in EPR mode, the Sink ***Shall*** stop the ***SinkEPRKeepAliveTimer*** timer whenever:

- The last bit of the ***GoodCRC*** Message ***EOP***, in response any Message from the Source, is transmitted by the Physical Layer.
- The Physical Layer Receives the last bit of the ***GoodCRC*** Message ***EOP*** in response to any Message sent to the Source.

If the timer expires the Sink ***Shall*** send an ***EPR_KeepAlive*** Message.

6.6.21.3 SourceEPRKeepAlive Timer

The ***SourceEPRKeepAliveTimer*** ***Shall*** be used by the Source's Policy Engine to ensure that communication between the Sink and Source occurs within a bounded time of ***tSourceEPRKeepAlive***. The Source ***Shall*** initialize and run this timer upon entry into the ***PE_SRC_Ready*** State when in EPR mode and ***Shall*** disable it upon exit from the ***PE_SRC_Ready*** State when EPR mode.

While operating in EPR mode, the Source ***Shall*** stop the ***SourceEPRKeepAliveTimer*** timer whenever:

- The last bit of the ***GoodCRC*** Message ***EOP***, in response any Message from the Sink, is transmitted by the Physical Layer.
- The Physical Layer receives the last bit of the ***GoodCRC*** Message ***EOP*** in response to any Message sent to the Source.

If the timer expires the Source ***Shall*** send ***Hard Reset*** Signaling.

6.6.21.4 tEPRSourceCableDiscovery

After Port Partners are Attached or after a Hard Reset or after a Power Role Swap or after a Fast Role Swap a Source ***Shall*** discover the Cable Plug within ***tEPRSourceCableDiscovery*** of entering the first Explicit Contract.

The Source ***Shall*** send the ***Discover Identity*** REQ Command, to the Cable Plug, within ***tEPRSourceCableDiscovery*** of receiving the ***GoodCRC*** Message acknowledging the ***PS_RDY*** Message as part of the Explicit Contract negotiation.

Note: if the Source is not the VCONN Source, ***tEPRSourceCableDiscovery***, will also include the time needed for the VCONN Swap.

6.6.22 Time Values and Timers

Table 6.69 "Time Values" summarizes the values for the timers listed in this section. For each Timer Value, a given implementation ***Shall*** pick a fixed value within the range specified. *Table 6.70 "Timers"* lists the timers.

Table 6.69 "Time Values"

Parameter	Value (min)	Value (nom)	Value (max)	Units	Reference
tACTempUpdate			500	ms	*Section 6.5.2.2.1*
tBISTContMode	30	45	60	ms	*Section 6.6.7.2*
tBISTCarrierMode			300	ms	*Section 6.6.7.1*
tBISTSharedTestMode			1	s	*Section 6.6.7.3*
tCableMessage	750			µs	*Section 6.6.14*
tChunkingNotSupported	40	45	50	ms	*Section 6.6.18.1*
tChunkReceiverRequest			15	ms	*Section 6.6.18.2*
tChunkReceiverResponse			15	ms	*Section 6.6.18.3*
tChunkSenderRequest	24	27	30	ms	*Section 6.6.18.2*
tChunkSenderResponse	24	27	30	ms	*Section 6.6.18.3*
tDataReset	200	225	250	ms	*Section 6.6.10.2*
tDataResetFail	300		400	ms	*Section 6.6.10.3*
tDataResetFailUFP	450		550	ms	*Section 6.6.10.4*
tDiscoverIdentity	40		50	ms	*Section 6.6.14*
tDRSwapHardReset			15	ms	*Section 6.6.11.3*
tDRSwapWait	100			ms	*Section 6.6.4.3*
tEnterUSB			500	ms	*Section 6.6.20*
tEnterUSBWait	100			ms	*Section 6.6.4.5*
tEnterEPR	450	500	550	ms	*Section 6.6.21.1*
tEPRSourceCableDiscovery			2	s	*Section 6.6.21.4*
tFirstSourceCap			250	ms	*Section 6.6.3.3*
tFRSwap5V			15	ms	*Section 6.6.17.1*
tFRSwapComplete			15	ms	*Section 6.6.17.2*
tFRSwapInit			15	ms	*Section 6.6.17.3*
tHardReset			5	ms	*Section 6.3.13*
tHardResetComplete	4	4.5	5	ms	*Section 6.6.9*
tSourceEPRKeepAlive	0.750	0.875	1.000	s	*Section 6.6.21.3*
tSinkEPRKeepAlive	0.250	0.375	0.500	s	*Section 6.6.21.2*
tNoResponse	4.5	5.0	5.5	s	*Section 6.6.6*
tPPSRequest			10	s	*Section 6.6.19.1*
tPPSTimeout	12.0	13.5	15.0	s	*Section 6.6.19.2*
tProtErrHardReset			15	ms	*Section 6.6.11.4*
tProtErrSoftReset			15	ms	*Section 6.6.9.2*
tPRSwapWait	100			ms	*Section 6.6.4.2*
tPSHardReset	25	30	35	ms	*Section 6.6.11.2*

tPSSourceOff	SPR Mode	750	835	920	ms	*Section 6.6.5.2*
	EPR Mode	1120	1260	1400		
tPSSourceOn	SPR Mode	390	435	480	ms	*Section 6.6.5.3*
tPSTransition	SPR Mode	450	500	550	ms	*Section 6.6.5.1*
	EPR Mode	830	925	1020		
tReceive		0.9	1.0	1.1	ms	*Section 6.6.1*
tReceiverResponse				15	ms	*Section 6.6.2*
tRetry				195	µs	*Section 6.6.1*
tSenderResponse		27	30	33	ms	*Section 6.6.2*
tSinkDelay				5	ms	*Section 5.7*
tSinkRequest		100			ms	*Section 6.6.4.1*
tSinkTx		16	18	20	ms	*Section 6.6.16*
tSoftReset				15	ms	*Section 6.8.1*
tSrcHoldsBus				50	ms	*Section 8.3.3.2*
tSwapSinkReady				15	ms	*Section 6.6.8.1*
tSwapSourceStart		20			ms	*Section 6.6.8.1*
tTransmit				195	µs	*Section 6.6.1*
tTypeCSendSourceCap		100	150	200	ms	*Section 6.6.3.1*
tTypeCSinkWaitCap		310	465	620	ms	*Section 6.6.3.2*
tVCONNSourceDischarge		160	200	240	ms	*Section 6.6.10.1*
tVCONNSourceOff				25	ms	*Section 6.6.13*
tVCONNSourceOn				50	ms	*Section 6.3.11*
tVCONNSourceTimeout		100	150	200	ms	*Section 6.6.13*
tVCONNSwapWait		100			ms	*Section 6.6.4.4*
tVDMBusy		50			ms	*Section 6.6.12.4*
tVDMEnterMode				25	ms	*Section 6.6.12.2*
tVDMExitMode				25	ms	*Section 6.6.12.3*
tVDMReceiverResponse				15	ms	*Section 6.6.12.1*
tVDMSenderResponse		24	27	30	ms	*Section 6.6.12.1*
tVDMWaitModeEntry		40	45	50	ms	*Section 6.6.12.2*
tVDMWaitModeExit		40	45	50	ms	*Section 6.6.12.3*

Table 6.70 "Timers"

Timer	Parameter	Used By	Reference
BISTContModeTimer	*tBISTContMode*	Policy Engine	*Section 6.6.7.2*
ChunkingNotSupportedTimer	*tChunkingNotSupported*	Policy Engine	*Section 6.6.18.1*
ChunkSenderRequestTimer	*tChunkSenderRequest*	Protocol	*Section 6.6.18.2*
ChunkSenderResponseTimer	*tChunkSenderResponse*	Protocol	*Section 6.6.18.3*
CRCReceiveTimer	*tReceive*	Protocol	*Section 6.6.1*
DataResetFailTimer	*tDataResetFail*	Policy Engine	*Section 6.6.10.3*
DataResetFailUFPTimer	*tDataResetFailUFP*	Policy Engine	*Section 6.6.10.4*
DiscoverIdentityTimer	*tDiscoverIdentity*	Policy Engine	*Section 6.6.15*
HardResetCompleteTimer	*tHardResetComplete*	Protocol	*Section 6.6.9*
NoResponseTimer	*tNoResponse*	Policy Engine	*Section 6.6.6*
PSHardResetTimer	*tPSHardReset*	Policy Engine	*Section 6.6.11.2*
PSSourceOffTimer	*tPSSourceOff*	Policy Engine	*Section 6.6.5.2*
PSSourceOnTimer	*tPSSourceOn*	Policy Engine	*Section 6.6.5.3*
PSTransitionTimer	*tPSTransition*	Policy Engine	*Section 6.6.5.1*
SenderResponseTimer	*tSenderResponse*	Policy Engine	*Section 6.6.2*
SinkEPREnterTimer	*tEnterEPR*	Policy Engine	*Section 6.6.21.1*
SinkEPRKeepAliveTimer	*tSinkEPRKeepAlive*	Policy Engine	*Section 6.6.21.2*
SinkPPSPeriodicTimer	*tPPSRequest*	Policy Engine	*Section 6.6.19.1*
SinkRequestTimer	*tSinkRequest*	Policy Engine	*Section 6.6.4*
SinkWaitCapTimer	*tTypeCSinkWaitCap*	Policy Engine	*Section 6.6.3.2*
SourceCapabilityTimer	*tTypeCSendSourceCap*	Policy Engine	*Section 6.6.3.1*
SourceEPRKeepAliveTimer	*tSourceEPRKeepAlive*	Policy Engine	*Section 6.6.21.3*
SourcePPSCommTimer	*tPPSTimeout*	Policy Engine	*Section 6.6.19.2*
SinkTxTimer	*tSinkTx*	Protocol Layer	*Section 6.6.16*
SwapSourceStartTimer	*tSwapSourceStart*	Policy Engine	*Section 6.6.8.1*
VCONNDischargeTimer	*tVCONNSourceDischarge*	Policy Engine	*Section 6.6.10.1*
VCONNOnTimer	*tVCONNSourceTimeout*	Policy Engine	*Section 6.6.13.1*
VDMModeEntryTimer	*tVDMWaitModeEntry*	Policy Engine	*Section 6.6.12.2*
VDMModeExitTimer	*tVDMWaitModeExit*	Policy Engine	*Section 6.6.12.3*
VDMResponseTimer	*tVDMSenderResponse*	Policy Engine	*Section 6.6.12.1*

6.7 Counters

6.7.1 MessageID Counter

The ***MessageIDCounter*** is a rolling counter, ranging from 0 to ***nMessageIDCount***, used to detect duplicate Messages. This value is used for the ***MessageID*** field in the Message Header of each transmitted Message.

Each Port ***Shall*** maintain a copy of the last ***MessageID*** value received from its Port Partner. Devices that support multiple ports, such as Hubs, ***Shall*** maintain copies of the last ***MessageID*** on a per Port basis. A Port which communicates using SOP* Packets ***Shall*** maintain copies of the last ***MessageID*** for each type of ***SOP**** it uses.

The transmitter ***Shall*** use the ***MessageID*** in a ***GoodCRC*** Message to verify that a particular Message was received correctly. The receiver ***Shall*** use the ***MessageID*** to detect duplicate Messages.

6.7.1.1 Transmitter Usage

The Transmitter ***Shall*** use the ***MessageID*** as follows:

- Upon receiving either ***Hard Reset*** Signaling, or a ***Soft_Reset*** Message, the transmitter ***Shall*** set its ***MessageIDCounter*** to zero and re-initialize its retry mechanism.
- If a ***GoodCRC*** Message with a ***MessageID*** matching the ***MessageIDCounter*** is not received before the ***CRCReceiveTimer*** expires, it ***Shall*** retry the same packet up to ***nRetryCount*** times using the same ***MessageID***.
- If a ***GoodCRC*** Message is received with a ***MessageID*** matching the current ***MessageIDCounter*** before the ***CRCReceiveTimer*** expires, the transmitter ***Shall*** re-initialize its retry mechanism and increment its ***MessageIDCounter***.
- If the Message is aborted by the Policy Engine, the transmitter ***Shall*** delete the Message from its transmit buffer, re-initialize its retry mechanism and increment its ***MessageIDCounter***.

6.7.1.2 Receiver Usage

The Receiver ***Shall*** use the ***MessageID*** as follows:

- When the first good packet is received after a reset, the receiver ***Shall*** store a copy of the received ***MessageID*** value.
- For subsequent Messages, if ***MessageID*** value in a received Message is the same as the stored value, the receiver ***Shall*** return a ***GoodCRC*** Message with that ***MessageID*** value and drop the Message (this is a retry of an already received Message). Note this ***Shall Not*** apply to the ***Soft_Reset*** Message which always has a ***MessageID*** value of zero.
- If ***MessageID*** value in the received Message is different than the stored value, the receiver ***Shall*** return a ***GoodCRC*** Message with the new ***MessageID*** value, store a copy of the new ***MessageID*** value and process the Message.

6.7.2 Retry Counter

The *RetryCounter* is used by a Port whenever there is a Message transmission failure (timeout of *CRCReceiveTimer*). If the *nRetryCount* retry fails, then the link ***Shall*** be reset using the Soft Reset mechanism.

The following rules apply to retries when there is a Message transmission failure (see also *Section 6.12.2.2 "Protocol Layer Message Transmission"*):

- Cable Plugs ***Shall Not*** retry Messages.
- Extended Messages of *Data Size* > *MaxExtendedMsgLegacyLen* that are not Chunked (*Chunked* flag set to zero) ***Shall Not*** be retried.
- Extended Messages of *Data Size* ≤ *MaxExtendedMsgLegacyLen* (*Chunked* flag set to zero or one) ***Shall*** be retried.
- Extended Messages of *Data Size* > *MaxExtendedMsgLegacyLen* that are Chunked (*Chunked* flag set to one) individual Chunks ***Shall*** be retried.

When messages are not retried, then the *RetryCounter* is not used. Higher layer protocols are expected to accommodate message delivery failure or failure to receive a *GoodCRC* Message.

6.7.3 Hard Reset Counter

The *HardResetCounter* is used to retry the Hard Reset whenever there is no response from the remote device (see *Section 6.6.6 "NoResponseTimer"*). Once the Hard Reset has been retried *nHardResetCount* times then it ***Shall*** be assumed that the remote device is non-responsive.

6.7.4 Capabilities Counter

The *CapsCounter* is used to count the number of *Source_Capabilities* Messages which have been sent by a Source at power up or after a Hard Reset. Implementation of the *CapsCounter* is ***Optional*** but ***May*** be used by any Source which wishes to preserve power by not sending *Source_Capabilities* Messages after a period of time.

When the *CapsCounter* is implemented and the Source detects that a Sink is Attached then after *nCapsCount* *Source_Capabilities* Messages have been sent the Source ***Shall*** decide that the Sink is non-responsive, stop sending *Source_Capabilities* Messages and disable PD.

A Sink ***Shall*** use the *SinkWaitCapTimer* to trigger the resending of *Source_Capabilities* Messages by a USB Power Delivery capable Source which has previously stopped sending *Source_Capabilities* Messages. Any Sink which is Attached and does not detect a *Source_Capabilities* Message, ***Shall*** issue *Hard Reset* Signaling when the *SinkWaitCapTimer* times out in order to reset the Source. Resetting the Source ***Shall*** also reset the *CapsCounter* and restart the sending of *Source_Capabilities* Messages.

6.7.5 Discover Identity Counter

When sending *Discover Identity* Messages to a Cable Plug a Port ***Shall*** maintain a count of Messages sent (*DiscoverIdentityCounter*). No more than *nDiscoverIdentityCount Discover Identity* Messages ***Shall*** be sent by the Port without receiving a *GoodCRC* Message response. A VCONN Swap ***Shall*** reset the *DiscoverIdentityCounter* to zero.

6.7.6 VDMBusyCounter

When sending Responder Busy responses to a Structured *Vendor_Defined* Message a UFP or Cable Plug ***Shall*** maintain a count of Messages sent (*VDMBusyCounter*). No more than *nBusyCount* Responder Busy responses ***Shall***

be sent. The *VDMBusyCounter* ***Shall*** be reset on sending a non-Busy response. Products wishing to meet *[USB Type-C 2.3]* requirements for Mode entry ***Should*** use an *nBusyCount* of 1.

6.7.7 Counter Values and Counters

Table 6.71 "Counter parameters" lists the counters used in this section and *Table 6.72 "Counters"* shows the corresponding parameters.

Table 6.71 "Counter parameters"

Parameter	Value	Reference
nBusyCount	5	*Section 6.7.6*
nCapsCount	50	*Section 6.7.4*
nDiscoverIdentityCount	20	*Section 6.7.5*
nHardResetCount	2	*Section 6.7.3*
nMessageIDCount	7	*Section 6.7.1*
nRetryCount	2	*Section 6.7.2*

Table 6.72 "Counters"

Counter	Max	Reference
CapsCounter	*nCapsCount*	*Section 6.7.4*
DiscoverIdentityCounter	*nDiscoverIdentityCount*	*Section 6.7.5*
HardResetCounter	*nHardResetCount*	*Section 6.7.3*
MessageIDCounter	*nMessageIDCount*	*Section 6.7.1*
RetryCounter	*nRetryCount*	*Section 6.7.2*
VDMBusyCounter	*nBusyCount*	*Section 6.7.6*

6.8 Reset

Resets are a necessary response to protocol or other error conditions. USB Power Delivery defines four different types of reset:

- Soft Reset, which resets protocol.
- Data Reset which resets the USB communications.
- Hard Reset which resets both the power supplies and protocol
- Cable Reset which resets the cable.

6.8.1 Soft Reset and Protocol Error

A ***Soft_Reset*** Message is used to cause a Soft Reset of protocol communication when this has broken down in some way. It ***Shall Not*** have any impact on power supply operation but is used to correct a Protocol Error occurring during an Atomic Message Sequence (AMS). The Soft Reset ***May*** be triggered by either Port Partner in response to the Protocol Error.

Protocol Errors are any unexpected Message during an AMS. If the first Message in an AMS has been passed to the Protocol Layer by the Policy Engine but has not yet been sent (i.e., a ***GoodCRC*** Message acknowledging the Message has not been received) when the Protocol Error occurs, the Policy Engine ***Shall Not*** issue a Soft Reset but ***Shall*** return to the ***PE_SNK_Ready*** or ***PE_SRC_Ready*** state and then process the incoming Message. If the incoming Message is an Unexpected Message received in the ***PE_SNK_Ready*** or ***PE_SRC_Ready*** state, the Policy Engine ***Shall*** issue a Soft Reset. If the Protocol Error occurs during an AMS this ***Shall*** lead to a Soft Reset in order to re-synchronize the Policy Engine state machines (see ***Section 8.3.3.4 "SOP Soft Reset and Protocol Error State Diagrams"***) except when the Voltage is transition when a Protocol Error ***Shall*** lead to a Hard Reset (see Section ***6.6.11.4 "tProtErrHardReset"*** and ***Section 8.3.3.2 "Policy Engine Source Port State Diagram"***). Details of AMS's can be found in Section 8.3.2.1.3 "Atomic Message Sequences".

An Unrecognized or Unsupported Message received in the ***PE_SNK_Ready*** or ***PE_SRC_Ready*** states, ***Shall Not*** cause a ***Soft_Reset*** Message to be generated but instead a ***Not_Supported*** Message ***Shall*** be generated.

A ***Soft_Reset*** Message ***Shall*** be sent regardless of the R_p value either ***SinkTxOk*** or ***SinkTxNG*** if it is the correct response in that state. Note this means that a ***Soft_Reset*** Message can be sent during an AMS regardless of the R_p value either ***SinkTxOk*** or ***SinkTxNG*** when responding to a Protocol Error.

Table 6.73 "Response to an incoming Message (except VDM)" and ***Table 6.74 "Response to an incoming VDM"*** summarize the responses that ***Shall*** be made to an incoming Message including VDMs.

Table 6.73 "Response to an incoming Message (except VDM)"

Recipient's Power Role	Recipient's state	Incoming Message			
		Recognized			Unrecognized
		Supported		Unsupported	
		Expected	Unexpected		
Source	*PE_SRC_Ready*	Process Message	*Soft_Reset* Message[2]	*Not_Supported* Message[3]	*Not_Supported* Message[3] (except for VDM) See 6.4.4.1 for UVDM, 6.4.4.2 for SVDM
	During AMS (power not transitioning[1])	Process Message	*Soft_Reset* Message[2]		
	During AMS (power transitioning[1])	Process Message	*Hard Reset* Signaling		
Sink	*PE_SNK_Ready*	Process Message	*Soft_Reset* Message[2]	*Not_Supported* Message[3]	*Not_Supported* Message[3] (except for VDM) See 6.4.4.1 for UVDM, 6.4.4.2 for SVDM
	During AMS (not power transitioned)	Process Message	*Soft_Reset* Message[2]		
	During AMS (power transitioned)	Process Message	*Hard Reset* Signaling		

1) Power transitioning" means the policy engine is in *PE_SRC_Transition_Supply* State or *PE_SNK_Transition_Sink* State or *PE_FRS_SNK_SRC_Send_Swap* State.

2) The *Soft_Reset* Message ***Shall*** be sent using the *SOP** of the incoming message.

3) The *Not_Supported* Message ***Shall*** be sent using the *SOP** of the incoming message.

Table 6.74 "Response to an incoming VDM"

Recipient's Role	Supported UVDM	Unsupported UVDM	Unrecognized UVDM	Supported SVDM	Unsupported SVDM	Unrecognized SVDM
DFP or UFP	Defined by vendor	*Not_Supported* Message	*Not_Supported* Message	See *Section 6.13.5*	*Not_Supported* Message	NAK Command
Cable Plug	Defined by vendor	Message ***Ignored***	Message ***Ignored***	See *Section 6.13.5*	Message ***Ignored***	NAK Command

A failure to see a *GoodCRC* Message in response to any Message within *tReceive* (after *nRetryCount* retries), when a Port Pair is Connected, is indicative of a communications failure resulting in a Soft Reset (see *Section 6.6.9.1 "tSoftReset"*).

A Soft Reset ***Shall*** impact the USB Power Delivery layers in the following ways:

- Physical Layer: Reset not required since the Physical Layer resets on each packet transmission/reception.
- Protocol Layer: Reset ***MessageIDCounter***, ***RetryCounter*** and state machines.
- Policy Engine: Reset state dependent behavior by performing an Explicit Contract negotiation. Note when in SPR Mode the Source sends a ***Source_Capabilities*** Message and when in EPR Mode the Source sends an ***EPR_Source_Capabilities*** Message.
- Power supply: ***Shall Not*** change.

A Soft Reset is performed using a sequence of protocol Messages (see ***Table 8.9 "AMS: Soft Reset"***). Message numbers ***Shall*** be set to zero prior to sending the ***Soft_Reset***/***Accept*** Message since the issue might be with the counters. The sender of a ***Soft_Reset*** Message ***Shall*** reset its ***MessageIDCounter*** and ***RetryCounter***, the receiver of the Message ***Shall*** reset its ***MessageIDCounter*** and ***RetryCounter*** before sending the ***Accept*** Message response. Any failure in the Soft Reset process will trigger a Hard Reset when SOP Packets are being used or Cable Reset, sent by the DFP only, for any other SOP* Packets; for example a ***GoodCRC*** Message is not received during the Soft Reset process (see ***Section 6.8.3 "Hard Reset"*** and ***Section 6.8.4 "Cable Reset"***).

6.8.2 Data Reset

A ***Data_Reset*** Message is used by a Port to reset its USB data connection and to exit all Alternate Modes both with its Port Partner and in the Cable Plug(s).

- The Data Reset process ***May*** be initiated by either Port Partner sending a ***Data_Reset*** Message.

A Data Reset impacts USB Power Delivery in the following ways:

- ***Shall Not*** change the Port Power Roles (Source/Sink) or Port Data Roles (DFP/UFP).
- ***Shall Not*** change the existing Explicit Contract.
- ***Shall*** cause all *Active Mode*s to be exited.
- ***Shall*** reset the cable by Power cycling VCONN.
- The DFP ***Shall*** become the VCONN Source.
- If the Data Reset process fails, then the Port ***Shall*** enter the ***ErrorRecovery*** State as defined in ***[USB Type-C 2.3]***.

See ***Section 6.3.14 "Data_Reset Message"*** for details of Data Reset operation.

6.8.3 Hard Reset

Hard Resets are signaled by an ordered set as defined in ***Section 5.6.4 "Hard Reset"***. Both the sender and recipient ***Shall*** cause their power supplies to return to their default states (see ***Section 7.3.4.1 "Source Initiated Hard Reset"*** and ***Section 7.3.4.2 "Sink Initiated Hard Reset"*** for details of Voltage transitions). In addition, their respective Protocol Layers ***Shall*** be reset as for the Soft Reset. This allows the Attached devices to be in a state where they can re-establish USB PD communication. Hard Reset is retried up to ***nHardResetCount*** times (see also ***Section 6.6.6 "NoResponseTimer"*** and ***Section 6.7.3 "Hard Reset Counter"***). Note that even though V_{BUS} drops to ***vSafe0V*** during a Hard Reset a Sink will not see this as a disconnect since this is expected behavior.

A Hard Reset ***Shall Not*** cause any change to either the R_p/R_d resistor being asserted.

If there has been a Data Role Swap the Hard Reset ***Shall*** cause the Port Data Role to be changed back to DFP for a Port with the R_p resistor asserted and UFP for a Port with the R_d resistor asserted.

When VCONN is supported (see ***[USB Type-C 2.3]***) the Hard Reset ***Shall*** cause the Port with the R_p resistor asserted to supply VCONN and the Port with the R_d resistor asserted to turn off VCONN.

In effect the Hard Reset will revert the Ports to their default state based on their CC line resistors. Removing and reapplying VCONN from the Cable Plugs also ensures that they re-establish their configuration as either SOP' or SOP'' based on the location of VCONN (see ***[USB Type-C 2.3]***).

If the Hard Reset is insufficient to clear the error condition, then the Port ***Shall*** use USB Type-C® ErrorRecovery as defined in ***[USB Type-C 2.3]***.

A Sink ***Shall*** be able to send ***Hard Reset*** signaling regardless of the value of R_p (see ***Section 5.7 "Collision Avoidance"***).

6.8.3.1 Cable Plugs and Hard Reset

Cable Plugs ***Shall Not*** generate ***Hard Reset*** Signaling but ***Shall*** monitor for ***Hard Reset*** Signaling between the Port Partners and ***Shall*** reset when this is detected (see ***Section 8.3.3.26.2.2 "Cable Plug Hard Reset State Diagram"***). The Cable Plugs ***Shall*** perform the equivalent of a power cycle returning to their initial power up state. This allows the Port Partners to be in a state where they can re-establish USB PD communication.

6.8.3.2 Modal Operation and Hard Reset

A Hard Reset ***Shall*** cause EPR Mode and all *Active Mode*s to be exited by both Port Partners and any Cable Plugs (see ***Section 6.4.4.3.4 "Enter Mode Command"***).

6.8.4 Cable Reset

Cable Resets are signaled by an ordered set as defined in ***Section 5.6.5 "Cable Reset"***. Both the sender and recipient of ***Cable Reset*** Signaling ***Shall*** reset their respective Protocol Layers. The Cable Plugs ***Shall*** perform the equivalent of a power cycle returning to their initial power up state. This allows the Port Partners to be in a state where they can re-establish USB PD communication.

The DFP must be supplying VCONN prior to a Cable Reset. If VCONN has been turned off the DFP ***Shall*** turn on VCONN prior to generating ***Cable Reset*** Signaling. If there has been a VCONN Swap and the UFP is currently supplying VCONN, the DFP ***Shall*** perform a VCONN Swap such that it is supplying VCONN prior to generating ***Cable Reset*** Signaling.

Only a DFP ***Shall*** generate ***Cable Reset*** Signaling. A DFP ***Shall*** only generate ***Cable Reset*** Signaling within an Explicit Contract.

A Cable Reset ***Shall*** cause all *Active Mode*s in the Cable Plugs to be exited (see ***Section 6.4.4.3.4 "Enter Mode Command"***).

6.9 Accept, Reject and Wait

The recipient of a ***Request***, ***EPR_Request***, ***PR_Swap***, ***DR_Swap*** , ***VCONN_Swap***, or ***Enter_USB*** Message Shall respond by sending one of the following responses:

- an ***Accept*** Message in response to a ***Valid*** request which can be serviced immediately (see ***Section 6.3.3 "Accept Message"***).
- a ***Wait*** Message in response to a ***Valid*** request which cannot be serviced immediately but could be serviced at a later time (see ***Section 6.3.12 "Wait Message"***). Note a ***Wait*** Message is not a ***Valid*** response to an ***EPR_Request*** Message.
- a ***Reject*** Message in response to an ***Invalid*** request or a request which is outside of the device's design capabilities (see ***Section 6.3.4 "Reject Message"***).

6.10 Collision Avoidance

To avoid message collisions due to asynchronous Messaging sent from the Sink, the Source sets R_p to ***SinkTxOk*** to indicate to the Sink that it is ok to initiate an AMS. When the Source wishes to initiate an AMS, it sets R_p to ***SinkTxNG***. When the Sink detects that R_p is set to ***SinkTxOk*** it ***May*** initiate an AMS. When the Sink detects that R_p is set to ***SinkTxNG*** it ***Shall Not*** initiate an AMS and ***Shall*** only send Messages that are part of an AMS the Source has initiated. Note that this restriction applies to SOP* AMS's i.e., for both Port to Port and Port to Cable Plug communications.

If a transition into the ***PE_SRC_Ready*** state will result in an immediate transition out of the ***PE_SRC_Ready*** state within ***tSrcHoldsBus*** e.g. it is due to a Protocol Error that has not resulted in a Soft Reset, then the notifications of the end of AMS and first Message in an AMS ***May Not*** be sent to avoid changing the R_p value unnecessarily.

Note: A Sink can still send ***Hard Reset*** signaling at any time.

6.11 Message Discarding

On receiving a received Message on SOP (except for a ***Ping*** Message), the Protocol Layer ***Shall Discard*** any pending SOP* Messages. A received Message on SOP'/SOP'' ***Shall Not*** cause any pending SOP* Messages to be ***Discarded***.

It is assumed that Messages using SOP'/SOP'' constitute a simple request/response AMS, with the Cable Plug providing the response so there is no reason for a pending SOP* Message to be ***Discarded***. There can only be one AMS between the Port Partners, and these also take priority over Cable Plug communications so a Message received on SOP will always cause a Message pending on SOP* to be ***Discarded***.

See ***Table 6.75 "Message discarding"*** for details of the Messages that ***Shall***/ ***Shall Not*** be Discarded.

Table 6.75 “Message discarding”

Message pending transmission	Message received	Message to be Discarded
SOP	SOP	Outgoing message
SOP	SOP’/SOP’’	Incoming message
SOP’	SOP	Outgoing message
SOP’	SOP’	Incoming message
SOP’	SOP’’	Incoming message
SOP’’	SOP	Outgoing message
SOP’’	SOP’	Incoming message
SOP’’	SOP’’	Incoming message

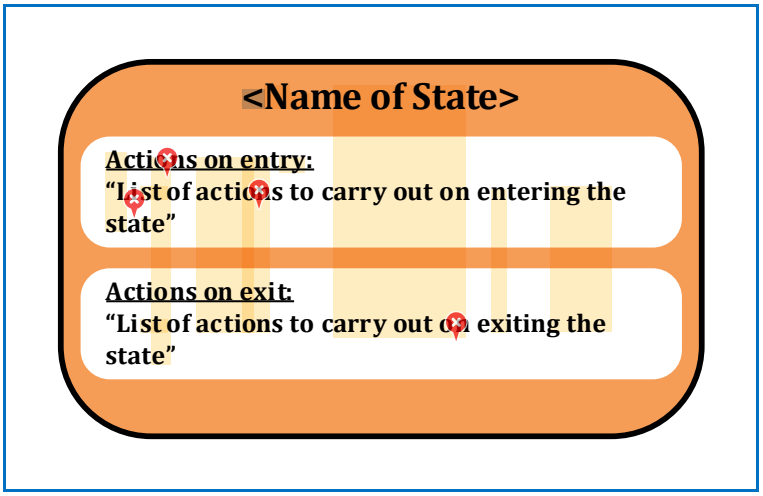
6.12 State behavior

6.12.1 Introduction to state diagrams used in Chapter 6

The state diagrams defined in *Section 6.12 "State behavior"* are ***Normative*** and ***Shall*** define the operation of the Power Delivery protocol layer. Note that these state diagrams are not intended to replace a well written and robust design.

Figure 6-55 "Outline of States" shows an outline of the states defined in the following sections. At the top there is the name of the state. This is followed by "Actions on entry" a list of actions carried out on entering the state and in some states "Actions on exit" a list of actions carried out on exiting the state.

Figure 6-55 "Outline of States"

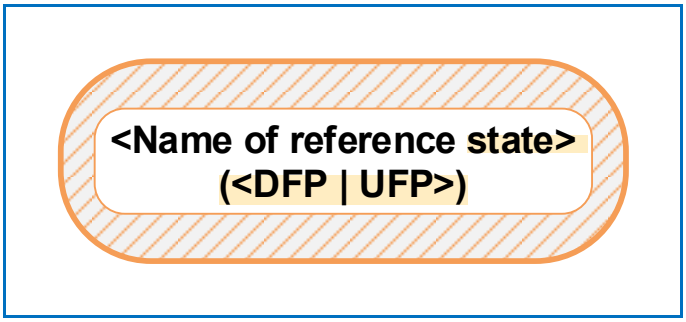


Transitions from one state to another are indicated by arrows with the conditions listed on the arrow. Where there are multiple conditions, these are connected using either a logical OR "|" or a logical AND "&." The inverse of a condition is shown with a "NOT" in front of the condition.

In some cases, there are transitions which can occur from any state to a particular state. These are indicated by an arrow which is unconnected to a state at one end, but with the other end (the point) connected to the final state.

In some state diagrams it is necessary to enter or exit from states in other diagrams. *Figure 6-56 "References to states"* indicates how such references are made. The reference is indicated with a hatched box. The box contains the name of the referenced state.

Figure 6-56 "References to states"



Timers are included in many of the states. Timers are initialized (set to their starting condition) and run (timer is counting) in the state it is referenced. As soon as the state is exited then the timer is no longer active. Timeouts of the timers are listed as conditions on state transitions.

Conditions listed on state transitions will come from one of three sources:

- Messages received from the PHY Layer.
- Events triggered within the Protocol Layer e.g., timer timeouts
- Message and related indications passed up to the Policy Engine from the Protocol Layer (Message sent; Message received etc.)

6.12.2 State Operation

The following section details Protocol Layer State Operation when sending and receiving SOP* Packets.

For each SOP* Communication being sent and received there ***Shall*** be separate Protocol Layer Transmission and Protocol Layer Reception and Hard Reset State Machine instances, with their own counter and timer instances. When Chunking is supported there ***Shall*** be separate Chunked Tx, Chunked Tx, and Chunked Message Router State Machine instances.

Soft Reset ***Shall*** only apply to the State Machine instances it is targeted at based on the type of SOP* Packet used to send the ***Soft_Reset*** Message. The Hard-Reset State Machine (including Cable Reset) ***Shall*** apply simultaneously to all Protocol Layer State Machine instances active in the DFP, UFP and Cable Plug (if present).

6.12.2.1 Protocol Layer Chunking

6.12.2.1.1 Architecture of Device Including Chunking Layer

The Chunking component resides in the Protocol Layer between the Policy Engine and Protocol Tx/Rx. ***Figure 6-57 "Chunking architecture Showing Message and Control Flow"*** illustrates the relationship between components.

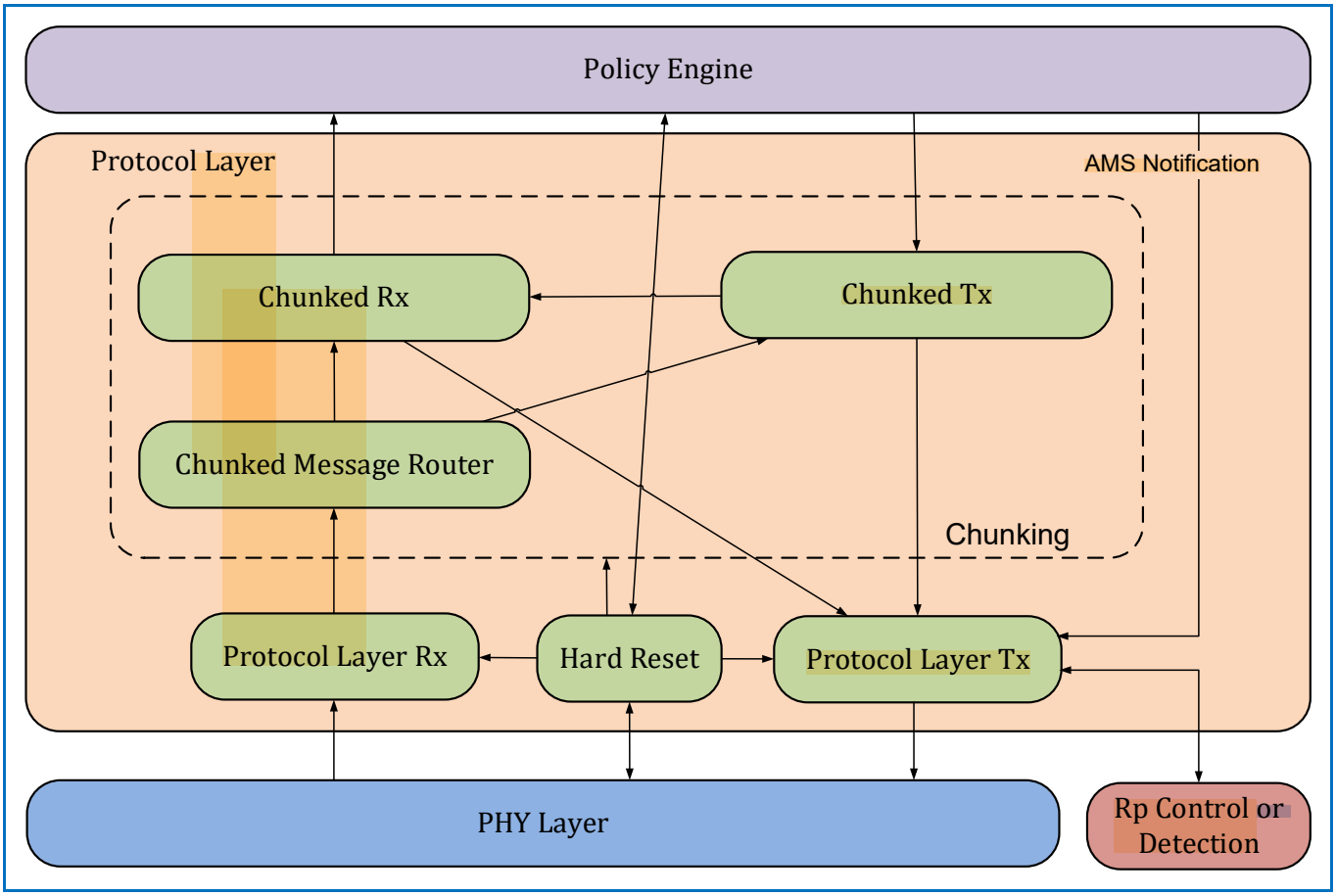
The Chunking Layer comprises three related state machines:

- Chunked Rx.
- Chunked Tx.
- Chunked Message Router.

Note that the consequence of this architecture is that the Policy Engine deals entirely in un-chunked messages. It will not receive (and might not respond to) a message until all the related chunks have been collated.

If a PD Device or Cable Marker has no requirement to handle any message requiring more than one Chunk of any Extended Message, it ***May*** omit the Chunking Layer. In this case it ***Shall*** implement the ***ChunkingNotSupportedTimer*** to ensure compatible operation with partners which support Chunking (see ***Section 6.6.18.1 "ChunkingNotSupportedTimer"*** and ***Section 8.3.3.6 "Not Supported Message State Diagrams"***).

Figure 6-57 "Chunking architecture Showing Message and Control Flow"



6.12.2.1.1.1 Optional Abort Mechanism

Long Chunked Messages bring with them the potential problem that they could prevent urgent messages from being transmitted in a timely manner. An ***Optional*** Abort mechanism is provided to remedy this problem.

The Abort Flag referred to in the diagrams below ***May*** be set and examined by the Policy Engine. The specific means are left to the implementer.

6.12.2.1.1.2 Aborting Sending a Long-Chunked Message

A long-Chunked Message being sent ***May*** be aborted by setting the ***Optional*** Abort Flag. The message ***Shall*** be considered aborted when the Abort Flag is again cleared by the Chunked Tx state machine.

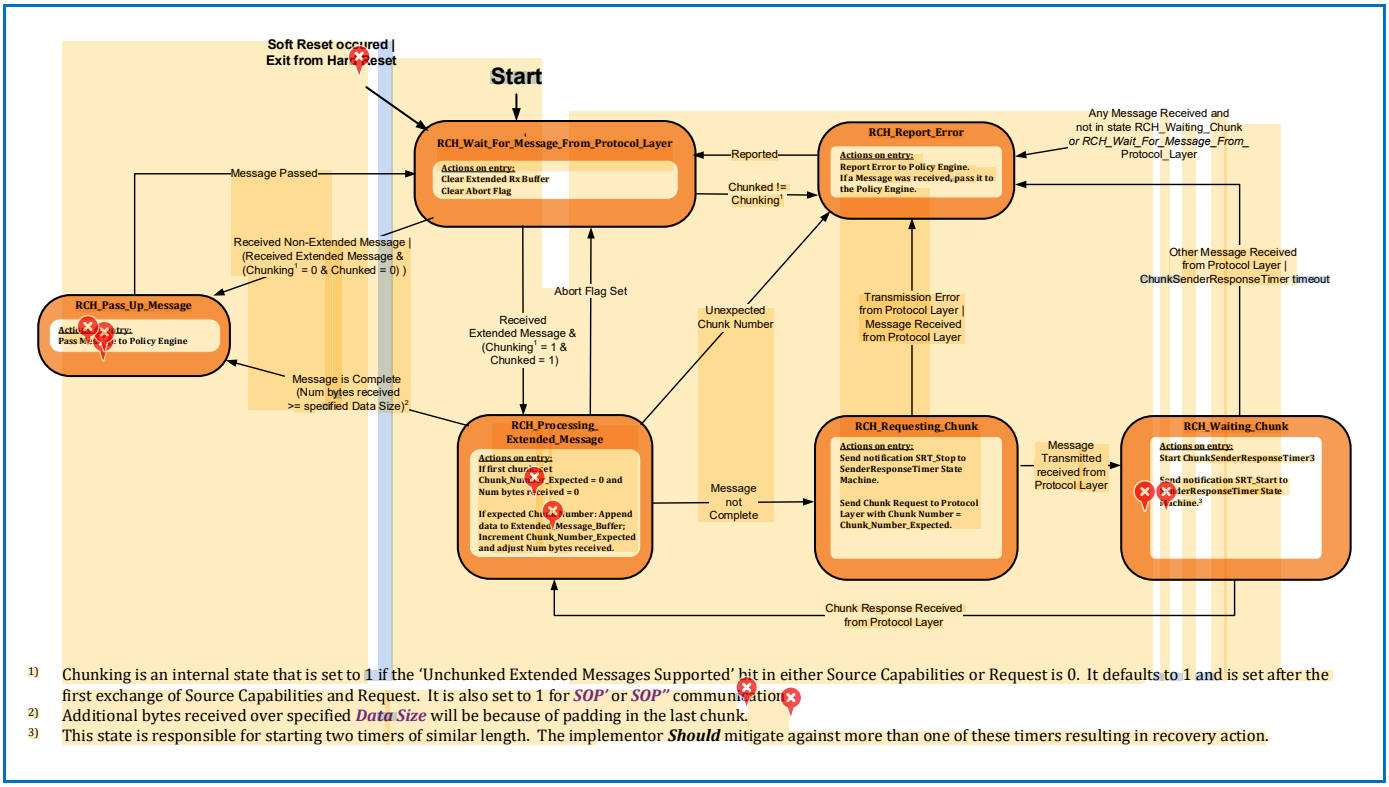
6.12.2.1.1.3 Aborting Receiving a Long-Chunked Message

If the ***Optional*** Abort mechanism has been implemented, any message sent while a Chunked Message receive is in progress will result in an error report being received by the Policy Engine, to indicate that the message request has been ***Discarded***. If the message was urgent the Policy Engine might set the Abort Flag, which will result in the incoming Chunked Message being aborted. The Abort Flag being cleared by the Chunked Rx state machine indicates that the urgent message can now be sent.

6.12.2.1.2 Chunked Rx State Diagram

Figure 6-58 "Chunked Rx State Diagram" shows the state behavior for the Chunked Rx State Machine. This recognizes whether chunked received messages are involved and deals with requesting chunks when they are. It also performs validity checks on all messages related to chunking.

Figure 6-58 "Chunked Rx State Diagram"



1) Chunking is an internal state that is set to 1 if the 'Unchunked Extended Messages Supported' bit in either Source Capabilities or Request is 0. It defaults to 1 and is set after the first exchange of Source Capabilities and Request. It is also set to 1 for *SOP'* or *SOP"* communication.
2) Additional bytes received over specified *Data Size* will be because of padding in the last chunk.
3) This state is responsible for starting two timers of similar length. The implementor ***Should*** mitigate against more than one of these timers resulting in recovery action.

6.12.2.1.2.1 RCH_Wait_For_Message_From_Protocol_Layer State

The Chunked Rx State Machine ***Shall*** enter the *RCH_Wait_For_Message_From_Protocol_Layer* state:

- At startup.
- As a result of a Soft Reset occurring.
- On exit from a Hard Reset.

On entry to the *RCH_Wait_For_Message_From_Protocol_Layer* state the Chunked Rx state machine clears the Extended Rx Buffer and clears the ***Optional*** Abort Flag.

In the *RCH_Wait_For_Message_From_Protocol_Layer* state the Chunked Rx state machine waits until the Chunked Message Router passes up a received message.

The Chunked Rx State Machine ***Shall*** transition to the *RCH_Pass_Up_Message* state when:

- A non-Extended Message is passed up from the Chunked Message Router.
- An Extended Message is passed up from the Chunked Message Router, and the Policy Engine has determined that we are not doing Chunking, and the Message has its ***Chunked*** bit set to 0b.

The Chunked Rx State Machine ***Shall*** transition to the *RCH_Processing_Extended_Message* state when:

- An Extended Message is passed up from the Chunked Message Router, and the Policy Engine has determined that we are doing Chunking, and the Message has its ***Chunked*** bit set to 1b.

6.12.2.1.2.2 RCH_Pass_Up_Message State

On entry to the *RCH_Pass_Up_Message* state the Chunked Rx state machine ***Shall*** pass the received message to the Policy Engine.

The Chunked Rx State Machine ***Shall*** transition to the *RCH_Wait_For_Message_From_Protocol_Layer* state when:

- The Message has been passed.

6.12.2.1.2.3 RCH_Processing_Extended_Message State

On entry to the *RCH_Processing_Extended_Message* state the Chunked Rx state machine ***Shall***:

- If this is the first chunk:
- Set Chunk_Number_Expected = 0.
- Set Num bytes received = 0.
- If chunk contains the expected Chunk Number:
- Append its data to the Extended_Message_Buffer.
- Increment Chunk_Number_Expected.
- Adjust Num bytes received.

The Chunked Rx State Machine ***Shall*** transition to the *RCH_Pass_Up_Message* state when:

- The message is complete (i.e., Num bytes received >= specified ***Data Size***. Note that the inequality allows for padding bytes in the last chunk, which are not actually part of the extended message).

The Chunked Rx State Machine ***Shall*** transition to the *RCH_Requesting_Chunk* state when:

- The Message is not yet complete.

The Chunked Rx State Machine ***Shall*** transition to the *RCH_Report_Error* state when:

- An unexpected Chunk Number is received.

The Chunked Rx State Machine ***Shall*** transition to the *RCH_Wait_For_Message_From_Protocol_Layer* state when:

- The ***Optional*** Abort Flag is set.

6.12.2.1.2.4 RCH_Requesting_Chunk State

On entry to the *RCH_Requesting_Chunk* state the Chunked Rx state machine ***Shall***:

- Send notification SRT_Stop to ***SenderResponseTimer*** state machine (see ***Section 8.3.3.1.1 "SenderResponseTimer State Diagram"***).
- Send Chunk Request to Protocol Layer with ***Chunk Number*** = Chunk_Number_Expected.

The Chunked Rx State Machine ***Shall*** transition to the *RCH_Waiting_Chunk* state when:

- Message Transmitted is received from the Protocol Layer.

The Chunked Rx State Machine ***Shall*** transition to the *RCH_Report_Error* state when:

- Transmission Error is received from the Protocol Layer, or

- A Message is received from the Protocol Layer.

6.12.2.1.2.5 RCH_Waiting_Chunk State

On entry to the *RCH_Waiting_Chunk* state the Chunked Rx state machine ***Shall***:

- Start the *ChunkSenderResponseTimer*.
- Send notification SRT_Start to *SenderResponseTimer* state machine (see *Section 8.3.3.1.1 "SenderResponseTimer State Diagram"*).

The Chunked Rx State Machine ***Shall*** transition to the *RCH_Processing_Extended_Message* state when:

- A Chunk is received from the Protocol Layer.

The Chunked Rx State Machine ***Shall*** transition to the *RCH_Report_Error* state when:

- A Message, other than a Chunk, is received from the Protocol Layer, or
- The *ChunkSenderResponseTimer* expires.

6.12.2.1.2.6 RCH_Report_Error State

The Chunked Rx State Machine ***Shall*** enter the *RCH_Report_Error* state:

- When any Message is received and the Chunked Rx State Machine is not in one of the states *RCH_Waiting_Chunk* or *RCH_Wait_For_Message_From_Protocol_Layer.*

On entry to the *RCH_Report_Error* state the Chunked Rx state machine ***Shall***:

- Report the error to the Policy Engine.
- If the state was entered because a Message was received, this Message ***Shall*** be passed to the Policy Engine.

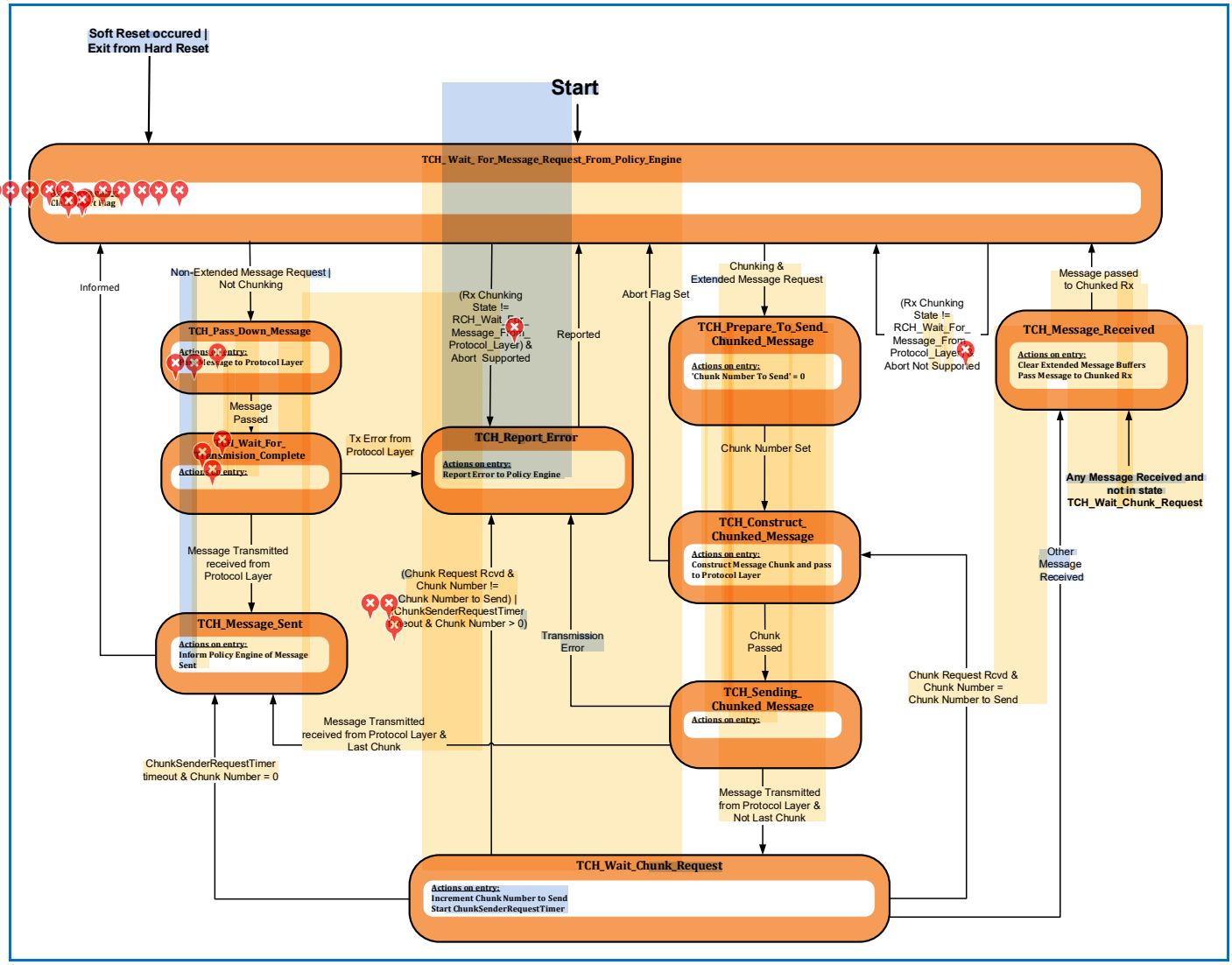
The Chunked Rx State Machine ***Shall*** transition to the *RCH_Wait_For_Message_From_Protocol_Layer* state when:

- The error has been reported.
- Any message received was passed to the Policy Engine.

6.12.2.1.3 Chunked Tx State Diagram

Figure 6-59 "Chunked Tx State Diagram" shows the state behavior for the Chunked Tx State Machine. This recognizes whether chunked transmitted messages are involved and deals with sending chunks and waiting for chunk requests when they are. It also performs validity checks on all related messages related to chunking.

Figure 6-59 "Chunked Tx State Diagram"



6.12.2.1.3.1 TCH_Wait_For_Message_Request_From_Policy_Engine State

The Chunked Tx State Machine ***Shall*** enter the *TCH_Wait_For_Message_Request_From_Policy_Engine* state:

- At startup.
- As a result of a Soft Reset occurring.
- On exit from a Hard Reset.

On entry to the *TCH_Wait_For_Message_Request_From_Policy_Engine* state the Chunked Tx state machine clears the ***Optional*** Abort Flag.

In the *TCH_Wait_For_Message_Request_From_Policy_Engine* state the Chunked Tx State Machine waits until the Policy Engine sends it a Message Request.

The Chunked Tx State Machine ***Shall*** transition to the *TCH_Pass_Down_Message* state when:

- A non-Extended Message Request is received from the Policy Engine, or
- A Message Request is received from the Policy Engine and the link is not Chunking.

The Chunked Tx State Machine ***Shall*** transition to the *TCH_Prepare_To_Send_Chunked_Message* state when:

- An Extended Message Request is received from the Policy Engine, and the link is Chunking.

The Chunked Tx State Machine ***Shall Discard*** the Message Request and remain in the *TCH_Wait_For_Message_Request_From_Policy_Engine* state when:

- The Chunked Rx state is any other than *RCH_Wait_For_Message_From_Protocol_Layer*, and the ***Optional*** Abort Flag has not been implemented.

The Chunked Tx State Machine ***Shall Discard*** the Message Request and enter the *TCH_Report_Error* state when:

- The Chunked Rx state is any other than *RCH_Wait_For_Message_From_Protocol_Layer* and the ***Optional*** Abort Flag has been implemented.

6.12.2.1.3.2 TCH_Pass_Down_Message State

On entry to the *TCH_Pass_Down_Message* state the Chunked Tx State Machine ***Shall*** pass the message to the Protocol Layer.

The Chunked Tx State Machine ***Shall*** transition to the *TCH_Wait_For_Transmision_Complete* state when:

- The message has been passed to the Protocol Layer.

6.12.2.1.3.3 TCH_Wait_For_Transmision_Complete State

The Chunked Tx State Machine ***Shall*** transition to the *TCH_Message_Sent* state when:

- Message Transmitted has been received from the Protocol Layer.

The Chunked Tx State Machine ***Shall*** transition to the *TCH_Report_Error* state when:

- Transmission Error has been received from the Protocol Layer.

6.12.2.1.3.4 TCH_Message_Sent State

On entry to the *TCH_Message_Sent* state the Chunked Tx State Machine ***Shall***:

- Inform the Policy Engine that the Message has been sent.

The Chunked Tx State Machine ***Shall*** transition to the *TCH_Wait_For_Message_Request_From_Policy_Engine* state when:

- The Policy Engine has been informed.

6.12.2.1.3.5 TCH_Prepare_To_Send_Chunked_Message State

On entry to the *TCH_Prepare_To_Send_Chunked_Message* state the Chunked Tx State Machine ***Shall***:

- Set 'Chunk Number To Send' to zero.

The Chunked Tx State Machine ***Shall*** transition to the *TCH_Construct_Chunked_Message* state when:

- 'Chunk Number To Send' has been set to zero.

6.12.2.1.3.6 TCH_Construct_Chunked_Message State

On entry to the *TCH_Construct_Chunked_Message* state the Chunked Tx State Machine ***Shall***:

- Construct a Message Chunk and pass it to the Protocol Layer.

The Chunked Tx State Machine ***Shall*** transition to the *TCH_Sending_Chunked_Message* state when:

- The Message Chunk has been passed to the Protocol Layer.

The Chunked Tx State Machine ***Shall*** transition to the *TCH_Wait_For_Message_Request_From_Policy_Engine* state when:

- The ***Optional*** Abort Flag is set.

6.12.2.1.3.7 TCH_Sending_Chunked_Message State

The Chunked Tx State Machine ***Shall*** transition to the *TCH_Wait_Chunk_Request* state when:

- Message Transmitted is received from Protocol Layer and this was not the last chunk.

The Chunked Tx State Machine ***Shall*** transition to the *TCH_Message_Sent* state when:

- Message Transmitted is received from Protocol Layer and this was the last chunk.

The Chunked Tx State Machine ***Shall*** transition to the *TCH_Report_Error* state when:

- Transmission Error has been received from the Protocol Layer.

6.12.2.1.3.8 TCH_Wait_Chunk_Request State

On entry to the *TCH_Wait_Chunk_Request* state the Chunked Tx State Machine ***Shall***:

- Increment Chunk Number to Send.
- Start *ChunkSenderRequestTimer.*

The Chunked Tx State Machine ***Shall*** transition to the *TCH_Report_Error* state when:

- A Chunk Request has been received and the Chunk Number does not equal Chunk Number to Send) or
- *ChunkSenderRequestTimer* has expired and Chunk Number is greater than zero.

The Chunked Tx State Machine ***Shall*** transition to the *TCH_Message_Sent* state when:

- *ChunkSenderRequestTimer* has expired and Chunk Number equals zero.

Note that this is the mechanism which allows the remote port partner or cable marker to omit the chunking layer. The Policy Engine will receive a Message Sent signal if the remote port partner or cable marker is present (*GoodCRC* Message received) but does not send a Chunk Request. After this the remote port partner will send a *Not_Supported* Message, or the Cable Marker will ***Ignore*** the Chunked Message.

The Chunked Tx State Machine ***Shall*** transition to the *TCH_Message_Received* state when:

- Any other message than Chunk Request is received.

6.12.2.1.3.9 TCH_Message_Received State

The Chunked Tx State Machine ***Shall*** enter the *TCH_Message_Received* state:

- When any Message is received, and the Chunked Tx State Machine is not in the *TCH_Wait_Chunk_Request* state.

On entry to the *TCH_Message_Received* state the Chunked Tx State Machine ***Shall***:

- Clear the Extended Message Buffers.
- Pass the received Message to Chunked Rx Engine.

The Chunked Tx State Machine ***Shall*** transition to the *TCH_Wait_For_Message_Request_From_Policy_Engine* state when:

- The received message has been passed to the Chunked Rx Engine.

6.12.2.1.3.10 TCH_Report_Error State

On entry to the *TCH_Report_Error* state the Chunked Tx State Machine ***Shall***:

- Report the error to the Policy Engine.

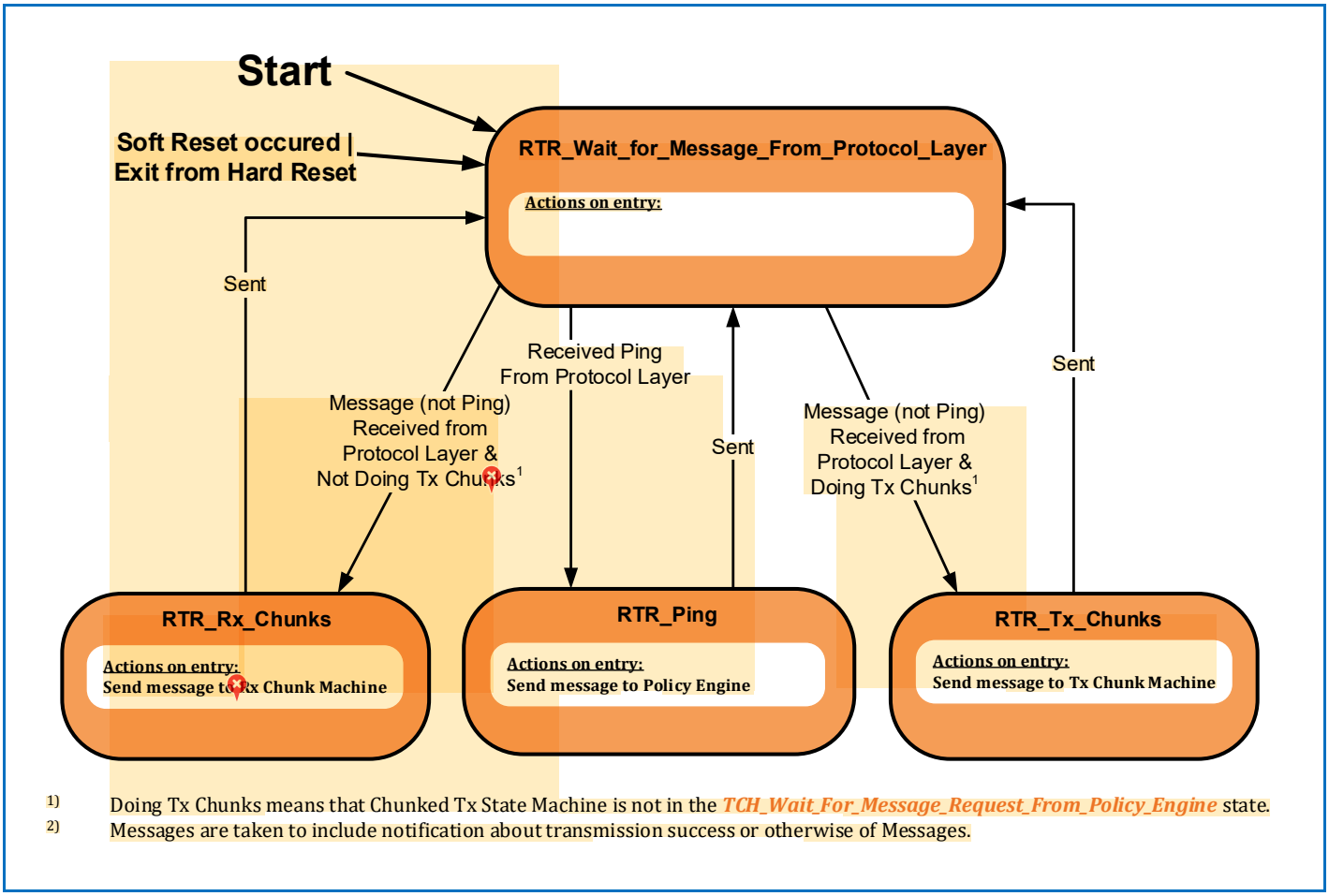
The Chunked Tx State Machine ***Shall*** transition to the *TCH_Wait_For_Message_Request_From_Policy_Engine* state when:

- The error has been reported.

6.12.2.1.4 Chunked Message Router State Diagram

Figure 6-60 "Chunked Message Router State Diagram" shows the state behavior for the Chunked Message Router. This determines to which state machine an incoming message is routed to (Chunked Rx, Chunked Tx or direct to Policy Engine).

Figure 6-60 "Chunked Message Router State Diagram"



1) Doing Tx Chunks means that Chunked Tx State Machine is not in the *TCH_Wait_For_Message_Request_From_Policy_Engine* state.
2) Messages are taken to include notification about transmission success or otherwise of Messages.

6.12.2.1.4.1 RTR_Wait_for_Message_From_Protocol_Layer State

In the *RTR_Wait_for_Message_From_Protocol_Layer* state the Chunked Message Router waits until the Protocol Layer sends it a received Message.

The Chunked Message Router ***Shall*** transition to the *RTR_Rx_Chunks* state when:

- A Message other than a *Ping* Message is received from the Protocol Layer, and the combined Chunking is not doing Tx Chunks.

The Chunked Message Router ***Shall*** transition to the *RTR_Tx_Chunks* state when:

- A Message other than a *Ping* Message is received from the Protocol Layer, and the combined Chunking is doing Tx Chunks.

The Chunked Message Router ***Shall*** transition to the *RTR_Ping* state when:

- A *Ping* Message is received from the Protocol Layer.

6.12.2.1.4.2 RTR_Rx_Chunks State

On entry to the *RTR_Rx_Chunks* state the Chunked Message Router ***Shall***:

- Send the message to the Chunked Rx State Machine.
- Transition to the *RTR_Wait_for_Message_From_Protocol_Layer* state.

6.12.2.1.4.3 RTR_Ping State

On entry to the *RTR_Ping* state the Chunked Message Router ***Shall***:

- Send the message to the Policy Engine.
- Transition to the *RTR_Wait_for_Message_From_Protocol_Layer* state.

6.12.2.1.4.4 RTR_Tx_Chunks State

On entry to the *RTR_Tx_Chunks* state the Chunked Message Router ***Shall***:

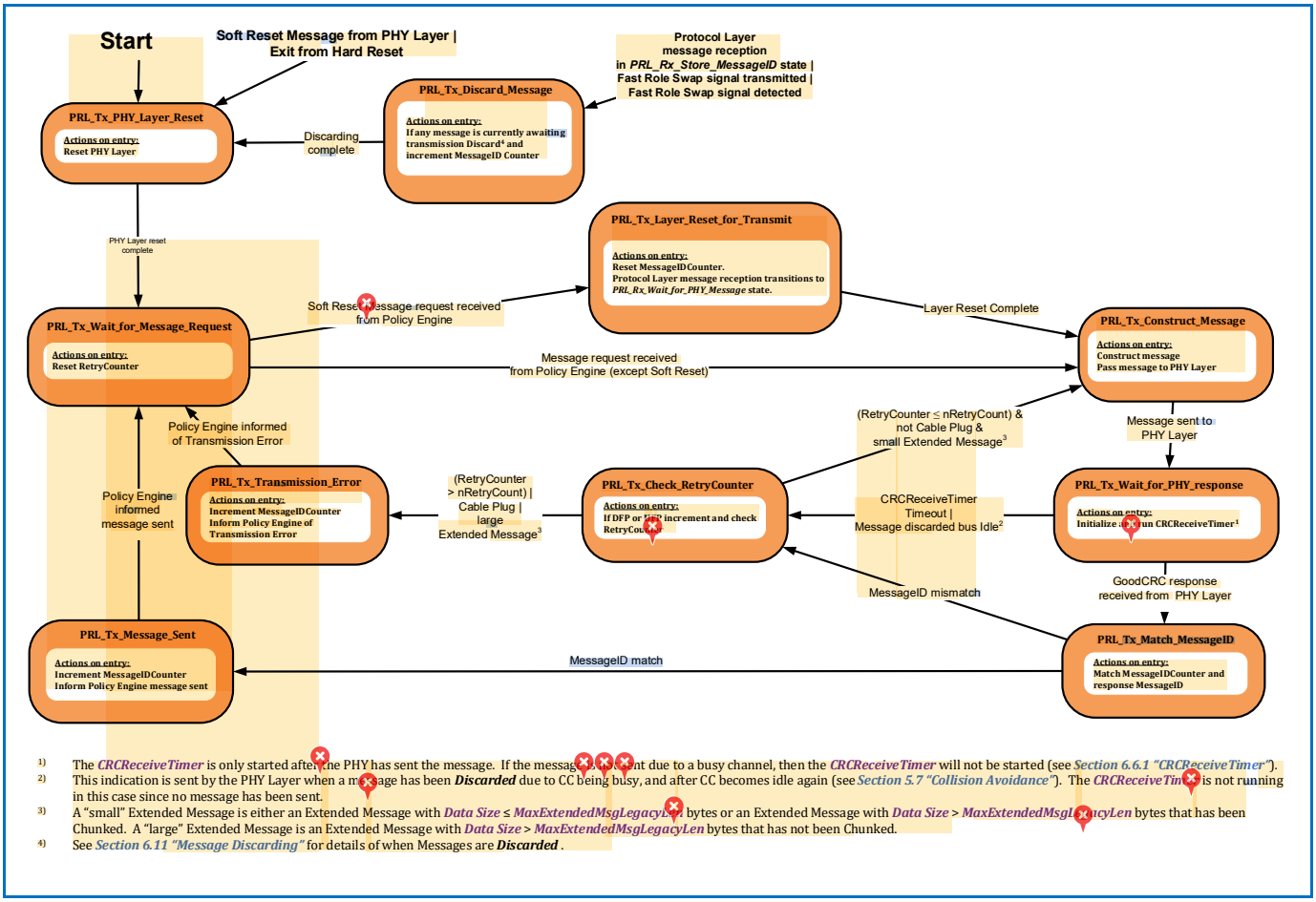
- Send the message to the Chunked Tx State Machine.
- Transition to the *RTR_Wait_for_Message_From_Protocol_Layer* state.

6.12.2.2 Protocol Layer Message Transmission

6.12.2.2.1 Common Protocol Layer Message Transmission State Diagram

Figure 6-61 "Common Protocol Layer Message Transmission State Diagram" shows the state behavior, common between the Source and the Sink, for the Protocol Layer when transmitting a Message.

Figure 6-61 "Common Protocol Layer Message Transmission State Diagram"



6.12.2.2.1.1 PRL_Tx_PHY_Layer_Reset State

The Protocol Layer ***Shall*** enter the *PRL_Tx_PHY_Layer_Reset* state:

- At startup.
- As a result of a Soft Reset request being received by the PHY Layer.
- On exit from a Hard Reset.

On entry to the *PRL_Tx_PHY_Layer_Reset* state the Protocol Layer ***Shall*** reset the PHY Layer (clear any outstanding Messages and enable communications).

The Protocol Layer ***Shall*** transition to the *PRL_Tx_Wait_for_Message_Request* state when:

- When the PHY Layer reset is complete.

6.12.2.2.1.2 PRL_Tx_Wait_for_Message_Request State

In the *PRL_Tx_Wait_for_Message_Request* state the Protocol Layer waits until the Policy Engine directs it to send a Message.

- On entry to the *PRL_Tx_Wait_for_Message_Request* state the Protocol Layer ***Shall*** reset the *RetryCounter*.

The Protocol Layer ***Shall*** transition to the *PRL_Tx_Construct_Message* state when:

- A Message request is received from the Policy Engine which is not a *Soft_Reset* Message.

The Protocol Layer ***Shall*** transition to the *PRL_Tx_Layer_Reset_for_Transmit* state when:

- A Message request is received from the Policy Engine which is a *Soft_Reset* Message.

6.12.2.2.1.3 PRL_Tx_Layer_Reset_for_Transmit State

On entry to the *PRL_Tx_Layer_Reset_for_Transmit* state the Protocol Layer ***Shall*** reset the *MessageIDCounter*. The Protocol Layer ***Shall*** transition Protocol Layer Message reception to the *PRL_Rx_Wait_for_PHY_Message* state (see *Section 6.12.2.3.1 "PRL_Rx_Wait_for_PHY_Message state"*) in order to reset the stored *MessageID*.

The Protocol Layer ***Shall*** transition to the *PRL_Tx_Construct_Message* state when:

- The layer reset actions in this state have been completed.

6.12.2.2.1.4 PRL_Tx_Construct_Message State

On entry to the *PRL_Tx_Construct_Message* state the Protocol Layer ***Shall*** construct the Message requested by the Policy Engine, or resend a previously constructed Message, and then pass this Message to the PHY Layer.

The Protocol Layer ***Shall*** transition to the *PRL_Tx_Wait_for_PHY_Response* state when:

- The Message has been sent to the PHY Layer.

6.12.2.2.1.5 PRL_Tx_Wait_for_PHY_Response State

On entry to the *PRL_Tx_Wait_for_PHY_Response* state, once the Message has been sent, the Protocol Layer ***Shall*** initialize and run the *CRCReceiveTimer* (see *Section 6.6.1 "CRCReceiveTimer"*).

The Protocol Layer ***Shall*** transition to the *PRL_Tx_Match_MessageID* state when:

- A *GoodCRC* Message response is received from the PHY Layer.

The Protocol Layer ***Shall*** transition to the *PRL_Tx_Check_RetryCounter* state when:

- The *CRCReceiveTimer* times out.
- Or the PHY Layer indicates that a Message has been ***Discarded*** due to the channel being busy but the channel is now idle (see *Section 5.7 "Collision Avoidance"*).

6.12.2.2.1.6 PRL_Tx_Match_MessageID State

On entry to the *PRL_Tx_Match_MessageID* state the Protocol Layer ***Shall*** compare the *MessageIDCounter* and the *MessageID* of the received *GoodCRC* Message.

The Protocol Layer ***Shall*** transition to the *PRL_Tx_Message_Sent* state when:

- The *MessageIDCounter* and the *MessageID* of the received *GoodCRC* Message match.

The Protocol Layer ***Shall*** transition to the *PRL_Tx_Check_RetryCounter* state when:

- The ***MessageIDCounter*** and the ***MessageID*** of the received ***GoodCRC*** Message do not match.

6.12.2.2.1.7 PRL_Tx_Message_Sent State

On entry to the ***PRL_Tx_Message_Sent*** state the Protocol Layer ***Shall*** increment the ***MessageIDCounter*** and inform the Policy Engine that the Message has been sent.

The Protocol Layer ***Shall*** transition to the ***PRL_Tx_Wait_for_Message_Request*** state when:

- The Policy Engine has been informed that the Message has been sent.

6.12.2.2.1.8 PRL_Tx_Check_RetryCounter State

On entry to the ***PRL_Tx_Check_RetryCounter*** state the Protocol Layer in a DFP or UFP ***Shall*** increment the value of the ***RetryCounter*** and then check it in order to determine whether it is necessary to retry sending the Message. Note that Cable Plugs do not retry Messages and so do not use the ***RetryCounter***.

The Protocol Layer ***Shall*** transition to the ***PRL_Tx_Construct_Message*** state in order to retry Message sending when:

- ***RetryCounter*** ≤ ***nRetryCount*** and
- This is not a Cable Plug and
- This is an Extended Message with ***Data Size*** ≤ ***MaxExtendedMsgLegacyLen*** or
- This is an Extended Message that has been Chunked.

The Protocol Layer ***Shall*** transition to the ***PRL_Tx_Transmission_Error*** state when:

- ***RetryCounter*** > ***nRetryCount*** or
- This is a Cable Plug, which does not retry.
- This is an Extended Message with ***Data Size*** > ***MaxExtendedMsgLegacyLen*** that has not been Chunked.

6.12.2.2.1.9 PRL_Tx_Transmission_Error State

On entry to the ***PRL_Tx_Transmission_Error*** state the Protocol Layer ***Shall*** increment the ***MessageIDCounter*** and inform the Policy Engine of the transmission error.

The Protocol Layer ***Shall*** transition to the ***PRL_Tx_Wait_for_Message_Request*** state when:

- The Policy Engine has been informed of the transmission error.

6.12.2.2.1.10 PRL_Tx_Discard_Message State

Protocol Layer Message transmission ***Shall*** enter the ***PRL_Tx_Discard_Message*** state whenever:

- Protocol Layer Message reception receives an incoming Message or
- The Fast Role Swap signal is being transmitted (see ***Section 5.8.5.6 "Fast Role Swap Transmission"***)
- The Fast Role Swap signal is detected (see ***Section 5.8.6.3 "Fast Role Swap Detection"***).

On entry to the ***PRL_Tx_Discard_Message*** state, if there is a Message queued awaiting transmission, the Protocol Layer ***Shall Discard*** the Message according to the rules in ***Section 6.11 "Message Discarding"*** and increment the ***MessageIDCounter***.

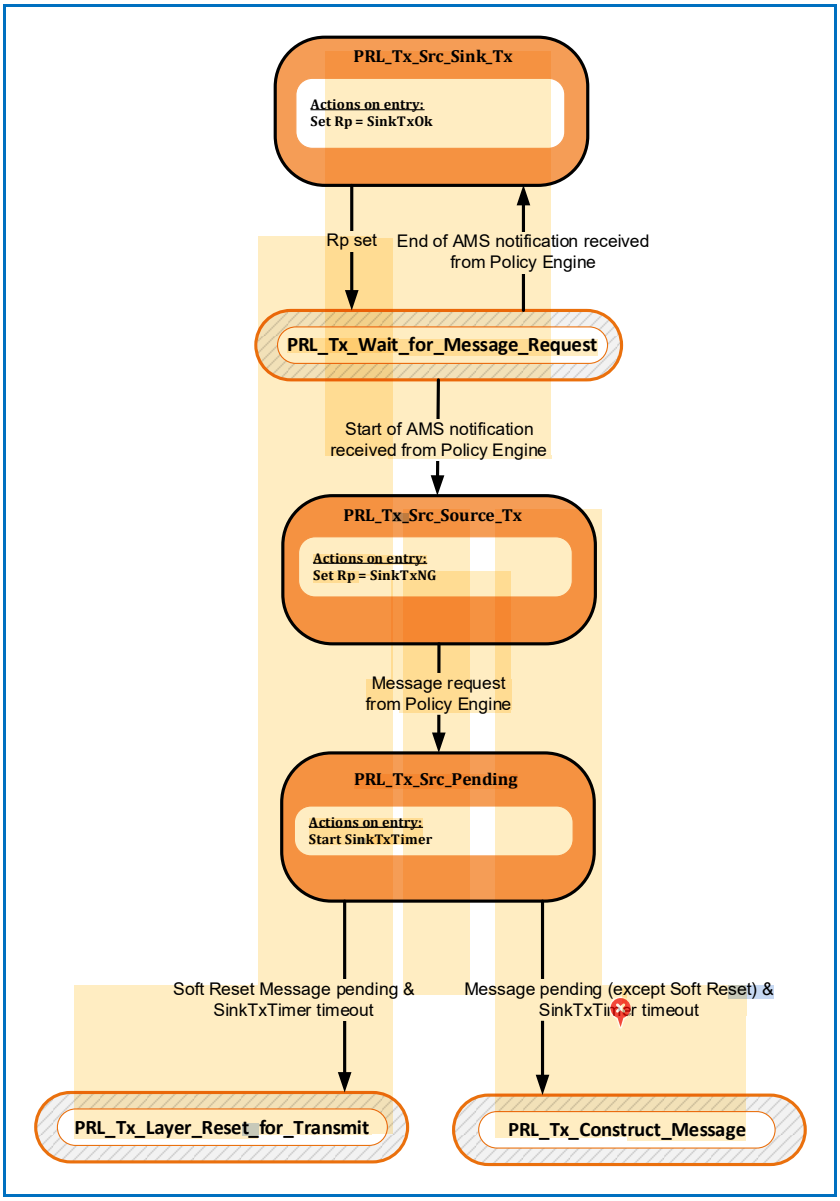
The Protocol Layer ***Shall*** transition to the ***PRL_Tx_PHY_Layer_Reset*** state when:

- Discarding is complete i.e., the Message queue is empty.

6.12.2.2.2 Source Protocol Layer Message Transmission State Diagram

Figure 6-62 "Source Protocol Layer Message Transmission State Diagram" shows the state behavior for the Protocol Layer in a Source when transmitting a Message.

Figure 6-62 "Source Protocol Layer Message Transmission State Diagram"



6.12.2.2.2.1 PRL_Tx_Src_Sink_Tx State

In the *PRL_Tx_Src_Sink_Tx* state the Source sets R_p to *SinkTxOk* allowing the Sink to start an Atomic Message Sequence (AMS).

The Protocol Layer in a Source Shall transition from the *PRL_Tx_Wait_for_Message_Request* state to the *PRL_Tx_Src_Sink_Tx* state when:

- A notification is received from the Policy Engine that the end of an AMS has been reached.

On entry to the *PRL_Tx_Src_Sink_Tx* state the Protocol Layer Shall request the PHY Layer to R_p to ***SinkTxOk***.

The Protocol Layer Shall transition to the *PRL_Tx_Wait_for_Message_Request* state when:

- R_p has been set.

6.12.2.2.2.2 PRL_Tx_Src_Source_Tx State

In the *PRL_Tx_Src_Source_Tx* state the Source sets R_p to ***SinkTxNG*** allowing the Source to start an Atomic Message Sequence (AMS).

The Protocol Layer in a Source Shall transition from the *PRL_Tx_Wait_for_Message_Request* state to the *PRL_Tx_Src_Source_Tx* state when:

- A notification is received from the Policy Engine that an AMS will be starting.

On entry to the *PRL_Tx_Src_Source_Tx* state the Protocol Layer Shall set R_p to ***SinkTxNG***.

The Protocol Layer ***Shall*** transition to the *PRL_Tx_Src_Pending* state when:

- A Message request is received from the Policy Engine.

6.12.2.2.2.3 PRL_Tx_Src_Pending State

In the *PRL_Tx_Src_Pending* state the Protocol Layer has a Message buffered ready for transmission.

On entry to the *PRL_Tx_Src_Pending* state the ***SinkTxTimer*** Shall be initialized and run.

The Protocol Layer ***Shall*** transition to the *PRL_Tx_Construct_Message* state when:

- The pending Message request from the Policy Engine is not a ***Soft_Reset*** Message and
- The ***SinkTxTimer*** times out.

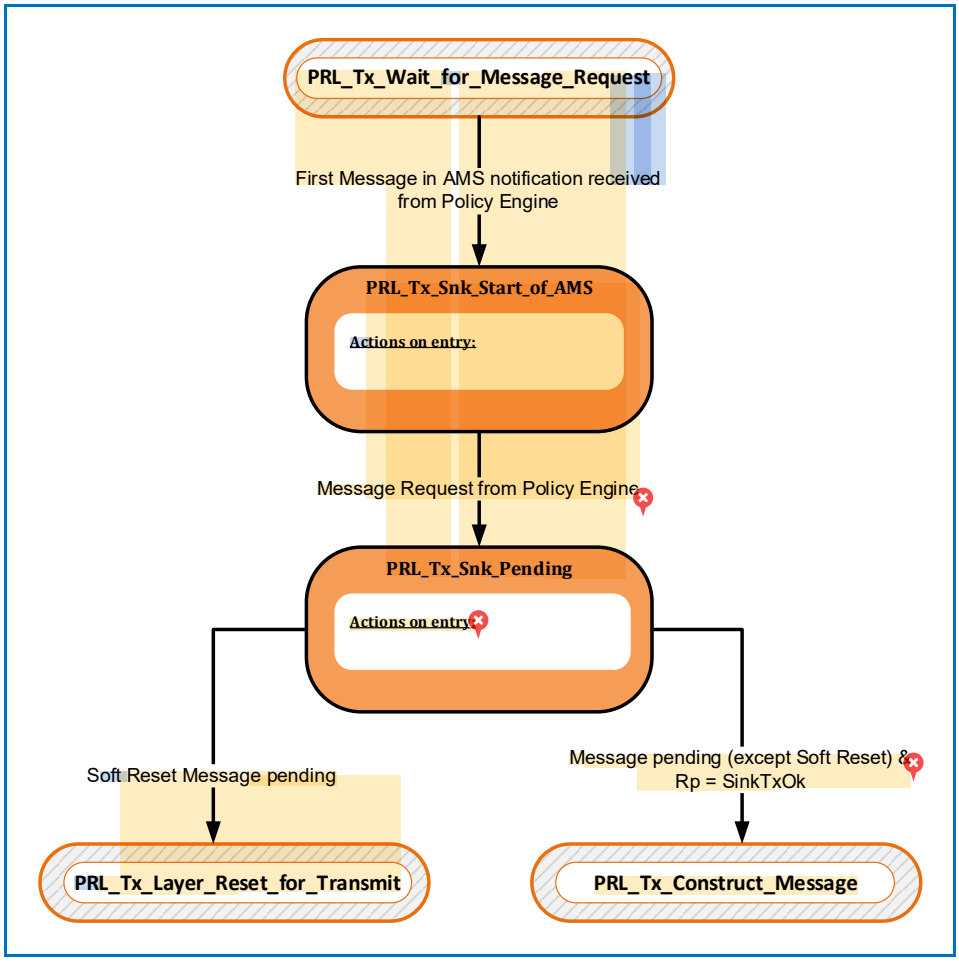
The Protocol Layer ***Shall*** transition to the *PRL_Tx_Layer_Reset_for_Transmit* state when:

- The pending Message request from the Policy Engine is a ***Soft_Reset*** Message and
- The ***SinkTxTimer*** times out.

6.12.2.2.3 Sink Protocol Layer Message Transmission State Diagram

Figure 6-63 "Sink Protocol Layer Message Transmission State Diagram" shows the state behavior for the Protocol Layer in a Sink when transmitting a Message.

Figure 6-63 "Sink Protocol Layer Message Transmission State Diagram"



6.12.2.2.3.1 PRL_Tx_Snk_Start_of_AMS State

In the *PRL_Tx_Snk_Start_of_AMS* state the Protocol Layer waits for the first Message in a Sink initiated AMS.

The Protocol Layer in a Sink Shall transition from the *PRL_Tx_Wait_for_Message_Request* state to the *PRL_Tx_Snk_Start_of_AMS* state when:

- A notification is received from the Policy Engine that the next Message the Sink will send is the start of an AMS.

The Protocol Layer ***Shall*** transition to the *PRL_Tx_Snk_Pending* state when:

- A Message request is received from the Policy Engine.

6.12.2.2.3.2 PRL_Tx_Snk_Pending State

In the *PRL_Tx_Snk_Pending* state the Protocol Layer has the first Message in a Sink initiated AMS ready to send and is waiting for R_p to transition to *SinkTxOk* before sending the Message.

The Protocol Layer ***Shall*** transition to the *PRL_Tx_Construct_Message* state when:

- A Message is Pending that is not a *Soft_Reset* Message and
- R_p is set to *SinkTxOk.*

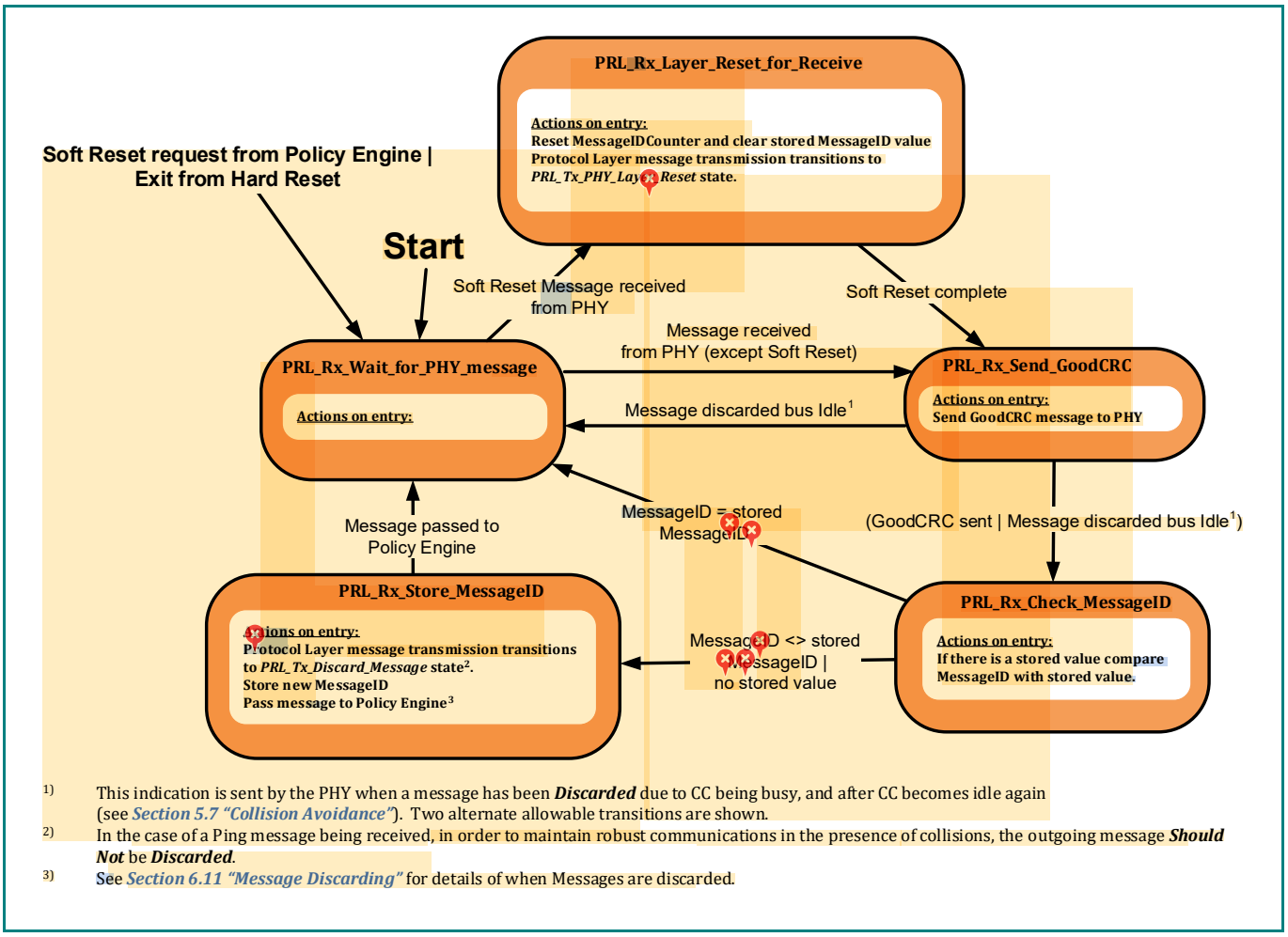
The Protocol Layer ***Shall*** transition to the *PRL_Tx_Layer_Reset_for_Transmit* state when:

- A *Soft_Reset* Message is pending.

6.12.2.3 Protocol Layer Message Reception

Figure 6-64 "Protocol layer Message reception" shows the state behavior for the Protocol Layer when receiving a Message.

Figure 6-64 "Protocol layer Message reception"



1) This indication is sent by the PHY when a message has been ***Discarded*** due to CC being busy, and after CC becomes idle again (see *Section 5.7 "Collision Avoidance"*). Two alternate allowable transitions are shown.
2) In the case of a Ping message being received, in order to maintain robust communications in the presence of collisions, the outgoing message ***Should Not*** be ***Discarded***.
3) See *Section 6.11 "Message Discarding"* for details of when Messages are discarded.

6.12.2.3.1 PRL_Rx_Wait_for_PHY_Message state

The Protocol Layer ***Shall*** enter the *PRL_Rx_Wait_for_PHY_Message* state:

- At startup.
- As a result of a Soft Reset request from the Policy Engine.
- On exit from a Hard Reset.

In the *PRL_Rx_Wait_for_PHY_Message* state the Protocol Layer waits until the PHY Layer passes up a received Message.

The Protocol Layer ***Shall*** transition to the *PRL_Rx_Send_GoodCRC* state when:

- A Message is passed up from the PHY Layer.

The Protocol Layer ***Shall*** transition to the *PRL_Rx_Layer_Reset_for_Receive* state when:

- A *Soft_Reset* Message is received from the PHY Layer.

6.12.2.3.2 PRL_Rx_Layer_Reset_for_Receive state

On entry to the *PRL_Rx_Layer_Reset_for_Receive* state the Protocol Layer ***Shall*** reset the *MessageIDCounter* and clear the stored *MessageID*. The Protocol Layer ***Shall*** transition Protocol Layer Message transmission to the *PRL_Tx_Wait_for_Message_Request* state (see *Section 6.12.2.2.1.2 "PRL_Tx_Wait_for_Message_Request State"*).

The Protocol Layer ***Shall*** transition to the *PRL_Rx_Send_GoodCRC* State when:

- The Soft Reset actions in this state have been completed.

6.12.2.3.3 PRL_Rx_Send_GoodCRC state

On entry to the *PRL_Rx_Send_GoodCRC* state the Protocol Layer ***Shall*** construct a *GoodCRC* Message and request the PHY Layer to transmit it.

The Protocol Layer ***Shall*** transition to the *PRL_Rx_Check_MessageID* state when:

- The *GoodCRC* Message has been passed to the PHY Layer.

When the PHY Layer indicates that a Message has been ***Discarded*** due to CC being busy but CC is now idle (see *Section 5.7 "Collision Avoidance"*), the Protocol Layer ***Shall*** either:

- Transition to the *PRL_Rx_Check_MessageID* state or
- Transition to the *PRL_Rx_Wait_for_PHY_Message* state.

6.12.2.3.4 PRL_Rx_Check_MessageID state

On entry to the *PRL_Rx_Check_MessageID* state the Protocol Layer ***Shall*** compare the *MessageID* of the received Message with its stored value if a value has previously been stored.

The Protocol Layer ***Shall*** transition to the *PRL_Rx_Wait_for_PHY_Message* state when:

- The *MessageID* of the received Message equals the stored *MessageID* value since this is a Message retry which ***Shall*** be ***Discarded***.

The Protocol Layer ***Shall*** transition to the *PRL_Rx_Store_MessageID* state when:

- The *MessageID* of the received Message does not equal the stored *MessageID* value since this is a new Message or
- This is the first received Message and no *MessageID* value is currently stored.

6.12.2.3.5 PRL_Rx_Store_MessageID state

On entry to the *PRL_Rx_Store_MessageID* state the Protocol Layer ***Shall*** transition Protocol Layer Message transmission to the *PRL_Tx_Discard_Message* state (except when a *Ping* Message has been received in which case the *PRL_Tx_Discard_Message* state ***Should Not*** be entered), replace the stored value of *MessageID* with the value of *MessageID* in the received Message and pass the Message up to the Policy Engine.

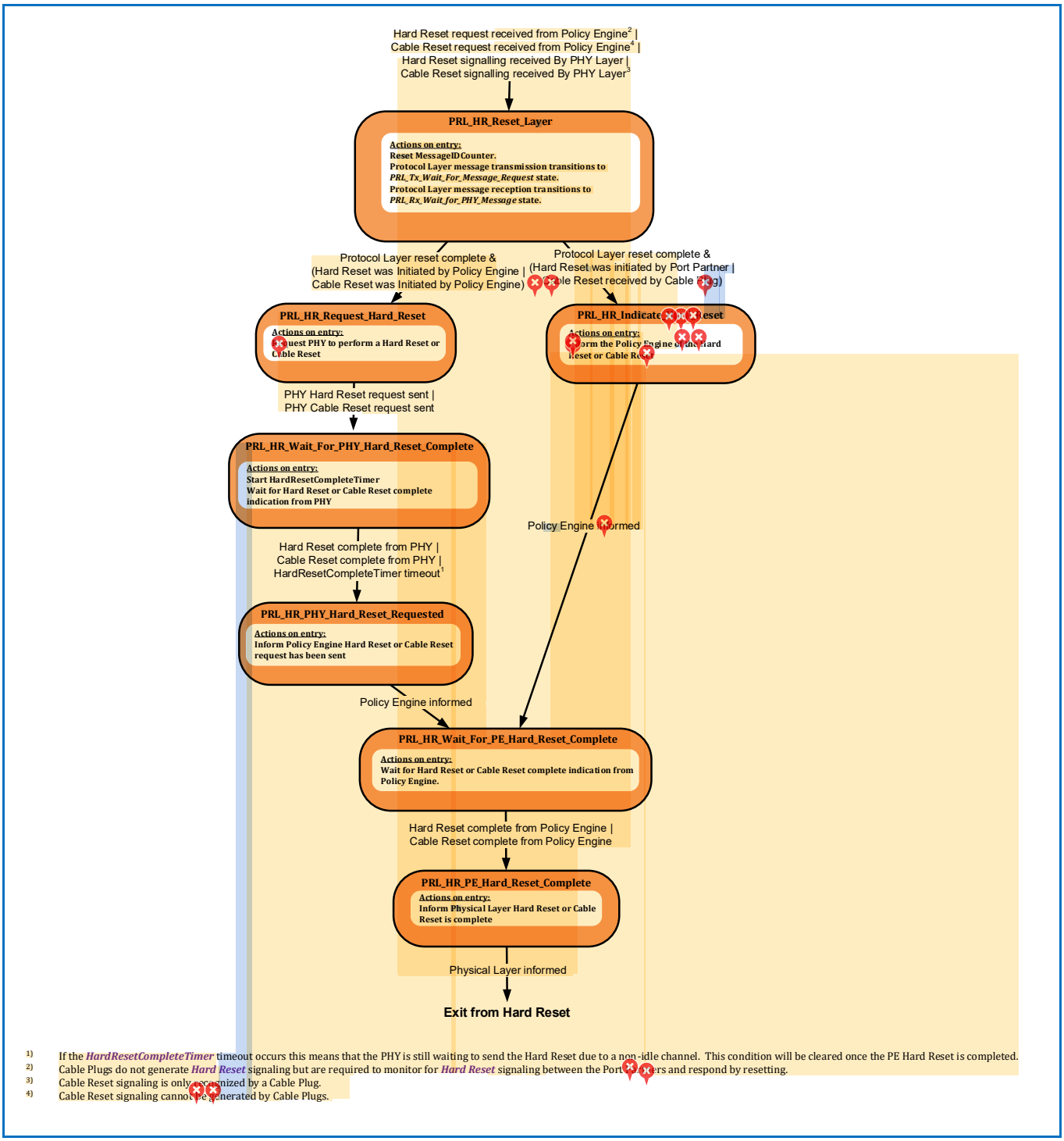
The Protocol Layer ***Shall*** transition to the *PRL_Rx_Wait_for_PHY_Message* state when:

- The Message has been passed up to the Policy Engine.

6.12.2.4 Hard Reset operation

Figure 6-65 "Hard/Cable Reset" shows the state behavior for the Protocol Layer when receiving a Hard Reset or Cable Reset request from the Policy Engine or *Hard Reset* Signaling or *Cable Reset* Signaling from the Physical Layer (see also *Section 6.8.3 "Hard Reset"* and *Section 6.8.4 "Cable Reset"*).

Figure 6-65 "Hard/Cable Reset"



1) If the *HardResetCompleteTimer* timeout occurs this means that the PHY is still waiting to send the Hard Reset due to a non-idle channel. This condition will be cleared once the PE Hard Reset is completed.
2) Cable Plugs do not generate *Hard Reset* signaling but are required to monitor for *Hard Reset* signaling between the Port Partners and respond by resetting.
3) Cable Reset signaling is only recognized by a Cable Plug.
4) Cable Reset signaling cannot be generated by Cable Plugs.

6.12.2.4.1 PRL_HR_Reset_Layer state

The *PRL_HR_Reset_Layer* State defines the mode of operation of both the Protocol Layer transmission and reception state machines during a Hard Reset or Cable Reset. During Hard Reset no USB Power Delivery Protocol Messages are sent or received; only *Hard Reset* Signaling is present after which the communication channel is assumed to have been disabled by the Physical Layer until completion of the Hard Reset. During Cable Reset no USB Power Delivery Protocol Messages are sent to or received by the Cable Plug but other USB Power Delivery communication ***May*** continue.

The Protocol Layer ***Shall*** enter the *PRL_HR_Reset_Layer* state from any other state when:

- A Hard-Reset Request is received from the Policy Engine or
- *Hard Reset* Signaling is received from the Physical Layer or
- A Cable Reset Request is received from the Policy Engine or
- *Cable Reset* Signaling is received from the Physical Layer.

On entry to the *PRL_HR_Reset_Layer* state the Protocol Layer ***Shall*** reset the *MessageIDCounter*. It ***Shall*** also reset the states of the Protocol Layer transmission and reception state machines to their starting points. The Protocol Layer transmission state machine ***Shall*** transition to the *PRL_Tx_Wait_for_Message_Request* state. The Protocol Layer reception state machine ***Shall*** transition to the *PRL_Rx_Wait_for_PHY_Message* state.

The Protocol Layer ***Shall*** transition to the *PRL_HR_Request_Hard_Reset* state when:

- The Protocol Layer's reset is complete and
- The Hard-Reset request has originated from the Policy Engine or
- The Cable Reset request has originated from the Policy Engine.

The Protocol Layer ***Shall*** transition to the *PRL_HR_Indicate_Hard_Reset* state when:

- The Protocol Layer's reset is complete and
- The Hard-Reset request has been passed up from the Physical Layer or
- A Cable Reset request has been passed up from the Physical Layer (Cable Plug only).

6.12.2.4.2 PRL_HR_Indicate_Hard_Reset state

On entry to the *PRL_HR_Indicate_Hard_Reset* state the Protocol Layer ***Shall*** indicate to the Policy Engine that either *Hard Reset* Signaling or *Cable Reset* Signaling has been received.

The Protocol Layer ***Shall*** transition to the *PRL_HR_Wait_for_PE_Hard_Reset_Complete* state when:

- The Indication to the Policy Engine has been sent.

6.12.2.4.3 PRL_HR_Request_Hard_Reset state

On entry to the *PRL_HR_Request_Hard_Reset* state the Protocol Layer ***Shall*** request the Physical Layer to send either *Hard Reset* Signaling or *Cable Reset* signaling.

The Protocol Layer ***Shall*** transition to the *PRL_HR_Wait_for_PHY_Hard_Reset_Complete* state when:

- The Physical Layer *Hard Reset* Signaling request has been sent or
- The Physical Layer *Cable Reset* Signaling request has been sent.

6.12.2.4.4 PRL_HR_Wait_for_PHY_Hard_Reset_Complete state

In the ***PRL_HR_Wait_for_PHY_Hard_Reset_Complete*** state the Protocol Layer ***Shall*** start the ***HardResetCompleteTimer*** and wait for the PHY Layer to indicate that the Hard Reset or Cable Reset has been completed.

The Protocol Layer ***Shall*** transition to the ***PRL_HR_PHY_Hard_Reset_Requested*** state when:

- A Hard-Reset complete indication is received from the PHY Layer or
- A Cable Reset complete indication is received from the PHY Layer or
- The ***HardResetCompleteTimer*** times out.

6.12.2.4.5 PRL_HR_PHY_Hard_Reset_Requested state

On entry to the ***PRL_HR_PHY_Hard_Reset_Requested*** state the Protocol Layer ***Shall*** inform the Policy Engine that the PHY Layer has been requested to perform a Hard Reset or Cable Reset.

The Protocol Layer ***Shall*** transition to the ***PRL_HR_Wait_for_PE_Hard_Reset_Complete*** state when:

- The Indication to the Policy Engine has been sent.

6.12.2.4.6 PRL_HR_Wait_for_PE_Hard_Reset_Complete state

In the ***PRL_HR_Wait_for_PE_Hard_Reset_Complete*** state the Protocol Layer ***Shall*** wait for the Policy Engine to indicate that the Hard Reset or Cable Reset has been completed.

The Protocol Layer ***Shall*** transition to the ***PRL_HR_PE_Hard_Reset_Complete*** state when:

- A Hard-Reset complete indication is received from the Policy Engine or
- A Cable Reset complete indication is received from the Policy Engine.

6.12.2.4.7 PRL_HR_PE_Hard_Reset_Complete

On entry to the ***PRL_HR_PE_Hard_Reset_Complete*** state the Protocol Layer ***Shall*** inform the Physical Layer that the Hard Reset or Cable Reset is complete.

The Protocol Layer ***Shall*** exit from the Hard Reset and return to normal operation when:

- The Physical Layer has been informed that the Hard Reset is complete so that it will re-enable the communications channel. If ***Hard Reset*** Signaling is still pending due to a non-idle channel this ***Shall*** be cleared and not sent or
- The Physical Layer has been informed that the Cable Reset is complete.

6.12.3 List of Protocol Layer States

Table 6.76 "Protocol Layer States" lists the states used by the various state machines.

Table 6.76 "Protocol Layer States"

State Name	Section
Protocol Layer Message Transmission	
Common Protocol Layer Message Transmission	
PRL_Tx_PHY_Layer_Reset	*Section 6.11.2.2.1.1*
PRL_Tx_Wait_for_Message_Request	*Section 6.11.2.2.1.2*
PRL_Tx_Layer_Reset_for_Transmit	*Section 6.11.2.2.1.3*
PRL_Tx_Construct_Message	*Section 6.11.2.2.1.4*
PRL_Tx_Wait_for_PHY_Response	*Section 6.11.2.2.1.5*
PRL_Tx_Match_MessageID	*Section 6.11.2.2.1.6*
PRL_Tx_Message_Sent	*Section 6.11.2.2.1.7*
PRL_Tx_Check_RetryCounter	*Section 6.11.2.2.1.8*
PRL_Tx_Transmission_Error	*Section 6.11.2.2.1.9*
PRL_Tx_Discard_Message	*Section 6.11.2.2.1.10*
Source Protocol Layer Message Transmission	
PRL_Tx_Src_Sink_Tx	*Section 6.11.2.2.2.1*
PRL_Tx_Src_Source_Tx	*Section 6.11.2.2.2.2*
PRL_Tx_Src_Pending	*Section 6.11.2.2.2.3*
Sink Protocol Layer Message Transmission	
PRL_Tx_Snk_Start_of_AMS	*Section 6.11.2.2.3.1*
PRL_Tx_Snk_Pending	*Section 6.11.2.2.3.2*
Protocol Layer Message Reception	
PRL_Rx_Wait_for_PHY_Message	*Section 6.11.2.3.1*
PRL_Rx_Layer_Reset_for_Receive	*Section 6.11.2.3.2*
PRL_Rx_Send_GoodCRC	*Section 6.11.2.3.3*
PRL_Rx_Check_MessageID	*Section 6.11.2.3.4*
PRL_Rx_Store_MessageID	*Section 6.11.2.3.5*
Hard Reset Operation	
PRL_HR_Reset_Layer	*Section 6.11.2.4.1*
PRL_HR_Indicate_Hard_Reset	*Section 6.11.2.4.2*
PRL_HR_Request_Hard_Reset	*Section 6.11.2.4.3*
PRL_HR_Wait_for_PHY_Hard_Reset_Complete	*Section 6.11.2.4.4*
PRL_HR_PHY_Hard_Reset_Requested	*Section 6.11.2.4.5*
PRL_HR_Wait_for_PE_Hard_Reset_Complete	*Section 6.11.2.4.6*
PRL_HR_PE_Hard_Reset_Complete	*Section 6.11.2.4.7*
Chunking	
Chunked Rx	
RCH_Wait_For_Message_From_Protocol_Layer	*Section 6.11.2.1.2.1*
RCH_Pass_Up_Message	*Section 6.11.2.1.2.2*
RCH_Processing_Extended_Message	*Section 6.11.2.1.2.3*
RCH_Requesting_Chunk	*Section 6.11.2.1.2.4*
RCH_Waiting_Chunk	*Section 6.11.2.1.2.5*

RCH_Report_Error	*Section 6.11.2.1.2.6*
Chunked Tx	
TCH_Wait_For_Message_Request_From_Policy_Engine	*Section 6.11.2.1.3.1*
TCH_Pass_Down_Message	*Section 6.11.2.1.3.2*
TCH_Wait_For_Transmision_Complete	*Section 6.11.2.1.3.3*
TCH_Message_Sent	*Section 6.11.2.1.3.4*
TCH_Prepare_To_Send_Chunked_Message	*Section 6.11.2.1.3.5*
TCH_Construct_Chunked_Message	*Section 6.11.2.1.3.6*
TCH_Sending_Chunked_Message	*Section 6.11.2.1.3.7*
TCH_Wait_Chunk_Request	*Section 6.11.2.1.3.8*
TCH_Message_Received	*Section 6.11.2.1.3.9*
TCH_Report_Error	*Section 6.11.2.1.3.10*
Chunked Message Router	
RTR_Wait_for_Message_From_Protocol_Layer	*Section 6.11.2.1.4.1*
RTR_Rx_Chunks	*Section 6.11.2.1.4.2*
RTR_Ping	*Section 6.11.2.1.4.3*
RTR_Tx_Chunks	*Section 6.11.2.1.4.4*

6.13 Message Applicability

The following tables outline the Messages supported by a given port, depending on its capability.

When a Message is supported the feature and Message sequence implied by the Message ***Shall*** also be supported. For example, Sinks using power for charging that support the ***GotoMin*** Message ***Shall*** be able to reduce their current draw when requested via a ***GotoMin*** Message.

The abbreviations in ***Table 6.77 "Message Applicability Abbreviations"*** are used in this section to denote the level of support required.

Table 6.77 "Message Applicability Abbreviations"

Abbreviation	Meaning	Description
N	***Normative***	***Shall*** be supported by this Port/Cable Plug.
CN	Conditional Normative	***Shall*** supported by a given Port/Cable Plug based on features.
R	Recommended	***Should*** be supported by this Port/Cable Plug.
O	***Optional***	***May*** be supported by this Port/Cable Plug.
NS	Not Supported	***Shall*** result in a ***Not_Supported*** Message response by this Port/Cable Plug when received.
I	***Ignore***	***Shall*** be ***Ignored*** by this Port/Cable Plug when received.
NK	NAK	This Port/Cable Plug ***Shall*** return Responder NAK to this Command when received.
NA	Not allowed	***Shall Not*** be transmitted by this Port/Cable Plug.
DR	Don't Recognize	There ***Shall*** no response at all (i.e., not even a ***GoodCRC*** Message) from this Port/Cable Plug when received.

For the case of ***Conditional Normative*** a note has been added to indicate the condition. "CN/" notation is used to indicate the level of support when the condition is not present.

"R/" and "O/" notation is used to indicate the response when the Recommended or ***Optional*** Message is not supported.

Note: that where NS/R/NK is indicated for Received Messages this ***Shall*** apply to the ***PE_CBL_Ready***, ***PE_SNK_Ready*** or ***PE_SRC_Ready*** states only since unexpected Messages received during a Message sequence are Protocol Errors (see ***Section 6.8.1 "Soft Reset and Protocol Error"***).

This section covers Control and Data Message support for Sources, Sink and Cable Plugs. It also covers VDM Command support for DFPs, UFPs and Cable Plugs.

6.13.1 Applicability of Control Messages

Table 6.78 "Applicability of Control Messages" details Control Messages that ***Shall***/***Should***/***Shall Not*** be transmitted and received by a Source, Sink, Cable Plug or VPD. Requirements for Dual-Role Power Ports and Dual-Role Data Ports ***Shall*** override any requirements for Source-only or Sink-Only Ports.

Table 6.78 "Applicability of Control Messages"

Message Type	Source	Sink	Dual-Role Power	Dual-Role Data	Cable Plug	VPD[12]
Transmitted Message						
Accept	N	N			N	N
Data_Reset	CN[13]/R	CN[13]/R			NA	NA
DR_Swap	O	O		N	NA	NA
FR_Swap	NA	NA	R		NA	NA
Get_Country_Codes	CN[10]/NA	CN[10]/NA			NA	NA
Get_PPS_Status	NA	CN[9]			NA	NA
Get_Sink_Cap	R	NA	N		NA	NA
Get_Sink_Cap_Extended	R	NA	R		NA	NA
Get_Source_Cap	NA	R	N		NA	NA
Get_Source_Cap_Extended	NA	R	R		NA	NA
Get_Source_Info	NA	R	R		NA	NA
Get_Revision	R	R			NA	NA
Get_Status	R	R			NA	NA
GoodCRC	N	N			N	N
GotoMin	CN[1]/O	NA			NA	NA
Not_Supported	N	N			NA	NA
Ping	O	NA			NA	NA
PR_Swap	NA	NA	N		NA	NA
PS_RDY	N	CN[4]/NA	N		NA	NA
Reject	N	O	O	O	CN[13]/NA	NA
Soft_Reset	N	N			NA	NA
VCONN_Swap	R	R			NA	NA
Wait	CN[2]/O	NA	O	O	NA	NA
Received Message						
Accept	N	N	N	N	I	I
Data_Reset	CN[13]/R	CN[13]/R			I	I
DR_Swap	O/NS	O/NS		N	I	I
FR_Swap	NS	NS	CN[7]/NS		I	I
Get_Country_Codes	CN[10]/NS	CN[10]/NS			I	I
Get_PPS_Status	CN[9]/NS	NS			I	I

Get_Sink_Cap	NS	N	N		I	I
Get_Sink_Cap_Extended	NS	N	N		I	I
Get_Source_Cap	N	NS	N		I	I
Get_Source_Cap_Extended	CN[5]/NS	NS	CN[5]/NS		I	I
Get_Source_Info	CN[14]	NS	N		I	I
Get_Revision	N	N			O/I	O/I
Get_Status	CN[6]/NS	CN[6]/NS	CN[6]/NS		CN[11]/I	I
GoodCRC	N	N			N	N
GotoMin	NS	P[3]			I	I
Not_Supported	N	N			CN[11]/I	I
Ping	NS	I			I	I
PR_Swap	NS	NS	N		I	I
PS_RDY	CN[4]/NS	N	N		I	I
Reject	CN[8]/NS	N	N	N	I	I
Soft_Reset	N	N			N	N
VCONN_Swap	CN[4]/ NS	CN[4]/ NS			I	I
Wait	CN[8]/NS	N	N	N	I	I

1) ***Should*** be supported by a PDUSB Hub with multiple Downstream Ports. ***Should*** be supported by a Host with multiple Downstream Ports.

2) ***Shall*** be supported when transmission of ***GotoMin*** Messages is supported.

3) ***Should*** be supported by Sinks which use PD power for charging.

4) ***Shall*** be supported by any Port that can supply VCONN.

5) ***Shall*** be supported products that support the ***Source_Capabilities_Extended*** Message.

6) ***Shall*** be supported by Sources that support the ***Alert*** Message.

7) ***Shall*** be supported when the Fast Role Swap signal is supported.

8) ***Shall*** be supported when ***VCONN_Swap*** is supported.

9) ***Shall*** be supported when SPR PPS is supported.

10) ***Shall*** be supported when required by a country authority.

11) ***Shall*** be supported by *Active Cable*s.

12) VPD includes CT-VPDs when not Connected to a Charger. PD communication with a CT-VPD ***Shall*** only take place when not Connected to a Charger.

13) ***Shall*** be supported by products that support ***[USB4]***.

14) ***Shall*** be supported by all Sources except single port chargers with invariant PDOs.

6.13.2 Applicability of Data Messages

Table 6.79 "Applicability of Data Messages" details Data Messages (except for VDM Commands) that ***Shall***/***Should***/***Shall Not*** be transmitted and received by a Source, Sink, Cable Plug or VPD. Requirements for Dual-Role Power Ports ***Shall*** override any requirements for Source-only or Sink-Only Ports.

Table 6.79 "Applicability of Data Messages"

Message Type	Source	Sink	Dual-Role Power	Cable Plug SOP'	Cable Plug SOP"	VPD[6]
Transmitted Message						
Source_Capabilities	N	NA	N	NA	NA	NA
Request	NA	N		NA	NA	NA
Get_Country_Info	CN[5]/O	CN[5]/O		NA	NA	NA
BIST	N[1]	N[1]		NA	NA	NA
Sink_Capabilities	NA	N	N	NA	NA	NA
Battery_Status	CN[2]	CN[2]		NA	NA	NA
Alert	CN[11]/R	CN[11]/R		NA	NA	NA
Enter_USB	CN[7]/O	CN[7]/O		NA	NA	NA
EPR_Request	NA	CN[9]		NA	NA	NA
EPR_Mode	CN[9]	CN[9]		NA	NA	NA

Source_Info	CN[10]	NA	N	NA	NA	NA
Revision	N	N		CN[12]/O/I	NA	O
Received Message						
Source_Capabilities	NS	N	N	I	I	I
Request	N	NS		I	I	I
Get_Country_Info	CN[5]/NS	CN[5]/NS		I	I	I
BIST	N[1]	N[1]		N[1]	N[1]	N[1]
Sink_Capabilities	CN[4]	NS	CN[4]	I	I	I
Battery_Status	CN[3]/NS	CN[3]/NS		I	I	I
Alert	R/NS	R/NS		I	I	I
Enter_USB	CN[7]/O	CN[7]/O		CN[8]/I	CN[8]/I	I
EPR_Request	CN[9]	NA		I	I	I
EPR_Mode	CN[9]	CN[9]		I	I	I
Source_Info	NA	N	N	I	I	I
Revision	N	N		I	I	I

1) For details of which BIST Modes and Messages ***Shall*** be supported see *Section 5.9* and *Section 6.4.3*.

2) ***Shall*** be supported by products that contain batteries.

3) ***Shall*** be supported by products that support the *Get_Battery_Status* Message.

4) ***Shall*** be supported by products that support the *Get_Sink_Cap* Message.

5) ***Shall*** be supported when required by a country authority.

6) VPD includes CT-VPDs when not Connected to a Charger. PD communication with a CT-VPD ***Shall*** only take place when not Connected to a Charger.

7) ***Shall*** be supported by products that support *[USB4]*.

8) ***Shall*** be supported by *Active Cable*s that support *[USB4]*.

9) ***Shall*** be supported by products that support Source operation in EPR Mode.

10) ***Shall*** be supported by all Source Ports except those with invariant PDOs.

11) ***Shall*** be supported when SPR PPS is supported.

12) ***Shall*** be supported by *Active Cable*s.

6.13.3 Applicability of Extended Messages

Table 6.80 "Applicability of Extended Messages" details Extended Messages (except for Extended VDM Commands) that ***Shall*/*Should*/ *Shall Not*** be transmitted and received by a Source, Sink, Cable Plug or VPD. Requirements for Dual-Role Power Ports ***Shall*** override any requirements for Source-only or Sink-Only Ports.

Table 6.80 "Applicability of Extended Messages"

Message Type	Source	Sink	Dual-Role Power	Cable Plug SOP'	Cable Plug SOP''	VPD[1,3]
Transmitted Message						
Battery_Capabilities	CN[1]/NA	CN[1]/NA		NA	NA	NA
Country_Codes	CN[10]/NA	CN[10]/NA		NA	NA	NA
Country_Info	CN[10]/NA	CN[10]/NA		NA	NA	NA
EPR_Source_Capabilities	CN[14]/NA	NA	CN[14]/NA	NA	NA	NA
EPR_Sink_Capabilities	NA	CN[14]/NA	CN[14]/NA	NA	NA	NA
Extended_Control	See *Section 6.13.4* for details					
Firmware_Update_Request	CN[7]/NA	CN[7]/NA		NA	NA	NA
Firmware_Update_Response	CN[7]/NA	CN[7]/NA		CN[7]/NA	O	NA
Get_Battery_Cap	R	R		NA	NA	NA
Get_Battery_Status	R	R		NA	NA	NA
Get_Manufacturer_Info	R	R		NA	NA	NA
Manufacturer_Info	R	R		R	NA	NA
PPS_Status	CN[8]/NA	NA		NA	NA	NA
Security_Request	CN[6]/NA	CN[6]/NA		NA	NA	NA
Security_Response	CN[6]/NA	CN[6]/NA		CN[6]/NA	NA	NA
Sink_Capabilities_Extended	NA	N	N	NA	NA	NA
Source_Capabilities_Extended	R	NA	R	NA	NA	NA
Status	CN[15]/R	CN[15]/R	CN[15]/R	CN[12]/NA	CN[12]/NA	NA
Vendor_Defined_Extended	O	O		O	O	O
Received Message						
Battery_Capabilities	CN[4]/NS	CN[4]/NS		I	I	I
Country_Codes	CN[10]/NS	CN[10]/NS		I	I	I
Country_Info	CN[10]/NS	CN[10]/NS		I	I	I
EPR_Source_Capabilities	NS	CN[14]/NS	CN[14]/NS	I	I	I
EPR_Sink_Capabilities	CN[14]/NS	NS	CN[14]/NS	I	I	I
Extended_Control	See *Section 6.13.4* for details					
Firmware_Update_Request	CN[7]/NS	CN[7]/NS		CN[7]/I	O	I
Firmware_Update_Response	CN[7]/NS	CN[7]/NS		I	I	I
Get_Battery_Cap	CN[1]/NS	CN[1]/NS		I	I	I

Get_Battery_Status	CN[1]/NS	CN[1]/NS		I	I	I
Get_Manufacturer_Info	R/NS	R/NS		R/I	I	I
Manufacturer_Info	CN[5]/NS	CN[5]/NS		I	I	I
PPS_Status	NS	CN[9]/NS		I	I	I
Security_Request	CN[6]/NS	CN[6]/NS		CN[6]/I	I	I
Security_Response	CN[6]/NS	CN[6]/NS		I	I	I
Sink_Capabilities_Extended	CN[11]/NS	NS	CN[11]/NS	I	I	I
Source_Capabilities_Extended	NS	CN[2]/NS	CN[2]/NS	I	I	I
Status	CN[3]/NS	CN[3]/NS		I	I	I
Vendor_Defined_Extended	O/NS	O/NS		O/I	O/I	O/I

1) ***Shall*** be supported by products that contain batteries.

2) ***Shall*** be supported by products that can transmit the *Get_Source_Cap_Extended* Message.

3) ***Shall*** be supported by products that can transmit the *Get_Status* Message.

4) ***Shall*** be supported by products that can transmit the *Get_Battery_Cap* Message.

5) ***Shall*** be supported by products that can transmit the *Get_Manufacturer_Info* Message.

6) ***Shall*** be supported by products that support USB security communication as defined in *[USBTypeCAuthentication 1.0]*.

7) ***Shall*** be supported by products that support USB firmware update communication as defined in *[USBPDFirmwareUpdate 1.0]*.

8) ***Shall*** be supported when PPS is supported.

9) ***Shall*** be supported by products that can transmit the *Get_PPS_Status*.

10) ***Shall*** be supported when required by a country authority.

11) ***Shall*** be supported by products that can transmit the *Get_Sink_Cap_Extended* Message.

12) ***Shall*** be supported by *Active Cable*s.

13) VPD includes CT-VPDs when not Connected to a Charger. PD communication with a CT-VPD ***Shall*** only take place when not Connected to a Charger.

14) ***Shall*** be supported by products that support operation in EPR Mode.

15) ***Shall*** be supported by Sources that support the *Alert* Message.

6.13.4 Applicability of Extended Control Messages

Table 6.81 "Applicability of Extended Control Messages" details Extended Control Messages that ***Shall***/***Should***/***Shall Not*** be transmitted and received by a Source, Sink, Cable Plug or VPD. Requirements for Dual-Role Power Ports and Dual-Role Data Ports ***Shall*** override any requirements for Source-only or Sink-Only Ports.

Table 6.81 "Applicability of Extended Control Messages"

Message Type	Source	Sink	Dual-Role Power	Dual-Role Data	Cable Plug	VPD[12]
Transmitted Message						
EPR_Get_Source_Cap	NA	CN[1]	CN[1]		NA	NA
EPR_Get_Sink_Cap	CN[1]	NA	CN[1]		NA	NA
EPR_KeepAlive	NA	CN[1]			NA	NA
EPR_KeepAlive_Ack	CN[1]	NA			NA	NA
Received Message						
EPR_Get_Source_Cap	CN[1]	NS	CN		I	I
EPR_Get_Sink_Cap	NS	CN[1]	CN[1]		I	I
EPR_KeepAlive	CN[1]	NS			I	I
EPR_KeepAlive_Ack	NS	CN[1]			I	I

1) ***Shall*** be supported by products that support EPR Mode.

6.13.5 Applicability of Structured VDM Commands

Table 6.82 "Applicability of Structured VDM Commands" details Structured VDM Commands that ***Shall***/***Should***/***Shall Not*** be transmitted and received by a DFP, UFP, Cable Plug or VPD. If Structured VDMs are not supported, the DFP or UFP receiving a VDM Command ***Shall*** send a *Not_Supported* Message in response.

Table 6.82 "Applicability of Structured VDM Commands"

Command Type	DFP	UFP	Cable Plug SOP'	Cable Plug SOP''	VPD[4]
Transmitted Command Request					
Discover Identity	CN[1,6]/R	R[2]	NA	NA	NA
Discover SVIDs	CN[1]/O	O	NA	NA	NA
Discover Modes	CN[1]/O	O	NA	NA	NA
Enter Mode	CN[1]/NA	NA	NA	NA	NA
Exit Mode	CN[1]/NA	NA	NA	NA	NA
Attention	O	O	NA	NA	NA
Received Command Request/Transmitted Command Response					
Discover Identity	CN[5,6]/R/NK[3]	CN[1,6]/R/NK[3]	N	I	N
Discover SVIDs	O/NK[3]	CN[1]/NK[3]	CN[1]/NK	I	NK
Discover Modes	O/NK[3]	CN[1]/NK[3]	CN[1]/NK	I	NK
Enter Mode	NK[3]	CN[1]/NK[3]	CN[1]/NK	O	NK
Exit Mode	NK[3]	CN[1]/NK[3]	CN[1]/NK	O	NK
Attention	O/I[3]	O/I[3]	I	I	I

1) ***Shall*** be supported when Modal Operation is supported.

2) ***May*** be transmitted by a UFP/Source during discovery (see *Section 6.4.4.3.1* and *Section 8.3.3.24.3*).

3) If Structured VDMs are not supported, the DFP or UFP receiving a VDM Command ***Shall*** send a *Not_Supported* Message in response.

4) VPD includes CT-VPDs when not Connected to a Charger. PD communication with a CT-VPD ***Shall*** only take place when not Connected to a Charger.

5) ***Shall*** be supported by products with more than one DFP.

6) ***Shall*** be supported by products that support *[USB4]*.

6.13.6 Applicability of Reset Signaling

Table 6.83 "Applicability of Reset Signaling" details Reset Signaling that ***Shall***/***Should***/ ***Shall Not*** be transmitted and received by a DFP/UFP or Cable Plug.

Table 6.83 "Applicability of Reset Signaling"

Signaling Type	DFP	UFP	Cable Plug SOP'	Cable Plug SOP''	VPD[2]
Transmitted Message/Signaling					
Soft_Reset	N	N	NA	NA	NA
Hard Reset	N	N	NA	NA	NA
Cable Reset	CN[1]	NA	NA	NA	NA
Received Message/Signaling					
Soft_Reset	N	N	N	N	N
Hard Reset	N	N	N	N	N
Cable Reset	DR	DR	N	N	N

1) ***Shall*** be supported when transmission of SOP' Packets are supported, and the Port can supply VCONN.

2) VPD includes CT-VPDs when not Connected to a Charger. PD communication with a CT-VPD ***Shall*** only take place when not Connected to a Charger.

6.13.7 Applicability of Fast Role Swap signal

Table 6.84 "Applicability of Fast Role Swap signal" details the Fast Role Swap signal that ***Shall***/***Should***/ ***Shall Not*** be transmitted and received by a Source or Sink.

Table 6.84 "Applicability of Fast Role Swap signal"

Command Type	Source	Sink	Dual-Role Power
Transmitted Message/Signaling			
Fast Role Swap	NA	NA	R
Received Message/Signaling			
Fast Role Swap	NA	NA	R

6.14 Value Parameters

Table 6.85 "Value Parameters" contains value parameters used in this section.

Table 6.85 "Value Parameters"

Parameter	Description	Value	Unit	Reference
MaxExtendedMsgLen	Maximum length of an Extended Message as expressed in the *Data Size* field.	260	Byte	*Section 6.4.8*
MaxExtendedMsgChunkLen		26	Byte	*Section 6.4.8*
MaxExtendedMsgLegacyLen		26	Byte	*Section 6.4.8*