

# Universal Serial Bus Power Delivery Specification

---

*Revision 2.0, V1.0-11 August 2014*

*1. 7 May 2015*

## Editors

Bob Dunstan Intel Corporation  
Richard Petrie DisplayLink

## Contributors

[Joseph Scanlon](#) [Advanced Micro Devices](#)  
[Howard Chang](#) [Allion Labs, Inc.](#)  
[Greg Stewart](#) [Analogix Semiconductor, Inc.](#)  
[Mehran Badii](#) [Analogix Semiconductor, Inc.](#)  
Bill Cornelius Apple  
Colin Whitby-Stevens Apple  
[Corey Axelowitz](#) [Apple](#)  
[Corey Lange](#) [Apple](#)  
James Orr Apple  
Jason Chung Apple  
[Jennifer Tsai](#) [Apple](#)  
[Karl Bowers](#) [Apple](#)  
Keith Porthouse Apple  
Paul Baker Apple  
[Reese Schreiber](#) [Apple](#)  
Sasha Tietz Apple  
Sree Raman Apple  
William Ferry Apple  
[Zaki Moussaoui](#) [Apple](#)  
[Shawn Meng](#) [Bizlink Technology Inc.](#)  
Alessandro Ingrassia Canova Tech  
Andrea Cognese Canova Tech  
[Davide Ghedin](#) [Canova Tech](#)

Matteo Casalin	Canova Tech
Nicola Scantamburlo	Canova Tech
Anup Nayak	Cypress Semiconductor
Rushil Kadakia	Cypress Semiconductor
<a href="#">Steven Wong</a>	<a href="#">Cypress Semiconductor</a>
<a href="#">Subu Sankaran</a>	<a href="#">Cypress Semiconductor</a>
<a href="#">Adolfo Montero</a>	<a href="#">Dell Inc.</a>
Bruce Montag	Dell Inc.
Gary Verdun	Dell, Inc.
<a href="#">Merle Wood</a>	<a href="#">Dell Inc.</a>
<a href="#">Mohammed Hijazi</a>	<a href="#">Dell Inc.</a>
<a href="#">Siddhartha Reddy</a>	<a href="#">Dell Inc.</a>
Dan Ellis	DisplayLink
<a href="#">Jason Young</a>	<a href="#">DisplayLink</a>
Peter Burgers	DisplayLink
Richard Petrie	DisplayLink
Chuck Trefts	Ellisys
<a href="#">Emmanuel Durin</a>	<a href="#">Ellisys</a>
<a href="#">Mario Pasquali</a>	<a href="#">Ellisys</a>
Chien-Cheng Kuo	Etron Technology, Inc.
Jack Yang	Etron Technology, Inc.
Richard Crisp	Etron Technology, Inc.
<a href="#">Shyanjia Chen</a>	<a href="#">Etron Technology, Inc.</a>
<a href="#">TsungTa Lu</a>	<a href="#">Etron Technology, Inc.</a>
<a href="#">Christian Klein</a>	<a href="#">Fairchild Semiconductor</a>
Oscar Freitas	Fairchild Semiconductor

PD Chair/Device Policy Lead

AJ Yang	Foxconn / Hon Hai
Steve Sedio	Foxconn / Hon Hai
Terry Little	Foxconn / Hon Hai
<a href="#">Dian Kurniawan</a>	<a href="#">Fresco Logic Inc.</a>
Adam Rodriguez	Google Inc.
<a href="#">Alec Berg</a>	<a href="#">Google Inc.</a>
David Schneider	Google Inc.
Jim Guerin	Google Inc.
<a href="#">Juan Fantin</a>	<a href="#">Google Inc.</a>
Ken Wu	Google Inc.
Mark Hayter	Google Inc.
<a href="#">Todd Broch</a>	<a href="#">Google Inc.</a>
Vincent Palatin	Google Inc.
Mike Engbretson	Granite River Labs
<a href="#">Rajaraman V</a>	<a href="#">Granite River Labs</a>
Alan Berkema	Hewlett Packard
Lee Atkinson	Hewlett Packard
Rahul Lakdawala	Hewlett Packard
Robin Castell	Hewlett Packard
<a href="#">Roger Benson</a>	<a href="#">Hewlett Packard</a>
Ron Schooley	Hewlett Packard
Vaibhav Malik	Hewlett Packard
Walter Fry	Hewlett Packard
Bob Dunstan	Intel Corporation
Brad Saunders	Intel Corporation
Chee Lim Nge	Intel Corporation

PD Chair/Protocol WG Lead

<a href="#"><u>Christine Krause</u></a>	<a href="#"><u>Intel Corporation</u></a>	
Dan Froelich	Intel Corporation	
David Harriman	Intel Corporation	
Guobin Liu	Intel Corporation	
Harry Skinner	Intel Corporation	
<a href="#"><u>Henrik Leegaard</u></a>	<a href="#"><u>Intel Corporation</u></a>	
Jervis Lin	Intel Corporation	
John Howard	Intel Corporation	
Karthi Vadivelu	Intel Corporation	
<del>Christine Krause</del>	<del>Intel Corporation</del>	
Leo Heiland	Intel Corporation	
Maarit Harkonen	Intel Corporation	
Nge Chee Lim	Intel Corporation	
Paul Durley	Intel Corporation	
Rahman Ismail	Intel Corporation	System Policy Lead
Ronald Swartz	Intel Corporation	
Sarah Sharp	Intel Corporation	
Scott Brenden	Intel Corporation	
Sridharan Ranganathan	Intel Corporation	
Steve McGowan	Intel Corporation	
Tim McKee	Intel Corporation	PD Chair/Compliance Lead
Toby Opferman	Intel Corporation	
Kenta Minejima	Japan Aviation Electronics Industry Ltd. (JAE)	
Mark Saubert	Japan Aviation Electronics Industry Ltd. (JAE)	
Toshio Shimoyama	Japan Aviation Electronics Industry Ltd. (JAE)	
<a href="#"><u>Gianluca Mariani</u></a>	<a href="#"><u>Lattice Semiconductor Corp</u></a>	

[Thomas Watza](#) [Lattice Semiconductor Corp](#)

[Vesa Lauri](#) [Lattice Semiconductor Corp](#)

Daniel H Jacobs LeCroy Corporation

Jake Jacobs LeCroy Corporation

Kimberley McKay LeCroy Corporation

Mike Micheletti LeCroy Corporation

Roy Chestnut LeCroy Corporation

Dave Thompson LSI Corporation

Alan Kinningham Luxshare-ICT

[Daniel Chen](#) [Luxshare-ICT](#)

[Josue Castillo](#) [Luxshare-ICT](#)

Chris Yokum MCCI Corporation

Geert Knapen MCCI Corporation

[Velmurugan Selvaraj](#) [MCCI Corporation](#)

[Brian Marley](#) [Microchip Technology Inc.](#)

[Dave Perchlik](#) [Microchip Technology Inc.](#)

Don Perkins Microchip Technology Inc.

John Sisto Microchip Technology Inc.

Josh Averyt Microchip Technology Inc.

[Kiet Tran](#) [Microchip Technology Inc.](#)

Mark Bohm Microchip Technology Inc.

[Matthew Kalibat](#) [Microchip Technology Inc.](#)

Mick Davis Microchip Technology Inc.

Rich Wahler Microchip Technology Inc.

Shannon Cash Microchip Technology Inc.

David Voth Microsoft Corporation

Jayson Kastens	Microsoft Corporation	
Kai Inha	Microsoft Corporation	
Marwan Kadado	Microsoft Corporation	
Randy Aull	Microsoft Corporation	
Shiu Ng	Microsoft Corporation	
Toby Nixon	Microsoft Corporation	
Vivek Gupta	Microsoft Corporation	
Yang You	Microsoft Corporation	
<a href="#">Ben Crowe</a>	<a href="#">MQP Electronics Ltd.</a>	
Pat Crowe	MQP Electronics Ltd.	
Sten Carlsen	MQP Electronics Ltd.	
Frank Borngräber	Nokia Corporation	
Kai Inha	Nokia Corporation	
Pekka Leinonen	Nokia Corporation	
Richard Petrie	Nokia Corporation	PD Vice-Chair/Device Policy Lead
Sten Carlsen	Nokia Corporation	Physical Layer WG Lead
Abhijeet Kulkarni	NXP Semiconductors	
Ahmad Yazdi	NXP Semiconductors	
Bart Vertenten	NXP Semiconductors	
<a href="#">Dong Nguyen</a>	<a href="#">NXP Semiconductors</a>	
Ken Jaramillo	NXP Semiconductors	
Krishnan TN	NXP Semiconductors	
Robert de Nie	NXP Semiconductors	
<a href="#">Rod Whitby</a>	<a href="#">NXP Semiconductors</a>	
<a href="#">Vijendra Kuroodi</a>	<a href="#">NXP Semiconductors</a>	
Robert Heaton	Obsidian Technology	

Bryan McCoy	ON Semiconductor	
Cor Voorwinden	ON Semiconductor	
Edward Berrios	ON Semiconductor	Power Supply WG Lead
Tom Duffy	ON Semiconductor	
<a href="#">Craig Wiley</a>	<a href="#">Parade Technologies Inc.</a>	
<a href="#">Ricardo Pregiteer</a>	<a href="#">Power Integrations</a>	
Narendra Mehta	Qualcomm, Inc.	
Terry Remple	Qualcomm, Inc.	
Yoram Rimoni	Qualcomm, Inc.	
Atsushi Mitamura	Renesas Electronics Corp.	
<a href="#">Dan Aoki</a>	<a href="#">Renesas Electronics Corp.</a>	
Kiichi Muto	Renesas Electronics Corp.	
Masami Katagiri	Renesas Electronics Corp.	
<a href="#">Nobuo Furuya</a>	<a href="#">Renesas Electronics Corp.</a>	
Patrick Yu	Renesas Electronics Corp.	
Peter Teng	Renesas Electronics Corp.	
Philip Leung	Renesas Electronics Corp.	
Steve Roux	Renesas Electronics Corp.	
Tetsu Sato	Renesas Electronics Corp.	
<a href="#">Tatsuya Irisawa</a>	<a href="#">Ricoh Company Ltd.</a>	
Akihiro Ono	Rohm Co., Ltd.	
Chris Lin	Rohm Co., Ltd.	
<a href="#">Hidenori Nishimoto</a>	<a href="#">Rohm Co. Ltd.</a>	
Manabu Miyata	Rohm Co., Ltd.	
<a href="#">Ruben Balbuena</a>	<a href="#">Rohm Co. Ltd.</a>	
Takashi Sato	Rohm Co., Ltd.	



Vijendra Kuroodi	Rohm Co., Ltd.	
Yusuke Kondo	Rohm Co., Ltd.	
<a href="#">Matti Kulmala</a>	<a href="#">Salcomp Plc</a>	
<a href="#">Toni Lehimo</a>	<a href="#">Salcomp Plc</a>	
<a href="#">Tong Kim</a>	<a href="#">Samsung Electronics Co. Ltd.</a>	
Alvin Cox	Seagate Technology LLC	Cab Con WG Lead
John Hein	Seagate Technology LLC	
Marc Noblitt	Seagate Technology LLC	
Ronald Rueckert	Seagate Technology LLC	
Tony Priborsky	Seagate Technology LLC	
John Sisto	SMSC	
Ken Gay	SMSC	
Mark Bohm	SMSC	
Richard Wahler	SMSC	
Shannon Cash	SMSC	
Tim Knowlton	SMSC	
William Chiechi	SMSC	
Fabien Friess	ST-Ericsson	
Giuseppe Platania	ST-Ericsson	
Jean-Francois Gatto	ST-Ericsson	
Milan Stamenkovic	ST-Ericsson	
Nicolas Florenchie	ST-Ericsson	
Patrizia Milazzo	ST-Ericsson	
<a href="#">Christophe Lorin</a>	<a href="#">ST-Microelectronics</a>	
<a href="#">Meriem Mersel</a>	<a href="#">ST-Microelectronics</a>	
Pascal Legrand	ST-Microelectronics	

[Meriem Mersel](#) [ST-Microelectronics](#)

Patrizia Milazzo ST-Microelectronics

[Joan Marrinan](#) [Tektronix](#)

[Kimberley McKay](#) [Teledyne-LeCroy](#)

[Matthew Dunn](#) [Teledyne-LeCroy](#)

[Tony Minchell](#) [Teledyne-LeCroy](#)

Anand Dabak Texas Instruments

[Bill Waters](#) [Texas Instruments](#)

Deric Waters Texas Instruments

Physical Layer WG Lead

Grant Ley Texas Instruments

Ingolf Frank Texas Instruments

Ivo Huber Texas Instruments

Javed Ahmad Texas Instruments

Jean Picard Texas Instruments

Martin Patoka Texas Instruments

Scott Jackson Texas Instruments

Srinath Hosur Texas Instruments

Steven Tom Texas Instruments

[Dydron Lin](#) [VIA Technologies, Inc.](#)

[Fong-Jim Wang](#) [VIA Technologies, Inc.](#)

Jay Tseng VIA Technologies, Inc.

[Rex Change](#) [VIA Technologies, Inc.](#)

Terrance Shih VIA Technologies, Inc.

Charles Neumann Western Digital Technologies, Inc.

Curtis Stevens Western Digital Technologies, Inc.

John Maroney Western Digital Technologies, Inc.



## Revision History

Revision	Version	Comments	Issue Date
1.0	1.0	Initial release Revision 1.0	5 July, 2012
1.0	1.1	Including errata through 31-October-2012	31 October 2012
1.0	1.2	Including errata through 26-June-2013	26 June, 2013
1.0	1.3	Including errata through 11-March-2014	11 March 2014
2.0	1.0	Initial release Revision 2.0	11 August 2014
<u>2.0</u>	<u>1.1</u>	<u>Including errata through 7-May 2015</u>	<u>7 May 2015</u>

### INTELLECTUAL PROPERTY DISCLAIMER

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

Please send comments via electronic mail to [techsup@usb.org](mailto:techsup@usb.org)

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

All product names are trademarks, registered trademarks, or service marks of their respective owners.

Copyright © 2010-~~2014~~2015 Hewlett-Packard Company, Intel Corporation, LSI Corporation, Microsoft Corporation, Renesas, STMicroelectronics, and Texas Instruments

All rights reserved.

## Table of Contents

<b>Contributors .....</b>	<b>2</b>
<b>Revision History .....</b>	<b>12</b>
<b>Table of Contents .....</b>	<b>13</b>
<b>List of Tables.....</b>	<b>23</b>
<b>List of Figures.....</b>	<b>28</b>
<b>1. Introduction .....</b>	<b>36</b>
1.1 Overview .....	36
1.2 Purpose.....	37
1.3 Scope.....	37
1.4 Conventions .....	37
1.4.1 Precedence.....	37
1.4.2 Keywords.....	37
1.4.3 Numbering .....	38
1.5 Related Documents .....	39
1.6 Terms and Abbreviations.....	39
1.7 Parameter Values.....	45
<b>2. Overview.....</b>	<b>46</b>
2.1 Introduction.....	46
2.2 Section Overview .....	47
2.3 USB Power Delivery Capable Devices .....	48
2.4 SOP* Communication with Cable Plugs.....	49
2.4.1 SOP' Communication.....	49
2.4.2 SOP'' Communication.....	50
2.5 Operational Overview.....	51
2.5.1 DFP Operation .....	51
2.5.2 UFP Operation.....	54
2.5.3 Cable Plug Operation .....	57
2.6 Architectural Overview.....	58
2.6.1 Policy .....	60

2.6.2	Message Formation and Transmission.....	61
2.6.3	Power supply .....	61
2.6.4	Cable and Connectors .....	62
<b>3.</b>	<b>Type-A and Type-B Cable Assemblies and Connectors .....</b>	<b>63</b>
3.1	Significant Features .....	63
3.1.1	Connectors .....	63
3.1.2	Compliant Cable Assemblies.....	65
3.1.3	USB Power Delivery Adapters (USB plug to USB receptacle) .....	66
3.1.4	Hardwired Captive PD Cable .....	66
3.1.5	Standard-A Insertion Detect .....	66
3.1.6	Standard-A PD Detect.....	66
3.1.7	Raw Cables .....	67
3.2	Connector Mating Interfaces.....	68
3.2.1	Standard-A Insertion Detect Mechanical Dimensions .....	68
3.2.2	USB PD Standard-A PD Detect Mechanical Requirement .....	68
3.2.3	USB 2.0 PD Standard-A Connector .....	69
3.2.4	USB 3.1 PD Standard-A Connector .....	74
3.2.5	USB 2.0 PD Standard-B Connector .....	78
3.2.6	USB 3.1 PD Standard-B Connector .....	81
3.3	Cable Assemblies .....	86
3.3.1	Non-marked Cable Assemblies .....	86
3.3.2	Marked Cable Assemblies .....	86
3.3.3	PD Cable Assembly Overmold Requirements.....	86
3.4	PD Cable Assembly Marking .....	88
3.4.1	Marker for PD Standard-A Connectors.....	88
3.4.2	Electronic Markers for Micro-A Plugs.....	88
3.4.3	Electronic Markers for PD Standard-B Plugs and Micro-B Plugs .....	89
3.5	USB PD Icon.....	90
3.6	USB Power Delivery Cable Requirements .....	91
3.6.1	Low Level Contact Resistance (EIA 364-23B).....	91
3.6.2	Open Circuit Resistance.....	91
3.6.3	Dielectric Strength (EIA 364-20).....	91
3.6.4	Insulation Resistance (EIA 364-21).....	91

3.6.5	Contact Current Rating .....	92
3.6.6	Differential Crosstalk between $V_{BUS}$ and the D+/D- Pair (EIA-360-90).....	92
3.6.7	PD Cable Assembly Shielding Connectivity .....	92
3.6.8	PD Cable $V_{BUS}$ Impedance .....	93
3.6.9	PD Cable Insertion Loss .....	93
3.6.10	PD Cable IR Drop Considerations .....	93
3.7	Electrical Parameters.....	95
<b>4.</b>	<b>Electrical Requirements.....</b>	<b>96</b>
4.1	Dead Battery Detection / Unpowered Port Detection.....	96
4.1.1	Type-A to Type-B Dead Battery Operation.....	96
4.1.2	Type-C to Type-C Dead Battery Operation .....	100
4.2	Cable IR Ground Drop (IR Drop) .....	100
4.3	A-Plug Insertion Detect.....	100
4.4	Cable Type Detection .....	100
4.4.1	Detecting Cabling Capabilities .....	100
4.4.2	Plug Type Determination .....	101
4.4.3	Detecting the PD Capabilities of the Standard-A Connector.....	102
4.4.4	Plug Type Detection except Standard-A .....	103
4.5	Low Power Devices using Micro-A Plug.....	104
4.6	Electrical Parameters.....	105
<b>5.</b>	<b>Physical Layer.....</b>	<b>106</b>
5.1	Physical Layer Overview .....	106
5.2	Physical Layer Functions.....	106
5.3	Symbol Encoding .....	107
5.4	Ordered Sets .....	109
5.5	Transmitted Bit Ordering.....	111
5.6	Packet Format.....	113
5.6.1	Packet Framing.....	113
5.6.2	CRC .....	115
5.6.3	Packet Detection Errors.....	117
5.6.4	Hard Reset.....	117
5.6.5	Cable Reset.....	118
5.7	Collision Avoidance.....	118

5.8	Physical Layer Signaling Schemes .....	119
5.8.1	Common Signaling Scheme Specifications .....	119
5.8.2	Binary Frequency Shift Keyed (BFSK) Signaling Scheme .....	121
5.8.3	Biphase Mark Coding (BMC) Signaling Scheme .....	129
5.8.4	Interoperability with BFSK and BMC .....	146
5.9	Built in Self-Test (BIST) .....	147
5.9.1	BIST PRBS Pattern .....	147
5.9.2	BIST Carrier Mode 0 .....	148
5.9.3	BIST Carrier Mode 1 .....	148
5.9.4	BIST Carrier Mode 2 .....	148
5.9.5	BIST Carrier Mode 3 .....	148
5.9.6	BIST Eye Pattern .....	149
5.9.7	BIST Test Data .....	149
5.9.8	BIST Parameters .....	149
5.9.9	BIST Test Applicability .....	149
<b>6.</b>	<b>Protocol Layer .....</b>	<b>151</b>
6.1	Overview .....	151
6.2	Messages .....	151
6.2.1	Message Construction .....	151
6.3	Control Message .....	154
6.3.1	GoodCRC Message .....	154
6.3.2	GotoMin Message .....	154
6.3.3	Accept Message .....	155
6.3.4	Reject Message .....	155
6.3.5	Ping Message .....	155
6.3.6	PS_RDY Message .....	156
6.3.7	Get_Source_Cap Message .....	156
6.3.8	Get_Sink_Cap Message .....	156
6.3.9	DR_Swap Message .....	156
6.3.10	PR_Swap Message .....	157
6.3.11	VCONN_Swap Message .....	157
6.3.12	Wait Message .....	158
6.3.13	Soft Reset Message .....	159



6.4	Data Message.....	159
6.4.1	Capabilities Message.....	160
6.4.2	Request Message .....	168
6.4.3	BIST Message .....	171
6.4.4	Vendor Defined Message.....	175
6.5	Timers .....	193
6.5.1	CRCReceiveTimer.....	193
6.5.2	SenderResponseTimer .....	193
6.5.3	Activity Timers .....	194
6.5.4	Capability Timers .....	195
6.5.5	SinkRequestTimer.....	195
6.5.6	Power Supply Timers .....	195
6.5.7	NoResponseTimer.....	196
6.5.8	BIST Timers .....	197
6.5.9	Power Role Swap Timers .....	198
6.5.10	Hard Reset Timers .....	198
6.5.11	Structured VDM Timers.....	198
6.5.12	VCONN Timers.....	200
6.5.13	tCableMessage .....	200
6.5.14	DiscoverIdentityTimer.....	200
6.5.15	Attention Timers.....	200
6.5.16	Time Values and Timers.....	200
6.6	Counters .....	203
6.6.1	MessageID Counter.....	203
6.6.2	Retry Counter.....	203
6.6.3	Hard Reset Counter .....	203
6.6.4	Capabilities Counter .....	203
6.6.5	BIST Error Counter .....	204
6.6.6	Discover Identity Counter.....	204
6.6.7	VDMBusyCounter .....	204
6.6.8	nAttentionCount .....	204
6.6.9	Counter Values and Counters.....	204
6.7	Reset.....	205
6.7.1	Soft Reset .....	205

6.7.2	Hard Reset.....	205
6.7.3	Cable Reset.....	206
6.8	State behavior .....	207
6.8.1	Introduction to state diagrams used in Chapter 6 .....	207
6.8.2	State Operation.....	207
6.8.3	BIST Operation .....	220
6.8.4	List of Protocol Layer States .....	224
6.9	Message Applicability .....	226
<b>7.</b>	<b>Power Supply .....</b>	<b>229</b>
7.1	Source Requirements.....	229
7.1.1	Behavioral Aspects .....	229
7.1.2	Source Bulk Capacitance .....	229
7.1.3	Types of Sources .....	230
7.1.4	Positive Voltage Transitions .....	230
7.1.5	Negative Voltage Transitions .....	231
7.1.6	Response to Hard Resets.....	231
7.1.7	Changing the Output Power Capability.....	233
7.1.8	Safe Operating Considerations .....	233
7.1.9	Output Voltage Tolerance and Range .....	233
7.1.10	Charging and Discharging the Bulk Capacitance on $V_{BUS}$ .....	234
7.1.11	Swap Standby for Sources .....	234
7.1.12	Source Peak Current Operation.....	235
7.1.13	BFSK over $V_{BUS}$ Considerations for Sources.....	235
7.2	Sink Requirements.....	239
7.2.1	Behavioral Aspects .....	239
7.2.2	Sink Bulk Capacitance.....	239
7.2.3	Sink Standby.....	240
7.2.4	Suspend Power Consumption .....	240
7.2.5	Zero Negotiated Current.....	240
7.2.6	Transient Load Behavior .....	240
7.2.7	Swap Standby for Sinks.....	240
7.2.8	Sink Peak Current Operation.....	241
7.2.9	BFSK over $V_{BUS}$ Considerations for Sinks.....	241

7.2.10	Safe Operating Considerations .....	243
7.3	Transitions .....	244
7.3.1	Increasing the Current .....	245
7.3.2	Increasing the Voltage.....	247
7.3.3	Increasing the Voltage and Current.....	249
7.3.4	Increasing the Voltage and Decreasing the Current .....	251
7.3.5	Decreasing the Voltage and Increasing the Current .....	253
7.3.6	Decreasing the Current.....	255
7.3.7	Decreasing the Voltage.....	257
7.3.8	Decreasing the Voltage and the Current.....	259
7.3.9	Sink Requested Power Role Swap.....	261
7.3.10	Source Requested Power Role Swap.....	264
7.3.11	GotoMin Current Decrease .....	267
7.3.12	Source Initiated Hard Reset .....	269
7.3.13	Sink Initiated Hard Reset .....	271
7.3.14	Type-A/B Hard Reset after a Power Role Swap .....	274
7.3.15	Type-A to Type-B Dead Battery Operation.....	282
7.3.19	No change in Current or Voltage.....	284
7.4	Electrical Parameters.....	286
7.4.1	Source Electrical Parameters .....	286
7.4.2	Sink Electrical Parameters.....	289
7.4.3	Common Electrical Parameters.....	291
<b>8.</b>	<b>Device Policy .....</b>	<b>292</b>
8.1	Overview .....	292
8.2	Device Policy Manager.....	292
8.2.1	Capabilities.....	293
8.2.2	System Policy .....	293
8.2.3	Control of Source/Sink.....	293
8.2.4	Cable Detection .....	294
8.2.5	Managing Power Requirements .....	294
8.2.6	Use of “Externally Powered” bit with Batteries and AC supplies .....	296
8.2.7	Interface to the Policy Engine .....	297
8.3	Policy Engine .....	299

8.3.1	Introduction.....	299
8.3.2	Message Sequence Diagrams.....	299
8.3.3	State Diagrams.....	400
<b>9.</b>	<b>System Policy .....</b>	<b>482</b>
9.1	Overview .....	482
9.1.1	PDUSB Device and Hub Requirements .....	484
9.1.2	Mapping to USB Device States.....	485
9.1.3	PD Software Stack .....	487
9.1.4	PDUSB Device Enumeration .....	488
9.2	PD Class Specific Descriptors.....	489
9.2.1	USB Power Delivery Capability Descriptor .....	489
9.2.2	Battery Info Capability Descriptor .....	491
9.2.3	PD Consumer Port Capability Descriptor .....	492
9.2.4	PD Provider Port Capability Descriptor .....	493
9.3	PD Class Specific Requests and Events .....	495
9.3.1	Class-specific Requests .....	495
9.3.2	PDUSB Hub Event Reporting .....	496
9.4	PDUSB Hub and PDUSB Peripheral Device Requests .....	498
9.4.1	ClearPortPDFeature (PDUSB Hub) .....	498
9.4.2	GetBatteryStatus (PDUSB Hub/Peripheral Device) .....	499
9.4.3	GetPortPartnerPDOObjects (PDUSB Hub) .....	500
9.4.4	GetPortPDStatus (PDUSB Hub) .....	501
9.4.5	SetPDFeature (PDUSB Hub/Peripheral Device).....	503
9.4.6	SetPortPDFeature (PDUSB Hub).....	505
9.4.7	SetPortPDOs (PDUSB Hub) .....	506
9.4.8	GetVDM (PDUSB Hub).....	506
9.4.9	SendVDM (PDUSB Hub).....	507
<b>A.</b>	<b>Power Profiles.....</b>	<b>508</b>
A.1	Profile Definitions.....	508
A.2	Voltage Selection Rationale .....	509
A.3	Relevance of Profiles to Sink Ports.....	509
A.4	System Level Examples.....	510
A.4.1	Notebook and HDD Examples .....	510

A.4.2	Display with Hub Example .....	511
<b>B.</b>	<b>CRC calculation.....</b>	<b>513</b>
B.1	C code example .....	513
B.2	Table showing the full calculation over one Message .....	515
<b>C.</b>	<b>Power Implementation Considerations .....</b>	<b>516</b>
C.1	Managing Isolation Impedance (BFSK) .....	516
C.1.1	In-band fCarrier Spurious Noise .....	516
C.1.2	Spurious Noise Test Setup and Calibration .....	517
C.2	Connector Detach Transients .....	520
C.3	Closed Loop Stability Effects .....	521
C.3.1	Basic Power Stage Small Signal AC Model.....	521
C.3.2	Feedback Past Isolation Inductor .....	523
<b>D.</b>	<b>Standard-A Mating Illustrations .....</b>	<b>525</b>
<b>E.</b>	<b>Physical Layer Informative Material .....</b>	<b>532</b>
E.1	Squelch Budget .....	533
<b>F.</b>	<b>PD Message Sequence Examples .....</b>	<b>535</b>
F.1	External power is supplied downstream.....	535
F.2	External power is supplied upstream .....	539
F.3	Giving back power.....	546
<b>G.</b>	<b>VDM Command Examples .....</b>	<b>559</b>
G.1	Discover Identity Example.....	559
G.1.1	Discover Identity Command request .....	559
G.1.2	Discover Identity Command response – Active Cable.....	559
G.1.3	Discover Identity Command response – Hub .....	560
G.2	Discover SVIDs Example .....	562
G.2.1	Discover SVIDs Command request.....	562
G.2.1	Discover SVIDs Command response.....	562
G.3	Discover Modes Example .....	564
G.3.1	Discover Modes Command request .....	564
G.3.2	Discover Modes Command response .....	564
G.4	Enter Mode Example .....	566
G.4.1	Enter Mode Command request.....	566

G.4.2	Enter Mode Command response.....	566
G.4.1	Enter Mode Command request with additional VDO .....	567
G.5	Exit Mode Example .....	569
G.5.1	Exit Mode Command request .....	569
G.5.2	Exit Mode Command response .....	569
G.6	Attention Example.....	571
G.6.1	Attention Command request .....	571
G.6.2	Attention Command request with additional VDO.....	571

## List of Tables

Table 1-1 Terms and Abbreviations .....	39
Table 3-1 Plugs Accepted By Receptacles .....	64
Table 3-2 USB 2.0 PD Standard-A Connector Pin Assignments .....	74
Table 3-3 USB 3.1 PD Standard-A Connector Pin Assignments .....	78
Table 3-4 USB 2.0 PD Standard-B Connector Pin Assignments.....	81
Table 3-5 USB 3.1 PD Standard-B Connector Pin Assignments.....	85
Table 3-6 USB PD Cable Assembly Overmold Maximum Dimensions .....	86
Table 3-7 Electrical Parameters .....	95
Table 4-1 Normal Dead Battery Operation.....	99
Table 4-2 Plug Type Determination.....	104
Table 4-3 Electrical Parameters .....	105
Table 4-4 Electrical Timers.....	105
Table 5-1 4b5b Symbol Encoding Table .....	107
Table 5-2 Ordered Sets .....	111
Table 5-3 Validation of Ordered Sets .....	111
Table 5-4 Data Size .....	111
Table 5-5 SOP ordered set.....	113
Table 5-6 SOP' ordered set .....	114
Table 5-7 SOP'' ordered set .....	114
Table 5-8 SOP'_Debug ordered set .....	115
Table 5-9 SOP''_Debug ordered set .....	115
Table 5-10 CRC-32 Mapping .....	116
Table 5-11 Hard Reset ordered set .....	117
Table 5-12 Cable Reset ordered set .....	118
Table 5-13 Common Normative Signaling Scheme Requirements .....	119
Table 5-14 Common Normative Signaling Scheme Requirements for Transmitter .....	119
Table 5-15 Common Normative Signaling Scheme Requirements for Receiver .....	119
Table 5-16 BFSK Common Normative Requirements .....	123
Table 5-17 BFSK Transceiver Isolation Impedance Normative Requirements .....	123
Table 5-18 BFSK Transmitter Normative Requirements.....	123
Table 5-19 BFSK Spectrum Mask Corners .....	126
Table 5-20 BFSK Receiver Normative Requirements.....	127
Table 5-21 BMC Tx Mask Definition, X Values .....	134
Table 5-22 BMC Tx Mask Definition, Y Values .....	135

Table 5-23 BMC Rx Mask Definition .....	140
Table 5-24 BMC Common Normative Requirements.....	142
Table 5-25 BMC Transmitter Normative Requirements .....	142
Table 5-26 BMC Receiver Normative Requirements .....	143
Table 5-27 Allowable Bit Errors vs. Number of Test Frames.....	148
Table 5-28 BIST Parameters .....	149
Table 5-29 BIST Mode support.....	149
Table 6-1 Message Header .....	152
Table 6-2 Control Message Types .....	154
Table 6-3 Data Message Types.....	160
Table 6-4 Power Data Object .....	161
Table 6-5 Type-A to Type-A Port Behavior .....	163
Table 6-6 Fixed Supply PDO - Source.....	163
Table 6-7 Fixed Power Source Peak Current Capability .....	164
Table 6-8 Variable Supply (non-battery) PDO - Source.....	165
Table 6-9 Battery Supply PDO - Source.....	165
Table 6-10 Fixed Supply PDO - Sink .....	166
Table 6-11 Variable Supply (non-battery) PDO - Sink .....	167
Table 6-12 Battery Supply PDO - Sink .....	167
Table 6-13 Fixed and Variable Request Data Object .....	168
Table 6-14 Fixed and Variable Request Data Object with GiveBack Support .....	168
Table 6-15 Battery Request Data Object.....	168
Table 6-16 Battery Request Data Object with GiveBack Support.....	169
Table 6-17 BIST Data Object .....	173
Table 6-18 Unstructured VDM Header .....	176
Table 6-19 Structured VDM Header .....	176
Table 6-20 Structured VDM Commands .....	177
Table 6-21 SVID Values .....	178
Table 6-22 Commands and Responses.....	179
Table 6-23 ID Header VDO.....	180
Table 6-24 Product Types .....	181
Table 6-25 Cert Stat VDO.....	182
Table 6-26 Product VDO .....	182
Table 6-27 Cable VDO .....	183
Table 6-28 AMA VDO.....	184
Table 6-29 Discover SVIDs Responder VDO .....	185



Table 6-30 Time Values.....	200
Table 6-31 Timers.....	202
Table 6-32 Counter parameters .....	204
Table 6-33 Counters .....	205
Table 6-34 Protocol Layer States.....	224
Table 6-35 Applicability of Control Messages.....	226
Table 6-36 Applicability of Type-C Specific Control Messages.....	227
Table 6-37 Applicability of Data Messages.....	227
Table 6-38 Applicability of VDM Commands.....	228
Table 7-1 Noise Spectral Mask Corners .....	238
Table 7-2 Noise Spectral Mask Corners .....	242
Table 7-3 Sequence Description for Increasing the Current .....	246
Table 7-4 Sequence Description for Increasing the Voltage .....	248
Table 7-5 Sequence Diagram for Increasing the Voltage and Current .....	250
Table 7-6 Sequence Description for Increasing the Voltage and Decreasing the Current .....	252
Table 7-7 Sequence Description for Decreasing the Voltage and Increasing the Current .....	254
Table 7-8 Sequence Description for Decreasing the Current .....	256
Table 7-9 Sequence Description for Decreasing the Voltage.....	258
Table 7-10 Sequence Description for Decreasing the Voltage and the Current.....	260
Table 7-11 Sequence Description for a Sink Requested Power Role Swap.....	262
Table 7-12 Sequence Description for a Source Requested Power Role Swap .....	265
Table 7-13 Sequence Description for a GotoMin Current Decrease.....	268
Table 7-14 Sequence Description for a Source Initiated Hard Reset.....	270
Table 7-15 Sequence Description for a Sink Initiated Hard Reset .....	272
Table 7-16 Sequence Description for Type-B New Source Initiated Hard Reset and Type-A New Sink Receives Hard Reset Signaling .....	275
Table 7-17 Sequence Description for Type-B New Source Initiated Hard Reset and Type-A New Sink Does Not Receive Hard Reset Signaling.....	277
Table 7-18 Sequence Description for Type-A New Sink Initiated Hard Reset and Type-B New Source Receives Hard Reset Signaling .....	279
Table 7-19 Sequence Description for Type-A New Sink Initiated Hard Reset and Type-B New Source Does Not Receive Hard Reset Signaling.....	281
Table 7-20 Sequence Description for Type-A to Type-B Dead Battery Operation .....	283
Table 7-21 Sequence Description for no change in Current or Voltage .....	285
Table 7-22 Source Electrical Parameters .....	286
Table 7-23 Sink Electrical Parameters .....	289
Table 7-24 Common Source/Sink Electrical Parameters.....	291

Table 8-1 Basic Message Flow .....	300
Table 8-2 Potential issues in Basic Message Flow .....	301
Table 8-3 Basic Message Flow with CRC failure.....	302
Table 8-4 Steps for a successful Power Negotiation.....	305
Table 8-5 Steps for a GotoMin Negotiation.....	308
Table 8-6 Steps for a Soft Reset .....	310
Table 8-7 Steps for Source initiated Hard Reset .....	314
Table 8-8 Steps for Sink initiated Hard Reset .....	317
Table 8-9 Steps for Source initiated Hard Reset – Sink long reset.....	320
Table 8-10 Steps for a Successful Type-A or Type-B Source Initiated Power Role Swap Sequence.....	324
Table 8-11 Steps for a Successful Type-A or Type-B Sink Initiated Power Role Swap Sequence .....	328
Table 8-12 Steps for Type-A or Type-B Source initiated Hard Reset (Power Role Swapped) .....	332
Table 8-13 Steps for Type-A or Type-B Sink initiated Hard Reset (Power Role Swapped) .....	336
Table 8-14 Steps for a Successful Type-C Source Initiated Power Role Swap Sequence .....	340
Table 8-15 Steps for a Successful Type-C Sink Initiated Power Role Swap Sequence .....	345
Table 8-16 Steps for Type-C Data Role Swap, UFP operating as Sink initiates .....	349
Table 8-17 Steps for Type-C Data Role Swap, UFP operating as Source initiates .....	352
Table 8-18 Steps for Type-C Data Role Swap, DFP operating as Source initiates .....	355
Table 8-19 Steps for Type-C Data Role Swap, DFP operating as Sink initiates .....	358
Table 8-20 Steps for Type-C DFP to UFP VCONN Source Swap .....	361
Table 8-21 Steps for Type-C UFP to DFP VCONN Source Swap .....	364
Table 8-22 Steps for DFP to UFP Discover Identity .....	366
Table 8-23 Steps for Source Port to Cable Plug Discover Identity .....	368
Table 8-24 Steps for DFP to Cable Plug Discover Identity .....	370
Table 8-25 Steps for DFP to UFP Enter Mode.....	375
Table 8-26 Steps for DFP to UFP Exit Mode .....	379
Table 8-27 Steps for DFP to Cable Plug Enter Mode .....	383
Table 8-28 Steps for DFP to Cable Plug Exit Mode .....	387
Table 8-29 Steps for UFP to DFP Attention .....	389
Table 8-30 Steps for BIST Receiver Mode test .....	393
Table 8-31 Steps for BIST Transmit Mode test .....	396
Table 8-32 Steps for BIST Eye Pattern Test .....	399
Table 8-33 Policy Engine States .....	476
Table 9-1 USB Power Delivery Type Codes .....	489
Table 9-2 USB Power Delivery Capability Descriptor .....	489
Table 9-3 Battery Info Capability Descriptor .....	491

Table 9-4 PD Consumer Port Descriptor .....	493
Table 9-5 PD Provider Port Descriptor .....	494
Table 9-6 PD Class Requests .....	495
Table 9-7 PD Class Request Codes .....	495
Table 9-8 PD Class Feature Selectors .....	496
Table 9-9 Clear Change Mask.....	498
Table 9-10 Battery Status Structure .....	499
Table 9-11 Port PD Status .....	501
Table 9-12 Battery Wake Mask.....	503
Table 9-13 Policy Mode Encoding .....	504
Table 9-14 Charging Policy Encoding .....	504
Table F-1 External power is supplied downstream.....	536
Table F-2 External power is supplied downstream.....	539
Table F-3 Giving back power .....	546
Table G-1 Discover Identity Command request from Initiator Example .....	559
Table G-2 Discover Identity Command response from Active Cable Responder Example .....	559
Table G-3 Discover Identity Command response from Hub Responder Example .....	561
Table G-4 Discover SVIDs Command request from Initiator Example.....	562
Table G-5 Discover SVIDs Command response from Responder Example .....	562
Table G-6 Discover Modes Command request from Initiator Example .....	564
Table G-7 Discover Modes Command response from Responder Example .....	564
Table G-8 Enter Mode Command request from Initiator Example .....	566
Table G-9 Enter Mode Command response from Responder Example.....	566
Table G-10 Enter Mode Command request from Initiator Example .....	567
Table G-11 Exit Mode Command request from Initiator Example.....	569
Table G-12 Exit Mode Command response from Responder Example .....	569
Table G-13 Attention Command request from Initiator Example .....	571
Table G-14 Attention Command request from Initiator with additional VDO Example .....	571

## List of Figures

Figure 2-1 Logical Structure of USB Power Delivery Capable Devices .....	48
Figure 2-2 SOP' Communication between Source and Cable Plug with no Explicit Contract or an Implicit Contract .....	49
Figure 2-3 SOP' Communication between DFP and Cable Plug with PD Explicit Contract .....	49
Figure 2-4 SOP'' Communication with the attached Cable Plugs .....	50
Figure 2-5 USB Power Delivery Communications Stack .....	58
Figure 2-6 USB Power Delivery Communication Over USB .....	59
Figure 2-7 High Level Architecture View .....	60
Figure 3-1 Standard-A Insertion Detect Schematic Representation .....	66
Figure 3-2 PD Standard-A No Plug Detection Circuit .....	67
Figure 3-3 Non-PD Plug Standard-A Detection Circuit .....	67
Figure 3-4 USB Thin Card Standard-A Detection Circuit .....	67
Figure 3-5 PD Plug Standard-A Detection Circuit .....	67
Figure 3-6 Insertion Detect Zone Mechanical Dimensions for the Standard-A Receptacle .....	68
Figure 3-7 PD Detect Plane Location Range for PD Standard-A Receptacles .....	69
Figure 3-8 PD Standard-A Plug Interface Dimensions .....	71
Figure 3-9 USB 2.0 PD Standard-A Receptacle Interface Dimensions .....	72
Figure 3-10 Reference Footprint for the USB 2.0 PD Standard-A Top Mount Single Receptacle (Informative) .....	73
Figure 3-11 USB 3.1 PD Standard-A Plug Interface Dimensions .....	75
Figure 3-12 Reference USB 3.1 PD Standard-A Receptacle Interface Dimensions (Informative) .....	76
Figure 3-13 Reference Footprint for the USB 3.1 PD Standard-A Top Mount Single Receptacle (Informative) .....	77
Figure 3-14 USB 2.0 PD Standard-B Plug Interface Dimensions .....	79
Figure 3-15 USB 2.0 PD Standard-B Receptacle Interface Dimensions .....	80
Figure 3-16 Reference Footprint for the USB 2.0 PD Standard-B Receptacle .....	81
Figure 3-17 USB 3.1 PD Standard-B Plug Interface Dimensions .....	82
Figure 3-18 USB 3.1 PD Standard-B Receptacle Interface Dimensions .....	83
Figure 3-19 Reference Footprint for the USB 3.1 PD Standard-B Receptacle .....	84
Figure 3-20 USB PD Cable Assembly Overmold Maximum Dimensions .....	87
Figure 3-21 Schematic of a Micro-A Plug Legacy Termination .....	88
Figure 3-22 Schematic of a Micro-A Plug Marker Indicating Low Power Capability .....	89
Figure 3-23 Schematic of a Micro-A PD Plug .....	89
Figure 3-24 Schematic of a B Plug Connector Marker Indicating 3A Capability .....	90
Figure 3-25 Schematic of a B Plug Connector Marker Indicating 5A Capability .....	90
Figure 3-26 Differential Near-End and Far-End Crosstalk Requirement between the D+/D- Pair and V <sub>BUS</sub> .....	92
Figure 3-27 Voltage Drop Measurement .....	93

Figure 4-1 Type-A to Type-B Dead Battery / Unpowered Port Detection Flow .....	98
Figure 4-2 Plug Type Determination.....	102
Figure 4-3 Standard-A Plug PD Capabilities Flow .....	103
Figure 4-4 Plug Type Detection Circuit .....	103
Figure 5-1 Interpretation of ordered sets .....	110
Figure 5-2 Transmit Order for Various Sizes of Data.....	112
Figure 5-3 USB Power Delivery Packet Format.....	113
Figure 5-4 CRC 32 generation .....	116
Figure 5-5 Line format of Hard Reset .....	118
Figure 5-6 Line format of Cable Reset .....	118
Figure 5-7 Inter-Frame Gap Timings .....	120
Figure 5-8 Transmitter Block Diagram .....	121
Figure 5-9 Receiver Block Diagram .....	122
Figure 5-10 Channel Diagram (Cable Type Detection not shown) .....	122
Figure 5-11 Eye diagram of BFSK Modulation.....	125
Figure 5-12 BFSK Transmit Spectral Mask, given in absolute terms relative to the maximum value of $v_{TX}$ .....	126
Figure 5-13 Line Format of Bit Stream.....	128
Figure 5-14 BMC Example.....	129
Figure 5-15 BMC Transmitter Block Diagram.....	129
Figure 5-16 BMC Receiver Block Diagram.....	130
Figure 5-17 BMC Encoded Start of Preamble .....	130
Figure 5-18 Transmitting or Receiving BMC Encoded Frame Terminated by Zero with High-to-Low Last Transition	131
Figure 5-19 Transmitting or Receiving BMC Encoded Frame Terminated by One with High-to-Low Last Transition	131
Figure 5-20 Transmitting or Receiving BMC Encoded Frame Terminated by Zero with Low to High Last Transition	131
Figure 5-21 Transmitting or Receiving BMC Encoded Frame Terminated by One with Low to High Last Transition	132
Figure 5-22 Waiting for idle after a BMC Encoded Frame Terminated by Zero with High-to-Low Last Transition .....	132
Figure 5-23 BMC Tx 'ONE' Mask.....	133
Figure 5-24 BMC Tx 'ZERO' Mask .....	134
Figure 5-25 BMC Rx 'ONE' Mask when Sourcing Power .....	136
Figure 5-26 BMC Rx 'ZERO' Mask when Sourcing Power .....	137
Figure 5-27 BMC Rx 'ONE' Mask when Power neutral .....	138
Figure 5-28 BMC Rx 'ZERO' Mask when Power neutral.....	138
Figure 5-29 BMC Rx 'ONE' Mask when Sinking Power .....	139
Figure 5-30 BMC Rx 'ZERO' Mask when Sinking Power .....	139
Figure 5-31 Transmitter Load Model for BMC Tx from a Source .....	141
Figure 5-32 Transmitter Load Model for BMC Tx from a Sink .....	141

Figure 5-33 Transmitter diagram illustrating zDriver .....	143
Figure 5-34 Example Multi-Drop Configuration showing two DRPs .....	144
Figure 5-35 Example Multi-Drop Configuration showing a DFP and UFP.....	145
Figure 5-36 Example implementation of the BIST generator and checker .....	147
Figure 5-37 Test Frame .....	147
Figure 5-38 Test Data Frame .....	149
Figure 6-1 High Level Message Structure .....	151
Figure 6-2 USB Power Delivery Packet Format including Message Payload .....	151
Figure 6-3 Example Capabilities Message with 2 Power Data Objects .....	160
Figure 6-4 BIST Message .....	172
Figure 6-5 Vendor Defined Message.....	175
Figure 6-6 Discover Identity Command response.....	180
Figure 6-7 Example Discover SVIDs response with 3 SVIDs .....	185
Figure 6-8 Example Discover SVIDs response with 4 SVIDs .....	185
Figure 6-9 Example Discover SVIDs response with 12 SVIDs followed by an empty response .....	185
Figure 6-10 Example Discover Modes response for a given SVID with 3 Modes .....	186
Figure 6-11 Successful Enter Mode sequence.....	187
Figure 6-12 Unsuccessful Enter Mode sequence due to NAK .....	187
Figure 6-13 Exit Mode sequence .....	188
Figure 6-14 Attention Command request/response sequence .....	189
Figure 6-15 Command request/response sequence .....	189
Figure 6-16 Enter/Exit Mode Process .....	191
Figure 6-17 Vendor Defined Message interrupted by a Power Delivery Message .....	192
Figure 6-18 Outline of States .....	207
Figure 6-19 References to states.....	207
Figure 6-20 Protocol Layer Message transmission.....	208
Figure 6-21 Protocol layer Message reception .....	211
Figure 6-22 Hard/Cable Reset .....	215
Figure 6-23 BIST Transmitter Test .....	220
Figure 6-24 BIST Receiver Test .....	222
Figure 7-1 Placement of Source Bulk Capacitance .....	229
Figure 7-2 Transition Envelope for Positive Voltage Transitions .....	230
Figure 7-3 Transition Envelope for Negative Voltage Transitions .....	231
Figure 7-4 Source $V_{BUS}$ Response to Hard Reset.....	232
Figure 7-5 Source $V_{CONN}$ Response to Hard Reset .....	233
Figure 7-6 Application of vSrcNew and vSrcValid limits after tSrcReady .....	234

Figure 7-7 Source Peak Current Overload .....	235
Figure 7-8 Noise Spectral Mask, given in absolute terms relative to the maximum value of $v_{TX}$ .....	237
Figure 7-9 vSafeDB Operating Region .....	238
Figure 7-10 Placement of Sink Bulk Capacitance .....	240
Figure 7-11 Noise Spectral Mask, given in absolute terms relative to the maximum value of $v_{TX}$ .....	242
Figure 7-12 Transition Diagram for Increasing the Current .....	245
Figure 7-13 Transition Diagram for Increasing the Voltage .....	247
Figure 7-14 Transition Diagram for Increasing the Voltage and Current .....	249
Figure 7-15 Transition Diagram for Increasing the Voltage and Decreasing the Current .....	251
Figure 7-16 Transition Diagram for Decreasing the Voltage and Increasing the Current .....	253
Figure 7-17 Transition Diagram for Decreasing the Current .....	255
Figure 7-18 Transition Diagram for Decreasing the Voltage .....	257
Figure 7-19 Transition Diagram for Decreasing the Voltage and the Current .....	259
Figure 7-20 Transition Diagram for a Sink Requested Power Role Swap .....	261
Figure 7-21 Transition Diagram for a Source Requested Power Role Swap .....	264
Figure 7-22 Transition Diagram for a GotoMin Current Decrease .....	267
Figure 7-23 Transition Diagram for a Source Initiated Hard Reset .....	269
Figure 7-24 Transition Diagram for a Sink Initiated Hard Reset .....	271
Figure 7-25 Transition Diagram for Type-B New Source Initiated Hard Reset and Type-A New Sink Receives Hard Reset Signaling .....	274
Figure 7-26 Transition Diagram for Type-B New Source Initiated Hard Reset and Type-A New Sink Does Not Receive Hard Reset Signaling .....	276
Figure 7-27 Transition Diagram for Type-A New Sink Initiated Hard Reset and Type-B New Source Receives Hard Reset Signaling .....	278
Figure 7-28 Transition Diagram for Type-A New Sink Initiated Hard Reset and Type-B New Source Does Not Receive Hard Reset Signaling .....	280
Figure 7-29 Type-A to Type-B Transition Diagram for Dead Battery Operation .....	282
Figure 7-30 Transition Diagram for no change in Current or Voltage .....	284
Figure 8-1 Example of daisy chained displays .....	297
Figure 8-2 Basic Message Exchange (Successful) .....	299
Figure 8-3 Basic Message flow indicating possible errors .....	300
Figure 8-4 Basic Message Flow with Bad CRC followed by a Retry .....	302
Figure 8-5 Successful Power Negotiation .....	304
Figure 8-6 Successful GotoMin operation .....	308
Figure 8-7 Soft Reset .....	310
Figure 8-8 Source initiated Hard Reset .....	313
Figure 8-9 Sink Initiated Hard Reset .....	316

Figure 8-10 Source initiated reset - Sink long reset .....	319
Figure 8-11 Type-A or Type-B Successful Power Role Swap Sequence Initiated by the Source .....	323
Figure 8-12 Type-A or Type-B Successful Power Role Swap Sequence Initiated by the Sink .....	327
Figure 8-13 Type-A or Type-B Source initiated Hard Reset (Power Role Swapped) .....	331
Figure 8-14 Type-A or Type-B Sink Initiated Hard Reset (Power Role Swapped) .....	334
Figure 8-15 Type-C Successful Power Role Swap Sequence Initiated by the Source .....	339
Figure 8-16 Type-C Successful Power Role Swap Sequence Initiated by the Type-C Sink .....	344
Figure 8-17 Type-C Data Role Swap, UFP operating as Sink initiates .....	348
Figure 8-18 Type-C Data Role Swap, UFP operating as Source initiates .....	351
Figure 8-19 Type-C Data Role Swap, DFP operating as Source initiates .....	354
Figure 8-20 Type-C Data Role Swap, DFP operating as Sink initiates .....	357
Figure 8-21 Type-C DFP to UFP VCONN Source Swap .....	360
Figure 8-22 Type-C UFP to DFP VCONN Source Swap .....	363
Figure 8-23 DFP to UFP Discover Identity .....	366
Figure 8-24 Source Port to Cable Plug Discover Identity .....	368
Figure 8-25 DFP to Cable Plug Discover Identity .....	370
Figure 8-26 DFP to UFP Enter Mode .....	374
Figure 8-27 DFP to UFP Exit Mode .....	379
Figure 8-28 DFP to Cable Plug Enter Mode .....	382
Figure 8-29 DFP to Cable Plug Exit Mode .....	387
Figure 8-30 UFP to DFP Attention .....	389
Figure 8-31 BIST Receiver Mode test .....	392
Figure 8-32 BIST Transmit Mode test .....	395
Figure 8-33 BIST Eye Pattern Test .....	398
Figure 8-34 Outline of States .....	400
Figure 8-35 References to states .....	400
Figure 8-36 Example of state reference with conditions .....	400
Figure 8-37 Example of state reference with the same entry and exit .....	401
Figure 8-38 Source Port Policy Engine state diagram .....	402
Figure 8-39 Sink Port state diagram .....	408
Figure 8-40 Source Port Soft Reset Diagram .....	413
Figure 8-41 Sink Port Soft Reset Diagram .....	415
Figure 8-42 Source Port Ping State Diagram .....	417
Figure 8-43 Dual-Role (initially Source Port) Ping State Diagram .....	417
Figure 8-44 Dual-Role (initially Sink Port) Ping State Diagram .....	418
Figure 8-45 State Diagram for Hard Reset of P/C in Sink Role .....	418



Figure 8-46 State Diagram for the Hard Reset of a C/P in Source Role .....	419
Figure 8-47 Consumer/Provider Dead Battery/Power Loss State Diagram .....	421
Figure 8-48 BFSK Provider/Consumer Dead Battery/Power Loss State Diagram .....	424
Figure 8-49: Type-C DFP to UFP Data Role Swap State Diagram .....	426
Figure 8-50: Type-C UFP to DFP Data Role Swap State Diagram .....	429
Figure 8-51: Dual-Role Port in Source to Sink Power Role Swap State Diagram .....	432
Figure 8-52: Dual-role Port in Sink to Source Power Role Swap State Diagram .....	435
Figure 8-53 Dual-Role (Source) Get Source Capabilities diagram .....	438
Figure 8-54 Dual-Role (Source) Give Sink Capabilities diagram .....	438
Figure 8-55 Dual-Role (Sink) Get Sink Capabilities State Diagram .....	439
Figure 8-56 Dual-Role (Sink) Give Source Capabilities State Diagram .....	439
Figure 8-57 VCONN Swap State Diagram .....	441
Figure 8-58 UFP Structured VDM Discover Identity State Diagram .....	444
Figure 8-59 UFP Structured VDM Discover SVIDs State Diagram .....	445
Figure 8-60 UFP Structured VDM Discover Modes State Diagram .....	446
Figure 8-61 UFP Structured VDM Enter Mode State Diagram .....	447
Figure 8-62 UFP Structured VDM Exit Mode State Diagram .....	448
Figure 8-63 UFP VDM Attention State Diagram .....	449
Figure 8-64 DFP to UFP VDM Discover Identity State Diagram .....	450
Figure 8-65 DFP VDM Discover Identity State Diagram .....	452
Figure 8-66 DFP VDM Discover SVIDs State Diagram .....	454
Figure 8-67 DFP VDM Discover Modes State Diagram .....	455
Figure 8-68 DFP VDM Mode Entry State Diagram .....	456
Figure 8-69 DFP VDM Mode Exit State Diagram .....	457
Figure 8-70 DFP VDM Attention State Diagram .....	459
Figure 8-71 Cable Ready VDM State Diagram .....	459
Figure 8-72 Cable Plug Structured VDM Discover Identity State Diagram .....	460
Figure 8-73 Cable Plug Structured VDM Discover SVIDs State Diagram .....	461
Figure 8-74 Cable Plug Structured VDM Discover Modes State Diagram .....	462
Figure 8-75 CablePlug Structured VDM Enter Mode State Diagram .....	463
Figure 8-76 CablePlug Structured VDM Exit Mode State Diagram .....	464
Figure 8-77 Cable Plug Soft Reset State Diagram .....	465
Figure 8-78 Cable Plug Hard Reset State Diagram .....	466
Figure 8-79 DFP Soft Reset or Cable Reset of a Cable Plug State Diagram .....	467
Figure 8-80 UFP Source Soft Reset of a Cable Plug State Diagram .....	468
Figure 8-81 Source Startup Structured VDM Discover Identity State Diagram .....	469

Figure 8-82 BIST Receive Mode State Diagram .....	470
Figure 8-83 BIST Transmit Mode State Diagram .....	472
Figure 8-84 BIST Carrier Mode and Eye Pattern State Diagram .....	473
Figure 9-1 Example PD Topology .....	483
Figure 9-2 Mapping of PD Topology to USB .....	484
Figure 9-3 USB Attached to USB Powered State Transition .....	485
Figure 9-4 Any USB State to USB Attached State Transition (When operating as a Consumer) .....	486
Figure 9-5 Any USB State to USB Attached State Transition (When operating as a Provider) .....	486
Figure 9-6 Any USB State to USB Attached State Transition (After a Type-C Data Role Swap) .....	487
Figure 9-7 Software stack on a PD aware OS .....	487
Figure 9-8 Enumeration of a PDUSB Device .....	488
Figure 9-9 Hub Status Change .....	497
Figure A-1 Interpretation of UL60950 .....	509
Figure A-2 Notebook is Capable of Meeting User's Requirements .....	510
Figure A-3 Notebook is not Capable of Meeting User's Requirements .....	511
Figure A-4 Display with Integrated Hub Capable of Meeting User's Requirements .....	512
Figure C-1 Typical System Electrical Model .....	516
Figure C-2 Typical Synchronous Buck Power Stage with Parasitics .....	517
Figure C-3 Spurious Noise Measurement Test Setup.....	518
Figure C-4 Current Transients when Cable/Load Removed.....	520
Figure C-5 Isolation Inductor Energy versus Load .....	521
Figure C-6 Simplified Small Signal AC Model.....	522
Figure C-7 Power Stage Phase And Gain with and without Isolation Inductors .....	523
Figure C-8 Simplified Small Signal AC Model (Feedback before and after Inductor zIsolation_P).....	524
Figure C-9 Simplified Small Signal AC Model (Feedback before and after Inductor zIsolation_P).....	524
Figure D-1 USB 3.1 Standard-A Plug with USB 2.0 PD or 3.1 PD Standard-A Receptacle.....	525
Figure D-2 USB 3.1 PD Standard-A Plug with USB 2.0 or 3.1 Standard-A Receptacle.....	526
Figure D-3 USB 2.0 PD or 3.1 PD Standard-A plug with USB 2.0 PD or 3.1 PD Standard-A receptacle .....	527
Figure D-4 USB 2.0 PD or 3.1 PD Standard-A Plug with USB 2.0 Standard-A Receptacle.....	528
Figure D-5 USB 2.0 Standard-A Plug with USB 2.0 or USB3.1 PD Standard-A Receptacle .....	529
Figure D-6 USB 2.0 Thin Card with USB 2.0 PD or 3.1 PD Standard-A Receptacle .....	530
Figure D-7 USB 3.1 Thin Card with USB 2.0 PD or 3.1 PD Standard-A Receptacle .....	531
Figure E-1 Squelch Budget .....	533
Figure F-1 External Power supplied downstream .....	535
Figure F-2 External Power supplied upstream.....	539
Figure F-3 Giving Back Power .....	546



## 1. Introduction

USB has evolved from a data interface capable of supplying limited power to a primary provider of power with a data interface. Today many devices charge or get their power from USB ports contained in laptops, cars, aircraft or even wall sockets. USB has become a ubiquitous power socket for many small devices such as cell phones, MP3 players and other hand-held devices. Users need USB to fulfill their requirements not only in terms of data but also to provide power to, or charge, their devices simply, often without the need to load a driver, in order to carry out “traditional” USB functions.

There are however, still many devices which either require an additional power connection to the wall, or exceed the USB rated current in order to operate. Increasingly, international regulations require better energy management due to ecological and practical concerns relating to the availability of power. Regulations limit the amount of power available from the wall which has led to a pressing need to optimize power usage. The USB Power Delivery Specification has the potential to minimize waste as it becomes a standard for charging devices that are not satisfied by [\[BC 1.2\]](#).

Wider usage of wireless solutions is an attempt to remove data cabling but the need for “tethered” charging remains. In addition, industrial design requirements drive wired connectivity to do much more over the same connector.

USB Power Delivery is designed to enable the maximum functionality of USB by providing more flexible power delivery along with data over a single cable. Its aim is to operate with and build on the existing USB ecosystem; increasing power levels from existing USB standards, for example Battery Charging, enabling new higher power use cases such as USB powered Hard Disk Drives (HDDs) and printers.

With USB Power Delivery the power direction is no longer fixed. This enables the product with the power (Host or Peripheral) to provide the power. For example, a display with a supply from the wall can power, or charge, a laptop. Alternatively, USB power bricks or chargers are able to supply power to laptops and other battery powered devices through their, traditionally power providing, USB ports.

USB Power Delivery enables hubs to become the means to optimize power management across multiple peripherals by allowing each device to take only the power it requires, and to get more power when required for a given application. For example battery powered devices can get increased charging current and then give it back temporarily when the user’s HDD requires to spin up. Optionally the hubs can communicate with the PC to enable even more intelligent and flexible management of power either automatically or with some level of user intervention.

USB Power Delivery allows Low Power cases such as headsets to negotiate for only the power they require. This provides a simple solution that enables USB devices to operate at their optimal power levels.

The Power Delivery Specification, in addition to providing mechanisms to negotiate power also can be used as a side-band channel for standard and vendor defined messaging. Power Delivery enables alternative modes of operation by providing the mechanisms to discover, enter and exit [Alternate Modes](#). The specification also enables discovery of cable capabilities such as supported speeds and current levels.

### 1.1 Overview

This specification defines how USB Devices may negotiate for more current and/or higher or lower voltages over the USB cable (using  $V_{BUS}$  or CC wire as the communications channel) than are defined in the [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USBType-C 1.0\]](#) or [\[BC 1.2\]](#) specifications. It allows Devices with greater power requirements than can be met with today’s specification to get the power they require to operate from  $V_{BUS}$  and negotiate with external power sources (e.g. wall warts). In addition, it allows a Source and Sink to swap power roles such that a Device could supply power to the Host. For example, a display could supply power to a notebook to charge its battery.

The USB Power Delivery Specification is guided by the following principles:

- 1) Works seamlessly with legacy USB Devices
- 2) Compatible with existing spec-compliant USB cables
- 3) Minimizes potential damage from non-compliant cables (e.g. ‘Y’ cables etc.)
- 4) Optimized for low-cost implementations

This specification defines mechanisms to discover, enter and exit Modes defined either by a standard or by a particular vendor. These Modes can be supported either by the Port Partner or by a cable connecting the two Port Partners.

The specification defines mechanisms to discover the capabilities of cables which can communicate using Power Delivery [signaling](#).

For Type-C Connectors this specification adds a mechanism to swap the data roles such that the upstream facing Port becomes the downstream facing Port and vice versa. It also enables a swap of the end supplying  $V_{CONN}$  to a powered cable.

## 1.2 Purpose

The USB Power Delivery specification defines a power delivery system covering all elements of a USB system including: Hosts, Devices, Hubs, Chargers and cable assemblies. This specification describes the architecture, protocols, power supply behavior, connectors and cabling necessary for managing power delivery over USB at up to 100W. This specification is intended to be fully compatible and extend the existing USB infrastructure. It is intended that this specification will allow system OEMs, power supply and peripheral developers adequate flexibility for product versatility and market differentiation without losing backwards compatibility.

USB Power Delivery is designed to operate independently of the existing USB bus defined mechanisms used to negotiate power which are:

- [\[USB 2.0\]](#), [\[USB 3.1\]](#) in band requests for high power interfaces.
- [\[BC1.2\]](#) mechanisms for supplying higher power (not mandated by this specification).
- [\[USBType-C 1.0\]](#) mechanisms for supplying higher power

Initial operating conditions remain the USB Default Operation as defined in [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USBType-C 1.0\]](#) or [\[BC 1.2\]](#).

- The DFP sources *vSafe5V* over  $V_{BUS}$ .
- The UFP consumes power from  $V_{BUS}$ .

## 1.3 Scope

This specification is intended as an extension to the existing [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USBType-C 1.0\]](#) and [\[BC 1.2\]](#) specifications. It addresses only the elements required to implement USB Power Delivery. It is targeted at power supply vendors, manufacturers of [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USBType-C 1.0\]](#) and [\[BC 1.2\]](#) Platforms, Devices and cable assemblies.

Normative information is provided to allow interoperability of components designed to this specification. Informative information, when provided, may illustrate possible design implementation.

## 1.4 Conventions

### 1.4.1 Precedence

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

### 1.4.2 Keywords

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

#### 1.4.2.1 Conditional Normative

**Conditional Normative** is a keyword used to indicate a feature that is mandatory when another related feature has been implemented. Designers are mandated to implement all such requirements, when the dependent features have been implemented, to ensure interoperability with other compliant Devices.

#### 1.4.2.2 **Deprecated**

**Deprecated** is a keyword used to indicate a feature, supported in previous releases of the specification, which is no longer supported.

#### 1.4.2.3 **Discarded**

**Discarded** is a keyword indicating that a Packet when received shall be thrown away by the PHY Layer and not passed to the Protocol Layer for processing. No **GoodCRC** Message shall be sent in response to the Packet.

#### 1.4.2.3.1.4.2.4 **Ignored**

**Ignored** is a keyword indicating Messages or Message fields which, when received, shall result in no action by the receiver, aside from returning a **GoodCRC** Message to acknowledge Message receipt.

#### 1.4.2.4.1.4.2.5 **May**

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

#### 1.4.2.5.1.4.2.6 **N/A**

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

#### 1.4.2.6.1.4.2.7 **Optional/Optionally/Optional Normative**

**Optional**, **Optionally** and **Optional Normative** are equivalent keywords that describe features not mandated by this specification. However, if an **Optional** feature is implemented, the feature shall be implemented as defined by this specification.

#### 1.4.2.7.1.4.2.8 **Reserved**

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

#### 1.4.2.8.1.4.2.9 **Shall/Normative**

**Shall** and **Normative** are equivalent keywords indicating a mandatory requirement. Designers are mandated to implement all such requirements to ensure interoperability with other compliant Devices.

#### 1.4.2.9.1.4.2.10 **Should**

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

### 1.4.3 Numbering

Numbers that are immediately followed by a lowercase "b" (e.g., 01b) are binary values. Numbers that are immediately followed by an uppercase "B" are byte values. Numbers that are immediately followed by a lowercase "h" (e.g., 3Ah) or are preceded by "0x" (e.g. 0xFF00) are hexadecimal values. Numbers not immediately followed by either a "b", "B", or "h" are decimal values.

## 1.5 Related Documents

- [\[USB2USB 2.0\]](#) – Universal Serial Bus Specification, Revision 2.0, plus ECN and Errata (this includes the entire document release package including the OTG&EH v2.0 and the Micro connector v1.01 specifications). [http://www.usb.org/developers/docs/usb20\\_docs/](http://www.usb.org/developers/docs/usb20_docs/).
- [\[USB3USB 3.1\]](#) – Universal Serial Bus 3.1 Specification, Revision 1 plus ECN and Errata (this includes the entire document release package including the OTG&EH v3.0 specification). [www.usb.org/developers/docs](http://www.usb.org/developers/docs).
- [\[BC1BC 1.2\]](#) – Battery Charging Specification, Revision 1.2 plus Errata (referred to in this document as the Battery Charging specification). [www.usb.org/developers/devclass\\_docs#approved](http://www.usb.org/developers/devclass_docs#approved).
- [\[USBPDCompliance\]](#) – USB Power Delivery Compliance Plan version 1.0 [http://www.usb.org/developers/powerdeliverydocs/devclass\\_docs/](http://www.usb.org/developers/powerdeliverydocs/devclass_docs/).
- [\[USBOTG2USBOTG 2.0\]](#) On-The-Go and Embedded Host Supplement to the Universal Serial Bus Revision 2.0 Specification.
- [\[Maxim37\]](#) – Maxim Engineering Journal, Volume 37, page 12 <http://pdfserv.maxim-ic.com/en/ej/EI37.pdf>.
- [\[USBType-C1C 1.0\]](#) – USB Type-C Specification [www.usb.org/developers/docs](http://www.usb.org/developers/docs)
- [\[IEC60958IEC 60958-1\]](#) IEC 60958-1 Digital Audio Interface Part:1 General Edition 3.0 2008-09 [www.iec.ch](http://www.iec.ch)

## 1.6 Terms and Abbreviations

This section defines terms used throughout this document. For additional terms that pertain to the Universal Serial Bus, see Chapter 2, “Terms and Abbreviations,” in [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USBType-C 1.0\]](#) and [\[BC 1.2\]](#).

Table 1-1 Terms and Abbreviations

Term	Description
Active Cable	A cable with a USB Plug on each end at least one of which is a Cable Plug supporting SOP', that also incorporates data bus signal conditioning circuits. The cable supports the Structured VDM <i>Discover Identity</i> to determine its characteristics in addition to other Structured VDM Commands (Electronically Marked Cable see <a href="#">[USBType-C 1.0]</a> ).
<a href="#">Active Mode</a>	<a href="#">A Mode which has been entered and not exited.</a>
Alternate Mode	As defined in <a href="#">[USBType-C 1.0]</a> . Equivalent to Mode in the PD Specification.
Alternate Mode Adapter (AMA)	A PDUSB Device which supports Alternate Modes as defined in <a href="#">[USBType-C 1.0]</a> . Note that since an AMA is a PDUSB Device it has a single UFP that is only addressable by SOP Packets.
Attached	USB Power Delivery ports which are mechanically joined with USB cable.
Binary Frequency Shift Keying (BFSK)	BFSK uses a pair of discrete frequencies to transmit binary (0s and 1s) information. In the Power Delivery BFSK system: <ul style="list-style-type: none"> <li>• Logic 0 is indicated by a frequency <math>f_{Carrier} - f_{Deviation}</math>.</li> <li>• Logic 1 is indicated by a frequency <math>f_{Carrier} + f_{Deviation}</math>.</li> </ul>
Biphase Mark Coding (BMC)	Modification of Manchester coding where each zero has one transition and a one has two transitions (see <a href="#">[IEC 60958-1]</a> ).
BIST	Built In Self Test – Power Delivery testing mechanism for the PHY Layer.
<a href="#">BIST Data Object (BDO)</a>	<a href="#">Data Object used by BIST Messages.</a>
<a href="#">BIST Mode</a>	<a href="#">A BIST receiver or transmitter test mode enabled by a BIST Message.</a>
Cable Plug	Term used to describe a PD Capable element in a Multi-Drop system addressed by SOP'/SOP'' Packets. Logically the Cable Plug is associated with a USB plug at one end of the cable. In a practical implementation the electronics may reside anywhere in the cable.
Cable Reset	This is initiated by <i>Cable Reset</i> Signaling from either the Source or DFP. It restores the Cable Plugs to their default, power up condition and resets the PD communications engine to its default state. It does not reset the Port Partners: <a href="#">but does restore VCONN to its attachment state.</a>
Cold Socket	A DFP receptacle that does not apply <i>vSafe5V</i> on $V_{BUS}$ until a plug insertion is detected.
Command	Request and response pair defined as part of a Structured Vendor Defined Message (see Section 6.4.4.2)

Field Code Changed

Term	Description
Configuration Channel (CC)	Single wire used by the BMC PHY Layer Signaling <a href="#">Scheme</a> (see <a href="#">[USBType-C 1.0]</a> ).
Connected	USB Power Delivery ports <a href="#">that have exchanged a Message and a GoodCRC which are actively communicating Message response</a> using the USB Power Delivery protocol <a href="#">so that both Port Partners know they each is PD Capable</a> .
Consumer	The capability of a PD Port (typically a Device's UFP) to sink power from the power conductor (e.g. V <sub>BUS</sub> ). This corresponds to a Type-B Port or a Type-C Port with R <sub>d</sub> asserted on its CC Wire.
Consumer/Provider	A Consumer with the additional capability to act as a Provider. This corresponds to a Dual-Role Type-B Port or a Dual-Role Type-C Port with R <sub>d</sub> asserted on its CC Wire.
<a href="#">Continuous BIST Mode</a>	<a href="#">A BIST Mode where the Port or Cable Plug being tested sends a continuous stream of test data.</a>
Contract	An agreement on both power level and direction reached between a Port Pair. A Contract may be explicitly negotiated between the Port Pair or may be an Implicit power level defined by the current state. While operating in Power Delivery mode there will always be either an Explicit or Implicit Contract in place. The Contract may only be altered in the case of a (re-)negotiation, Power Role Swap, Data Role Swap, Hard Reset or failure of the Source.
<a href="#">Control Message</a>	<a href="#">A Message is defined as a Control Message when the Number of Data Objects field in the Message Header is set to 0. The Control Message consists only of a Message Header and a CRC.</a>
<a href="#">Data Message</a>	<a href="#">A Data Message consists of a Message Header 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.</a>
<a href="#">Data Object</a>	<a href="#">32 bit object which contains information specific to different types of Data Message. Power, Request, BIST and Vendor Data Objects are defined.</a>
Data Role Swap	Process of exchanging the DFP (Host) and UFP (Device) roles between Port Partners using the <a href="#">[USBType-C 1.0]</a> connector.
Dead Battery	A device has a Dead Battery when the battery in a device is unable to power its functions.
Device	When lower cased (device), it refers to any USB product, either USB Device or USB Host. When in upper case refers to a USB Device (Peripheral or Hub).
Device Policy Manager	Module running in a Provider or Consumer that applies Local Policy to each Port in the Device via the Policy Engine.
Discovery Process	Command sequence using Structured Vendor Defined Messages resulting in identification of the Port Partner, its supported SVIDs and Modes.
Downstream Facing Port (DFP)	Typically a Type-A Port on a Device as defined in <a href="#">[USB 2.0]</a> , <a href="#">[USB 3.1]</a> or Type-C Port as defined in <a href="#">[USBType-C 1.0]</a> . The default Host and Source.
<a href="#">DRP (Type-C)</a>	<a href="#">DRP as defined in [USBType-C 1.0]. Note that this is not the same as a Power Delivery Dual-Role Port.</a>
Dual-Role Device	A product containing one or more Dual-Role Ports that are capable of operating as either a Source or a Sink.
Dual-Role Port	A Consumer/Provider or Provider/Consumer capable Port that is a Port capable of operating as either a Source or Sink.
End of Packet (EOP)	K-code marker used to delineate the end of a packet.
Enter Mode Process	Command sequence using Structured Vendor Defined Messages resulting in the Port Partners entering a Mode.
Exit Mode Process	Command sequence using Structured Vendor Defined Messages resulting in the Port Partners exiting a Mode.
Explicit Contract	An agreement reached between a Port Pair as a result of the Power Delivery negotiation process. An Explicit Contract is established (or continued) when a Source sends an <a href="#">Accept</a> Message in response to a <a href="#">Request</a> Message sent by a Sink <a href="#">followed by a PS_RDY Message indicating that the power supply is ready; this corresponds to the PE_SRC_Ready state for a Source Policy Engine and the PE_SNK_Ready state for a Sink Policy Engine</a> . The Explicit Contract may be altered through the re-negotiation process. All Port pairs, except for those involving low power devices, are required to make an Explicit Contract.



Term	Description
Frame	Generic term referring to an atomic communication transmitted by PD such as a Packet, Test Frame or Signaling.
Hard Reset	This is initiated by <i>Hard Reset</i> Signaling from either Port Partner. It restores V <sub>BUS</sub> to USB Default Operation and resets the PD communications engine to its default state in both Port Partners as well as in any attached Cable Plugs. <u>For Type-C connectors it restores both Port Partners to their default Data Roles and returns the V<sub>CONN</sub> Source to the Source Port.</u>
HDD	A Hard Disk Drive.
<u>Initiator</u> <u>ID Header VDO</u>	<u>The VDO in a Discover Identity</u> <u>The initial sender of a Command request in the form of a query. Command immediately following the VDM Header. The ID Header VDO contains information corresponding to the Power Delivery Product.</u>
Implicit Contract	An agreement on power levels between a Port Pair which occurs, not as a result of the Power Delivery negotiation process, but as a result of the current state e.g.during a Power Role Swap or Dead Battery operation, or on detection of a low power device. Implicit Contracts, except for those involving low power devices, are temporary since the Port pair is required to immediately negotiate an Explicit Contract.
<u>Initiator</u>	<u>The initial sender of a Command request in the form of a query.</u>
IR Drop	The voltage drop across the cable and connectors between the Source and the Sink. It is a function of the resistance of the ground and power wire in the cable plus the contact resistance in the connectors times the current flowing over the path.
K-code	Special symbols provided by the 4b5b coding scheme. K-codes are used to signal Hard Reset and Cable reset, and delineate Packet boundaries.
Local Policy	Every PD Capable device has its own Policy, called the Local Policy, that is executed by its Policy Engine to control its power delivery behavior. The Local Policy at any given time may be the default policy, hard coded or modified by changes in operating parameters or one provided by the system Host or some combination of these. The Local Policy optionally may be changed by a System Policy Manager.
Low Power	<u>Mode</u> <u>State</u> in which Sources powered by e.g. a single cell Li battery that want to minimize the power they output over V <sub>BUS</sub> without the requirement to fully support the PD negotiation process.
Low Power Device	Devices which can be powered e.g. by a single cell Li battery without the requirement to fully support the PD negotiation process.
<u>Modal Operation</u> <u>Message</u>	<u>The packet payload consisting of a Message Header for Control Messages and a Message Header and data for Data Messages as defined in Section 5.6.1.2.5</u> <u>State where one or more Modes are active. Modal Operation ends when there are no longer any active Modes.</u>
Message <u>Header</u>	<u>The packet payload consisting of a header for control messages and a header and data for data messages as defined in Section</u> <u>Every Message starts with a 16-bit Message Header containing basic information about the Message and the PD Port's Capabilities.</u>
Messaging	Communication in the form of Messages as defined in Chapter 5.9.9.
<u>Modal Operation</u>	<u>State where there are one or more Active Modes. Modal Operation ends when there are no longer any Active Modes.</u>
Mode	Operation defined by a Vendor or Standard's organization, which is associated with a SVID, whose definition is outside the scope of USB-IF specifications. Entry to and exit from the Mode uses the Enter Mode and Exit Mode Processes. Modes are equivalent to "Alternate Modes" as described in <u>[USBDType-C 1.0]</u> .
Multi-Drop	Refers to a Power Delivery system with one or more Cable Plugs where communication is to the Cable Plugs rather than the Port Partner. Multi-Drop systems share the Power Delivery communication channel with the Port Partners.
Negotiation	This is the PD process whereby: <ol style="list-style-type: none"> <li>1. The Source advertises its capabilities.</li> <li>2. The Sink requests one of the advertised capabilities.</li> <li>3. The Source acknowledges the request and alters its output to satisfy the request.</li> </ol> The result of the negotiation is a Contract for power delivery/consumption between the Port Pair.

Term	Description
Packet	One entire unit of PD communication including a Preamble, <i>SOP*</i> , payload, CRC and <i>EOP</i> as defined in Section 5.6.
Passive Cable	Cable with a USB Plug on each end at least one of which is a Cable Plug supporting SOP' that does not incorporate data bus signal conditioning circuits. Supports the Structured VDM <i>Discover Identity</i> to determine its characteristics (Electronically Marked Cable see [USBType-C.1.0]). Note this specification does not discuss Passive Cables which are not Electronically Marked Cables.
PD	USB Power Delivery
PD Capable	A Port that supports USB Power Delivery.
PD Connection	<del>A Port Pair that are</del> See Connected <del>i.e. actively communicating.</del>
PDUSB	USB Device Port or USB Host Port that is both PD capable and capable of USB Communication. See also PDUSB Host, PDUSB Device and PDUSB Hub.
PDUSB Device	A USB Device with a PD Capable UFP. A PDUSB Device is only addressed by SOP Packets.
PDUSB Host	A USB Host which is PD Capable on at least one of its DFPs. A PDUSB Host is only addressed by SOP Packets.
PDUSB Hub	A port expander USB Device with a UFP and one or more DFPs which is PD Capable on at least one of its Ports. A PDUSB Hub is only addressed by SOP Packets.
PDUSB Peripheral	A USB Device with a PD Capable UFP which is not a PDUSB Hub. A PDUSB Peripheral is only addressed by SOP Packets.
PHY Layer	The Physical Layer responsible for sending and receiving Messages across either V <sub>BUS</sub> or CC between a Port Pair.
Policy	Policy defines the behavior of PD capable parts of the system and defines the capabilities it advertises, requests made to (re)negotiate power and the responses made to requests received.
Policy Engine	The Policy Engine interprets the Device Policy Manager's input in order to implement Policy for a given Port and directs the Protocol Layer to send appropriate Messages.
Port	An interface typically exposed through a receptacle, or via a plug on the end of a hard-wired captive cable. USB Power Delivery defines the interaction between a Port Pair.
Port Pair	Two attached PD Capable Ports.
Port Partner	A Contract is negotiated between a Port Pair connected by a USB cable. These ports are known as Port Partners.
Power Conductor	The wire delivering power from the Source to Sink. For example USB's V <sub>BUS</sub> .
Power Consumer	See Consumer
<u>Power Data Object (PDO)</u>	<u>Data Object used to expose a Source Port's power capabilities or a Sink's power requirements as part of a Source_Capabilities or Sink_Capabilities Message respectively. Fixed, Variable and Battery Power Data Objects are defined.</u>
Power Delivery Mode	Operation after a Contract has initially been established between a Port pair. This mode persists during normal Power Delivery operation, including after a Power Role Swap. Power Delivery mode can only be exited by detaching the ports, applying a Hard Reset or by the Source removing power (except when power is removed during the Power Role Swap procedure).
Power Provider	See Provider
<u>Power Reserve</u>	<u>Power which is kept back by a Source in order to ensure that it can meet total power requirements of attached Sinks on at least one Port.</u>
<u>Power Role Swap</u>	<u>Process of exchanging the Source and Sink roles between Port Partners.</u>
Preamble	Start of a transmission which is used to enable the receiver to lock onto the carrier. The Preamble consists of a 64-bit sequence of alternating 0s and 1s starting with a "0" and ending with a "1" which is not 4b5b encoded.
<u>Product Type</u>	<u>Product categorisation returned as part of the Discover Identity Command.</u>
<u>Product Type VDO</u>	<u>VDO identifying a certain Product Type in the ID Header VDO of a Discover Identity Command.</u>

Term	Description
Protocol Error	An unexpected <del>or unknown message, that cannot be handled by a given implementation</del> <u>Message during an atomic Message sequence.</u>
Protocol Layer	The entity that forms the Messages used to communicate information between Port Partners.
Provider	A capability of a PD Port (typically a Host, Hub, or Wall Wart DFP) to source power over the power conductor (e.g. V <sub>BUS</sub> ). This corresponds to a Type-A Port or a Type-C Port with R <sub>p</sub> asserted on its CC Wire.
Provider/Consumer	A Provider with the additional capability to act as a Consumer. This corresponds to a Dual-Role Type-A Port or a Dual-Role Type-C Port with R <sub>p</sub> asserted on its CC Wire.
Re-negotiation	A process wherein one of the Port Partners wants to alter the negotiated Contract.
<u>Request Data Object (RDO)</u> <del>Power Reserve</del>	<u>Data Object used by a Sink Port to negotiate a Contact as a part of a RequestPower which is kept back by a Source in order to ensure that it can meet total power requirements of attached Sinks on at least one Port. Message.</u>
<del>Power Role Swap</del>	<del>Process of exchanging the Source and Sink roles between Port Partners.</del>
Responder	The receiver of a Command request sent by an Initiator that replies with a Command response.
Safe Operation	Sources must have the ability to tolerate <u>vSafe5V</u> applied by both Port Partners.
<del>Signalling</del> <u>Signaling</u>	A Preamble followed by an ordered set of four K-codes used to indicate a particular line symbol e.g. <u>Hard Reset</u> as defined in Section 5.4.
<u>Signaling Scheme</u>	<u>Physical mechanism used to transmit bits. BMC and BFSK Signaling Schemes are defined in this specification.</u>
Single-Role Port	A Port that is a Port only capable of operating as a Source or Sink, but not both.
Sink	The Port consuming power from V <sub>BUS</sub> ; most commonly a Device.
Soft Reset	A process that resets the PD communications engine to its default state.
<del>Start of Packet (SOP)</del>	<del>K-code marker used to delineate the start of a packet. Three start of packet sequences are defined: <u>SOP</u>, <u>SOP'</u> and <u>SOP''</u>, with <u>SOP*</u> used to refer to all three in place of <u>SOP/SOP'/SOP''</u>.</del>
SOP Communication	Communication using SOP Packets, also implies that a Message sequence is being followed.
<del>SOP' Communication</del> <u>SOP' Packet</u>	<del>Any Power Delivery Packet which starts with an <u>SOP</u> Communication with a Cable Plug using <u>SOP'</u> Packets, also implies that a message sequence is being followed.</del>
<del>SOP'' Communication</del>	<del>Communication with a Cable Plug using <u>SOP''</u> Packets, also implies that a message sequence is being followed.</del>
SOP* Communication	Communication with a Cable Plug using SOP* Packets, also implies a Message sequence is being followed.
SOP* Packet	<del>A term referring to any Power Delivery Packet which starts starting with an either <u>SOP</u>, <u>SOP'</u> or <u>SOP''</u>.</del>
<del>SOP' Communication</del>	<del>Communication with a Cable Plug using <u>SOP'</u> Packets, also implies that a Message sequence is being followed.</del>
SOP' Packet	Any Power Delivery Packet which starts with an <u>SOP'</u> used to communicate with a Cable Plug.
<del>SOP'' Communication</del>	<del>Communication with a Cable Plug using <u>SOP''</u> Packets, also implies that a Message sequence is being followed.</del>
SOP'' Packet	Any Power Delivery Packet which starts with an <u>SOP''</u> used to communicate with a Cable Plug when SOP' Packets are being used to communicate with the other Cable Plug.
<del>SOP* Packet</del>	<del>A term referring to any Power Delivery Packet starting with either <u>SOP</u>, <u>SOP'</u> or <u>SOP''</u>.</del>
Source	A role a Port is currently taking to supply power over V <sub>BUS</sub> ; most commonly a Host or Hub DFP.
Standard ID (SID)	16-bit unsigned value assigned by the USB-IF to a given industry standard.
Standard or Vendor ID (SVID)	Generic term referring to either a VID or a SID. SVID is used in place of the phrase "Standard or Vendor ID".
<del>Start of Packet (SOP)</del>	<del>K-code marker used to delineate the start of a packet. Three start of packet sequences are defined: <u>SOP</u>, <u>SOP'</u> and <u>SOP''</u>, with <u>SOP*</u> used to refer to all three in place of <u>SOP/SOP'/SOP''</u>.</del>

Term	Description
System Policy	Overall system policy generated by the system, broken up into the policies required by each Port Pair to affect the system policy. It is programmatically fed to the individual devices for consumption by their Policy Engines.
System Policy Manager	Module running on the USB Host. It applies the System Policy through communication with PD capable Consumers and Providers that are also connected to the Host via USB.
<u>Test Frame</u>	<u>Frame consisting of a Preamble, <i>SOP*</i>, followed by test data (See Section 5.9).</u>
<u>Test Pattern</u>	<u>Continuous stream of test data in a given sequence (See Section 5.9).</u>
Tester	The Tester is assumed to be a piece of test equipment that manages the BIST testing process of a PD UUT.
<u>Test Frame</u>	<u>Frame consisting of a Preamble, <i>SOP*</i>, followed by test data (See Section 5.9).</u>
<u>Test Pattern</u>	<u>Continuous stream of test data in a given sequence (See Section 5.9).</u>
Type-A	Term used to refer to any A plug or receptacle including Micro-A plugs and Standard-A plugs and receptacles, including the PD and non-PD versions. Micro-AB receptacles are assumed to be a combination of Type-A and Type-B.
Type-B	Terms used to refer to any B-plug or receptacle including Micro-B plugs and Standard-B plugs and receptacles, including the PD and non-PD versions. Micro-AB receptacles are assumed to be a combination of Type-A and Type-B.
Type-C	Term used to refer to the Type-C connector plug or receptacle as defined in <a href="#">[USBType-C 1.0]</a> .
Unit Interval (UI)	The time to transmit a single data bit on the wire.
Unit Under Test (UUT)	The PD device that is being tested by the Tester and responds to the initiation of a particular BIST test sequence.
Upstream Facing Port (UFP)	Typically a B Port on a Device as defined in <a href="#">[USB 2.0]</a> , <a href="#">[USB 3.1]</a> or Type-C Port as defined in <a href="#">[USBType-C 1.0]</a> . The default Device and Sink.
USB Attached State	Synonymous with the <a href="#">[USB 2.0]</a> and <a href="#">[USB 3.1]</a> definition of the attached state
USB Default Operation	Operation of a Port at attach or after a Hard Reset where the DFP Source applies <i>vSafe0V</i> or <i>vSafe5V</i> on $V_{BUS}$ and the UFP Sink is operating at <i>vSafe5V</i> as defined in <a href="#">[USB 2.0]</a> , <a href="#">[USB 3.1]</a> , <a href="#">[USBType-C 1.0]</a> or <a href="#">[BC 1.2]</a> .
USB Device	Either a hub or a peripheral device as defined in <a href="#">[USB 2.0]</a> and <a href="#">[USB 3.1]</a> .
USB Host	The host computer system where the USB host controller is installed as defined in <a href="#">[USB 2.0]</a> and <a href="#">[USB 3.1]</a> .
USB Powered State	Synonymous with the <a href="#">[USB 2.0]</a> and <a href="#">[USB 3.1]</a> definition of the powered state.
USB Safe State	State of the Type-C connector when there are pins to be re-purposed (see <a href="#">[USBType-C 1.0]</a> ) so they are not damaged by and do not cause damage to their Port Partner.
USB-IF PD SID (PD SID)	Standard ID allocated to this specification by the USB Implementer's Forum.
<u>VCONN Powered Accessory</u>	<u>An accessory that is powered from VCONN to operate in a Mode (see <a href="#">[USBType-C 1.0]</a>).</u>
<u>VCONN Source</u>	<u>The Type-C Port responsible for sourcing VCONN.</u>
VCONN Swap	Process of exchanging the <u>supplier of VCONN/VCONN Source</u> between Port Partners.
<u>VDM Header</u>	<u>The first Data Object following the Message Header in a Vendor Defined Message. The VDM Header contains the SVID relating to the VDM being sent and provides information relating to the Command in the case of a Structured VDM (see Section 6.4.4).</u>
<u>VVendor Data Object (VDO)</u>	<u>Data Object used to send Vendor specific information as part of a Vendor Defined Same as power (i.e. voltage * current = power) Message.</u>
Vendor Defined Message (VDM)	PD Data Message defined for vendor/standards usage. These are further partitioned into Structured VDM Messages, where Commands are defined in this specification, and Unstructured VDM Messages which are entirely Vendor Defined (see Section 6.4.4).
Vendor ID (VID)	16-bit unsigned value assigned by the USB-IF to a given Vendor.
<u>VI</u>	<u>Same as power (i.e. voltage * current = power)</u>
Wall Wart	A power supply or "power brick" that is plugged into an AC outlet. It supplies DC power to power a device or charge a battery.

## 1.7 Parameter Values

The parameters in this specification are expressed in terms of absolute values. For details of how each parameter is measured in compliance please see [\[USBPDCompliance\]](#).

## 2. Overview

This section contains no normative requirements.

### 2.1 Introduction

In USB Power Delivery, pairs of directly attached ports negotiate voltage, current and/or direction of power flow over the USB cable, using  $V_{BUS}$  or the CC wire as the communications channel. The mechanisms used, operate independently of other USB methods used to negotiate power. Type-C connectors can support the CC wire as the communications channel and in addition can support  $V_{BUS}$  communication but not concurrently. Type-A and Type-B connectors can only support  $V_{BUS}$  communication.

USB Power Delivery also acts as a side-band channel enabling support for Standard or Vendor defined Modal Operation. Modes are associated with a Standard or Vendor ID (SVID). Power Delivery Structured VDM Messages can be used to discover supported SVIDs and Modes and then to enter and exit Modes as required. Multiple [Active](#) Modes can also be in operation at the same time.

Any Contract negotiated using this specification, supersedes any and all previous power contracts established whether from standard [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USBType-C 1.0\]](#) or [\[BC 1.2\]](#) mechanisms. While in Power Delivery Mode there will be a Contract in place (either Explicit or Implicit) determining the power level available and the direction of that power. The Port Pair remains in Power Delivery Mode until the Port Pair is detached, there is a Hard Reset or the Source removes power (except during a Power Role Swap when the initial Source removes power in order to for the new Source to apply power).

An Explicit Contract is negotiated by the process of the Source sending a set of Capabilities, from which the Sink is required to request a particular capability and then the Source accepting this request.

An Implicit Contract is the specified level of power allowed in particular states (i.e. during and after a Power Role Swap, in dead battery operation or when operating with a low power device). Except for the case of low power devices, Implicit Contracts are temporary; Port Pairs are required to immediately negotiate an Explicit Contract. In the low power device case the Implicit Contract persists for as long as the Port Pair remains attached and the Source continues to supply power.

Each Provider has a Local Policy, governing power allocation to its Ports. Sinks also have their own Local Policy governing how they draw power. A System Policy can be enacted over USB that allows modification to these local policies and hence management of overall power allocation in the system.

When PD Capable devices are attached to each other, the DFPs and UFPs initially default to standard USB Default Operation. The DFP supplies *vSafe5V* and the UFP draws current in accordance with the rules defined by [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USBType-C 1.0\]](#) or [\[BC 1.2\]](#) specifications. After Power Delivery negotiation has taken place power can be supplied at higher, or lower, voltages and higher currents than defined in these specifications. It is also possible to perform a Power Role Swap to exchange the power supply roles such that the DFP receives power and the UFP supplies power. For a Type-C connector it is possible to perform a Data Role Swap such that the DFP becomes the UFP and vice-versa and to perform a  $V_{CONN}$  Swap to change the end supplying  $V_{CONN}$  to the cable.

Prior to an Explicit Contract the Source can discover the capabilities of the attached cable assembly. This is important for [\[USBType-C 1.0\]](#) where 5A cabling is marked as well as other details of the cable assembly such as the supported speed. Cable discovery occurs on initial attachment of a Port Pair, before an Explicit Contract has been established where the DFP is the Source. It is also possible to carry out cable discovery after a Power Role Swap prior to establishing an Explicit Contract, where the UFP is the Source and an Implicit Contract is in place.

Once an Explicit Contract is in place only the DFP is permitted to communicate with the attached cable assembly. This communication can consist of discovering capabilities but may also include discover of SVIDs, Modes and the entering/exiting of Modes supported by the cable assembly.

## 2.2 Section Overview

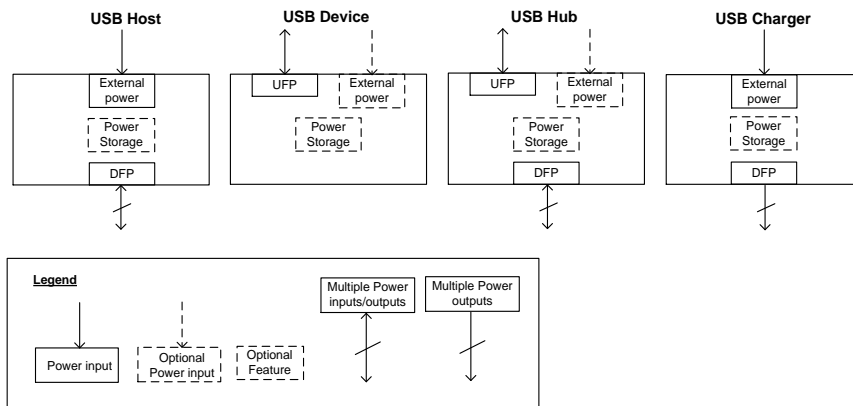
This specification contains the following sections:

Section 1	Introduction, conventions used in the document, list of terms and abbreviations, references and details of parameter usage.
Section 2	Overview of the document including a description of the operation of PD and the architecture.
Section 3	Mechanical and electrical characteristics of the cables and connectors used by PD.
Section 4	Electrical requirements for Dead Battery operation and cable detection.
Section 5	Details of the PD PHY Layer requirements
Section 6	Protocol Layer requirements including the Messages, timers, counters and state operation.
Section 7	Power supply requirements for both Providers and Consumers.
Section 8	Device Policy Manager requirements. Policy Engine Message sequence diagrams and state diagrams
Section 9	USBPD Device requirements including mapping of $V_{BUS}$ to USB states. System Policy Manager requirements including descriptors, events and requests.
Appendix A	Details of PD Power Profiles.
Appendix B	Example CRC calculations.
Appendix C	Considerations for power supply implementations.
Appendix D	Mating illustrations for the Standard-A.
Appendix E	Information relating to PHY Layer implementations.
Appendix F	Scenarios illustrating Device Policy Manager operation.
Appendix G	Examples of Structured VDM usage.

## 2.3 USB Power Delivery Capable Devices

Some examples of USB Power Delivery capable devices can be seen in Figure 2-1 (a Host, a Device, a Hub, and a Charger). These are given for reference only and do not limit the possible configurations of products that can be built using this specification.

Figure 2-1 Logical Structure of USB Power Delivery Capable Devices



Each USB Power Delivery capable device is assumed to be made up of at least one Port. Providers are assumed to have a Source and Consumers a Sink. Each device contains one, or more, of the following components:

- UFPs that:
  - Sink power (a Consumer).
  - Optionally source power (a Consumer/Provider).
  - Optionally communicate via USB.
  - Communicate using SOP Packets.
- DFPs that:
  - Source power (a Provider).
  - Optionally Sink power (a Provider/Consumer).
  - Optionally communicate via USB.
  - Communicate using SOP Packets
    - Optionally Communicate using SOP\* Packets.
- A Source that may be:
  - An external power source e.g. AC.
  - Power Storage (e.g. battery).
  - Derived from another Port (e.g. bus-powered Hub).
- A Sink that may be:
  - Power Storage (e.g. a battery).
  - Used to power internal functions.
  - Used to power devices attached to other devices (e.g. a bus-powered Hub).



## 2.4 SOP\* Communication with Cable Plugs

### 2.4.1 SOP' Communication

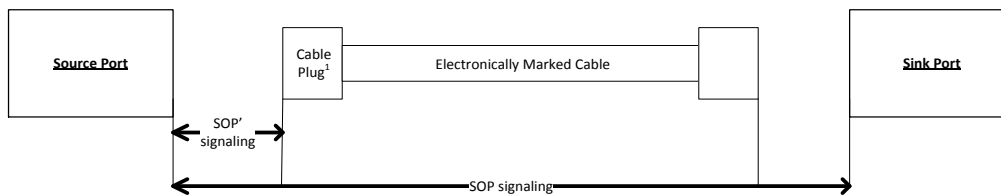
SOP' Communication is recognized by electronics in one Cable Plug (which may be attached to either the UFP or DFP). SOP Communication between the Port Partners is not recognized by the Cable Plug. Note: that the term Cable Plug in the SOP' Communication case is used to represent a logical entity in the cable which is capable of PD Communication and which may or may not be physically located in the plug. Figure 2-2 outlines the usage of SOP' Communications between a Source and a Cable Plug. Figure 2-3 outlines the usage of SOP' Communication between DFP and a Cable Plug.

Both SOP Communication and SOP' Communication take place over a single wire (either  $V_{BUS}$  or CC). This means that the SOP\* Communication periods must be coordinated to prevent important communication from being blocked. For a product which does not recognize SOP' Packets, this will look like a non-idle channel, leading to **ignored/missed** packets and retries. Communications between the Port Partners take precedence meaning that communications with the Cable Plug can be interrupted, but will not lead to error recovery steps such as Hard Reset.

When no Explicit Contract or an Implicit Contract is in place (e.g. after a Power Role Swap) the Source (either the DFP or UFP) can communicate with the Cable Plug using SOP' Packets in order to determine its characteristics (see Figure 2-2). During this phase all communication with the Cable Plug is initiated and controlled by the Source which acts to prevent conflicts between SOP\* Packets. The Sink does not communicate with the Cable Plug, even if it is the DFP, and **ignores/Discards** any SOP' Packets received.

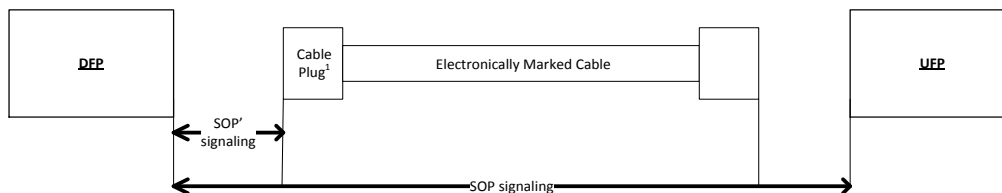
When an Explicit Contract is in place the DFP (either the Source or the Sink) can communicate with the Cable Plug using SOP' Packets (see Figure 2-3). During this phase all communication with the Cable Plug is initiated and controlled by the DFP which acts to prevent conflicts between SOP\* Packets. The UFP does not communicate with the Cable Plug, even if it is the Source and **ignores/Discards** any SOP' Packets received.

Figure 2-2 SOP' **signaling/Communication** between Source and Cable Plug with no Explicit Contract or an Implicit Contract



<sup>1</sup> Cable Plug can be physically attached to either the Source or Sink Port.

Figure 2-3 SOP' **signaling/Communication** between DFP and Cable Plug with PD Explicit Contract



<sup>1</sup> Cable Plug can be physically attached to either the DFP or UFP.

## 2.4.2 SOP'' Communication

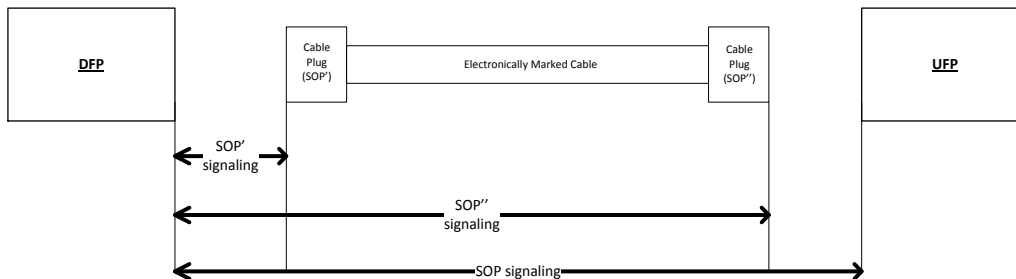
SOP' Packets are recognized by electronics in one Cable Plug attached to the DFP and are not recognized by the UFP or the other Cable Plug. Similarly SOP'' Packets are recognized by electronics in one Cable Plug attached to the UFP and are not recognized by the DFP or the other Cable Plug. Communication between the Port Partners uses SOP Packets which is not recognized by either Cable Plug. Note: that the term Cable Plug in the SOP'/SOP'' Communication case is used to represent a logical entity in the cable which is capable of PD Communication and which may or may not be physically located in the plug but is physically associated with the plug. Figure 2-4 outlines the usage of SOP'' Communication between a DFP and a Cable Plug.

SOP\* Communication takes place over a single wire (either  $V_{BUS}$  or CC). This means that the SOP\* Communication periods must be coordinated to prevent important communication being blocked. For a product which does not recognize SOP'/SOP'' Packets, this will look like a non-idle channel, leading to **ignored/missed** packets and retries. Cable Plugs are forbidden from communicating with each other and can only respond to a Command request. Communications between the Port Partners take precedence meaning that communications with the Cable Plug can be interrupted, but do not lead to error recovery steps such as Hard Reset.

When no Explicit Contract or an Implicit Contract is in place (e.g. after a Power Role Swap), no SOP'' communication takes place.

When an Explicit Contract is in place the DFP (either the Source or the Sink) can communicate with the Cable Plugs using SOP'/SOP'' Packets (see Figure 2-3). During this phase all communication with the Cable Plugs is initiated and Controlled by the DFP which acts to prevent conflicts between SOP\* Packets. The UFP does not communicate with the Cable Plug, even if it is the Source, and **ignores/Discards** any SOP'/SOP'' Packets received.

Figure 2-4 SOP'' Communication with the attached Cable Plugs



## 2.5 Operational Overview

A USB Power Delivery Port supplying power is known as a Source and a Port consuming power is known as a Sink. There is only one Source Port and one Sink Port in each PD connection between Port Partners. Where USB products support USB Power Delivery protocols a USB DFP is initially a Source and a USB UFP is initially a Sink, although USB Power Delivery also enables both the Source/Sink roles and DFP/UFP roles to be swapped; ~~DFP/DFP~~/UFP roles can only be swapped on the Type-C connector.

The following sections describe the high level operation of ports taking on the roles of DFP, UFP, Source and Sink. These sections do not describe operation that is not allowed; however if a certain behavior is not described then it is probably not supported by this specification.

For details of how PD maps to USB states in a PDUUSB Device see Section 9.1.2.

### 2.5.1 DFP Operation

The DFP operates differently depending on attachment status:

- At Attach (no PD Connection or Contract):
  - The DFP can detect attachment of an A plug or detect Sink attachment via a C-Plug.
  - The DFP typically sets  $V_{BUS}$  to *vSafe5V*.
- Before PD Connection (no PD Connection or PD Contract):
  - Prior to sending *Source\_Capabilities* Messages the DFP detects the type of cabling attached and may alter its advertised capabilities depending on the type of cable detected.
    - For an A-plug or B-plug, plug detection will be carried out to determine the current carrying capabilities of the cable.
    - For a C-plug the default capability is 3A, but SOP' Communication can be used to determine other capabilities of the cable. The DFP can attempt to communicate with one of the Cable Plugs using SOP' Packets. If the Cable Plug responds then communication takes place.
  - The DFP periodically advertises its capabilities by sending *Source\_Capabilities* Messages.  
~~○ The DFP does not generate SOP' Packets, is not required to detect SOP' Packets and will ignore them.~~
- Establishing PD Connection (no PD Connection or Contract):
  - Presence of a Port Partner is detected either:
    - By receiving a *GoodCRC* Message in response to a *Source\_Capabilities* Message.
    - By receiving *Hard Reset* Signaling.  
~~○ The DFP does not generate SOP' or SOP'' Packets, is not required to detect SOP' or SOP'' Packets and will ignore them.~~
- Establishing Explicit Contract (PD Connection but no Explicit Contract or Implicit Contract after a Power Role Swap):
  - A DFP acting as a Source receives a *Request* Message from the UFP and responds with an *Accept* Message, if this is a valid request, followed by a *PS\_RDY* Message when its power supply is ready to source power at the agreed level. At this point an Explicit Contract has been agreed.
  - A DFP acting as a Sink receives a *Source\_Capabilities* Message from the UFP and responds with a *Request* Message. If this is a valid request the DFP receives an *Accept* Message followed by a *PS\_RDY* Message when its power supply is ready to source power at the agreed level. At this point an Explicit Contract has been agreed.
  - A DFP does not generate SOP' or SOP'' Packets, is not required to detect SOP' or SOP'' Packets and ~~will ignore~~Discards them.
- During PD Connection (Explicit Contract):

- The DFP processes and responds (if a response is required) to all Messages received and sends appropriate Messages whenever its Local Policy requires.
    - The DFP can communicate with a Cable Plug using SOP' or SOP'' Communication at any time it is not engaged in any other SOP Communications.
      - If SOP Packets are received by the DFP, during SOP' or SOP'' Communication, the SOP' or SOP'' Communication is immediately terminated (the Cable Plug times out and does not retry)
      - If the DFP needs to initiate an SOP Communication during an ongoing SOP' or SOP'' Communication (e.g. for a Capabilities change) then the SOP' or SOP'' Communications will be interrupted.
    - DFP acting as a Source:
      - A DFP acting as a Source informs the UFP acting as a Sink whenever its capabilities change, by sending a *Source\_Capabilities* Message.
      - A DFP acting as a Source, after a period of *tSourceActivity*, sends out *Ping* Messages if no other Messages are being sent.
        - *Ping* Messages are optional when the Source Port is operating at *vSafe5V* and are not needed at all if a *[USBType-C 1.0]* connector is being used.
      - A DFP using the *[USBType-C 1.0]* connector and acting as a Source will always have  $R_p$  asserted on its CC wire.
    - DFP acting as a Sink (Power Role Swapped):
      - A DFP acting as a Sink whose power needs have changed indicates this to the Source with a new *Request* Message. The Sink Port may request one of the capabilities previously offered by the Source, even if this is the *vSafe5V* output offered by *[USB 2.0]*, *[USB 3.1]*, *[USBType-C 1.0]* or *[BC 1.2]*, in order to enable future power negotiation.
        - A DFP acting as a Sink not requesting any capability with a *Request* Message results in an error.
        - A DFP acting as a Sink unable to fully operate at the offered capabilities requests an offered capability but indicates a capability mismatch i.e. that it would prefer another power level also providing a physical indication of the failure to the End User (e.g. using an LED).
      - If a DFP acting as a Sink does not receive any Messages within *tSinkActivity*, it will issue *Hard Reset* Signaling.
        - A DFP acting as a Sink using the *[USBType-C 1.0]* connector does not run the *SinkActivityTimer* and therefore does not time out on Messages since it uses the detach detection mechanisms on this connector instead.
      - A DFP using the *[USBType-C 1.0]* connector and acting as a Sink will always have  $R_d$  asserted on its CC wire.
  - ~~DFP which is a Type-C DRP~~
  - A DFP using the *[USBType-C 1.0]* connector may:
    - initiate or receive a request for an exchange of data roles. After a Data Role Swap the DFP (Host) becomes a UFP (Device). The direction of power and ~~VCONN Source of VCONN~~ remain unchanged.
    - initiate or receive a request for an exchange of VCONN Source. During a VCONN Swap VCONN is applied by both ends (make before break). The direction of power and DFP/UFP roles remain unchanged.
- Detach or Communications Failure

- There are several ways to detect Detach
    - A DFP acting as a Source detects the failure to receive a *GoodCRC* Message in response to a *Ping* Message or other Message within *tReceive* leads to a Soft Reset, within *tSoftReset* of the *CRCReceiveTimer* expiring followed by a Hard Reset within *tHardReset* of the *CRCReceiveTimer* expiring to begin restoring  $V_{BUS}$  to USB Default Operation within ~50ms.
      - Receiving no response to further attempts at communication is interpreted by the DFP as a detach event (see Error handling).
      - *Ping* Messages are optional when the Source Port is operating at *vSafe5V* and are not needed at all if a *[USBType-C 1.0]* connector is being used.
    - A DFP acting as a Sink detects the removal of  $V_{BUS}$  and interprets this as the end of the PD Connection.
      - This is unless the *vSafe0V* is due to either a Hard Rest or Power Role Swap.
    - A DFP acting as a Source or Sink detects plug removal and restores  $V_{BUS}$  to USB Default Operation within ~800ms (i.e. using Type-A insertion detect or Type-C detach detection via CC).
  - Upon detach a DFP acting as either a Source or Sink returns to unattached behavior including returning its power supply to sourcing the USB default of either *vSafe0V* or *vSafe5V*.
- Error handling
    - *Minor*-Protocol Errors are handled by a *Soft\_Reset* Message issued by either Port Partner, that resets counters, timers and states, but does not change the negotiated voltage and current or the Port's role (e.g. Source or Sink) and does not cause an exit from Modal Operation.
    - Serious errors are handled by *Hard Reset* Signaling issued by either Port Partner, that resets protocol as for a Soft Reset but also returns the power supply to USB Default Operation (*vSafe0V* or *vSafe5V* output) in order to protect the Sink. A Consumer/Provider operating as a Source returns to its default operation as a Sink Port after the Hard Reset. A Hard Reset also causes all Active Modes to be exited such that the Source is no longer in Modal Operation.
      - After a Hard Reset it is expected that the Port Partner will respond within *tNoResponse*. If this does not occur then 2 further Hard Resets are carried out before the DFP goes to the *PE\_SRC\_Disabled* state at which point the PD Connection is assumed to have been terminated. *[USBType-C 1.0]* Ports can perform additional error recovery steps at this point by entering the *ErrorRecovery* state.

## 2.5.2 UFP Operation

The UFP operates differently depending on attachment status:

- $V_{BUS}$  Unpowered (no PD Connection or Contract):
  - The UFP monitors  $V_{BUS}$  for the presence of *vSafe5V*.
- $V_{BUS}$  Powered (no PD Connection or Contract):
  - The UFP detects the presence of *vSafe5V* on  $V_{BUS}$  and waits for a *Source\_Capabilities* Message indicating the presence of a PD capable Source.
  - If the UFP does not receive a *Source\_Capabilities* Message within *tSinkWaitCap/tTypeCSinkWaitCap* then it issues *Hard Reset* Signaling in order to cause the Source Port to send a *Source\_Capabilities* Message if the Source Port is PD capable.
  - The UFP does not generate SOP' or SOP'' Packets, is not required to detect SOP' or SOP'' Packets and ~~will ignoreDiscards~~ them.
- Establishing PD Connection (no PD Connection or Contract):
  - The UFP receives a *Source\_Capabilities* Message and responds with a *GoodCRC* Message.
  - The UFP does not generate SOP' or SOP'' Packets, is not required to detect SOP' or SOP'' Packets and ~~will ignoreDiscards~~ them.
- Establishing Explicit Contract (PD Connection but no Explicit Contract or Implicit Contract after a Power Role Swap):
  - UFP acting as Sink:
    - A UFP acting as a Sink receives a *Source\_Capabilities* Message from the DFP and responds with a *Request* Message. If this is a valid request the UFP receives an *Accept* Message followed by a *PS\_RDY* Message when its power supply is ready to source power at the agreed level. At this point an Explicit Contract has been agreed.
      - The Sink Port may request one of the capabilities offered by the Source, even if this is the *vSafe5V* output offered by [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USBType-C 1.0\]](#) or [\[BC 1.2\]](#), in order to enable future power negotiation.
        - For Sinks using an A-plug or B-plug the request will be limited by the type of cabling detected. Plug detection will be carried out to determine the current carrying capabilities of the cable.
        - For Sinks using a C-plug the request is made from any capabilities offered by the Source.
        - A Sink not requesting any capability with a *Request* Message results in an error.
      - A Sink unable to fully operate at the offered capabilities requests the default capability but indicates that it would prefer another power level and provide a physical indication of the failure to the end user (e.g. using an LED).
    - A UFP does not generate SOP' or SOP'' Packets, is not required to detect SOP' or SOP'' Packets and ~~will ignoreDiscards~~ them.
  - UFP acting as Source: ~~(Power Role Swapped)~~:
    - Prior to sending *Source\_Capabilities* Messages a UFP acting as a Source detects the type of cabling attached and may alter its advertised capabilities depending on the type of cable detected.
      - For an A-plug or B-plug, plug detection will be carried out to determine the current carrying capabilities of the cable.



- initiate or receive a request for an exchange of VCONN Source. During a VCONN Swap VCONN is applied by both ends (make before break). The direction of power and DFP/UFP roles remain unchanged.

- Detach or Communications Failure

- There are several ways to detect Detach
  - A UFP acting as a Sink detects the removal of  $V_{BUS}$  and interprets this as the end of the PD Connection.
    - This is unless the *vSafe0V* is due to either a Hard Rest or Power Role Swap.
  - A UFP acting as a Source detects the failure to receive a *GoodCRC* Message in response to a *Ping* Message or other Message within *tReceive*. This leads to a Soft Reset, within *tSoftReset* of the *CRCReceiveTimer* expiring followed by a Hard Reset within *tHardReset* of the *CRCReceiveTimer* expiring to begin restoring  $V_{BUS}$  to USB Default Operation within ~50ms.
    - Receiving no response to further attempts at communication is interpreted by the UFP as a detach event (see Error handling).
    - *Ping* Messages are optional when the Source Port is operating at *vSafe5V* and are not needed at all if a *[USBType-C.1.0]* connector is being used.
  - A DFP acting as a Source or Sink detects plug removal and restores  $V_{BUS}$  to USB Default Operation within ~800ms (i.e. using Type-A insertion detect or Type-C detach detection via CC).
- Upon detach a UFP acting as either a Sink or Source stops supplying power and returns to its default behavior as an unattached Sink Port.

- Error handling

- ~~Minor~~ Protocol Errors are handled by a *Soft\_Reset* Message issued by either Port Partner, that resets counters, timers and states, but does not change the negotiated voltage and current or the Port's role (e.g. Source or Sink) and does not cause an exit from Modal Operation.
- Serious errors are handled by issuing *Hard\_Reset* Signaling that resets protocol as for a Soft Reset but also returns the power supply to its USB default current and voltage. A Provider/Consumer operating as a Sink returns to its USB Default Operation as a Source Port outputting *vSafe0V* or *vSafe5V* after the Hard Reset. A Hard Reset also causes all Active Modes to be exited such that the Sink is no longer in Modal Operation.
  - After a Hard Reset it is expected that the Port Partner will respond within *tNoResponse*. If this does not occur then 2 further Hard Resets are carried out before the UFP stays in the *PE\_SNK\_Wait\_for\_Capabilities* state at which point the PD Connection is assumed to have been terminated. *[USBType-C.1.0]* Ports can perform additional error recovery steps at this point.



### 2.5.3 Cable Plug Operation

The DFP operates differently depending on attachment status:

- Before PD Connection (no Contract):
  - The DFP operating as a Source can attempt to communicate with one of the Cable Plugs using SOP' Packets
    - If the Cable Plug responds then communication takes place.
- Establishing PD Connection (no Contract):
  - The DFP does not generate SOP' Packets, is not required to detect SOP' Packets and ~~will ignore Discards~~ them.
- Establishing Explicit Contract (PD Connection but no Explicit Contract or Implicit Contract after a Power Role Swap):
  - The DFP or UFP operating as a Source can attempt to communicate with one of the Cable Plugs using SOP' Packets
    - If the Cable Plug responds then communication takes place.
- During PD Connection (Explicit Contract):
  - The DFP can communicate with a Cable Plug, using SOP' Packets or SOP'' Packets, at any time it is not engaged in any other SOP Communications.
  - If SOP Packets are received by the DFP, during SOP' or SOP'' Communication, the SOP' or SOP'' Communication is immediately terminated (the Cable Plug times out and does not retry)
  - If the DFP needs to initiate an SOP Communication during an ongoing SOP' or SOP'' Communication (e.g. for a Capabilities change) then the SOP' or SOP'' Communications will be interrupted.
- Detach or Communications Failure:
  - There is no communication timeout scheme between the DFP and Cable Plug since communications may be interrupted at any time.
- Error handling:
  - The Cable Plug can detect **Hard Reset** Signaling to determine that the DFP and UFP have been reset and will need to reset itself (equivalent to a power cycle).
    - The Cable Plug cannot generate **Hard Reset** Signaling itself.
  - A Cable Plug can detect **Cable Reset** Signaling to determine that it will need to reset itself (equivalent to a power cycle).

## 2.6 Architectural Overview

*This logical architecture is not intended to be taken as an implementation architecture. An implementation architecture is, by definition, a part of product definition and is therefore outside of the scope of this specification.*

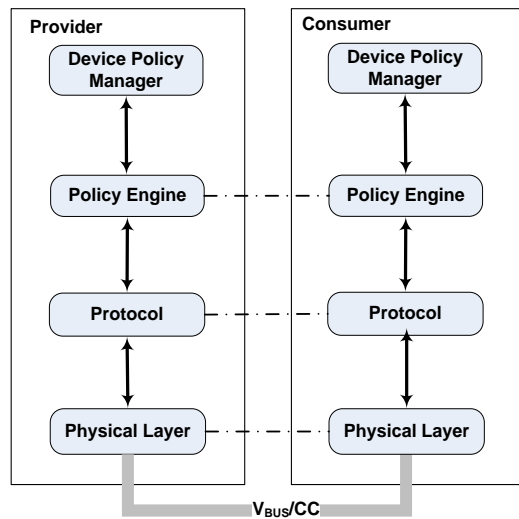
This section outlines the high level logical architecture of USB Power Delivery referenced throughout this specification. In practice various implementation options are possible based on many different possible types of PD device. PD devices may have many different configurations e.g. USB or non-USB communication, single versus multiple ports, dedicated power supplies versus supplies shared on multiple ports, hardware versus software based implementations etc. The architecture outlined in this section is therefore provided only for reference in order to indicate the high level **logical** model used by the PD specification. This architecture is used to identify the key concepts and also to indicate logical blocks and possible links between them.

The USB Power Delivery architecture in each USB Power Delivery capable Device is made up of a number of major components.

The communications stack seen in Figure 2-5 consists of:

- A **Device Policy Manager** (see Section 8.2) that exists in all devices and manages USB Power Delivery resources within the device across one or more ports based on the Device's Local Policy.
- A **Policy Engine** (see Section 8.3) that exists in each USB Power Delivery Port implements the Local Policy for that Port.
- A **Protocol Layer** (see Chapter 5.9.9) that enables Messages to be exchanged between a Source Port and a Sink Port.
- A **Physical Layer** (see Chapter 5) that handles transmission and reception of bits on the wire and handles data transmission.

Figure 2-5 USB Power Delivery Communications Stack



Additionally USB Power Delivery devices which can operate as USB devices may communicate over USB (see Figure 2-6). An optional **System Policy Manager** (see Chapter 9) that resides in the USB Host communicates with the PD Device over USB, via the root Port and potentially over a tree of USB Hubs. The **Device Policy Manager** interacts with the USB interface in each device in order to provide and update PD related information in the USB domain. Note that a PD device is not required to have a USB device interface.

Figure 2-6 USB Power Delivery Communication Over USB

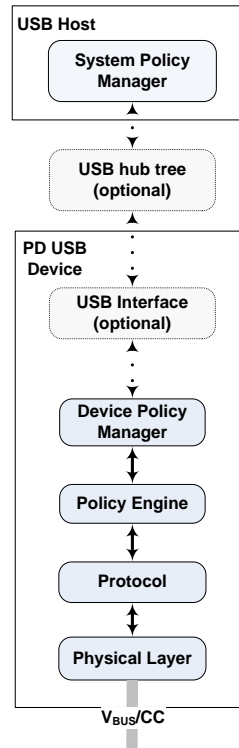
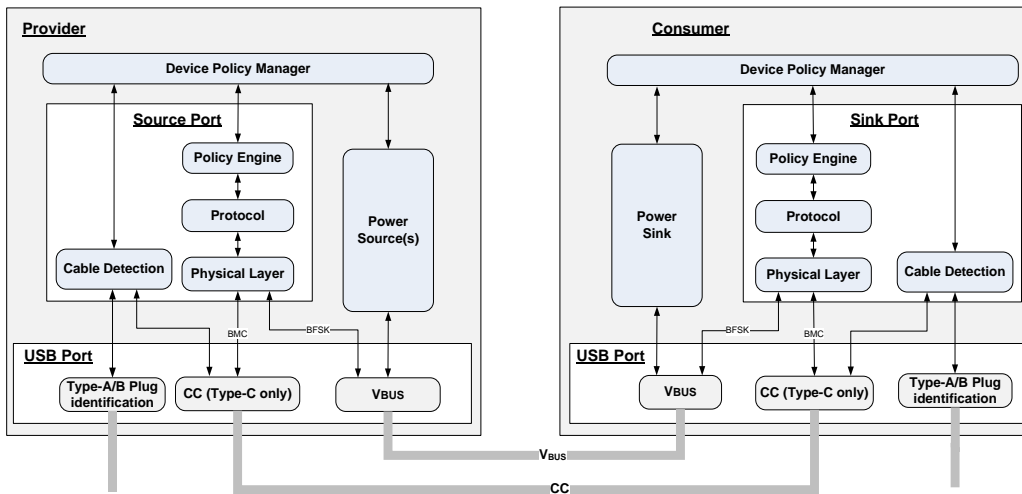


Figure 2-7 shows the logical blocks between two attached PD ports. In addition to the communication stack described above there are also:

- For a Provider or Dual-Role Device: one or more **Sources** providing power to one or more ports.
- For a Consumer or Dual-Role Device: a **Sink** consuming power.
- A **Cable Detection** module (see Section 4.4) that:
  - Detects presence of  $V_{BUS}$  for Sink Ports
  - Identifies the type of PD cable attached
- ~~For~~ **USB Power Delivery uses either** Type-A and Type-B **PD Cabling** is defined in Section 3 ~~that carries both power and USB Power Delivery signaling over  $V_{BUS}$ . Type-C uses or~~ standard cabling defined in [\[USB 2.0\]](#), [\[USB 3.1\]](#), and [\[USBType-C 1.0\]](#).
- ~~The~~ **Device Policy Manager** talks to the communication stack, Source/Sink and the cable detection block in order to manage the resources in the Provider or Consumer.

Figure 2-7 illustrates a Provider and a Consumer. Dual-Role Devices such as Provider/Consumers or Consumer/Providers can be constructed by combining the elements of both Provider and Consumer into a single device. Providers can also contain multiple Source Ports each with their own communications stack and cable detection.

Figure 2-7 High Level Architecture View



### 2.6.1 Policy

There are two possible levels of Policy:

- 1) System Policy applied system wide by the System Policy Manager across multiple Providers or Consumers.
- 2) Local Policy enforced on a Provider or Consumer by the Device Policy Manager.

Policy comprises several logical blocks:

- System Policy Manager (system wide).
- Device Policy Manager (one per Provider or Consumer).
- Policy Engine (one per Source or Sink Port).

#### 2.6.1.1 System Policy Manager

Since the USB Power Delivery protocol is essentially point to point, implementation of a System Policy requires communication by an additional data communication mechanism i.e. USB. The System Policy Manager monitors and controls System Policy between various Providers and Consumers connected via USB. The System Policy Manager resides in the USB Host and communicates via USB with the Device Policy Manager in each connected Device. Devices without USB data communication capability or are not data connected, will not be able to participate in System Policy.

The System Policy Manager is optional in any given system so USB Power Delivery Providers and Consumers can operate without it being present. This includes systems where the USB Host does not provide a System Policy Manager and may also include "headless" systems without any USB Host. In those cases where a Host is not present, USB Power Delivery is useful for charging purposes, or the powering of devices since useful USB functionality is not possible. Where there is a USB Host but no System Policy Manager, Providers and Consumers can negotiate power between themselves, independently of USB power rules, but are more limited in terms of the options available for managing power.

### 2.6.1.2 Device Policy Manager

The Device Policy Manager provides mechanisms to monitor and control the USB Power Delivery system within a particular Consumer or Provider. The Device Policy Manager enables Local Policies to be enforced across the system by communication with the System Policy Manager. Local Policies are enacted on a per Port basis by the Device Policy Manager's control of the Source/Sink Ports and by communication with the Policy Engine and Cable Detection for that Port.

### 2.6.1.3 Policy Engine

Providers and Consumers are free to implement their own Local Policies on their directly connected Source or Sink Ports. These will be supported by negotiation and status mechanisms implemented by the Policy Engine for that Port. The Policy Engine interacts directly with the Device Policy Manager in order to determine the present Local Policy to be enforced. The Policy Engine will also be informed by the Device Policy Manager whenever there is a change in Local Policy (e.g. a capabilities change).

## 2.6.2 Message Formation and Transmission

### 2.6.2.1 Protocol Layer

The Protocol Layer forms the Messages used to communicate information between a pair of ports. It is responsible for forming Capabilities Messages, requests and acknowledgements. Additionally it forms Messages used to swap roles and maintain presence. It receives inputs from the Policy Engine indicating which Messages to send and indicates the responses back to the Policy Engine.

The basic protocol uses a push model where the Provider pushes its capabilities to the Consumer that in turn responds with a request based on the offering. However, the Consumer may asynchronously request the Provider's present capabilities and may select another voltage/current.

### 2.6.2.2 PHY Layer

The PHY Layer is responsible for sending and receiving Messages across either the  $V_{BUS}$  or CC wire. It consists of a transceiver that superimposes a ~~signal~~ **Signaling Scheme** (BFSK on  $V_{BUS}$  or BMC on CC) on the wire. The PHY Layer is responsible for managing data on the wire. It tries to avoid collisions on the wire, recovering from them when they occur. It also detects errors in the Messages using a CRC.

## 2.6.3 Power supply

### 2.6.3.1 Source

Each Provider will contain one or more Sources that are shared between one or more ports. These Sources are controlled by the Local Policy. Sources start up in USB Default Operation where the Port applies **vSafe0V** or **vSafe5V** on  $V_{BUS}$  and return to this state on detach or after a Hard Reset. If the Source applies **vSafe0V** as their default, it detects attach events and transitions its output to **vSafe5V** upon detecting an attach.

### 2.6.3.2 Sink

Consumers are assumed to have one Sink connected to a Port. This Sink is controlled by Local Policy. Sinks start up in USB Default Operation where the Port can operate at **vSafe5V** with USB default specified current levels and return to this state on detach or after a Hard Reset.

### 2.6.3.3 Dual-Role Ports

Dual-Role ports ~~are found in~~ **can be either a Provider/Consumer or Consumer/Provider meaning that they have the ability to operate as either a Source or a Sink.** A Provider/Consumer ~~will have one or more ports~~ **is a Port** that can operate as either a Source Port (default) or a Sink Port. A Consumer/Provider ~~will have~~ **is** a Port that can operate either as a Sink Port (default) or a Source Port.

Combination supplies that support Dual-Role ports follow the default rules as well as supporting a return to their own default role and state on a detach event or Hard Reset.

#### 2.6.3.4 Dead Battery or Lost Power Detection

The USB Power Delivery specification defines mechanisms intended to communicate with and charge a Provider/Consumer with a Dead Battery. A Provider/Consumer may also use this mechanism to get power when they are disconnected from their normal power source. All Consumer/Providers support this mechanism.

### 2.6.4 Cable and Connectors

#### 2.6.4.1 Cable Detection

The Cable Detection block provides mechanisms to detect the type of cable attached. This information is provided to the Device Policy Manager. It adjusts the Local Policy accordingly and may also communicate cabling issues to the System Policy Manager for further action.

The USB Power Delivery specification assumes certified USB cables as defined in this specification or in the [\[USB 2.0\]](#), [\[USB 3.1\]](#), or [\[USBType-C 1.0\]](#) specifications.

For Type-A and Type-B connectors the existence of a large number of non-compliant legacy cables, particularly 'Y' and 'W' cables are problematic. These kinds of cables, in combination with the higher voltages that PD can deliver, have the potential to permanently damage the user's equipment. PD defines mechanisms to detect Type-A and Type-B PD capable cables.

For Type-C connectors, PD uses the certified USB cables and associated detection mechanisms as defined in [\[USBType-C 1.0\]](#).

#### 2.6.4.2 Interactions between Non-PD, BC and PD devices

USB Power Delivery only operates when two USB Power Delivery devices are directly connected. When a Device finds itself a mixed environment, where the other device does not support the USB Power Delivery Specification, the existing rules on supplying *vSafe5V* as defined in the [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[BC 1.2\]](#) or [\[USBType-C 1.0\]](#) specifications are applied.

There are two primary cases to consider:

- The Host (DFP) is non-PD and as such will not send any advertisements. An attached PD capable Device will not see any advertisements and operates using the rules defined in the [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[BC 1.2\]](#) or [\[USBType-C 1.0\]](#) specifications.
- The Device (UFP) is non-PD and as such will not see any advertisements and therefore will not respond. The Host (DFP) will continue to supply *vSafe5V* to  $V_{BUS}$  as specified in the [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[BC 1.2\]](#) or [\[USBType-C 1.0\]](#) specifications.

#### 2.6.4.3 Power Profiles

Power Profiles are used to define voltages and current ranges that may be offered by USB Power Delivery Sources. Although Profiles are defined for Sources, Sinks may refer to a Profile that will meet their power requirements. See Appendix A for further details.

### 3. Type-A and Type-B Cable Assemblies and Connectors

USB Power Delivery is designed to operate at voltages and currents outside the specified ranges defined in [\[USB 2.0\]](#), [\[USB 3.1\]](#) and [\[BC 1.2\]](#) specifications. For this reason it is critical to understand the characteristics of the USB Power Delivery cable assemblies and the connectors used. The following sections define the assumed characteristics of both non-PD Type-A and Type-B cable assemblies (non-marked cable assemblies) and USB Power Delivery cable assemblies designed to take advantage of the USB Power Delivery system (marked cable assemblies). Cable assemblies for use with Type-C connectors are defined in [\[USBType-C 1.0\]](#) and are not covered in this specification.

The considerable presence of non-compliant legacy cable assemblies presents an additional challenge to safely deliver voltages other than *vSafe5V*. 'Y' cable assemblies, for example, provide topologies and connections that are not included in the original USB specifications. As a result, there is a requirement to identify cable assemblies used in support of this specification that deliver more than *vSafe5V*.

The additional requirements for USB Power Delivery cable assemblies are defined in this section. Unlike typical USB philosophy, both the Device and the Host are responsible for detecting the insertion of a USB Power Delivery cable assembly and make requests for power accordingly.

#### 3.1 Significant Features

This section identifies the significant features of the USB Power Delivery connectors and cable assemblies. This section references other parts of the document where further details can be found.

##### 3.1.1 Connectors

The USB PD specification defines the following connectors:

- PD Standard-A plug and receptacle
- PD Standard-B plug and receptacle
- PD Micro-A plug
- PD Micro-AB receptacle
- PD Micro-B plug and receptacle

USB PD connectors are differentiated from their standard USB counterparts in their current carrying capability as well as electrical markings or physical features, while remaining mechanically compatible. The USB PD Micro connectors are mechanically identical to their non-PD counterparts. The USB PD Standard connector mechanical differences are specified in this chapter.

- PD Micro-A, PD Micro-AB, and PD Micro-B connectors shall be capable of carrying 3A current as specified in Section 3.6.5.1.
- PD Standard-A and PD Standard-B connectors shall be capable of carrying 5A current as specified in Section 3.6.5.2.

Refer to Section 3.4 for marking details. For details on the usage of the PD icon refer to Section 3.5.

Appendix D illustrates mating conditions of Standard-A connector combinations.

Table 3-1 lists the compatible plugs and receptacles and possible roles which may be supported in each case. Note that for AB receptacles it is not required that if the Provider/Consumer role is supported with an A-plug inserted, that the Consumer/Provider role is supported with a B-plug inserted or vice-versa. Standard, non-PD, receptacles used in USB Power Delivery capable products are limited to a nominal *vSafe5V* at 1.5A as defined in [\[BC 1.2\]](#).

Field Code Changed

Field Code Changed

Table 3-1 Plugs Accepted By Receptacles

Receptacle	Plugs Accepted	Possible Role
USB 2.0 Standard-A	USB 2.0 Standard-A, USB 2.0 PD Standard-A, USB 3.1 Standard-A, or USB 3.1 PD Standard-A	Provider, Provider/Consumer
USB 2.0 PD Standard-A	USB 2.0 PD Standard-A, USB 2.0 Standard-A, USB 3.1 PD Standard-A, or USB 3.1 Standard-A	Provider, Provider/Consumer
USB 3.1 Standard-A	USB 3.1 Standard-A, USB 3.1 PD Standard-A, USB 2.0 Standard-A, or USB 2.0 PD Standard-A	Provider, Provider/Consumer
USB 3.1 PD Standard-A	USB 3.1 PD Standard-A, USB 3.1 Standard-A, USB 2.0 PD Standard-A, or USB 2.0 Standard-A	Provider, Provider/Consumer
USB 2.0 Standard-B	USB 2.0 Standard-B or USB 2.0 PD Standard-B	Consumer, Consumer/Provider
USB 2.0 PD Standard-B	USB 2.0 PD Standard-B or USB 2.0 Standard-B	Consumer, Consumer/Provider
USB 3.1 Standard-B	USB 3.1 Standard-B, USB 3.1 PD Standard-B, USB 2.0 Standard-B, USB 2.0 PD Standard-B or USB 3.1 Powered-B	Consumer, Consumer/Provider
USB 3.1 PD Standard-B	USB 3.1 PD Standard-B, USB 3.1 Standard-B, USB 2.0 PD Standard-B, USB 2.0 Standard-B, or USB 3.1 Powered-B	Consumer, Consumer/Provider
USB 3.1 Powered-B	USB 3.1 Powered-B, USB 3.1 Standard-B, USB 3.1 PD Standard-B, USB 2.0 Standard-B, or USB 2.0 PD Standard-B	Consumer, Consumer/Provider
USB 2.0 Micro-B	USB 2.0 Micro-B or USB 2.0 PD Micro-B	Consumer, Consumer/Provider
USB 2.0 PD Micro-B	USB 2.0 PD Micro-B or USB 2.0 Micro-B	Consumer, Consumer/Provider
USB 3.1 Micro-B	USB 3.1 Micro-B, USB 3.1 PD Micro-B, USB 2.0 Micro-B, or USB 2.0 PD Micro-B	Consumer, Consumer/Provider
USB 3.1 PD Micro-B	USB 3.1 PD Micro-B, USB 3.1 Micro-B, USB 2.0 PD Micro-B, or USB 2.0 Micro-B	Consumer, Consumer/Provider
USB 2.0 Micro-AB	USB 2.0 Micro-A, or USB 2.0 PD Micro-A	Provider, Provider/Consumer,
	USB 2.0 Micro-B, USB 2.0 PD Micro-B	Consumer, Consumer/Provider
USB 2.0 PD Micro-AB	USB 2.0 PD Micro-A, or USB 2.0 Micro-A	Provider, Provider/Consumer
	USB 2.0 PD Micro-B, USB 2.0 Micro-B,	Consumer, Consumer/Provider
USB 3.1 Micro-AB	USB 2.0 Micro-A, or USB 2.0 PD Micro-A, USB 3.1 Micro-A, USB 3.1 PD Micro-A	Provider, Provider/Consumer
	USB 2.0 Micro-B, USB 2.0 PD Micro-B, USB 3.1 Micro-B, USB 3.1 PD Micro-B	Consumer, Consumer/Provider
USB 3.1 PD Micro-AB	USB 2.0 PD Micro-A, or USB 2.0 Micro-A USB 3.1 PD Micro-A, USB 3.1 Micro-A	Provider, Provider/Consumer
	USB 2.0 PD Micro-B, USB 2.0 Micro-B, USB 3.1 PD Micro-B, USB 3.1 Micro-B	Consumer, Consumer/Provider

### 3.1.1.1 USB 2.0 PD Standard-A Connector

The USB 2.0 PD Standard-A connector is defined as the host connector. It has the same mating interface as the USB 2.0 Standard-A connector, but with mechanical differences to provide a means of detecting insertion and PD support. Refer to Section 3.2.3 for mechanical details, pin assignments, and descriptions.



A USB 2.0 PD Standard-A receptacle accepts a USB 2.0 PD Standard-A plug, a USB 2.0 Standard-A plug, a USB 3.1 PD Standard-A plug, or a USB 3.1 Standard-A plug. Similarly, a USB 2.0 PD Standard-A plug may be mated with a USB 2.0 PD Standard-A receptacle, a USB 2.0 Standard-A receptacle, a USB 3.1 PD Standard-A receptacle, or a USB 3.1 Standard-A receptacle.

#### 3.1.1.2 USB 3.1 PD Standard-A Connector

The USB 3.1 PD Standard-A connector is defined as a host connector. It has the same mating interface as the USB 3.1 Standard-A connector, but with mechanical differences to provide a means of detecting insertion and PD support. Refer to Section 3.2.4 for mechanical details, pin assignments, and descriptions.

A USB 3.1 PD Standard-A receptacle accepts a USB 3.1 PD Standard-A plug, a USB 3.1 Standard-A plug, a USB 2.0 PD Standard-A plug, or a USB 2.0 Standard-A plug. Similarly, a USB 3.1 PD Standard-A plug may be mated with a USB 3.1 PD Standard-A receptacle, a USB 3.1 Standard-A receptacle, a USB 2.0 PD Standard-A receptacle, or a USB 2.0 Standard-A receptacle.

#### 3.1.1.3 USB 2.0 PD Standard-B Connector

The USB 2.0 PD Standard-B connector is defined to facilitate delivery of up to 100 Watts. An ID pin supports PD identification. Refer to Section 3.2.5 for mechanical details, pin assignments, and descriptions.

A USB 2.0 PD Standard-B receptacle accepts either a USB 2.0 PD Standard-B plug or a USB 2.0 Standard-B plug. Inserting a USB 3.1 PD Standard-B plug, a USB 3.1 Standard-B plug, or a USB 3.1 Powered-B plug into a USB 2.0 PD Standard-B receptacle is physically disallowed. A USB 2.0 PD Standard-B plug may be mated with a USB 2.0 PD Standard-B receptacle, a USB 2.0 Standard-B receptacle, a USB 3.1 PD Standard-B receptacle, a USB 2.0 Standard-B receptacle, or a USB 3.1 Powered-B receptacle.

#### 3.1.1.4 USB 3.1 PD Standard-B Connector

The USB 3.1 PD Standard-B connector is defined to facilitate delivery of up to 100 Watts. An ID pin supports PD identification. Refer to Section 3.2.6 for mechanical details, pin assignments, and descriptions.

A USB 3.1 PD Standard-B receptacle accepts a USB 3.1 PD Standard-B plug, a USB 3.1 Standard-B plug, a USB 3.1 Powered-B plug, a USB 2.0 PD Standard-B plug, or a USB 2.0 Standard-B plug. Inserting a USB 3.1 PD Standard-B plug or a USB 3.1 Standard-B plug into a USB 2.0 PD Standard-B receptacle or a USB 2.0 Standard-B receptacle is physically disallowed. A USB 3.1 PD Standard-B plug may be mated with a USB 3.1 PD Standard-B receptacle, a USB 3.1 Standard-B receptacle, or a USB 3.1 Powered-B receptacle.

### 3.1.2 Compliant Cable Assemblies

The USB Power Delivery Specification defines the following cable assemblies:

- USB 3.1 PD Standard-A plug to USB 3.1 PD Standard-B plug
- USB 3.1 PD Standard-A plug to USB 3.1 PD Micro-B plug
- USB 3.1 PD Micro-A plug to USB 3.1 PD Micro-B plug
- USB 3.1 PD Micro-A plug to USB 3.1 PD Standard-B plug
- USB 2.0 PD Standard-A plug to USB 2.0 PD Standard-B plug
- USB 2.0 PD Standard-A plug to USB 2.0 PD Micro-B plug
- USB 2.0 PD Micro-A plug to USB 2.0 PD Micro-B plug
- USB 2.0 PD Micro-A plug to USB 2.0 PD Standard-B plug

PD cable assemblies shall have one plug on each end. PD Cable assemblies with multiple connectors on either end are expressly prohibited due to the danger of back-powering USB ports with high voltages. Any cable combinations not explicitly defined in the list above shall not be permitted.

Connectors on each end of the PD cable assembly shall be labeled with the appropriate USB PD icon defined in Section 3.5.

### 3.1.3 USB Power Delivery Adapters (USB plug to USB receptacle)

This section has been deleted. Adapters from a USB plug to a USB receptacle are not supported by PD because their use may result in unacceptable  $V_{GND\_DROP}$ , restricting USB 2.0 signaling capability.

### 3.1.4 Hardwired Captive PD Cable

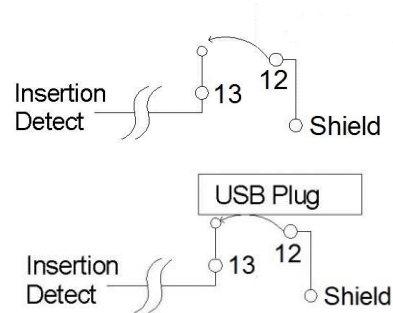
A hardwired captive PD cable assembly has a single PD plug at one end and a single hardwired (non-removable) connection at the other end. A hardwired captive PD cable assembly is PD compliant if it meets the requirements of this specification at the connector end cable marking or marking detection, negotiation capability, etc. and supports USB PD. A hardwired captive PD cable assembly may support USB signaling. The initial role of a Port with a hardwired captive cable assembly may be either a Provider or a Consumer as determined by the type of plug.

The connector on the hardwired captive PD cable assembly shall be labeled with the appropriate USB PD icon defined in Section 3.5.

### 3.1.5 Standard-A Insertion Detect

Insertion Detect is a feature added to the Standard-A receptacle to support Cold Socket capability. It may be implemented in a Standard-A receptacle or a PD Standard-A receptacle. Insertion Detect provides an open circuit condition on pin 13 when a plug is not detected in the Standard-A receptacle and a connection to the receptacle shield on pin 13 when a plug is present. Schematic representation is provided in Figure 3-1 to illustrate the electrical implementation of the detection mechanism if pin 12 is present. Implementation is vendor-specific. The Insertion Detect feature shall be implemented for Cold Socket Standard-A applications and optional for all other Standard-A implementations. See Section 3.2.1 for the mechanical requirements, and Sections 3.6.1, 3.6.2 and 4.1.2 for the electrical description.

Figure 3-1 Standard-A Insertion Detect Schematic Representation



### 3.1.6 Standard-A PD Detect

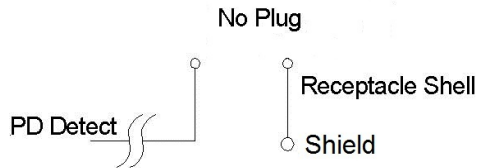
The PD Detect contact is a feature added to the PD Standard-A receptacle to detect the presence of a PD Standard-A plug. Implementation of PD Detect shall include either pin 10, pin 11, or both pin 10 and pin 11 in the PD Standard-A receptacle. If both pins 10 and 11 are present then both shall be connected to the PD Detect logic. Depending on the detection mechanism and location, two pins may be required to ensure PD Detect if the plug is skewed within the receptacle. If two pins are used for PD Detect, then a closed circuit to the shield on either pin is considered PD detected.

PD Detect shall be an open circuit when PD is not detected (refer to Figure 3-2, Figure 3-3 and Figure 3-4) and shall be connected to the receptacle shield when a PD Standard-A plug is present (Figure 3-5). PD Detect shall be a closed circuit to the shield when the PD plug is detected.

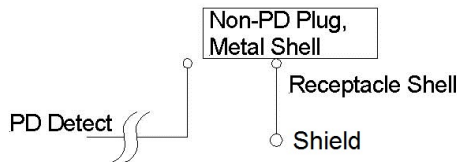
The PD Standard-A receptacle shall not connect PD Detect to the receptacle shield when a USB Thin Card is inserted, as shown in Figure 3-4.

Schematic representation is provided (Figure 3-5) to illustrate the electrical implementation of the detection mechanism when the implementation uses the PD Standard-A plug shield as a conductor to complete the detection circuit. The mechanics of the implementation are vendor-specific.

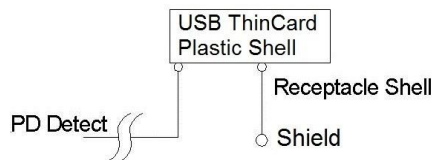
**Figure 3-2 PD Standard-A No Plug Detection Circuit**



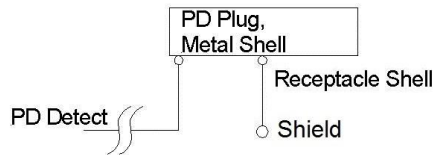
**Figure 3-3 Non-PD Plug Standard-A Detection Circuit**



**Figure 3-4 USB Thin Card Standard-A Detection Circuit**



**Figure 3-5 PD Plug Standard-A Detection Circuit**



See Appendix D for mechanical details of different connector mating situations. See Section 3.6.1 and 3.6.2 for the electrical performance requirements.

### 3.1.7 Raw Cables

Attention should be given for the possibility of signal interference from  $V_{BUS}$ , especially at higher operating current and voltage, in PD cable assemblies that are capable of Hi-Speed or Enhanced SuperSpeed data communication. See Section 3.6.6 crosstalk requirements.

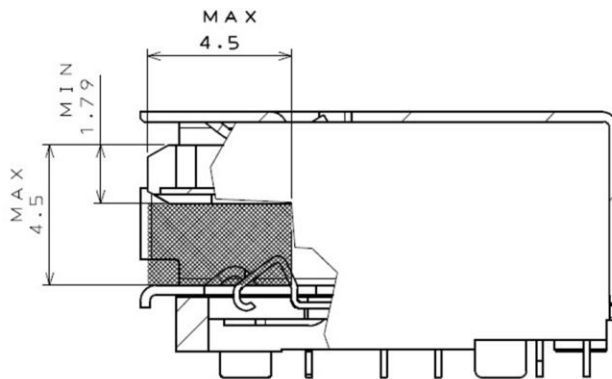
## 3.2 Connector Mating Interfaces

This section defines the connector mating interfaces, including the connector interface drawings, pin assignments and descriptions. All dimensions are in mm.

### 3.2.1 Standard-A Insertion Detect Mechanical Dimensions

The Insertion Detect feature is optional if Cold Socket is not implemented. Figure 3-6 shows the mechanical dimensions for the zone where the Insertion Detect mechanism on the Standard-A and PD Standard-A receptacle shall activate when any type of Standard-A plug is inserted. This zone applies to any USB 2.0 or USB 3.1 versions of receptacles. The detection mechanism shall be designed such that the insertion detect electrical requirements are met, regardless of power being applied. Special consideration should be given to the location of shell features such as retention springs and cutout areas. The sample implementation shown in Figure 3-6 is for reference only.

Figure 3-6 Insertion Detect Zone Mechanical Dimensions for the Standard-A Receptacle

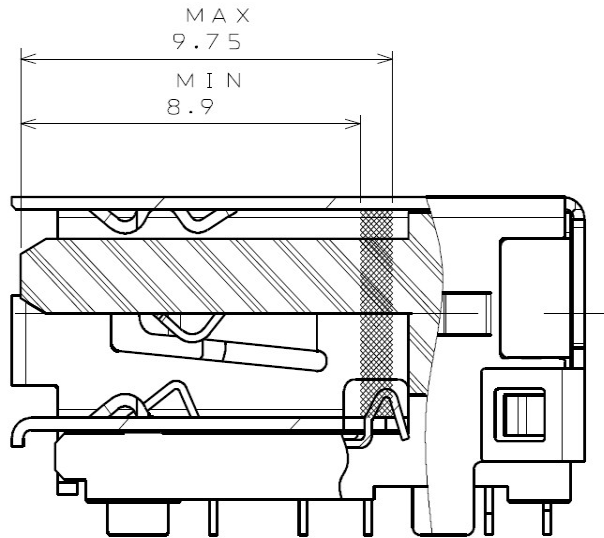


### 3.2.2 USB PD Standard-A PD Detect Mechanical Requirement

Figure 3-7 shows the mechanical dimensions for the range where the PD Detect mechanism on the PD Standard-A receptacle shall indicate PD Detect when a PD Standard-A plug is inserted. The dimensions shall apply to planes parallel to Datum A and shall apply to both USB 2.0 and USB 3.1 versions of PD Standard-A receptacles. Implementation is vendor specific. The sample implementation in the Figure 3-7 is for reference only. PD detect electrical characteristics are defined in Section 3.1.6. The implementation shall also conform to the following requirements:

- The metal shell of the USB PD Standard-A plug is nominally 1.3 mm longer than the USB Standard-A plug to provide a means for PD Detect.
- The metal shell of the USB PD Standard-A plug shall be gold plated on the inner and outer surfaces a minimum of 1.6 mm from the leading edge.
- Contact for PD Detect shall occur on either the inner or the outer surface of the plug shell.
- The insertion of a USB ThinCard shall not indicate PD Detect.

Figure 3-7 PD Detect Plane Location Range for PD Standard-A Receptacles



### 3.2.3 USB 2.0 PD Standard-A Connector

#### 3.2.3.1 Interface Definition

Figure 3-8, Figure 3-9, and Figure 3-10 show the USB 2.0 PD Standard-A plug, the USB 2.0 PD Standard-A receptacle and a reference footprint for the USB 2.0 PD Standard-A receptacle, respectively.

Mechanical requirements governing mating interoperability of the USB 2.0 PD Standard-A receptacle and USB PD Standard-A plug are included in this specification. For the USB 2.0 PD receptacle, Datum A was moved from the front edge of the receptacle shell to the front of the tongue to be consistent with [USB 3.1] and to provide proper dimensioning for PD Detect features. Similarly for the USB 2.0 PD Standard-A plug, Datum A was moved from the front edge of the plug shell to the back of the tongue.

Figure 3-8 defines the mechanical requirements that govern the mating interoperability that shall be followed for the USB 2.0 Standard-A plug. See Section 3.2.2 for additional USB PD Standard-A plug requirements.

Figure 3-9 defines the normative mechanical requirements the USB 2.0 PD Standard-A receptacle that govern the mating interoperability and informative dimensions for solder tail locations, a sample Insertion Detect implementation, and a sample PD Detect implementation. Dimensions associated with solder tails, the sample Insertion Detect feature, and the sample PD Detect features are not normative. The solder pin locations are vendor-specific and included in the drawings for reference only. See Section 3.2.1 and Section 3.2.2 for normative Insertion Detect and PD Detect mechanical requirements, respectively.

The through-hole footprint in Figure 3-10 is shown as an example. Other footprints are allowed.

General considerations:

- There may be some increase in the USB 2.0 PD Standard-A receptacle connector depth (into a system board) as compared to the USB 2.0 Standard-A receptacle to accommodate detection of the Standard-A plug's longer shell.
- Drawings for stacked USB 2.0 PD Standard-A receptacles are not shown in this specification. They are allowed, as long as they meet all the electrical requirements defined in [USB 2.0] and the mating interoperability mechanical

and electrical requirements defined in this specification. When designing stacked USB 2.0 PD Standard-A receptacles, PD Detect contacts shall be provided in all PD capable receptacles.

Figure 3-8 PD Standard-A Plug Interface Dimensions

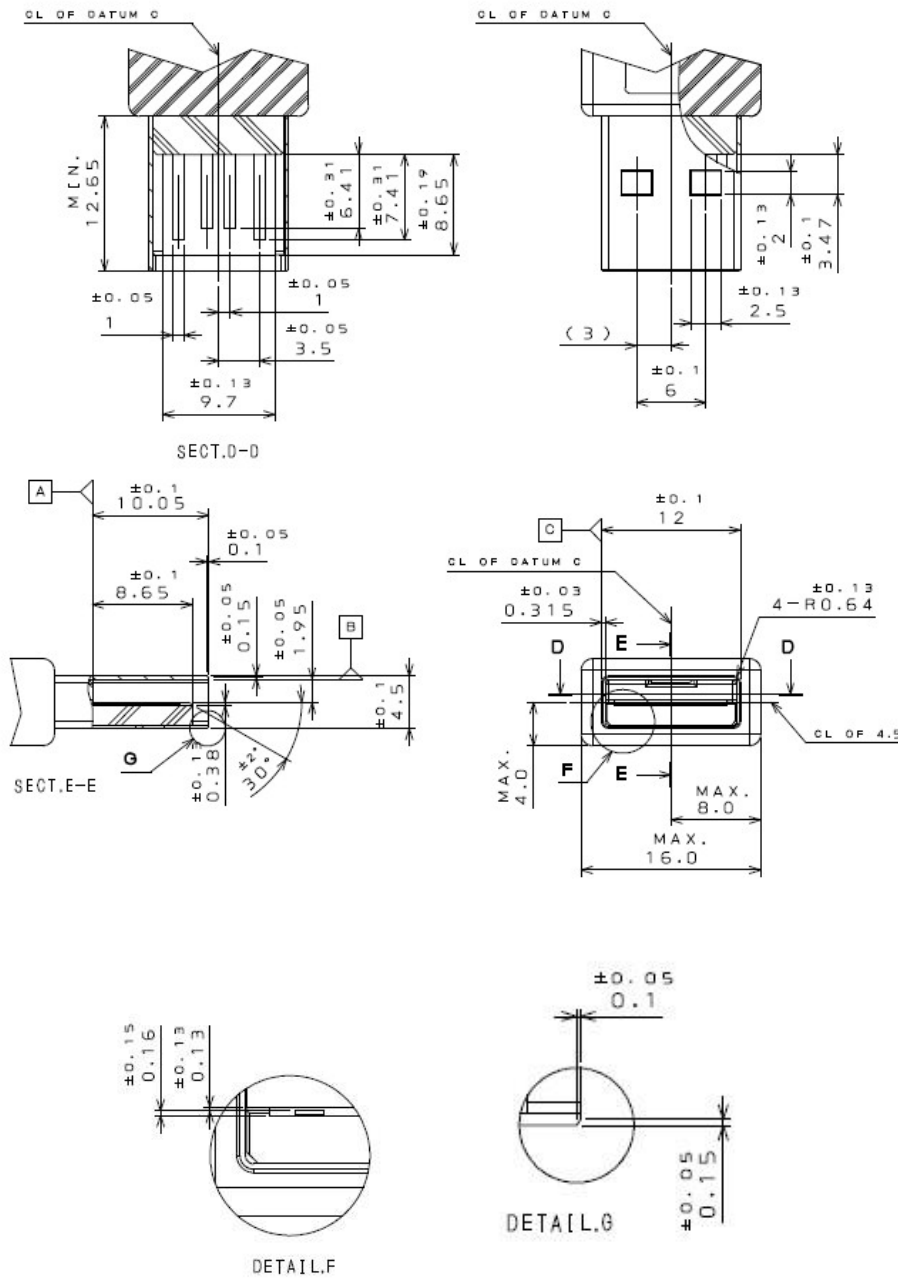
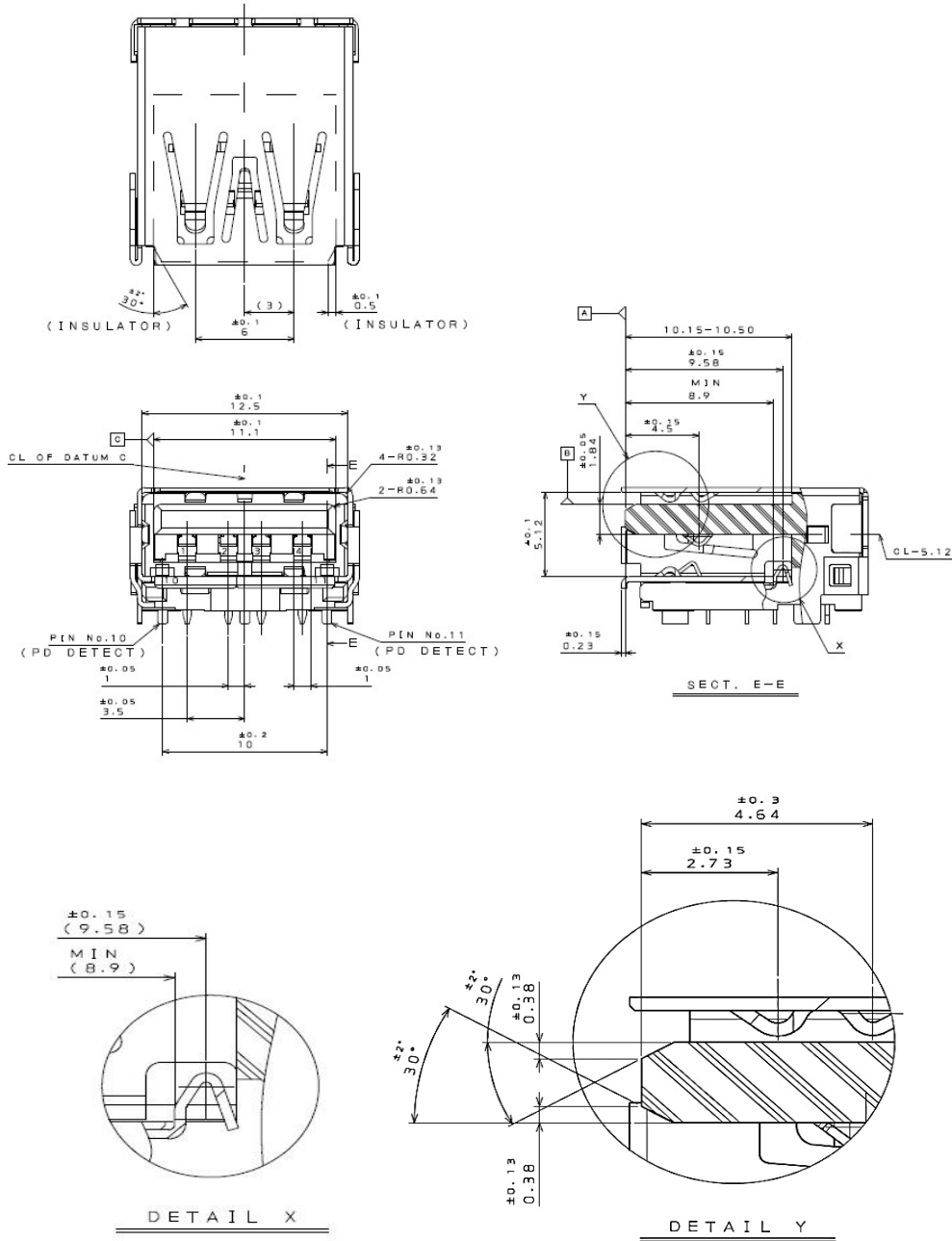


Figure 3-9 USB 2.0 PD Standard-A Receptacle Interface Dimensions





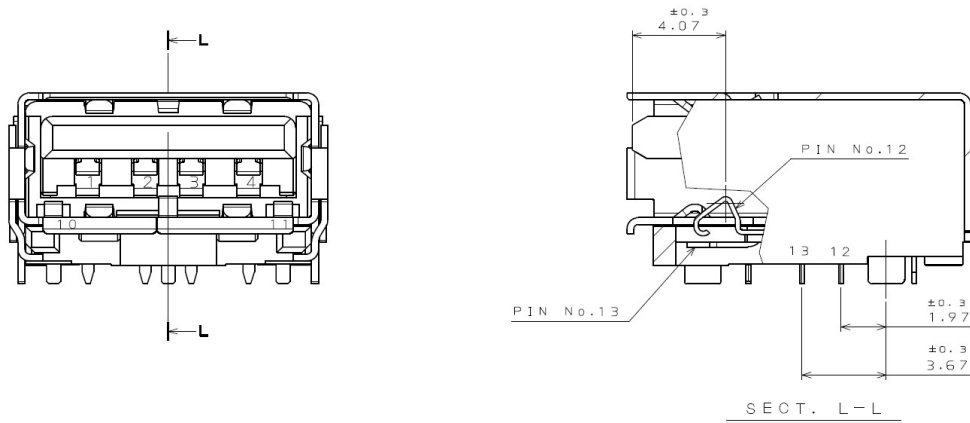
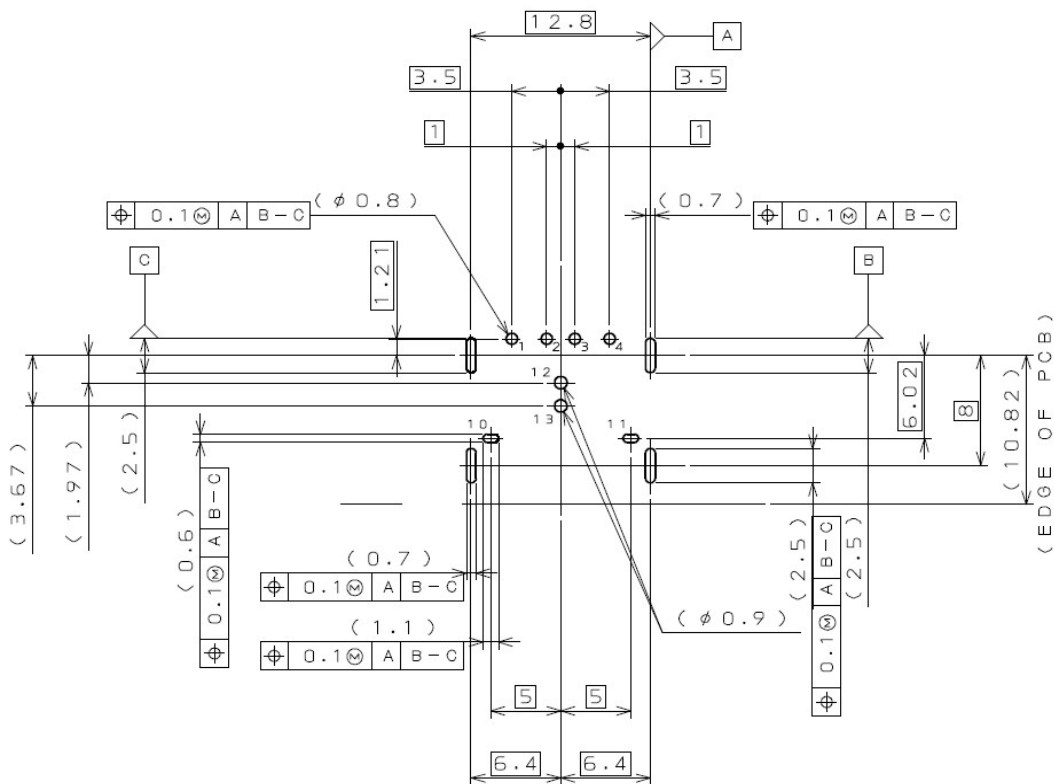


Figure 3-10 Reference Footprint for the USB 2.0 PD Standard-A Top Mount Single Receptacle (Informative)



### 3.2.3.2 Pin Assignments and Description

The usage and assignments of the pins that shall be used in the USB 2.0 PD Standard-A connector are defined in Table 3-2.

Table 3-2 USB 2.0 PD Standard-A Connector Pin Assignments

Pin Number <sup>1</sup>	Signal Name	Description	Mating Sequence
1	V <sub>BUS</sub>	Power	Third
2	D-	Differential pair as defined in <a href="#">[USB 2.0]</a>	Fourth
3	D+		
4	GND	Ground for power return	Third
10 <sup>2</sup>	PD DETECT 1	Contact in PD receptacle to detect a PD plug	Last
11 <sup>2</sup>	PD DETECT 2	Contact in PD receptacle to detect a PD plug	Last
12 <sup>3</sup> ,13	INSERTION DETECT	Receptacle only. Detects insertion of a plug into the receptacle. Optional except for Cold Socket applications.	Second
Shell	Shield	Connector metal shell	First
Note 1: Pin numbers not included in this table do not have contacts present. Pin numbering is consistent with location across multiple USB connector types. Note 2: Implementation of PD DETECT shall include: a) either pin 10 or pin 11. b) both pin 10 and pin 11. Note 3: Pin 12, if present, shall be connected to Shield.			

The physical location of the pins in the connector is illustrated in Figure 3-9.

## 3.2.4 USB 3.1 PD Standard-A Connector

### 3.2.4.1 Interface Definition

Figure 3-11, Figure 3-12, and Figure 3-13 show the USB 3.1 PD Standard-A plug, the USB 3.1 PD Standard-A receptacle and a reference footprint for the USB 3.1 PD Standard-A receptacle, respectively.

Figure 3-11 defines the mechanical requirements of the USB 3.1 PD Standard-A plug that govern the mating interoperability that are different than the dimensions in the [\[USB 2.0\]](#) and [\[USB 3.1\]](#) specifications, that shall be followed, for the USB 3.1 Standard-A plug. See Section 3.2.1 for additional USB PD Standard-A plug requirements.

Figure 3-12 provides an informative example of a USB 3.1 PD Standard-A receptacle including dimensions for solder tail locations, a sample Insertion Detect implementation, and a sample PD Detect implementation. The dimensions in Figure 3-12 associated with solder tails, the sample Insertion Detect feature, and the sample PD Detect features are not normative. The solder pin locations are vendor-specific and included in the drawings for reference only. The USB 3.1 PD Standard-A receptacle mating interface shall comply with the dimensions defined in the [\[USB 2.0\]](#) and [\[USB 3.1\]](#) specifications for the USB 3.1 Standard-A receptacle. See Section 3.2.1 and Section 3.2.2 for normative Insertion Detect and PD Detect mechanical requirements, respectively.

The through-hole footprint in Figure 3-13 is shown as an example. Other footprints are allowed.

General considerations:

- Drawings for stacked USB 3.1 PD Standard-A receptacles are not shown in this specification. They are allowed, as long as they meet all the electrical and mechanical requirements defined in [\[USB 2.0\]](#), [\[USB 3.1\]](#), and this specification. When designing a stacked USB 3.1 PD Standard-A receptacle, PD Detect contacts shall be provided in all PD capable receptacles.

Figure 3-11 USB 3.1 PD Standard-A Plug Interface Dimensions

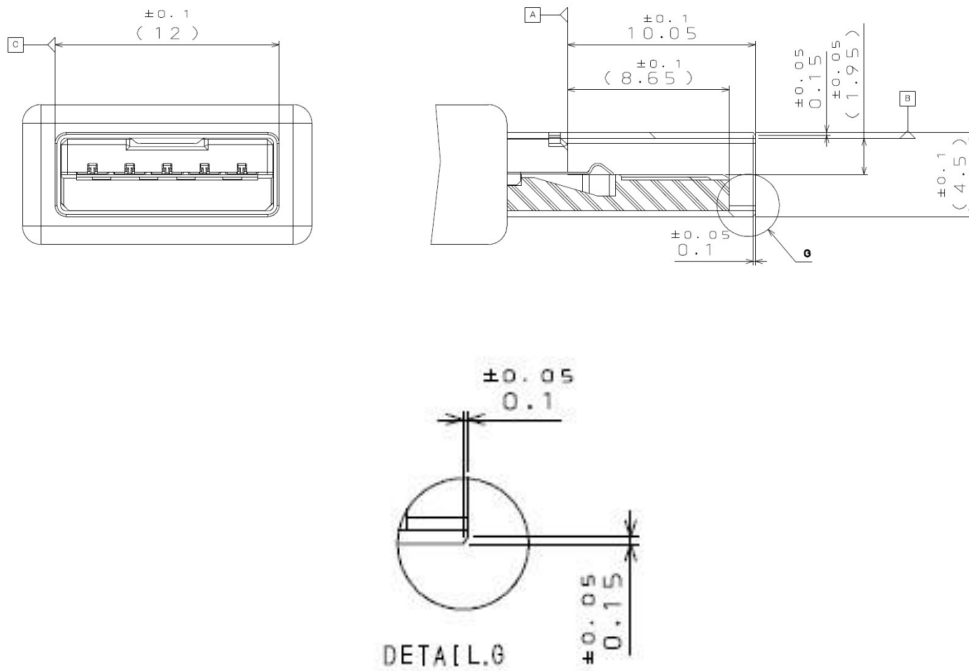


Figure 3-12 Reference USB 3.1 PD Standard-A Receptacle Interface Dimensions (Informative)

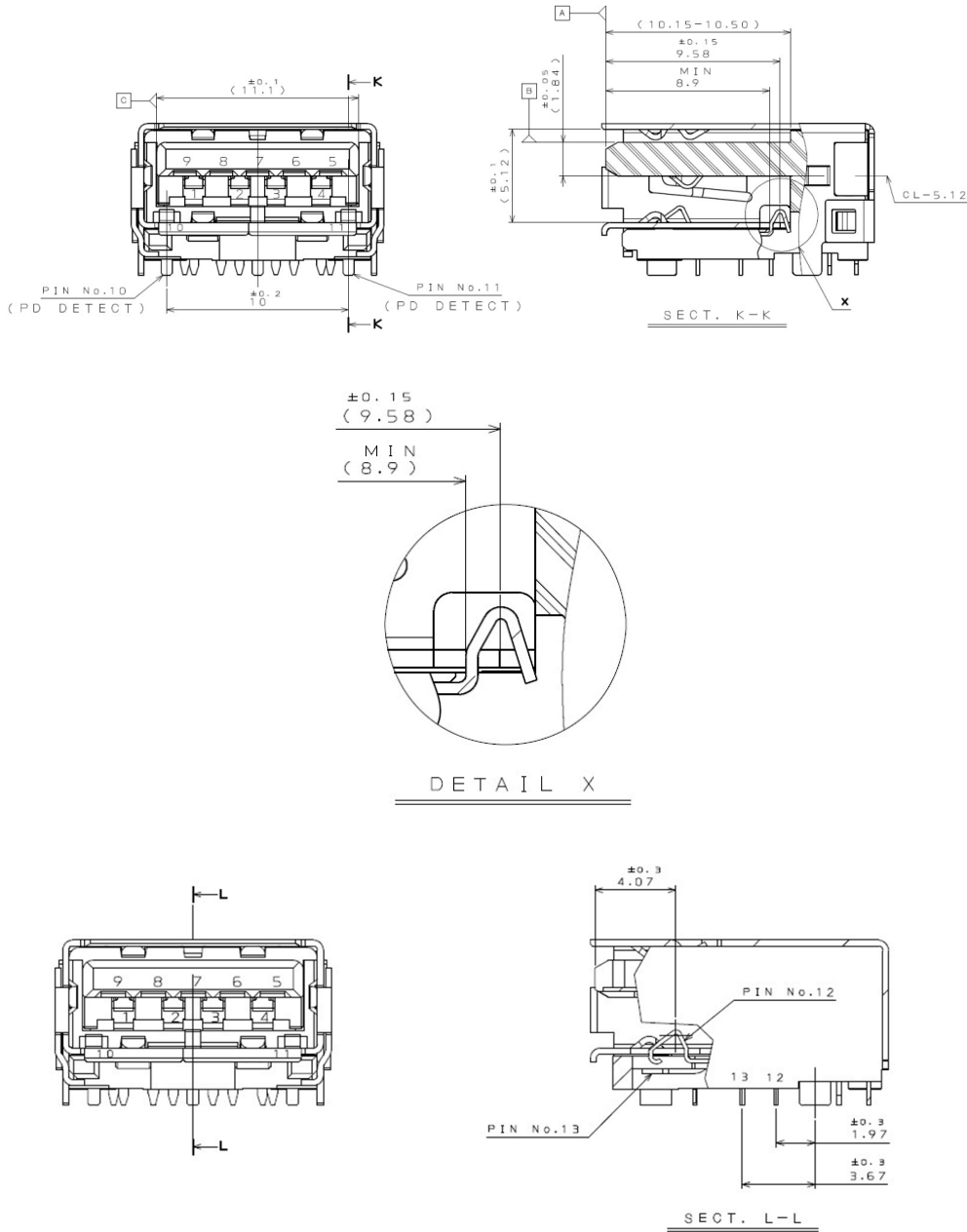
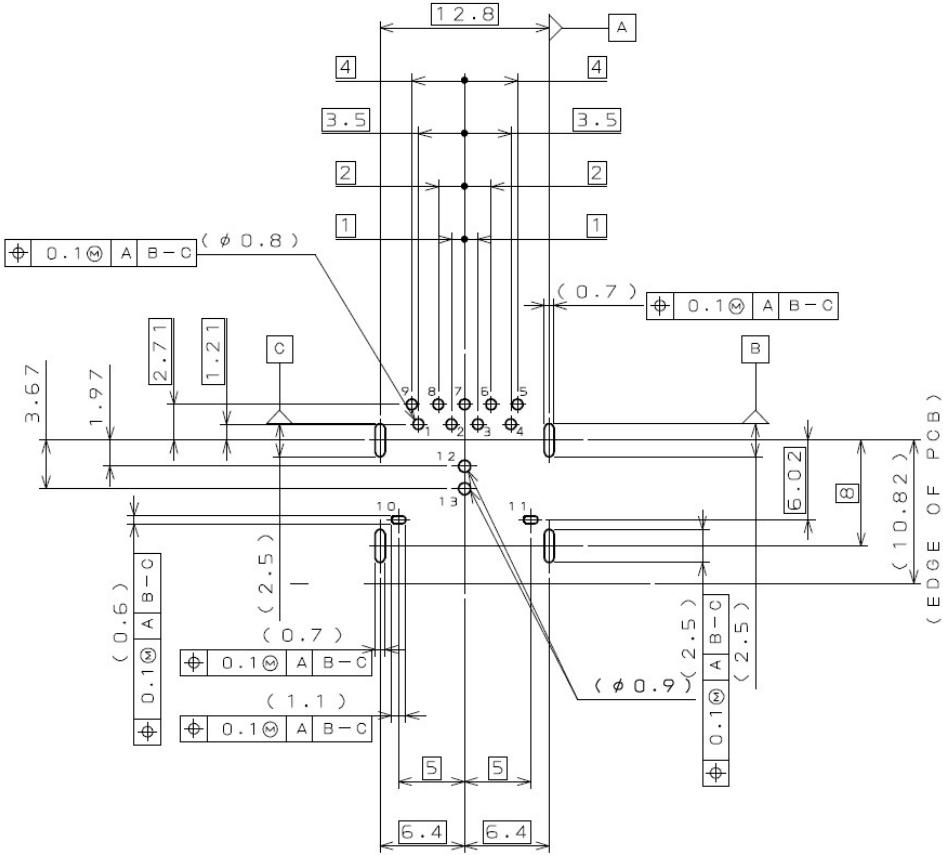


Figure 3-13 Reference Footprint for the USB 3.1 PD Standard-A Top Mount Single Receptacle (Informative)



### 3.2.4.2 Pin Assignments and Description

The usage and assignments of the pins that shall be used in the USB 3.1 PD Standard-A connector are defined in Table 3-3.

Table 3-3 USB 3.1 PD Standard-A Connector Pin Assignments

Pin Number <sup>1</sup>	Signal Name <sup>2</sup>	Description	Mating Sequence
1	V <sub>BUS</sub>	Power	Third
2	D-	Differential pair as defined in <a href="#">[USB 2.0]</a>	Fourth
3	D+		
4	GND	Ground for power return	Third
5	StdA_SSRX-	Enhanced SuperSpeed receiver differential pair	Fifth
6	StdA_SSRX+		
7	GND_DRAIN	Ground for signal return	
8	StdA_SSTX-	Enhanced SuperSpeed transmitter differential pair	
9	StdA_SSTX+		
10 <sup>3</sup>	PD DETECT 1	Contact in PD receptacle to detect a PD plug	Last
11 <sup>3</sup>	PD DETECT 2	Contact in PD receptacle to detect a PD plug	Last
12 <sup>4</sup> , 13	INSERTION DETECT	Receptacle only. Detects insertion of a plug into the receptacle. Optional except for Cold Socket applications.	Second
Shell	Shield	Connector metal shell	First
<p>Note 1: Pin numbers not included in this table do not have contacts present. Pin numbering is consistent with location across multiple USB connector types.</p> <p>Note 2: Tx and Rx are defined from the host perspective.</p> <p>Note 3: Implementation of PD DETECT shall include :</p> <ul style="list-style-type: none"> <li>a) either pin 10 or pin 11.</li> <li>b) both pin 10 and pin 11.</li> </ul> <p>Note 4: Pin 12, if present, shall be connected to Shield.</p>			

The physical location of the pins in the connector is illustrated in Figure 3-12. Note: pins 1 to 4 are referred to as the USB 2.0 pins, while pins 5 to 9 are referred to as the Enhanced SuperSpeed pins.

## 3.2.5 USB 2.0 PD Standard-B Connector

### 3.2.5.1 Interface Definition

Figure 3-15, Figure 3-14, and Figure 3-16 show the USB 2.0 PD Standard-B plug, the USB 2.0 PD Standard-B receptacle, and a reference footprint for the USB 2.0 PD Standard-B receptacle, respectively. Solder pin locations on the USB 2.0 PD Standard-B receptacle are vendor-specific and are included in the drawings for reference only. The views in the plug and receptacle figures correspond to those shown in the [\[USB 2.0\]](#) base specification and non-reference dimensions define the ID pin for PD applications. The USB 2.0 PD Standard-B plug and receptacle shall comply with the mechanical requirements specified in the [\[USB 2.0\]](#) base specification and the ID pin as specified in Figure 3-15 and Figure 3-14.

Figure 3-14 USB 2.0 PD Standard-B Plug Interface Dimensions

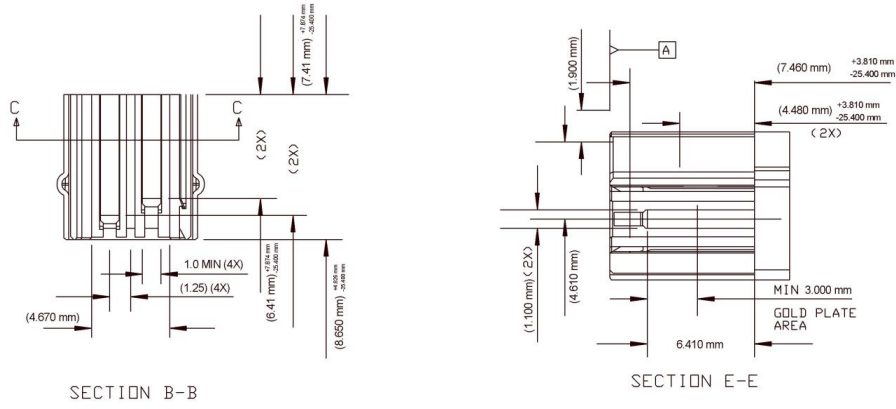
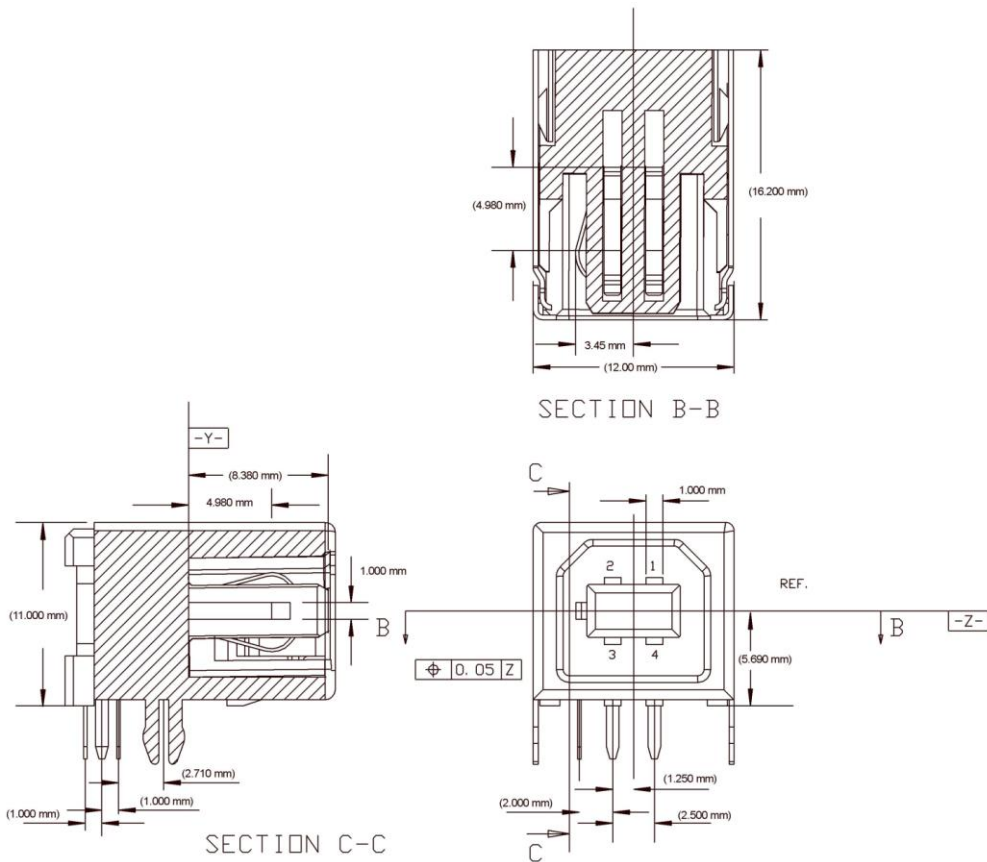


Figure 3-15 USB 2.0 PD Standard-B Receptacle Interface Dimensions







the USB 3.1 PD Standard-B receptacle are vendor-specific and included in the drawings for reference only. The views in the plug and receptacle figures correspond to those shown in the [\[USB 3.1\]](#) base specification and non-reference dimensions define the ID pin for PD applications. The USB 3.1 PD Standard-B plug and receptacle shall comply with all other mechanical requirements specified in the [\[USB 3.1\]](#) base specification and the ID pin as specified in Figure 3-17 and Figure 3-18.

**Figure 3-17 USB 3.1 PD Standard-B Plug Interface Dimensions**

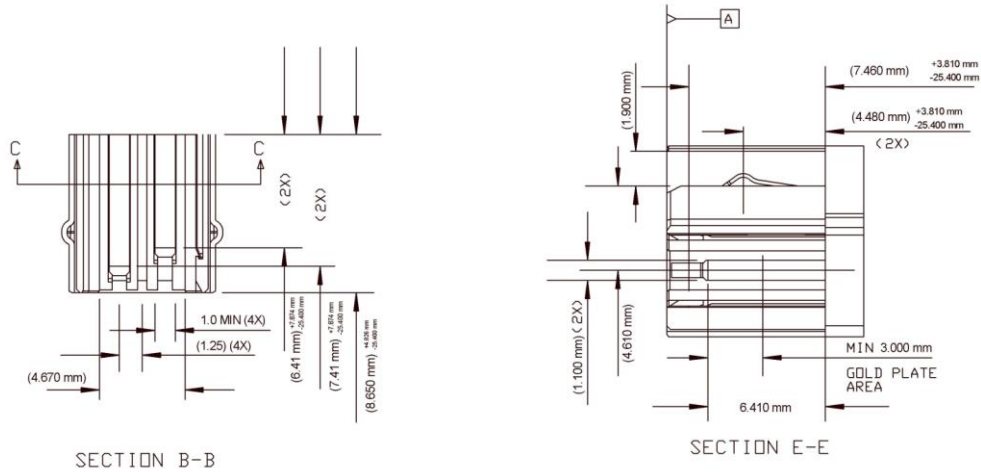
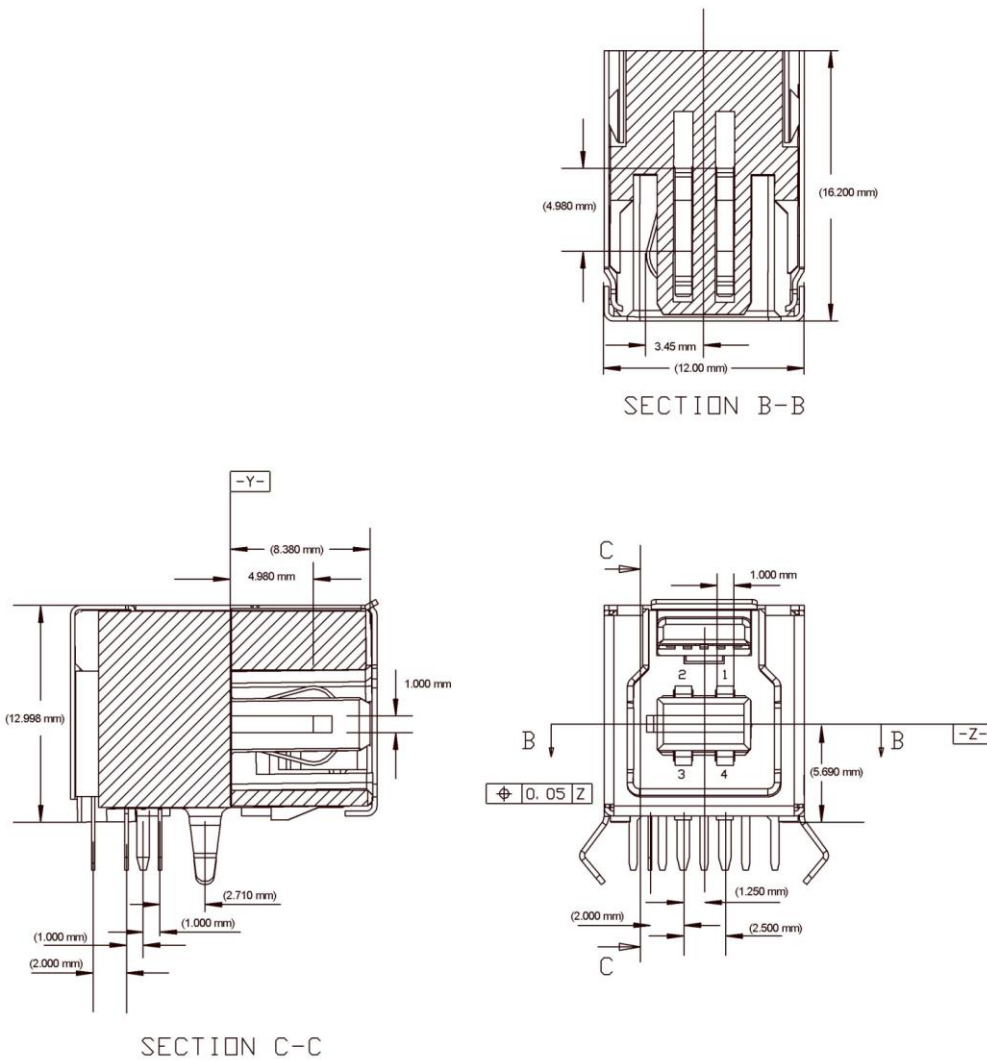


Figure 3-18 USB 3.1 PD Standard-B Receptacle Interface Dimensions



GENERAL TOLERANCE IS +/- 0.10  
UNLESS OTHERWISE SPECIFIED



### 3.2.6.2 Pin Assignments and Descriptions

The usage and assignments of the pins that shall be used in the USB 3.1 PD Standard-B connector are defined in Table 3-5.

**Table 3-5 USB 3.1 PD Standard-B Connector Pin Assignments**

Pin Number <sup>1</sup>	Signal Name <sup>2</sup>	Description	Mating Sequence
1	V <sub>BUS</sub>	Power	Second
2	D-	Differential pair as defined in <a href="#">[USB 2.0]</a>	Third or beyond
3	D+		
4	GND	Ground for power return	Second
5	StdB_SSTX-	Enhanced SuperSpeed transmitter differential pair	Third or beyond
6	StdB_SSTX+		
7	GND_DRAIN	Ground for signal return	
8	StdB_SSRX-	Enhanced SuperSpeed receiver differential pair	
9	StdB_SSRX+		
11 <sup>3</sup>	ID	Identification of PD capability	
Shell	Shield	Connector metal shell	First
<p>Note 1: Pin numbers not included in this table do not have contacts present. Pin numbering is consistent with location across multiple USB connector types.</p> <p>Note 2: Tx and Rx are defined from the device perspective.</p> <p>Note 3: ID pin as defined in core USB specifications and extended by the USB Power Delivery specifications. See Section 3.4.3. Pin 11 was defined by <a href="#">[USB 3.1]</a> as the Ground Return for DPWR (power supplied by the device) for the USB 3.1 Powered-B connector.</p>			

The physical locations of the pins in the connector are illustrated in Figure 3-16 and Figure 3-17.

### 3.3 Cable Assemblies

USB Power Delivery introduces the concept of an electronically marked cable assembly. The particular marking denotes the cable assembly's characteristics and is electronically detected. This provides devices on each end of the cable assembly a means to detect and identify the cable assembly capabilities. The ID pin definition is extended to electronically mark cable assemblies. An ID pin has been added to the Standard-B Connectors to create PD Standard-B Connectors as defined in Section 3.2.5 and Section 3.2.6. The B-Device shall detect the cable assembly and make requests that are consistent with the cable assembly's capabilities. Similarly, the Standard-A Connector shell has been modified to create a PD Standard-A Connector to allow detection of cable insertion and to identify if the cable assembly is PD-capable. The combination of these PD cable assembly markings provide a robust system to safely deliver high power across the USB cable assembly.

The portion of this specification that allows the negotiation of voltages other than the default *vSafe5V* and currents in excess of 1.5A shall apply only to Devices with marked PD cable assemblies.

#### 3.3.1 Non-marked Cable Assemblies

Limitations are placed on the use of legacy cable assemblies (i.e., ones not marked). Legacy cable assemblies include all cable assemblies with USB 2.0 or USB 3.1 Standard-A Connectors, USB 2.0 or USB 3.1 Standard-B Connectors, USB 3.1 Powered-B Connectors, and USB 2.0 or USB 3.1 Micro-B Connectors. Devices shall only use these non-marked legacy cable assemblies at *vSafe5V* and up to 1.5A as described by [\[USB 2.0\]](#), [\[USB 3.1\]](#) and [\[BC 1.2\]](#).

#### 3.3.2 Marked Cable Assemblies

Marking allows Port Partners to detect the insertion of a USB Power Delivery cable assembly and its electrical characteristics. Port Partners shall not negotiate for currents in excess of the electrical characteristics indicated by the cable's marking.

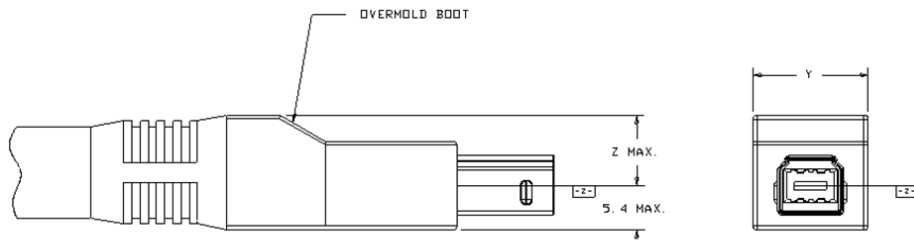
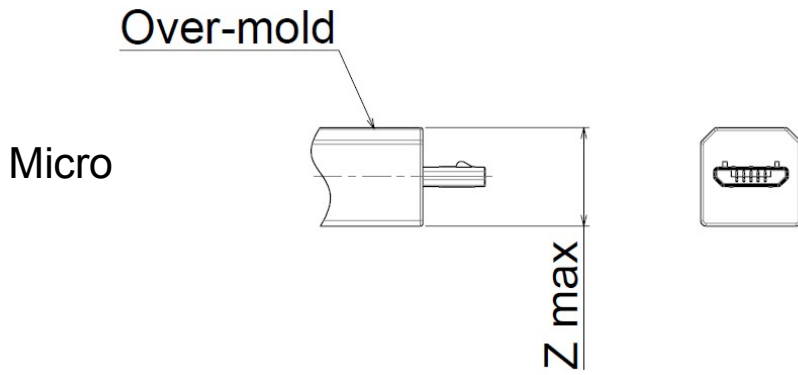
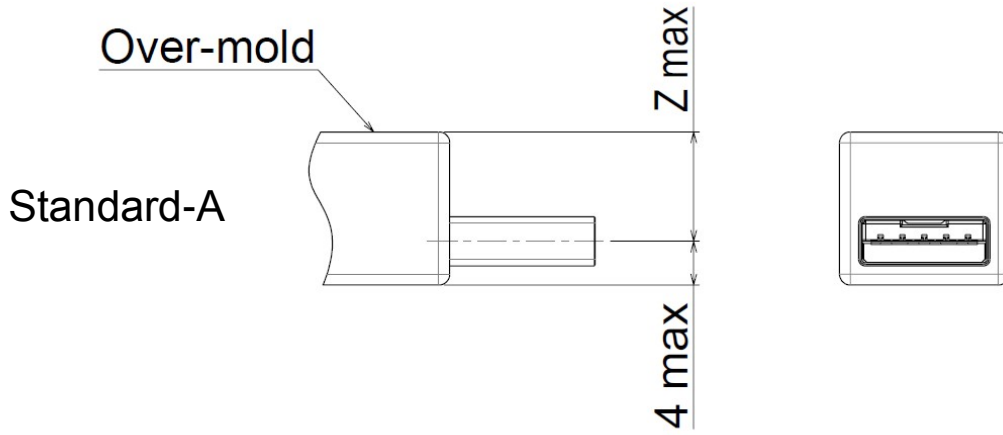
#### 3.3.3 PD Cable Assembly Overmold Requirements

It is strongly recommended that the PD cable assembly overmold dimensions conform to the requirements defined by [\[USB 2.0\]](#) and [\[USB 3.1\]](#). Depending on the PD cable assembly current rating and length, the wire gauge to meet the IR drop defined in 3.6.9 may require a cable diameter that results in a larger overmold at the cable/connector interface. Table 3-6 and Figure 3-20 define the maximum allowed overmold dimensions for the PD cable assembly. PD cable assemblies using these larger limits may result in mechanical interferences (e.g., may block use of a connector slot in stacked connector configurations) therefore, the overmold dimensions should conform to requirements defined by [\[USB 2.0\]](#) and [\[USB 3.1\]](#) if possible.

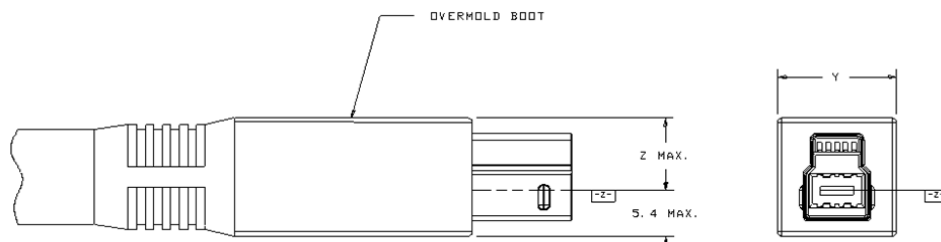
Table 3-6 USB PD Cable Assembly Overmold Maximum Dimensions

Dimension	Connector Type	3A	5A
Z max	PD Standard-A	5.00	10.00
Z max	PD Micro	9.00	-
Y max	PD Standard-B	-	14.00
Z max	PD Standard-B	-	8.60

Figure 3-20 USB PD Cable Assembly Overmold Maximum Dimensions



MAXIMUM OVERMOLD BOOT  
FOR USB 2.0 STD B PLUG



MAXIMUM OVERMOLD BOOT  
FOR USB 3.0 STD B PLUG

### 3.4 PD Cable Assembly Marking

Paragraphs in this section described the markings for the plug connectors at each end of the PD cable assembly. References to cable assembly in this section apply to cable assemblies which include a PD plug connector. Component values are found in Table 3-7.

#### 3.4.1 Marker for PD Standard-A Connectors

The PD Standard-A Connector shell is 1.3 mm longer than the shell of the legacy Standard-A Connector. The PD Standard-A Connector shell shall be detected by the DFP connector using a PD Standard-A receptacle connector and associated marking detection circuitry to indicate that the cable assembly is a PD cable assembly. Marking detection circuitry is vendor specific.

#### 3.4.2 Electronic Markers for Micro-A Plugs

In the standard cable assembly with a Micro-A plug, the ID-pin is grounded. The data and shield connections shall be made per the [\[USB 2.0\]](#) and the [\[USB 3.1\]](#) specifications.

##### 3.4.2.1 Legacy Micro-A Plug

Figure 3-21 Schematic of a Micro-A Plug Legacy Termination

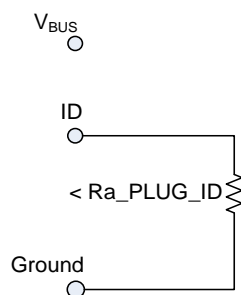


Figure 3-21 shows the schematic diagram of the Micro-A plug termination in a legacy cable assembly. Note the ID pin is connected to ground through an impedance of  $R_{a\_PLUG\_ID}$  (as defined in the Micro-USB Cables and Connectors specification v1.01 in [\[USB 2.0\]](#)) to indicate that the plug is an A plug. For PD to remain backward compatible, the low



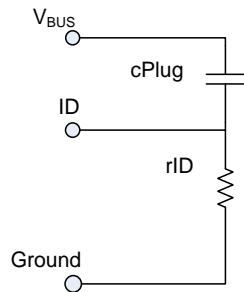
impedance to ground shall be maintained. However, any value less than  $rID$  is assumed to be interpreted by a legacy Port as a Micro-A plug.

PD uses two new markers to allow the detection of a Low Power cable assembly and PD cable assembly in addition to the detection of a legacy cable.

### 3.4.2.2 Low Power Micro-A Plug

Devices with a hardwired captive cable with a Micro-A plug connector supporting a one cell Lithium battery as their power source or where very low power consumption is important may terminate the Micro-A Plug's ID pin to ground with  $rID$  and to  $V_{BUS}$  with  $cPlug$ . This termination may be detected electrically. See 4.4.4 for additional information regarding Low Power Device characteristics.

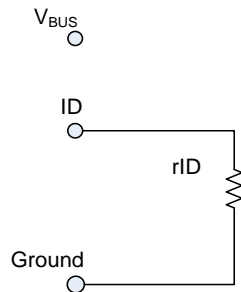
Figure 3-22 Schematic of a Micro-A Plug Marker Indicating Low Power Capability



### 3.4.2.3 PD Micro-A Plug

PD cable assemblies with Micro-A plug connectors shall be marked with  $rID$  terminating the ID pin to ground. See Figure 3-23 for a schematic diagram of the connector termination in a PD cable assembly using a Micro-A plug.

Figure 3-23 Schematic of a Micro-A PD Plug



## 3.4.3 Electronic Markers for PD Standard-B Plugs and Micro-B Plugs

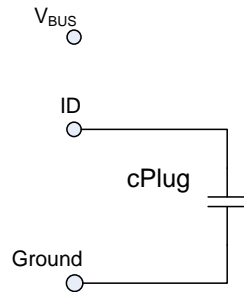
Electronic markers for PD cable assemblies with PD Standard-B plug connectors and Micro-B plug connectors are specified in this section. A legacy Micro-B connector's ID pin exhibits a very high resistance to ground. To maintain backward compatibility, capacitors are used as markers.

Note the 3A and 5A markers are mutually exclusive; hence both markers shall not be present at the same time.

### 3.4.3.1 3A Capable PD B Plug

The schematic diagram for a 3A-capable PD cable assembly detailing how a PD Standard-B plug connector or a Micro-B plug connector shall be marked is shown in Figure 3-24.

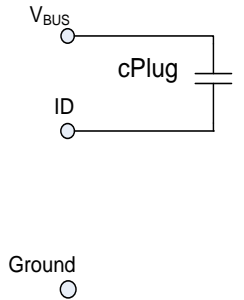
Figure 3-24 Schematic of a B Plug Connector Marker Indicating 3A Capability



### 3.4.3.2 5A Capable PD B Plug

The schematic diagram for a 5A-capable PD cable assembly detailing how a PD Standard-B plug connector shall be marked is shown in Figure 3-25.

Figure 3-25 Schematic of a B Plug Connector Marker Indicating 5A Capability



## 3.5 USB PD Icon

The USB PD cable assemblies shall display the USB icon appropriate for PD. A dimensioned drawing and allowable usage of the icon are supplied with the license from the USB-IF.

## 3.6 USB Power Delivery Cable Requirements

The methods used to mark the USB PD cable assemblies and USB PD cable adaptors are found in Section 3.4. The USB PD connector family shall conform to all requirements in Section 3, in addition to the requirements specified in the [\[USB 2.0\]](#), [\[USB 3.1\]](#) or [\[BC 1.2\]](#) specifications. Test sequences shall conform to EIA-364-1000.001 as specified in the Environmental Requirements section of [\[USB 3.1\]](#).

Some USB PD cable assemblies may be designed for use as power only (i.e., no USB data communication). Requirements that relate to the transfer of USB data may not apply to these cables.

### 3.6.1 Low Level Contact Resistance (EIA 364-23B)

The power contacts of a 3A PD cable assembly shall conform to the following requirements:

- 20m $\Omega$  (Max) initial for  $V_{BUS}$  and GND contacts.
- Maximum change (delta) of +10m $\Omega$  after environmental stresses.
- Measure at 20mV (Max) open circuit at 100mA.

The power contacts of a 5A PD cable assembly shall conform to the following requirements:

- 20m $\Omega$  (Max) initial for  $V_{BUS}$  and GND contacts.
- Maximum change (delta) of +10m $\Omega$  after environmental stresses.
- Measure at 20mV (Max) open circuit at 100mA.

The following requirements shall apply, independent of power being applied, to the resistance of a PD Standard-A plug receptacle shell to the PD Standard-A receptacle PD DETECT contact with a PD Standard-A plug in the mated condition or to the resistance between the INSERTION DETECT contacts when a Standard-A plug is present in the Standard-A receptacle supporting Insertion Detect. For PD Detect, there are two contact interfaces to the plug shell included in series in this measurement:

- 200m $\Omega$  (Max) initial.
- Maximum change (delta) of 300m $\Omega$  after environmental stresses.
- Measure at 20mV (Max) open circuit at 100mA.

### 3.6.2 Open Circuit Resistance

The following requirements shall apply, independent of power being applied, to the resistance of the PD Standard-A receptacle shell to the PD Standard-A receptacle PD DETECT contact(s) when no PD Standard-A plug is inserted in the mated condition or a non-PD Standard-A plug is inserted in the mated condition or to the resistance between the INSERTION DETECT contacts of a Standard-A receptacle supporting Insertion Detect when a Standard-A plug is not inserted:

- $\geq 10M\Omega$  initial.
- $\geq 10M\Omega$  after environmental stress.

### 3.6.3 Dielectric Strength (EIA 364-20)

No breakdown shall occur when 100VAC (RMS) is applied between adjacent contacts of unmated and mated connectors.

### 3.6.4 Insulation Resistance (EIA 364-21)

There shall be a minimum of 100M $\Omega$  insulation resistance between adjacent contacts of unmated and mated connectors.

### 3.6.5 Contact Current Rating

#### 3.6.5.1 3A PD Connector Mated Pair (EIA 364-70, Method 2)

When a current of 3.0 A is applied to the  $V_{BUS}$  pin and its corresponding GND pin (i.e., pin 1 and pin 5 of the PD Micro-A Connector, PD Micro-AB Connector, or PD Micro-B Connector), the delta temperature shall not exceed +30°C at any point on the connectors under test, when measured at an ambient temperature of 25°C.

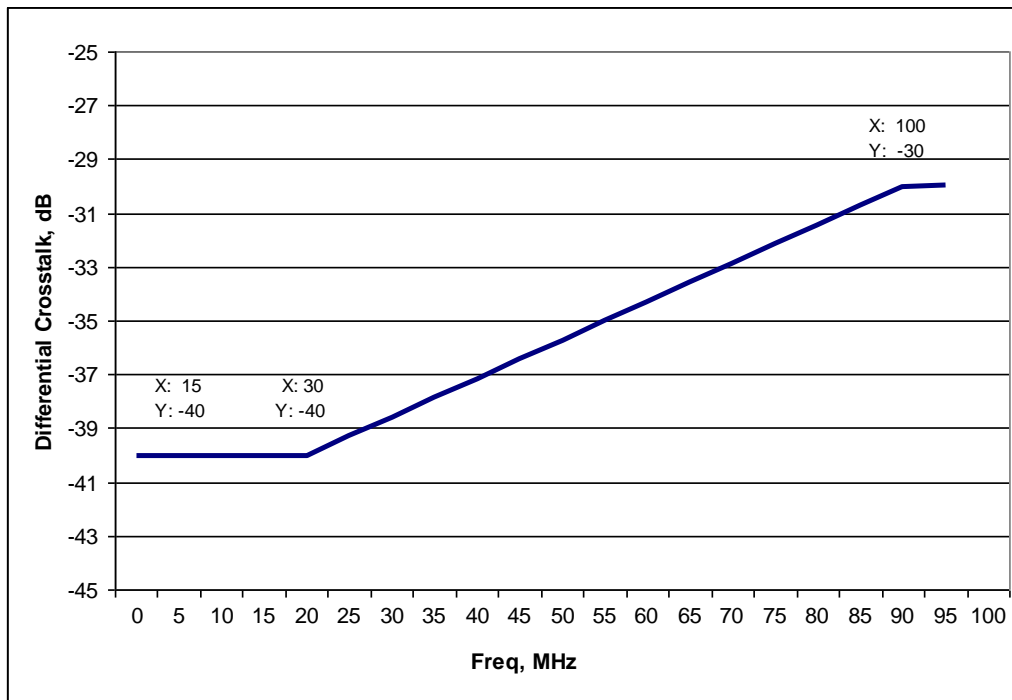
#### 3.6.5.2 5A PD Connector Mated Pair (EIA 364-70, Method 2)

When a current of 5.0A is applied to the  $V_{BUS}$  pin and its corresponding GND pin (i.e., pin 1 and pin 4 of the PD Standard-B Connector or PD Standard-A Connector), the delta temperature shall not exceed +30°C at any point on the connectors under test, when measured at an ambient temperature of 25°C.

### 3.6.6 Differential Crosstalk between $V_{BUS}$ and the D+/D- Pair (EIA-360-90)

The differential, near-end, and far-end, crosstalk between the D+/D- pair and  $V_{BUS}$  shall be managed not to exceed the limit shown in Figure 3-26.

Figure 3-26 Differential Near-End and Far-End Crosstalk Requirement between the D+/D- Pair and  $V_{BUS}$



### 3.6.7 PD Cable Assembly Shielding Connectivity

The shield conductor shall be attached to the connector shell at both ends of the cable and shall provide a resistance of no greater than 1.0Ω from end to end. The shield shall not be connected to ground within the cable.

### 3.6.8 PD Cable $V_{BUS}$ Impedance

The cable impedance shall meet the requirements specified in Table 5-18.

### 3.6.9 PD Cable Insertion Loss

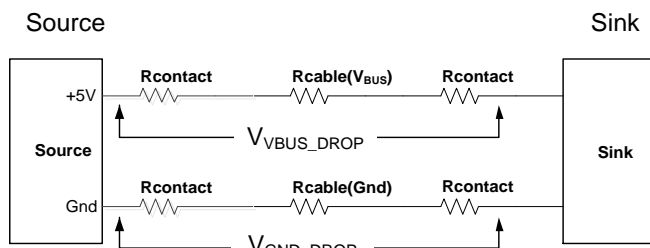
The insertion loss of the cable shall not exceed *insertionLoss*.

### 3.6.10 PD Cable IR Drop Considerations

The maximum voltage drop between the Source and Sink ports is defined to:

- Insure signal integrity of the USB 2.0 signal wires.
- Quantify the worst case voltage seen at the input of a Sink.

Figure 3-27 Voltage Drop Measurement



As shown in Figure 3-27, voltage drop across the cable is measured independently on both the GND and the  $V_{BUS}$  connections. It is inclusive of the cable and connectors at both ends.

#### 3.6.10.1 Voltage Drop on the Ground

A PD cable assembly shall ensure  $V_{GND\_DROP}$  does not exceed the maximum value listed in Table 3-7 for the signal ground reference between the Source's power source connection to the USB receptacle and the Sink's connection to its internal power conversion block, if present.  $V_{GND\_DROP}$  shall be measured at *iCable* and measured either:

- At the receptacle's solder pad where it attaches to the printed circuit board
- Or where the captive cable is attached to the printed circuit board in the device.

$V_{GND\_DROP}$  shall include the effects of the following:

- 1) The mated contact resistance of both the A-side and the B-side (if present) connections for GND.
- 2) The series resistance of the cable's GND wire.
- 3) The rated current of the cable (*iCable*).

$V_{GND\_DROP}$  for removable cables equals  $(2 * R_{contact} + R_{cable}(GND)) * i_{Cable}$ .  $V_{GND\_DROP}$  for captive cables equals  $(1 * R_{contact} + R_{cable}(GND)) * i_{Cable}$ .

The only variable in the above equations is  $R_{cable}(GND)$ . The wire size and its length are the major contributors to  $V_{GND\_DROP}$ . The ground wire size shall be selected so that  $V_{GND\_DROP}$  at a current of *iCable* does not exceed the maximum value listed in Table 3-7.

#### 3.6.10.2 Overall IR Drop between a Source and a Sink

A PD cable assembly shall provide a maximum voltage drop of *vIRDrop.Cable* for  $V_{BUS}$  to GND at the Sink end of the cable. The  $V_{BUS}$  to GND voltage drop shall include  $V_{VBUS\_DROP}$  plus  $V_{GND\_DROP}$ .

$V_{VBUS\_DROP}$  shall be measured at a current of *iCable* and measured either:

- At the receptacle's solder pad where it attaches to the printed circuit board

- Or where the captive cable is attached to the printed circuit board in the device.

$V_{VBUS\_DROP}$  shall include the effects of the following

- The mated contact resistance of both the A-side and B-side (if present) connections for  $V_{BUS}$ .
- The series resistance of cable's  $V_{BUS}$  wire.
- The rated current of the cable ( $i_{Cable}$ ).

$V_{VBUS\_DROP}$  for removable cables equals  $(2 * R_{contact} + R_{cable}(V_{BUS})) * i_{Cable}$ .  $V_{VBUS\_DROP}$  for captive cables equals  $(1 * R_{contact} + R_{cable}(V_{BUS})) * i_{Cable}$ .

The only variable in the above equations is  $R_{cable}(V_{BUS})$ . The wire size and its length are the major contributors to  $V_{VBUS\_DROP}$ . The  $V_{BUS}$  wire size shall be selected so that  $V_{VBUS\_DROP}$  at a current of  $i_{Cable}$  does not exceed the maximum value listed in Table 3-7.

### 3.6.10.3 Example Calculation of Overall IR Drop

The goal of this example is to design a compliant PD Cable with a Standard A and a Micro B plug (e.g.  $i_{Cable} = 3A$ ) that is 1m in length.

The allowable  $R_{cable}(GND)$  to achieve this is found by:

- 1)  $R=E/I$  or  $E = IR \rightarrow R_{cable}(GND) = \text{the maximum value for } V_{GND\_DROP} \text{ listed in Table 3-7} / i_{Cable} \rightarrow 0.375/3 = 125m\Omega$ .
- 2) Mated contact resistance is defined as  $30m\Omega$  (Max) times 2 mated pair connections or  $60m\Omega$ .
- 3)  $R_{cable}(GND) = 125m\Omega - 60m\Omega = 65m\Omega$  (the budget for the ground wire in the cable).

The allowable  $R_{cable}(V_{BUS})$  to achieve this can be found by:

- 1)  $R=E/I$  or  $R = IR \rightarrow R_{cable}(V_{BUS}) = \text{the maximum value for } V_{VBUS\_DROP} \text{ listed in Table 3-7} / i_{Cable} \rightarrow 0.625/3 = 208m\Omega$ .
- 2) Mated contact resistance is defined as  $30m\Omega$  (Max) times 2 mated pair connections or  $60m\Omega$ .
- 3)  $R_{cable}(V_{BUS}) = 208m\Omega - 60m\Omega = 148m\Omega$  (the budget for the  $V_{BUS}$  wire in the cable).

In this example, a 22AWG @  $52.94m\Omega/m$  used for Ground and a 26AWG @  $134m\Omega/m$  used for  $V_{BUS}$  meets the *VIRDrop\_Cable* requirements for a 1m cable.

### 3.7 Electrical Parameters

Table 3-7 shows the parameters used in this Section.

Table 3-7 Electrical Parameters

Parameter	Minimum	Nominal	Maximum	Units	Description
<i>aInsertionLoss</i>			1.5	dB	As measured within <i>fRange</i> .
<i>cPlug</i>	5	10	15	nF	Across temperature -20°C to +80°C at <i>vSafe5V</i> for the minimum and at <i>vSafe0V</i> for the maximum. Minimum voltage rating of 50V. See Figure 3-22, Figure 3-24, and Figure 3-25.
<i>iCable</i>	5			A	If only Standard connectors are used.
	3			A	If a Micro connector is present at any point.
<i>rID</i>	900	1000	1100	Ω	See Figure 3-22 and Figure 3-23.
<i>vIRDrop_Cable</i>			1	V	Sum of <i>V<sub>VBUS_DROP</sub></i> and <i>V<sub>GND_DROP</sub></i> .
<i>V<sub>GND_DROP</sub></i> <sup>1</sup>			375	mV	Note: this value is the same as <i>V<sub>GND_OFFSET</sub></i> as defined in <a href="#">[BC.1.2]</a> .
<i>V<sub>VBUS_DROP</sub></i> <sup>1</sup>			625	mV	
<p>Note 1: For charging-only cables (e.g. cables without data lines), the maximum value of either <i>V<sub>VBUS_DROP</sub></i> or <i>V<sub>GND_DROP</sub></i> may be exceeded; however <i>V<sub>VBUS_DROP</sub></i> plus <i>V<sub>GND_DROP</sub></i> shall not exceed <i>vIRDrop_Cable</i>.</p>					

## 4. Electrical Requirements

This chapter covers the platform's electrical requirements for implementing USB Power Delivery.

USB Power Delivery may be implemented alongside the [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[BC 1.2\]](#) and [\[USBType-C 1.0\]](#) specifications. In the case where a Device requests power via the Battery Charging Specification and then the USB Power Delivery Specification, it shall follow the USB Power Delivery Specification until the Port Pair is unattached or there is a Hard Reset. If the USB Power Delivery connection is lost, the Port shall return to its default state, see Section 6.7.2.

### 4.1 Dead Battery Detection / Unpowered Port Detection

The Power Delivery specification provides a mechanism for a USB Device to provide power to a USB Host under the circumstances where the USB Host:

- Has a Dead Battery that requires charging or
- Has lost its power source or
- Does not have a power source or
- Does not want to provide power.

The USB Peripheral primarily acts as a USB Device that may also provide power to the USB Host if the correct detection is carried out.

The following sections detail mechanisms used for:

- Type-A to Type-B connections:
  - The USB Device is a Consumer/Provider acting as a Source
  - The USB Host is a Provider/Consumer acting as a Sink.

For mapping of  $V_{BUS}$  level to USB states during Dead Battery detection see Figure 9-3.

#### 4.1.1 Type-A to Type-B Dead Battery Operation

##### 4.1.1.1 Overview

A Consumer/Provider shall back power the Provider/Consumer at a reduced current level. This enables the Provider/Consumer's PD transmitter on its DFP to send a continuous stream of alternating '0s' and '1s' (referred to in this section as a Bit Stream) that the Consumer/Provider's UFP's PD receiver can readily detect. The use of a current limited Source (**vSafeDB**) to back power  $V_{BUS}$  is intended to minimize the risk of damage to legacy USB DFPs. At the start of the process when the Consumer/Provider sees no voltage on  $V_{BUS}$ , it probes the bus by back powering  $V_{BUS}$  to see if its Port Partner has a Dead Battery or unpowered Port that it wants powered. If so, the Consumer/Provider outputs **vSafe5V** on  $V_{BUS}$  and becomes the de facto power Provider.

When the process is complete, both ports will have performed an Implicit Power Role Swap without the use of the **PR\_Swap** Message (see Section 8.3.2.7.1.2) where the Consumer/Provider is now operating as the Source while the Provider/Consumer operates as a Sink. With communications and cable capabilities established, the Provider/Consumer may negotiate for a voltage/current combination to charge its battery or to operate.

##### 4.1.1.2 Consumer/Provider Operational Details

In operation (see Figure 4-1), a Consumer/Provider Port that does not detect **vSafe5V** on  $V_{BUS}$  shall periodically apply a current limited five volt supply (**vSafeDB**) to  $V_{BUS}$  in an attempt to ascertain the presence of a Provider/Consumer Port that wants to be powered. In response to **vSafeDB** on  $V_{BUS}$ , a Provider/Consumer Port that wants power shall transmit the Bit Stream that a Consumer/Provider can detect indicating the presence of a Port that wants to be powered. When the Consumer/Provider detects a Provider/Consumer Port wanting to be powered, it shall apply **vSafe5V** to  $V_{BUS}$ . In all other cases, for example when connected to a legacy Port or an unpowered Provider Port, there will be no response and the Consumer/Provider Port shall not apply full power (**vSafe5V**) to  $V_{BUS}$ . The



Consumer/Provider shall continue to periodically probe for the presence of a Provider/Consumer Port that wants power. The limitations on the current and power applied to back power  $V_{BUS}$  and its controlled duration shall be applied in order to prevent damage to legacy ports.

The Consumer/Provider may use other means to detect the presence of a Port Partner before applying the method described above.

#### 4.1.1.3 Provider/Consumer Operational Details

In operation (see Figure 4-1), the Provider/Consumer with a Dead Battery shall begin (as it must if its battery is truly dead) by outputting nothing on  $V_{BUS}$ . When power is applied to  $V_{BUS}$  (i.e. back powered),  $V_{BUS}$  is used by the Provider/Consumer to power its transmitter and to output the Bit Stream within the limited power offered by *vSafeDB*.

The Provider/Consumer shall begin outputting the Bit Stream on  $V_{BUS}$  within *tSendBitStream*. This is necessary because the application of full  $V_{BUS}$  power is dependent on the time it takes the Consumer/Provider to detect the presence of activity on  $V_{BUS}$  and to adjust its power supply.

The Provider/Consumer may have additional circuitry that requires more power than is available from  $V_{BUS}$  when powered by *vSafeDB*. The Provider/Consumer Port can assume that full *vSafe5V* power is available *tWaitForPower* after it started sending the Bit Stream. After *tWaitForPower*, *vSafe5V* will be available to the Provider/Consumer to bring up any remaining logic required to process PD Messages on  $V_{BUS}$ . When this logic is ready, the Provider/Consumer Port shall signal its Port Partner that is ready to operate as Consumer by stopping the Bit Stream.

#### 4.1.1.4 Sequence of operation

Figure 4-1 illustrates the flow for Dead Battery/Unpowered Port detection for both Provider/Consumer ports and Consumer/Provider ports. To ensure consistent behavior all Consumer/Provider ports shall have the ability to detect a Provider/Consumer with a Dead Battery or Unpowered Port.



Table 4-1 Normal Dead Battery Operation

Step	Provider/Consumer	Consumer/Provider
1	The Provider/Consumer shall not drive $V_{BUS}$ , allowing it to remain at <i>vSafe0V</i> .	The Consumer/Provider shall start <i>DBDetectTimer</i> , used to determine when next to apply <i>vSafeDB</i> .
2	While <i>vSafeDB</i> is not present, the Provider/Consumer shall: <ul style="list-style-type: none"> <li>• If willing to act as a Source, go and do this. (The first action will be to output <i>vSafe5V</i>).</li> <li>• If it does not wish to be powered, remain in this step.</li> </ul>	Until the <i>DBDetectTimer</i> expires, the Consumer/Provider shall check whether $V_{BUS}$ is above <i>vSafe0V</i> . If it is, it shall start to operate as a Sink, and leave the Dead Battery Operation procedure.
3		When the <i>DBDetectTimer</i> expires, the Consumer/Provider shall start to output <i>vSafeDB</i> , in order to power the Provider/Consumer Bit Stream generation circuitry. The process of preparing this may take up to <i>tTurnOnSafeDB</i> . The Consumer/Provider shall then start the <i>BitStreamDetectTimer</i> , which determines how long to wait for the Bit Stream. The Bit Stream detector shall see at least 128 alternating '0' and '1' bits before deciding that the Bit Stream is present.
4		If the <i>BitStreamDetectTimer</i> expires before the Bit Stream is detected, the Consumer/Provider shall perform the following procedure. <b>Apply vSafe0V Procedure</b> The Consumer/Provider shall attempt to discharge the maximum permitted capacitance (this can theoretically be as much as <i>cSrcBulk</i> max or <i>cSrcBulkShared</i> max) on $V_{BUS}$ at a rate which will reduce its voltage to <i>vSafe0V</i> within <i>tDBDischargeVbus</i> max bearing in mind that $V_{BUS}$ may be being driven with <i>vSafe5V</i> and not capable of being discharged. After <i>tDBDischargeVbus</i> max the Consumer/Provider shall stop attempting to discharge $V_{BUS}$ . After a time of <i>tDBSourceOff</i> from starting to discharge $V_{BUS}$ , then the Consumer/Provider shall go back to step 1.
5	If the Provider/Consumer wishes to be powered, and <i>vSafeDB</i> is present, it shall start to output Bit Stream within <i>tSendBitStream</i> of <i>vSafeDB</i> being available, and start the <i>WaitForPowerTimer</i> . When budgeting for <i>tSendBitStream</i> the Provider/Consumer shall allow for: <ul style="list-style-type: none"> <li>• The time required for <i>vSafeDB</i> to charge the capacitance presented by itself to a level at which it can begin operation.</li> <li>• The time required for enough of its own circuitry to power up to transmit a Bit Stream.</li> <li>• The fact that <i>tTurnOnImpliedSink</i> is included in this budget</li> </ul> Note that the effect of $V_{BUS}$ capacitance on these timings is significant. The designer should verify their own design in this respect.	

Step	Provider/Consumer	Consumer/Provider
6		If the Bit Stream is detected, the Consumer/Provider shall output <i>vSafe5V</i> to $V_{BUS}$ and start the <i>DeviceReadyTimer</i> . This timer is used to ensure that the Provider/Consumer is ready to operate as a Sink within a reasonable time.
7	When the <i>WaitForPowerTimer</i> expires, the Provider/Consumer assumes that <i>vSafe5V</i> is present on $V_{BUS}$ . It shall then bring up the full PD system and when ready to receive and process <i>Source_Capabilities</i> Messages, stop sending the Bit Stream.	If the <i>DeviceReadyTimer</i> expires, the Consumer/Provider shall perform the <i>Apply vSafe0V Procedure</i> defined in step 4 above, ending by going to step 1.
8	When the <i>Source_Capabilities</i> Message is received, the Provider/Consumer shall determine the connected cable type, and start operation as a Sink.	If the Bit Stream is detected to have stopped, the Consumer/Provider shall go and start operating as a Source. The first action will be to send the <i>Source_Capabilities</i> Message.
<b>Note:</b> If at any time the Provider/Consumer sees <i>vSafe0V</i> on $V_{BUS}$ , it shall ensure that it is not sending Bit Stream within <i>tBitStreamOff</i> , and go to step 1.		

#### 4.1.2 Type-C to Type-C Dead Battery Operation

Dead Battery charging operation for connections between Type-C connectors is defined in [\[USBType-C 1.0\]](#).

### 4.2 Cable IR Ground Drop (IR Drop)

Every PD Sink Port capable of USB communications may be susceptible to unreliable USB communication if the voltage drop across ground falls outside of the acceptable common mode range for the USB Hi-Speed transceivers data lines ( $V_{GND\_DROP}$ ) due to excessive current draw. Certified USB cabling is specified such that such errors should not occur (See Section 3.6.10).

### 4.3 A-Plug Insertion Detect

The USB Power Delivery specification defines an Insertion Detect mechanism for the Standard-A plug. It consists of a contact that connects with the shield when a Standard-A plug or a PD Standard-A plug is inserted into the receptacle. The shield is essentially connected to ground through at most *rID* max.

The Micro-A plug's ID pin is used for Insertion Detect. It is essentially connected to ground through at most a 1K $\Omega$  resistance. This is the default defined in the Micro-A connector specification for [\[USB 2.0\]](#) and [\[USB 3.1\]](#). See Section 3.4.2 for more details.

The Insertion Detect feature:

- When a plug is present for Cold Socket applications shall be used to indicate when to apply power to  $V_{BUS}$
- Should be used to indicate to the PD logic to start sending *Source\_Capabilities* Messages when a plug is present.
- Should be used to indicate to the PD logic to put  $V_{BUS}$  back to USB Default Operation when the plug is removed.

The Insertion Detect feature for Standard-A receptacles shall be present for Cold Socket but is optional for all other Standard-A applications.

### 4.4 Cable Type Detection

#### 4.4.1 Detecting Cabling Capabilities

Non-compliant cables, such as 'Y' and 'W' cables create the potential to damage hardware were the PD system to allow more than *vSafe5V* to be placed on  $V_{BUS}$ . To prevent this, all PD A-plug to B-plug assemblies are made in a way that can be electronically detected. Only A-plug to B-plug assemblies that are marked for PD shall be used for voltages

higher than *vSafe5V* or current levels higher than 1.5A. Standard Type-C cable assemblies are rated for PD voltages higher than *vSafe5V* and current levels of at least 3A.

Cable type detection is a multi-step process that both the Source and Sink perform. This section provides flow of the cable type detection based on the electronic markings defined in Section 3.4. The Source shall limit maximum capabilities it offers so as not to exceed the capabilities of the type of plug detected. Requests made by the Sink shall not exceed the capabilities of the type of plug.

The cable detection process is usually run when the Source is powered up, after a Power Role Swap or when power is applied to a Sink. The exact method used to detect these events is up to the manufacturer and shall meet the following requirements:

- Sources shall run the cable detection process prior to the Source sending *Source\_Capabilities* Messages offering voltages in excess of *vSafe5V* or currents in excess of:
  - 1.5A for Type-A/Type-B
  - 3A for Type-C.
- Sinks with Type-A and Type-B connectors shall run the cable detection process prior to sending any *Request* Messages.
- Sinks with Type-C connectors shall select Capabilities from the offered Source Capabilities assuming that the Source has already determined the Capabilities of the cable.
- Provider/Consumers with dead batteries shall wait until after receipt of the first *Source\_Capabilities* Message before running the cable detection process and making a request for power.

Sources shall detect the type of attached cable and either limit the Capabilities they offer or operate in a Low Power mode based on:

- The receptacle type (i.e. Standard, Micro or Type-C) and its known current carrying capability.
- The current carrying capability of the plug determined by:
  - The plug type (i.e. Type-C, PD, non-PD or Low Power).
  - Cable capabilities determined using Structured VDM Messages (see Section 6.4.4.2) sent using SOP' Communication (see Section 2.4).

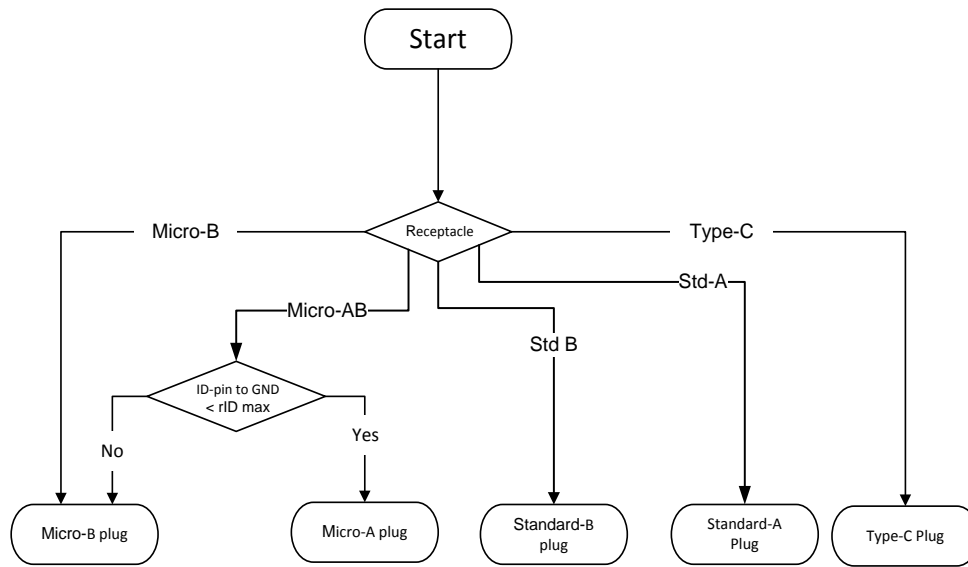
Sinks, except those with Type-C receptacles, shall detect the type of attached cable and limit their requests based on:

- The receptacle type (i.e. Standard, or Micro) and its known current carrying capability.
- The current carrying capability of the plug determined by:
  - The plug type (i.e. PD or non-PD).

#### 4.4.2 Plug Type Determination

Figure 4-2 shows the flow for the first portion of the multi-step process that shall be used to determine the kind of cable. It begins by determining the kind of plug inserted into the receptacle. Specifically the plug type: Standard-A, Standard-B, Micro-A, Micro-B or Type-C, is determined. Detection of Type-C plugs is defined in [\[USBType-C 1.0\]](#).

Figure 4-2 Plug Type Determination



#### 4.4.3 Detecting the PD Capabilities of the Standard-A Connector

The PD version of the Standard-A receptacle has one or two additional contacts that are used to:

- Optional detect if Standard-A plug is inserted in or removed from the receptacle.
- Required detect if the Standard-A plug is PD Capable or not.

The following description assumes that the two switches in the Standard-A receptacle are detected by having pull-up resistors to a positive voltage and looking at the voltage, so a high voltage indicates no connection in the switch and a low voltage indicates a connection in the switch. Other detection circuits may be used.

Figure 4-3 Standard-A Plug PD Capabilities Flow

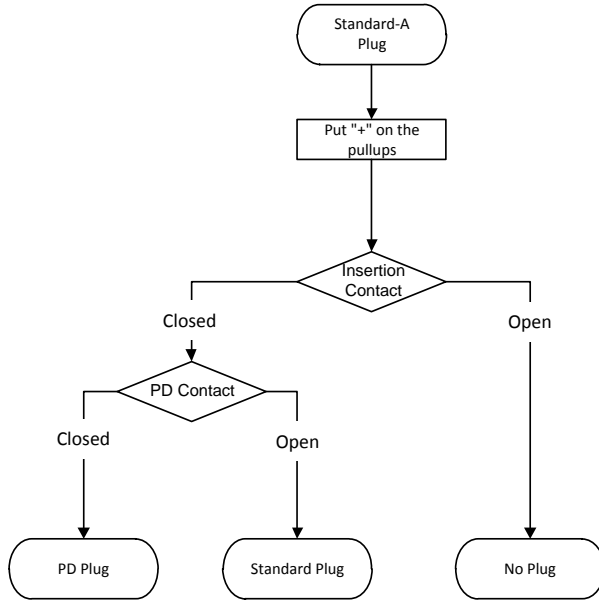
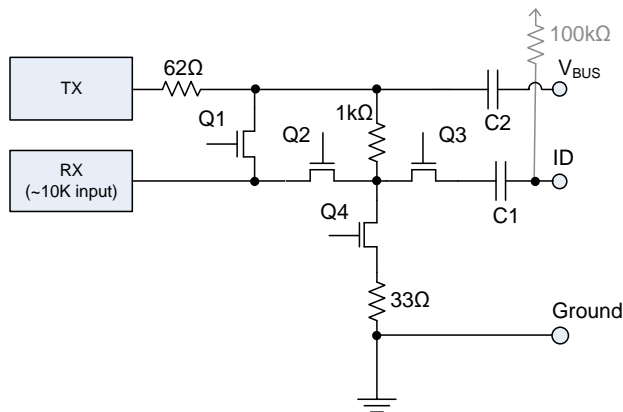


Figure 4-3 illustrates how the detection contacts in the A receptacle shall be used. Insertion Detect is an optional feature (see Section 3.1.5). When not present, the path through the Figure 4-3 follows the Insertion Detect 'Closed' arc.

#### 4.4.4 Plug Type Detection except Standard-A

Figure 4-4 Plug Type Detection Circuit



The example circuit shown in Figure 4-4 is used to detect the electronic markings on the ID pin indicating the type of Micro connector.

The TX is used to put a carrier signal on the  $V_{BUS}$  line and the RX is used to detect whether a signal is present or not (typically the Squelch is used for this purpose). Q1 - Q4 are used to create a series of circuits where the voltage output (as measured by the RX) in each step is used to determine the configuration of the plug in accordance with Table 4-2.

It is important that B-Plug type detection takes place when the line is idle and connected at the remote end. The ideal measurement opportunity is after sending the *GoodCRC* Message in response to the *Source\_Capabilities* Message and before sending the *Request* Message as the line can be expected to be connected and idle because the sink will be the next to transmit.

Note: If the far-end is not terminated or not terminated correctly (see Section 5.8.2.5) then the cable-type detection described below may give erroneous results if signal levels at the cable input are reduced excessively by the reflections of the cable.

In normal operation Q1 is conducting (turned ON) and Q2, Q3 and Q4 are not conducting (turned OFF).

In order to check the plug type using the circuit in Figure 4-4, a series of steps are performed; the result of each step is recorded as a "0" or "1". The steps are:

- 1) Q1, Q3 and Q4 not conducting (turned OFF)
- 2) Q2 conducting (turned ON)
- 3) Check Squelch -> open - "1", else "0" -> bit 1
- 4) Q3 conducting (turned ON)
- 5) Check Squelch -> open - "1", else "0" -> bit 2
- 6) Q4 conducting (turned ON)
- 7) Check Squelch -> open - "1", else "0" -> bit 3

Table 4-2 summarizes the results.

Table 4-2 Plug Type Determination

bit 1	bit 2	bit 3	Micro-A plug	Micro-B or PD Standard-B plug	Approximate level at RX when detecting bit 1	Approximate level at RX when detecting bit 2	Approximate level at RX when detecting bit 3
1	1	1	Low Power	PD (5A)	~ 0dB	~ 0dB	~ -9dB
1	1	0	PD	Legacy <sup>1</sup>	~ 0dB	~ -6dB (PD) ~ 0dB (Legacy)	~ -30dB
1	0	1	Fault	Fault			
1	0	0	Legacy	PD (3A)	~ 0dB	~ -40dB	~ -40dB
0	1	1	Fault	Fault			
0	1	0	Fault	Fault			
0	0	1	Fault	Fault			
0	0	0	Fault	Fault			

Note 1: A legacy Standard-B plug does not have an ID pin but will be detected as legacy via the ID pin in the receptacle..

## 4.5 Low Power Devices using Micro-A Plug

The Low Power feature enables Sources to minimize the power they output over  $V_{BUS}$  without using the PD negotiation process. These devices utilize a captive cable ending in a Micro-A plug.

Sources that are able to detect the Low Power plug shall automatically begin an Implicit Contract equivalent to a Battery PDO (Max Voltage = *vLowPower* max, Min Voltage = *vLowPower* min, Max Power = *pLowPower* nom) and shall not transmit or respond to PD Messaging or Signaling while the Low Power plug is connected. Low Power Devices (Sinks) shall be able to operate normally when powered by any voltage in the range *vLowPower* and shall not transmit USB PD Messaging or Signaling.

The entry process into Low Power operation is as follows:



1. The Source detects a Low Power Device is attached (see Table 4-2).
2. The Port Pair then operates on an Implicit Contract:
  - a. The Source supplies a voltage in the range *vLowPower* and at least *pLowPower* nominal
  - b. The Sink shall be able to operate on a voltage on the range *vLowPower* and shall draw no more than *pLowPower* nominal.
3. The Source may monitor the voltage and when it falls below *vLowPower* min reapply *vSafe5V* in an attempt to continue operating. Alternatively, the Source may not monitor  $V_{BUS}$  and when it falls to this level; both it and the Low Power Device will likely fail for lack of power.

Exit from Low Power operation occurs when the Low Power plug is detached.

Sources that are not able to detect the Low Power plug shall treat the plug as a PD micro-A plug. These sources would therefore supply *vSafe5V* constantly to a Low Power Device (Sink).

## 4.6 Electrical Parameters

Table 4-3 shows the parameters used in this section.

Table 4-3 Electrical Parameters

Parameter	Minimum	Nominal	Maximum	Units	Section
<i>pLowPower</i>		250		mW	4.5
<i>tBitStreamDetect</i>	100		300	ms	4.1
<i>tBitStreamOff</i>			100	ms	4.1
<i>tDBDetect</i>		10	15	s	4.1
<i>tDBDischargeVbus</i>			90	ms	4.1
<i>tDBSourceOff</i>		200	220	ms	4.1
<i>tDeviceReady</i>	60		90	s	4.1
<i>tSendBitStream</i>			95	ms	4.1
<i>tWaitForPower</i>	20			ms	4.1.1
<i>vLowPower</i>	2.5		<i>vSafe5V</i> max	V	4.5

Table 4-4 Electrical Timers

Timer	Parameter	Used By	Section
<i>BitStreamDetectTimer</i>	<i>tBitStreamDetect</i>	PolicyEngine	4.1, 8.3.3.6.1.5
<i>DBDetectTimer</i>	<i>tDBDetect</i>	PolicyEngine	4.1, 8.3.3.6.1.5
<i>DBSourceOffTimer</i>	<i>tDBSourceOff</i>	PolicyEngine	4.1, 8.3.3.6.1.5
<i>DeviceReadyTimer</i>	<i>tDeviceReady</i>	PolicyEngine	4.1, 8.3.3.6.1.5
<i>WaitForPowerTimer</i>	<i>tWaitForPower</i>	Policy Engine	4.1.1, 8.3.3.6.1.6

## 5. Physical Layer

### 5.1 Physical Layer Overview

The Physical Layer (PHY Layer) defines the signaling technology for USB Power Delivery. This chapter defines the electrical requirements and parameters of the PD Physical Layer required for interoperability between USB PD devices.

### 5.2 Physical Layer Functions

The USB PD Physical Layer consists of a pair of transmitters and receivers that communicate across a single signal wire ( $V_{BUS}$  or CC). All communication is half duplex. The PHY Layer practices collision avoidance to minimize communication errors on the channel.

The transmitter performs the following functions:

- Receive packet data from the protocol layer
- Calculate and append a CRC
- Encode the packet data including the CRC (i.e. the payload)
- Transmit the Packet (Preamble, *SOP\**, payload, CRC and *EOP*) across the channel using either
  - A Binary Frequency Shift Keyed (BFSK) modulated carrier over  $V_{BUS}$  or
  - Biphase Mark Coding (BMC) over CC

The receiver performs the following functions:

- For BFSK detect the modulated carrier from the channel
- Recover the clock and lock onto the Packet from the Preamble
- Detect the *SOP\**
- Decode the received data including the CRC
- Detect the *EOP* and validate the CRC
  - If the CRC is valid, deliver the packet data to the protocol layer.
  - If the CRC is not valid, flush the received data.

### 5.3 Symbol Encoding

Except for the Preamble, all communications on the line shall be encoded with a line code to ensure a reasonable level of DC-balance and a suitable number of transitions. This encoding makes receiver design less complicated and allows for more variations in the receiver design.

4b5b line code shall be used. This encodes 4-bit data to 5-bit symbols for transmission and decodes 5-bit symbols to 4-bit data for consumption by the receiver.

The 4b5b code provides data encoding along with special symbols. Special symbols are used to signal *Hard Reset*, and delineate packet boundaries.

Table 5-1 4b5b Symbol Encoding Table

Name	4b	5b Symbol	Description
0	0000	11110	hex data 0
1	0001	01001	hex data 1
2	0010	10100	hex data 2
3	0011	10101	hex data 3
4	0100	01010	hex data 4
5	0101	01011	hex data 5
6	0110	01110	hex data 6
7	0111	01111	hex data 7
8	1000	10010	hex data 8
9	1001	10011	hex data 9
A	1010	10110	hex data A
B	1011	10111	hex data B
C	1100	11010	hex data C
D	1101	11011	hex data D
E	1110	11100	hex data E
F	1111	11101	hex data F
<i>Sync-1</i>	K-code	11000	Startsynch #1
<i>Sync-2</i>	K-code	10001	Startsynch #2
<i>RST-1</i>	K-code	00111	Hard Reset #1
<i>RST-2</i>	K-code	11001	Hard Reset #2
<i>EOP</i>	K-code	01101	EOP End Of Packet
Reserved	Error	00000	<del>Do Not Use</del> Shall not be used
Reserved	Error	00001	<del>Do Not Use</del> Shall not be used
Reserved	Error	00010	<del>Do Not Use</del> Shall not be used
Reserved	Error	00011	<del>Do Not Use</del> Shall not be used
Reserved	Error	00100	<del>Do Not Use</del> Shall not be used
Reserved	Error	00101	<del>Do Not Use</del> Shall not be used
<i>Sync-3</i>	K-code	00110	Startsynch #3
Reserved	Error	01000	<del>Do Not Use</del> Shall not be used
Reserved	Error	01100	<del>Do Not Use</del> Shall not be used

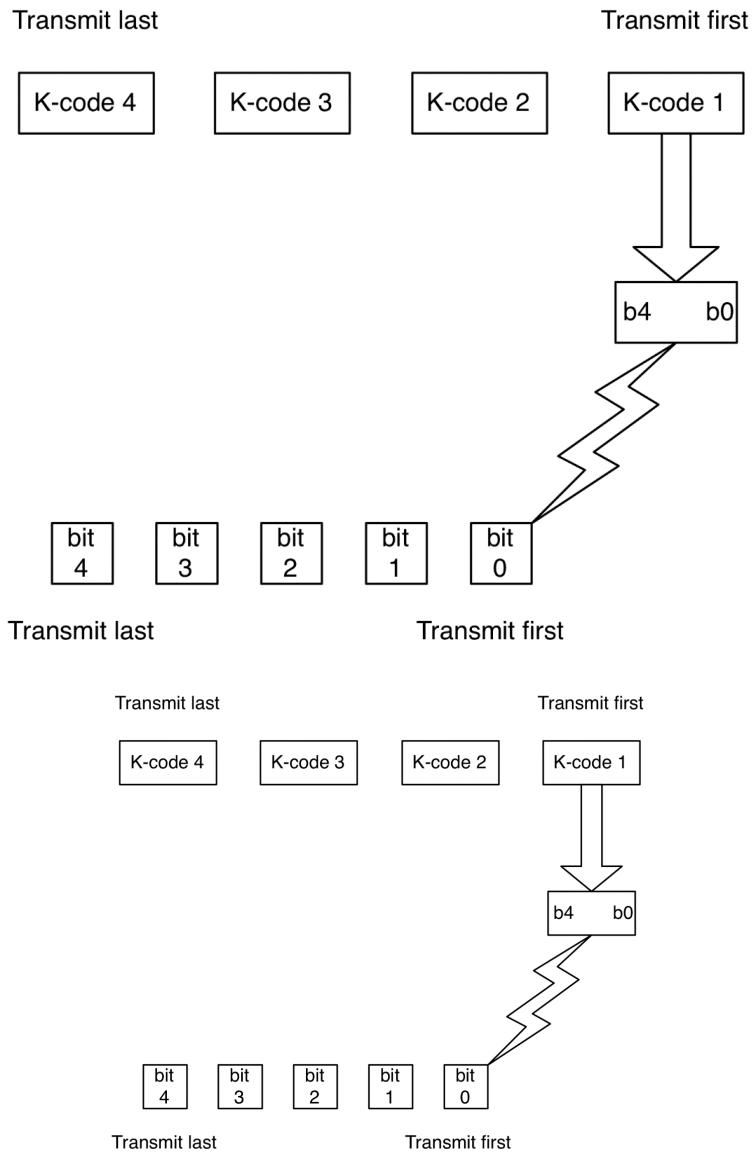
Name	4b	5b Symbol	Description
Reserved	Error	10000	<del>Do Not Use</del> Shall not be used
Reserved	Error	11111	<del>Do Not Use</del> Shall not be used

## 5.4 Ordered Sets

Ordered sets shall be interpreted according to Figure 5-1.

An ordered set consists of 4 K-codes sent as shown in Figure 5-1.

Figure 5-1 Interpretation of ordered sets



A list of the ordered sets used by USB Power Delivery can be seen in Table 5-2. *SOP\** is a generic term used in place of *SOP/SOP'/SOP''*.

**Table 5-2 Ordered Sets**

Ordered Set	Section
<i>SOP</i>	5.6.1.2.1
<i>SOP'</i>	5.6.1.2.2
<i>SOP''</i>	5.6.1.2.3
<i>Hard Reset</i>	5.6.4
<i>Cable Reset</i>	5.6.5
<i>SOP'_Debug</i>	5.6.1.2.4
<i>SOP''_Debug</i>	5.6.1.2.5

The receiver shall search for all four K-codes and when it finds at least three in the correct place, it may interpret it as a valid ordered set (see Table 5-3).

**Table 5-3 Validation of Ordered Sets**

	1st code	2nd code	3rd code	4th code
Valid	Corrupt	K-code	K-code	K-code
Valid	K-code	Corrupt	K-code	K-code
Valid	K-code	K-code	Corrupt	K-code
Valid	K-code	K-code	K-code	Corrupt
Valid (perfect)	K-code	K-code	K-code	K-code
Not Valid (example)	K-code	Corrupt	K-code	Corrupt

## 5.5 Transmitted Bit Ordering

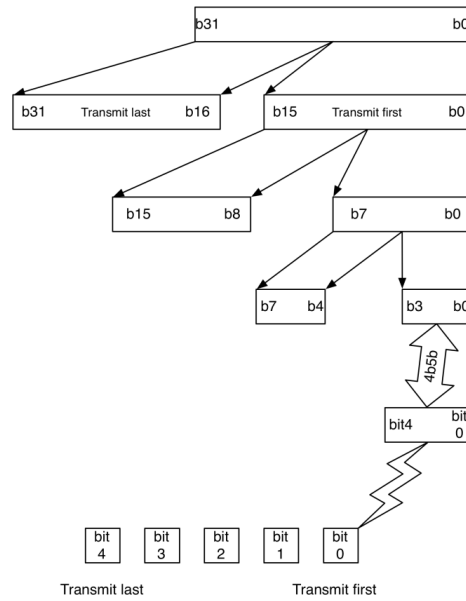
This section describes the order of bits on the wire that shall be used when transmitting data of varying sizes. Table 5-4 shows the different data sizes that are possible.

Figure 5-2 shows the transmission order that shall be followed.

**Table 5-4 Data Size**

	Unencoded	Encoded
Byte	8-bits	10-bits
Word	16-bits	20-bits
DWord	32-bits	40-bits

Figure 5-2 Transmit Order for Various Sizes of Data

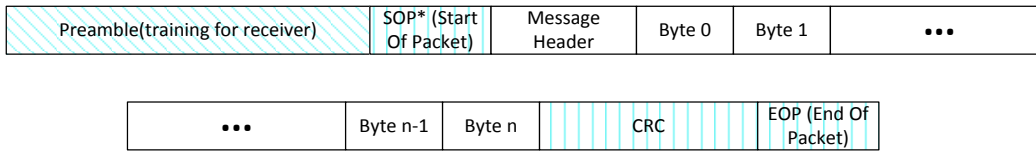




## 5.6 Packet Format

The packet format shall consist of a Preamble, an *SOP\**, (see Section 5.6.1.2), packet data including [the Message Header](#), a CRC and an *EOP* (see Section 5.6.1.5). The packet format is shown in Figure 5-3 and indicates which parts of the packet shall be 4b/5b encoded. Once 4b/5b encoded, the entire Packet shall be transmitted either using BFSK over  $V_{BUS}$  or BMC over CC. Note that when using BMC the Preamble is BMC encoded.

Figure 5-3 USB Power Delivery Packet Format



LEGEND:

Training sequence provided by the Physical layer, <b>not</b> encoded with 4b/5b	Provided by the Physical layer, encoded with 4b/5b	Provided by the Protocol layer, encoded with 4b/5b
---	--	--

### 5.6.1 Packet Framing

The transmission starts with a Preamble that is used to allow the receiver to lock onto the carrier. It is followed by a *SOP\** (Start of Packet). The packet is terminated with an *EOP* (End of Packet) K-code.

#### 5.6.1.1 Preamble

The Preamble is used to achieve lock in the receiver by presenting an alternating series of "0s" and "1s", so the average frequency is the carrier frequency. Unlike the rest of the packet, the Preamble shall not be 4b/5b encoded.

The Preamble shall consist of a 64-bit sequence of alternating 0s and 1s. The Preamble shall start with a "0" and shall end with a "1".

#### 5.6.1.2 Start of Packet Sequences

##### 5.6.1.2.1 Start of Packet Sequence (SOP)

*SOP* is an ordered set. The *SOP* ordered set is defined as: three *Sync-1* K-codes followed by one *Sync-2* K-code (see Table 5-5).

Table 5-5 SOP ordered set

K-code number	K-code in code table
1	<i>Sync-1</i>
2	<i>Sync-1</i>
3	<i>Sync-1</i>
4	<i>Sync-2</i>

A Power Delivery Capable Provider, Provider/Consumer, Consumer or Consumer/Provider shall be able to detect and communicate with packets using *SOP*. If a valid *SOP* is not detected (see Table 5-3) then the whole transmission shall be ~~ignored~~**Discarded**.

Sending and receiving of SOP Packets shall be limited to PD Capable DFPs and UFPs only (i.e. PD Capable Ports on PDUSB Hosts and PDUSB Devices). Cable Plugs shall neither send nor receive SOP Packets. Note that PDUSB Devices, even if they have the physical form of a cable (e.g. AMAs), are still ~~limited~~**required to respond** to SOP Packets ~~only~~.

#### 5.6.1.2.2 Start of Packet Sequence Prime (SOP')

The **SOP'** ordered set is defined as: two **Sync-1** K-codes followed by two **Sync-3** K-codes (see Table 5-6).

Table 5-6 SOP' ordered set

K-code number	K-code in code table
1	<b>Sync-1</b>
2	<b>Sync-1</b>
3	<b>Sync-3</b>
4	<b>Sync-3</b>

A Cable Plug capable of SOP' Communications shall only detect and communicate with packets starting with **SOP'**.

A DFP or Source needing to communicate with a Cable Plug capable of SOP' Communications, attached between a Port Pair will be able to communicate using both packets starting with **SOP'** to communicate with the Cable Plug and starting with **SOP** to communicate with its Port Partner. The DFP or Source shall co-ordinate SOP and SOP' Communication so as to avoid collisions.

For a Cable Plug supporting SOP' Communications, if a valid **SOP'** is not detected (see Table 5-3) then the whole transmission shall be **ignoredDiscarded**. For the DFP or Source supporting SOP' Communications if a valid **SOP** or **SOP'** is not detected (see Table 5-3) then the whole transmission shall be **ignoredDiscarded**. When there is an Explicit Contract in place a UFP shall not send SOP' Packets and shall **ignoreDiscard** all packets starting with **SOP'**. When there is no Explicit Contract or an Implicit Contract in place a Sink shall not send SOP' Packets and shall **ignoreDiscard** all packets starting with **SOP'**.

#### 5.6.1.2.3 Start of Packet Sequence Double Prime (SOP'')

The **SOP''** ordered set is defined as the following sequence of K-codes: **Sync-1, Sync-3, Sync-1, Sync-3** (see Table 5-7).

Table 5-7 SOP'' ordered set

K-code number	K-code in code table
1	<b>Sync-1</b>
2	<b>Sync-3</b>
3	<b>Sync-1</b>
4	<b>Sync-3</b>

A Cable Plug capable of SOP'' Communication, shall have a SOP' Communication capability in the other Cable Plug. No cable shall only support SOP'' Communication. A Cable Plug to which SOP'' Communication is assigned shall only detect and communicate with packets starting with **SOP''** and shall **ignoreDiscard** any other packets.

A DFP needing to communicate with such a Cable Plug, attached between a Port Pair will be able to communicate using packets starting with **SOP'** and **SOP''** to communicate with the Cable Plugs and packets starting with **SOP** to communicate with its Port Partner. A DFP which supports SOP'' Communication shall also support SOP' Communication and shall co-ordinate SOP\* Communication so as to avoid collisions.

For the Cable Plug supporting SOP'' Communication, if a valid **SOP''** is not detected (see Table 5-3) then the whole transmission shall be **ignoredDiscarded**. For the DFP if a valid **SOP\*** is not detected (see Table 5-3) then the whole transmission shall be **ignoredDiscarded**. A UFP shall not send SOP'' Packets and shall **ignoreDiscard** all Packets starting with **SOP''**.

#### 5.6.1.2.4 Start of Packet Sequence Prime Debug (SOP'\_Debug)

The **SOP'\_Debug** ordered set is defined as the following sequence of K-codes: **Sync-1, RST-2, RST-2, Sync-3** (see Table 5-8). The usage of this Ordered Set is presently undefined.

Table 5-8 SOP'\_Debug ordered set

K-code number	K-code in code table
1	<i>Sync-1</i>
2	<i>RST-2</i>
3	<i>RST-2</i>
4	<i>Sync-3</i>

#### 5.6.1.2.5 Start of Packet Sequence Double Prime Debug (SOP''\_Debug)

The *SOP''\_Debug* ordered set is defined as the following sequence of K-codes: *Sync-1, RST-2, Sync-3, Sync-2* (see Table 5-9). The usage of this Ordered Set is presently undefined.

Table 5-9 SOP''\_Debug ordered set

K-code number	K-code in code table
1	<i>Sync-1</i>
2	<i>RST-2</i>
3	<i>Sync-3</i>
4	<i>Sync-2</i>

#### 5.6.1.3 Packet Payload

The packet payload is delivered from the protocol layer (Section 6.2) and shall be encoded with the hex data codes from Table 5-1.

#### 5.6.1.4 CRC

The CRC shall be inserted just after the payload. It is described in Section 5.6.2.

#### 5.6.1.5 End of Packet (EOP)

The end of packet marker shall be a single *EOP* K-code as defined in Table 5-1. This shall mark the end of the CRC. After the *EOP*, the CRC-residual shall be checked. If the CRC is not good, the whole transmission shall be discarded, if it is good, the packet shall be delivered to the Protocol Layer. Note an *EOP* may be used to prematurely terminate a Packet e.g. before sending *Hard Reset* Signaling.

### 5.6.2 CRC

The ~~packet~~Message Header and data shall be protected by a 32-bit CRC.

CRC-32 protects the data integrity of the data payload. CRC-32 is defined as follows:

- The CRC-32 polynomial shall be = 04C1 1DB7h.
- The CRC-32 Initial value shall be = FFFF FFFFh.
- CRC-32 shall be calculated for all bytes of the payload not inclusive of any packet framing symbols (i.e. excludes the Preamble, *SOP\**, *EOP*).
- CRC-32 calculation shall begin at byte 0 bit 0 and continue to bit 7 of each of the bytes of the packet.
- The remainder of CRC-32 shall be complemented.
- The residual of CRC-32 shall be C704 DD7Bh.

Note: This inversion of the CRC-32 remainder adds an offset of FFFF FFFFh that will create a constant CRC-32 residual of C704 DD7Bh at the receiver side.

Note: The CRC implementation is identical to the one used in [USB 3.1].

Figure 5-4 is an illustration of CRC-32 generation. The output bit ordering shall be as detailed in Table 5-10.

Figure 5-4 CRC 32 generation

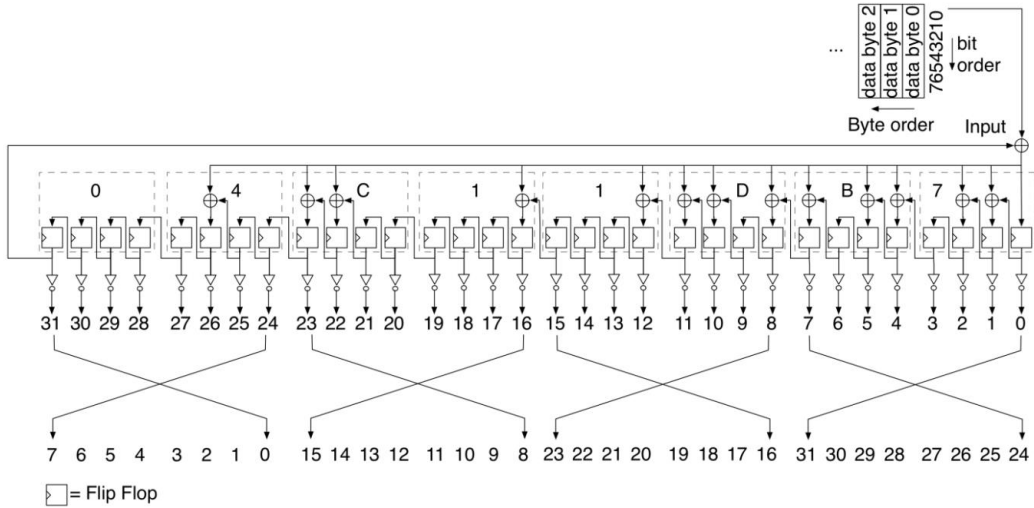


Table 5-10 CRC-32 Mapping

CRC-32	Result bit Position in CRC-32 Field
0	31
1	30
2	29
3	28
4	27
5	26
6	25
7	24
8	23
9	22
10	21
11	20
12	19
13	18
14	17
15	16
16	15
17	14
18	13
19	12
20	11
21	10
22	9
23	8

CRC-32	Result bit Position in CRC-32 Field
24	7
25	6
26	5
27	4
28	3
29	2
30	1
31	0

The CRC-32 shall be encoded before transmission.

### 5.6.3 Packet Detection Errors

CRC errors, or errors detected while decoding encoded symbols using the code table, shall be treated the same way; the Message shall be discarded and a *GoodCRC* Message shall not be returned.

While the receiver is processing a packet, if at any time  $V_{BUS}$  becomes idle the receiver shall stop processing the packet and discard it (no *GoodCRC* Message is returned). See Section 5.8.2.6.4 for the definition of BFSK idle and Section 5.8.3.6.1 for the definition of BMC idle.

### 5.6.4 Hard Reset

*Hard Reset* Signaling is an ordered set of bytes sent with the purpose to be recognized by the PHY Layer. The *Hard Reset* Signaling ordered set is defined as: three *RST-1* K-codes followed by one *RST-2* K-code (see Table 5-11).

Table 5-11 Hard Reset ordered set

K-code number	K-code in code table
1	<i>RST-1</i>
2	<i>RST-1</i>
3	<i>RST-1</i>
4	<i>RST-2</i>

A device shall perform a Hard Reset when it receives *Hard Reset* Signaling. After receiving the *Hard Reset* Signaling, the device shall reset as described in Section 6.7.2. If a valid *Hard Reset* is not detected (see Table 5-3) then the whole transmission shall be ~~ignored~~Discarded.

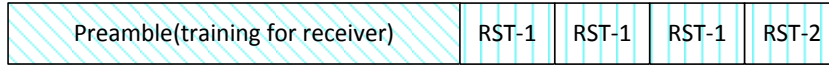
A Cable Plug shall perform a Hard Reset when it detects *Hard Reset* Signaling being sent between the Port Partners. After receiving the *Hard Reset* Signaling, the device shall reset as described in Section 6.7.2.

The procedure for sending *Hard Reset* Signaling shall be as follows:

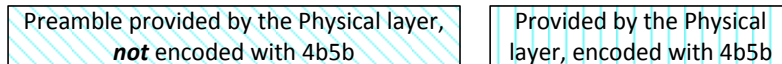
1. If the PHY Layer is currently sending a Message, the Message shall be interrupted by sending an *EOP* K-code and the rest of the Message discarded.
- ~~2.1. Wait *tInterFrameGap*.~~
- ~~2.2. If  $V_{BUS}$  is not idle, wait for it to become idle (see Section 5.8.2.6.4 for the definition of BFSK idle and Section 5.8.3.6.1 for the definition of BMC idle)~~
3. ~~Wait *tInterFrameGap*.~~
4. ~~If  $V_{BUS}$  is still idle~~ send the Preamble followed by the 4 K-codes for *Hard Reset* Signaling.
5. Disable the channel (i.e. stop sending and receiving), reset the PHY Layer and inform the Protocol Layer that the PHY Layer has been reset.
6. Re-enable the channel when requested by the Protocol Layer.

Figure 5-5 shows the line format of *Hard Reset* Signaling which is a Preamble followed by the *Hard Reset* Ordered Set.

Figure 5-5 Line format of Hard Reset



LEGEND:



### 5.6.5 Cable Reset

**Cable Reset** Signaling is an ordered set of bytes sent with the purpose to be recognized as an embedded **Control Message** to the PHY Layer. The **Cable Reset** Signaling ordered set is defined as the following sequence of K-codes: **RST-1, Sync-1, RST-1, Sync-3** (see Table 5-12).

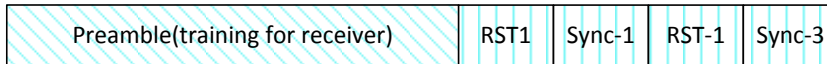
Table 5-12 Cable Reset ordered set

K-code number	K-code in code table
1	<b>RST-1</b>
2	<b>Sync-1</b>
3	<b>RST-1</b>
4	<b>Sync-3</b>

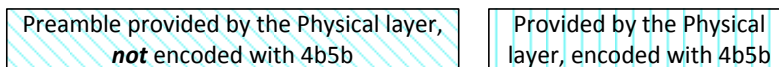
**Cable Reset Signaling shall only be sent by the DFP.** The **Cable Reset** Ordered Set is used to reset the Cable Plugs without the need to Hard Reset the Port Partners. The state of the Cable Plug after the **Cable Reset** Signaling shall be equivalent to power cycling the Cable Plug.

Figure 5-6 shows the line format of **Cable Reset** Signaling which is a Preamble followed by the **Cable Reset** Ordered Set.

Figure 5-6 Line format of Cable Reset



LEGEND:



## 5.7 Collision Avoidance

The PHY Layer shall monitor the channel for data transmission and only initiate transmissions when  $V_{BUS}$  or CC is idle. If the bus idle condition is present, it shall be considered safe to start a transmission provided the conditions detailed in Section 5.8.1.4 are met. The bus idle condition shall be checked immediately prior to transmission. If transmission cannot be initiated then the packet shall be discarded. If the packet is discarded because  $V_{BUS}$  or CC is not idle, the PHY Layer shall signal to the protocol layer that it has discarded the Message as soon as  $V_{BUS}$  or CC becomes idle. See Section 5.8.2.6.4 for the definition of idle  $V_{BUS}$  and Section 5.8.3.6.1 for the definition of idle CC.

## 5.8 Physical Layer Signaling Schemes

### 5.8.1 Common Signaling **Scheme** Specifications

This section defines receiver and transmitter requirements which are common across different signaling schemes.

#### 5.8.1.1 Common Signaling **Scheme** Parameters

The electrical requirements specified in Table 5-13 shall apply to both the transmitter and receiver.

Table 5-13 Common Normative Signaling **Scheme** Requirements

Parameter	Description	Min	Nom	Max	Units	Comment
<i>fBitRate</i>	Bit rate	270	300	330	Kbps	

#### 5.8.1.2 Common Transmitter Signaling **Scheme** Specifications

Table 5-14 Common Normative Signaling **Scheme** Requirements for Transmitter

Parameter	Description	Min	Nom	Max	Units	Comment
<i>pBitRate</i>	Maximum difference between the bit-rate during the part of the packet following the Preamble and the reference bit-rate.			0.25	%	The reference bit rate is the average bit rate of the last 32 bits of the Preamble.
<i>tInterFrameGap</i>	Time from the end of last bit of a Frame until the start of the first bit of the next Preamble.	25			μs	
<i>tStartDrive</i>	Time before the start of the first bit of the Preamble when the transmitter shall start driving the line.	-1		1	μs	

##### 5.8.1.2.1 Bit Rate Drift

Limits on the drift in *fBitRate* are set in order to help low-complexity receiver implementations.

*fBitRate* is the reciprocal of the average bit duration from the previous 32 bits at a given portion of the packet. The change in *fBitRate* during a packet shall be less than *pBitRate*. The reference bit rate (refBitRate) is the average *fBitRate* over the last 32 bits of the Preamble. *fBitRate* throughout the packet, including the **EOP**, shall be within *pBitRate* of refBitRate. *pBitRate* is expressed as a percentage:

$$pBitRate = | fBitRate - refBitRate | / refBitRate \times 100\%$$

The transmitter shall have the same *pBitRate* for all packet types. The **BIST Carrier Mode 2** and Bit Stream signals are continuous signals without a payload. When checking *pBitRate* any set of 1044 bits (20 bit **SOP** followed by 1024 PRBS bits) within a continuous signal may be considered as the part of the packet following the Preamble and the 32 preceding bits considered to be the last 32 bits of the Preamble used to compute refBitRate.

#### 5.8.1.3 Common Receiver Signaling **Scheme** Specifications

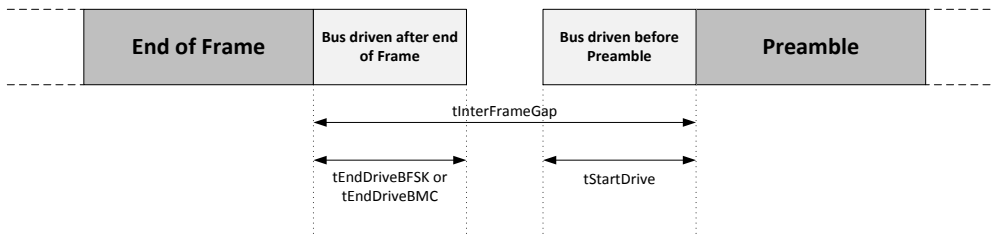
Table 5-15 Common Normative Signaling **Scheme** Requirements for Receiver

Parameter	Description	Min	Nom	Max	Units	Comment
<i>nBER</i>	Bit error rate, S/N = 25 dB			10 <sup>-6</sup>		

#### 5.8.1.4 Inter-Frame Gap

Figure 5-7 illustrates the inter-Frame gap timings.

Figure 5-7 Inter-Frame Gap Timings



The transmitter shall drive the bus for no longer than  $t_{\text{EndDriveBFSK}}$  or  $t_{\text{EndDriveBMC}}$  (as appropriate) after transmitting the final bit of the Frame. Detailed requirements for terminating the Frame and ceasing to drive the bus are given separately for BFSK and BMC.

Before starting to transmit the next Frame's Preamble the transmitter of the next Frame shall ensure that it waits for  $t_{\text{InterFrameGap}}$  after either:

1. Transmitting the previous frame, or
2. Receiving the previous frame, or
3. Observing an idle condition on  $V_{\text{BUS}}$  or CC (see Section 5.7).

Note: the transmitter is also required to verify a bus idle condition immediately prior to starting transmission of the next Frame.

The transmitter of the next Frame may vary the start of the Preamble by  $t_{\text{StartDrive}}$  (see Section 5.8.2.5.2).

See also Section 5.8.3.1 for figures detailing the timings relating to transmitting, receiving and observing idle in relating to Frames.



### 5.8.2 Binary Frequency Shift Keyed (BFSK) Signaling Scheme

The Binary Frequency Shift Keyed (BFSK) Signaling Scheme over  $V_{BUS}$  uses a carrier of  $f_{Carrier}$  modulated with the information to avoid the noise from the power supplies. Continuous Phase Binary Frequency Shift Keying (BFSK) shall be used to encode bits for transmission on the channel. In this specification, BFSK shall be understood to mean continuous phase BFSK. A signal of amplitude  $v_{TX}$  shall be injected onto  $V_{BUS}$  using a carrier frequency,  $f_{Carrier}$ . The following logic states shall be used:

- Logic 0 is indicated by a frequency  $f_{Carrier} - f_{Deviation}$ .
- Logic 1 is indicated by a frequency  $f_{Carrier} + f_{Deviation}$ .

The PHY Layer functions are shown in Figure 5-8, Figure 5-9, and Figure 5-10. The PHY Layer is expected to keep power consumption low, especially when only the squelch detector is required to be active. In the active mode, where any of the functions listed above may be executed, the PHY Layer Block power consumption should be minimized. In the squelch mode, when only the squelch detector is required, the power consumption should be minimized.

Figure 5-8 Transmitter Block Diagram

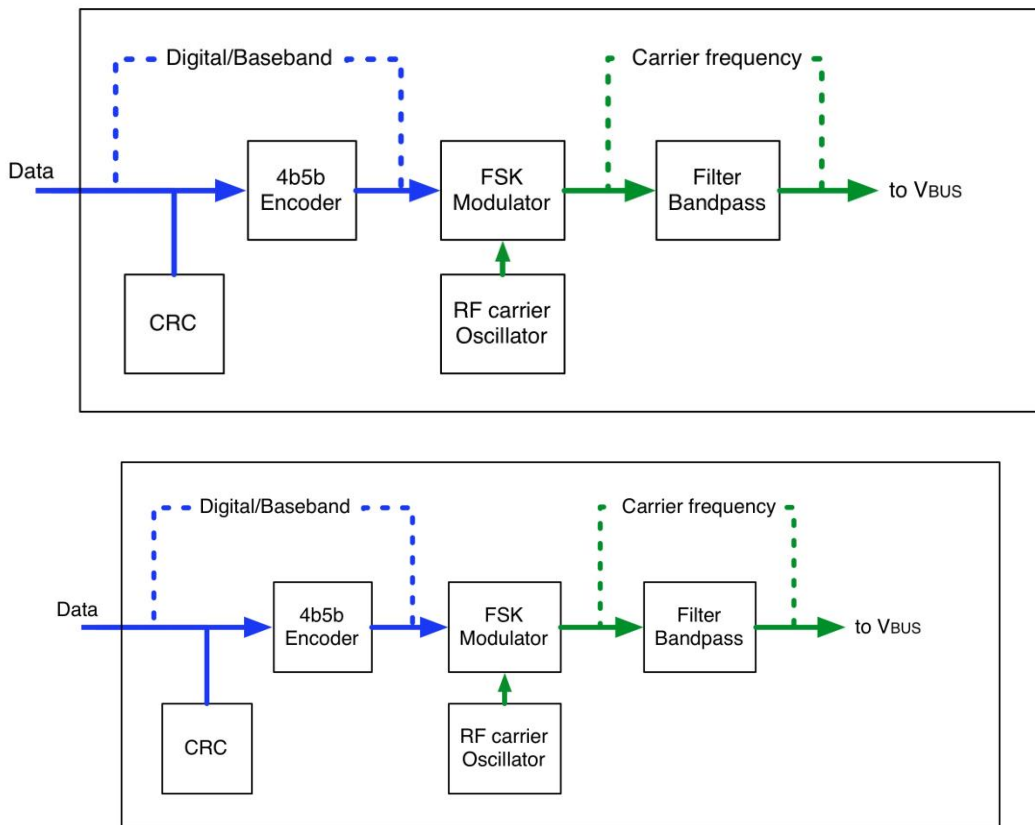


Figure 5-9 Receiver Block Diagram

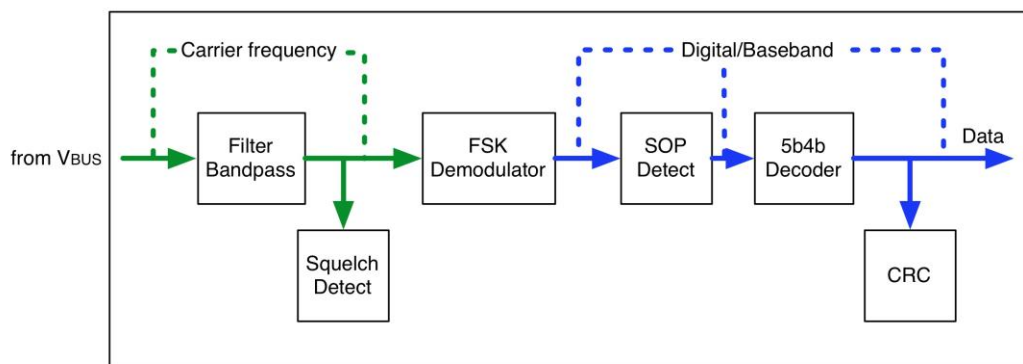
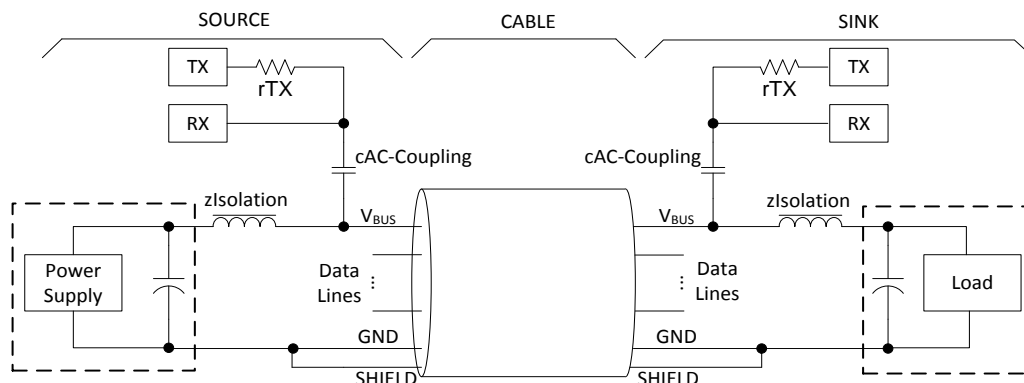


Figure 5-10 Channel Diagram (Cable Type Detection not shown)



### 5.8.2.1 Channel Overview

The channel connects two PHYs together via a single-ended serial signal. The PD signal uses USB  $V_{BUS}$  as the channel, and the system is designed to treat the  $V_{BUS}$  line in the cable as a transmission line terminated at both ends with matching impedance. The PHY Layer shall be AC coupled to  $V_{BUS}$  or be tolerant of the maximum possible DC voltage that may be present on  $V_{BUS}$ . For example, the PHY Layer may be AC coupled to  $V_{BUS}$  through a capacitor at each end. While transmitting, the Source, or Sink, shall apply an AC signal with amplitude  $v_{TX}$  to  $V_{BUS}$  as measured between  $z_{Isolation}$  and cAC-Coupling.

### 5.8.2.2 Transceiver Isolation Impedance

The Source and Sink shall place an isolation impedance between the  $V_{BUS}$  wire bulk capacitance and the  $V_{BUS}$  pin on the connector to allow the AC coupled USB Power Delivery transceiver to communicate over  $V_{BUS}$ .

The isolation impedance shall have an impedance of  $z_{Isolation}$  at any frequency within  $f_{Range}$ . See Sections 7.1.13.1 and 7.2.9.1 for additional detail.

Note: Isolation impedance surges may occur and need to be clamped in some manner. There are many variables that could enter into the nature of the surge that are not controlled by the spec and subject to implementation. The

clamping method is therefore an implementation choice. Implementers are strongly advised to read through Appendix-C to evaluate the need for this in their implementation.

### 5.8.2.3 Transceiver AC Coupling Capacitance

The Source and Sink shall be AC coupled to  $V_{BUS}$  or be tolerant of the maximum possible DC voltage that may be present on  $V_{BUS}$ . For example, the Source and Sink may be AC coupled to  $V_{BUS}$  by placing an AC coupling capacitance cAC-Coupling between the  $V_{BUS}$  pin on the connector and the transceiver as indicated in Figure 5-10. A suggested value for cAC-Coupling in this case would be between 3nF and 10nF.

### 5.8.2.4 BFSK Common Specifications

This section defines the common receiver and transmitter requirements.

#### 5.8.2.4.1 BFSK Common Parameters

The electrical requirements specified in Table 5-16 shall apply to both the transmitter and receiver.

Table 5-16 BFSK Common Normative Requirements

Parameter	Description	Min	Nom	Max	Units	Comment
<i>fCarrier</i>	BFSK carrier frequency	22.4	23.2	24.0	MHz	
<i>fDeviation</i>	BFSK frequency deviation	450	500	600	kHz	

Table 5-17 BFSK Transceiver Isolation Impedance Normative Requirements

Parameter	Description	Min	Nom	Max	Units	Comment
<i>zIsolation</i>	Impedance Allowed	80			$\Omega$	Measured at any frequency within <i>fRange</i> .
<i>fRange</i>	Range of frequencies used in communication	20.4		26	MHz	A spacing of 2MHz around <i>fCarrier</i> is allowed for the signal including FM tolerance and deviation.

One example implementation of a transceiver isolation impedance using the parameters listed in Table 5-17 is a 1 $\mu$ H inductor (see Appendix C).

### 5.8.2.5 BFSK Transmitter Specifications

Limits on the drift in *fCarrier*, and *fDeviation* are set in order to help low-complexity receiver implementations.

The transceiver shall terminate the  $V_{BUS}$  line with *rTX* while it is powered, whether it is transmitting, receiving or waiting for the squelch to close.

#### 5.8.2.5.1 BFSK Transmitter Requirements

The requirements specified in Table 5-18 shall apply to the transmitter.

Table 5-18 BFSK Transmitter Normative Requirements

Name	Description	Min	Nom	Max	Units	Comment
<i>pCarrierFreq</i>	Maximum difference between the carrier-frequency during the part of the packet following the Preamble and the reference carrier-frequency.			0.2	%	The reference carrier frequency is the average carrier frequency of the last 32 bits of the Preamble.

Name	Description	Min	Nom	Max	Units	Comment
<i>pDevFreq</i>	Maximum difference between the deviation frequency during the part of the packet following the Preamble and the reference deviation frequency.			1	%	The reference deviation frequency is the average deviation frequency during the last 32 bits of the Preamble.
<i>rTX</i>	The termination resistance and the cable impedance for test and calculation.	52	62	72	Ω	The impedance at the point between the TX block and the <i>rTX</i> resistor in Figure 5-10 is assumed to be 0Ω when the transceiver is powered (See also Figure 4-4).
<i>tEndDriveBFSK</i>	Time to cease driving the line after the end of the last bit of a Frame	0		4	μs	
<i>vTX</i>	TX voltage injected on V <sub>BUS</sub>	100	150	200	mVRMS	This is the voltage on V <sub>BUS</sub> when terminated by a nominal <i>rTX</i> through a cable no longer than 250mm whose characteristic impedance matches the termination impedance.

#### 5.8.2.5.1.1 Carrier Noise

The carrier phase noise should be considered when designing the transmitter system in order to achieve the required BER (*nBER*). The carrier phase noise can be measured while the transmitter sends a *BIST Carrier Mode 0* or *BIST Carrier Mode 1*.

#### 5.8.2.5.1.2 Carrier-Frequency Drift

*fCarrier* is the carrier frequency over the previous 10 bits at a given point in the packet. The change in *fCarrier* during a packet shall be less than *pCarrierFreq*. The reference carrier frequency (refCarrierFreq) is the average *fCarrier* of the last 32 bits of the Preamble. *fCarrier* during the packet, including the *EOP*, shall be within *pCarrierFreq* of the reference carrier frequency. *pCarrierFreq* is expressed as a percentage:

$$pCarrierFreq = |fCarrier - refCarrierFreq| / refCarrierFreq \times 100\%$$

The transmitter shall have the same *pCarrierFreq* for all packet types. The *BIST Carrier Mode 3* signal is an example of a continuous signal without a Preamble. When checking *pCarrierFreq* any set of 1044 bits within a continuous signal may be considered as the part of the packet following the Preamble and the 32 preceding bits considered to be the last 32 bits of the Preamble used to compute refCarrierFreq.

#### 5.8.2.5.1.3 Deviation-Frequency Drift

*fDeviation* is the frequency deviation during one bit at any point in the packet. The change in *fDeviation* shall be less than *pDevFreq*. The reference deviation frequency (refDevFreq) is the average *fDeviation* during the last 32 bits of the Preamble. *fDeviation* during the packet, including the *EOP*, shall be within *pDevFreq* of the reference deviation frequency. *pDevFreq* is expressed as a percentage:

$$pDevFreq = |fDeviation - refDevFreq| / refDevFreq \times 100\%$$

The transmitter shall have the same *fDeviation* for all packet types. The *BIST Carrier Mode 0* and *BIST Carrier Mode 1* signals are examples of continuous signals without a Preamble. When checking *pDevFreq* any set of 1044 bits within a continuous signal may be considered as the part of the packet following the Preamble and the 32 preceding bits considered to be the last 32 bits of the Preamble for computing *refDevFreq*.

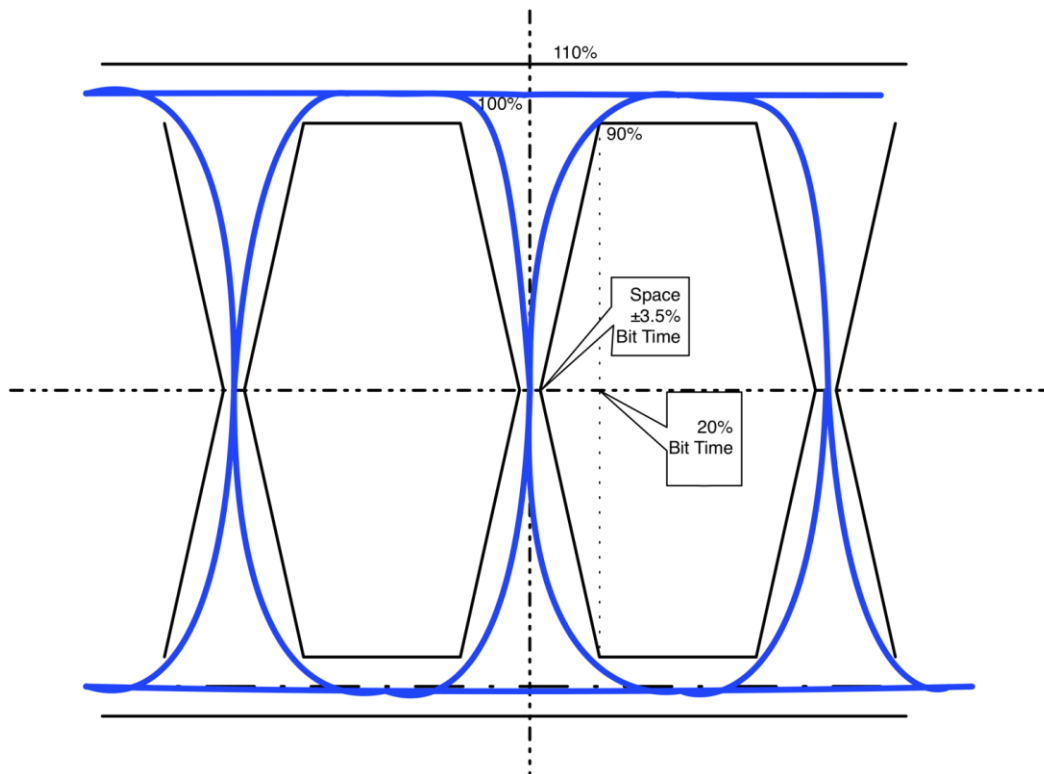
#### 5.8.2.5.2 Transmitter Characteristics

In order to allow for low cost and simple receivers, there is a requirement for the transmitted waveform to have a minimum of edge steepness. The transmitted waveform shall fulfill the eye diagram mask in Figure 5-11.

In Figure 5-11 the Y-axis corresponds to the nominal deviation. A continuous string of '1's will produce a constant frequency, which is designated 100% in this figure. This frequency must fulfil the relevant requirements regarding tolerances as described in Table 5-16. Similarly a continuous stream of '0's corresponds to -100% on the Y-axis. This allows for a pre-filtering of the signal used for modulation if desired but limits the amount of filtering allowed. The designer can balance the design between filtering at the baseband or the RF level as appropriate as long as all of the requirements are fulfilled.

When starting to transmit a frame the transmitter shall enable its carrier *tStartDrive* before the start of the first preamble bit. It shall stop transmitting a carrier within *tEndDriveBFSK* of the end of the last transmitted symbol.

Figure 5-11 Eye diagram of BFSK Modulation

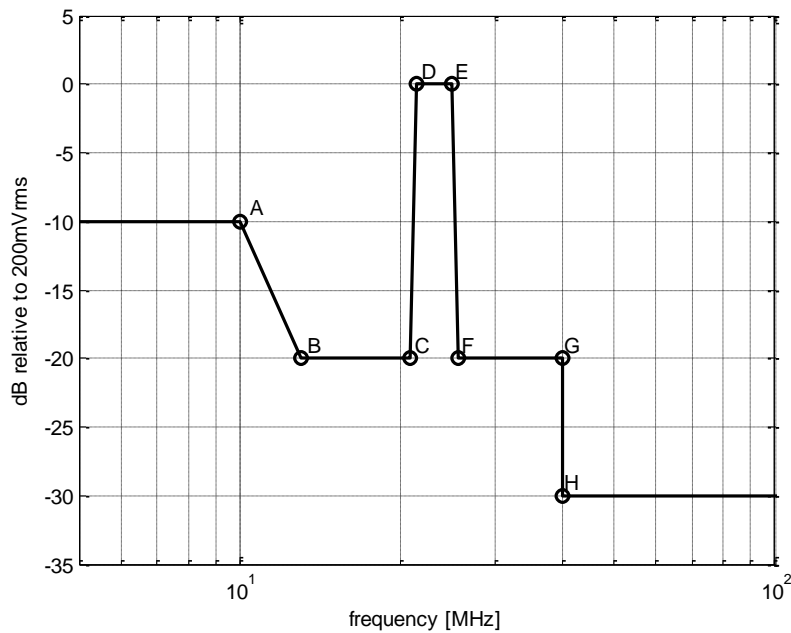


In order to manage the noise emitted from the cables, the emitted spectrum, on  $V_{BUS}$  at the transmitting device receptacle, shall comply with the mask in Figure 5-12 when  $V_{BUS}$  is terminated by a nominal *rTX* at the connector

through a cable no longer than 250mm whose characteristic impedance matches the termination impedance. Normal rules and regulations for noise emissions shall still be applicable.

Side lobes outside the coverage of Figure 5-12 shall be kept below the level as the figure shows.

Figure 5-12 BFSK Transmit Spectral Mask, given in absolute terms relative to the maximum value of vTX



The corners in Figure 5-12 are specified in Table 5-19.

Table 5-19 BFSK Spectrum Mask Corners

Frequency (MHz)	Maximum Allowed Signal Level	
	(dB)	(mVrms)
< 10 (A)	-10	63.2
13 (B) to 20.8(C)	-20	20
21.45(D) to 24.89 (E)	0	200
25.54(F) to 40 (G)	-20	20
>40 (H)	-30	6.3

### 5.8.2.6 BFSK Receiver Specifications

#### 5.8.2.6.1 BFSK Receiver Electrical Parameters

The requirements specified in Table 5-20 shall apply to the receiver (except *vSquelchDetecting*). There are two different squelch modes for the receiver. The Squelch Detection mode is used when the receiver is implementing cable-type detection following the suggested method in Section 4.4. The Squelch Operating mode is used when the receiver is watching for a packet to arrive. These two squelch modes have different required sensitivities. In the

Squelch Detection mode, the receiver detects signals exceeding *vSquelchDetecting*. In the Squelch Operating mode the receiver shall meet the requirement *nBER* when the signal level exceeds *vSquelchOperating*.

The input impedance of the receiver (the RX block in Figure 5-10 and Figure 4-4) is *zRX*. The input impedance as measured between  $V_{BUS}$  and GND is determined by *rTX* (see Section 5.8.2.5). The high impedance *zRX* is required for cable-type detection (see Section 4.4.4).

The receiver shall meet the *nBER* performance requirement in Table 5-20 when the voltage received on  $V_{BUS}$  is within the allowable range of *vRX*. Cables close to a quarter wavelength with characteristic impedance higher or lower than *rTX* may attenuate or amplify the signal level, so the allowable range of *vRX* includes margin above and below the allowable *vTX*. The ranges for *vRX* were selected to cover the variation seen in legacy cables.

Table 5-20 BFSK Receiver Normative Requirements

Name	Description	Min	Nom	Max	Units	Comment
<i>tBitStreamComplete</i>	The Bit Stream has stopped if the squelch has closed for <i>tBitStreamComplete</i> .	1		3	ms	
<i>vRX</i>	RX voltage received on $V_{BUS}$	55	150	300	mVRMS	This is the voltage on $V_{BUS}$ (the link terminated by <i>rTX</i> ). Legacy cables close to a quarter wavelength with characteristic impedance higher or lower than <i>rTX</i> may attenuate or amplify the signal level by up to 6dB. In practice, the minimum value will be the actual <i>vSquelchOperating</i> value.
<i>vSquelchDetecting</i>	Squelch detection sensitivity in Squelch Detection Mode	15	20	25	mVRMS	Informative only.
<i>vSquelchOperating</i>	Squelch detection sensitivity In Squelch Operating Mode	35		55	mVRMS	This parameter only pertains to the signal content within the frequency band of interest within <i>fRange</i> .
<i>zRX</i>	Receiver input impedance	10			k $\Omega$	15% tolerance, measured from $V_{BUS}$ to GND in the band from 19 MHz to 27 MHz. Note: The input impedance as measured between $V_{BUS}$ and GND is determined by <i>rTX</i> (see Section 5.10). The high impedance <i>zRX</i> is required for cable-type detection (see Section 4.4.4).

#### 5.8.2.6.2 Receiver Filter Specification

The design of the receiver filter represented by the “Filter Bandpass” in

Figure 5-9 Receiver Block Diagram

is implementation specific, but shall take into account the out-of-band power-supply noise (see Sections 7.1.13.2 and 7.2.9.2).

5.8.2.6.3 Crosstalk in the cables

In order to maintain good communications, the cables shall fulfill the crosstalk requirements in Section 3.6.6.

5.8.2.6.4 Definition of Idle

For the BFSK Signaling Scheme  $V_{BUS}$  shall be declared idle when the signal level is less than *vSquelchOperating*. The power supply noise allowed by Figure 7-8 and Figure 7-11 shall not cause the receiver to indicate the channel is busy.

5.8.2.7 Bit Stream

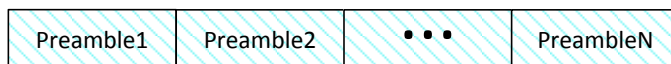
A Bit Stream transmission is defined to allow a Consumer/Provider using BFSK on a Type-B connector to detect a Provider/Consumer with a Dead Battery on a Type-A connector (see Section 4.1).

The transmitter of the Provider/Consumer that implements Dead Battery Support shall be able to transmit a Bit Stream consisting of alternating “0s” and “1s” which may be viewed as concatenating multiple Preambles as shown in Figure 5-13 (note that the last Preamble may not contain 64 bits). The PHY Layer shall continue to transmit the Bit Stream until the PD System is ready for a *Source\_Capabilities* Message or  $V_{BUS}$  becomes *vSafe0V* (see Figure 4-1).

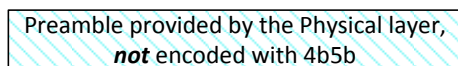
The Consumer/Provider’s receiver shall declare a Bit Stream is detected after detecting 128 consecutive bits that match the Preamble pattern of alternating “0s” and “1s”.

After the Bit Stream is detected, the receiver shall indicate that the Bit Stream has stopped when the squelch has closed (the signal level is below *vSquelchOperating*) for *tBitStreamComplete*.

Figure 5-13 Line Format of Bit Stream



LEGEND:





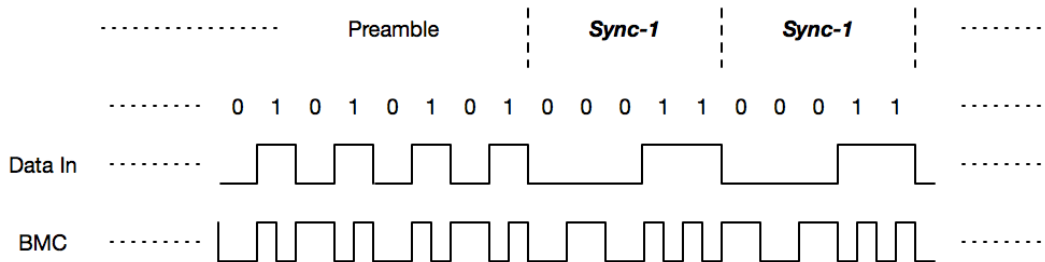
### 5.8.3 Biphas Mark Coding (BMC) Signaling Scheme

Biphase Mark Coding (BMC) is an alternative physical layer for carrying USB Power Delivery Messages. This encoding assumes a dedicated DC connection, identified as the CC wire, which is used for sending PD Messages.

Biphase Mark Coding is a version of Manchester coding (see [\[IEC 60958-1\]](#)). In BMC, there is a transition at the start of every bit time (UI) and there is a second transition in the middle of the UI when a 1 is transmitted. BMC is effectively DC balanced, (each 1 is DC balanced and two successive zeroes are DC balanced, regardless of the number of intervening 1's). It has bounded disparity (limited to 1 bit over an arbitrary packet, so a very low DC level).

Figure 5-14 illustrates Biphas Mark Coding. This example shows the transition from a Preamble to the *Sync-1* K-codes of the *SOP* Ordered Set at the start of a Message. Note that other K-codes can occur after the Preamble for Signaling such as *Hard Reset* and *Cable Reset*.

Figure 5-14 BMC Example



#### 5.8.3.1 Encoding and signaling

BMC uses DC coupled baseband signaling on CC. Figure 5-15 shows a block diagram for a Transmitter and Figure 5-16 shows a block diagram for the corresponding Receiver.

Figure 5-15 BMC Transmitter Block Diagram

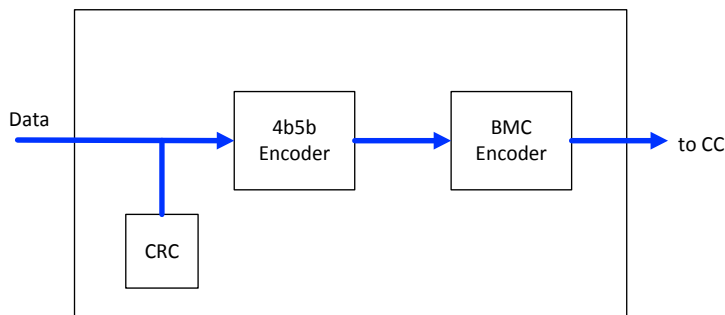
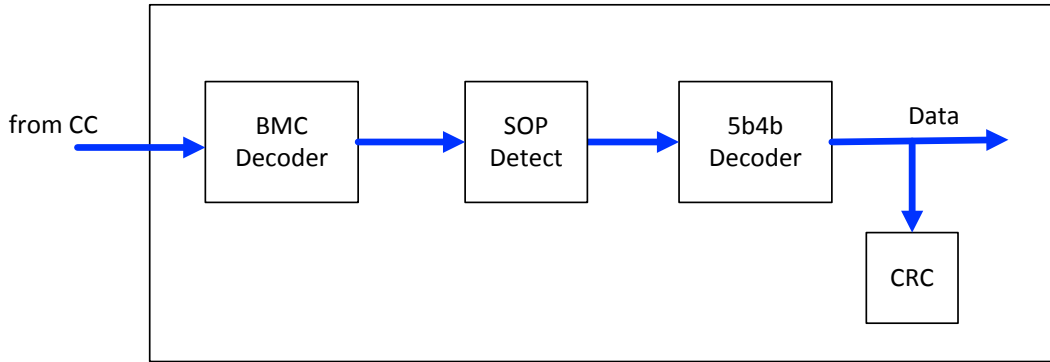


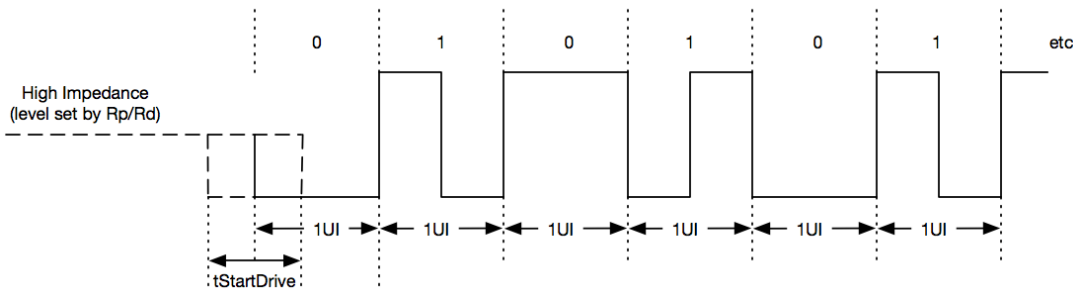
Figure 5-16 BMC Receiver Block Diagram



The USB PD baseband signal shall be driven on the CC wire with a tri-state driver that shall cause a *vSwing* swing on CC. The tri-state driver is slow rate limited (see min rise/fall time in Section 5.8.3.5) to limit coupling to D+/D- and to other signal lines in the Type-C fully featured cables (see [USBType-C 1.0]). This slow rate limiting can be performed either with driver design or an RC filter on the driver output.

When sending the Preamble, the transmitter shall start by transmitting a low level. The receiver shall tolerate the loss of the first edge. The transmitter may vary the start of the Preamble by *tStartDrive* min (see Figure 5-17).

Figure 5-17 BMC Encoded Start of Preamble



The transmitter shall terminate the final bit of the Frame by an edge (the “trailing edge”) to help ensure that the receiver clocks the final bit. If the trailing edge results in the transmitter driving CC low (i.e. the final half-UI of the frame is high), then the transmitter:

1. Shall continue to drive CC low for *tHoldLowBMC*.
2. Then shall continue to drive CC low for *tEndDriveBMC* measured from the trailing edge of the final bit of the Frame.
3. Then shall release CC to high impedance.

Figure 5-18 illustrates the end of a BMC encoded Frame with an encoded zero for which the final bit of the Frame is terminated by a high to low transition. Figure 5-19 illustrates the end of a BMC Encoded frame with an encoded one for which the final bit of the Frame is terminated by a high to low transition. Both figures also illustrate the *InterFrameGap* timing requirement before the start of the next Frame when the Port has either been transmitting or receiving the previous Frame (see Section 5.8.1.4).

Figure 5-18 Transmitting or Receiving BMC Encoded Frame Terminated by Zero with High-to-Low Last Transition

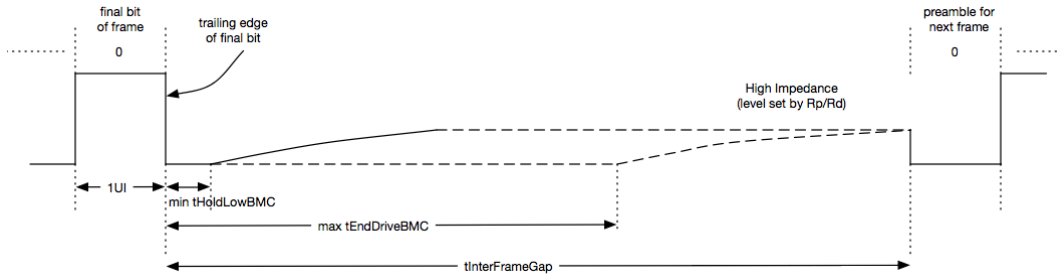
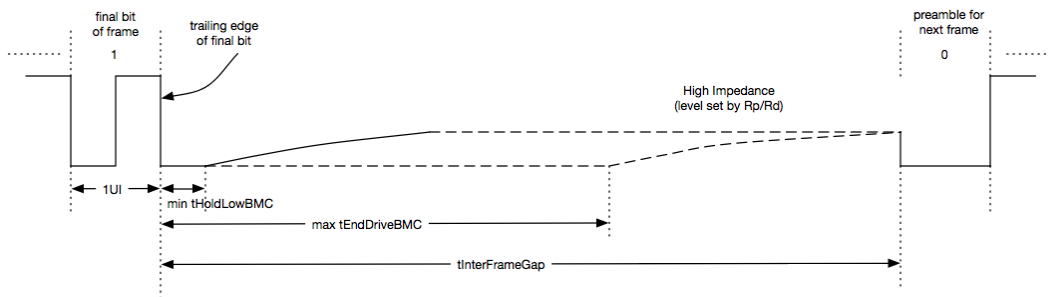


Figure 5-19 Transmitting or Receiving BMC Encoded Frame Terminated by One with High-to-Low Last Transition



If the trailing edge results in the transmitter driving CC high (i.e. the final half-UI of the frame is low), then the transmitter:

1. Shall continue to drive CC high for 1 UI.
2. Then shall drive CC low for  $t_{HoldLowBMC}$ .
3. Then shall continue to drive CC low for  $t_{EndDriveBMC}$  measured from the final edge of the final bit of the Frame.
4. Then shall release CC to high impedance.

Figure 5-20 illustrates the ending of a BMC encoded Frame that ends with an encoded zero for which the final bit of the Frame is terminated by a low to high transition. Figure 5-21 illustrates the ending of a BMC encoded Frame that ends with an encoded zero for which the final bit of the Frame is terminated by a low to high transition. Both figures also illustrate the  $t_{InterFrameGap}$  timing requirement before the start of the next Frame when the Port has either been transmitting or receiving the previous Frame (see Section 5.8.1.4).

Figure 5-20 Transmitting or Receiving BMC Encoded Frame Terminated by Zero with Low to High Last Transition

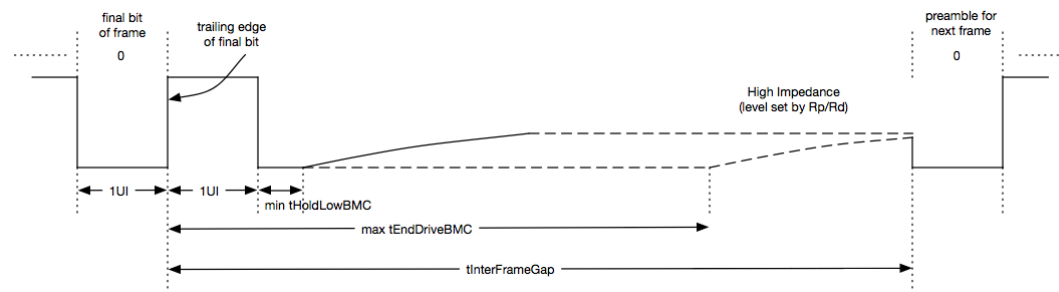


Figure 5-21 Transmitting or Receiving BMC Encoded Frame Terminated by One with Low to High Last Transition

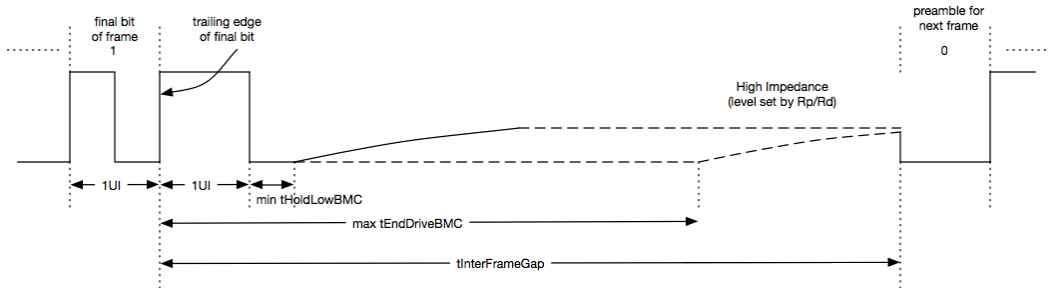
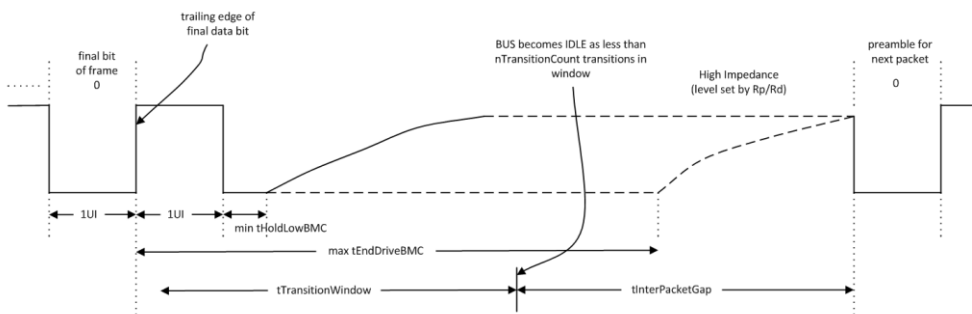


Figure 5-22 illustrates the end of a BMC encoded Frame with an encoded zero for which the final bit of the Frame is terminated by a high to low transition. The figure also illustrates the *tTransitionWindow* followed *tInterFrameGap* timing requirement before the start of the next Frame, when the Port did not either transmit or receive the previous Frame and is waiting for bus idle before transmission of the next Frame (see Section 5.8.1.4).

Figure 5-22 Waiting for idle after a BMC Encoded Frame Terminated by Zero with High-to-Low Last Transition



Note: There is no requirement to maintain a timing phase relationship between back-to-back packets.

### 5.8.3.2 Transmit and Receive Masks

#### 5.8.3.2.1 Transmit Masks

The transmitted signal shall not violate the masks defined in Figure 5-23, Figure 5-24, Table 5-21 and Table 5-22 at the output of a load equivalent to the cable model and receiver load model described in Section 5.8.3.3. ~~The receiver shall be capable of receiving a signal that does not violate the masks defined in and and. The sign of the ground offset will not be changed during a message. Therefore, the minimum voltage swing between the high level and the low level of the received signal will be at least  $Y4Rx - Y2Rx + 250mV$ . The parameters are specified to be appropriate to either edge-triggered or oversampling receiver implementations. Note: the measurement of the transmitter does not need to accommodate a change in signal offset due to the ground offset when current is flowing in the cable.~~

The masks are defined for 'ONE' and 'ZERO' separately as BMC enforces a transition at the midpoint of the unit interval while a 'ONE' is transmitted.

~~The boundaries of Rx outer mask are specified to accommodate a change in signal amplitude due to the ground offset. They are therefore larger than the boundaries of the Tx outer mask. Similarly, the boundaries of the Rx inner mask are smaller than the boundaries of the Tx inner mask.~~

The transmitter shall have a rise and fall time no faster than  $t_{Rise}$ . The transmitted signal shall have a fall time no faster than  $t_{Fall}$  the minima specified in [5.6.2.2](#). The maximum limits on the rise and fall times are enforced by the Tx inner masks.

Figure 5-23 BMC Tx 'ONE' Mask

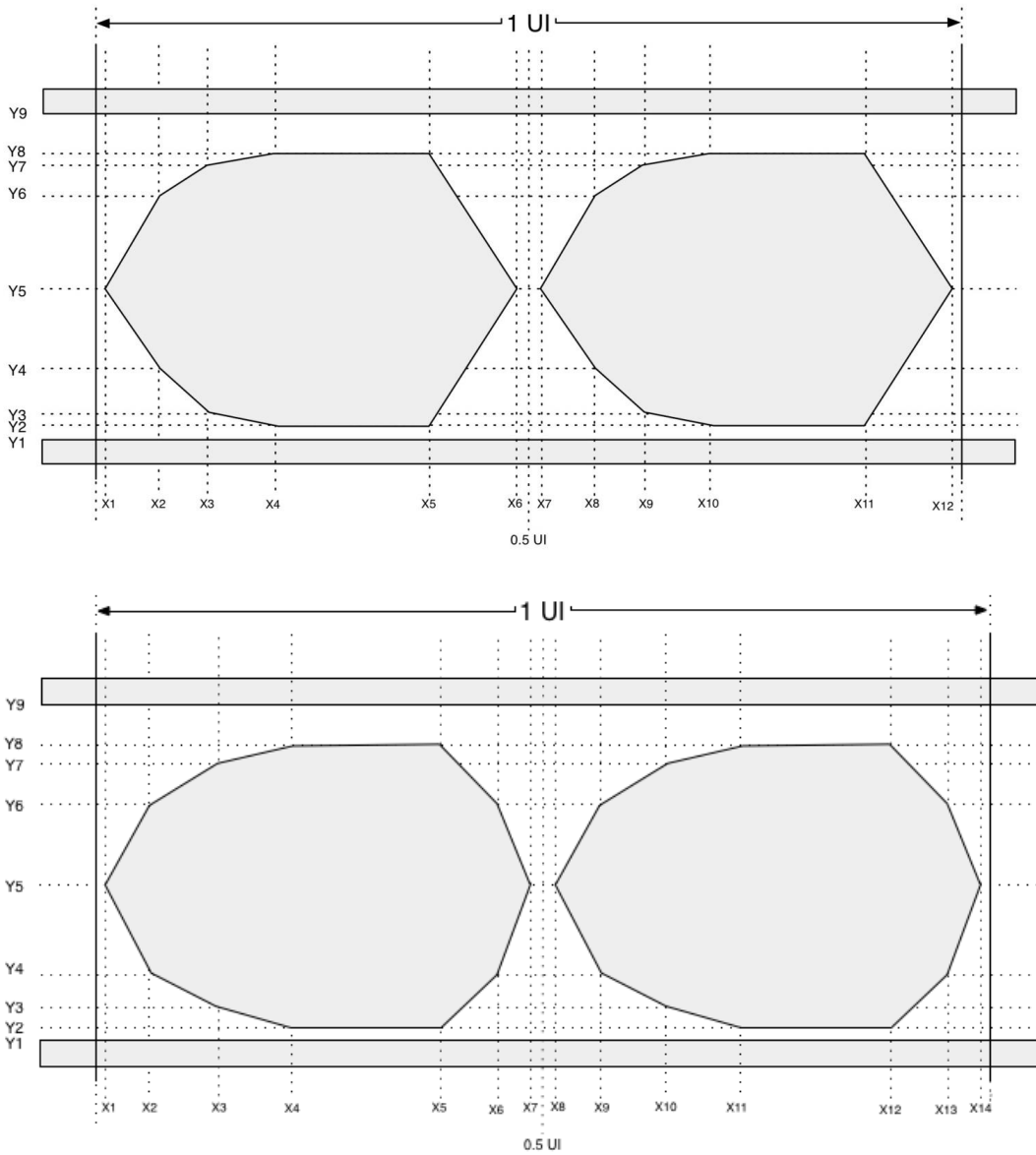


Figure 5-24 BMC Tx 'ZERO' Mask

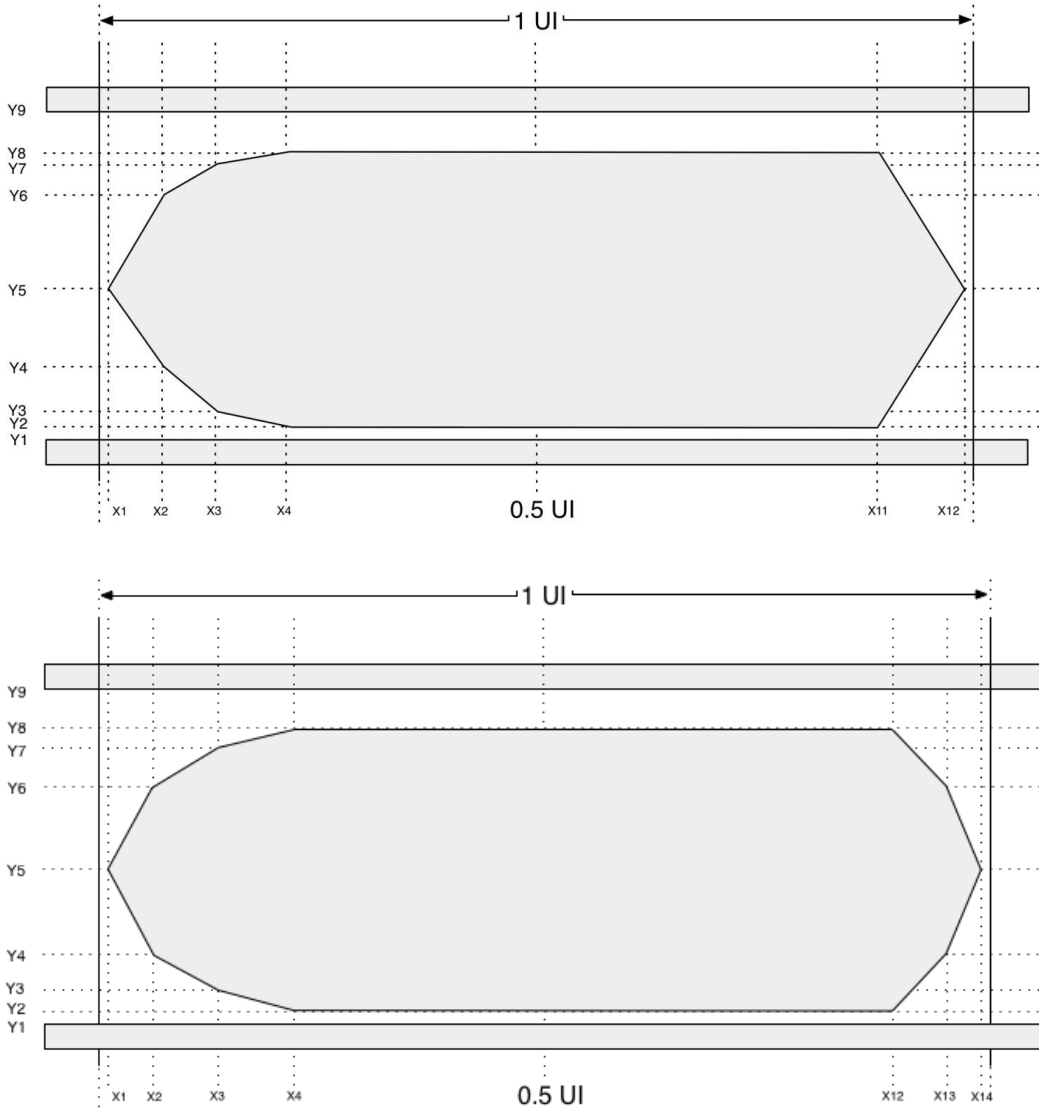


Table 5-21 BMC Tx Mask Definition, X Values

Name	Description	Value	Units
X1Tx	Left Edge of Mask	0.015	UI
X2Tx	see figure	0.07507	UI

Name	Description	Value	Units
X3Tx	see figure	0.12515	UI
X4Tx	see figure	0.225	UI
X5Tx	see figure	0.37535	UI
X6Tx	see figure	0.48543	UI
X7Tx	see figure	0.545485	UI
X8Tx	see figure	0.575515	UI
X9Tx	see figure	0.62557	UI
X10Tx	see figure	0.765	UI
X11Tx	see figure	0.87575	UI
X12Tx	see figure	0.85	UI
X13Tx	see figure	0.93	UI
<del>X12Tx</del> X14Tx	Right Edge of Mask	0.985	UI

Table 5-22 BMC Tx Mask Definition, Y Values

Name	Description	Value	Units
Y1Tx	Lower bound of Outer mask	-0.075	V
Y2Tx	Lower bound of inner mask	0.075	V
Y3Tx	see figure	0.11315	V
Y4Tx	see figure	0.225325	V
Y5Tx	Inner mask vertical midpoint	0.5625	V
Y6Tx	see figure	0.98	V
Y7Tx	see figure	1.0120975	V
Y8Tx	see figure	1.0504	V
Y9Tx	Upper Bound of Outer mask	1.2	V

#### 5.8.3.2.2 Receive Masks

A Provider using the BMC Signaling Scheme shall be capable of receiving a signal that complies with the mask when sourcing power as defined in Figure 5-25, Figure 5-26 and Table 5-23. The Provider Rx mask is bounded by sweeping a Tx mask compliant signal, with added vNoiseActive between power neutral and Provider offsets.

A Consumer using the BMC Signaling Scheme shall be capable of receiving a signal that complies with the mask when sinking power as defined in Figure 5-29, Figure 5-30 and Table 5-23. The Consumer Rx mask is bounded by sweeping a Tx mask compliant signal, with added vNoiseActive between power neutral and Consumer offsets.

Every product using the BMC Signaling Scheme shall be capable of receiving a signal that complies with the mask when power neutral as defined in Figure 5-27, Figure 5-28 and Table 5-23.

Dual-Role Devices shall meet the receiver requirements for a Provider when providing power during any transmission using the BMC Signaling Scheme or a Consumer when consuming power during any transmission using the BMC Signaling Scheme.

Cable Plugs shall meet the receiver requirements for both a Provider and a Consumer during any transmission using the BMC Signaling Scheme.

The parameters used in the masks are specified to be appropriate to either edge triggered or oversampling receiver implementations.

The masks are defined for 'ONE' and 'ZERO' separately as BMC enforces a transition at the midpoint of the unit interval while a 'ONE' is transmitted.

The Rx masks are defined to bound the Rx noise after the Rx bandwidth limiting filter with the time constant  $t_{RxFilter}$  has been applied.

The boundaries of Rx outer mask,  $Y1Rx$  and  $Y5Rx$ , are specified according to  $v_{Swing\ max}$  and accommodate half of  $v_{NoiseActive}$  from cable noise coupling and the signal offset  $v_{IRDropGNDC}$  due to the ground offset when current is flowing in the cable.

The vertical dimension of the Rx inner mask,  $Y4Rx - Y2Rx$ , for power neutral is derived by reducing the vertical dimension of the Tx inner mask,  $Y7Tx - Y3Tx$ , at time location  $X3Tx$  by  $v_{NoiseActive}$  to account for cable noise coupling. The received signal is composed of a waveform compliant to the Tx mask plus  $v_{NoiseActive}$ .

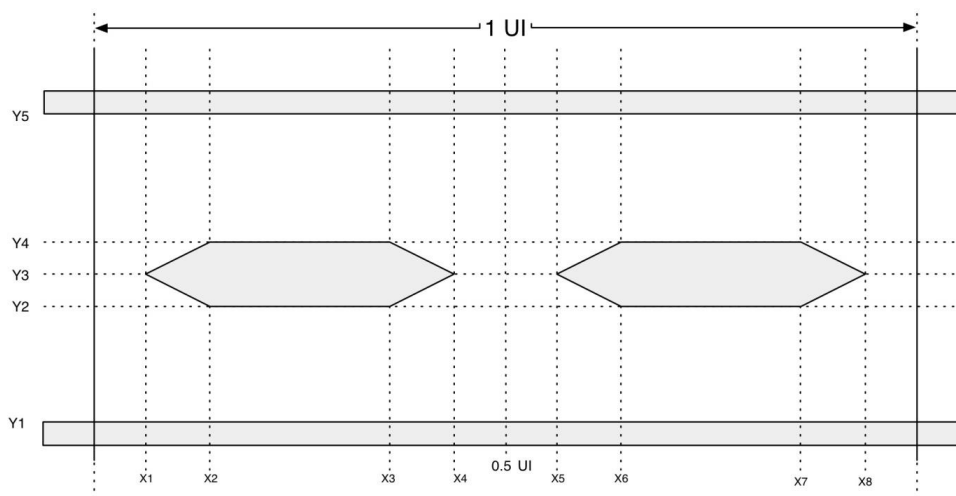
The vertical dimension of the Rx inner mask for sourcing power is derived by reducing the vertical dimension of the Tx inner mask by  $v_{NoiseActive}$  and  $v_{IRDropGNDC}$  to account for both cable noise coupling and signal DC offset. The received signal is composed of a waveform compliant to the Tx mask plus the maximum value of  $v_{NoiseActive}$  plus  $v_{IRDropGNDC}$  where the  $v_{IRDropGNDC}$  value transitions between the minimum and the maximum values as allowed in this spec.

The vertical dimension of the Rx inner mask for sinking power is derived by reducing the vertical dimension of the Tx inner mask by  $v_{NoiseActive\ max}$  and  $v_{IRDropGNDC\ max}$  for account for both cable noise coupling and signal DC offset. The received signal is composed of a waveform compliant to the Tx mask plus the maximum value of  $v_{NoiseActive}$  plus  $v_{IRDropGNDC}$  where the  $v_{IRDropGNDC}$  value transitions between the minimum and the maximum values as allowed in this spec.

The center line of the Rx inner mask,  $Y3Rx$ , is at half of the nominal  $v_{Swing}$  for power neutral, and is shifted up by half of  $v_{IRDropGNDC\ max}$  for sourcing power and is shifted down by half of  $v_{IRDropGNDC\ max}$  for sinking power.

The receiver sensitivity shall be set such that the receiver does not treat noise on an un-driven signal path as an incoming signal. Signal amplitudes below  $v_{NoiseIdle\ max}$  shall be treated as noise when BMC is idle.

Figure 5-25 BMC Rx 'ONE' Mask when Sourcing Power





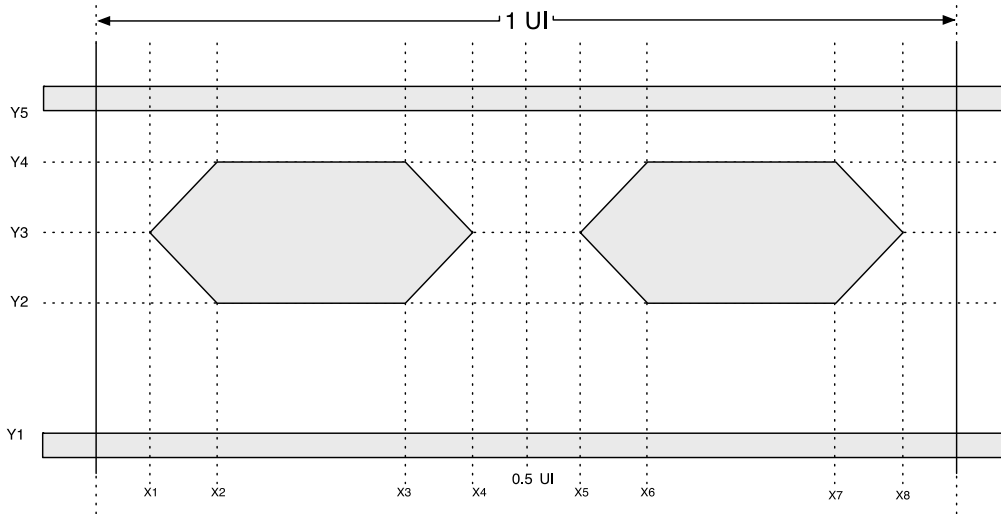
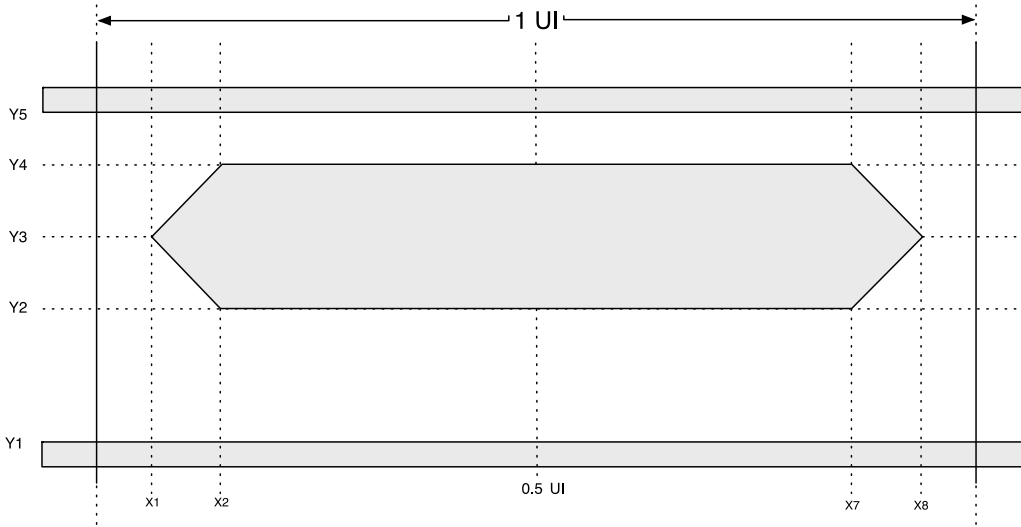
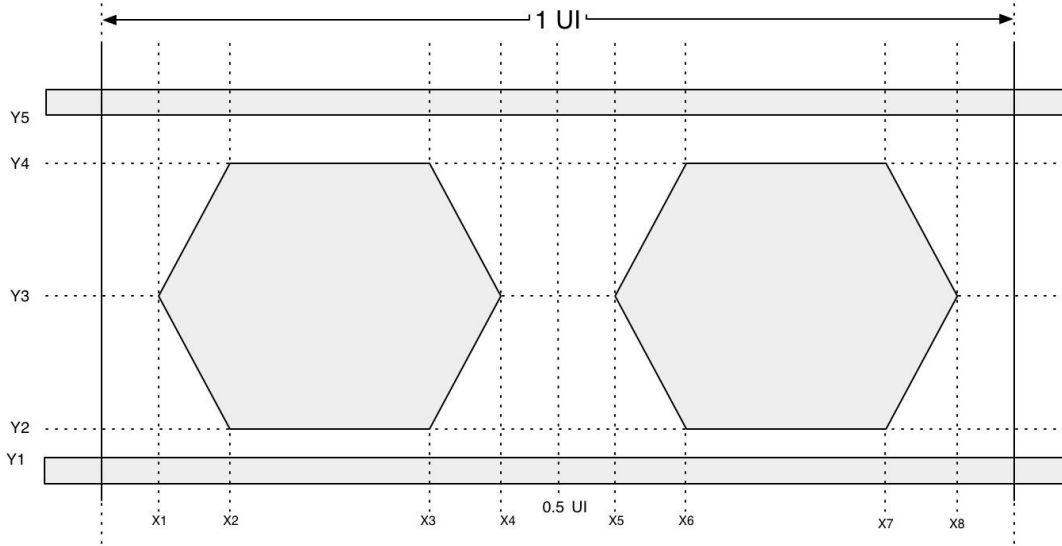


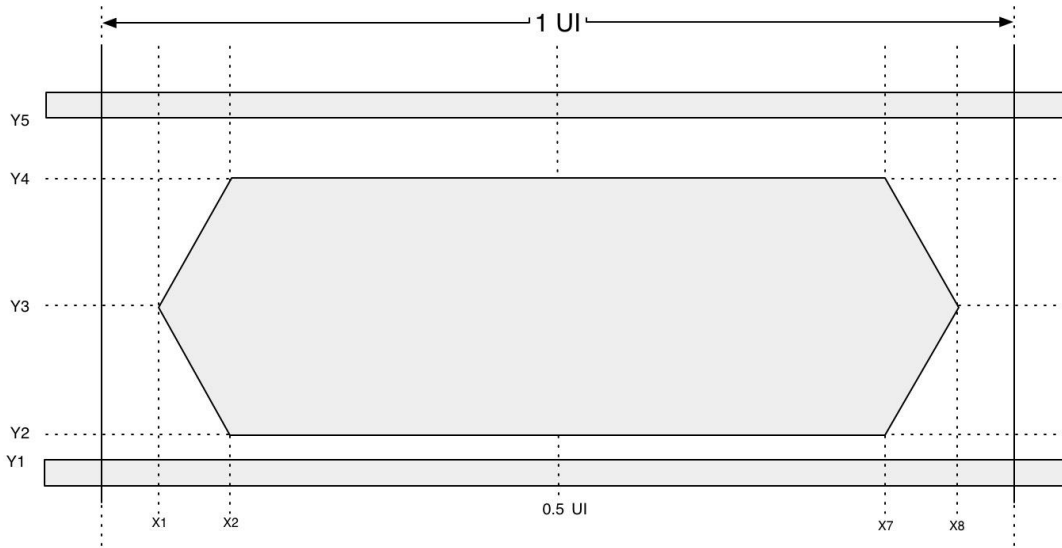
Figure 5-26 BMC Rx 'ZERO' Mask when Sourcing Power



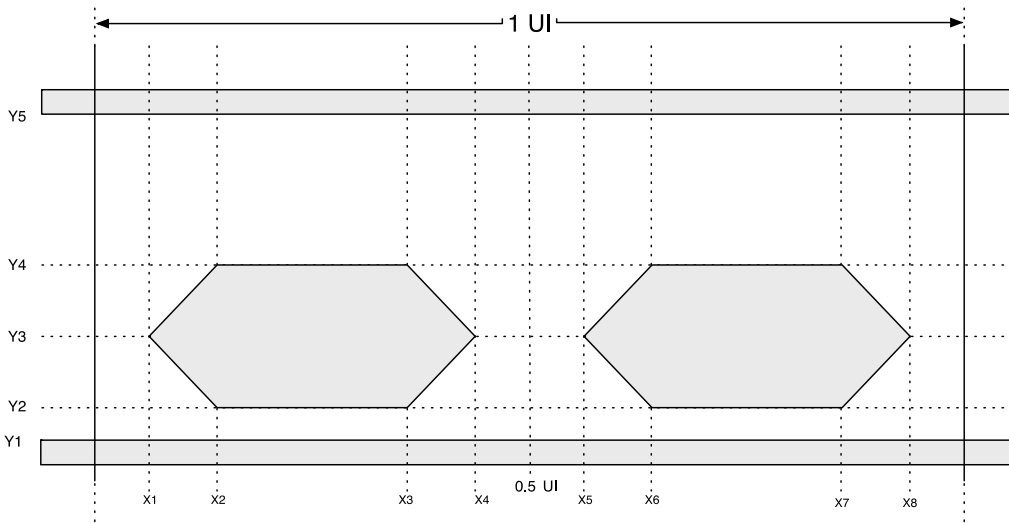
**Figure 5-27 BMC Rx 'ONE' Mask when Power neutral**



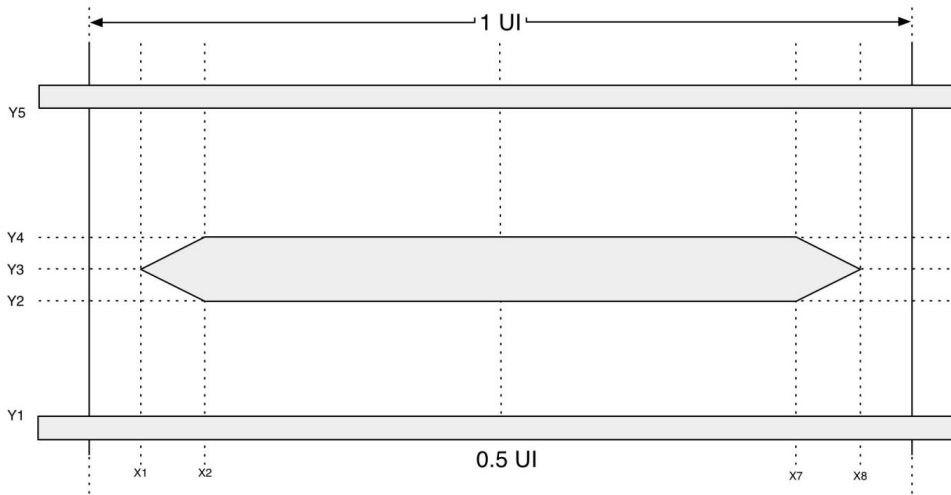
**Figure 5-28 BMC Rx 'ZERO' Mask when Power neutral**



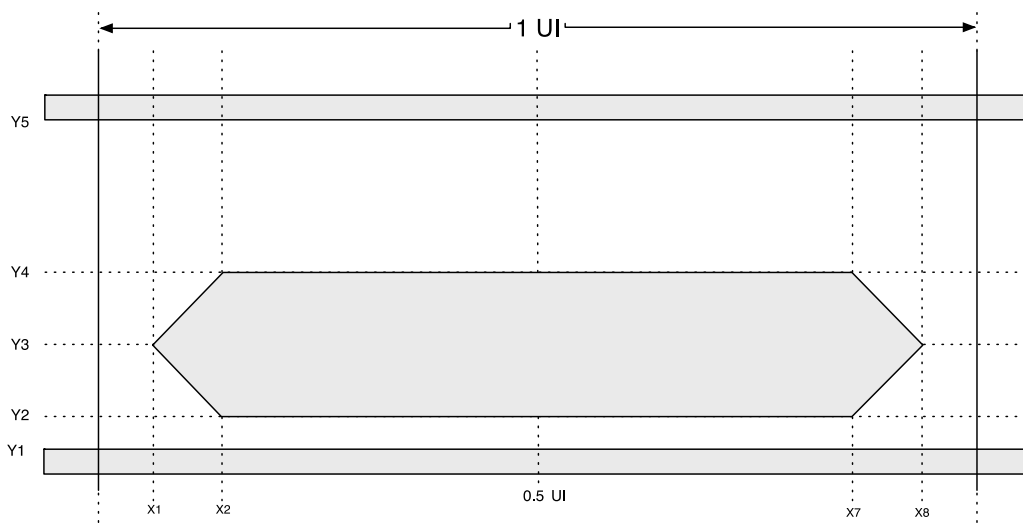
**Figure 5-29** BMC Rx 'ONE' Mask when Sinking Power



**Figure 5-30**



**BMC Rx 'ZERO' Mask when Sinking Power**



**Table 5-23 BMC Rx Mask Definition**

Name	Description	Value	Units
<i>X1Rx</i>	Left Edge of Mask	0.07	UI
<i>X2Rx</i>	Top Edge of Mask	0.15	UI
<i>X3Rx</i>	See figure	0.35	UI
<i>X4Rx</i>	See figure	0.43	UI
<i>X5Rx</i>	See figure	0.57	UI
<i>X6Rx</i>	See figure	0.65	UI
<i>X7Rx</i>	See figure	0.85	UI
<i>X8Rx</i>	See figure	0.93	UI
<i>Y1Rx</i>	Lower bound of Outer Mask	-0.253325	V
<i>Y2Rx</i>	Lower Bound of Inner Mask	<i>Y3Rx</i> - 0.205 when sourcing power <sup>1</sup> or sinking power <sup>1</sup> <i>Y3Rx</i> 0.43 - 0.33 when power neutral <sup>1</sup>	V
<i>Y3Rx</i>	Center line of Inner Mask	0.5506875 Sourcing Power <sup>1</sup> 0.5625 Power Neutral <sup>1</sup> 0.4375 Sinking Power <sup>1</sup>	V

Name	Description	Value	Units
<i>Y4Rx</i>	Upper bound of Inner mask	<i>Y3Rx</i> + 0.205 when sourcing power! or <i>Y3Rx</i> 0.67 + 0.33 when power neutral!	V
<i>Y5Rx</i>	Upper bound of the Outer mask	1.455325	V

Note 1: The position of the center line of the Inner Mask is dependent on whether the receiver is Sourcing or Sinking power or is Power Neutral (see earlier in this section).

### 5.8.3.3 Transmitter Load Model

The transmitter load model for the eye mask tests and *vSwing* shall be equivalent to the circuit outlined in Figure 5-31 for a Source and Figure 5-32 for a Sink. It is formed by the concatenation of a cable load model and a receiver load model. See [USBType-C 1.0] for details of the *Rp* and *Rd* resistors.

Figure 5-31 Transmitter Load Model for BMC Tx from a Source

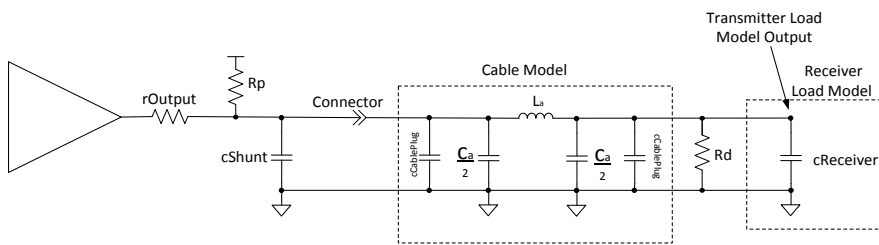
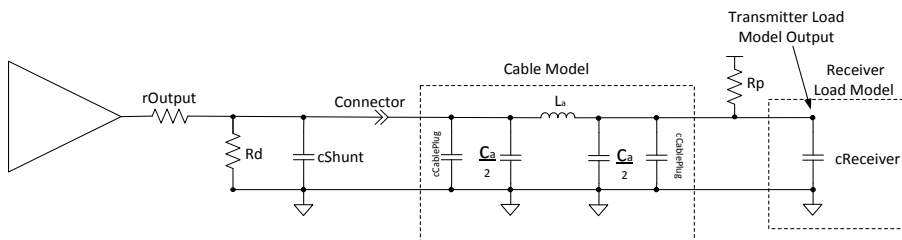


Figure 5-32 Transmitter Load Model for BMC Tx from a Sink



The transmitter system components *rOutput* and *cShunt* are illustrated for informative purposes, and do not form part of the transmitter load model. See Section 5.8.3.5 for a description of the transmitter system design.

The value of the modeled cable inductance, *La*, (in nH) shall be calculated from the following formula:

$$La = t_{CableDelay} \frac{t_{CableDelay}_{max} - t_{CableDelay}_{min}}{t_{CableDelay}_{max} - t_{CableDelay}_{min}} * z_{Cable}_{min}$$

*tCableDelay* is the modeled signal propagation delay through the cable, and *zCable* is the modeled cable impedance.

The modeled cable inductance is 640 nH for a cable with *zCable*<sub>min</sub> = 32 Ω and *tCableDelay*<sub>max</sub> = 20 nS.

The value of the modeled cable capacitance, *Ca*, (in pF) shall be calculated from the following formula:

$$C_a = \frac{t_{CableDelay_{max}}}{z_{Cable_{min}}}$$

The modeled cable capacitance is  $C_a = 625$  pF for a cable with  $z_{Cable_{min}} = 32 \Omega$  and  $t_{CableDelay_{max}} = 20$  nS. Therefore,  $C_a/2 = 312.5$  pF.

**cCablePlug** models the capacitance of the plug at each end of the cable. **cReceiver** models the capacitance of the receiver. The maximum values shall be used in each case.

Note: the transmitter load model assumes that there are no other return currents on the ground path.

#### 5.8.3.4 BMC Common specifications

This section defines the common receiver and transmitter requirements.

##### 5.8.3.4.1 BMC Common Parameters

The electrical requirements specified in Table 5-24 shall apply to both the transmitter and receiver.

Table 5-24 BMC Common Normative Requirements

Name	Description	Min	Nom	Max	Units	Comment
<i>tUnitInterval</i> <sup>1</sup>	Unit Interval	3.03		3.70	μs	1/ <i>fBitRate</i>
<i>cCablePlug</i> <sup>2</sup>	Capacitance for a Cable Plug			25	pF	Each plug on a cable assembly can have capacitance up to this value
<i>tCableDelay</i>	Signal propagation delay through the cable			20	ns	
<i>zCable</i>	Cable characteristic impedance on CC Wire as defined in [USBType-C 1.0].	32		65	Ω	See [USBType-C 1.0] for Cable impedance values.

Note 1: *tUnitInterval* denotes the time to transmit an unencoded data bit, not the shortest high or low times on the wire after encoding with BMC. A single data bit cell has duration of 1UI, but a data bit cell with value 1 will contain a centrally placed 01 or 10 transition in addition to the transition at the start of the cell.

Note 2: The capacitance of the bulk cable is not included in the *cCablePlug* definition. It is modeled as transmission line in both modeling and compliance processes. However *cCablePlug* does include the BMC transceiver's capacitance in the Cable Plug(s).

#### 5.8.3.5 BMC Transmitter Specifications

The transmitter shall meet the specifications defined in Table 5-25.

Table 5-25 BMC Transmitter Normative Requirements

Name	Description	Min	Nom	Max	Units	Comment
<i>tEndDriveBMC</i>	Time to cease driving the line after the end of the last bit of the Frame.			23	μs	Min value is limited by <i>tHoldLowBMC</i> .
<i>tFall</i>	Fall Time	300			ns	10 % and 90 % amplitude points, minimum is under an unloaded condition.
<i>tHoldLowBMC</i>	Time to cease driving the line after the final high-to-low transition.	1			μs	Max value is limited by <i>tEndDriveBMC</i> .
<i>tRise</i>	Rise time	300			ns	10 % and 90 % amplitude points, minimum is under an unloaded condition.

Name	Description	Min	Nom	Max	Units	Comment
<i>vSwing</i>	Voltage Swing	1.05	1.125	1.2	V	Applies to both no load condition and under the load condition specified in Section 5.8.3.3.
<i>zDriver</i>	Transmitter output impedance	33		75	Ω	Source output impedance at the Nyquist frequency of <a href="#">USB 2.0</a> low speed (750 kHz) while the source is driving the CC line.

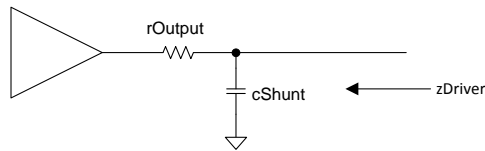
*cReceiver* is the capacitance that a DFP or UFP shall present on the CC line when the DFP or UFP's receiver is not transmitting on the line. The transmitter may have more capacitance than *cReceiver* while driving the CC line, but must meet the waveform mask requirements. Once transmission is complete, the transmitter shall disengage capacitance in excess of *cReceiver* from the CC wire within *tInterFrameGap*.

Source output impedance *zDriver* is determined by the driver resistance and the shunt capacitance of the source and is hence a frequency dependent term. *zDriver* impacts the noise ingress in the cable. It is specified such that the noise at the Receiver is bounded.

*zDriver* is defined by the following equation:

$$zDriver = \frac{rOutput}{1 + s * rOutput * cShunt}$$

Figure 5-33 Transmitter diagram illustrating *zDriver*



*cShunt* shall not cause a violation of *cReceiver* when not transmitting.

### 5.8.3.6 BMC Receiver Specifications

The receiver shall meet the specifications defined in Table 5-26.

Table 5-26 BMC Receiver Normative Requirements

Name	Description	Min	Nom	Max	Units	Comment
<i>cReceiver</i>	CC receiver capacitance	200		600	pF	The DFP or UFP system shall have capacitance within this range when not transmitting on the line.
<i>nTransitionCount</i>	Transitions for signal detect	3				Number of transitions to be detected to declare bus non-idle.
<i>tRxFilter</i>	Rx bandwidth limiting filter (digital or analog)	100			ns	Time constant of a single pole filter to limit broad-band noise ingress <sup>1</sup> .
<i>tTransitionWindow</i>	Time window for detecting non-idle	12		20	μs	
<i>zBmcRx</i>	Receiver Input Impedance	<del>401</del>			MΩ	

Name	Description	Min	Nom	Max	Units	Comment
<i>vNoiseActive</i>	Noise amplitude when BMC is active.			165	mV	Peak-to-peak noise from V <sub>BUS</sub> , USB 2.0 and SBU lines after the Rx bandwidth limiting filter with the time constant <i>tRxFilter</i> has been applied.
<i>vNoiseIdle</i>	Noise amplitude when BMC is idle.			300	mV	Peak-to-peak noise from V <sub>BUS</sub> , USB 2.0 and SBU lines after the Rx bandwidth limiting filter with the time constant <i>tRxFilter</i> has been applied.
<i>vIRDropGNDc</i>	Cable Ground IR Drop			250	mV	As specified in [USBType-C 1.0]

Note 1. Broad-band noise ingress is due to coupling in the cable interconnect.

#### 5.8.3.6.1 Definition of Idle

BMC packet collision is avoided by the detection of signal transitions at the receiver. This is the equivalent of squelch for FSK modulation. Detection is active when *nTransitionCount* transitions occur at the receiver within a time window of *tTransitionWindow*. After waiting *tTransitionWindow* without detecting *nTransitionCount* transitions the bus shall be declared idle.

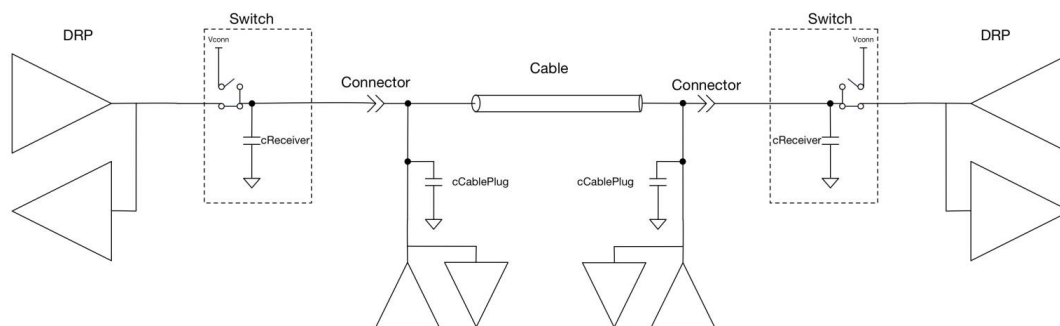
Refer to Section 5.8.1.4 for details of when transmissions may start.

#### 5.8.3.6.2 Multi-Drop

The BMC Signaling Scheme is suitable for use in Multi-Drop configurations containing one or two BMC Multi-Drop transceivers connected to the CC wire, for example where one or both ends of a cable contains a Multi-Drop transceiver. In this specification the location of the Multi-Drop transceiver is referred to as the Cable Plug.

Figure 5-34 below illustrates a typical Multi Drop configuration with two DRPs.

Figure 5-34 Example Multi-Drop Configuration showing two DRPs

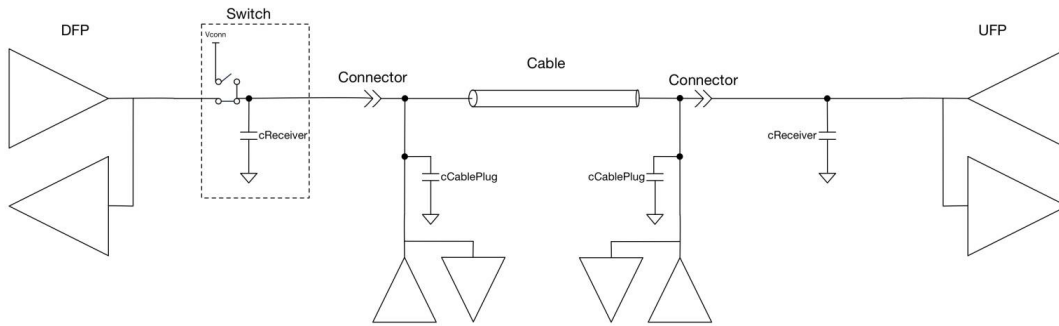


The Multi-Drop transceiver shall obey all the electrical characteristics specified in this section except for those relating to capacitance. The maximum capacitance allowed for the Multi-Drop node when not driving the line is *cCablePlug*. There are no constraints as to the distance of the Multi-Drop transceiver from the end of the plug. The Multi-Drop transceiver(s) may be located anywhere along the cable including the plugs. The Multi-Drop transceiver suffers less from ground offset compared to the transceivers in the host or device and contributes no significant reflections.



Since sourcing  $V_{CONN}$  is not mandated for UFPs, it is possible to have a configuration where there is no switch in the UFP as outlined in Figure 5-35. In this configuration, the capacitance on the CC line contained within the UFP shall still be within *cReceiver* except when transmitting.

Figure 5-35 Example Multi-Drop Configuration showing a DFP and UFP



#### 5.8.4 Interoperability with BFSK and BMC

In order to interoperate with systems supporting either BFSK or BMC, without requiring an adapter to convert between the two Signaling ~~systems~~**Schemes**, manufacturers may choose to support both BMC over the CC wire and BFSK over  $V_{BUS}$  on a Type-C connector. Products with Type-C connectors shall not support BFSK without supporting BMC. Note that any system utilizing the Type-C connector can see BFSK signaling on  $V_{BUS}$  when a suitable Type-A/B to Type-C adapter is used.

When both BMC and BFSK Signaling Schemes are supported by a **[USBType-C 1.0]** Port:

- A Source shall first attempt to become Connected with its Port Partner, using BMC; the attempt failing when the Source enters the **PE\_SRC\_Disabled** state for a Source (see Section 8.3.3.2).
- If the Source cannot become Connected using BMC then the Source may attempt to become Connected with its Port Partner using BFSK over  $V_{BUS}$  (re-entering the **PE\_SRC\_Startup** state for a Source see Section 8.3.3.2).
- A Sink shall be able to receive, in the **PE\_SNK\_Wait\_for\_Capabilities** state (see Section 8.3.3.3), a **Source\_Capabilities** Message sent either over the CC wire using BMC or over  $V_{BUS}$  using BFSK.
- The Sink, after the **SinkWaitCapTimer** has timed out in the **PE\_SNK\_Wait\_for\_Capabilities** state (see Section 8.3.3.3), shall issue **Hard Reset** Signaling over both BMC and BFSK.
- Once the Port Partners are Connected they shall continue to use the same signaling scheme, either BMC or BFSK, until a Detach or Hard Reset occurs.
- If either Port Partner issues **Hard Reset** Signaling it shall issue **Hard Reset** Signaling over both BMC and BFSK.

A Source or DFP may communicate using BMC with a Cable Plug regardless of the Signaling Scheme currently being used with its Port Partner.

## 5.9 Built in Self-Test (BIST)

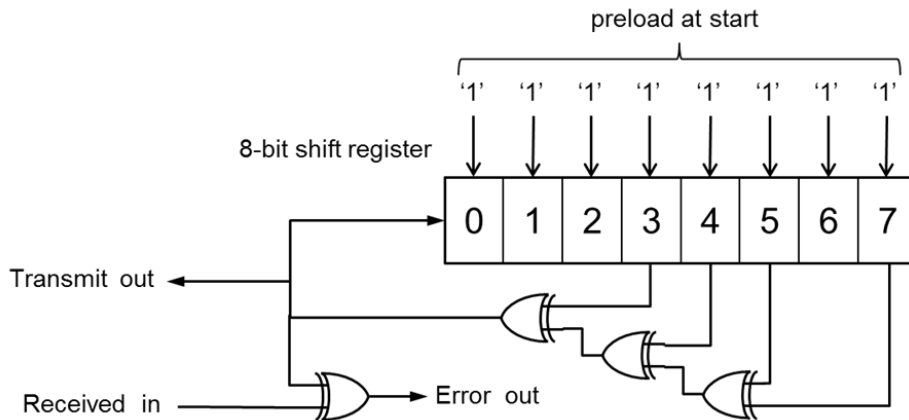
### 5.9.1 BIST PRBS Pattern

The generator polynomial for the PRBS-8 pattern shall be  $x^8 + x^6 + x^5 + x^4 + 1$ .

Figure 5-36 shows an example implementation of the PRBS-generator and checker.

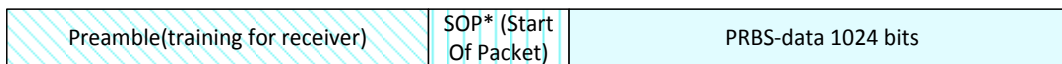
The preloaded pattern shall be "all ones" i.e. all 8 bits in the shift register shall be set to "1". The pattern shall be preloaded when the request to enter test mode is given or received.

Figure 5-36 Example implementation of the BIST generator and checker

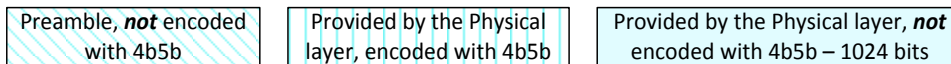


In BIST Transmit or Receiver Test Frames are constructed as shown in Figure 5-37 with a test pattern as defined in Section 5.9.1. Note that the Test Frame does not include an *EOP*. At least *nBISTConfidence* of these Test Frames shall be sent/received without error (see Section 5.9.1.1).

Figure 5-37 Test Frame



LEGEND:



The PRBS data shall be continued without change in the PRBS generator between Test Frames. If the payloads from all Test Frames were concatenated the resulting stream shall look like it was generated directly by the BIST generator.

The Test Frame shall have a fixed length and the only other signaling that shall be recognized in the test mode is the *Hard Reset* Signaling, which shall be used to exit the test mode.

Since the payload length (*nBISTPayload*) and the BIST pattern cycle length are relatively prime, every pattern will eventually appear in every position providing a test of all pattern related weaknesses.

#### 5.9.1.1 Test Frame Transmission

The number of bits transferred needed to demonstrate the required *nBER* (see Table 5-27) at a 99% confidence level is  $4.61 \times 10^6$  (see [Maxim37]). To reach this level of confidence, a minimum of *nBISTConfidence* Test Frames shall be transmitted. To end the test sequence, *Hard Reset* Signaling shall be sent.

If errors are detected more bits shall be sent, see [Maxim37]). The number of Test Frames versus the number of allowable error is given in Table 5-27.

Table 5-27 Allowable Bit Errors vs. Number of Test Frames

N (number of allowable errors)	Minimum number of Test Frames required for Confidence level 99%
0	4502
1	6483
2	8209
3	9810
4	11333
5	12802
6	14230
7	15625
8	16995
10	19673
15	26117
20	32328
30	44337

#### 5.9.1.2 Error Counters

The UUT shall maintain a count of errors detected *BISTErrorCounter* (see Section 6.6.5). The number of errors shall be compared to the number of errors expected from the number of sent bits and the allowed error rate. Typical testing would take place at each supported voltage and in the presence of an acceptable noise level.

#### 5.9.2 BIST Carrier Mode 0

In *BIST Carrier Mode 0*, the Physical Layer shall send out a continuous string of "0"s. This produces a continuous frequency that will allow measurement of *fCarrier - fDeviation*.

#### 5.9.3 BIST Carrier Mode 1

In *BIST Carrier Mode 1*, the Physical Layer shall send out a continuous string of "1"s. This produces a continuous frequency that will allow measurement of *fCarrier + fDeviation*.

#### 5.9.4 BIST Carrier Mode 2

In *BIST Carrier Mode 2*, the Physical Layer shall send out a continuous string of alternating "1"s and "0"s. This enables the measurement of power supply noise and frequency drift.

#### 5.9.5 BIST Carrier Mode 3

In *BIST Carrier Mode 3*, the Physical Layer shall send out a continuous string of sixteen "1"s, followed by sixteen "0"s, followed by sixteen "1"s, etc. This enables the measurement of *fCarrier*.

### 5.9.6 BIST Eye Pattern

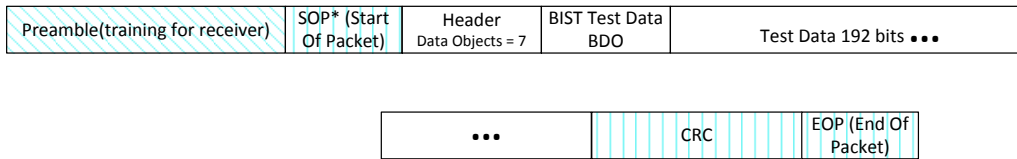
In **BIST Eye Pattern**, the Physical Layer shall send out a continuous string of bits in accordance with Section 5.9.1. This produces a signal that will allow measurement of the eye pattern and of the spectrum mask.

### 5.9.7 BIST Test Data

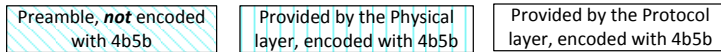
A **BIST Test Data** Message is used by the Tester to send various Tester generated test patterns to the UUT in order to test the UUT's receiver.

Figure 5-38 shows the Test Data Frame which shall be sent by the Tester to the UUT. The **BIST** Message, with a **BIST Test Data BIST** Data Object consists of a Preamble, followed by **SOP\***, followed by the **Message** Header with a data length of 7 Data Objects, followed ~~by the BIST Request Object containing a BIST Test Data BIST~~ Data Object, followed by 6 Data Objects containing Test data, followed by the CRC and then an **EOP**.

Figure 5-38 Test Data Frame



LEGEND:



### 5.9.8 BIST Parameters

Table 5-28 BIST Parameters

Parameter	Description	Min	Nom	Max	Units	Comment
<b>nBISTConfidence</b>	Number of Test Frames to transmit in order to reach 99% confidence level	4502			Frames	
<b>nBISTPayload</b>	Number of bits in a BIST Test Frame payload	1024		1024	Bits	

### 5.9.9 BIST Test Applicability

Table 5-29 shows the BIST Modes which shall be supported, depending on the Signaling Scheme (BMC or BFSK) supported by a device.

Table 5-29 BIST Mode support

Test	BFSK	BMC	Comment
<b>BIST Receiver Mode</b>	√		
<b>BIST Transmit Mode</b>	√		
<b>BIST Eye Pattern</b>	√		
<b>BIST Carrier Mode 0</b>	√		
<b>BIST Carrier Mode 1</b>	√		
<b>BIST Carrier Mode 2</b>	√	√	

Test	BFSK	BMC	Comment
<i>BIST Carrier Mode 3</i>	√		
<i>BIST Test Data</i>	√	√	

## 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 2.5 for an overview of the theory of operation of USB Power Delivery.

### 6.2 Messages

This specification defines two 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

#### 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 High Level Message Structure

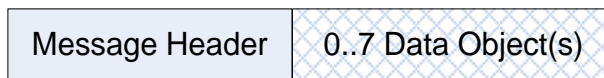
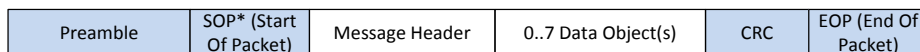


Figure 6-2 illustrates the Message as part of a Packet showing the parts are provided by the Protocol and PHY Layers.

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



Legend:



### 6.2.1.1 Message Header

Every Message shall start with a 16-bit **Message Header**, as shown in Figure 6-1 and defined in Table 6-1. The **Message Header** contains the basic information used by about the Message and the Physical Layer to send the message to its PD Port **Partner 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)	Signaling Start of Packet	Field Name	Notes
15	N/A	Reserved	Shall be set to 0
14..12	SOP*	<b>Number of Data Objects</b>	See Section 6.2.1.2
11..9	SOP*	<b>MessageID</b>	See Section 6.2.1.3
8	SOP only	<b>Port Power Role</b>	See Section 6.2.1.4
	SOP'/SOP''	<b>Cable Plug</b>	See Section 6.2.1.7
7..6	SOP*	<b>Specification Revision</b>	See Section 6.2.1.5
5	SOP only	<b>Port Data Role</b>	See Section 6.2.1.6
	SOP'/SOP''	Reserved	Shall be set to 0
4	N/A	Reserved	Shall be set to 0
3..0	SOP*	<b>Message Type</b>	See Section 6.2.1.8

### 6.2.1.2 Number of Data Objects

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.

### 6.2.1.3 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: during BIST, when sending Test Frames, the **MessageID** is not incremented by the sender and is Ignored by the receiver.

The **MessageID** field shall apply to all SOP\* Packet types.

### 6.2.1.4 Port Power Role

The 1-bit **Port Power Role** field shall indicate the current power role of the Port:

- 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 **Request** 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 Figure 8-6 and Figure 8-7).



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 Figure 8-6 and Figure 8-7).

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 no error recovery action shall be taken if it is incorrect.

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

#### 6.2.1.5 Specification Revision

The 2-bit **Specification Revision** field shall indicate the revision of the Power Delivery Specification supported by the Device.

- 00b –Revision 1.0
- 01b –Revision 2.0
- 10b - 11b – Reserved, shall not be used

On receipt of a Message Header with a higher revision number than that supported, a Port shall respond using the highest revision number it supports.

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

#### 6.2.1.6 Port Data Role

The 1-bit **Port Data Role** field shall indicate the current data role of the Port:

- 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.

Should a Type-C Port receive a Message with the **Port Data Role** field set to the same Data Role as its current Data Role, except for the **GoodCRC** Message, Type-C error recovery actions as defined in [USBType-C 1.0] shall be performed.

For a 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 Rd asserted and 1b for a Port with Rp asserted.

#### 6.2.1.7 Cable Plug

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

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

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

#### 6.2.1.8 Message Type

The 4-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 6-2 (Control Message) or Table 6-3 (Data Message).

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

### 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 0. 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 3..0) and are summarized in Table 6-2. 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 *SignalingStart 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 6-2 Control Message Types

Bits 3..0	Message Type	Sent by	Description	Valid <i>SignalingStart of Packet</i>
0000	Reserved	N/A	All values not explicitly defined are Reserved and shall not be used.	
0001	<i>GoodCRC</i>	Source, Sink or Cable Plug	See Section 6.3.1.	SOP*
0010	<i>GotoMin</i>	Source only	See Section 6.3.2.	SOP only
0011	<i>Accept</i>	Source, Sink or Cable Plug	See Section 6.3.3.	SOP*
0100	<i>Reject</i>	Source or Sink	See Section 6.3.4.	SOP only
0101	<i>Ping</i>	Source only	See Section 6.3.5.	SOP*
0110	<i>PS_RDY</i>	Source or Sink	See Section 6.3.6.	SOP only
0111	<i>Get_Source_Cap</i>	Source or Sink	See Section 6.3.7.	SOP only
1000	<i>Get_Sink_Cap</i>	Source or Sink	See Section 6.3.8.	SOP only
1001	<i>DR_Swap</i>	Source or Sink	See Section 6.3.9	SOP only
1010	<i>PR_Swap</i>	Source or Sink	See Section 6.3.10.	SOP only
1011	<i>VCONN_Swap</i>	<del>DPP</del> Source or Sink	See Section 6.3.11	SOP only
1100	<i>Wait</i>	Source or Sink	See Section 6.3.12	SOP only
1101	<i>Soft_Reset</i>	Source or Sink	See Section 6.3.13.	SOP*
1110-1111	Reserved	N/A	All values not explicitly defined are Reserved and shall not be used.	

#### 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 during the data stream (see Section 6.4.3).

Field Code Changed

#### 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 may 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 as a means to harvest power in order 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. In essence, 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). 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 8-6.

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 to signal the Sink that the Source is willing to meet the *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 *Soft\_Reset* Message to indicate that it has completed its Soft Reset.

The *Accept* Message shall be sent within *tReceiverResponse* of the receipt of the last bit of the Message (see Section 1.1.1.1).

### 6.3.4 Reject Message

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

- It shall be sent to signal the Sink 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 by the recipient of a *PR\_Swap* Message to indicate it is unable to do a Power Role Swap ~~at this time.~~
- It shall be sent by the recipient of a *DR\_Swap* Message to indicate it is unable to do a Data Role Swap ~~at this time.~~
- It shall be sent by the recipient of a *VCONN\_Swap* Message to indicate it is unable to do a VCONN Swap ~~at this time.~~
- It shall be sent by a Source without Dual-Role capability in response to a *Get\_Sink\_Cap* Message.
- It shall be sent by a Sink without Dual-Role capability in response to a *Get\_Source\_Cap* Message.

The *Reject* Message shall be sent within *tReceiverResponse* of the receipt of the last bit of Message (see Section 1.1.1.1).

### 6.3.5 Ping Message

#### 6.3.5.1 Pings on Type-A and Type-B connectors

The *Ping* Message is used on Type-A and Type-B connectors to determine the continued presence of the Sink when no other messaging is taking place (see Figure 8-38 in Section 8.3). The *Ping* Message is sent periodically, every *tSourceActivity*, by the Source to the Sink.

Once a Contract is established Sources shall periodically send the *Ping* Message every *tSourceActivity* after the last Message has been sent/received except when:

- The system is not operating in USB Power Delivery Mode (i.e. in standard [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USBType-C 1.0\]](#) or [\[BC 1.2\]](#) operation).
- A Provider or Provider/Consumer is operating as a Source at *vSafe5V* in the *PE\_SRC\_Ready* state (i.e. power negotiation has already taken place see Section 8.3.3.2.6).

#### 6.3.5.2 Pings on Type-C Connectors

Type-C connectors have an alternative mechanism to determine Sink presence so when the Port Partners are both connected using Type-C connectors the *Ping* Message is not necessary. A Sink using a Type-C connector shall not 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) to indicate its power supply has reached the desired operating condition. See Section 8.3.2.3 or Section 8.3.2.7.1.2 for examples of use.

#### 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 capability of its Port Partner (e.g. Dual-Role capable). The Port shall respond by returning a *Source\_Capabilities* Message (see Section 6.4.1.1.1). A Sink Port, without Dual-Role capability, shall return a *Reject* Message.

#### 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 capability of its Port Partner (e.g. Dual-Role capable). The Port shall respond by returning a *Sink\_Capabilities* Message (see Section 6.4.1.1.2). A Source Port, without Dual-Role capability, shall return a *Reject* Message.

#### 6.3.9 DR\_Swap Message

The *DR\_Swap* Message is used to exchange DFP and UFP operation between Port Partners both utilizing Type-C connectors while maintaining the direction of power flow over  $V_{BUS}$ . The DR\_Swap process can be used by [\[USBType-C 1.0\]](#) Port Partners whether or not they support USB Communications capability. A DFP that supports USB Communication Capability starts as the USB Host on attachment of [\[USBType-C 1.0\]](#) Ports. A UFP that supports USB Communication Capability starts as the USB Device on attachment of [\[USBType-C 1.0\]](#) The DFP is the USB Host; the UFP is the USB Device Ports.

The DFP is the "bus master" for Vendor Defined Message communications so the *DR\_Swap* Message is used to exchange the Port Partner responsible for this communication.

[\[USBType-C 1.0\]](#) DRPs shall have the capability to perform a Data Role Swap from the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* states. [\[USBType-C 1.0\]](#) DFPs and [\[USBType-C 1.0\]](#) 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/reattach in relation to any USB communication which is ongoing between the Port Partners. If there are any Active Modes 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). If the Cable Plug has any Active Modes then the DFP shall not issue a *DR\_Swap* Message and shall cause all Active Modes shall 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, *Reject* Message or *Wait* Message.

- 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, Type-C error recovery actions, such as disconnect, as defined in [*USBType-C 1.0*] will be necessary.

See Sections 0, 8.3.3.6.2.1 and 1.1.1.1.1.1.1 for further details.

### 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, *Reject* Message or *Wait* Message.

- 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*.

Since a UFP Source can attempt to send a *Discover Identity Command* using SOP' to a Cable Plug prior to the establishment of an Explicit Contract, a DFP Sink shall disable the receiving of SOP' Messages until an Explicit Contract has been established. This ensures that only the Cable Plug responds with a *GoodCRC Message* to the *Discover Identity Command*.

For Type-C Ports the Source shall have  $R_p$  asserted on the CC wire and the Sink shall have  $R_d$  asserted on the CC wire. When performing a Power Role Swap from Source to Sink, a Type-C Port shall change ~~is its~~ CC Wire resistor from  $R_p$  to  $R_d$ . When performing a Power Role Swap from Sink to Source, a Type-C Port shall change is CC Wire resistor from  $R_d$  to  $R_p$ . The DFP (Host), UFP (Device) roles and ~~VCONN~~ Source ~~of VCONN~~ shall remain unchanged during the Power Role Swap process.

For more information regarding the Power Role Swap, refer to Sections 7.3.9 and 7.3.10 in the Power Supply chapter, Sections 8.3.2.7.1, 8.3.2.8.1, 8.3.3.6.3.1 and 8.3.3.6.3.2 in the Device Policy chapter and Section 9.1.2 for  $V_{BUS}$  mapping to USB states.

### 6.3.11 *VCONN\_Swap* Message

The *VCONN\_Swap* Message shall be supported by any Type-C Port that that utilizes the SSTX and SSRX pins and supports the *PR\_Swap* Message.

The *VCONN\_Swap* Message may ~~only~~ be sent by ~~the DFP~~ either Port Partner to request an exchange of VCONN ~~power sources~~ Source. The ~~UFP~~ recipient of the Message shall respond by sending an *Accept* Message, *Reject* Message or *Wait* Message.

- If an *Accept* Message is sent, the ~~VCONN source and sink~~ Port Partners shall ~~deperform~~ 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.
- 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.

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.

Note: VCONN shall be continually sourced during the VCONN Swap process in order to maintain power to the Cable Plug(s) i.e. make before break.

### 6.3.12 Wait Message

The *Wait* Message is a valid response to a *Request*, a ~~*PR\_Swap*~~ ~~or~~ *DR\_Swap* ~~or~~ *VCONN\_Swap* Message.

- It shall be sent to signal the Sink that the Source is unable to meet the request at this time.
- It shall be sent by the recipient of a *PR\_Swap* Message to indicate it is unable to do a Power Role Swap at this time.
- It shall be sent by the recipient of a *DR\_Swap* Message to indicate it is unable to do a Data Role Swap at this time.
- It shall be sent by the ~~UFP on receiving~~ recipient of a *VCONN\_Swap* Message to indicate it is unable to do a VCONN Swap at this time.

The *Wait* Message shall be sent within *tReceiverResponse* of the receipt of the last bit of the Message (see Section 1.1.1.1).

#### 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 shall 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* to ensure that there is *tSinkRequest* between requests.

#### 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 intrusive mode or in hybrid mode when a request cannot be satisfied (see Section 9.4.5). In this case the Hub shall send a *Wait* Message in response to a *PR\_Swap* Message from its Port Partner. Once the System Policy Manager has completed its evaluation of the request the Hub should initiate the Power Role Swap process again by sending a *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 Data 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.

### 6.3.12.4 Wait in response to a VCONN\_Swap Message

The *Wait* Message is used ~~by the UFP~~ 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 ~~UFP device~~ receiving the *VCONN\_Swap* Message needs to evaluate the request further. ~~Sender of the VCONN\_Swap DFP Message~~ should initiate the VCONN Swap process again at a future time by resending a *VCONN\_Swap* 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 6.7.1). 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 in order 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).~~

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 6.7).

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.

If, after a Power or Data Role Swap, a different Port starts to communicate with a given Cable Plug, the Cable Plug's Protocol Layer needs to be reset in order to ensure *MessageID* synchronization. If the Source or DFP 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 Source or DFP 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.

## 6.4 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 ~~four~~several types of Data Objects:

- ~~• 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~~
- BIST Data Object (BDO) used for PHY Layer compliance testing
- ~~• 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

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 6-3. 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 ~~Signaling~~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 6-3 Data Message Types

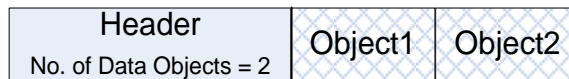
Bits 3..0	Type	Sent by	Description	Valid SignalingStart of Packet
0000	Reserved		All values not explicitly defined are Reserved and shall not be used.	
0001	<i>Source_Capabilities</i>	Source or Dual-Role operating as Sink	See Section 6.4.1.2	SOP only
0010	<i>Request</i>	Sink only	See Section 6.4.2	SOP only
0011	<i>BIST</i>	Tester, Source or Sink	See Section 6.4.3	SOP*
0100	<i>Sink_Capabilities</i>	Sink or Dual-Role	See Section 6.4.1.3	SOP only
0101-1110	Reserved		All values not explicitly defined are Reserved and shall not be used.	
1111	<i>Vendor_Defined</i>	Source, Sink or Cable Plug	See Section 6.4.4	SOP*

### 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.

Figure 6-3 Example Capabilities Message with 2 Power Data Objects



In Figure 6-3, the *Number of Data Objects* field is 2: *vSafe5V* plus one other voltage.

Power Data Objects (PDO) 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 the most commonly 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 than can be directly connected to  $V_{BUS}$ .

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 in order 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, in order 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 Section 3.1.1 and [USBType-C 1.0]). The Source shall limit its offered capabilities to the maximum current supported by its receptacle and attached plug. A Sink with a Type-A or a Type-B receptacle shall limit its requested capabilities to the maximum current supported by its receptacle and attached plug. A Sink with a Type-C receptacle may make a request from any of the capabilities offered by the Source. For further details see Section 4.4.

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 a number of 32-bit Power Data Objects (see Table 6-4).

Table 6-4 Power Data Object

Bit(s)	Description	
B31..30	Value	Parameter
	00b	Fixed supply ( $V_{min} = V_{max}$ )
	01b	Battery
	10b	Variable Supply (non-battery)
	11b	Reserved
B29..0	Specific Power Capabilities are described by the PDOs 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) either as part of advertising Port capabilities, or in response to a *Get\_Source\_Cap* Message.

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 in the following cases:

- By the Source Port from the PE\_SRC\_Ready state upon a change in its ability to supply power
- By a Source Port or Dual-Role Port in response to a *Get\_Source\_Cap* Message

##### 6.4.1.1.2 Use by Sinks

Sinks send a *Sink\_Capabilities* Message (see Section 6.4.1.3) in response to a *Get\_Sink\_Cap* Message.

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 devices

Dual-Role devices send a *Source\_Capabilities* Message (see Section 6.4.1) as part of advertising Port capabilities when operating in Source role. Dual-Role devices send a *Source\_Capabilities* Message (see Section 6.4.1) in response to a *Get\_Source\_Cap* Message regardless of their present operating role. Similarly Dual-Role devices send a *Sink\_Capabilities* Message (see Section 6.4.1.3) 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 6-4) as part of a *Source\_Capabilities* Message (see Figure 6-3). Power Data Objects are used to convey a Source Port's capabilities to provide power including Dual-Role 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. 12V) 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. DRPs that support VCONN Powered Accessories do not source  $V_{BUS}$  (see [USBType-C 1.0]) however when sourcing VCONN they shall advertise *vSafe5V* with the Maximum Current set to 0mA in the first Power Data Object.

A Sink 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.

##### 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 — Receipt of Unexpected Source Capabilities Message

##### 6.4.1.2.2 Non-Standard attachment of Type-A Ports

When a Provider/Consumer Port, with a Type-A receptacle, that is operating as a Source receives an unexpected *Source\_Capabilities* Message (e.g. one not in response to a *Get\_Source\_Cap* Message), it shall silently remove *vSafe5V* from  $V_{BUS}$  within *tPSSourceOff* of receiving the *Source\_Capabilities* Message *EOP*. Reception of a *Source\_Capabilities* Message indicates that the user has attached two Type-A Ports together via non-standard cabling so  $V_{BUS}$  is removed to prevent back-powering.

Table 6-5 shows the behavior of various combinations of ~~some of the Port types~~ Type-A Ports when attached in a non-standard configuration.

Table 6-5 ~~Port~~Type-A to Type-A Port Behavior

Port 1	Port 2	Description
Provider/Consumer	Provider/Consumer	Either Port 1 or Port 2 will remove $V_{BUS}$ depending on which sends its Capabilities first.
Provider/Consumer	Provider	Port 1 will silently remove <i>vSafe5V</i> from $V_{BUS}$
Provider	Provider	Safe operation (see Section 7.1.8).
Provider/Consumer	Legacy host	Safe operation (see Section 7.1.8).
<del>Consumer/Provider</del>	<del>Consumer/Provider</del>	<del>Undefined.</del>
Legacy host	Legacy host	Outside scope of this specification

#### 6.4.1.2.3 Source Fixed Supply Power Data Object

Table 6-6 describes the Fixed Supply (00b) PDO. See ~~Chapter~~Section 7 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 through 25. All other Fixed Supply Power Data Objects shall set bits 29...22 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 device which offers no capabilities in its default role or when external power is required in order 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 6-6 Fixed Supply PDO - Source

Bit(s)	Description
B31..30	Fixed supply
B29	Dual-Role Power
B28	USB Suspend Supported
B27	Externally Powered
B26	USB Communications Capable
B25	Data Role Swap
B24..22	Reserved – shall be set to zero.
B21..20	Peak Current
B19..10	Voltage in 50mV units
B9..0	Maximum Current in 10mA units

##### 6.4.1.2.3.1 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.

##### 6.4.1.2.3.2 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 3.1*], [*USBType-C 1.0*] or [*BC 1.2*]. After a Contract has been negotiated:

- If the USB Suspend Supported flag is set, then the Sink shall follow the [\[USB 2.0\]](#) or [\[USB 3.1\]](#) 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 7.2.4).
- If the USB Suspend Supported flag is cleared, then the Sink shall ~~ignore~~[not apply](#) the [\[USB 2.0\]](#) or [\[USB 3.1\]](#) 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).

#### 6.4.1.2.3.3 Externally Powered

The Externally Powered bit shall only be set when an AC “wall” supply is providing 100% of the Source’s power.

This means that either:

- There is an AC supply, e.g. a wall wart, directly connected to the Source.
- Or, in the case of a PDUSB Hub:
  - The Hub is receiving 100% of its power from a PD Source with its Externally Powered bit set.
  - The Hub is receiving 100% of its power from multiple PD Sources all with their Externally Powered bits set.

#### 6.4.1.2.3.4 USB Communications Capable

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

#### 6.4.1.2.3.5 Data Role Swap

The Data Role Swap bit shall be set when the Port is a Type-C ~~DRP~~[DRP](#) (see [\[USBType-C 1.0\]](#)) and supports the *DR\_Swap* Message. This is a static capability which shall remain fixed for a given device.

#### 6.4.1.2.3.6 Peak Current

The USB Power Delivery Fixed Supply is only required to deliver the amount of current requested in the Operating Current (*I<sub>oc</sub>*) field of an RDO. In some usages however, for example computer systems, where there are short bursts of activity, it may 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) 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 6-7). Supplies that do not support an overload capability shall set these bits to 00b in the corresponding Fixed Supply PDO.

**Table 6-7 Fixed Power Source Peak Current Capability**

Bits 21..20	Description
00	Peak current equals <i>I<sub>oc</sub></i> (default)
01	Overload Capabilities: <ol style="list-style-type: none"> <li>1. Peak current equals 150% <i>I<sub>oc</sub></i> for 1ms @ 5% duty cycle (low current equals 97% <i>I<sub>oc</sub></i> for 19ms)</li> <li>2. Peak current equals 125% <i>I<sub>oc</sub></i> for 2ms @ 10% duty cycle (low current equals 97% <i>I<sub>oc</sub></i> for 18ms)</li> <li>3. Peak current equals 110% <i>I<sub>oc</sub></i> for 10ms @ 50% duty cycle (low current equals 90% <i>I<sub>oc</sub></i> for 10ms)</li> </ol>

Bits 21..20	Description
10	Overload Capabilities: <ol style="list-style-type: none"> <li>1. Peak current equals 200% <math>I_{OC}</math> for 1ms @ 5% duty cycle (low current equals 95% <math>I_{OC}</math> for 19ms)</li> <li>2. Peak current equals 150% <math>I_{OC}</math> for 2ms @ 10% duty cycle (low current equals 94% <math>I_{OC}</math> for 18ms)</li> <li>3. Peak current equals 125% <math>I_{OC}</math> for 10ms @ 50% duty cycle (low current equals 75% <math>I_{OC}</math> for 10ms)</li> </ol>
11	Overload Capabilities: <ol style="list-style-type: none"> <li>1. Peak current equals 200% <math>I_{OC}</math> for 1ms @ 5% duty cycle (low current equals 95% <math>I_{OC}</math> for 19ms)</li> <li>2. Peak current equals 175% <math>I_{OC}</math> for 2ms @ 10% duty cycle (low current equals 92% <math>I_{OC}</math> for 18ms)</li> <li>3. Peak current equals 150% <math>I_{OC}</math> for 10ms @ 50% duty cycle (low current equals 50% <math>I_{OC}</math> for 10ms)</li> </ol>

#### 6.4.1.2.4 Variable Supply (non-battery) Power Data Object

Table 6-9 describes a Variable Supply (non-battery) (10b) PDO for a Source. See Section 7 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 actually 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.

Table 6-8 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

#### 6.4.1.2.5 Battery Supply Power Data Object

Table 6-9 describes a Battery (01b) PDO for a Source. See ~~Chapter~~Section 7 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, only the Battery PDO uses power instead of current.

The Sink may monitor the battery voltage.

Table 6-9 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

#### 6.4.1.3 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 6-4). These are returned as part of a *Sink Capabilities* Message in response to a *Get\_Sink\_Cap* Message (see Figure 6-3).

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 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. 12V). 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 6-10 describes the Sink Fixed Supply (00b) PDO. See Chapter 0 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 6-10 Fixed Supply PDO - Sink**

Bit(s)	Description
B31..30	Fixed supply
B29	Dual-Role Power
B28	Higher Capability
B27	Externally Powered
B26	USB Communications Capable
B25	Data Role Swap
B24..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 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.

##### 6.4.1.3.1.2 Higher Capability

In the case that the Sink needs more than **vSafe5V** (e.g. 12V) to provide full functionality, then the Higher Capability bit shall be set.

##### 6.4.1.3.1.3 Externally Powered

The Externally Powered bit shall only be set when an AC "wall" supply is providing 100% of the Sink's power.

This means that either:

- There is an AC supply, e.g. a wall wart, directly connected to the Sink.
- Or, in the case of a PDUSB Hub:
  - The Hub is receiving 100% of its power from a PD Source with its Externally Powered bit set.
  - The Hub is receiving 100% of its power from multiple PD Sources all with their Externally Powered bits set.

#### 6.4.1.3.1.4 USB Communications Capable

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

#### 6.4.1.3.1.5 Data Role Swap

The Data Role Swap bit shall be set when the Port is a Type-C DRP (see [\[USBType-C 1.0\]](#)) and supports the *DR\_Swap* Message. This is a static capability which shall remain fixed for a given device.

#### 6.4.1.3.2 Variable Supply (non-battery) Power Data Object

Table 6-11 describes a Variable Supply (non-battery) (10b) PDO used by a Sink. See Section 7 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 6-11 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 6-12 describes a Battery (01b) PDO used by a Sink. See [ChapterSection 7](#) 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 6-12 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

## 6.4.2 Request Message

A **Request** Message shall be sent by a Sink to request power, typically during the request phase of a 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 8.3.2.3). A **Request** Message shall return one and only one Sink Request Data Object that shall identify the Power Data Object being requested.

The **Request** Message includes the requested power level. For example, if the **Source\_Capabilities** Message includes a Fixed Supply PDO that offers 12V @ 1.5A and if the Sink only wants 12V @ 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 takes one of two forms depending on the kind of power requested. The Fixed Power Data Object and Variable Power Data Object share a common format (see Table 6-13 and Table 6-14). The Battery Power Data Object uses a different format (see Table 6-15 and Table 6-16).

**Table 6-13 Fixed and Variable Request Data Object**

Bits	Description
B31	Reserved - shall be set to zero
B30..28	Object position (000b is Reserved <b>and shall not be used</b> )
B27	GiveBack flag = 0
B26	Capability Mismatch
B25	USB Communications Capable
B24	No USB Suspend
B23..20	Reserved - shall be set to zero.
B19..10	Operating current in 10mA units
B9..0	Maximum Operating Current 10mA units

**Table 6-14 Fixed and Variable Request Data Object with GiveBack Support**

Bits	Description
B31	Reserved - shall be set to zero
B30..28	Object position (000b is Reserved <b>and shall not be used</b> )
B27	GiveBack flag = 1
B26	Capability Mismatch
B25	USB Communications Capable
B24	No USB Suspend
B23..20	Reserved - shall be set to zero.
B19..10	Operating Current in 10mA units
B9..0	Minimum Operating Current 10mA units

**Table 6-15 Battery Request Data Object**

Bits	Description
B31	Reserved - shall be set to zero
B30..28	Object position (000b is Reserved <b>and shall not be used</b> )
B27	GiveBackFlag = 0
B26	Capability Mismatch
B25	USB Communications Capable
B24	No USB Suspend
B23..20	Reserved - shall be set to zero.
B19..10	Operating Power in 250mW units



Bits	Description
B9..0	Maximum Operating Power in 250mW units

Table 6-16 Battery Request Data Object with GiveBack Support

Bits	Description
B31	Reserved – shall be set to zero
B30..28	Object position (000b is Reserved, <b>and shall not be used</b> )
B27	GiveBackFlag = 1
B26	Capability Mismatch
B25	USB Communications Capable
B24	No USB Suspend
B23..20	Reserved - shall be set to zero.
B19..10	Operating Power in 250mW units
B9..0	Minimum Operating Power in 250mW units

#### 6.4.2.1 Object Position

The value in the Object Position field shall indicate which object in the *Source\_Capabilities* Message the RDO refers. The value 1 always indicates the 5V Fixed Supply PDO as it is the first object following the *Source\_Capabilities* Message Header. The number 2 refers to the next PDO and so forth.

#### 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 Sink cannot satisfy its power requirements from the capabilities offered by the Source. In this case the Sink shall make a valid request from the offered capabilities and shall set the Capability Mismatch bit (see Section 8.2.5.2).

When a Sink returns a Request Data Object in response to advertised capabilities with this bit set, it indicates that the Sink wants power that the Source cannot provide. This may be due to either a voltage that is not available or the amount of available current. At this point the Source can use the information in the *Request* Message combined with the contents of the *Sink\_Capabilities* Message to ascertain the Voltage and Current required by the Sink for full operation.

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 *Source\_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 ~~or equal to~~ 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 [\[USB 2.0\]](#) or [\[USB 3.1\]](#) 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 [\[USB 2.0\]](#) or [\[USB 3.1\]](#) 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 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, 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. This field shall apply to the Fixed and Variable RDO.

#### 6.4.2.7 Maximum Operating Current

The Maximum Operating Current field in the 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). 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 Sections 6.3.12.1 and 8.2.5.1).

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 for more details of the usage of the Capabilities Mismatch bit.

This field shall apply to the Fixed and Variable RDO.

#### 6.4.2.8 Minimum Operating Current

The Minimum Operating Current field in the 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 ~~or equal to~~ the Minimum Operating Current value.

This field shall apply to the Fixed and Variable RDO.

#### 6.4.2.9 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.

#### 6.4.2.10 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 Sections 6.3.12.1 and 8.2.5.1).

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 for more details of the usage of the Capabilities Mismatch bit.

This field shall apply to the Battery RDO.

#### 6.4.2.11 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.

### 6.4.3 BIST Message

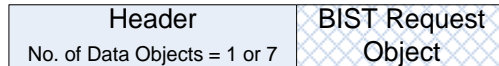
The *BIST* Message is sent to request the Port to enter a Physical Layer test mode that performs one of the following functions:

- Enter the receiver mode which does the following:
  - Zero accumulated BISTErrorCounter
  - Process a series of BIST Test Frames and compare the received data to the expected data pattern
  - Count the number of errors received and add to the BISTErrorCounter
- Enter a transmit mode to send BIST Test Frames to the Tester
- ~~Enter a Continuous BIST Mode to send a continuous stream of test data to the Tester~~
- ~~Send BIST test to the UUT~~

- Return the BISTErrorCounter to the Tester

The Message format is as follows:

Figure 6-4 BIST Message



All ports shall be able to be a Unit Under Test (UUT) only when operating at *vSafe5V*. BIST Modes to be supported are defined in Section 5.9.9. For each supported BIST Mode the following operations shall be implemented based on the reception of the appropriate *BIST* Message *BIST* Data Object (see Table 6-17):

- Process reception of a *BIST Receiver Mode BIST* Data Object
- Process reception of a *BIST Transmit Mode BIST* Data Object
- Generate a *Returned BIST Counters BIST* Data Object response within a *BIST* Message in response to each received Test Frame.
- Process reception of a *BIST Carrier Mode 0 BIST* Data Object that shall result in the generation of the appropriate carrier signal
- Process reception of a *BIST Carrier Mode 1 BIST* Data Object that shall result in the generation of the appropriate carrier signal
- Process reception of a *BIST Carrier Mode 2 BIST* Data Object that shall result in the generation of the appropriate carrier signal.
- Process reception of a *BIST Carrier Mode 3 BIST* Data Object that shall result in the generation of the appropriate carrier signal.
- Process reception of a *BIST Eye Pattern 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.

It is optional for a Port to take on the role of a Tester. ~~All Ports shall support the defined BIST Modes listed in this Section.~~

When a Port receives a *BIST* Message *BIST* Data Object for a BIST Mode when Power Role swapped or 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 that it supports, 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 6.5.8.4).

It is anticipated that dedicated Testers will exist. Those testers may not be required to implement a local receiver test. However, a Tester shall always be able to complete the operations required when a *BIST* Message with *BIST* Data Object *BIST Transmit Mode* is sent by the Tester.

The usage model of the PHY Layer BIST modes generally assumes that some controlling agent will request a test of its Port Partner. A UUT Port minimally has to process a request to enter test mode and return error counters. A Tester Port shall have a means to place the UUT Port into receiver test mode and retrieve the error counters from the UUT. A Port, that is not part of a Tester, is not expected to be the initiator of a receiver test operation, but is not precluded from doing so.

In Section 0 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 6-17.

Table 6-17 BIST Data Object

Bit(s)	Value	Parameter	Description	Reference
B31..28	0000b	<i>BIST Receiver Mode</i>	Requests receiver to enter BIST Receiver Mode	See Section 0
	0001b	<i>BIST Transmit Mode</i>	Requests receiver to enter BIST Transmit Mode	See Section 6.4.3.2
	0010b	<i>Returned BIST Counters</i>	Returned error counters	See Section 6.4.3.3
	0011b	<i>BIST Carrier Mode 0</i>	Requests transmitter to enter BIST Carrier Mode 0	See Section 6.4.3.4
	0100b	<i>BIST Carrier Mode 1</i>	Requests transmitter to enter BIST Carrier Mode 1	See Section 6.4.3.5
	0101b	<i>BIST Carrier Mode 2</i>	Request Transmitter to enter BIST Carrier Mode 2	See Section 6.4.3.6
	0110b	<i>BIST Carrier Mode 3</i>	Request Transmitter to enter BIST Carrier Mode 3	See Section 6.4.3.7
	0111b	<i>BIST Eye Pattern</i>	Requests transmitter to enter BIST Eye Pattern.	See Section 6.4.3.8
	1000b	<i>BIST Test Data</i>	Sends a Test Data Frame.	See Section 6.4.3.9
	1001b-1111b		All values not explicitly defined are Reserved and shall not be used.	
B27..16			Reserved and shall be set to zero.	
B15..0		16-bit unsigned integer	When Request Type is <i>Returned BIST Counters</i> , this field shall contain the contents of <i>BISTErrorCounter</i> otherwise it shall be set to zero.	See Section 6.4.3.3

Note 1: *BIST* messages shall be ignored unless  $V_{BUS}$  is at *vSafe5V* and an Explicit Contract is in place.

#### 6.4.3.1 BIST Receiver Mode

This operation shall be used to initiate a UUT remote receiver test by sending a *BIST* Message containing a *BIST Receiver Mode* BIST Data Object.

On receiving the request, the UUT shall zero its *BISTErrorCounter* and both the Tester and the UUT shall preload their PRBS generator with the designated pattern (see Section 5.9.1).

The receiver (UUT) shall acknowledge the *BIST* Message with a *GoodCRC* Message.

The UUT enters BIST Receiver Mode after the *BIST* Message has been received (see Section 6.5.8.1). At this time the UUT shall be able to receive a Test Frame from the Tester and to respond appropriately with a *BIST* Message with a *BIST* Data Object of *Returned BIST Counters* (see Section 6.5.8.5). After reception of the first Test Frame, further Test Frames are sent at a rate determined by the Tester.

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

#### 6.4.3.2 BIST Transmit Mode

Loopback mode is not possible so BIST Transmit Mode shall be used to request a UUT transmitter test by sending a *BIST* Message containing a *BIST Transmit Mode* BIST Data Object.

Before initiating the request the Tester shall zero its *BISTErrorCounter* and preload the PRBS generator with the designated pattern (see Section 5.9.1). On receiving the request the UUT shall preload the PRBS generator with the designated pattern (see Section 5.9.1).

The receiver in the UUT shall acknowledge the *BIST* Message with a *GoodCRC* Message. The UUT enters BIST Transmit Mode after the *BIST* Message has been received (see Section 6.5.8.1). The UUT shall start transmitting the

first Test Frame no later than  $t_{BISTMode}$  max of the last bit of the *EOP* of the *BIST* Message used to initiate the test is received by the Physical Layer. Each subsequent Test Frame shall be started:

- Either on reception of a *BIST* Message, with a *Returned BIST Counters BIST* Data Object or
- On expiry of the *BISTReceiveErrorTimer* and
- No later than  $t_{BISTResponse}$  after the first bit of the Preamble of the previous Test Frame has been transmitted on  $V_{BUS}$  by the Physical Layer.

The Tester shall preload its PRBS checker with the designated pattern and start counting errors. After receiving a suitable number of Test Frames, the Tester shall freeze its error counter. The UUT shall be reset by sending *Hard Reset* Signaling instead of a *BIST* Message.

#### 6.4.3.3 Returned BIST Counters

The *BIST* Message, with a *Returned BIST Counters BIST* Data Object, shall contain the error counters obtained during the receiver test. During BIST, when sending Test Frames, the *MessageID* of the *BIST* Message, with a *Returned BIST Counters BIST* Data Object shall be Ignored.

#### 6.4.3.4 BIST Carrier Mode 0

Upon receipt of a *BIST* Message, with a *BIST Carrier Mode 0 BIST* Data Object, the UUT shall send out a continuous string of "0"s. This produces a continuous frequency that will allow measurement of  $f_{Carrier} - f_{Deviation}$ .

The UUT shall ~~be reset by an out of band mechanism to exit the Continuous BIST Mode within  $t_{BISTContMode}$  of this Continuous BIST Mode being enabled (see Section 6.5.8.4 mode (e.g. removing power)).~~

#### 6.4.3.5 BIST Carrier Mode 1

Upon receipt of a *BIST* Message, with a *BIST Carrier Mode 1 BIST* Data Object, the UUT shall send out a continuous string of "1"s. This produces a continuous frequency that will allow measurement of  $f_{Carrier} + f_{Deviation}$ .

The UUT shall ~~be reset by an out of band mechanism to exit the Continuous BIST Mode within  $t_{BISTContMode}$  of this Continuous BIST Mode being enabled (see Section 6.5.8.4 mode (e.g. removing power)).~~

#### 6.4.3.6 BIST Carrier Mode 2

Upon receipt of a *BIST* Message, with a *BIST Carrier Mode 2 BIST* Data Object, the UUT shall send out a continuous string of alternating "1"s and "0"s. Note: that in the case that the BMC Signaling Scheme is used the "1"s and "0"s will in addition be BMC encoded.

The UUT shall ~~be reset by an out of band mechanism to exit the Continuous BIST Mode within  $t_{BISTContMode}$  of this Continuous BIST Mode being enabled (see Section 6.5.8.4 mode (e.g. removing power)).~~

#### 6.4.3.7 BIST Carrier Mode 3

Upon receipt of a *BIST* Message, with a *BIST Carrier Mode 3 BIST* Data Object, the UUT shall send out a continuous string of sixteen "1"s, followed by sixteen "0"s.

The UUT shall ~~be reset by an out of band mechanism to exit the Continuous BIST Mode within  $t_{BISTContMode}$  of this Continuous BIST Mode being enabled (see Section 6.5.8.4 mode (e.g. removing power)).~~

#### 6.4.3.8 BIST Eye Pattern

Upon receipt of a *BIST* Message, with a *BIST Eye Pattern BIST* Data Object, the UUT shall send out a continuous string of bits in accordance with section 5.9.1. This produces a signal that will allow measurement of the eye pattern and of the spectrum mask.

The UUT shall ~~be reset by some out exit the Continuous BIST Mode within  $t_{BISTContMode}$  of band mechanism to stop this Continuous BIST Mode being enabled (see Section 6.5.8.4 testing mode (e.g. removing power)).~~

#### 6.4.3.9 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 5.9.7 for the definition of the Test Data Frame.

The UUT shall be ~~reset~~ by ~~sending Hard Resetsome out-of-band mechanism Signaling to stop this testing mode (e.g. removing power)-reset 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* A Port receiving a message that it does not recognize shall ignore the message.~~

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* Message could be used to enable the Source to offer additional power via a *Source\_Capabilities* Message.

The Message format shall be as shown in Figure 6-5.

Figure 6-5 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 6-18. 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 6-19.

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 no Contract or an Implicit Contract (in place after a Power Role Swap) is in place in order to discover Cable capabilities (see Section 8.3.3.10.11). A VDM Message sequence shall not interrupt any other PD Message Sequence. A VDM Message sequence shall be interruptible by any other PD Message Sequence.

##### 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.
- A Port receiving an Unstructured VDM for a VID that it does not recognize shall Ignore the Message.
- 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.
- A DFP or UFP which does not support Unstructured VDMs shall Ignore any Unstructured VDMs received.
- A "command" sequence shall be interruptible e.g. due to the need for a Message sequence using SOP Packets.
- Unstructured VDMs shall only be used during Modal Operation in the context of an Active Mode.
- Unstructured VDMs may be used with SOP\* Packets.

Table 6-18 illustrates the VDM Header bits.

**Table 6-18 Unstructured VDM Header**

Bit(s)	Description	Reference
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 6-19.

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 8.3.3.10.11).
- Only the DFP shall be an Initiator of Structured VDMs except for the *Attention* Command that shall only be initiated by the UFP.
- Only the UFP or a Cable Plug shall be a Responder to Structured VDMs.
- Structured VDMs shall not be initiated or responded to under any other circumstances.
- A DFP or UFP which does not support Structured VDMs shall Ignore any Structured VDMs received.
- A DFP, UFP or Cable Plug which supports Structured VDMs and receiving an Structured VDM for a SVID that it does not recognize shall reply with a NAK Command.
- A Command sequence shall be interruptible e.g. due to the need for a Message sequence using SOP Packets.

**Table 6-19 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



Bit(s)	Field	Description
B14..13	Structured VDM Version	Version Number of the Structured VDM (not this specification Version): <ul style="list-style-type: none"> <li>Version 1.0 = 0</li> <li>Values 1-3 are Reserved <u>and shall not be used</u></li> </ul>
B12..11	Reserved	<u>For Commands 0..15</u> shall be set to 0 and shall be Ignored <u>SVID Specific Commands (16..31) defined by the SVID.</u>
B10..8	Object Position	For the <i>Enter Mode</i> , <i>Exit Mode</i> and <i>Attention</i> Commands: <ul style="list-style-type: none"> <li>000b = Reserved <u>and shall not be used.</u></li> <li>001b..110b = Index into the list of VDOs to identify the desired Mode VDO</li> <li><u>111b = Exit all <i>Active</i> Modes</u> (equivalent of a power on reset). Shall <u>not only</u> be used with the <i>Exit Mode</i> Command.</li> </ul> <u>Commands 0..3, 7..15:</u> <ul style="list-style-type: none"> <li><u>000b</u></li> <li><u>001b..111b = Reserved and shall not be used.</u></li> </ul> <u>SVID Specific Commands (16..31) defined by the SVID.</u>
B7..6	Command Type	00b = Initiator 01b = Responder ACK 10b = Responder NAK 11b = Responder BUSY
B5	Reserved	Shall be set to 0 and shall be Ignored
B4..0	Command <sup>1</sup>	0 = Reserved, shall not be used 1 = <i>Discover Identity</i> 2 = <i>Discover SVIDs</i> 3 = <i>Discover Modes</i> 4 = <i>Enter Mode</i> 5 = <i>Exit Mode</i> 6 = <i>Attention</i> 7-15 = Reserved, shall not be used 16..31 = SVID Specific Commands
Note 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 6-20 shows the Commands, which SVID to use with each Command and which the only SOP\* may values which shall be used.

Table 6-20 Structured VDM Commands

Command	VDM Header SVID Field	SOP* used
<i>Discover Identity</i>	Shall only use the <i>PD SID</i> .	<del>Valid with</del> <u>Shall only use SOP/SOP'</u>
<i>Discover SVIDs</i>	Shall only use the <i>PD SID</i> .	<del>Shall only use SOP/SOP'</del> <u>Valid with-</u>
<i>Discover Modes</i>	Valid with any SVID.	<del>Shall only use SOP/SOP'</del> <u>Valid with-</u>
<i>Enter Mode</i>	Valid with any SVID.	Valid with <i>SOP*</i> .
<i>Exit Mode</i>	Valid with any SVID.	Valid with <i>SOP*</i> .
<i>Attention</i>	Valid with any SVID.	Valid with <i>SOP</i> .

Command	VDM Header SVID Field	SOP* used
SVID Specific Commands	Valid with any SVID.	Valid with <i>SOP*</i> (defined by <a href="#">SVID</a> ).

#### 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 6-21 lists specific SVID values referenced by this specification.

Table 6-21 SVID Values

Parameter	Value	Description
<i>PD SID</i>	0xFF00	Standard ID allocated to this specification.
<i>VID Unassigned</i>	0x0000	<del>Manufacturer does not have a USB-IF assigned Vendor ID e.g. because they do not manufacture USB communications capable devices. Shall only be returned in a Structured VDM <i>Discover Identity</i> ACK Command response as part of the ID Header (see Section 6.4.4.3.1.1).</del>

#### 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 field indicates 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. At this time, there is only one version (1.0) defined. This field shall be set to zero to indicate Version 1.0.

On receipt of a VDM ~~message~~ Header with a higher Version number than that supported, a Port shall respond using the highest Version number it supports.

#### 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 ~~Initiator/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 when not required by the Command.

#### 6.4.4.2.5 Command Type

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

An Initiator shall set the field to "Initiator" to indicate that this is a Command request from an Initiator.

"Responder ACK" is the normal return ~~indicating and shall be sent to indicate~~ that the Command request was received and handled normally.

"Responder NAK" ~~is shall be~~ returned when the Command request ~~was not understood~~ ~~has an invalid parameter (e.g. invalid SVID or couldMode) or cannot~~ not be acted upon ~~at this time (e.g. a Mode which has a dependency on another Mode)~~. The handling of "Responder NAK" is left up to the Initiator.

“Responder BUSY” ~~is~~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 DFP shall wait *tVDMBusy* after a “Responder BUSY” response is received before retrying the Command request.

#### 6.4.4.2.6 Command

This field is used to identify devices and manage their operational Modes. The commands defined in this specification shall be used to manage Modes on the USB Type-C connector. There is a further range of Command values left for the vendor to use to manage additional extensions.

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 sent by the Initiator in reply to the Responder’s response.

If the Structured VDM Command request is not recognized it shall be NAKed.

#### 6.4.4.3 Use of Commands

The *VDM Header for a Structured VDM HeaderMessage* 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 6.4.4.3.6.

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

Table 6-22 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 6-22 Commands and Responses**

Command	Allowed Response	Reference
<i>Discover Identity</i>	ACK, NAK, BUSY	6.4.4.3.1
<i>Discover SVIDs</i>	ACK, NAK, BUSY	6.4.4.3.1.10
<i>Discover Modes</i>	ACK, NAK, BUSY	6.4.4.3.3
<i>Enter Mode</i>	ACK, NAK	6.4.4.3.4
<i>Exit Mode</i>	ACK, <del>NAK</del>	6.4.4.3.5
<i>Attention</i>	None	6.4.4.3.6

Examples of Command usage can be found in Appendix G.

#### 6.4.4.3.1 Discover Identity

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

The Discover Identity Command shall be used to determine whether a given Cable Plug is PD Capable (see Sections 8.3.3.9.2 and 8.3.3.10.11). 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. Note that a Cable Plug will not be ready for PD Communication until tVCONNStable after VCONN has been applied (see [USBType-C 1.0]). During Cable Plug discovery, when there is an Explicit Contract, Discover Identity Commands are sent at a rate defined by the DiscoverIdentityTimer (see Section 6.5.14) up to a maximum of nDiscoverIdentityCount times (see Section 6.6.6).

A PD-Capable Cable Plug 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 shall return a Discover Identity Command ACK in response to a Discover Identity Command request sent to SOP.

The SVID in the **Discover Identity** Command shall be set to the **PD SID** (see Table 6-21) by both the Initiator and the Responder for this Command.

The **Discover Identity** Command sent back by the Responder contains an ID Header **VDO**, a Cert Stat VDO, a **Product VDO** and some **Product Type specific**-VDOs which depend on the Product Type (see Figure 6-6). This specification defines the following **Product Type specific**-VDOs:

- ~~Product VDO (see Section )~~
- Cable VDO (see Section 0).
- Alternate Mode Adapter VDO (see Section 6.4.4.3.1.10)

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 responder shall be Ignored.

Figure 6-6 Discover Identity Command response

Header No. of Data Objects = 4-7 <sup>1</sup>	VDM Header	ID Header VDO	Cert Stat VDO	Product VDO	0..3 <sup>2</sup> Product Type VDO(s)
--	------------	---------------	---------------	-------------	---------------------------------------

<sup>1</sup> Only Data objects defined in this specification can be sent as part of the Discover Identity Command.

<sup>2</sup> The following sections define the number and content of the VDOs for each Product Type.

##### 6.4.4.3.1.1 ID Header **VDO**

The ID Header **VDO** contains ~~the Vendor ID~~ information corresponding to the Power Delivery Product. The fields in the ID Header **VDO** shall be as defined in Table 6-23.

Table 6-23 ID Header **VDO**

Bit(s)	Description	Reference
B31	Data Capable as USB Host: <ul style="list-style-type: none"> <li>• Shall be set to one if the product is capable of enumerating USB Devices.</li> <li>• Shall be set to zero otherwise</li> </ul>	
B30	Data Capable as a USB Device: <ul style="list-style-type: none"> <li>• Shall be set to one if the product is capable of enumerating as a USB Device.</li> <li>• Shall be set to zero otherwise</li> </ul>	
B29..27	Product Type:	

Bit(s)	Description	Reference
	<ul style="list-style-type: none"> <li>• 000b – Undefined</li> <li>• 001b – Hub</li> <li>• 010b – Peripheral</li> <li>• 011b – Passive Cable</li> <li>• 100b – Active Cable</li> <li>• 101b – Alternate Mode Adapter (AMA)</li> <li>• <del>110b, 111b, 110b</del> – Reserved, shall not be used.</li> </ul>	
B26	Modal Operation Supported: <ul style="list-style-type: none"> <li>• Shall be set to one if the product supports Modal Operation.</li> <li>• Shall be set to zero otherwise</li> </ul>	
B25..16	Reserved. Shall be set to zero.	
B15..0	16-bit unsigned integer. USB Vendor ID	<a href="#">[USB 2.0]</a> / <a href="#">[USB 3.1]</a>

#### 6.4.4.3.1.2 Data Capable as a USB Host

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

#### 6.4.4.3.1.3 Data Capable as a USB Device

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

#### 6.4.4.3.1.4 Product Type

The Product Type field indicates the type of Product, whether a VDO will be returned and if so the type of VDO to be returned. Table 6-24 defines the [Product Type](#) VDOs which shall be returned.

Table 6-24 Product Types

Product Type	Description	Product Type VDO	Reference
Undefined	Shall be used where no other Product Type value is appropriate.	<a href="#">None</a>	
Hub	Shall be used when the Product is a USB Hub.	<del>Product VDO</del> <a href="#">None</a>	6.4.4.3.1.8
Peripheral	Shall be used when the Product is a USB Device other than a USB Hub.	<del>Product VDO</del> <a href="#">None</a>	
Active Cable	Shall be used when the Product is a cable that incorporates signal conditioning circuits.	Cable VDO	0
Passive Cable	Shall be used when the Product is a cable that does not incorporate signal conditioning circuits.	Cable VDO	0
Alternate Mode Adapter	Shall be used when the Product is a USB Device that supports one or more Alternate Modes.	<del>Product VDO</del> AMA VDO	6.4.4.3.1.10

#### 6.4.4.3.1.5 Modal Operation Supported

The Modal Operation Supported bit is used to indicate whether or not the Product supports Modes.

#### 6.4.4.3.1.6 Vendor ID

Manufacturers ~~who have a Vendor ID assigned by USB-IF~~ shall set the Vendor ID field to ~~this~~[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 [\[USB 2.0\]](#)~~[USB 2.0]~~ and [\[USB 3.1\]](#)~~[USB 3.0]~~).

~~Manufacturers who do not have a Vendor ID assigned by USB-IF shall set the Vendor ID field to **Vendor ID Unassigned**.~~

#### 6.4.4.3.1.7 Cert Stat VDO

The Cert Stat VDO ~~contains~~shall contain the Test ID (TID) ~~allocated~~assigned by USB-IF to the product during certification. The fields in the Cert Stat VDO shall be as defined in Table 6-25.

Table 6-25 Cert Stat VDO

Bit(s)	Description	Reference
B31..20	Reserved, shall be set to zero.	
B19..0	20-bit unsigned integer. <u>TID</u>	<u>Assigned by USB-IF</u> <del>compliance</del>

#### 6.4.4.3.1.8 Product VDO

~~The Product VDO shall be returned when the Product Type is Hub, Peripheral or Alternate Mode Adapter.~~ The Product VDO contains identity information relating to the product. The fields in the Product VDO shall be as defined in Table 6-26.

Table 6-26 Product VDO

Bit(s)	Description	Reference
B31..16	16-bit unsigned integer. USB Product ID	<u>[USB 2.0]</u> / <u>[USB 3.1]</u>
B15..0	16-bit unsigned integer. bcdDevice	<u>[USB 2.0]</u> / <u>[USB 3.1]</u>

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. For USB Devices or Hubs which support USB communications the Product ID and bcdDevice fields shall be identical to the Product ID and bcdDevice fields defined in the product's USB Device Descriptor (see [USB 2.0] and [USB 3.1]).

~~When the Vendor ID field in the ID Header is set to **Vendor ID Unassigned** the Product VDO shall not be sent.~~

#### 6.4.4.3.1.9 Cable VDO

The Cable VDO defined in this section shall be sent when the Product Type is given as Passive or Active Cable. Table 6-27 defines the Cable VDO which shall be sent.

Table 6-27 Cable VDO

Bit(s)	Field	Description
B31..28	Cable HW Version	0000b..1111b assigned by the VID owner
B27..24	Cable Firmware Version	0000b..1111b assigned by the VID owner
B23..20	Reserved	Shall be set to zero.
B19..18	Type-C plug to Type-A/B/C/Captive	00b = Type-A 01b = Type-B 10b = Type-C 11b = reserved,Captive
B17	Type-C plug to Plug/Receptacle	0 = Plug 1 = Receptacle (not valid when B19..18 set to Type-C or Captive)
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 = Both ends Passive, VCONN not required 01b = Both ends Passive, VCONN required 10b = One end Active, one end passive, VCONN required 11b = Both ends Active, VCONN required
B10	SSTX1 Directionality Support	0 = Fixed 1 = Configurable
B9	SSTX2 Directionality Support	0 = Fixed 1 = Configurable
B8	SSRX1 Directionality Support	0 = Fixed 1 = Configurable
B7	SSRX2 Directionality Support	0 = Fixed 1 = Configurable
B6.5	V <sub>BUS</sub> Current Handling Capability	00b = 4.5A V <sub>BUS</sub> not through cable 01b = 3A 10b = 5A 11b = Reserved, shall not be used.
B4	V <sub>BUS</sub> through cable	0 = No 1 = Yes
B3	SOP” controller present?	1 = SOP” controller present 0 = No SOP” controller present
B2..0	USB Superspeed Signaling Support	000b = USB 2.0 only 001b = [USB 3.1] Gen1 010b = [USB 3.1] Gen1 and Gen2 011b.. 111b = Reserved, shall not be used

6.4.4.3.1.10 Alternate Mode Adapter VDO

The Alternate Mode Adapter (AMA) VDO defined in this section shall be sent when the Product Type is given as Alternate Mode Adapter. Table 6-28 defines the AMA VDO which shall be sent.

Table 6-28 AMA VDO

Bit(s)	Field	Description
B31..28	<del>Cable</del> HW Version	0000b..1111b assigned by the VID owner
B27..24	<del>Cable</del> Firmware Version	0000b..1111b assigned by the VID owner
B23..12	Reserved.	Shall be set to zero.
B11	SSTX1 Directionality Support	0 = Fixed 1 = Configurable
B10	SSTX2 Directionality Support	0 = Fixed 1 = Configurable
B9	SSRX1 Directionality Support	0 = Fixed 1 = Configurable
B8	SSRX2 Directionality Support	0 = Fixed 1 = Configurable
B7..5	V <sub>CONN</sub> power	V <sub>CONN</sub> power needed by adapter for full functionality 000b = 1W 001b = 1.5W 010b = 2W 011b = 3W 100b = 4W 101b = 5W 110b = 6W 111b = Reserved, <del>shall not be used</del>
B4	V <sub>CONN</sub> required	0 = No 1 = Yes
B3	V <sub>BUS</sub> required	0 = No 1 = Yes
B2..0	USB Superspeed Signaling Support	000b = USB 2.0 only 001b = [USB3.1] Gen1 and USB 2.0 010b = [USB3.1] Gen1, Gen2 and USB 2.0 011b = USB 2.0 billboard only 100b.. 111b = Reserved, shall not be used

#### 6.4.4.3.2 Discover SVIDs

The *Discover SVIDs* Command is used by an Initiator (DFP) to determine the SVIDs for which a Responder (UFP or Cable Plug) 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 6-21) by both the Initiator and the Responder for this Command. The SVIDs are returned 2 per VDO (see Table 6-29). ~~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.

If the Responder supports ~~12 or more than 12~~ SVIDs then ~~this~~ the *Discover SVIDs* Command ~~may~~ 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.~~



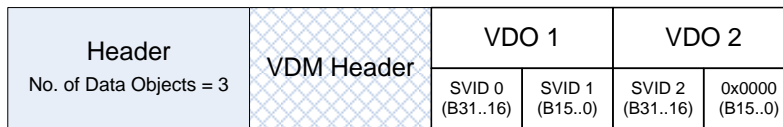
Note: that since a Cable Plug does not retry Messages if the GoodCRC Message from the DFP becomes corrupted the Cable Plug will consider the Discover SVIDs Command ACK unsend and will send the same list of SVIDs again.

Figure 6-7 shows an example response to the Discover SVIDs Command request with two VDOs containing three SVIDs. Figure 6-8 shows an example response with two VDOs containing four SVIDs followed by an empty VDO to terminate the response. Figure 6-9 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 6-29 Discover SVIDs Responder VDO**

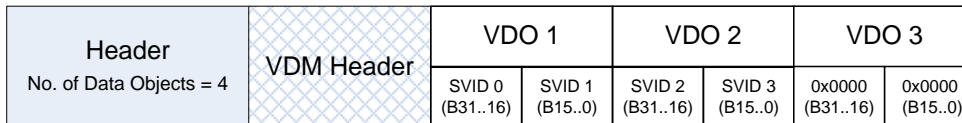
Bit(s)	Field	Description
B31..16	SVID n	16 bit unsigned integer, assigned by the USB-IF or 0x0000 if this is the last VDO and the Responder supports an even number of SVIDs.
B15..0	SVID n+1	16 bit unsigned integer, assigned by the USB-IF or 0x0000 if this is the last VDO and the Responder supports an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

**Figure 6-7 Example Discover SVIDs response with 3 SVIDs**

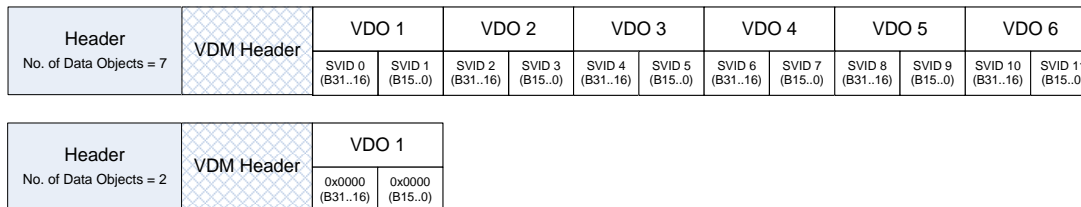


**Figure – Example Discover SVIDs response with 4 SVIDs**

**Figure 6-8 Example Discover SVIDs response with 4 SVIDs**



**Figure 6-9 Example Discover SVIDs response with 12 SVIDs followed by an empty response**

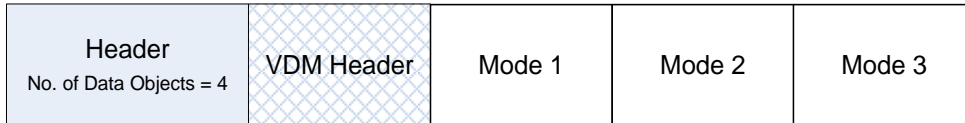


#### 6.4.4.3.3 Discover Modes

The Discover Modes Command is used by an Initiator (DFP) to determine the Modes a Responder (UFP or Cable Plug) supports for a given ID. The Responder to the Discover Modes Command request shall return a Message Header with the **Number of Data Objects** field set to a value of 1 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. Figure 6-10 shows an example of a *Discover Modes* Command response from a Responder which supports three Modes for a given SVID.

Figure 6-10 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 is allowed to 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 6-10). 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. 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, BUSY) 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 Modes are active at the same time Modal Operation shall start on entry to the first Mode.

On receiving an *Enter Mode* Command request 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. Receipt of the *GoodCRC* Message corresponding to the ACK confirms to the Responder that the Initiator has entered the is in an Active Mode and is ready to operate.

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 6-11 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 6-12 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 6-11 Successful Enter Mode sequence

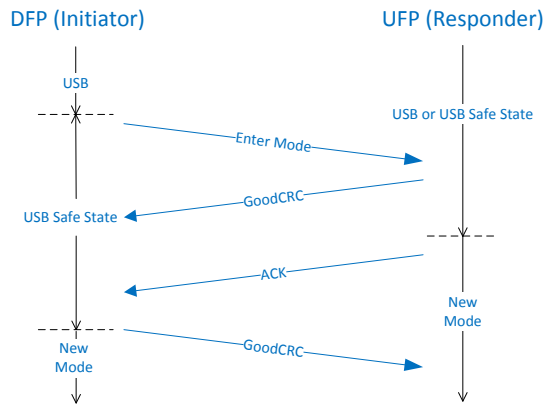
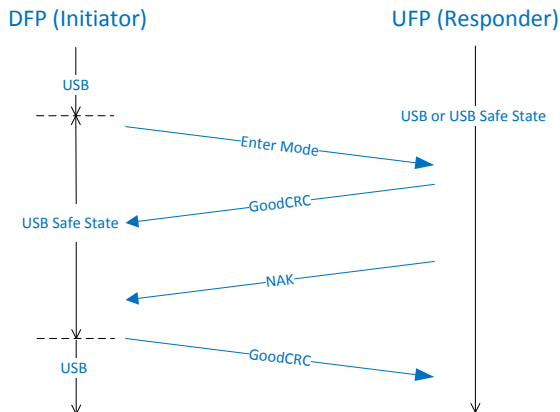


Figure 6-12 Unsuccessful Enter Mode sequence due to NAK



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).

The following events shall also cause the Port Partners and Cable Plug(s) to exit ~~of all presently~~ Active Modes:

- A PD Hard Reset
- The Port Partners or Cable Plug(s) are disconnected
- ~~A Data Role Swap~~
- A Cable Reset (only exits **Modes active in the Cable Plug(s) Plug's Active Modes**)

The Initiator shall return to USB Operation within  $t_{VDMExitMode}$  of a disconnect or of **Hard Reset** Signaling being detected. ~~During the Data Role Swap Process an Initiator shall return to USB Operation prior to changing its CC resistor.~~

The Responder shall return to either USB operation or USB Safe State within *tVDMExitMode* of a disconnect or of *Hard Reset* Signaling being detected. ~~During the Data Role Swap Process a Responder shall go to either USB Operation or USB Safe Mode prior to transitioning its CC resistor.~~

~~A DR\_Swap Message shall not be sent during Modal Operation between the Port Partners (see Section 6.3.9).~~

#### 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 of operation* and return to normal USB operation. Only the DFP is allowed to initiate the Exit Mode Process.

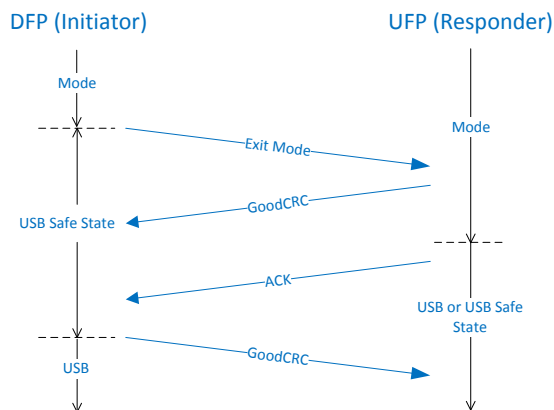
The value in the Object Position field shall indicate to which Mode in the *Discover Modes* Command the VDO refers (see Figure 6-10) and shall have been used previously in an *Enter Mode* Command request for ~~an Active Mode which is currently active~~. The value 1 always indicates the first Mode as 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 shall be set to 1 since an Exit Mode Command shall not contain any VDOs.~~

The Responder shall exit ~~theits Active~~ *Active Mode* before sending the response Message. The Initiator shall exit ~~theits Active~~ *Active Mode* before sending *GoodCRC* Message in response to the ACK. Receipt of the *GoodCRC* Message confirms to the Responder that the Initiator has exited the Mode. The Responder ~~is shall~~ not allowed to return ~~either a NAK or 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 ~~theits Active~~ *Active Mode*.

Figure 6-13 shows the sequence of events during the transition between exiting ~~an Active~~ *Active Mode* and USB operation. It illustrates when the Responder's Mode changes and when the Initiator's Mode changes.

Figure 6-13 Exit Mode sequence



#### 6.4.4.3.6 Attention

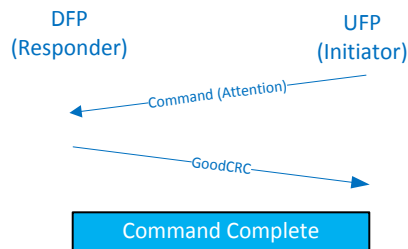
The *Attention* Command may be used by the Initiator (UFP) to notify the Responder (DFP) that it requires service.

The value in the Object Position field shall indicate to which Mode in the *Discover Modes* Command the VDO refers (see Figure 6-10) and shall have been used previously in an *Enter Mode* Command request for ~~an Active Mode which is currently active~~. 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. 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. No more than *nAttentionCount* Commands shall be sent during any given *tAttentionAverage (max)* period. The spacing between successive *Attention* Commands shall be at least *tAttentionSpacing* except that a single burst of no more than 2 *Attention* Commands with a spacing of at least *tAttentionBurstSpacing* may be sent during any given *tAttentionAverage (max)* period.

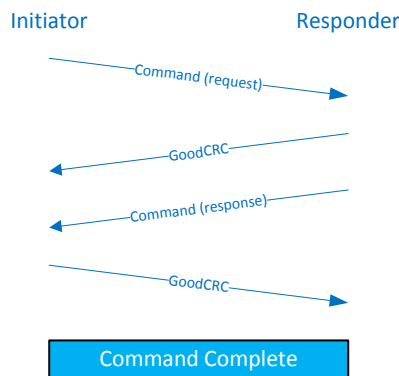
Figure 6-14 Attention Command request/response sequence



#### 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 6-15 shows the request/response sequence including the *GoodCRC* Messages.

Figure 6-15 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 has been supplied which is invalid
- BUSY if it recognizes the SVID and the Command but cannot process the Command request at this time

#### 6.4.4.4.1 Discovery Process

The Initiator (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.

The result of the Discovery Process is that both the Initiator (DFP) and Responder (UFP or Cable Plug) identify the Modes they mutually support. It is then up to the Initiator to select the Mode it wants to use and use the *Enter Mode* Command using the SVID and Object Position appropriate for that Mode.

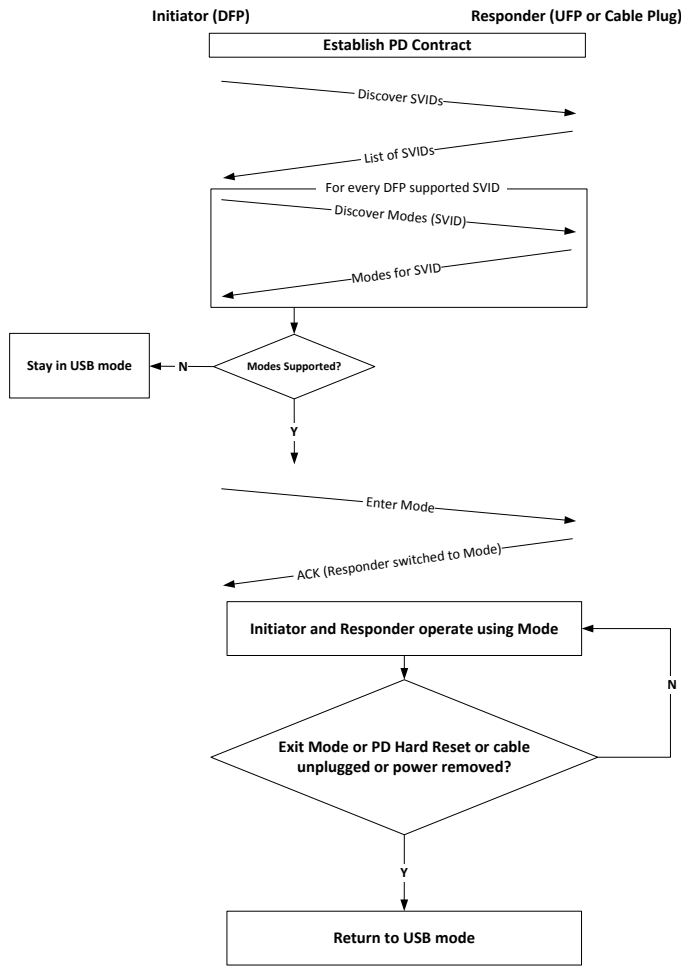
#### 6.4.4.4.2 Enter Vendor Mode / Exit Vendor Mode Processes

The Initiator (DFP), upon finding a Mode it supports, 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 operation is terminated/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. In an unmanaged termination, triggered by a Power Delivery Hard Reset (i.e. *Hard Reset* Signaling sent by either Port Partner) or ~~a Data Role Swap or~~ by cable detach (device unplugged), the *Active Mode* shall still be exited but there may 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 *[USBType-C 1.0]* following an exit from a Mode.

The overall Message flow is illustrated in Figure 6-16.

Figure 6-16 Enter/Exit Mode Process



#### 6.4.4.5 VDM Message Timing and Normal PD Messages

Any Command Process or other VDM sequence may be interrupted by any other USB PD Message. The Vendor or Standards defined state operation shall comprehend this and continue to operate as expected when processing any other USB PD Messages.

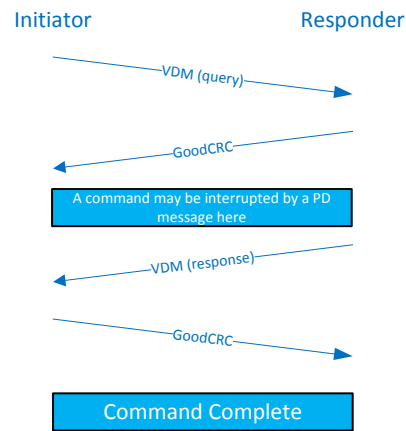
The timing and interspersing of VDMs between regular PD Messages shall be done without perturbing the PD Message sequences. This requirement shall apply to both Unstructured VDMs and Structured VDMs. Structured and Unstructured VDM sequences shall not be atomic although they may require a strict albeit interrupted ordering.

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.

VDM sequences shall be interruptible after the return of a *GoodCRC* Message has been completed. In the case where there is an error in transmission of the *Vendor\_Defined* Message, as for any other PD Message, the *Vendor\_Defined* Message will not be retried, but instead the incoming Message will be processed by the Policy Engine. This means that the *Vendor\_Defined* Message will need to be re-sent after the USB PD Message sequence has completed.

The overall Message flow is illustrated in Figure 6-17.

Figure 6-17 Vendor Defined Message interrupted by a Power Delivery Message





## 6.5 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.

### 6.5.1 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, ~~depending on the message, to either retry *nRetryCount* times. Note that Cable Plugs do not retry Messages (see Section 6.6.2, or to give up and assume communications have been lost).~~ Sending of the Preamble corresponding to the retried Message shall start within

<i>tRetry</i>		75	$\mu$ s	6.5.1
---------------	--	----	---------	-------

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.

#### ~~6.5.1.1.1.1.1 BISTReceiveErrorTimer~~

~~The *BISTReceiveErrorTimer* shall be used by the sender's Protocol Layer during BIST operation to detect when a Test Frame has been lost and to trigger the transmission of the next Test Frame. Failure to receive an acknowledgement of a Test Frame (message with a Data Object of *Returned BIST Counters*) whether caused by a bad CRC on the receiving end or by a garbled Message within *tBISTReceive* is detected when the *BISTReceiveErrorTimer* expires.~~

~~The sender's Protocol Layer response when a *BISTReceiveErrorTimer* expires shall be to continue operation.~~

~~The *BISTReceiveErrorTimer* shall be started when the *nBISTPayload* bit past the last bit of the *SOP\** has been transmitted by the Physical Layer. The *BISTReceiveErrorTimer* shall be stopped when the last bit of the *EOP* corresponding to the *BIST* Message with a Data Object of ~~has been received by the Physical Layer.~~~~

~~The Protocol Layer receiving a Test Frame shall respond with a message with a Data Object of *Returned BIST Counters* within *tTransmit* in order to ensure that the sender's *BISTReceiveErrorTimer* does not expire. The *tTransmit* shall be measured from when the last bit of the Test Frame has been received by the Physical Layer until the first bit of the Preamble of the *BIST* Message has been transmitted by the Physical Layer.~~

### 6.5.2 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.

The Policy Engine's response when the *SenderResponseTimer* expires shall be dependent on the Message sent (see Section 8.3).

The *SenderResponseTimer* shall be started from the time the last bit of the *GoodCRC* Message *EOP* (i.e. the *GoodCRC* Message 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 expected response Message *EOP* has been received 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 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.

### 6.5.3 Activity Timers

The Protocol Layer uses activity timers to ensure that there is adequate activity to allow the Source to know that a Sink is still present. The activity timer's value and use are specific to the role the device is playing – either Source or Sink. The values are selected to allow one missed response to a *Ping* Message without the Source returning its output to the default state. Activity Timers are not used in conjunction with the *[USBType-C.1.0]* connector (see Section 6.3.5.2)

#### 6.5.3.1 SourceActivityTimer

A PD Source that is not using a Type-C connector is required to maintain a minimal level of bus traffic in order to detect Sink detach. It does so by periodically sending a *Ping* Message during a connection to a PD Sink, whenever there is no other Message traffic.

In order to maintain bus activity the Source shall start its *SourceActivityTimer* as described in Section 8.3.3.2. Whenever the timer expires, after *tSourceActivity*, the Source shall send a *Ping* Message. It shall then initialize and restart the *SourceActivityTimer* ready for the next *Ping* Message. This ensures that Message activity is maintained in cases where the time between normal Messages would exceed the Sink's activity timeout. For example power supply transitions may take longer than the activity timeout meaning that a *Ping* is sent prior to the *PS\_RDY* Message.

The *SourceActivityTimer* shall be reset and restarted on entry to the *PE\_SRC\_Ready* state (see Section 8.3.3.2). This occurs when:

- The last bit of the *GoodCRC* Message *EOP* has been received by the Physical Layer (i.e. a Message has been successfully sent prior to entering the *PE\_SRC\_Ready* state).
- The last bit of any Message *EOP* has been received by the Physical Layer.
- The *SenderResponseTimer* times out.

A failure to see a *GoodCRC* Message in response to a *Ping* Message within *tReceive* (after *nRetryCount* retries) is indicative of a communications failure. This shall cause the Source to send a *Soft\_Reset* Message, transmission of which shall be completed within *tSoftReset* of the *CRCReceiveTimer* expiring (see Section 6.7.1).

See Section 8.3.3.6 for more details of when *Ping* Messages are transmitted.

#### 6.5.3.2 SinkActivityTimer

The Sink shall support the *SinkActivityTimer*. Sinks shall observe an absence of a *Ping*, or other Messages within *tSinkActivity*, as an indication of communications failure and as such shall issue *Hard\_Reset* Signaling in order to return to USB Default Operation. Initialization and restarting of the *SinkActivityTimer* is described in Section 8.3.3.7. Any additional action taken is Device implementation specific. Sinks shall also use the absence of  $V_{BUS}$  to return to USB Default Operation unless this is part of an ongoing Power Role Swap sequence.

The *SinkActivityTimer* shall be re-initialized and re-started when the last bit of a *Ping*, or any other Message *EOP*, has been received by the Physical Layer.

Note: to avoid triggering unnecessary Hard Resets the *SinkActivityTimer* shall not be run when the Sink is:

- Using a *[USBType-C.1.0]* connector or

- Using a Type-A operating in its default Source role or using a Type-B connector operating in its default Sink role, at *vSafe5V*, in the *PE\_SNK\_Ready* state. This is because *Ping* Messages are optional in these cases, (see Section 6.3.5).

See Section 8.3.3.3 for more details of when the *SinkActivityTimer* is run.

## 6.5.4 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.

### 6.5.4.1 SourceCapabilityTimer

Prior to a successful negotiation a Source shall use the *SourceCapabilityTimer* to periodically send out a *Source\_Capabilities* Message every *tTypeCSendSourceCap* for the Type-C connector and *tSendSourceCap* for all other connectors while:

- An A-plug 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 more details of when *Source\_Capabilities* Messages are transmitted.

### 6.5.4.2 SinkWaitCapTimer

The Sink shall support the *SinkWaitCapTimer*. When a Sink observes an absence of *Source\_Capabilities* Messages, after  $V_{BUS}$  is present, for a duration of *tTypeCSinkWaitCap* for the Type-C connector and *tSinkWaitCap* for all other connectors the Sink shall issue *Hard Reset* Signaling in order to restart the sending of *Source\_Capabilities* Messages by the Source (see Section 6.6.4).

See Section 8.3.3.3 for more details of when the *SinkWaitCapTimer* are run.

### 6.5.4.3 tFirstSourceCap

After Port Partners are attached or after a Hard Reset or after a Power Role Swap a Source shall send its first *Source\_Capabilities* Message within *tFirstSourceCap* of  $V_{BUS}$  reaching *vSafe5V*. This ensures that the Sink receives a *Source\_Capabilities* Message before the Sink's *SinkWaitCapTimer* expires.

## 6.5.5 SinkRequestTimer

The *SinkRequestTimer* is used to ensure that the time between Sink *Request* Messages, after a *Wait* Message has been received from the Source, is a minimum of *tSinkRequest* (see Section 6.3.12).

The *SinkRequestTimer* shall be started when 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 a *Wait* Message, before issuing 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*.

## 6.5.6 Power Supply Timers

### 6.5.6.1 PStTransitionTimer

The *PStTransitionTimer* 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 *tpStTransition* 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).

The *PSTransitionTimer* shall be started when the last bit of an *Accept* or *GotoMin* Message *EOP* has been received by the Physical Layer. The *PSTransitionTimer* shall be stopped when the last bit of the *PS\_RDY* Message *EOP* has been received by the Physical Layer.

#### 6.5.6.2 PSSourceOffTimer

The *PSSourceOffTimer* is used by the Policy Engine in Dual-Role 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 a Hard Reset and return to USB Default Operation.

If a *PR\_Swap* Message request has been sent by the Dual-Role Device currently acting as a Source the Sink can respond with an *Accept* Message. When the last bit of the *EOP* of the *GoodCRC* Message corresponding to this *Accept* Message is received by the Sink, then the *PSSourceOffTimer* shall be started.

If a *PR\_Swap* Message request has been sent by the Dual-Role Device currently acting as a Sink the Source can respond with an *Accept* Message. When the last bit of the *EOP* of this *Accept* Message is received by the Sink then the *PSSourceOffTimer* shall be started.

The *PSSourceOffTimer* shall be stopped when:

- The last bit of the *EOP* of the *PS\_RDY* Message is received.

The *PSSourceOffTimer* relates to the time taken for the remote Dual-Role Device to stop supplying power (see also Sections 7.3.9 and 7.3.10). The timer shall time out if a *PS\_RDY* Message has not been received from the remote Dual-Role Device within *tPSSourceOff* indicating this has occurred.

#### 6.5.6.3 PSSourceOnTimer

The *PSSourceOnTimer* is used by the Policy Engine in Dual-Role 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 a Hard Reset and return to USB Default Operation.

The *PSSourceOnTimer* shall be started when:

- The last bit of the *EOP* of the *GoodCRC* Message corresponding to the transmitted *PS\_RDY* Message is received by the Physical Layer

The *PSSourceOnTimer* shall be stopped when:

- The last bit of the *EOP* of the *PS\_RDY* Message is received by the Physical Layer

The *PSSourceOnTimer* relates to the time taken for the remote Dual-Role Device to start sourcing power (see also Sections 7.3.9 and 7.3.10) and will time out if a *PS\_RDY* Message indicating this has not been received within *tPSSourceOn*.

#### 6.5.7 NoResponseTimer

The *NoResponseTimer* is used by the Policy Engine in a Source or Sink 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 Sink fails to receive a *Source\_Capabilities* Message received 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 Sink shall issue additional Hard Resets up to *nHardResetCount* times (see Section 6.7.2).

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 6.7.2).

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.

## 6.5.8 BIST Timers

### 6.5.8.1 *tBISTMode*

*tBISTMode* is used to define the maximum time that a UUT has to enter a BIST mode when requested by a Tester.

A UUT shall enter the appropriate BIST mode within *tBISTMode* of the last bit of the *EOP* of the *BIST* Message used to initiate the test is received by the Physical Layer. When in BIST Receiver Mode the UUT is ready to receive a Test Frame from the Tester and to respond appropriately with the *BISTErrorCounter* (see Sections 6.5.8.5 and 0). When in BIST Transmit mode the UUT starts transmitting Test Frames (see Section 6.4.3.2). For modes transmitting a continuous carrier signal (i.e. carrier modes and eye pattern) transmission shall start as soon as the UUT enters BIST mode.

### 6.5.8.2 *tBISTResponse*

*tBISTResponse* defines the maximum time which a UUT has to respond with a Test Frame when operating in BIST Transmit Mode (see Section 6.4.3.2).

### 6.5.8.3 *BISTStartTimer*

The *BISTStartTimer* is used by the Tester to ensure that there is a delay of more than *tBISTMode* before sending the first Test Frame after requesting *BIST Receiver Mode*. The *BISTStartTimer* is initialized and run once the *BIST* Message containing the *BIST* Data Object has been sent i.e. a *GoodCRC* Message has been received. The first Test Frame is not sent until after the *BISTStartTimer* has expired.

### 6.5.8.4 *BISTContModeTimer*

The *BISTContModeTimer* is used by a UUT to ensure that a Continuous BIST 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 the last bit of the bit of the *EOP* of *GoodCRC* Message sent in response to the *BIST* Message used to enable the Continuous BIST Mode.

### 6.5.8.5 *BISTReceiveErrorTimer*

The *BISTReceiveErrorTimer* shall be used by the sender's Protocol Layer during BIST operation to detect when a Test Frame has been lost and to trigger the transmission of the next Test Frame. Failure to receive an acknowledgement of a Test Frame (*BIST* Message with a *BIST* Data Object of *Returned BIST Counters*) whether caused by a bad CRC on the receiving end or by a garbled Message within *tBISTReceive* is detected when the *BISTReceiveErrorTimer* expires.

The sender's Protocol Layer response when a *BISTReceiveErrorTimer* expires shall be to continue operation.

The *BISTReceiveErrorTimer* shall be started when the *nBISTPayload* bit past the last bit of the *SOP\** has been transmitted by the Physical Layer. The *BISTReceiveErrorTimer* shall be stopped when the last bit of the *EOP*

corresponding to the *BIST* Message with a *BIST* Data Object of *Returned BIST Counters* has been received by the Physical Layer.

The Protocol Layer receiving a Test Frame shall respond with a *BIST* Message with a *BIST* Data Object of *Returned BIST Counters* within *tTransmit* in order to ensure that the sender's *BISTReceiveErrorTimer* does not expire. The *tTransmit* shall be measured from when the last bit of the Test Frame has been received by the Physical Layer until the first bit of the Preamble of the *BIST* Message has been transmitted by the Physical Layer.

## 6.5.9 Power Role Swap Timers

### 6.5.9.1 SwapRecoveryTimer

The *SwapRecoveryTimer* is used by a Provider/Consumer acting in Sink role during a Hard Reset. The Provider/Consumer shall wait *tSwapRecover* after either sending or receiving *Hard Reset* Signaling before turning on its Power Source.

### 6.5.9.2 SwapSourceStartTimer

The *SwapSourceStartTimer* shall be used by the new Source, after a Power 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.

## 6.5.10 Hard Reset Timers

### 6.5.9.36.5.10.1 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.

### 6.5.10.2 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  $V_{BUS}$ . The Source shall wait *tPSHardReset* after sending *Hard Reset* Signaling before starting to transition the  $V_{BUS}$  voltage to  $v_{Safe0V}$ .

### 6.5.10.3 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.

## 6.5.106.5.11 Structured VDM Timers

### 6.5.10.16.5.11.1 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).

The *VDMResponseTimer* shall be started from the time the last bit of the *GoodCRC* Message *EOP* (i.e. the *GoodCRC* Message 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 expected VDM Command response *EOP* has been received 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 received by the Physical Layer until the first bit of the response Message Preamble has been transmitted by the Physical Layer.

#### [6.5.10.26.5.11.2](#) *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 8.3.3.9.5).

The *VDMModeEntryTimer* shall be started from the time the last bit of the *GoodCRC* Message *EOP* (i.e. 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 expected Structured VDM Command response (ACK, NAK or BUSY) *EOP* has been received 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 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.

#### [6.5.10.26.5.11.3](#) *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 8.3.3.9.6).

The *VDMModeExitTimer* shall be started from the time the last bit of the *GoodCRC* Message *EOP* (i.e. the *GoodCRC* Message 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 expected Structured VDM Command response ACK *EOP* has been received 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.

#### ~~6.5.10~~6.5.11.4 **tVDMBusy**

The Initiator shall wait at least *tVDMBusy*, after receiving a BUSY Command response, before repeating the Structured VDM request again.

### ~~6.5.11~~6.5.12 **VCONN Timers**

#### ~~6.5.11.1~~6.5.12.1 **VCONNOnTimer**

The *VCONNOnTimer* is used by the DFP or UFP during a VCONN Swap.

The *VCONNOnTimer* shall be started when:

- The last bit of the *EOP* of the *Accept* Message is received by the DFP.
- The last bit of the *EOP* of *GoodCRC* Message corresponding to the *Accept* Message is received by the UFP.

The *VCONNOnTimer* shall be stopped when:

- The last bit of the *EOP* of the *PS\_RDY* Message is received by the DFP or UFP.

The DFP or UFP, prior to sending the *PS\_RDY* Message, shall have turned VCONN On.

#### ~~6.5.11.2~~6.5.12.2 **tVCONNSourceOff**

The *tVCONNSourceOff* time applies during a Vconn Swap. The initial VCONN Source shall cease sourcing VCONN within *tVCONNSourceOff* of receipt of the last bit of the *EOP* of the *PS\_RDY* Message.

### **6.5.13 tCableMessage**

The sender of an SOP' or SOP'' packet (either a DFP or Cable Plug), that is not a *GoodCRC* Message, shall wait *tCableMessage* after the last bit of the *EOP* of the *GoodCRC* transmitted in response to the previous SOP' or SOP'' Packet before sending another SOP' or SOP'' Packet. This ensures that there is sufficient idle time between packets for a UFP to generate an asynchronous Message. Retransmission shall occur as described in Section 6.5.1 with *tCableMessage* applying to the last transmitted SOP' or SOP'' Packet.

### **6.5.14 DiscoverIdentityTimer**

The *DiscoverIdentityTimer* is used by a DFP 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 the DFP shall not send any further SOP'/SOP'' Messages.

### **6.5.15 Attention Timers**

Structured VDM *Attention Commands* are limited to a rate of no more than *nAttentionCount* *Attention Commands* during any given *tAttentionAverage* period. In addition Structured VDM *Attention Commands* have to be spaced at least *tAttentionSpacing* (min) apart except for a single burst of no more than 2 *Attention Commands* during any given *tAttentionAverage* period that have to be spaced at least *tAttentionBurstSpacing* apart.

See Section 6.4.4.3.6 for more details.

### ~~6.5.12~~6.5.16 **Time Values and Timers**

Table 6-30 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 6-31 lists the timers.



Table 6-30 Time Values

Parameter	Value (min)	Value (max)	Units	Section
<i>tAttentionAverage</i>		<u>10</u>	<u>s</u>	6.5.15
<i>tAttentionBurstSpacing</i>	<u>100</u>		<u>ms</u>	6.5.15
<i>tAttentionSpacing</i>	<u>250</u>		<u>ms</u>	6.5.15
<i>tBISTMode</i>		300	ms	6.5.8.1
<i>tBISTContMode</i>	<u>30</u>	<u>60</u>	<u>ms</u>	6.5.8.4
<i>tBISTReceive</i>	1.0	1.2	ms	6.5.8.5
<i>tBISTResponse</i>		15	ms	6.5.8.2
<i>tCableMessage</i>	<u>750</u>		<u>µs</u>	6.5.13
<i>tDiscoverIdentity</i>	<u>40</u>	<u>50</u>	<u>ms</u>	6.5.13
<i>tDRSwapHardReset</i>		<u>15</u>	<u>ms</u>	6.5.10.3
<i>tFirstSourceCap</i>		<u>250</u>	<u>ms</u>	
<i>tHardReset</i>		5	ms	6.3.13
<i>tHardResetComplete</i>	<u>4</u>	5	ms	6.5.10
<i>tNoResponse</i>	4.5	5.5	s	6.5.7
<i>tPSHardReset</i>	<u>25</u>	<u>35</u>	<u>ms</u>	6.5.10.2
<i>tPSSourceOff</i>	750	920	ms	6.5.6.2
<i>tPSSourceOn</i>	390	480	ms	6.5.6.3
<i>tPSTransition</i>	450	550	ms	6.5.6.1
<i>tReceive</i>	0.9	1.1	ms	6.5.1
<i>tReceiverResponse</i>		15	ms	1.1.1.1
<i>tRemoveV<sub>bus</sub></i>		<del>920</del>	<del>ms</del>	<del>6.4.1.2.2</del>
<i>tRetry</i>		75	µs	6.5.1
<i>tSenderResponse</i>	24	30	ms	1.1.1.1
<i>tSendSourceCap</i>	1	2	s	6.5.4.1
<i>tSinkActivity</i>	120	150	ms	6.5.3.2
<i>tSinkRequest</i>	100		ms	6.5.5
<i>tSinkWaitCap</i>	2.1	2.5	s	6.5.4.2
<i>tSoftReset</i>		<u>515</u>	ms	6.5.3.1, 6.7.1
<i>tSourceActivity</i>	40	50	ms	6.5.3.1
<i>tSwapSinkReady</i>		15	ms	6.5.9.2
<i>tSwapSourceStart</i>	20		ms	6.5.9.2
<i>tTransmit</i>		195	µs	6.5.1
<i>tTypeCSendSourceCap</i>	100	200	ms	6.5.4.1
<i>tTypeCSinkWaitCap</i>	<del>240</del> <u>310</u>	<del>250</del> <u>620</u>	ms	6.5.4.2
<i>tVCONNSourceOff</i>		25	ms	6.5.12
<i>tVCONNSourceOn</i>		100	ms	6.5.12
<i>tVDMBusy</i>	<del>400</del> <u>50</u>		ms	6.5.11.4

Parameter	Value (min)	Value (max)	Units	Section
<i>tVDMEnterMode</i>		25	ms	6.5.11.2
<i>tVDMExitMode</i>		25	ms	6.5.11.3
<i>tVDMReceiverResponse</i>		15	ms	6.5.11.1
<i>tVDMSenderResponse</i>	24	30	ms	6.5.11.1
<i>tVDMWaitModeEntry</i>	40	<del>100</del> 50	ms	6.5.11.2
<i>tVDMWaitModeExit</i>	40	<del>100</del> 50	ms	6.5.11.3

Table 6-31 Timers

Timer	Parameter	Used By	Section
<i>BISTContModeTimer</i>	<i>tBISTContMode</i>	Policy Engine	6.5.8.4
<i>BISTReceiveErrorTimer</i>	<i>tBISTReceive</i>	Protocol	6.5.8.5
<i>BISTStartTimer</i>	Defined by Tester	Policy Engine	6.5.8.3
<i>CRCReceiveTimer</i>	<i>tReceive</i>	Protocol	6.5.1
<i>DiscoverIdentityTimer</i>	<i>tDiscoverIdentity</i>	Policy Engine	
<i>HardResetCompleteTimer</i>	<i>tHardResetComplete</i>	Protocol	6.5.10
<i>NoResponseTimer</i>	<i>tNoResponse</i>	Policy Engine	6.5.7
<i>PSHardResetTimer</i>	<i>tPSHardReset</i>	Policy Engine	6.5.10.2
<i>PSSourceOffTimer</i>	<i>tPSSourceOff</i>	Policy Engine	6.5.6.2
<i>PSSourceOnTimer</i>	<i>tPSSourceOn</i>	Policy Engine	6.5.6.3
<i>PSTransitionTimer</i>	<i>tPSTransition</i>	Policy Engine	6.5.6.1
<i>SenderResponseTimer</i>	<i>tSenderResponse</i>	Policy Engine	1.1.1.1
<i>SinkActivityTimer</i>	<i>tSinkActivity</i>	Policy Engine	6.5.3.2
<i>SinkRequestTimer</i>	<i>tSinkRequest</i>	Policy Engine	6.5.5
<i>SinkWaitCapTimer</i>	<i>tSinkWaitCap</i> , <i>tTypeCSinkWaitCap</i>	Policy Engine	6.5.4.2
<i>SourceActivityTimer</i>	<i>tSourceActivity</i>	Policy Engine	6.5.3.1
<i>SourceCapabilityTimer</i>	<i>tSendSourceCap</i> , <i>tTypeCSendSourceCap</i>	Policy Engine	6.5.4.1
<i>SwapRecoveryTimer</i>	<i>tSwapRecover</i>	Policy Engine	6.5.9
<i>SwapSourceStartTimer</i>	<i>tSwapSourceStart</i>	Policy Engine	6.5.9.2
<i>VCONNOnTimer</i>	<i>tVCONNSourceOn</i>	Policy Engine	6.5.12.1
<i>VDMModeEntryTimer</i>	<i>tVDMWaitModeEntry</i>	Policy Engine	6.5.11.2
<i>VDMModeExitTimer</i>	<i>tVDMWaitModeExit</i>	Policy Engine	6.5.11.3
<i>VDMResponseTimer</i>	<i>tVDMSenderResponse</i>	Policy Engine	6.5.11.1

## 6.6 Counters

### 6.6.1 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 DFP or Source 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.

#### 6.6.1.1 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 ~~higher-layer protocol~~ *Policy Engine*, the transmitter shall delete the Message from its transmit buffer, re-initialize its retry mechanism and increment its *MessageIDCounter*.

#### 6.6.1.2 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.

### 6.6.2 Retry Counter

The *RetryCounter* is used by a DFP or UFP 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. Cable Plugs shall not retry Messages.

### 6.6.3 Hard Reset Counter

The *HardResetCounter* is used to retry the Hard Reset whenever there is no response from the remote device (see Section 6.5.7). Once the Hard Reset has been retried *nHardResetCount* times then it shall be assumed that the remote device is non-responsive.

### 6.6.4 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 ~~issue signaling up to times (see Section ) before it decides~~decide that the Sink is non-responsive ~~and it stops, 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.

### 6.6.5 BIST Error Counter

The Tester and UUT shall maintain a count of errors detected *BISTErrorCounter* (see Sections 0 and 6.4.3.2). The *BISTErrorCounter* shall contain the number of bits in error during a BIST Transmit or Receive test.

The *BISTErrorCounter* shall be a 16-bit counter that shall be set to zero upon the BIST test start. The *BISTErrorCounter* shall freeze at a value of FFFFh.

### 6.6.6 Discover Identity Counter

When sending *Discover Identity* Messages to a Cable Plug the DFP shall maintain a count of Messages sent (*DiscoverIdentityCounter*). No more than *nDiscoverIdentityCount* *Discover Identity* Messages shall be sent by the DFP receiving a *GoodCRC* Message response. A Data Role Swap shall reset the *DiscoverIdentityCounter* to zero.

### 6.6.7 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 [USBType-C 1.0] requirements for Mode entry should use an *nBusyCount* of 1.

### 6.6.8 nAttentionCount

Structured VDM *Attention* Commands are limited to a rate of no more than *nAttentionCount* *Attention* Commands during any given *tAttentionAverage* period (see Section 6.4.4.3.6).

### 6.6.6.6.9 Counter Values and Counters

Table 6-33 lists the counters used in this section and Table 6-32 shows the corresponding parameters.

Table 6-32 Counter parameters

Parameter	Value	Section
<i>nAttentionCount</i>	<u>10</u>	6.6.8
<i>nBusyCount</i>	<u>5</u>	6.6.7
<i>nCapsCount</i>	50	6.6.4
<i>nDiscoverIdentityCount</i>	<u>20</u>	6.6.6
<i>nHardResetCount</i>	2	6.6.3
<i>nMessageIDCount</i>	7	6.6.1
<i>nRetryCount</i>	<u>23</u>	6.6.2

Table 6-33 Counters

Counter	Max	Section
<i>BISTErrorCounter</i>	FFFFh	6.6.5
<i>CapsCounter</i>	<i>nCapsCount</i>	6.6.4
<i>DiscoverIdentityCounter</i>	<i>nDiscoverIdentityCount</i>	6.6.6
<i>HardResetCounter</i>	<i>nHardResetCount</i>	6.6.3
<i>MessageIDCounter</i>	<i>nMessageIDCount</i>	6.6.1
<i>RetryCounter</i>	<i>nRetryCount</i>	6.6.2
<i>VDMBusyCounter</i>	<i>nBusyCount</i>	6.6.7

## 6.7 Reset

Resets are a necessary response to protocol or other error conditions. USB Power Delivery defines two different types of reset; a Soft Reset, that resets protocol, and a Hard Reset which resets both the power supplies and protocol.

### 6.7.1 Soft Reset

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 shall be used whenever there is a Protocol Error ~~that it is not possible to handle in another way.~~ The Soft Reset may be triggered by either Port Partner in response to an error.

Protocol Errors that shall lead to a Soft Reset are unexpected Messages during one of the atomic Message sequences defined in Section 8.3.2. In this case the Policy Engine state machines need to be resynchronized. An unrecognized Message, received in the PE\_SNK\_Ready or PE\_SRC\_Ready states, shall not cause a Soft\_Reset Message to be generated but shall be Ignored.

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 to send a *Soft\_Reset* Message, transmission of which shall be completed within *tSoftReset* of the *CRCReceiveTimer* expiring.

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
- Power supply: Shall not change

A Soft Reset is performed using a sequence of protocol Messages (see Table 8-6). Message numbers shall be set to zero prior to sending the *Soft\_Reset/Accept* Message since the issue may 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 for any other SOP\* Packets; for example a *GoodCRC* Message is not received during the Soft Reset process (see Sections 6.7.2 and ).

### 6.7.2 Hard Reset

#### 6.7.2.1 Hard Reset Common Requirements

Hard Resets are signaled by an ordered set as defined in Section 5.6.4. Both the sender and recipient shall cause their power supplies to return to their default states (see Sections 7.3.12 to 7.3.14.4 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 Sections 6.5.7 and 6.6.3). 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.

#### 6.7.2.2 Type-A/B and Hard Reset

When there has been a Type-A/B Power Role Swap, a Hard Reset shall cause the Port Partners to return to their default Source/Sink roles. A Type-A Port shall return to Source operation. A Type-CB Port shall return to Sink operation.

#### 6.7.2.3 Type-C and Hard Reset

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  $V_{CONN}$  is supported (see [USBType-C 1.0]) the Hard Reset shall cause the Port with the  $R_p$  resistor asserted to supply  $V_{CONN}$  and the Port with the  $R_d$  resistor asserted to turn off  $V_{CONN}$ .

In effect the Hard Reset will revert the Ports to their default state based on their CC line resistors. Removing and reapplying  $V_{CONN}$  from the Cable Plugs also ensures that they re-establish their configuration as either SOP' or SOP' based on the location of  $V_{CONN}$  (see [USBType-C 1.0], to the Data Role or to the Source of  $V_{CONN}$ ).

If the Hard Reset is insufficient to clear the error condition then the Port should use error recovery mechanisms as defined in [USBType-C 1.0].

#### 6.7.2.4 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 8.3.3.10.8). The Cable Plugs shall perform the equivalent of a power cycle returning to their initial power up state. This allows the attached products to be in a state where they can re-establish USB PD communication.

#### 6.7.2.5 Modal Operation and Hard Reset

A Hard Reset shall cause all *Active* Modes to be exited by both Port Partners and any Cable Plugs (see Section 6.4.4.3.4).

### 6.7.3 Cable Reset

Cable Resets are signaled by an ordered set as defined in Section 5.6.5. 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 attached products to be in a state where they can re-establish USB PD communication.

The DFP has to be supplying  $V_{CONN}$  prior to a Cable Plugs Reset to ensure that the Cable Plugs correctly configure SOP' and SOP' after the Cable Reset is complete. If  $V_{CONN}$  has been turned off the DFP shall turn on  $V_{CONN}$  prior to generating *Cable Reset* Signaling. If there has been a  $V_{CONN}$  Swap and the UFP is currently supplying  $V_{CONN}$ , the DFP shall perform a  $V_{CONN}$  Swap such that it is supplying  $V_{CONN}$  prior to generating *Cable Reset* Signaling.

Only a DFP shall generate *Cable Reset* Signaling but a DFP shall reset when only generate *Cable Reset* Signaling is present within an Explicit Contract.

A Cable Reset shall cause all *Active* Modes ~~currently active~~ in the Cable Plugs to be exited (see Section 6.4.4.3.4).

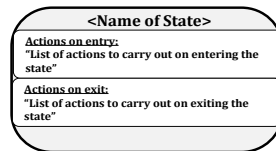
## 6.8 State behavior

### 6.8.1 Introduction to state diagrams used in Chapter 6

The state diagrams defined in Section 6.8 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 6-18 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 6-18 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 6-19 indicates how such references are made. The reference is indicated with a hatched box. The box contains the name of the referenced state.

Figure 6-19 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 particular 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.)

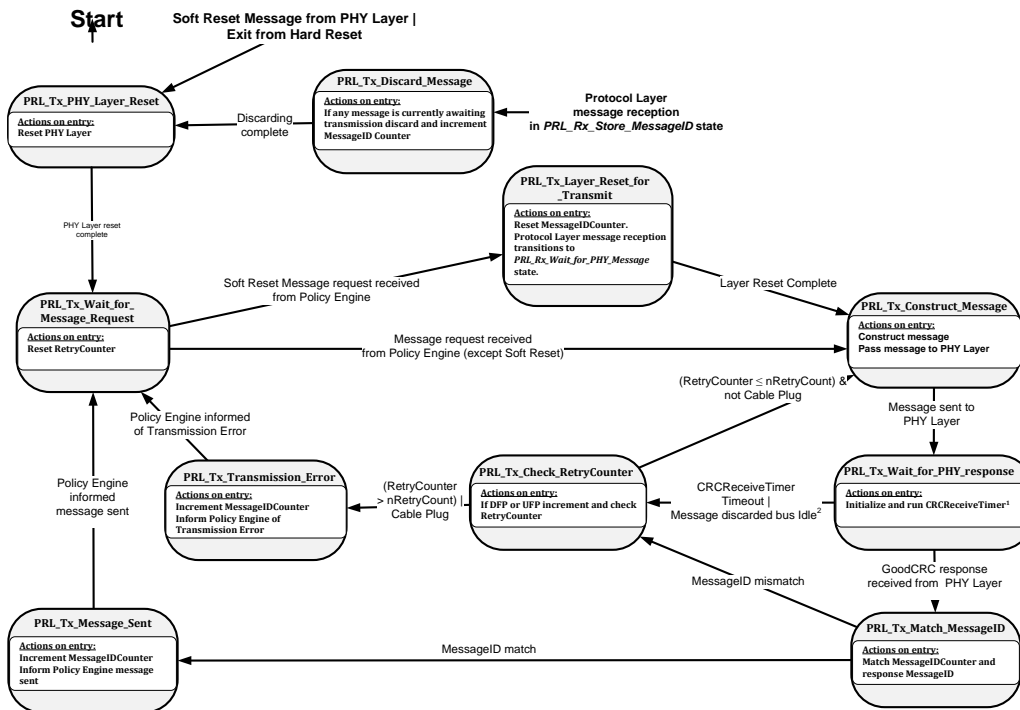
### 6.8.2 State Operation

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 BIST State Machine instances, with their own counter and timer 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).

### 6.8.2.1 Protocol Layer Message Transmission

Figure 6-20 shows the state behavior for the Protocol Layer when transmitting a Message.

Figure 6-20 Protocol Layer Message transmission



<sup>1</sup> The **CRCReceiveTimer** is only started after the PHY has sent the message. If the message is not sent due to a busy channel then the **CRCReceiveTimer** will not be started (see Section 6.5.1).

<sup>2</sup> This indication is sent by the PHY Layer when a message has been discarded due to  $V_{BUS}$  or CC being busy, and after  $V_{BUS}$  or CC becomes idle again (see Section 5.7). The **CRCReceiveTimer** is not running in this case since no message has been sent.

#### 6.8.2.1.1 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.

#### 6.8.2.1.2 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. ~~This shall be the startup state for the Protocol Layer message transmission and also the starting state when exiting from a Hard Reset.~~

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.

#### 6.8.2.1.3 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 6.8.2.2.1) 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.

#### 6.8.2.1.4 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.

#### 6.8.2.1.5 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 6.5.1).

The Protocol Layer shall transition to the *PRL\_Tx\_Match\_MessageID* state when:

- A *GoodCRC* Message response is receive 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).

#### 6.8.2.1.6 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.

#### 6.8.2.1.7 PRL\_Tx\_Message\_Sent state

On entry to the *PRL\_Tx\_Message\_Sent* state the Protocol Layer shall increment the *MessageIDCounter*, ~~reset the~~ 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.

#### 6.8.2.1.8 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.

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.

#### 6.8.2.1.9 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.

#### 6.8.2.1.10 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.

On entry to the *PRL\_Tx\_Discard\_Message* state, if there is a Message queued awaiting transmission, the Protocol Layer shall discard the Message 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.

### 6.8.2.2 Protocol Layer Message Reception

Figure 6-21 shows the state behavior for the Protocol Layer when receiving a Message.

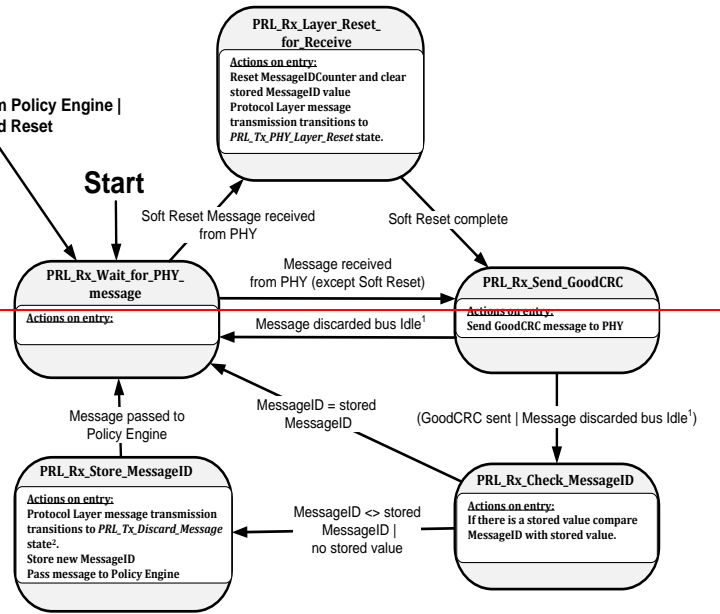
Field Code Changed

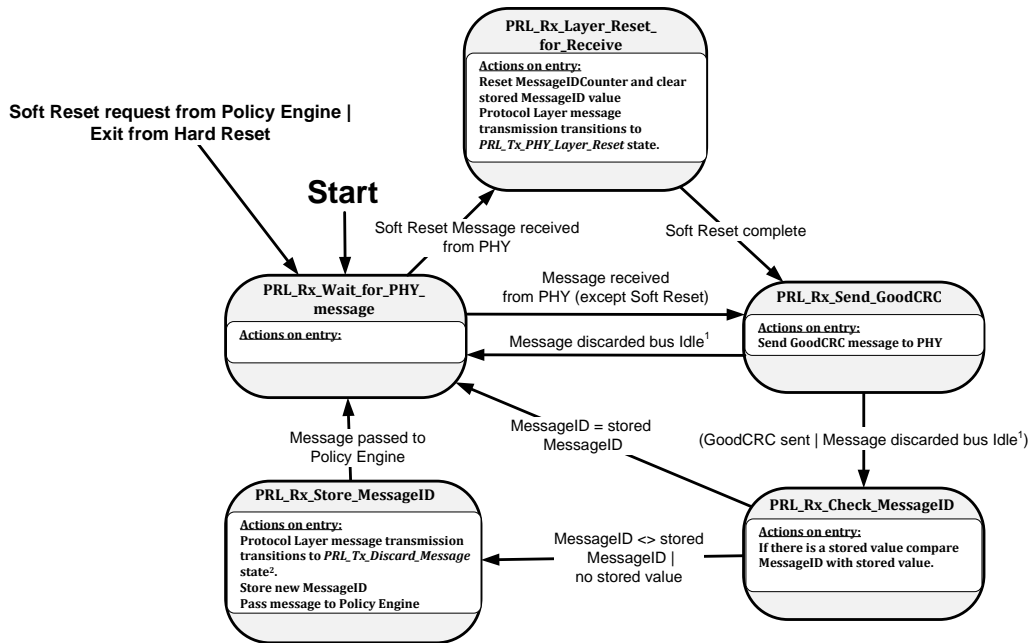
Field Code Changed

Figure 6-21 Protocol layer Message reception

Soft Reset request from Policy Engine |  
Exit from Hard Reset

Start





<sup>1</sup>This indication is sent by the PHY when a message has been discarded due to  $V_{BUS}$  or CC being busy, and after  $V_{BUS}$  or CC becomes idle again (see Section 5.7). Two alternate allowable transitions are shown.

<sup>2</sup>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.

#### 6.8.2.2.1 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.

#### 6.8.2.2.2 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 6.8.2.1.1).

The Protocol Layer shall transition to the *PRL\_Rx\_Send\_GoodCRC* State when:

- The Soft Reset actions in this state have been completed.

#### 6.8.2.2.3 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 ~~or~~

When the PHY Layer indicates that a Message has been discarded due to  $V_{BUS}$  or CC being busy but  $V_{BUS}$  or CC is now idle (see Section 5.7), the Protocol Layer shall either:

- Transition to the *PRL\_Rx\_Check\_MessageID* state or
- Transition to the *PRL\_Rx\_Wait\_for\_PHY\_Message* state

#### 6.8.2.2.4 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 ~~.~~

#### 6.8.2.2.5 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.

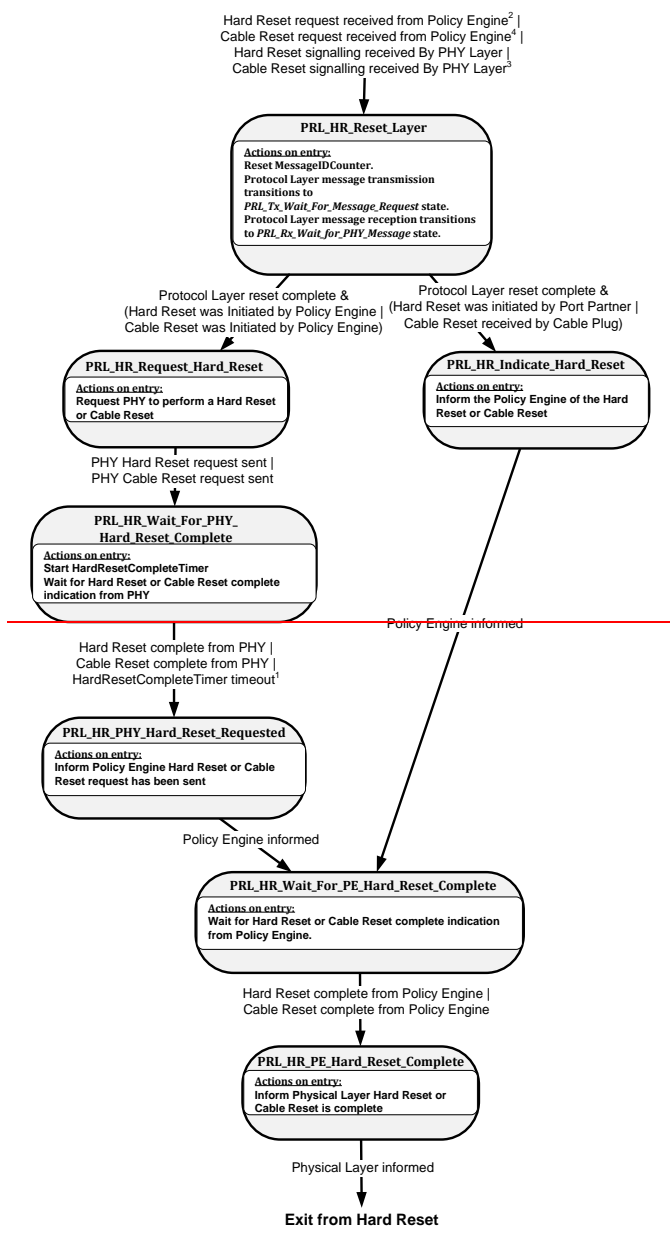
The Protocol Layer shall transition to the *PRL\_Rx\_Wait\_for\_PHY\_Message* state when:

- The Message has been passed up to the Policy Engine.

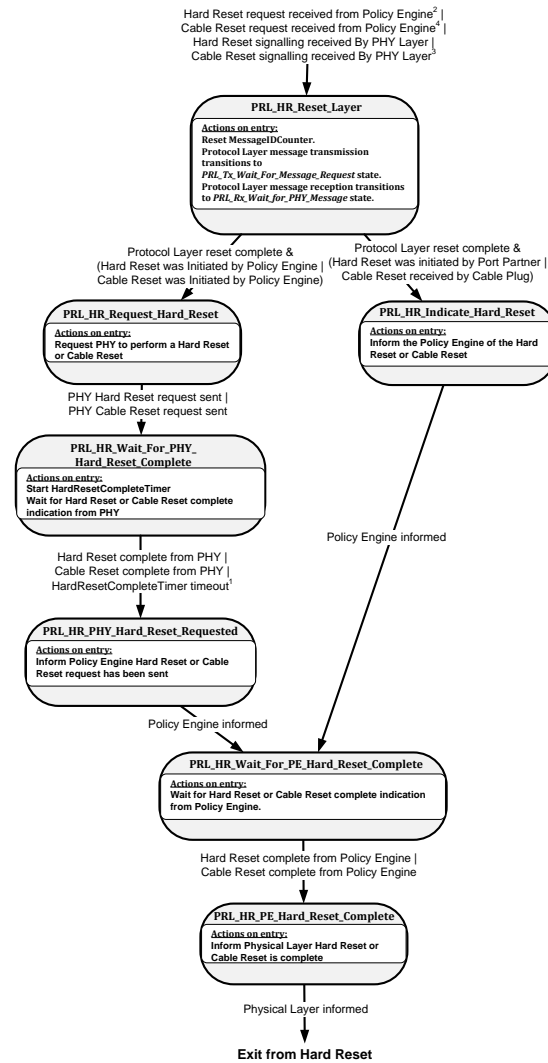
### 6.8.2.3 Hard Reset operation

Figure 6-22 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~~Sections 6.7.2 and 6.7.3).

Figure 6-22 Hard/Cable Reset







<sup>1</sup> If the HardResetTimer 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.

<sup>2</sup> 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.

<sup>3</sup> Cable Reset signalling is only recognised by a Cable Plug.

<sup>4</sup> Cable Reset signaling cannot be generated by Cable Plugs

#### 6.8.2.3.1 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 ~~this mode~~ **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).

#### 6.8.2.3.2 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.

#### 6.8.2.3.3 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.

#### 6.8.2.3.4 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\_PE\_Hard\_Reset\_Complete** 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.

#### 6.8.2.3.5 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.

#### 6.8.2.3.6 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.

#### 6.8.2.3.7 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:

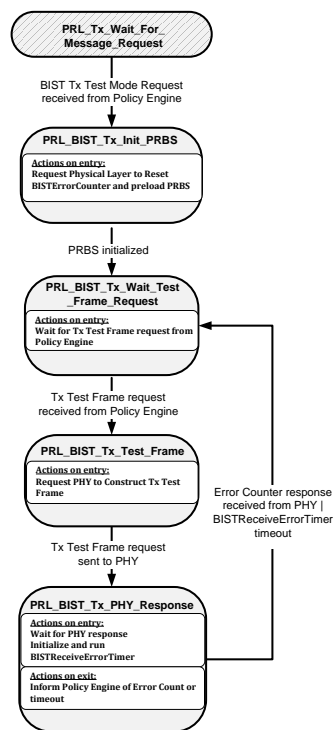
- The Physical Layer has been informed that the Hard Reset is complete so that it will re-enable the communications channel. If *Hard Reset* Signaling is still pending due to a non-idle channel this shall be cleared and not sent ~~or~~
- The Physical Layer has been informed that the Cable Reset is complete.

## 6.8.3 BIST Operation

### 6.8.3.1 BIST Transmitter Test

Figure 6-23 shows the state behavior for the Protocol Layer when in BIST Transmitter Test mode and transmitting BIST Transmitter Test Frames. The Protocol Layer changes from normal operation for Protocol Message Layer Transmission (see Section 6.8.2.1) to BIST Transmitter Test Mode when directed by the Policy Engine.

Figure 6-23 BIST Transmitter Test



#### 6.8.3.1.1 PRL\_BIST\_Tx\_Init\_PRBS state

The Protocol Layer shall enter the *PRL\_BIST\_Tx\_Init\_PRBS* state from the *PRL\_Tx\_Wait\_for\_Message\_Request* state (see Section 6.8.2.1.1) when:

- The Policy Engine requests the Protocol Layer to enter BIST Transmitter Test Mode.

On entry to the *PRL\_BIST\_Tx\_Init\_PRBS* state the Protocol Layer shall request the Physical Layer to reset the *BISTErrorCounter* and preload the PRBS (see Section 5.9.1).

The Protocol Layer shall transition to the *PRL\_BIST\_Tx\_Wait\_Test\_Frame\_Request* state when:

- The PRBS generator has been preloaded.

#### 6.8.3.1.2 PRL\_BIST\_Tx\_Wait\_Test\_Frame\_Request state

On entry to the **PRL\_BIST\_Tx\_Wait\_Test\_Frame\_Request** state the Protocol Layer shall wait for a Test Frame from the Policy Engine.

The Protocol Layer shall transition to the **PRL\_BIST\_Tx\_Test\_Frame** state when:

- When a request to transmit a Test Frame is received from the Policy Engine.

#### 6.8.3.1.3 PRL\_BIST\_Tx\_Test\_Frame state

On entry to the **PRL\_BIST\_Tx\_Test\_Frame** state the Protocol Layer shall request the Physical Layer to construct the Test Frame.

The Protocol Layer shall transition to the **PRL\_BIST\_Tx\_PHY\_Response** state when:

- The Test Frame request has been sent to the Physical Layer.

#### 6.8.3.1.4 PRL\_BIST\_Tx\_PHY\_Response state

In the **PRL\_BIST\_Tx\_PHY\_Response** state the Protocol Layer shall wait for the Physical Layer to provide a response to the Test Frame.

On entry to the **PRL\_BIST\_Tx\_PHY\_Response** state the Protocol Layer shall initialize and run the **BISTReceiveErrorTimer**.

On exit from the **PRL\_BIST\_Tx\_PHY\_Response** state the Protocol Layer shall inform the Policy Engine either of the received Error Counter value or of a **BISTReceiveErrorTimer** timeout.

The Protocol Layer shall transition to the **PRL\_BIST\_Tx\_Wait\_Test\_Frame\_Request** state (see Section 6.8.3.1.2) when:

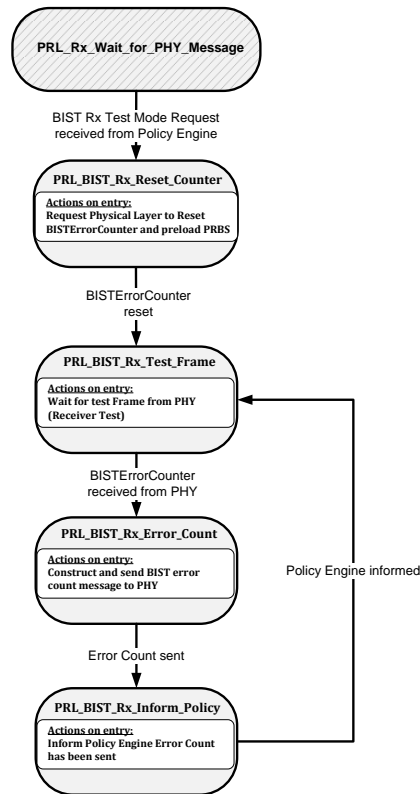
- An error counter response is received from the Physical Layer.
- Or the **BISTReceiveErrorTimer** times out.

### 6.8.3.2 BIST Receiver Test

Figure 6-24 shows the state behavior for the Protocol Layer when in BIST Receiver Test mode and receiving BIST Receiver Test Frames. The Protocol Layer changes from normal operation for Protocol Layer Message Reception (see Section 6.8.2.2) to BIST Receiver Test Mode when directed by the Policy Engine. Exit from this mode of operation is via Hard Reset (see Section 6.8.2.3).

Field Code Changed

Figure 6-24 BIST Receiver Test



6.8.3.2.1 PRL\_BIST\_Rx\_Reset\_Counter state

The Protocol Layer shall enter the *PRL\_BIST\_Rx\_Reset\_Counter* state from the *PRL\_Rx\_Wait\_for\_PHY\_Message* states when:

- A BIST Receiver Test Mode request is received from the Policy Engine.

On entry to the *PRL\_BIST\_Rx\_Reset\_Counter* state the Protocol Layer shall request the Physical Layer to reset the *BISTErrorCounter* and preload the PRBS (see Section 5.9.1).

The Protocol Layer shall transition to the *PRL\_BIST\_Rx\_Test\_Frame* state when:

- The *BISTErrorCounter* has been reset by the Physical Layer.

6.8.3.2.2 PRL\_BIST\_Rx\_Test\_Frame state

In the *PRL\_BIST\_Rx\_Test\_Frame* State the Protocol Layer shall wait for the Physical Layer to receive the next Test Frame.

The Protocol Layer shall transition to the *PRL\_BIST\_Rx\_Error\_Count* state when:

- The current value of the *BISTErrorCounter* is received from the PHY Layer.

#### 6.8.3.2.3 PRL\_BIST\_Rx\_Error\_Count state

On entry to the *PRL\_BIST\_Rx\_Error\_Count* state the Protocol Layer shall construct a *BIST* Message with a *BIST* Data Object of *Returned BIST Counters* using the *BISTErrorCounter* value returned by the Physical Layer. This *BIST* Message shall be passed to the Physical Layer for transmission.

The Protocol Layer shall transition to the *PRL\_BIST\_Rx\_Inform\_Policy* state when:

- The *BIST* Message has been sent.

#### 6.8.3.2.4 PRL\_BIST\_Rx\_BIST\_Inform\_Policy state

On entry to the *PRL\_BIST\_Rx\_Inform\_Policy* state the Protocol Layer shall inform the Policy Engine that the *BIST* Message containing the *BISTErrorCounter* has been sent.

The Protocol Layer shall transition to the *PRL\_BIST\_Rx\_Test\_Frame* state when:

- The Policy Engine has been informed that the *BIST* Message containing the *BISTErrorCounter* has been sent.

## 6.8.4 List of Protocol Layer States

Table 6-34 lists the states used by the various state machines.

Table 6-34 Protocol Layer States

State name	Section
<b>Protocol Layer Message Transmission</b>	
<i>PRL_Tx_PHY_Layer_Reset</i>	6.8.2.1.1
<i>PRL_Tx_Wait_for_Message_Request</i>	6.8.2.1.2
<i>PRL_Tx_Layer_Reset_for_Transmit</i>	6.8.2.1.3
<i>PRL_Tx_Construct_Message</i>	6.8.2.1.4
<i>PRL_Tx_Wait_for_PHY_Response</i>	6.8.2.1.5
<i>PRL_Tx_Match_MessageID</i>	6.8.2.1.6
<i>PRL_Tx_Message_Sent</i>	6.8.2.1.7
<i>PRL_Tx_Check_RetryCounter</i>	6.8.2.1.8
<i>PRL_Tx_Transmission_Error</i>	6.8.2.1.9
<i>PRL_Tx_Discard_Message</i>	6.8.2.1.10
<b>Protocol Layer Message Reception</b>	
<i>PRL_Rx_Wait_for_PHY_Message</i>	6.8.2.2.1
<i>PRL_Rx_Layer_Reset_for_Receive</i>	6.8.2.2.2
<i>PRL_Rx_Send_GoodCRC</i>	6.8.2.2.3
<i>PRL_Rx_Check_MessageID</i>	6.8.2.2.4
<i>PRL_Rx_Store_MessageID</i>	6.8.2.2.5
<b>Hard Reset Operation</b>	
<i>PRL_HR_Reset_Layer</i>	6.8.2.3.1
<i>PRL_HR_Indicate_Hard_Reset</i>	6.8.2.3.2
<i>PRL_HR_Request_Hard_Reset</i>	6.8.2.3.3
<i>PRL_HR_Wait_for_PHY_Hard_Reset_Complete</i>	6.8.2.3.4
<i>PRL_HR_PHY_Hard_Reset_Requested</i>	6.8.2.3.5
<i>PRL_HR_Wait_for_PE_Hard_Reset_Complete</i>	6.8.2.3.6
<i>PRL_HR_PE_Hard_Reset_Complete</i>	6.8.2.3.7
<b>BIST Transmitter Test</b>	
<i>PRL_BIST_Tx_Init_PRBS</i>	6.8.3.1.1
<i>PRL_BIST_Tx_Wait_Test_Frame_Request</i>	6.8.3.1.2
<i>PRL_BIST_Tx_Test_Frame</i>	6.8.3.1.3
<i>PRL_BIST_Tx_PHY_Response</i>	6.8.3.1.4
<b>BIST Receiver Test</b>	
<i>PRL_BIST_Rx_Reset_Counter</i>	6.8.3.2.1
<i>PRL_BIST_Rx_Test_Frame</i>	6.8.3.2.2
<i>PRL_BIST_Rx_Error_Count</i>	6.8.3.2.3
<i>PRL_BIST_Rx_Inform_Policy</i>	6.8.3.2.4





## 6.9 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 following abbreviations are used:

- M – Mandatory; shall be supported by this Port/Cable Plug
- CM – Conditionally Mandatory; shall be 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 not be transmitted by this Port/Cable Plug and shall be Ignored by this Port/Cable Plug when received.
- RJ – Reject; this Port/Cable Plug shall return a *Reject* Message when received
- NK – NAK; this Port/Cable Plug shall return Responder NAK to this Command when received

For the case of Conditional Mandatory a note has been added to indicate the condition. “CM/” 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/RJ/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 6.7.1).

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.

Table 6-35 details Control Messages (except for those specific to the Type-C connector) that shall/should/shall not be transmitted and received by a Source, Sink or Cable Plug. Requirements for Dual-Role Power Ports shall override any requirements for Source-only or Sink-Only Ports.

**Table 6-35 Applicability of Control Messages**

Message Type	Source	Sink	Dual-Role Power	Cable Plug
<b>Transmitted Message</b>				
<i>GoodCRC</i>	M	M		M
<i>GotoMin</i>	CM <sup>1</sup> /O	NS		NS
<i>Accept</i>	M	M		M
<i>Reject</i>	M	M		NS
<i>Ping</i>	CM <sup>2</sup> /O	NS		NS
<i>PS_RDY</i>	M	NS	M	NS
<i>Get_Source_Cap</i>	O	M	M	NS
<i>Get_Sink_Cap</i>	M	O	M	NS
<i>PR_Swap</i>	NS	NS	M	NS
<i>Wait</i>	CM <sup>3</sup> /O	NS	O	NS
<i>Soft_Reset</i>	M	M		NS
<b>Received Message</b>				
<i>GoodCRC</i>	M	M		M

Message Type	Source	Sink	Dual-Role Power	Cable Plug
<i>GotoMin</i>	NS	R <sup>d</sup>		NS
<i>Accept</i>	M	M		NS
<i>Reject</i>	M	M		NS
<i>Ping</i>	NS	NS		NS
<i>PS_RDY</i>	NS	M	M	NS
<i>Get_Source_Cap</i>	M	RI	M	NS
<i>Get_Sink_Cap</i>	RI	M	M	NS
<i>PR_Swap</i>	RI	RI	M	NS
<i>Wait</i>	NS	M	M	NS
<i>Soft_Reset</i>	M	M		M
<p><u>Note 1: Shall be supported by a Hub with multiple Downstream Ports. Should be supported by a Host with multiple Downstream Ports.</u></p> <p><u>Note 2: Shall be supported by BFSK systems.</u></p> <p><u>Note 3: Shall be supported when transmission of <i>GotoMin</i> Messages is supported.</u></p> <p><u>Note 4: Should be supported by Sinks which use PD power for charging.</u></p>				

Table 6-36 details Control Messages specific to the Type-C connector that shall/should/shall not be transmitted and received by a DFP, UFP or Cable Plug. Requirements for Dual-Role Data Ports shall override any requirements for DFP-only or UFP-Only Ports.

Table 6-36 Applicability of Type-C Specific Control Messages

Message Type Sent	DFP	UFP	Dual-Role Data	Cable Plug
<b>Transmitted Message</b>				
<i>DR_Swap</i>	NS	NS	M	NS
<i>Wait</i>			Q	
<i>VCONN_Swap</i>	R	R		NS
<b>Received Message</b>				
<i>DR_Swap</i>	RI	RI	M	NS
<i>Wait</i>			M	
<i>VCONN_Swap</i>	CM <sup>1</sup> /RI	CM <sup>1</sup> /RI		NS
<p><u>Note 1: Shall be supported by any Type-C Port that that utilizes the SSTX and SSRX pins and supports the <i>PR_Swap</i> Message.</u></p>				

Table 6-37 details Data Messages (except for VDM Commands) that shall/should/shall not be transmitted and received by a Source, Sink or Cable Plug. Requirements for Dual-Role Power Ports shall override any requirements for Source-only or Sink-Only Ports.

Table 6-37 Applicability of Data Messages

Message Type	Source	Sink	Dual-Role Power	Cable Plug
<b>Transmitted Message</b>				
<i>Source_Capabilities</i>	M	NS	M	NS
<i>Request</i>	NS	M		NS

<b>Message Type</b>	<b>Source</b>	<b>Sink</b>	<b>Dual-Role Power</b>	<b>Cable Plug</b>
<i>BIST</i>	<u>M<sup>2</sup></u>	<u>M<sup>2</sup></u>		<u>NS</u>
<i>Sink Capabilities</i>	<u>NS</u>	<u>M</u>	<u>M</u>	<u>NS</u>
<b>Received Message</b>				
<i>Source Capabilities</i>	<u>NS<sup>1</sup></u>	<u>M</u>	<u>M</u>	<u>NS</u>
<i>Request</i>	<u>M</u>	<u>NS</u>		<u>NS</u>
<i>BIST</i>	<u>M<sup>2</sup></u>	<u>M<sup>2</sup></u>		<u>M<sup>2</sup></u>
<i>Sink Capabilities</i>	<u>M</u>	<u>NS</u>	<u>M</u>	<u>NS</u>
<u>Note 1: See Section 1.1.1.1.1.</u>				
<u>Note 2: For details of which BIST Modes and Messages shall be supported see Sections 5.9.9 and 6.4.3.</u>				

Table 6-38 details VDM Commands that shall/should/shall not be transmitted and received by a DFP, UFP or Cable Plug. Requirements for Modal Operation shall override any requirements for DFP or UFP Ports or Cable Plugs without Modal Operation.

**Table 6-38 Applicability of VDM Commands**

<b>Command Type</b>	<b>DFP</b>	<b>UFP</b>	<b>Cable Plug</b>	<b>Modal Operation Supported</b>
<b>Transmitted Command</b>				
<i>Discover Identity</i>	<u>R</u>	<u>R<sup>2</sup></u>	<u>NS</u>	<u>CM<sup>1</sup></u>
<i>Discover SVIDs</i>	<u>NS</u>	<u>NS</u>	<u>NS</u>	<u>CM<sup>1</sup></u>
<i>Discover Modes</i>	<u>NS</u>	<u>NS</u>	<u>NS</u>	<u>CM<sup>1</sup></u>
<i>Enter Mode</i>	<u>NS</u>	<u>NS</u>	<u>NS</u>	<u>CM<sup>1</sup></u>
<i>Exit Mode</i>	<u>NS</u>	<u>NS</u>	<u>NS</u>	<u>CM<sup>1</sup></u>
<i>Attention</i>	<u>NS</u>	<u>Q</u>	<u>NS</u>	
<b>Received Command</b>				
<i>Discover Identity</i>	<u>NS</u>	<u>R/NK</u>	<u>M</u>	<u>CM<sup>3</sup></u>
<i>Discover SVIDs</i>	<u>NS</u>	<u>NK</u>	<u>NK</u>	<u>CM<sup>3</sup></u>
<i>Discover Modes</i>	<u>NS</u>	<u>NK</u>	<u>NK</u>	<u>CM<sup>3</sup></u>
<i>Enter Mode</i>	<u>NS</u>	<u>NK</u>	<u>NK</u>	<u>CM<sup>3</sup></u>
<i>Exit Mode</i>	<u>NS</u>	<u>NK</u>	<u>NK</u>	<u>CM<sup>3</sup></u>
<i>Attention</i>	<u>O/NS</u>	<u>NK</u>	<u>NS</u>	
<u>Note 1: Shall only be supported by a DFP.</u>				
<u>Note 2: May be transmitted by a UFP/Source during discovery (see Sections 6.4.4.3.1 and 8.3.3.10.11).</u>				
<u>Note 3: Shall only be supported by a UFP or Cable Plug.</u>				

## 7. Power Supply

### 7.1 Source Requirements

#### 7.1.1 Behavioral Aspects

The Source in a Provider or Provider/Consumer exhibits the following behaviors.

- Shall be backward compatible with legacy  $V_{BUS}$  ports.
- Shall supply the default [USB 2.0], [USB 3.1], [USBType-C 1.0] or [BC 1.2] voltage and current to  $V_{BUS}$  when the USB cable is attached (USB Default Operation).
- Shall supply the default [USB 2.0], [USB 3.1], [USBType-C 1.0] or [BC 1.2] voltage and current to  $V_{BUS}$  when a Contract does not exist (USB Default Operation).
- Shall return *vSafe0V* for some time then return to *vSafe5V* when *Hard Reset* Signaling is received.
- Shall control  $V_{BUS}$  voltage transitions as bound by undershoot, overshoot and transition time requirements.

The Source in a Consumer/Provider exhibits the following behaviors.

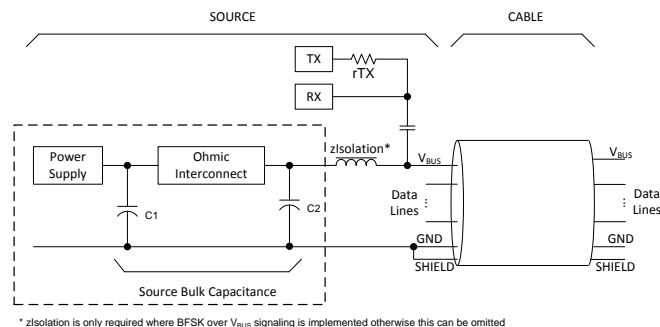
- Shall support Dead Battery operation as defined in Section 4.1.1 for Type-A to Type-B connections and as defined in [USBType-C 1.0] for Type-C to Type-C connections.
- Shall return to Sink operation when *Hard Reset* Signaling is received.
- Shall control  $V_{BUS}$  voltage transitions as bound by undershoot, overshoot and transition time requirements.

#### 7.1.2 Source Bulk Capacitance

The Source shall have a bulk capacitance located between the output of the power supply and the transceiver isolation impedance as shown in Figure 7-1. The Source bulk capacitance shall not be placed between the transceiver isolation impedance and the USB receptacle. The bulk capacitance consists of C1 and C2 as shown in Figure 7-1. The Ohmic Interconnect may consist of PCB traces for power distribution or power switching devices. The capacitance may be a single capacitor, a capacitor bank or distributed capacitance. If the power supply is shared across multiple ports, bulk capacitance is defined as *cSrcBulkShared*. If the power supply is dedicated to a single Port, the minimum bulk capacitance is defined as *cSrcBulk*.

The Source bulk capacitance is allowed to change for a newly negotiated power level. The capacitance change shall occur before the Source is ready to operate at the new power level. During a Power Role Swap, the Default Source shall transition to Swap Standby before operating as the new Sink. Any change in bulk capacitance required to complete the Power Role Swap shall occur during Swap Standby.

Figure 7-1 Placement of Source Bulk Capacitance



### 7.1.3 Types of Sources

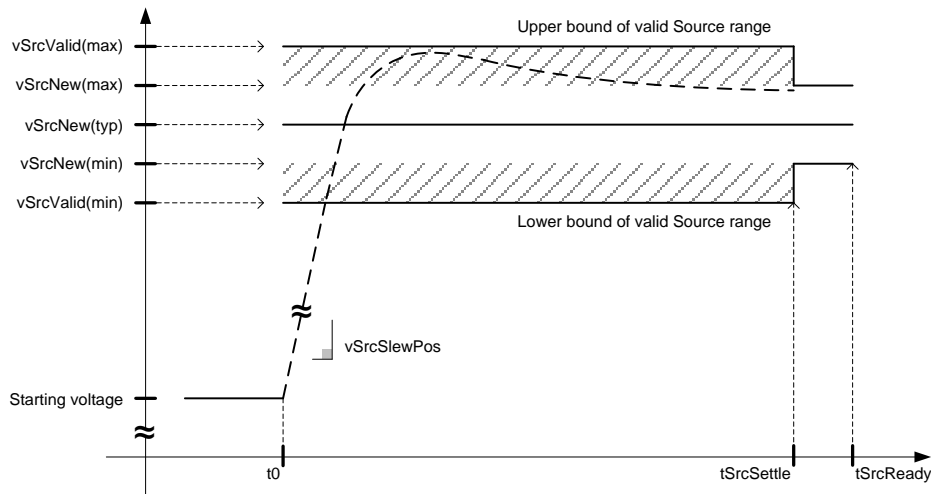
Consistent with the Power Data Objects discussed in Section 6.4.1, the three possible power supply types that are available as Sources in a USB Power Delivery System are:

- Fixed Supply is used to expose well regulated fixed voltage power supplies. Sources shall support at least one fixed power source capable of supplying *vSafe5V*. The output voltage of a Fixed Supply shall be specified in terms of an absolute tolerance, *vSrcNew*, relative to the nominal value. Refer to Table 7-22 for the output voltage tolerance specification.
- Variable power supply (non-battery) is used to expose very poorly regulated Sources. The output voltage of a Variable power supply (non-battery) shall be specified as *vSrcNew*, in terms of an absolute maximum output voltage and an absolute minimum output voltage.
- Battery Supply is used to expose batteries that can be connected directly as a Source to  $V_{BUS}$ . The output voltage of a Battery Supply shall be specified as *vSrcNew*, in terms of an absolute maximum output voltage and an absolute minimum output voltage.

### 7.1.4 Positive Voltage Transitions

The Source shall transition  $V_{BUS}$  from the starting voltage to the higher new voltage in a controlled manner. The negotiated new voltage (e.g., 12V or 20V) defines the typical value for *vSrcNew*. If the newly negotiated voltage is 5V, then *vSafe5V* limits shall apply. During the positive transition the Source shall be able to supply the Sink standby power and the transient current to charge the total bulk capacitance on  $V_{BUS}$ . The slew rate of the positive transition shall not exceed *vSrcSlewPos*. The Source output voltage during a positive transition shall settle within the Source output tolerance range *vSrcNew*. The Source shall be able to supply the negotiated power level at the new voltage by *tSrcReady*. The positive voltage transition shall remain monotonic while the transitioning voltage is below *vSrcValid* min and shall remain within the *vSrcValid* range upon crossing *vSrcValid* min as shown in Figure 7-2. See Section 7.1.9 for requirements relating to transients. The starting time,  $t_0$ , in Figure 7-2 starts *tSrcTransition* after the last bit of the *EOP* of the *GoodCRC* Message has been transmitted on  $V_{BUS}$  received by the Source.

Figure 7-2 Transition Envelope for Positive Voltage Transitions



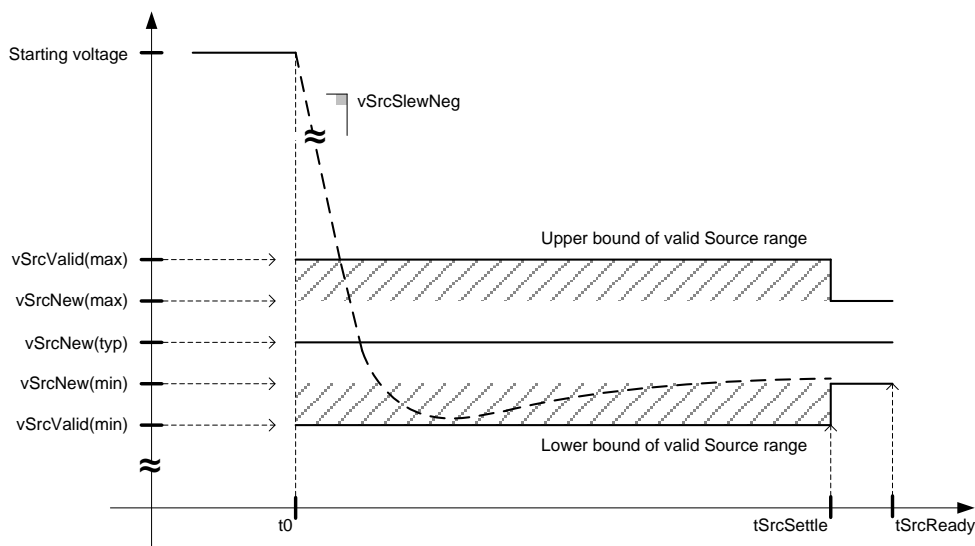
When a positive voltage transition is started, the  $V_{BUS}$  voltage may decrease at the onset of the transition. When the starting voltage on  $V_{BUS}$  for the transition is  $v_{Safe5V}$ , the  $V_{BUS}$  voltage shall not droop below  $v_{Safe5V}$  min. When the starting voltage on  $V_{BUS}$  is any voltage other than  $v_{Safe5V}$ , the  $V_{BUS}$  voltage shall not droop below  $v_{SrcValid}$  min.

During normal operation, not including  $V_{BUS}$  transitions or discharge,  $V_{BUS}$  shall not go beyond the limits of  $v_{SrcValid}$  (or  $v_{Safe5V}$  if  $V_{BUS}$  is 5V). This limitation shall apply to static and transient  $V_{BUS}$  behavior across all single port and multi-port power configurations.

### 7.1.5 Negative Voltage Transitions

Negative voltage transitions are defined as shown in Figure 7-3 and are specified in a similar manner to positive voltage transitions. Figure 7-3 does not apply to  $v_{Safe0V}$  transitions. If the newly negotiated voltage is 5V, then  $v_{Safe5V}$  limits shall apply. The slew rate of the negative transition shall not exceed  $v_{SrcSlewNeg}$ . The negative voltage transition shall remain monotonic while the transitioning voltage is above  $v_{SrcValid}$  max and shall remain within the  $v_{SrcValid}$  range upon crossing  $v_{SrcValid}$  max as shown in Figure 7-3. [See Section 7.1.9 for requirements relating to transients.](#) The starting time,  $t_0$ , in Figure 7-3 starts ***tSrcTransition*** after the last bit of the *EOP* of the *GoodCRC* Message has been ***transmitted on  $V_{BUS}$  received by the Source.***

Figure 7-3 Transition Envelope for Negative Voltage Transitions



### 7.1.6 Response to Hard Resets

**Hard Reset** Signaling indicates a communication failure has occurred and the Source shall ***revert drive  $V_{BUS}$  to  $v_{Safe0V}$  a safe operating voltage as specified shown*** in Figure 7-4. ~~The safe operating voltage is specified as-~~ The USB connection may reset during a Hard Reset since the  $V_{BUS}$  voltage will be less than  $v_{Safe5V}$  for an extended period of time. After establishing the safe voltage condition on  $V_{BUS}$ , the power supply shall wait ***tSrcRecover*** before powering  $V_{BUS}$  to  $v_{Safe5V}$ . ***A Source using a Type-C connector shall conform to the  $V_{CONN}$  timing as specified in [USBType-C 1.0].***

Device operation during and after a Hard Reset is defined as follows:

- Self-powered devices should not disconnect from USB during a Hard Reset (see Section 9.1.2).
- Self-powered devices operating at more than  $v_{Safe5V}$  may not maintain full functionality after a **Hard Reset**.

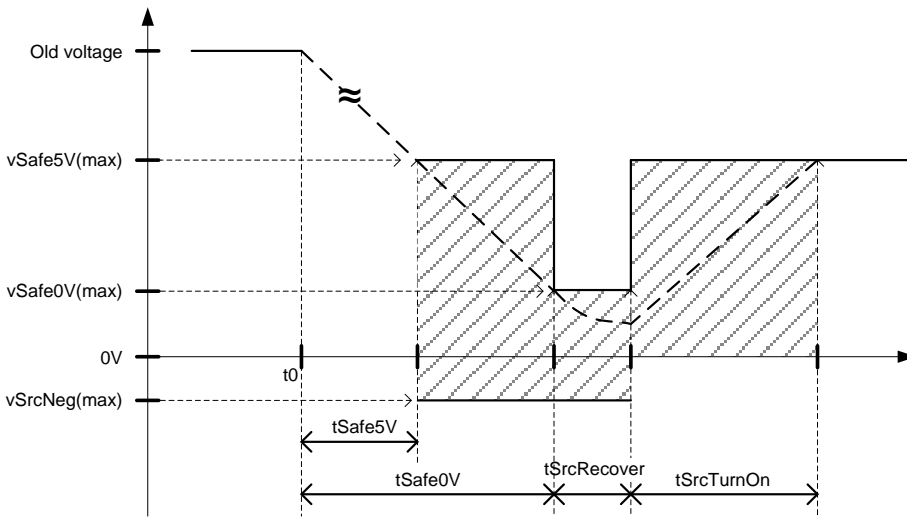
- Bus powered devices will disconnect from USB during a Hard Reset due to the loss of their power source.

When a Hard Reset occurs the Source shall start to transition the  $V_{BUS}$  voltage to  $v_{Safe0V}$  and the  $V_{CONN}$  voltage to  $v_{ConnDischarge}$  (see [USBType-C 1.0]) either:

- $t_{PSHardReset}$  after the last bit of the *Hard Reset* Signaling has been received from the Sink or
- $t_{PSHardReset}$  after the last bit of the *Hard Reset* transmitted on  $V_{BUS}$  Signaling has been sent by the Source

The Source shall meet both  $t_{Safe5V}$  and  $t_{Safe0V}$  relative to the start of the voltage transition as shown in Figure 7-4.

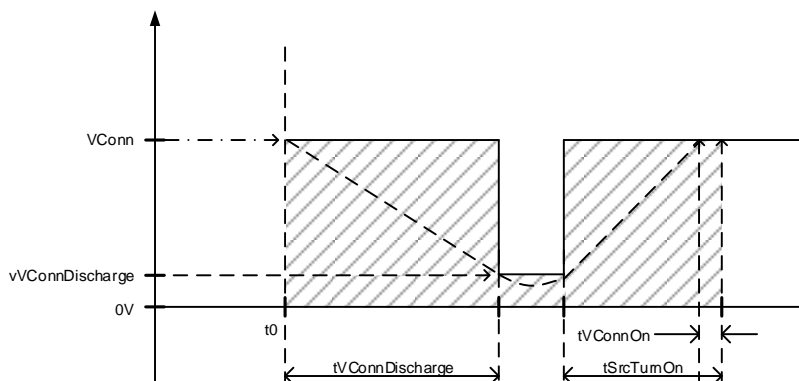
Figure 7-4 Source  $V_{BUS}$  Response to Hard Reset



The Source shall meet  $t_{VConnDischarge}$  relative to the start of the voltage transition as shown in Figure 7-5. The source shall meet  $t_{VconnOn}$  relative to  $V_{BUS}$  reaching  $v_{Safe5V}$ . Note:  $t_{VConnDischarge}$ ,  $v_{ConnDischarge}$  and  $t_{VconnOn}$  are defined in [USBType-C 1.0].



**Figure 7-5 Source VCONN Response to Hard Reset**



### 7.1.7 Changing the Output Power Capability

Some USB Power Delivery negotiations will require the Source to adjust its output power capability without changing the output voltage. In this case the Source shall be able to supply a higher or lower load current within *tSrcReady*.

### 7.1.8 Safe Operating Considerations

The Source shall provide short circuit current limiting to protect its Port from prolonged exposure to excessive current draw. For the purposes of this standard, excessive current draw is defined as any current that significantly exceeds the output capability of the power supply or the contact rating of the receptacle. The maximum steady state operating current of the Source shall be the lesser of these two values. For example, the Source Port powered by a 6A power supply connected to two 3A receptacles will provide a maximum continuous operating current of 3A on each receptacle. The short circuit protection mechanism shall be designed to avoid tripping at the maximum continuous operating current of the Source Port and shall not interfere with negotiated current levels. The protected Port shall recover automatically, by performing a Hard Reset, when the fault is removed and resume operation at *vSafe5V* within *tShortCctRecover* of the fault removal. Mechanical or user intervention shall not be required to reset the short circuit protection mechanism unless the Provider is purely a power source, which does not support USB communication, when mechanical intervention may be used.

Safe operation mandates that Power Delivery Sources shall be tolerant of *vSafe5V* being present on  $V_{BUS}$  when simultaneously applying power to  $V_{BUS}$ . This shall not interfere with normal PD communication.

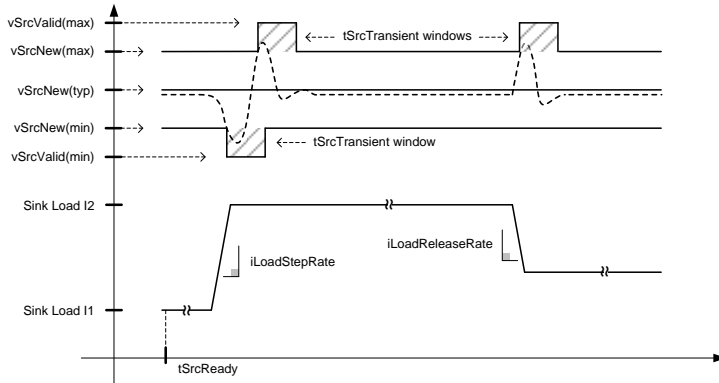
A USB detach can be detected mechanically (e.g. Standard-A receptacle with insertion detect) or electrically (e.g. CC detection on the Type-C or via the ID pin on the Micro-A plug or using Attach Detection Protocol from [USBOTG 2.0]). When the Source is capable of detecting a detach either mechanically or electrically, the Source shall transition to *vSafe0V* by *tSafe0V* relative to when the detach event occurred (i.e. physical removal of the plug). During the transition to *vSafe0V* the  $V_{BUS}$  voltage shall be below *vSafe5V* max by *tSafe5V* relative to when the detach event occurred (i.e. physical removal of the plug) and shall not exceed *vSafe5V* max after this time.

### 7.1.9 Output Voltage Tolerance and Range

The *vSrcNew* and *vSrcValid* limits apply to  $V_{BUS}$  Source output voltage after the voltage transition is complete (i.e. after *tSrcReady*) as follows. During static load conditions the Source output voltage shall remain within the *vSrcNew* range. The range defined by *vSrcNew* accounts for DC regulation accuracy, line load regulation and output ripple. In response to a transient load event (as described below) the Source output voltage shall not go beyond the range specified by *vSrcValid*. The amount of time the Source output voltage can be in the band between *vSrcNew* and

$v_{SrcValid}$  shall not exceed  $t_{SrcTransient}$ . Refer to Table 7-22 for the output voltage tolerance specifications. Figure 7-6 illustrates the application of these limits.

**Figure 7-6 Application of  $v_{SrcNew}$  and  $v_{SrcValid}$  limits after  $t_{SrcReady}$**



**The Source output voltage tolerance, shall be measured at the connector receptacle. The output tolerance includes load regulation, transient response and output ripple.**

The Source output voltage shall be measured at the connector receptacle. The stability of the Source shall be tested in 25% load step increments from minimum load to maximum load and also from maximum load to minimum load. The transient behavior of the load current is defined in Section 7.2.6. The time between each step ~~should~~shall be sufficient to allow for the output voltage to settle ~~in~~ between load steps. ~~The output voltage tolerance does not include undershoot and overshoot that may occur during voltage transitions.~~ In some systems it may be necessary to design the Source to compensate for the voltage drop between the output stage of the power supply electronics and the receptacle contact. The determination of when compensation may be necessary is left to the discretion of the implementation. ~~Refer to for the output voltage tolerance specifications that shall be used for the Source.~~

### 7.1.10 Charging and Discharging the Bulk Capacitance on $V_{BUS}$

The Source shall charge and discharge the bulk capacitance on  $V_{BUS}$  whenever the Source voltage is negotiated to a different value. The charging or discharging occurs during the voltage transition and shall not interfere with the Source's ability to meet  $t_{SrcReady}$ .

### 7.1.11 Swap Standby for Sources

Sources and Sinks of a Dual-Role Port shall support Swap Standby. Swap Standby occurs for the Source after the Source power supply has discharged the bulk capacitance on  $V_{BUS}$  to  $v_{Safe0V}$  as part of the Power Role Swap transition.

While in Swap Standby:

- The Source shall not drive  $V_{BUS}$  that is therefore expected to remain at  $v_{Safe0V}$ .
- Any discharge circuitry that was used to achieve  $v_{Safe0V}$  shall be removed from  $V_{BUS}$ .
- The Dual-Role Port shall be configured as a Sink
- The USB connection shall not reset even though  $v_{Safe5V}$  is no longer present on  $V_{BUS}$  (see Section 9.1.2).

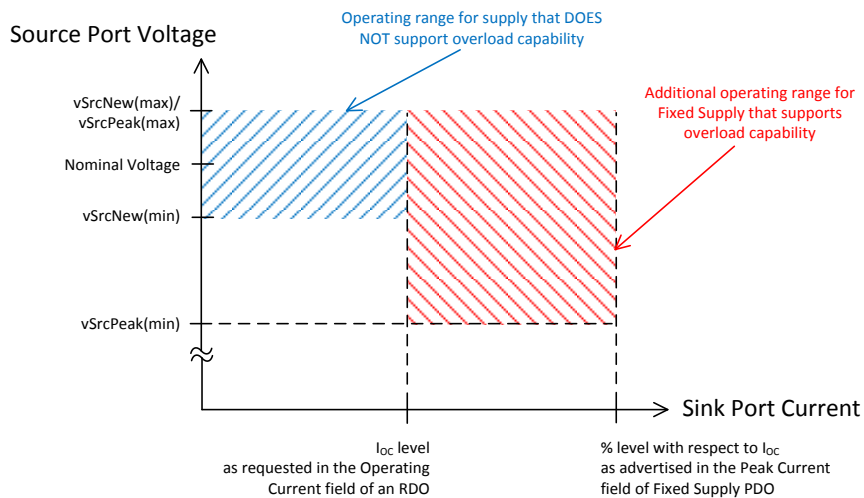
The  $PS\_RDY$  Message associated with the Source being in Swap Standby shall be sent after the  $V_{BUS}$  drive is removed. The time for the Source to transition to Swap Standby shall not exceed  $t_{SrcSwapStdby}$ . Upon entering Swap Standby the Source has relinquished its role as Source and is ready to become the new Sink. The transition time from Swap

Standby to being the new Sink shall be no more than  $t_{NewSink}$ . The new Sink may start using power after the new Source sends the **PS\_RDY** Message.

### 7.1.12 Source Peak Current Operation

A Source that has the Fixed Supply PDO Peak Current bits set to 01b, 10b and 11b shall be designed to support one of the overload capabilities defined in Table 6-7. The overload conditions are bound in magnitude, duration and duty cycle as listed in Table 6-7. Sources are not required to support continuous overload operation. When overload conditions occur, the Source is allowed the range of  $v_{SrcPeak}$  (instead of  $v_{SrcNew}$ ) relative to the nominal value (see Figure 7-7). When the overload capability is exceeded, the Source is expected take whatever action is necessary to prevent electrical or thermal damage to the Source. The Source may send a new **Source\_Capabilities** Message with the Fixed Supply PDO Peak Current bits set to 00b to prohibit overload operation even if an overload capability was previously negotiated with the Sink.

**Figure 7-7. Source Peak Current Overload**



### 7.1.13 BFSK over $V_{BUS}$ Considerations for Sources

The following sections list Source power supply considerations when applying BFSK signaling to  $V_{BUS}$ .

#### 7.1.13.1 Transceiver Isolation Impedance

For BFSK signaling over  $V_{BUS}$  an isolation impedance as specified in Section 5.8.2.2 is used to isolate the transceiver from the capacitive loading of the Source. The DC component of the isolation impedance introduces additional voltage loss between the Source power path and the connector. The Source output impedance, the total  $V_{BUS}$  bulk capacitance, the transceiver isolation impedance(s), the USB cable and the Sink network creates a complex output isolation impedance that should be taken into consideration when designing the Source. Refer to Appendix C.1 for more information on this topic.

#### 7.1.13.2 Noise Injected on $V_{BUS}$

The Source shall not interfere with the USB Power Delivery BFSK waveform that is transmitted on  $V_{BUS}$ . Power stage switching harmonics or poor layout and component selection may result in a significant amount of in-band noise injected on  $V_{BUS}$ . The characteristics of the noise will be implementation specific and in some cases an additional filter may be required to meet the in-band signal-to-noise ratio requirement of  $snr_{Src}$ . Refer to Section 5.8.2 for the

reference impedance, carrier signal amplitude and transmission bandwidth of the USB Power Delivery BFSK waveform. Refer to Appendix C for more information on this topic.

Out-of-band noise (including spurs) can also affect the reception of the BFSK signal; therefore the Source shall limit its out-of-band noise below the spectrum shown in Figure 7-8 and the levels indicated in Table 7-1. Figure 7-8 shows the level of allowed noise relative to the level of the allowed in-band noise [for a Source: dBV (minimum value of  $v_{TX}$ ) -  $snr_{Src}$ ]. The limits shown in Figure 7-8 and

Table 7-1 ~~does~~ not include any other regulations (such as FCC regulations) that govern signal emissions.

The  $V_{BUS}$ -to-GND noise measurement shall be made at the connector. The noise measurement can be made using a breakout board or equivalent method that does not introduce additional loading of the AC impedance of the  $V_{BUS}$  wire. Such measurement access methods will require calibration to ensure accurate measurement results.

Figure 7-8 Noise Spectral Mask, given in absolute terms relative to the maximum value of  $v_{TX}$

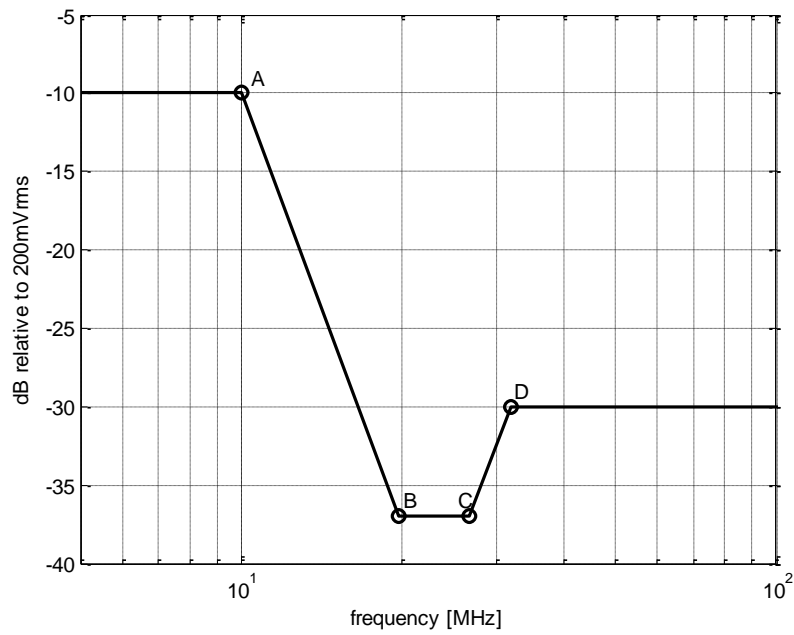


Table 7-1 Noise Spectral Mask Corners

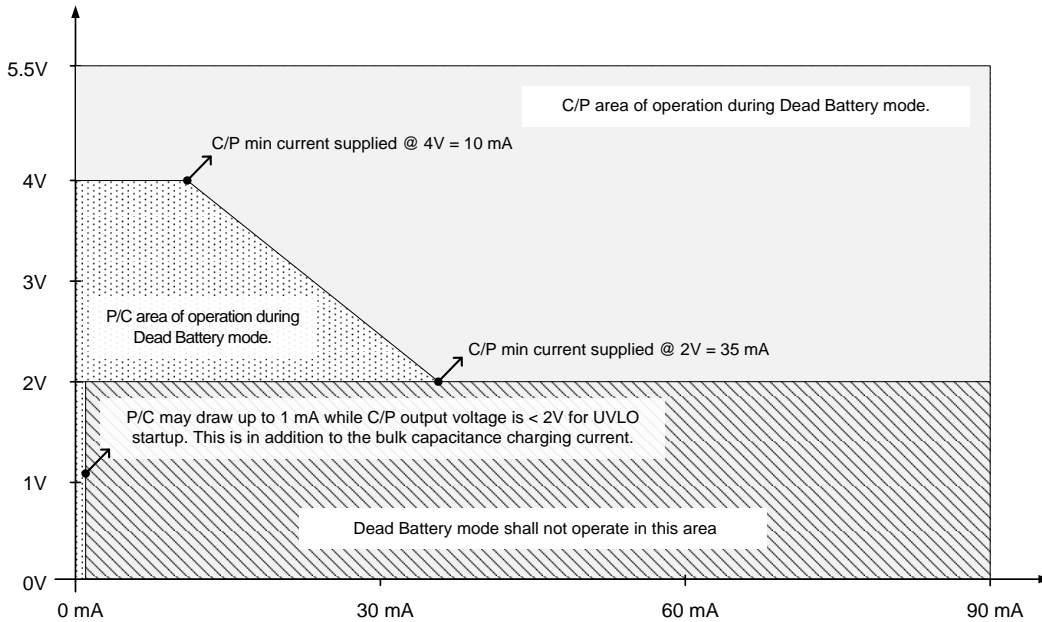
Frequency (MHz)	Maximum Allowed Signal Level	
	(dB)	(mVpp)
<10 (A)	-10	178.7
19.7(B) to 26.7 (C)	-37	7.92
>32 (D)	-30	17.8

### 7.1.13.3 Dead Battery Operation

Dead Battery operation shall be supported by all Consumer/Providers. A Consumer/Provider shall apply *vSafeDB* to its Port when it detects its Port voltage to be *vSafe0V* (see Section 4.1). The Operation Region of *vSafeDB* source shall be as shown in Figure 7-9. For example if the Provider/Consumer attempts to draw 30mA the voltage will remain above 2.4V; if the Provider/Consumer attempts to draw more than 90mA the voltage will drop to *vSafe0V*. The Consumer/Provider shall fully support the *vSafeDB* Operating Region as shown in Figure 7-9 on reaching 2V. The output voltage shall reach 2V and comply with the *vSafeDB* Operating Region within *tTurnOnSafeDB* of when it first applies voltage to  $V_{BUS}$ . The Source bulk capacitance shall be limited to *cSrcBulkDB* for the Consumer/Provider when applying *vSafeDB*. The Consumer/Provider shall be limited to *cSnkBulk* when not applying *vSafeDB*.

A detailed description of the Consumer/Provider's behavior during Dead Battery operation is provided in Section 8.3.3.6.1.5.

Figure 7-9 vSafeDB Operating Region



## 7.2 Sink Requirements

### 7.2.1 Behavioral Aspects

The Sink in a Consumer or Consumer/Provider exhibits the following behaviors.

- Shall be backward compatible with legacy  $V_{BUS}$  ports.
- Shall draw the default [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USBType-C 1.0\]](#) or [\[BC 1.2\]](#)  $V_{BUS}$  current when the USB cable is attached (USB Default Operation).
- Shall draw the default [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USBType-C 1.0\]](#) or [\[BC 1.2\]](#)  $V_{BUS}$  current when a Contract does not exist (USB Default Operation).
- Shall return to the default [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USBType-C 1.0\]](#) or [\[BC 1.2\]](#)  $V_{BUS}$  when responding to a Hard Reset (USB Default Operation).
- Shall control  $V_{BUS}$  in-rush current when increasing current consumption.

The Sink in a Provider/Consumer exhibits the following behaviors.

- May support Dead Battery operation as defined in Section 4.1.1 for Type-A to Type-B connections and as defined in [\[USBType-C 1.0\]](#) for Type-C to Type-C connections.
- Shall return to Source operation when responding to a Hard Reset.
- Shall control  $V_{BUS}$  in-rush current when increasing and decreasing current consumption.

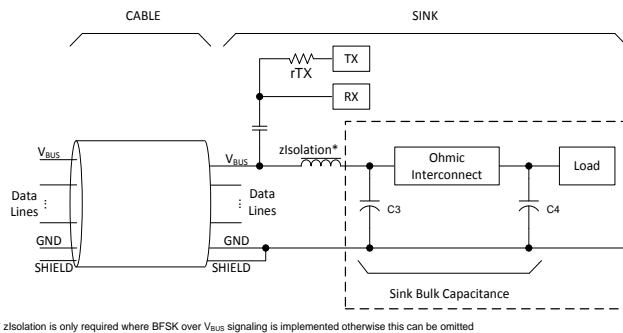
### 7.2.2 Sink Bulk Capacitance

The Sink shall have a bulk capacitance,  $c_{SnkBulk}$ , located between the input of the power supply and the transceiver isolation impedance as shown in Figure 7-10. The Sink bulk capacitance shall not be placed between the transceiver isolation impedance and the USB receptacle. The bulk capacitance consists of C3 and C4 as shown in Figure 7-10. The Ohmic Interconnect may consist of PCB traces for power distribution or power switching devices. The capacitance may be a single capacitor, a capacitor bank or distributed capacitance. An upper bound of  $c_{SnkBulkPd}$  shall not be exceeded so that the transient charging, or discharging, of the total bulk capacitance on  $V_{BUS}$  can be accounted for during voltage transitions. ~~If more bypass~~

~~The Sink bulk capacitance that is within the  $c_{SnkBulk}$  max or  $c_{SnkBulkPd}$  max is required in the device, then the device must incorporate some form of  $V_{BUS}$  surge current limiting as described in Section 11.4.4.1.~~

~~The Sink bulk capacitance limits~~ is allowed to change to support a newly negotiated power level. The capacitance can be changed when the Sink enters Sink Standby or during a voltage transition or when the Sink begins to operate at the new power level. Regardless of when the change occurs, the capacitance change shall occur in such a manner that does not introduce a  $V_{BUS}$  transient current greater than  $i_{CapChange}$ . During a Power Role Swap the Default Sink shall transition to Swap Standby before operating as the new Source. Any change in bulk capacitance required to complete the Power Role Swap shall occur during Swap Standby.

Figure 7-10 Placement of Sink Bulk Capacitance



\* zIsolation is only required where BFSK over V<sub>BUS</sub> signaling is implemented otherwise this can be omitted

### 7.2.3 Sink Standby

The Sink shall transition to Sink Standby before a positive or negative voltage transition of V<sub>BUS</sub>. During Sink Standby the Sink shall reduce its average power draw during Sink Standby to *pSnkStdby* and its peak power draw shall not exceed *pSnkStdbyLimit* to allow for. This allows the Source to manage the voltage transition as well as supply sufficient operating current to the Sink to maintain PD operation during the transition. The Sink shall complete this transition to Sink Standby within *tSnkStdby* after evaluating the *Accept* Message from the Source. The transition when returning to Sink operation from Sink Standby shall be completed within *tSnkNewPower*. The *pSnkStdby* requirement shall only apply if the Sink power draw is higher than this level.

### 7.2.4 Suspend Power Consumption

When Source has set its USB Suspend Supported flag (see Section 6.4.1.2.3.2), a Sink shall go to the lowest power state during USB suspend. The lowest power state shall be *pSnkSusp* or lower for a PDUSB Peripheral and *pHubSusp* or lower for a PDUSB Hub. There is no requirement for the Source voltage to be changed during USB suspend.

### 7.2.5 Zero Negotiated Current

When a Sink Requests zero current as part of a power negotiation with a Source, the Sink shall go to the lowest power state, *pSnkSusp* or lower, where it can still communicate using PD signaling.

### 7.2.6 Transient Load Behavior

When a Sink's operating current changes due to a load step, load release or any other change in load level, the positive or negative overshoot of the new load current shall not exceed the range defined by *iOvershoot*. For the purposes of measuring *iOvershoot* the new load current value is defined as the average steady state value of the load current after the load step has settled. The rate of change of any shift in Sink load current during normal operation shall not exceed *iLoadStepRate* (for load steps) and *iLoadReleaseRate* (for load releases) as measured at the Sink receptacle.

### 7.2.7 Swap Standby for Sinks

The Sink capability in a Dual-Role Port shall support Swap Standby. Swap Standby occurs for the Sink after evaluating the *Accept* Message from the Source during a Power Role Swap negotiation. While in Swap Standby the Sink's current draw shall not exceed *iSnkSwapStdby* from V<sub>BUS</sub> and the Dual-Role Port shall be configured as a Source after V<sub>BUS</sub> has been discharged to *vSafe0V* by the existing Initial Source. The Sink's USB connection should not be reset even though *vSafe5V* is not present on the V<sub>BUS</sub> conductor (see Section 9.1.2). The time for the Sink to transition to SwapStandby shall be no more than *tSnkSwapStdby*. When in Swap Standby the Sink has relinquished its role as Sink and will



prepare to become the new Source. The transition time from Swap Standby to new Source shall be no more than  $t_{NewSrc}$ .

### 7.2.8 Sink Peak Current Operation

Sinks shall only make use of a Source overload capability when the corresponding Fixed Supply PDO Peak Current bits are set to 01b, 10b and 11b (see Section 6.4.1.2.3.6). Sinks shall manage thermal aspects of the overload event by not exceeding the average negotiated output of a Fixed Supply that supports Peak Current operation.

Sinks that depend on the Peak Current capability for enhanced system performance shall also function correctly when attached to a Source that does not offer the Peak Current capability or when the Peak Current capability has been inhibited by the Source.

### 7.2.9 BFSK over $V_{BUS}$ Considerations for Sinks

The following sections list Sink considerations when applying BFSK signaling to  $V_{BUS}$ .

#### 7.2.9.1 Transceiver Isolation Impedance

For BFSK signaling over  $V_{BUS}$  an isolation impedance as specified in Section 5.8.2.2 is used to isolate the transceiver from the capacitive loading of the Sink. The DC component of the isolation impedance introduces additional voltage loss between the connector and the Sink power path. The Sink input impedance, the total  $V_{BUS}$  bulk capacitance, the transceiver isolation impedance(s), the USB cable and the Source creates a complex input isolation impedance that should be taken into consideration when designing the Sink. Refer to Appendix C.1 for more information on this topic.

#### 7.2.9.2 Noise Reflected on $V_{BUS}$

The Sink shall not interfere with the USB Power Delivery BFSK waveform that is transmitted on  $V_{BUS}$ . Power stage switching harmonics or poor layout and component selection may result in a significant amount of in-band noise reflected on  $V_{BUS}$ . The characteristics of the noise will be implementation specific and in some cases an additional filter may be required to meet the in-band signal-to-noise ratio requirement of  $snr_{Snk}$ . Refer to Section 5.8.2 for the reference impedance, carrier signal amplitude and transmission bandwidth of the USB Power Delivery BFSK waveform. Refer to Appendix C.1.1 for more information on this topic.

Out-of-band noise (including spurs) can also affect the reception of the BFSK signal; therefore the Sink shall limit its out-of-band noise below the spectrum shown in Figure 7-11 and as detailed in Table 7-2. Figure 7-11 also shows the noise level that shall be allowed relative to the level of the allowed in-band noise [for a Sink:  $dBV$  (minimum value of  $v_{TX}$ ) -  $snr_{Snk}$ ]. The limits shown in Figure 7-11 and Table 7-2 do not include any other regulations (such as FCC regulations) that govern signal emissions.

The  $V_{BUS}$ -to-GND noise measurement shall be made at the connector.

Figure 7-11 Noise Spectral Mask, given in absolute terms relative to the maximum value of vTX

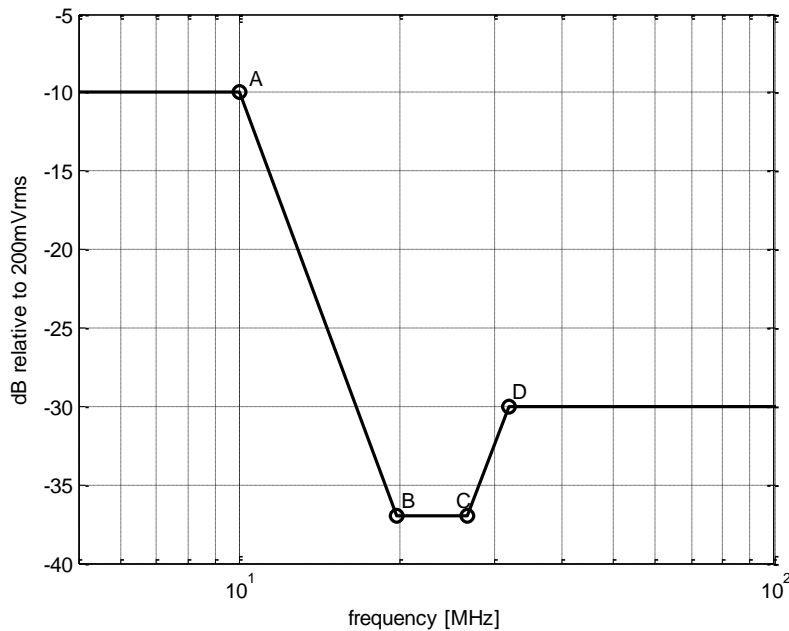


Table 7-2 Noise Spectral Mask Corners

Frequency (MHz)	Maximum Allowed Signal Level	
	(dB)	(mVpp)
<10 (A)	-10	178.7
19.7(B) to 26.7 (C)	-37	7.92
>32 (D)	-30	17.8

### 7.2.9.3 Dead Battery Operation

Dead Battery operation is only supported by Provider/Consumers that want to perform an Implicit (i.e. non-negotiated) Power Role Swap when attached to a Consumer/Provider. When a Provider/Consumer Port is not able to Source power (or has chosen to not source power) and would like to be powered by a Consumer/Provider, the Provider/Consumer shall discharge its Port voltage to *vSafe0V*, present cSnkBulk and be ready to power up the transceiver if *vSafeDB* is applied to its Port. When operating from *vSafeDB*, the Provider/Consumer, acting as a Sink, shall be able to operate from any Source that complies with the Operating Region of Figure 7-9. The Provider/Consumer may regain the ability to Source power (or decide to do so) at any time and shall not apply power to its Port when *vSafeDB* is present. The Provider/Consumer shall not commence Dead Battery operation or draw more than 1 mA until the voltage on  $V_{BUS}$  is above 2V. The Sink bulk capacitance shall be limited to *cSnkBulkDB* for the Provider/Consumer acting as the Sink during Dead Battery operation. A detailed description of the Provider/Consumer's behavior during Dead Battery operation is provided in Sections 4.1 and 7.1.13.3.

### 7.2.10 Safe Operating Considerations

When a Source is detached from a Sink, the Sink shall continue to draw power from its input bulk capacitance until  $V_{BUS}$  is discharged to  $v_{Safe5V}$  or lower by no longer than  $t_{Safe5V}$  from the detach event. This safe Sink requirement shall apply to all Sinks operating with a negotiated  $V_{BUS}$  level greater than  $v_{Safe5V}$  and shall apply during all low power and high power operating modes of the Sink.

If the detach is detected during a Sink low power state, such as USB Suspend, the Sink can then draw as much power as needed from its bulk capacitance since a Source is no longer attached. In order to achieve a successful detach detect based on  $V_{BUS}$  voltage level droop, the Sink power consumption must be high enough so that  $V_{BUS}$  will decay below  $v_{SrcValid}(min)$  well within  $t_{Safe5V}$  after the Source bulk capacitance is removed due to the detach. Once adequate  $V_{BUS}$  droop has been achieved, a discharge circuit can be enabled to meet the safe Sink requirement.

To illustrate the point, the following set of Sink conditions will not meet the safe Sink requirement without additional discharge circuitry:

- Negotiated  $V_{BUS} = 20V$
- Maximum allowable supplied  $V_{BUS}$  voltage = 21.5V
- Maximum bulk capacitance = 30 $\mu$ F
- Power consumption at detach = 12.5mW

When the detach occurs (hence removal of the Source bulk capacitance) the 12.5mW power consumption will draw down the  $V_{BUS}$  voltage from the worst-case maximum level of 21.5V to 17V in approximately 205 ms. At this point, with  $V_{BUS}$  well below  $v_{SrcValid}(min)$  an approximate 100 mW discharge circuit can be enabled to increase the rate of Sink bulk capacitance discharge and meet the safe Sink requirement. The power level of the discharge circuit is dependent on how much time is left to discharge the remaining voltage on the Sink bulk capacitance. If a Sink has the ability to detect the detach in a different manner and in much less time than  $t_{Safe5V}$ , then this different manner of detection can be used to enable a discharge circuit, allowing even lower power dissipation during low power modes such as USB Suspend.

In most applications, the safe Sink requirement will limit the maximum Sink bulk capacitance well below the  $c_{SnkBulkPd}$  limit. A detach occurring during Sink high power operating modes must quickly discharge the Sink bulk capacitance to  $v_{Safe5V}$  or lower as long as the Sink continues to draw adequate power until  $V_{BUS}$  has decayed to  $v_{Safe5V}$  or lower.

### 7.3 Transitions

The following sections illustrate the power supply's response to various types of negotiations. The negotiation cases take into consideration for the examples are as follows:

- Higher Power Transitions
  - Increase the current
  - Increase the voltage
  - Increase the voltage and the current
- Relatively Constant Power Transitions
  - Increase the voltage and decrease the current
  - Decrease the voltage and increase the current
- Lower Power Transitions
  - Decrease the current
  - Decrease the voltage
  - Decrease the voltage and the current
- Power Role Swap Transitions
  - Source requests a Power Role Swap
  - Sink requests a Power Role Swap
- Go To Minimum Current Transition
- Response to *Hard Reset* Signaling
  - Source issues *Hard Reset* Signaling
  - Sink issues *Hard Reset* Signaling
  - New Source issues *Hard Reset* Signaling and New Sink Receives *Hard Reset* Signaling.
  - New Source issues *Hard Reset* Signaling and New Sink Does Not Receive *Hard Reset* Signaling.
  - New Sink issues *Hard Reset* Signaling and New Source Receives *Hard Reset* Signaling.
  - New Sink issues *Hard Reset* Signaling and New Source Does Not Receive *Hard Reset* Signaling.
- Dead Battery Operation.
- No change in Current or Voltage.

The transition from [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USBType-C 1.0\]](#) or [\[BC 1.2\]](#) operation into Power Delivery Mode can also lead to a Power Transition since this is the initial Contract negotiation. The following types of Power Transitions shall also be applied when moving from [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USBType-C 1.0\]](#) or [\[BC 1.2\]](#) operation into Power Delivery Mode:

- High Power
- Relatively Constant Power
- Lower Power Transitions
- No change in Current or Voltage.

### 7.3.1 Increasing the Current

The interaction of the System Policy, Device Policy, and power supply that shall be followed when increasing the current is shown in Figure 7-12. The sequence that shall be followed is described in Table 7-3. The timing parameters that shall be followed are listed in Table 7-22 and Table 7-23. Note in this figure, the Sink has previously sent a *Request* Message to the Source.

Figure 7-12 Transition Diagram for Increasing the Current

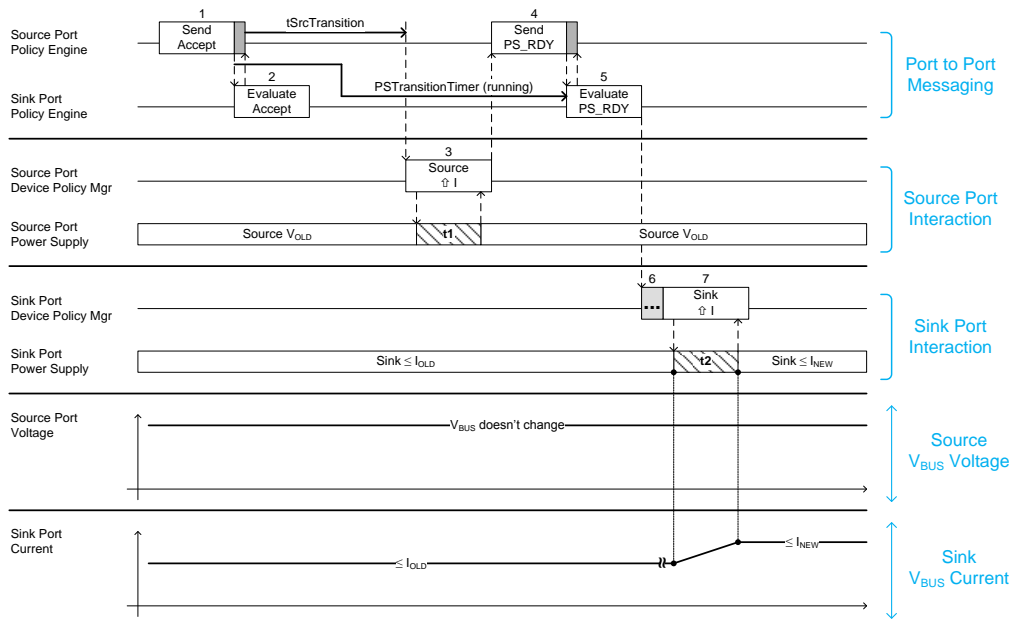


Table 7-3 Sequence Description for Increasing the Current

Step	Source Port	Sink Port
1	Policy Engine sends the <i>Accept</i> Message to the Sink.	Policy Engine receives the <i>Accept</i> Message and starts the <i>PSTransitionTimer</i> .
2	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink. The Policy Engine waits <i>tSrcTransition</i> before telling the Device Policy Manager to instruct the power supply to modify its output power.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine then evaluates the <i>Accept</i> Message.
3	After <i>tSrcTransition</i> , the Policy Engine tells the Device Policy Manager to instruct the power supply to change its output power capability. The power supply shall be ready to operate at the new power level within <i>tSrcReady</i> (t1). The power supply informs the Device Policy Manager that it is ready to operate at the new power level. The power supply status is passed to the Policy Engine.	
4	The Policy Engine sends the <i>PS_RDY</i> Message to the Sink.	The Policy Engine receives the <i>PS_RDY</i> Message from the Source.
5	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine then evaluates the <i>PS_RDY</i> Message from the Source and tells the Device Policy Manager it is okay to operate at the new power level.
6		The Sink may begin operating at the new power level any time after evaluation of the <i>PS_RDY</i> Message. This time duration is indeterminate.
7		The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. The time duration (t2) depends on the magnitude of the load change.

### 7.3.2 Increasing the Voltage

The interaction of the System Policy, Device Policy, and power supply that shall be followed when increasing the voltage is shown in Figure 7-13. The sequence that shall be followed is described in Table 7-4. The timing parameters that shall be followed are listed in Table 7-22, Table 7-23 and Table 7-24. Note in this figure, the Sink has previously sent a *Request* Message to the Source.

Figure 7-13 Transition Diagram for Increasing the Voltage

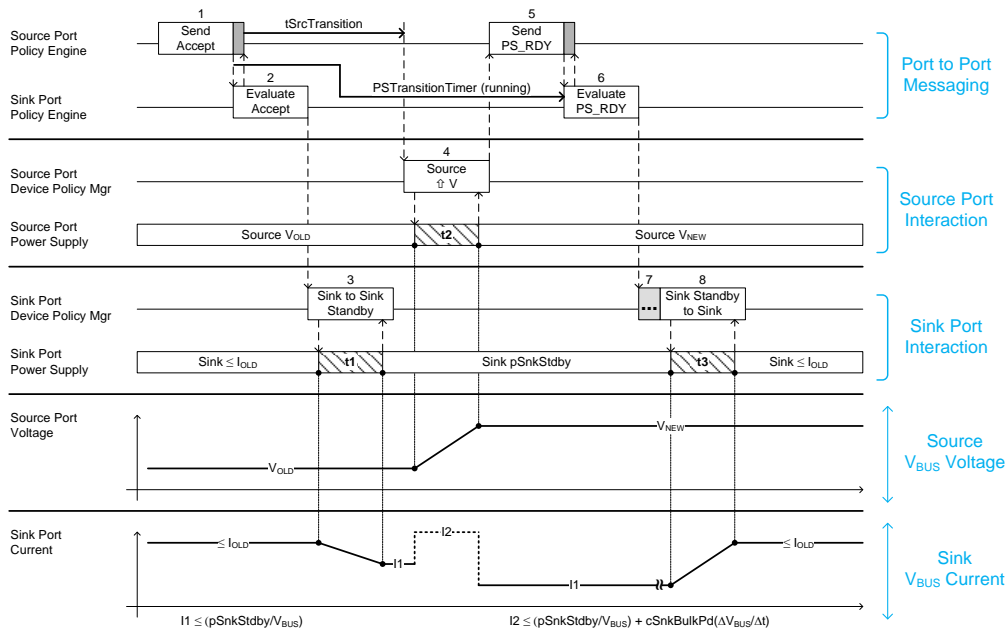


Table 7-4 Sequence Description for Increasing the Voltage

Step	Source Port	Sink Port
1	Policy Engine sends the <i>Accept</i> Message to the Sink.	Policy Engine receives the <i>Accept</i> Message and starts the <i>PSTransitionTimer</i> .
2	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink. The Policy Engine waits <i>tSrcTransition</i> before telling the Device Policy Manager to instruct the power supply to modify its output power.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine. Policy Engine then evaluates the <i>Accept</i> Message.
3		Policy Engine tells the Device Policy Manager to instruct the power supply to reduce power consumption to <i>pSnkStdbby</i> within <i>tSnkStdbby</i> (t1); t1 shall complete before <i>tSrcTransition</i> . The Sink shall not violate transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.
4	After <i>tSrcTransition</i> , the Policy Engine tells the Device Policy Manager to instruct the power supply to change its output voltage to operate at the new power level. The power supply shall be ready to operate at the new power level within <i>tSrcReady</i> (t2). The power supply informs the Device Policy Manager that it is ready to operate at the new power level. The power supply status is passed to the Policy Engine.	
5	The Policy Engine sends the <i>PS_RDY</i> Message to the Sink.	The Policy Engine receives the <i>PS_RDY</i> Message from the Source.
6	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine then evaluates the <i>PS_RDY</i> Message from the Source and tells the Device Policy Manager it is okay to operate at the new power level.
7		The Sink may begin operating at the new power level any time after evaluation of the <i>PS_RDY</i> Message. This time duration is indeterminate.
8		The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. The time duration (t3) depends on the magnitude of the load change.



### 7.3.3 Increasing the Voltage and Current

The interaction of the System Policy, Device Policy, and power supply that shall be followed when increasing the voltage and current is shown in Figure 7-14. The sequence that shall be followed is described in Table 7-5. The timing parameters that shall be followed are listed in Table 7-22, Table 7-23 and Table 7-24. Note in this figure, the Sink has previously sent a *Request* Message to the Source.

Figure 7-14 Transition Diagram for Increasing the Voltage and Current

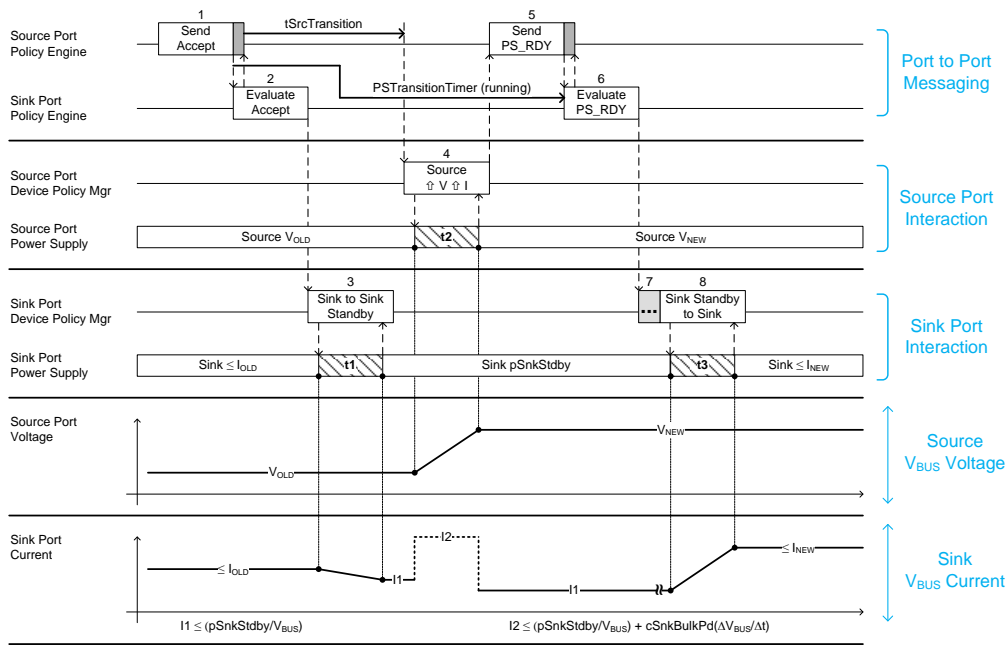


Table 7-5 Sequence Diagram for Increasing the Voltage and Current

Step	Source Port	Sink Port
1	Policy Engine sends the <i>Accept</i> Message to the Sink.	Policy Engine receives the <i>Accept</i> Message and starts the <i>PSTransitionTimer</i> .
2	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink. The Policy Engine waits <i>tSrcTransition</i> before telling the Device Policy Manager to instruct the power supply to modify its output power.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine then evaluates the <i>Accept</i> Message.
3		Policy Engine tells the Device Policy Manager to instruct the power supply to reduce power consumption to <i>pSnkStdbby</i> within <i>tSnkStdbby</i> ( $t_1$ ); $t_1$ shall complete before <i>tSrcTransition</i> . The Sink shall not violate transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.
4	After <i>tSrcTransition</i> , the Policy Engine tells the Device Policy Manager to instruct the power supply to change its output voltage to operate at the new power level. The power supply shall be ready to operate at the new power level within <i>tSrcReady</i> ( $t_2$ ). The power supply informs the Device Policy Manager that it is ready to operate at the new power level. The power supply status is passed to the Policy Engine.	
5	The Policy Engine sends the <i>PS_RDY</i> Message to the Sink.	The Policy Engine receives the <i>PS_RDY</i> Message from the Source.
6	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine then evaluates the <i>PS_RDY</i> Message from the Source and tells the Device Policy Manager it is okay to operate at the new power level.
7		The Sink may begin operating at the new power level any time after evaluation of the <i>PS_RDY</i> Message. This time duration is indeterminate.
8		The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. The time duration ( $t_3$ ) depends on the magnitude of the load change.

### 7.3.4 Increasing the Voltage and Decreasing the Current

The interaction of the System Policy, Device Policy, and power supply that shall be followed when increasing the voltage and decreasing the current is shown in Figure 7-15. The sequence that shall be followed is described in Table 7-6. The timing parameters that shall be followed are listed in Table 7-22, Table 7-23 and Table 7-24. Note in this figure, the Sink has previously sent a *Request* Message to the Source.

Figure 7-15 Transition Diagram for Increasing the Voltage and Decreasing the Current

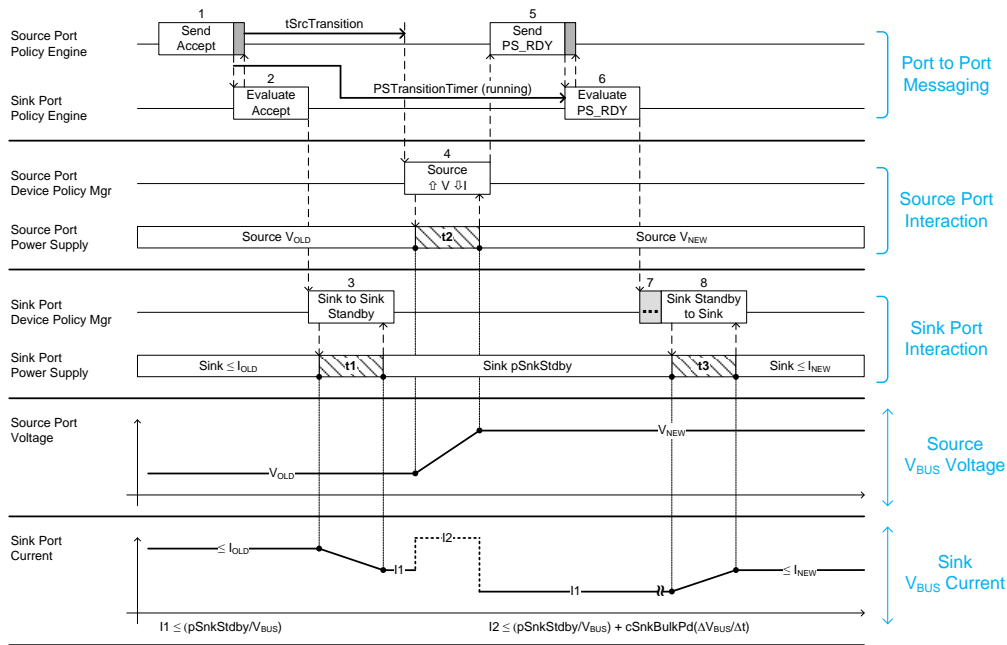


Table 7-6 Sequence Description for Increasing the Voltage and Decreasing the Current

Step	Source Port	Sink Port
1	Policy Engine sends the <i>Accept</i> Message to the Sink.	Policy Engine evaluates the <i>Accept</i> Message and starts the <i>PSTransitionTimer</i> .
2	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink. The Policy Engine waits <i>tSrcTransition</i> before telling the Device Policy Manager to instruct the power supply to modify its output power.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine then evaluates the <i>Accept</i> Message.
3		Policy Engine tells the Device Policy Manager to instruct the power supply to reduce power consumption to <i>pSnkStdbby</i> within <i>tSnkStdbby</i> ( $t_1$ ); $t_1$ shall complete before <i>tSrcTransition</i> . The Sink shall not violate transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.
4	After <i>tSrcTransition</i> , the Policy Engine tells the Device Policy Manager to instruct the power supply to change its output voltage to operate at the new power level. The power supply shall be ready to operate at the new power level within <i>tSrcReady</i> ( $t_2$ ). The power supply informs the Device Policy Manager that it is ready to operate at the new power level. The power supply status is passed to the Policy Engine.	
5	The Policy Engine sends the <i>PS_RDY</i> Message to the Sink.	The Policy Engine receives the <i>PS_RDY</i> Message from the Source.
6	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine then evaluates the <i>PS_RDY</i> Message from the Source and tells the Device Policy Manager it is okay to operate at the new power level.
7		The Sink may begin operating at the new power level any time after evaluation of the <i>PS_RDY</i> Message. This time duration is indeterminate.
8		The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. The time duration ( $t_3$ ) depends on the magnitude of the load change.

### 7.3.5 Decreasing the Voltage and Increasing the Current

The interaction of the System Policy, Device Policy, and power supply that shall be followed when decreasing the voltage and increasing the current is shown in Figure 7-16. The sequence that shall be followed is described in Table 7-7. The timing parameters that shall be followed are listed in Table 7-22, Table 7-23 and Table 7-24. Note in this figure, the Sink has previously sent a *Request* Message to the Source.

Figure 7-16 Transition Diagram for Decreasing the Voltage and Increasing the Current

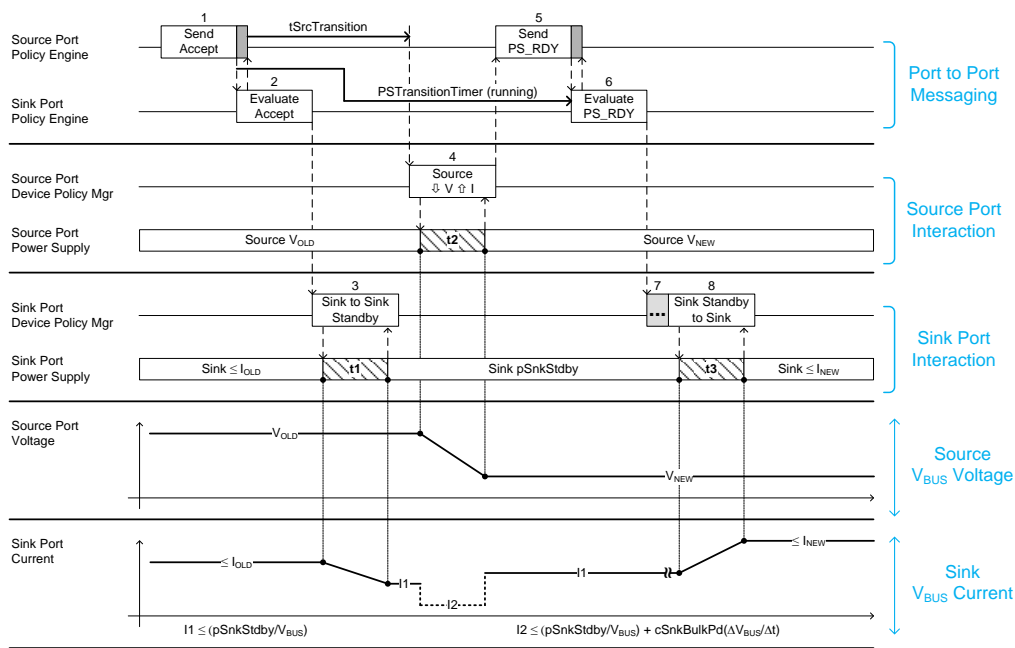


Table 7-7 Sequence Description for Decreasing the Voltage and Increasing the Current

Step	Source Port	Sink Port
1	Policy Engine sends the <i>Accept</i> Message to the Sink.	Policy Engine receives the <i>Accept</i> Message and starts the <i>PSTransitionTimer</i> .
2	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink. The Policy Engine waits <i>tSrcTransition</i> before telling the Device Policy Manager to instruct the power supply to modify its output power.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine then evaluates the <i>Accept</i> Message.
3		Policy Engine tells the Device Policy Manager to instruct the power supply to reduce power consumption to <i>pSnkStdbby</i> within <i>tSnkStdbby</i> ( $t_1$ ); $t_1$ shall complete before <i>tSrcTransition</i> . The Sink shall not violate transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.
4	After <i>tSrcTransition</i> , the Policy Engine tells the Device Policy Manager to instruct the power supply to change its output voltage to operate at the new power level. The power supply shall be ready to operate at the new power level within <i>tSrcReady</i> ( $t_2$ ). The power supply informs the Device Policy Manager that it is ready to operate at the new power level. The power supply status is passed to the Policy Engine.	
5	The Policy Engine sends the <i>PS_RDY</i> Message to the Sink.	The Policy Engine receives the <i>PS_RDY</i> Message from the Source.
6	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine then evaluates the <i>PS_RDY</i> Message from the Source and tells the Device Policy Manager it is okay to operate at the new power level.
7		The Sink may begin operating at the new power level any time after evaluation of the <i>PS_RDY</i> Message. This time duration is indeterminate.
8		The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. The time duration ( $t_3$ ) depends on the magnitude of the load change.

### 7.3.6 Decreasing the Current

The interaction of the System Policy, Device Policy, and power supply that shall be followed when decreasing the current is shown in Figure 7-17. The sequence that shall be followed is described in Table 7-8. The timing parameters that shall be followed are listed in Table 7-22, Table 7-23 and Table 7-24. Note in this figure, the Sink has previously sent a *Request* Message to the Source.

Figure 7-17 Transition Diagram for Decreasing the Current

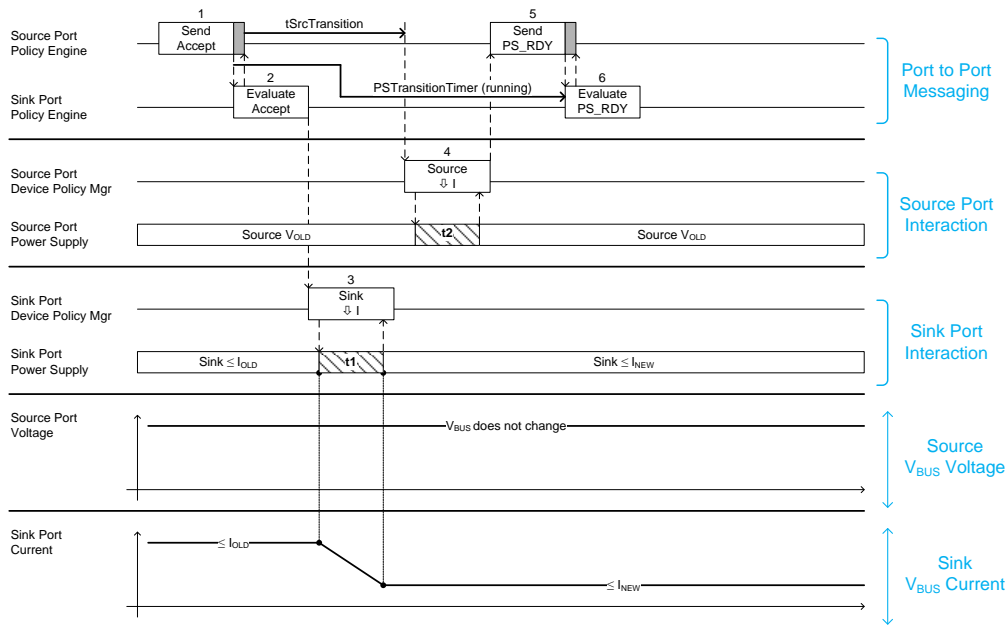


Table 7-8 Sequence Description for Decreasing the Current

Step	Source Port	Sink Port
1	Policy Engine sends the <i>Accept</i> Message to the Sink.	Policy Engine receives the <i>Accept</i> Message starts <i>PSTransitionTimer</i> .
2	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink. The Policy Engine waits <i>tSrcTransition</i> before telling the Device Policy Manager to instruct the power supply to modify its output power.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine then evaluates the <i>Accept</i> Message. Policy Engine tells the Device Policy Manager to instruct the power supply to reduce power consumption.
3		The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. The Sink shall be able to operate with lower current within <i>tSnkNewPower</i> (t1) ; t1 shall complete before <i>tSrcTransition</i> .
4	After <i>tSrcTransition</i> , the Policy Engine tells the Device Policy Manager to instruct the power supply to change its output power capability. The power supply shall be ready to operate at the new power level within <i>tSrcReady</i> (t2). The power supply informs the Device Policy Manager that it is ready to operate at the new power level. The power supply status is passed to the Policy Engine.	
5	The Policy Engine sends the <i>PS_RDY</i> Message to the Sink.	The Policy Engine receives the <i>PS_RDY</i> Message from the Source.
6	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine evaluates the <i>PS_RDY</i> Message from the Source. The Sink is already operating at the new power level so no further action is required.



### 7.3.7 Decreasing the Voltage

The interaction of the System Policy, Device Policy, and power supply that shall be followed when decreasing the voltage is shown in Figure 7-18. The sequence that shall be followed is described in Table 7-9. The timing parameters that shall be followed are listed in Table 7-22, Table 7-23 and Table 7-24. Note in this figure, the Sink has previously sent a *Request* Message to the Source.

Figure 7-18 Transition Diagram for Decreasing the Voltage

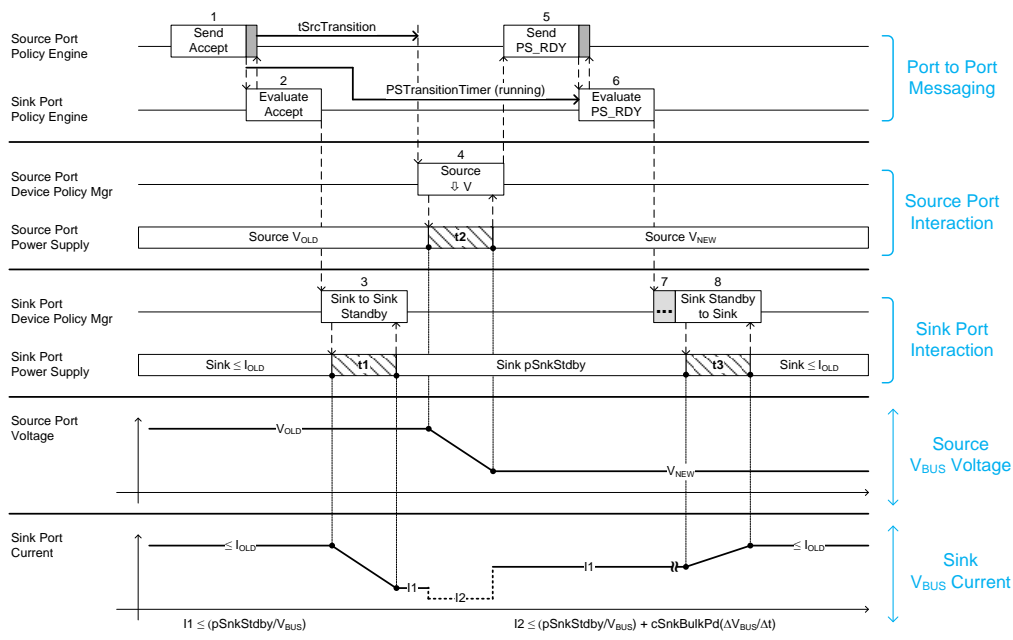


Table 7-9 Sequence Description for Decreasing the Voltage

Step	Source Port	Sink Port
1	Policy Engine sends the <i>Accept</i> Message to the Sink.	Policy Engine receives the <i>Accept</i> Message and starts the <i>PSTransitionTimer</i> .
2	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink. The Policy Engine waits <i>tSrcTransition</i> before telling the Device Policy Manager to instruct the power supply to modify its output power.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine then evaluates the <i>Accept</i> Message.
3		Policy Engine tells the Device Policy Manager to instruct the power supply to reduce power consumption to <i>pSnkStdbby</i> within <i>tSnkStdbby</i> ( $t_1$ ); $t_1$ shall complete before <i>tSrcTransition</i> . The Sink shall not violate transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.
4	After <i>tSrcTransition</i> , the Policy Engine tells the Device Policy Manager to instruct the power supply to change its output voltage to operate at the new power level. The power supply shall be ready to operate at the new power level within <i>tSrcReady</i> ( $t_2$ ). The power supply informs the Device Policy Manager that it is ready to operate at the new power level. The power supply status is passed to the Policy Engine.	
5	The Policy Engine sends the <i>PS_RDY</i> Message to the Sink.	The Policy Engine receives the <i>PS_RDY</i> Message from the Source.
6	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine then evaluates the <i>PS_RDY</i> Message from the Source and tells the Device Policy Manager it is okay to operate at the new power level.
7		The Sink may begin operating at the new power level any time after evaluation of the <i>PS_RDY</i> Message. This time duration is indeterminate.
8		The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. The time duration ( $t_3$ ) depends on the magnitude of the load change.

### 7.3.8 Decreasing the Voltage and the Current

The interaction of the System Policy, Device Policy, and power supply that shall be followed when decreasing the voltage and current is shown in Figure 7-19. The sequence that shall be followed is described in Table 7-10. The timing parameters that shall be followed are listed in Table 7-22, Table 7-23 and Table 7-24. Note in this figure, the Sink has previously sent a *Request* Message to the Source.

Figure 7-19 Transition Diagram for Decreasing the Voltage and the Current

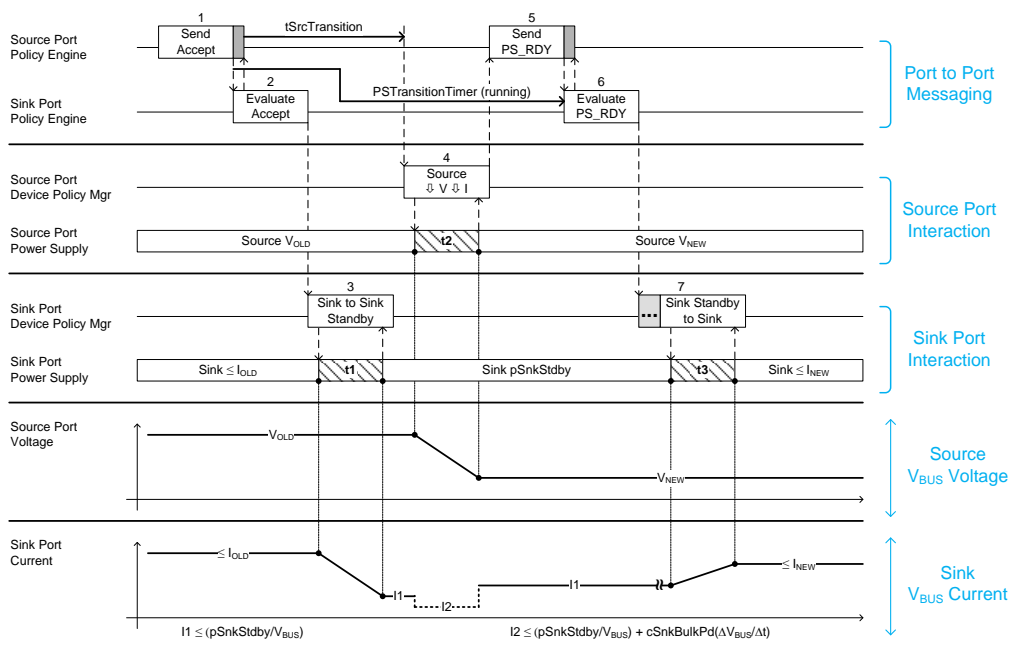


Table 7-10 Sequence Description for Decreasing the Voltage and the Current

Step	Source Port	Sink Port
1	Policy Engine sends the <i>Accept</i> Message to the Sink.	Policy Engine receives the <i>Accept</i> Message and starts the <i>PSTransitionTimer</i> .
2	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink. The Policy Engine waits <i>tSrcTransition</i> before telling the Device Policy Manager to instruct the power supply to modify its output power.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine then evaluates the <i>Accept</i> Message.
3		Policy Engine tells the Device Policy Manager to instruct the power supply to reduce power consumption to <i>pSnkStdby</i> within <i>tSnkStdby</i> ( $t_1$ ); $t_1$ shall complete before <i>tSrcTransition</i> . The Sink shall not violate transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.
4	After <i>tSrcTransition</i> , the Policy Engine tells the Device Policy Manager to instruct the power supply to change its output voltage to operate at the new power level. The power supply shall be ready to operate at the new power level within <i>tSrcReady</i> ( $t_2$ ). The power supply informs the Device Policy Manager that it is ready to operate at the new power level. The power supply status is passed to the Policy Engine.	
5	The Policy Engine sends the <i>PS_RDY</i> Message to the Sink.	The Policy Engine receives the <i>PS_RDY</i> Message from the Source.
6	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine then evaluates the <i>PS_RDY</i> Message from the Source and tells the Device Policy Manager it is okay to operate at the new power level.
7		The Sink may begin operating at the new power level any time after evaluation of the <i>PS_RDY</i> Message. This time duration is indeterminate.
8		The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. The time duration ( $t_3$ ) depends on the magnitude of the load change.

### 7.3.9 Sink Requested Power Role Swap

The interaction of the System Policy, Device Policy, and power supply that shall be followed during a Sink requested Power Role Swap is shown in Figure 7-20. The sequence that shall be followed is described in Table 7-11. The timing parameters that shall be followed are listed in Table 7-23. Note in this figure, the Sink has previously sent a *PR\_Swap* Message to the Source.

Figure 7-20 Transition Diagram for a Sink Requested Power Role Swap

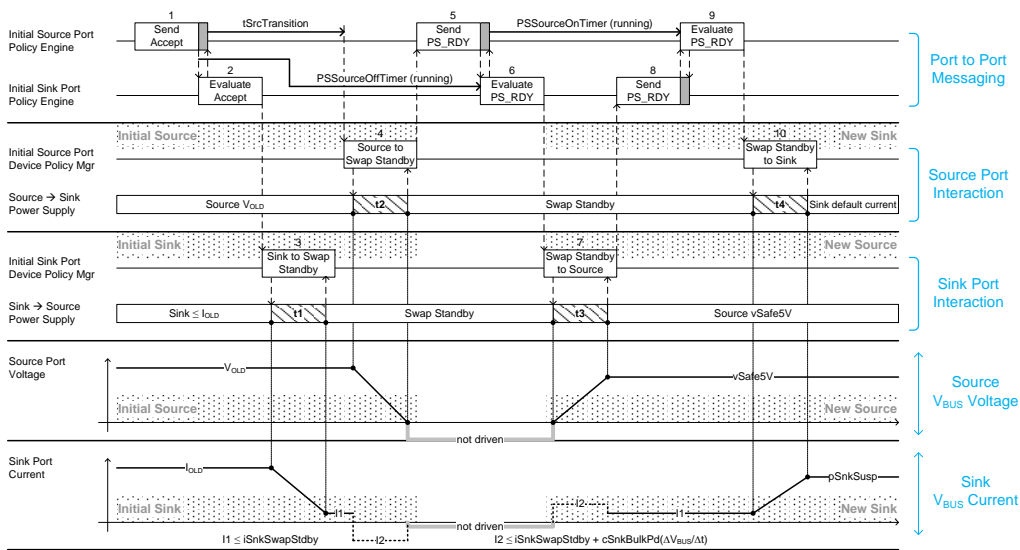


Table 7-11 Sequence Description for a Sink Requested Power Role Swap

Step	Initial Source Port → New Sink Port	Initial Sink Port → New Source Port
1	Policy Engine sends the <i>Accept</i> Message to the Initial Sink.	Policy Engine receives the <i>Accept</i> and starts the <i>PSSourceOffTimer</i> .
2	Protocol Layer receives the <i>GoodCRC</i> Message from the Initial Sink. The Policy Engine waits <i>tSrcTransition</i> before telling the Device Policy Manager to instruct the power supply to transition to Swap Standby.	Protocol Layer sends the <i>GoodCRC</i> Message to the Initial Source. Policy Engine then evaluates the <i>Accept</i> Message.
3		Policy Engine tells the Device Policy Manager to instruct the power supply to transition to Swap Standby within <i>tSnkStdbby</i> (t1); t1 shall complete before <i>tSrcTransition</i> . When in Sink Standby the Initial Sink shall not draw more than <i>iSnkSwapStdbby</i> (I1). The Sink shall not violate transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.
4	After <i>tSrcTransition</i> , the Policy Engine tells the Device Policy Manager to instruct the power supply to transition to Swap Standby (see Section 7.1.11). The power supply shall complete the transition to Swap Standby within <i>tSrcSwapStdbby</i> (t2). The power supply informs the Device Policy Manager that it is ready to operate as the new Sink. The power supply status is passed to the Policy Engine.	
5	The power supply is ready and the Policy Engine sends the <i>PS_RDY</i> Message to the device that will become the new Source.	
6	Protocol Layer receives the <i>GoodCRC</i> Message from the device that will become the new Source. Policy Engine starts the <i>PSSourceOnTimer</i> . Upon sending the <i>PS_RDY</i> Message and receiving the <i>GoodCRC</i> Message the Initial Source is ready to be the new Sink.	Policy Engine stops the <i>PSSourceOffTimer</i> . The Protocol Layer sends the <i>GoodCRC</i> Message to the new Sink. Policy Engine tells the Device Policy to instruct the power supply to operate as the new Source.
7		The power supply as the new Source transitions from Swap Standby to sourcing default <i>vSafe5V</i> within <i>tNewSrc</i> (t3). The power supply informs the Device Policy Manager that it is operating as the new Source.
8	Policy Engine receives the <i>PS_RDY</i> Message from the Source.	Device Policy Manager informs the Policy Engine the power supply is ready and the Policy Engine sends the <i>PS_RDY</i> Message to the new Sink.
9	Policy Engine stops the <i>PSSourceOnTimer</i> . Protocol Layer sends the <i>GoodCRC</i> Message to the new Source. Policy Engine evaluates the <i>PS_RDY</i> Message from the new Source and tells the Device Policy Manager to instruct the power supply to draw current as the new Sink.	Protocol Layer receives the <i>GoodCRC</i> Message from the new Sink.

Step	Initial Source Port → New Sink Port	Initial Sink Port → New Source Port
10	<p>The power supply as the new Sink transitions from Swap Standby to drawing <i>pSnkSusp</i> within <i>tNewSnk</i> (<i>t4</i>). The power supply informs the Device Policy Manager that it is operating as the new Sink. At this point subsequent negotiations between the new Source and the new Sink may proceed as normal. The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. The time duration (<i>t4</i>) depends on the magnitude of the load change.</p>	

### 7.3.10 Source Requested Power Role Swap

The interaction of the System Policy, Device Policy, and power supply that shall be followed during a Source requested Power Role Swap is shown in Figure 7-21. The sequence that shall be followed is described in Table 7-12. The timing parameters that shall be followed are listed in Table 7-22. Note in this figure, the Sink has previously sent a *PR\_Swap* Message to the Source.

Figure 7-21 Transition Diagram for a Source Requested Power Role Swap

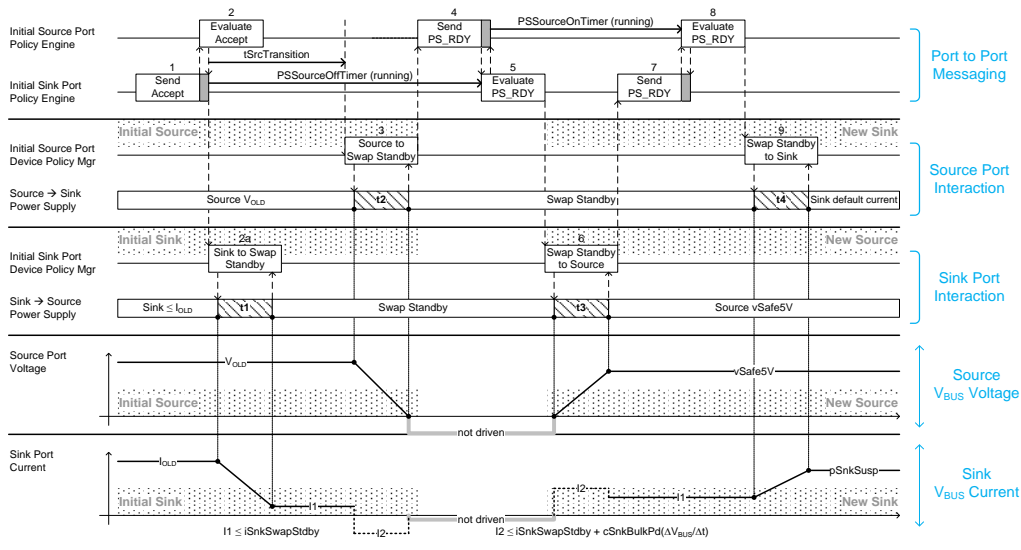




Table 7-12 Sequence Description for a Source Requested Power Role Swap

Step	Initial Source Port → New Sink Port	Initial Sink Port → New Source Port
1	Policy Engine receives the <i>Accept</i> Message.	Policy Engine sends the <i>Accept</i> Message to the Initial Source.
2	Protocol Layer sends the <i>GoodCRC</i> Message to the Initial Sink. Policy Engine evaluates the <i>Accept</i> Message, and then waits <i>tSrcTransition</i> before telling the Device Policy Manager to instruct the power supply to transition to Swap Standby.	Protocol Layer receives the <i>GoodCRC</i> Message from the Initial Source. Policy Engine starts the <i>PSSourceOffTimer</i> .
2a		The Policy Engine tells the Device Policy Manager to instruct the power supply to transition to Swap Standby. The power supply shall complete the transition to Swap Standby within <i>tSnkStdbby</i> (t1) ; t1 shall complete before <i>tSrcTransition</i> . The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. Policy Engine starts <i>PSSourceOffTimer</i> . When in Sink Standby the Initial Sink shall not draw more than <i>iSnkSwapStdbby</i> (I1).
3	After <i>tSrcTransition</i> , the Policy Engine tells the Device Policy Manager to instruct the power supply to transition to Swap Standby (see Section 7.1.11). The power supply shall complete the transition to Swap Standby within <i>tSrcSwapStdbby</i> (t2). The power supply informs the Device Policy Manager that it is ready to operate as the new Sink. The power supply status is passed to the Policy Engine.	
4	The Policy Engine sends the <i>PS_RDY</i> Message to the soon to be new Source.	Policy Engine receives the <i>PS_RDY</i> Message and stops the <i>PSSourceOffTimer</i> .
5	Protocol Layer receives the <i>GoodCRC</i> Message from the soon to be new Source. Policy Engine starts the <i>PSSourceOnTimer</i> . At this point the Initial Source is ready to be the new Sink.	Protocol Layer sends the <i>GoodCRC</i> Message to the new Sink. Upon evaluating the <i>PS_RDY</i> Message the Initial Sink is ready to operate as the new Source. Policy Engine tells the Device Policy to instruct the power supply to operate as the new Source.
6		The power supply as the new Source transitions from Swap Standby to sourcing default <i>vSafe5V</i> within <i>tNewSrc</i> (t3). The power supply informs the Device Policy Manager that it is operating as the new Source.
7	Policy Engine receives the <i>PS_RDY</i> Message and stops the <i>PSSourceOnTimer</i> .	Device Policy Manager informs the Policy Engine the power supply is ready and the Policy Engine sends the <i>PS_RDY</i> Message to the new Sink.
8	Protocol Layer sends the <i>GoodCRC</i> Message to the new Source. Policy Engine evaluates the <i>PS_RDY</i> Message from the new Source and tells the Device Policy Manager to instruct the power supply to draw current as the new Sink.	Protocol Layer receives the <i>GoodCRC</i> Message from the new Sink.

Step	Initial Source Port → New Sink Port	Initial Sink Port → New Source Port
9	<p>The power supply as the new Sink transitions from Swap Standby to drawing <i>pSnkSusp</i> within <i>tNewSnk</i> (<math>t_4</math>). The power supply informs the Device Policy Manager that it is operating as the new Sink. At this point subsequent negotiations between the new Source and the new Sink may proceed as normal. The new Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. The time duration (<math>t_4</math>) depends on the magnitude of the load change.</p>	

### 7.3.11 GotoMin Current Decrease

The interaction of the System Policy, Device Policy, and power supply that shall be followed during a GotoMin current decrease is shown in Figure 7-22. The sequence that shall be followed is described in Table 7-13. The timing parameters that shall be followed are listed in Table 7-22 and Table 7-13.

Figure 7-22 Transition Diagram for a GotoMin Current Decrease

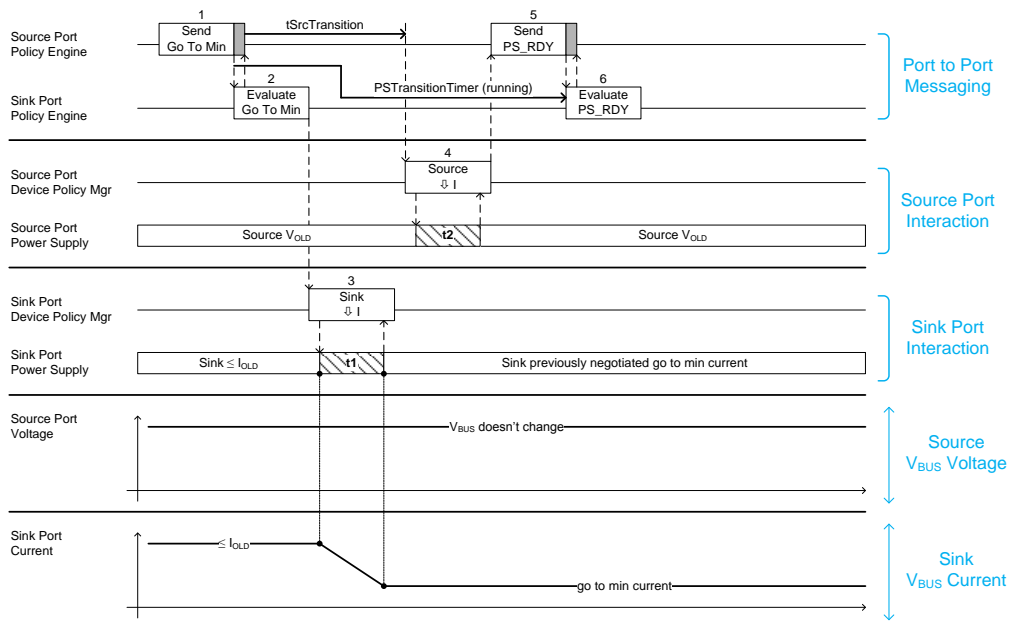


Table 7-13 Sequence Description for a GotoMin Current Decrease

Step	Source Port	Sink Port
1	Policy Engine sends the <i>GotoMin</i> Message to the Sink.	Policy Engine receives the <i>GotoMin</i> Message and starts the <i>PSTransitionTimer</i> .
2	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink. The Policy Engine waits <i>tSrcTransition</i> before telling the Device Policy Manager to instruct the power supply to modify its output power.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine then evaluates the <i>GotoMin</i> Message.
3		Policy Engine tells the Device Policy Manager to instruct the power supply to reduce power consumption, within <i>tSnkNewPower</i> (t1), to the pre-negotiated go to reduced power level ); t1 shall complete before <i>tSrcTransition</i> . The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.
4	After <i>tSrcTransition</i> , the Policy Engine tells the Device Policy Manager to instruct the power supply to change its output power capability. The power supply shall be ready to operate at the new power level within <i>tSrcReady</i> (t2). The power supply informs the Device Policy Manager that it is ready to operate at the new power level. The power supply status is passed to the Policy Engine.	
5	The Policy Engine sends the <i>PS_RDY</i> Message to the Sink.	The Policy Engine receives the <i>PS_RDY</i> Message.
6	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine evaluates the <i>PS_RDY</i> Message from the Source and no further action is required.

### 7.3.12 Source Initiated Hard Reset

The interaction of the System Policy, Device Policy, and power supply that shall be followed during a Source Initiated Hard Reset is shown in Figure 7-23. The sequence that shall be followed is described in Table 7-14. The timing parameters that shall be applied are listed in Table 7-22 and Table 7-23.

Figure 7-23 Transition Diagram for a Source Initiated Hard Reset

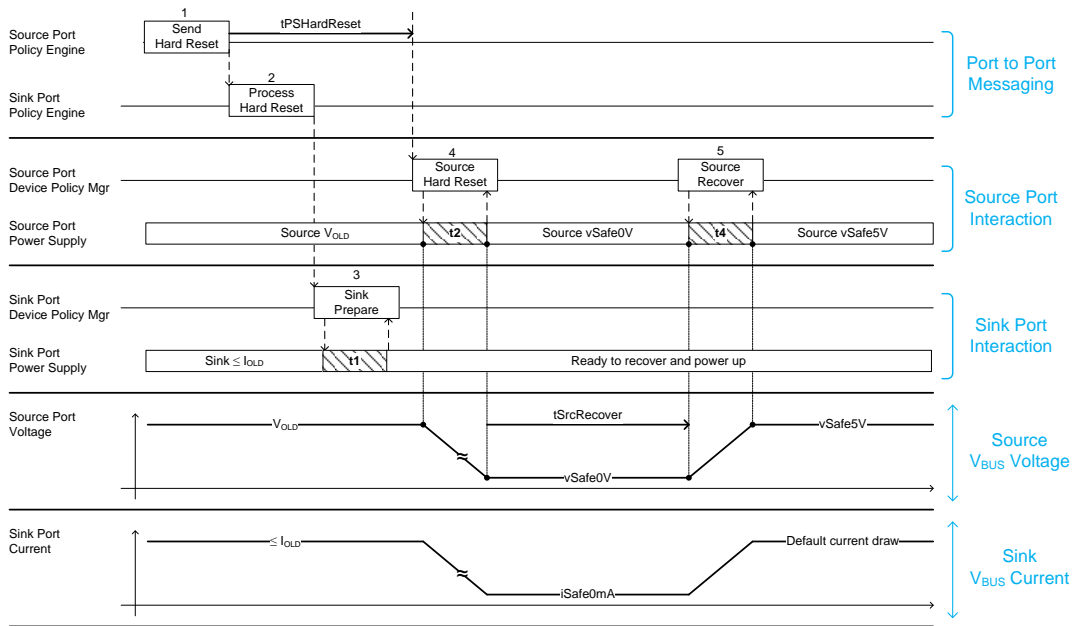


Table 7-14 Sequence Description for a Source Initiated Hard Reset

Step	Source Port	Sink Port
1	Policy Engine sends <i>Hard Reset</i> Signaling to the Sink.	Sink receives <i>Hard Reset</i> Signaling.
<del>2a</del>	<del>Policy Engine tells the Device Policy Manager to instruct the power supply to perform a Hard Reset. The transition to shall occur within (t1).</del>	<del>The Sink shall not draw more than when <math>V_{BUS}</math> is driven to -Policy Engine is informed of the Hard Reset. Policy Engine tells the Device Policy Manager to instruct the power supply to prepare for a Hard Reset.</del>
<del>2b</del>		<del>Policy Engine is informed ofThe Sink prepares for the Hard Reset within <math>t_{SnkHardResetPrepare}</math>. Policy Engine tells (t1) and passes an indication to the Device Policy Manger The Sink shall not draw more than <math>i_{Safe0mA}</math> Manager to instruct the power supply when <math>V_{BUS}</math> is driven to <math>v_{Safe0V}</math> prepare to recover from a Hard Reset.</del>
<del>3</del>	<del>Policy Engine waits <math>t_{PSHardReset}</math> after sending <i>Hard Reset</i> Signaling and then tells the Device Policy Manager to instruct the power supply to perform a Hard Reset. The transition to <math>v_{Safe0V}</math> shall occur within <math>t_{Safe0V}</math> (t2).</del>	<del>The power supply shall be ready to resume normal operation within (t2) and passes an indication to the Device Policy Manager.</del>
45	After $t_{SrcRecover}$ the Source applies power to $V_{BUS}$ in an attempt to re-establish communication with the Sink and resume USB Default Operation. The transition to $v_{Safe5V}$ shall occur within $t_{SrcTurnOn}$ (t3t4).	The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.

### 7.3.13 Sink Initiated Hard Reset

The interaction of the System Policy, Device Policy, and power supply that shall be followed during a Sink Initiated Hard Reset is shown in Figure 7-24. The sequence that shall be followed is described in Table 7-15. The timing parameters that shall be followed are listed in Table 7-22 and Table 7-23.

Figure 7-24 Transition Diagram for a Sink Initiated Hard Reset

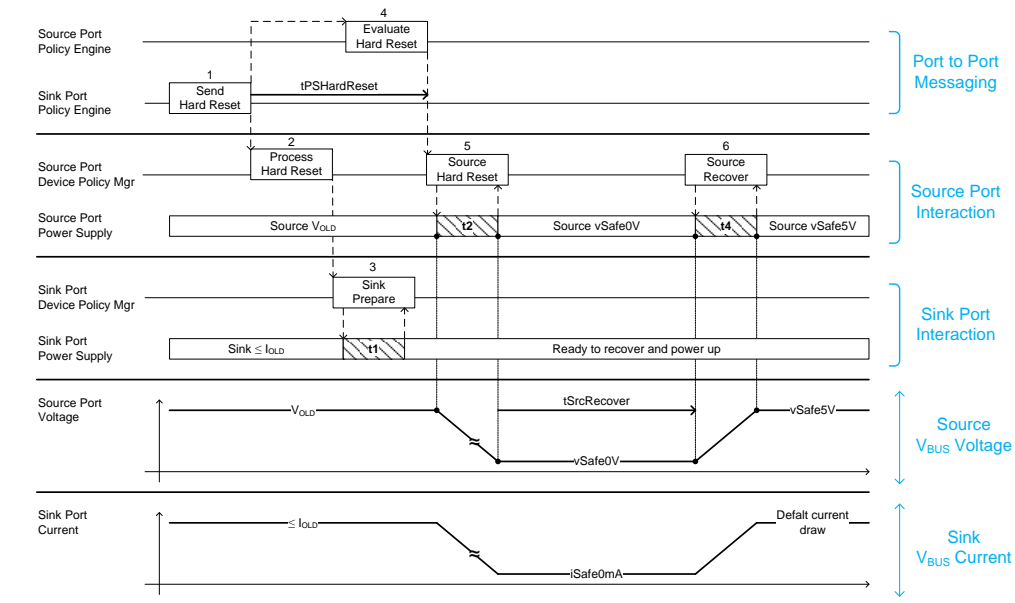


Table 7-15 Sequence Description for a Sink Initiated Hard Reset

Step	Source Port	Sink Port
1		Policy Engine sends <i>Hard Reset</i> Signaling to the Source.
<del>2</del>		<del>Policy Engine tells the Device Policy Manager to instruct the power supply to prepare for a Hard Reset.</del>
<del>2a3</del>		<del>Policy Engine tells the Sink prepares for the Device Policy Manager to instruct the power supply to prepare to recover from a Hard Reset. If action is taken the Sink shall be prepared within <math>t_{SnkHardResetPrepare}</math> (t1) and passes an indication to the Device Policy Manger. The Sink shall not draw more than <math>i_{Safe0mA}</math> when <math>V_{BUS}</math> is driven to <math>v_{Safe0V}</math>.</del>
<del>2b4</del>	Policy Engine is informed of the Hard Reset.	
<del>35</del>	Policy Engine <del>waits <math>t_{PSHardReset}</math> after receiving <i>Hard Reset</i> Signaling and then</del> tells the Device Policy Manager to instruct the power supply to <del>transition to perform a Hard Reset.</del> The transition <del>to <math>v_{Safe0V}</math></del> shall occur within <del><math>t_{Safe0V}</math></del> (t2).	<del>The Sink shall not draw more than <math>i_{Safe0mA}</math> when <math>V_{BUS}</math> is driven to <math>v_{Safe0V}</math>.</del>
46	After $t_{SrcRecover}$ the Source applies power to <del>the Sink</del> $V_{BUS}$ in an attempt to re-establish communication <del>with the Sink</del> and resume USB Default Operation. The transition to $v_{Safe5V}$ shall occur within <del><math>t_{SrcTurnOn}</math></del> ( <del>t3</del> t4).	The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.



New

### 7.3.14 Type-A/B Hard Reset after a Power Role Swap

The following sections indicate the transitions made when a Hard Reset occurs between Type-A Port acting as Sink and a Type-B Port acting as Source.

#### 7.3.13-17.3.14.1 Type-B New Source Initiates Hard Reset and Type-A New Sink Receives Hard Reset Signaling

The interaction of the System Policy, Device Policy, and power supply that shall be followed during a Hard Reset initiated by the Type-B new Source when the Type-A new Sink receives *Hard Reset* Signaling is shown in Figure 7-25. The sequence that shall be followed is described in Table 7-16. The timing parameters that shall be followed are listed in Table 7-22 and Table 7-23.

The default roles for Provider/Consumers and Consumer/Providers after a Hard Reset are as defined in Section 6.7.2. After a Hard Reset, a Provider/Consumer shall return to its Source role and a Consumer/Provider shall return to its Sink role.

Figure 7-25 Transition Diagram for Type-B New Source Initiated Hard Reset and Type-A New Sink Receives Hard Reset Signaling

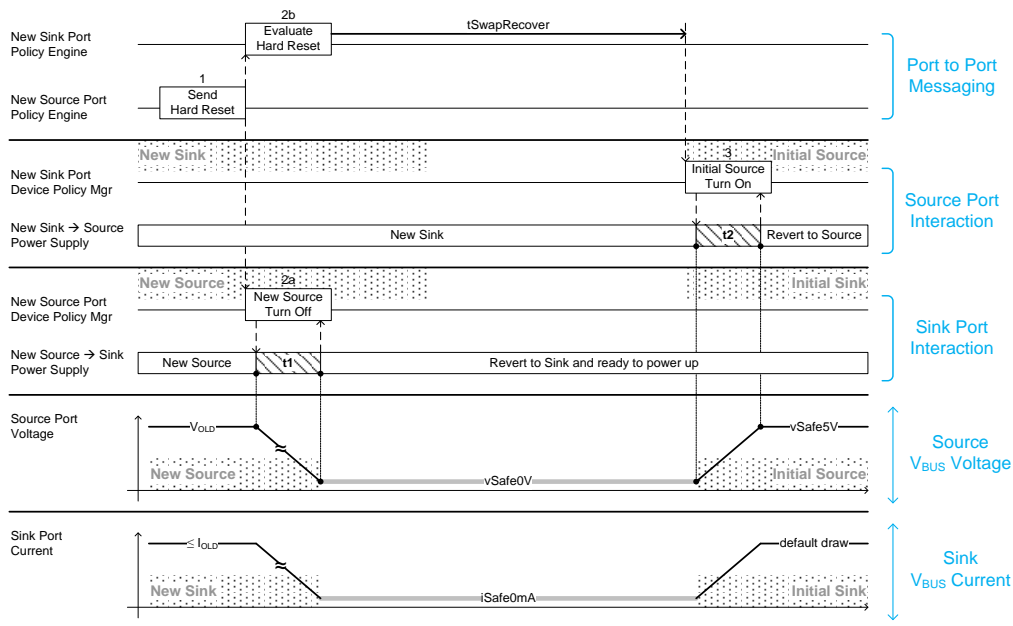


Table 7-16 Sequence Description for **Type-B** New Source Initiated Hard Reset and **Type-A** New Sink Receives Hard Reset Signaling

Step	New Source Port	New Sink Port
1	Policy Engine sends <i>Hard Reset</i> Signaling to the new Sink and tells the Device Policy Manager to turn off the new Source.	
2a	The Device Policy Manager instructs the power supply to turn off the Source and revert to Sink operation by <i>tSafe0V</i> (t1).	
2b		Policy Engine receives and evaluates a Hard Reset notification, waits <i>tSwapRecover</i> and then tells the Device Policy Manager to turn on the new Source.
3	Sink is drawing no more than default current as defined in <a href="#">[USB 2.0]</a> , <a href="#">[USB 3.1]</a> , <a href="#">[USBType-C 1.0]</a> or <a href="#">[BC 1.2]</a> .	Source applies <i>vSafe5V</i> when $V_{BUS}$ is within <i>vSafe0V</i> . The transition to <i>vSafe5V</i> shall be completed with <i>tSrcTurnOn</i> (t2). The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.

7.3.13-27.3.14.2 **Type-B** New Source Initiates Hard Reset and **Type-A** New Sink Does Not Receive Hard Reset Signaling

The interaction of the System Policy, Device Policy, and power supply that shall be followed during a Hard Reset initiated by the **Type-B** new Source when the **Type-A** new Sink does not receive the **Hard Reset** Signaling is shown in Figure 7-26. The sequence that shall be followed is described in Table 7-17. The timing parameters that shall be followed are listed in Table 7-22 and Table 7-23.

~~The default roles for Provider/Consumers and Consumer/Providers after a Hard Reset are as defined in Section 6.7.2. After a Hard Reset, a Provider/Consumer shall return to its Source role and a Consumer/Provider shall return to its Sink role.~~

Figure 7-26 Transition Diagram for **Type-B** New Source Initiated Hard Reset and **Type-A** New Sink Does Not Receive Hard Reset Signaling

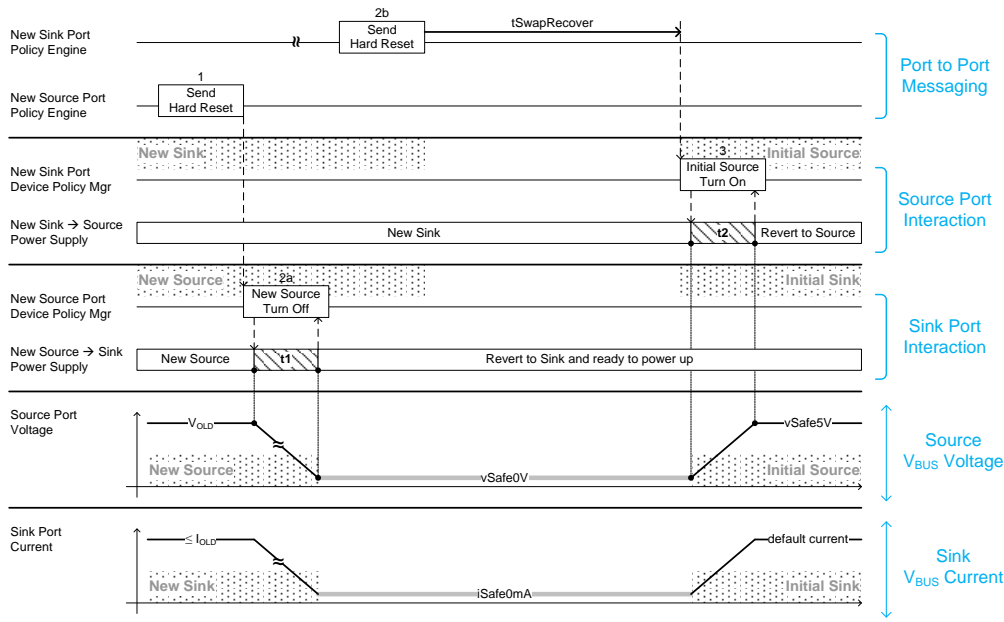


Table 7-17 Sequence Description for **Type-B** New Source Initiated Hard Reset and **Type-A** New Sink Does Not Receive Hard Reset Signaling

Step	New Source Port	New Sink Port
1	Policy Engine attempts to send the <i>Hard Reset</i> Signaling to the new Sink and tells the Device Policy Manager to turn off the new Source.	
2a		The new Sink has not received a Message from the new Source and decides to initiate a Hard Reset , waits <i>tSwapRecover</i> , then tells the Device Policy Manager to turn on the new Source.
2b		The Device Policy Manager instructs the power supply to turn off the Source and revert to Sink operation by <i>tSafe0V</i> (t1).
3	Source shall not re-apply <i>vSafe5V</i> until $V_{BUS}$ is within <i>vSafe0V</i> . The transition to <i>vSafe5V</i> shall be completed within <i>tSrcTurnOn</i> (t2).	Sink is drawing no more than default current as defined in <i>[USB 2.0]</i> , <i>[USB 3.1]</i> , <i>[USBType-C 1.0]</i> or <i>[BC 1.2]</i> . The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.

7.3.13.3 **Type-A New Sink Initiates Hard Reset and Type-B New Source Receives Hard Reset Signaling**

The interaction of the System Policy, Device Policy, and power supply that shall be followed during a Hard Reset initiated by the **Type-A** new Sink when the **Type-B** new Source receives the **Hard Reset** Signaling is shown in Figure 7-27. The sequence that shall be followed is described in Table 7-18. The timing parameters that shall be followed are listed in Table 7-22 and Table 7-23.

The default roles for Provider/Consumers and Consumer/Providers after a Hard Reset are as defined in Section 6.7.2a. Provider/Consumer shall return to its Source role and a Consumer/Provider shall return to its Sink role.

Figure 7-27 Transition Diagram for **Type-A** New Sink Initiated Hard Reset and **Type-B** New Source Receives Hard Reset Signaling

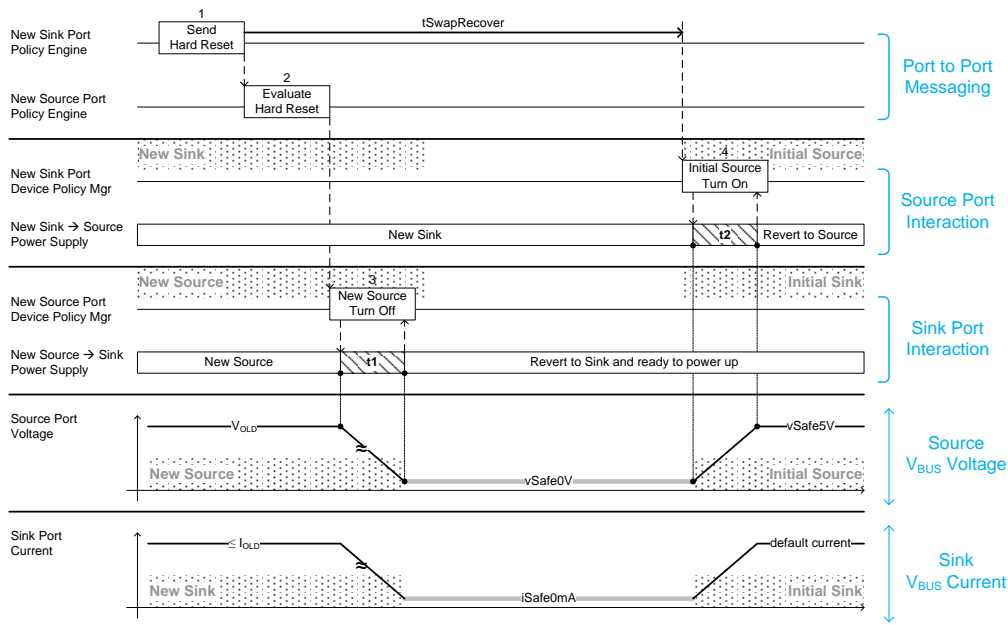


Table 7-18 Sequence Description for **Type-A** New Sink Initiated Hard Reset and **Type-B** New Source Receives Hard Reset Signaling

Step	New Source Port	New Sink Port
1		Policy Engine sends the <i>Hard Reset</i> Signaling to the new Source and waits <i>tSwapRecover</i> before telling the Device Policy Manger to revert to Source operation.
2	Policy Engine evaluates the Hard Reset notification and tells the Device Policy Manager to turn off the new Source.	
3	The Device Policy Manager instructs the power supply to turn off the Source and revert to Sink operation by <i>tSafe0V</i> (t1).	
4	The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.	Source applies <i>vSafe5V</i> when $V_{BUS}$ is within <i>vSafe0V</i> . The transition to <i>vSafe5V</i> shall be completed with <i>tSrcTurnOn</i> (t2).

7.3.13-47.3.14.4 **Type-A New Sink Initiates Hard Reset and Type-B New Source Does Not Receive Hard Reset Signaling**

The interaction of the System Policy, Device Policy, and power supply that shall be followed during a Hard Reset initiated by the **Type-A** new Sink when the **Type-B** new Source does not receive the **Hard Reset** Signaling is shown in Figure 7-28. The sequence that shall be followed is described in Table 7-19. The timing parameters that shall be followed are listed in Table 7-22 and Table 7-23.

The default roles for Provider/Consumers and Consumer/Providers after a Hard Reset are as defined in Section 6.7.2. After a Hard Reset, a Provider/Consumer shall return to its Source role and a Consumer/Provider shall return to its Sink role.

Figure 7-28 Transition Diagram for **Type-A** New Sink Initiated Hard Reset and **Type-B** New Source Does Not Receive Hard Reset Signaling

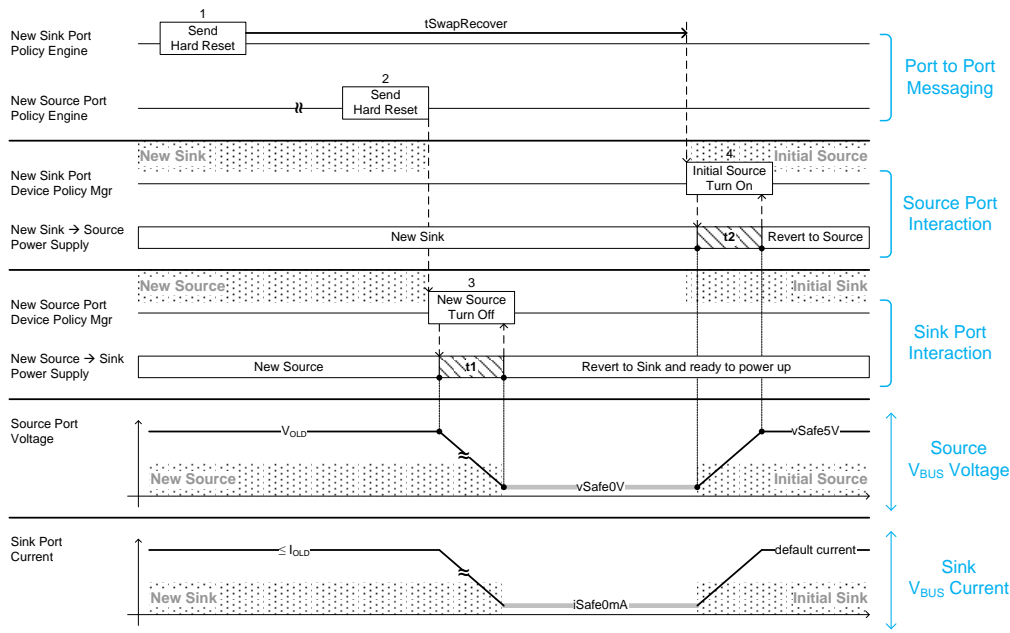




Table 7-19 Sequence Description for **Type-A** New Sink Initiated Hard Reset and **Type-B** New Source Does Not Receive Hard Reset Signaling

Step	New Source Port	New Sink Port
1		Policy Engine attempts to send the <i>Hard Reset</i> Signaling to the new Source and waits <i>tSwapRecover</i> before telling the Device Policy Manger to revert to Source operation.
2	The new Source has not received a Message from the new Sink and decides to initiate a Hard Reset and tells the Device Policy Manager to revert to Source operation.	
3		The Device Policy Manager instructs the Power supply to turn off the Source and revert to Sink operation by <i>tSafe0V</i> (t1).
4	Source shall not re-apply <i>vSafe5V</i> until $V_{BUS}$ is within <i>vSafe0V</i> . The transition to <i>vSafe5V</i> shall be completed within <i>tSrcTurnOn</i> (t2).	The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.

### 7.3.147.3.15 Type-A to Type-B Dead Battery Operation

The interaction of the System Policy, Device Policy, and power supply that shall be followed during Type-A to Type-B Dead Battery operation is shown in Figure 7-29. The sequence that shall be followed is described in Table 7-20. The timing parameters that shall be followed are listed in Table 4-4, Table 7-22 and Table 7-23. The Initial Source Port is not applying power to  $V_{BUS}$  at the beginning of this sequence.

After a Hard Reset, a Provider/Consumer shall return to its Source role and a Consumer/Provider shall return to its Sink role.

Figure 7-29 Type-A to Type-B Transition Diagram for Dead Battery Operation

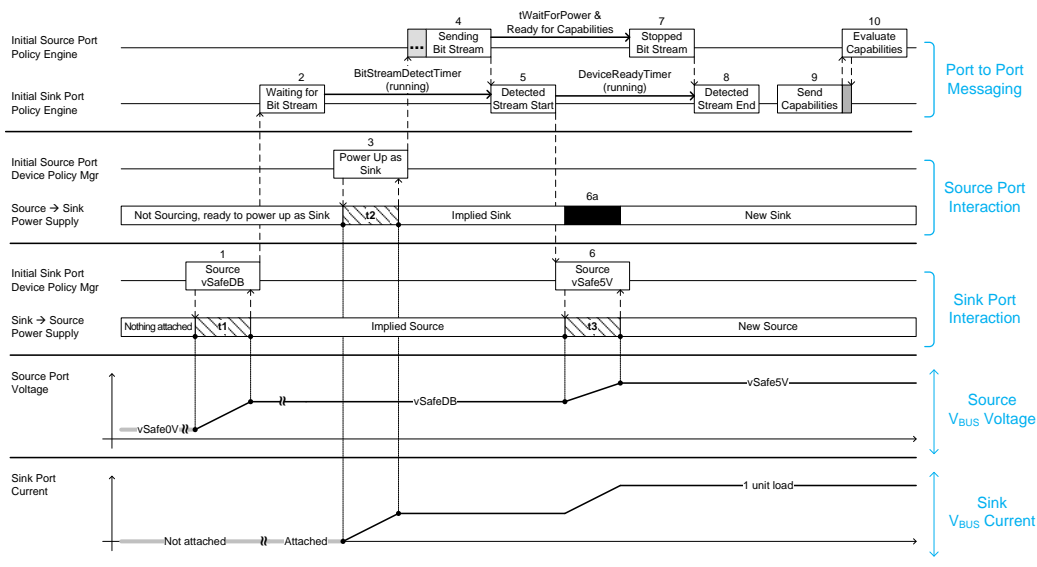


Table 7-20 Sequence Description for Type-A to Type-B Dead Battery Operation

Step	Initial Source Port	Initial Sink Port
1		The Initial Sink detects $V_{BUS}$ is within $vSafe0V$ and performs an implied Power Role Swap to apply $vSafeDB$ to $V_{BUS}$ . The turn on time for $vSafeDB$ is $tTurnOnSafeDB$ (t1).
2		The implied Source begins to wait for the Bit Stream after having applied $vSafeDB$ to $V_{BUS}$ . If an attach does not occur before $BitStreamDetectTimer$ times out, then the implied Source removes $vSafeDB$ and discharges $V_{BUS}$ to $vSafe0V$ . The Dead Battery start up routine for the Source will loop this behavior until an attach occurs (see Section 4.1.1). The loop is not illustrated in this diagram.
3	At some point in time an unpowered Source waiting to power up as the implied Sink is attached to an implied Source. The turn on time of the implied Sink is $tTurnOnImpliedSink$ (t2). The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.	
4	If the implied Sink is unpowered it sends the Bit Stream after having been powered up for some time, by $vSafeDB$ from the implied Source.	
5		When the Bit Stream is detected the $DeviceReadyTimer$ is started and this diagram assumes the Bit Stream will end before the $DeviceReadyTimer$ has timed out.
6		The implied Source will apply $vSafe5V$ to $V_{BUS}$ after the start of the Bit Stream has been detected. The transition time to change from $vSafeDB$ to $vSafe5V$ is $tSafeDBtoSafe5V$ (t3).
6a	The implied Sink will transition to new Sink operation as the $V_{BUS}$ voltage changes from $vSafeDB$ to $vSafe5V$ and may draw up to 1 unit load (as specified in [USB 2.0] or [USB 3.1]) once $vSafe5V$ is reached. The Sink shall not violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.	
7	The new Sink stops sending the Bit Stream when the Policy Engine is ready to accept a $Source\_Capabilities$ Message.	
8		The new Source detects the end of the Bit Stream.
9		The Policy Engine of the new Source sends a $Source\_Capabilities$ Message to the new Sink.
10	The Policy Engine of the new Sink evaluates the $Source\_Capabilities$ Message from the new Source and responds as normal for PD negotiation.	

### 7.3.19 No change in Current or Voltage

The interaction of the System Policy, Device Policy, and power supply that shall be followed when the Sink requests the same Voltage and Current as it is currently operating at is shown in Figure 7-30. The sequence that shall be followed is described in Table 7-21. The timing parameters that shall be followed are listed in Table 7-22 and Table 7-23.

Figure 7-30 Transition Diagram for no change in Current or Voltage

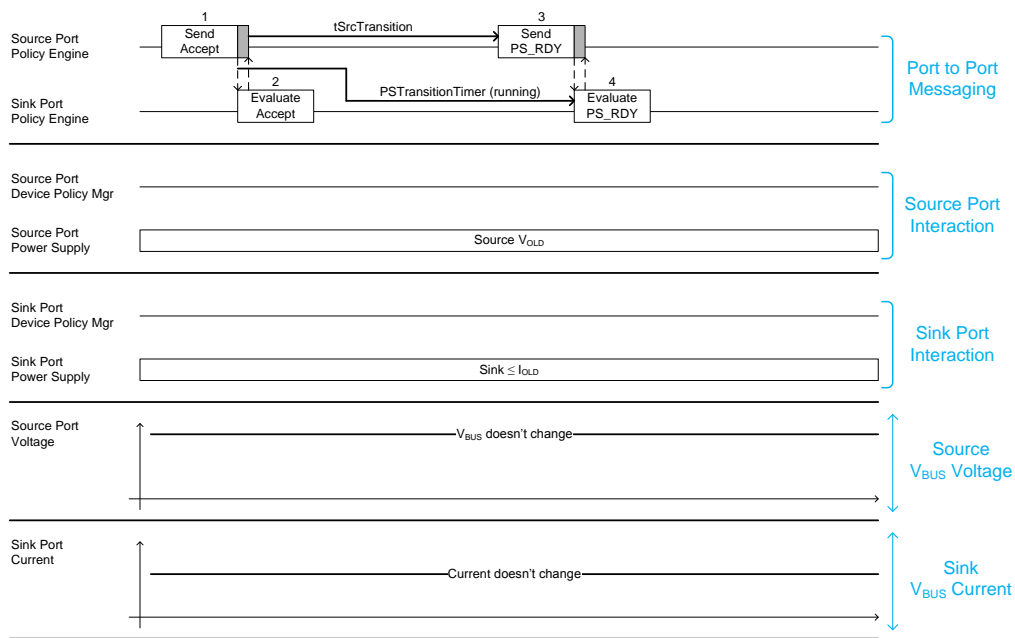


Table 7-21 Sequence Description for no change in Current or Voltage

Step	Source Port	Sink Port
1	Policy Engine sends the <i>Accept</i> Message to the Sink.	Policy Engine receives the <i>Accept</i> Message and starts the <i>PSTransitionTimer</i> .
2	Protocol Layer receives the <i>GoodCRC</i> Message from the Sink.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine then evaluates the <i>Accept</i> Message.
3	The Policy Engine waits <i>tSrcTransition</i> then sends the <i>PS_RDY</i> Message to the Sink.	Policy Engine receives the <i>PS_RDY</i> Message.
4	Policy Engine receives the <i>GoodCRC</i> Message from the Sink. Note: the decision that no power transition is required could be made either by the Device Policy Manager or the power supply depending on implementation.	Protocol Layer sends the <i>GoodCRC</i> Message to the Source. Policy Engine evaluates the <i>PS_RDY</i> Message.

## 7.4 Electrical Parameters

### 7.4.1 Source Electrical Parameters

The Source Electrical Parameters that shall be followed are specified in Table 7-22 and Table 4-4.

Table 7-22 Source Electrical Parameters

Parameter	Description	MIN	TYP	MAX	UNITS	Reference
<i>cSrcBulk<sup>1</sup></i>	Source bulk capacitance when a Port is powered from a dedicated supply.	10			μF	Section 7.1.2
<i>cSrcBulkShared<sup>1</sup></i>	Source bulk capacitance when a Port is powered from a shared supply.	120			μF	Section 7.1.2
<i>cSrcBulkDB</i>	Source bulk capacitance on V <sub>BUS</sub> when applying <i>vSafeDB</i> .	0		10	μF	Section 7.1.13.3
<i>snrSrc</i>	Source's output Signal-to-noise ratio.	31			dB	Section 7.1.13.2
<i>tNewSnk</i>	Time allowed for an initial Source in Swap Standby to transition new Sink operation.			15	ms	Figure 7-20, Figure 7-21
<i>vSrcNew</i> / <i>vSrcNew<sup>2</sup></i>	Fixed Supply output tolerance measured at the Source receptacle.	-5		+5	%	Figure 7-2 Figure 7-3
	Variable supply	As defined by the negotiated min, max levels				
	Battery Supply	As defined by the negotiated min, max levels				
<i>vSrcPeak</i>	<u>The range that a Fixed Supply in Peak Current operation is allowed when overload conditions occur.</u>	<u>-15</u>		<u>+5</u>	<u>%</u>	Table 6-7 Figure 7-7
<i>vSrcValid</i>	The range in addition to <i>vSrcNew</i> which a newly negotiated voltage is considered valid during and after a transition. These limits do not apply to V <sub>BUS</sub> = <i>vSafe5V</i> .	-0.5		0.5	V	Figure 7-2 Figure 7-3
<i>tSafeDBtoSafe5V</i>	Transition time from <i>vSafeDB</i> to <i>vSafe5V</i> .			15	ms	Table 7-20
<i>tShortCctRecover</i>	Source recovery time from short circuit on fault removal.			120	s	Section 7.1.8
<i>tSrcReady</i>	Time from positive/negative transition start (t <sub>0</sub> ) to when V <sub>BUS</sub> is within the range defined by <i>vSrcNew</i> .			285	ms	Figure 7-2, Figure 7-3
<i>tSrcRecover</i>	Time allotted for the Source to recover.	0.66		1	s	Section 7.1.6

Parameter	Description	MIN	TYP	MAX	UNITS	Reference
<i>tSrcSettle</i>	Settling time from positive/negative transition start (t0) to when transitioning voltage is within the specified tolerance of the final voltage.			275	ms	Figure 7-2
<i>tSrcSwapStdby</i>	The maximum time for the Source to transition to Swap Standby.			650	ms	Table 7-11 Table 7-12
<i>tSrcTransient</i>	<u>The maximum time for the Source output voltage to be between <i>vSrcNew</i> and <i>vSrcValid</i> in response to a load transient.</u>			5	ms	<a href="#">Section 7.1.9</a>
<i>tSrcTransition</i>	<u>The time the Source waits before transitioning the power supply to ensure that the Sink has sufficient time to prepare.</u>	25		35	ms	
<i>tSrcTurnOn</i>	Transition time from <i>vSafe0V</i> to <i>vSafe5V</i> .			275	ms	Table 7-14 Table 7-15 Table 7-16 Table 7-17 Table 7-18 Table 7-19
<i>tTurnOnSafeDB</i>	Time to turn on <i>vSafeDB</i> source.			15	ms	Table 7-20
<i>vSrcSlewNeg</i>	Maximum slew rate allowed for negative voltage transitions. Limits current based on a 3 A connector rating and maximum Sink bulk capacitance of 100 $\mu$ F.			-30	mV/ $\mu$ s	Section 7.1.5 Figure 7-3
<i>vSrcSlewPos</i>	Maximum slew rate allowed for positive voltage transitions. Limits current based on a 3 A connector rating and maximum Sink bulk capacitance of 100 $\mu$ F.			30	mV/ $\mu$ s	Section 7.1.4 Figure 7-2
<i>vSrcNeg</i>	Most negative voltage allowed during transition.			-0.3	V	Figure 7-4
<i>tSafe5V</i>	<u>Time to reach <i>vSafe5V</i> max.</u>			275	ms	<a href="#">Section 7.1.5</a> <a href="#">Figure 7-4</a>
<i>tSafe0V</i>	<u>Time to reach <i>vSafe0V</i> max.</u>			650	ms	<a href="#">Section 7.1.6</a> <a href="#">Figure 7-4</a> <a href="#">Table 7-14</a> <a href="#">Table 7-15</a> <a href="#">Table 7-16</a> <a href="#">Table 7-17</a> <a href="#">Table 7-18</a> <a href="#">Table 7-19</a>

Parameter	Description	MIN	TYP	MAX	UNITS	Reference
<p>Note 1: The Source shall charge and discharge the total bulk capacitance to meet the transition time requirements.</p> <p>Note 2: When operating at <i>vSafe5V</i> voltage tolerance is as specified in <a href="#">[USB 2.0]</a> or <a href="#">[USB 3.1]</a> <del>Note 2: The parameter toSrc does not apply to Batteries nor Variable Power Supplies. These supply types are exposed during negotiation as a absolute min and max voltage levels.</del></p> <p>Note 3: NV stands for Negotiated Voltage i.e. the voltage requested by the Sink.</p>						



## 7.4.2 Sink Electrical Parameters

The Sink Electrical Parameters that shall be followed are specified in Table 7-23 and Table 4-4.

Table 7-23 Sink Electrical Parameters

Parameter	Description	MIN	TYP	MAX	UNITS	Reference
<i>cSnkBulk<sup>1</sup></i>	Sink bulk capacitance on V <sub>BUS</sub> at attach.	1		10	μF	Section 7.2.2
<i>cSnkBulkDB<sup>1</sup></i>	Bulk capacitance on V <sub>BUS</sub> when acting as a Sink during Dead Battery.	1		10	μF	Section 7.2.9.3
<i>cSnkBulkPd</i>	Bulk capacitance on V <sub>BUS</sub> a Sink is allowed after a successful negotiation.	1		100	μF	Section 7.2.2
<i>iCapChange</i>	Transient current allowed to flow when the Sink changes its bulk capacitance.			10	mA	Section 7.2.2
<i>iLoadStepRate</i>	Load step di/dt. Refer to <a href="#">[USBType-C1.0] Section 3.7.3.3.2</a> for cable details.			150	mA/μs	Section 7.2.6
<i>iLoadReleaseRate</i>	Load release di/dt. Refer to <a href="#">[USBType-C1.0] Section 3.7.3.3.2</a> for cable details.	-150			mA/μs	Section 7.2.6
<i>iOvershoot</i>	Positive or negative overshoot when a load change occurs; relative to the settled value after the load change. Refer to USB <a href="#">[USBType-C1.0] Section 3.7.3.3.2</a> for cable details.	-230		230	mA	Section 7.2.6
<i>iSafe0mA</i>	Maximum current a Sink is allowed to draw when V <sub>BUS</sub> is driven to <i>vSafeOV</i> .			1.0	mA	Figure 7-23 Figure 7-24
<i>iSnkSwapStdby</i>	Maximum current a Sink can draw during Swap Standby. Ideally this current is very near to 0 mA largely influenced by Port leakage current.			2.5	mA	Section 7.2.7
<i>pHubSusp</i>	Suspend power consumption for a hub. 25mW + 25mW per downstream Port for up to 4 ports.			125	mW	Section 7.2.4
<i>pSnkStdby</i>	<a href="#">Average</a> power consumption <a href="#">during over a 10ms interval while in Sink Standby.</a>			150	mW	Section 7.2.3
<i>pSnkStdbyLimit</i>	<a href="#">Absolute maximum standby power consumption while in Sink Standby.</a>			500	mW	<a href="#">Section 7.2.3</a>
<i>pSnkSusp</i>	Suspend power consumption for a peripheral device.			25	mW	Section 7.2.4
<i>snrSnk</i>	Signal-to-noise ratio of Sink's noise reflected on V <sub>BUS</sub> .	31			dB	Section 7.2.9.2

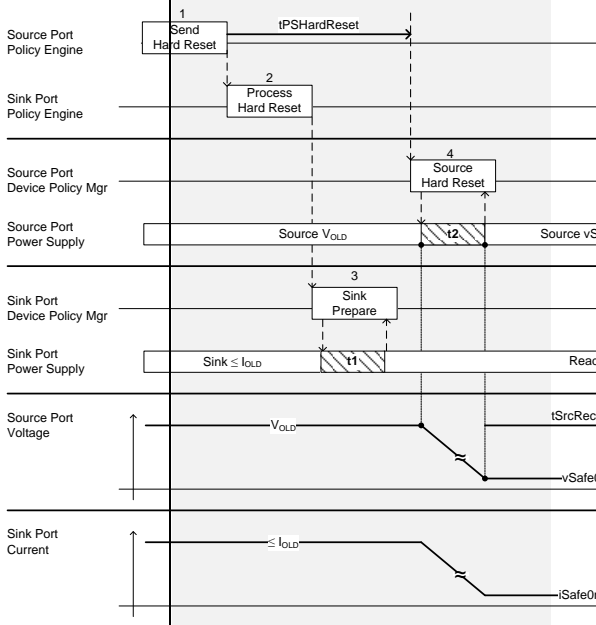
Parameter	Description	MIN	TYP	MAX	UNITS	Reference
<i>tNewSnkRevertToSrc</i>	Hard Reset during Power Role Swap timing parameter.			90	ms	Figure 7-25, Figure 7-26, Figure 7-27,  Figure 7-28
<i>tNewSrc</i>	Maximum time allowed for an initial Sink in Swap Standby to transition to new Source operation.			275	ms	Section 7.2.7 Table 7-11 Table 7-12
<i>tSnkHardResetPrepare</i>	Time allotted for the Sink power electronics to prepare for a Hard Reset.			15	ms	Table 7-15
<i>tSnkNewPower</i>	Maximum transition time between power levels.			15	ms	Section 7.2.3
<i>tSnkRecover</i>	Time for the Sink to resume USB Default Operation.			150	ms	 <p>The diagram illustrates the timing for a Sink Hard Reset. It shows the following sequence of events:</p> <ul style="list-style-type: none"> <li><b>1:</b> Source Port Policy Engine sends a Hard Reset (IPSHardReset) to the Sink Port Policy Engine.</li> <li><b>2:</b> Sink Port Policy Engine processes the Hard Reset.</li> <li><b>3:</b> Sink Port Device Policy Manager initiates Sink Prepare, which occurs while Sink Port Power Supply current is limited to <math>\leq I_{OLD}</math>. The duration of Sink Prepare is <math>t_1</math>.</li> <li><b>4:</b> Source Port Device Policy Manager initiates Source Hard Reset, which occurs while Source Port Power Supply voltage is at <math>V_{OLD}</math>. The duration of Source Hard Reset is <math>t_2</math>.</li> </ul> <p>Waveforms for Source Port Voltage and Sink Port Current show the transition from <math>V_{OLD}</math> to <math>V_{SAFE0}</math> and from <math>\leq I_{OLD}</math> to <math>i_{SAFE0}</math> respectively. Labels <math>t_{SrcRec}</math> and <math>i_{SAFE0}</math> are also present.</p>

Table 7-14  
Table 7-14

Parameter	Description	MIN	TYP	MAX	UNITS	Reference
<i>tSnkStdbby</i>	Time to transition to Sink Standby from Sink.			15	ms	Section 7.2.3
<del><i>tSnkTransition</i></del> <del><i>tSnkSwapStdbby</i></del>	Maximum time for the Sink to transition to SwapStandby.	<del>20</del>		<del>35</del> <del>15</del>	ms	Section 7.2.7
<i>tSwapRecover</i>	Provides settling time after a hard reset before powering back up.	0.5		1	s	Table 7-18 Table 7-19
<i>tTurnOnImpliedSink</i>	Turn on time of implied Sink during Dead Battery operation.			15	ms	Table 7-20

Note 1: If more bypass capacitance than *cSnkBulk* max or *cSnkBulkPd* max is required in the device, then the device must incorporate some form of V<sub>BUS</sub> surge current limiting as described in [\[USB 3.1\]](#) Section 11.4.4.1.

### 7.4.3 Common Electrical Parameters

Electrical Parameters that are common to both the Source and the Sink that shall be followed are specified in Table 7-24.

Table 7-24 Common Source/Sink Electrical Parameters

Parameter	Description	MIN	TYP	MAX	UNITS	Reference
<i>tSafe0V</i>	<a href="#">Time to reach vSafe0V max.</a>			<a href="#">650</a>	<a href="#">ms</a>	<a href="#">Section 7.1.6</a> Figure 7-4 Table 7-14 Table 7-15 Table 7-16 Table 7-17 Table 7-18 Table 7-19
<i>tSafe5V</i>	<a href="#">Time to reach vSafe5V max.</a>			<a href="#">275</a>	<a href="#">ms</a>	<a href="#">Section 7.1.5</a> Figure 7-4
<i>vSafe0V</i>	Safe operating voltage at “zero volts”.	0		0.8	V	Section 7.1.6
<i>vSafe5V</i>	Safe operating voltage at 5V. See <a href="#">[USB 2.0]</a> and <a href="#">[USB 3.1]</a> for allowable V <sub>BUS</sub> voltage range.	<a href="#">[USB 2.0]</a> / <a href="#">[USB 3.1]</a>		<a href="#">[USB 2.0]</a> / <a href="#">[USB 3.1]</a>	V	Section 7.1.6
<i>vSafeDB</i>	Safe operating voltage for Dual-Role ports operating as DB Source. See <a href="#">[USB 2.0]</a> and <a href="#">[USB 3.1]</a> for allowable V <sub>BUS</sub> voltage range.	<a href="#">[USB 2.0]</a> / <a href="#">[USB 3.1]</a>	Refer to Operating Region in Figure 7-9	<a href="#">[USB 2.0]</a> / <a href="#">[USB 3.1]</a>	V	Section 7.1.13.3 Section 7.2.9.3 Table 7-20

## 8. Device Policy

### 8.1 Overview

This section describes the Device Policy and Policy Engine that implements it. For an overview of the architecture and how the Device Policy Manager fits into this architecture, please see Section 2.6.

Field Code Changed

### 8.2 Device Policy Manager

The Device Policy Manager is responsible for managing the power used by one or more USB Power Delivery ports. In order to have sufficient knowledge to complete this task it needs relevant information about the device it resides in. Firstly it has a priori knowledge of the device including the capabilities of the power supply and the receptacles on each Port since these will for example have specific current ratings. It also has to know information from the cable detection module regarding cable insertion, type and rating of cable etc. It also has to have information from the power supply about changes in its capabilities as well as being able to request power supply changes. With all of this information the Device Policy Manager is able to provide up to date information regarding the capabilities available to a specific Port and to manage the power resources within the device.

When working out the capabilities for a given Source Port the Device Policy Manager will take into account firstly the current rating of the Port's receptacle and whether the inserted cable is PD or non-PD rated and if so what is the capability of the plug. This will set an upper bound for the capabilities which may be offered. After this the Device Policy Manager will consider the available power supply resources since this will bound which voltages and currents may be offered. Finally, the Device Policy Manager will consider what power is currently allocated to other ports, which power is in the Power Reserve and any other amendments to Policy from the System Policy Manager. The Device Policy Manager will offer a set of capabilities within the bounds detailed above.

When selecting a capability for a given Sink Port the Device Policy Manager will first look at the capabilities offered by the Source. The Device Policy Manager will take into account the current rating of the Port's receptacle and for Type-A/Type-B plugs whether the inserted cable is PD or non-PD rated and if so what is the capability of the plug. This will set an upper bound for the capabilities which may be requested. The Device Policy Manager will also consider which capabilities are required by the Sink in order to operate. If an appropriate match for Voltage and Current can be found within the limits of the receptacle and cable then this will be requested from the Source. If an appropriate match cannot be found then a request for an offered voltage and current will be made, along with an indication of a capability mismatch.

For Dual-Role Ports the Device Policy Manager manages the functionality of both a Source and a Sink. In addition it is able to manage the Power Role Swap process between the two. In terms of power management this could mean that a Port which is initially consuming power as a Sink is able to become a power resource as a Source. Conversely, attached Sources may request that power be provided to them.

The functionality within the Device Policy Manager (and to a certain extent the Policy Engine) is scalable depending on the complexity of the device, including the number of different power supply capabilities and the number of different features supported for example System Policy Manager interface or Capability Mismatch, and the number of ports being managed. Within these parameters it is possible to implement devices from very simple power supplies to more complex power supplies or devices such as USB hubs or Hard Drives. Within multiport devices it is also permitted to have a combination of USB Power Delivery and non-USB Power Delivery ports which should all be managed by the Device Policy Manager.

As noted in Section 2.6 the logical architecture used in the PD specification will vary depending on the implementation. This means that different implementations of the Device Policy Manager may be relative small or large depending on the complexity of the device, as indicated above. It is also possible to allocate different responsibilities between the Policy Engine and the Device Policy Manager, which will lead to different types of architectures and interfaces.

The Device Policy Manager is responsible for the following:

- Maintaining the Local Policy for the device
- For a Source, monitoring the present capabilities and triggering notifications of the change.
- For a Sink, evaluating and responding to capabilities related requests from the Policy Engine for a given Port.
- Control of the Source/Sink in the device.
- Control of the cable detection module for each Port.
- Interface to the Policy Engine for a given Port.

The Device Policy Manager is responsible for the following optional features when implemented:

- Communications with the System Policy over USB.
- For Sources with multiple ports monitoring and balancing power requirements across these ports.
- Monitoring of batteries and AC power supplies.

### 8.2.1 Capabilities

The Device Policy Manager in a Provider shall know the power supplies available in the device and their capabilities. In addition it shall be aware of any other PD Sources of power such as batteries and AC inputs. The available power sources and existing demands on the device shall be taken into account when presenting capabilities to a Sink.

The Device Policy Manager in a Consumer shall know the requirements of the Sink and use this to evaluate the capabilities offered by a Source. It shall be aware of its own power sources e.g. Batteries or AC supplies where these have a bearing on its operation as a Sink.

The Device Policy Manager in a Provider/Consumer or Consumer/Provider shall combine the above capabilities and shall also be able to present the dual-role nature of the device to an attached PD Capable device.

### 8.2.2 System Policy

A given PD Capable device may have no USB capability, or PD may have been added to a USB device in such a way that PD is not integrated with USB. In these two cases there shall be no requirement for the Device Policy Manager to interact with the USB interface of the device. The following requirements shall only apply to PD devices that expose PD functionality over USB.

The Device Policy Manager shall communicate over USB with the System Policy Manager according to the requirements detailed in Chapter 9. Whenever requested the Device Policy Manager shall implement a Local Policy according to that requested by the System Policy Manager. For example the System Policy Manager may request that a battery powered Device temporarily stops charging so that there is sufficient power for a HDD to spin up.

The System Policy Manager may operate in an "Intrusive" POLICY\_MODE where supported by a PD Capable device. This means that each and every request, which can be handled by the System Policy Manager, received by the Device Policy Manager is passed up to the USB Host for a response. In practice this means that there is a Device Policy Manager running on the USB Host and communicating via the USB bus to the PD Module. This enables software based implementations of the Device Policy Manager but has serious implications in terms of response times which have to be taken into consideration.

Note: that due to timing constraints, a PD Capable device shall be able to respond autonomously to all time-critical PD related requests.

### 8.2.3 Control of Source/Sink

The Device Policy Manager for a Provider shall manage the power supply for each PD Source Port and shall know at any given time what the negotiated power is. It shall request transitions of the supply and inform the Policy Engine whenever a transition completes.

The Device Policy Manager for a Consumer shall manage the Sink for each PD Sink Port and shall know at any given time what the negotiated power is.

The Device Policy Manager for a Provider/Consumer or Consumer/Provider shall manage the transition between Source/Sink roles for each PD Dual-Role Port and shall know at any given time what operational role the Port is in.

The Device Policy Manager for a Consumer/Provider shall be able to detect and handle cases where back-powering is required due to a Dead Battery on the Provider/Consumer side. It shall determine the absence of  $V_{BUS}$  and the Provider/Consumer's ability to be back-powered from the Cable Detection module. It shall then initiate Provider role and instruct the power supply to start providing default output power. Refer to Section 4.1.

The Device Policy Manager for a Provider/Consumer may be able to detect and handle cases where back-powering is possible due to a Dead Battery. Where supported it shall determine the presence of  $V_{BUS}$  from the Cable Detection module. It shall then initiate Consumer role and instruct the power supply to start sinking default input power.

## 8.2.4 Cable Detection

### 8.2.4.1 Device Policy Manager in a Provider

The Device Policy Manager in the Provider shall control the Cable Detection module and shall be able to use the Cable Detection module to determine the attachment of a cable and for Type-A/Type-B whether it is a USB Power Delivery or non-USB Power Delivery cable.

This information and the type of receptacle on the local device shall be used to determine the capabilities of the Port and attached cabling. Note: that it may be necessary for the Device Policy Manager to also initiate additional discovery via Structured VDM Messages in order to determine the full capabilities of the cabling (see Section 6.4.4.2).

### 8.2.4.2 Device Policy Manager in a Consumer

The Device Policy Manager in a Consumer controls the Cable Detection module and shall be able to use the Cable Detection module to determine the attachment of a USB Power Delivery or non-USB Power Delivery cable.

The type of cabling and receptacle together determine the current carrying ability of the Port and attached cabling.

### 8.2.4.3 Device Policy Manager in a Consumer/Provider

The Device Policy Manager in a Consumer/Provider inherits characteristics of Consumers and Providers and shall control the Cable Detection module in order to support the Dead Battery back-powering case to determine the following for a given Port:

- Attachment of a USB Power Delivery Provider/Consumer which supports Dead Battery back-powering
- Presence of  $V_{BUS}$ .

### 8.2.4.4 Device Policy Manager in a Provider/Consumer

The Device Policy Manager in a Provider/Consumer inherits characteristics of Consumers and Providers and may control the Cable Detection module in order to support the Dead Battery back-powering case to determine the following for a given Port:

- Presence of  $V_{BUS}$ .

## 8.2.5 Managing Power Requirements

The Device Policy Manager in a Provider shall be aware of the power requirements of all devices connected to its Source Ports. This includes being aware of any reserve power that may be required by devices in the future and ensuring that power is shared optimally amongst attached PD Capable devices. This is a key function of the Device Policy Manager, whose implementation is critical to ensuring that all PD Capable devices get the power they require in a timely fashion in order to facilitate smooth operation. This is balanced by the fact that the Device Policy Manager is responsible for managing the sources of power that are, by definition, finite.

The Consumer's Device Policy Manager shall ensure that it takes no more power than is required to perform its functions and gives back unneeded power whenever possible (in such cases the Provider shall maintain a Power Reserve to ensure future operation is possible).

### 8.2.5.1 Managing the Power Reserve

There may be some products where a Device has certain functionality at one power level and a greater functionality at another, for example a Printer/Scanner that operates only as a printer with one power level and as a scanner if it can get more power. Visibility of the linkage between power and functionality will only be apparent at the USB Host; however the Device Policy Manager provides the mechanisms to manage the power requirements of such Devices.

Devices with the GiveBack flag cleared report Operating Current and Maximum Operating Current (see Section 6.4.2). For many Devices the Operating Current and the Maximum Operating Current will be the same. Devices with highly variable loads, such as Hard Disk Drives, may use Maximum Operating Current.

Devices with the GiveBack flag set report Operating Current and Minimum Operating Current (see Section 6.4.2). For many Devices the Operating Current and the Minimum Operating Current will be the same. Devices that charge their own batteries may use the Minimum Operating Current and GiveBack flag.

For example in the first case, a mobile device may require 500mA to operate, but would like an additional 1000mA to charge its battery. The mobile device would set the GiveBack flag (see Section 6.4.2.2) and request 500mA in the Minimum Operating Current field and 1500mA in the Operating Current field (provided that 1500mA was offered by the Source) indicating to the Provider that it could temporarily recover the 1000mA to meet a transitory request.

In the second case, a Hard Disk Drive (HDD) may require 2A to spin-up, but only 1A to operate. At startup the HDD would request Maximum Operating Current of 2A and an Operating Current of 2A. After the drive is spun-up and ready to operate it would make another request of 1A for its Operating Current and 2A for its Maximum Operating Current. Over time, its inactivity timers may expire and the HDD will go to a lower power state. When the HDD is next accessed, it has to spin-up again. So it will request an Operating Current of 2A and a Maximum Operating Current of 2A. The Provider may have the extra power available immediately and can immediately honor the request. If the power is not available, the Provider may have to harvest power, for example use the *GotoMin* Message to get back some power before honoring the HDD's request. In such a case, the HDD would be told to wait using a *Wait* Message. The HDD continues to Request additional power until the request is finally granted.

It shall be the Device Policy Manager's responsibility to allocate power and maintain a Power Reserve so as not to over-subscribe its available power resource. A Device with multiple ports such as a Hub shall always be able to meet the incremental demands of the Port requiring the highest incremental power from its Power Reserve.

The *GotoMin* Message is designed to allow the Provider to reclaim power from one Port to support a Consumer on another Port that temporarily requires additional power to perform some short term operation. In the example above, the mobile device that is being charged reduces its charge rate to allow a Device Policy Manager to meet a request from an HDD for start-up current required to spin-up its platters. Any power which is available to be reclaimed using a *GotoMin* Message may be counted as part of the Power Reserve.

A Consumer requesting power shall take into account its operational requirements when advertising its ability to temporarily return power. For example, a mobile device with a Dead Battery that is being used to make a call should make a request that retains sufficient power to continue the call. When the Consumer's requirements change, it shall re-negotiate its power to reflect the changed requirements.

### 8.2.5.2 Power Capability Mismatch

A capability mismatch occurs when a Consumer cannot obtain required power from a Provider [\(or the Source is not PD Capable\)](#) and the Consumer requires such capabilities to operate. Different actions are taken by the Device Policy Manager and the System Policy Manager in this case.

#### 8.2.5.2.1 Local device handling of mismatch

The Consumer's Device Policy Manager shall cause a Message to be displayed to the end user that a power capability mismatch has occurred. Examples of such feedback can include:

- For a simple Device an LED may be used to indicate the failure. For example, during connection the LED could be solid amber. If the connection is successful the LED could change to green. If the connection fails it could be red or alternately blink amber.

- A more sophisticated Device with a user interface, e.g., a mobile device or monitor, should provide notification through the user interface on the Device.

The Provider's Device Policy Manager may cause a Message to be displayed to the user of the power capability mismatch.

#### 8.2.5.2.2 Device Policy Manager Communication with System Policy

In a USB Power Delivery aware system with an active System Policy manager (see Section 8.2.2), the Device Policy Manager shall notify the System Policy Manager of the mismatch. This information shall be passed back to the System Policy Manager using the mechanisms described in Chapter 8.3.3. The System Policy Manager should ensure that the user is informed of the condition. When another Port in the system could satisfy the Consumer's power requirements the user should be directed to move the Device to the alternate Port.

In order to identify a more suitable Source Port for the Consumer the System Policy Manager shall communicate with the Device Policy Manager in order to determine the Consumer's requirements. The Device Policy Manager shall use a **Get\_Sink\_Cap** Message (see Section 6.3.8) to discover which power levels can be utilized by the Consumer.

### 8.2.6 Use of "Externally Powered" bit with Batteries and AC supplies

The Device Policy Manager in a Provider or Consumer may monitor the status of any variable sources of power that could have an impact on its capabilities as a Source such as Batteries and AC supplies and reflect this in the "Externally Powered" bit (see Section 6.4.1.3.1.3) provided as part of the Source or Sink Capabilities Message (see Section 6.4.1). When monitored, and a USB interface is supported, the External Power status (see Section 9.4.4) and the battery state (see Section 9.4.2) shall also be reported to the System Policy Manager using the USB interface.

#### 8.2.6.1 AC Supplies

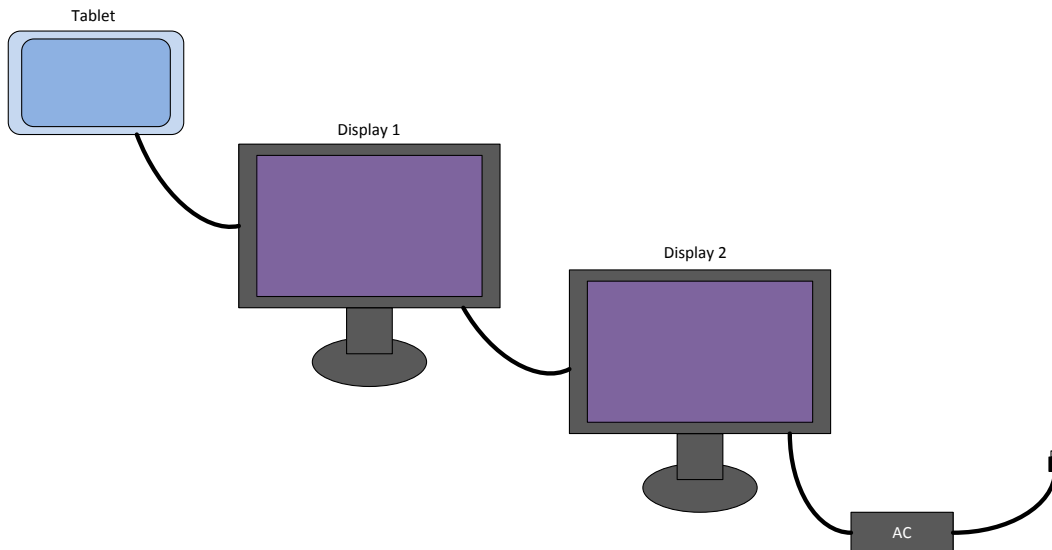
The Externally Powered bit provided by Sources and Sinks (see Sections 6.4.1.2.3.3 and 6.4.1.3.1.3) refers to the presence of an external AC power supply (i.e. from a wall wart) that is providing 100% of the power to a given device. This means that the device is solely getting its power from this power supply and is not aggregating power from other, non-external, power sources. Logically this power can either be from an AC supply directly connected to the device or from an AC supply connected to an attached device, which is also getting 100% of its power from this power supply. The Externally Powered bit is in this way communicated through a PD system indicating that the origin of the power is from a single or multiple AC supplies:

- If the "Externally Powered" bit is set then that power is originally sourced from an AC supply.
- Devices capable of consuming on multiple ports can only claim that they are "Externally Powered" for the power advertised as a provider Port if 100% of the consumed power is from external supplies, (e.g. multiple AC supplies).
- This concept applies as the power is routed through multiple provider and Consumer tiers, so, as an example. Power provided out of a monitor that is connected to a monitor that gets power from an AC supply, will claim it is "Externally Powered" even though it is not directly connected to the AC supply.

An example use case is a Tablet computer that is used with two USB A/V displays that are daisy chained (see Figure 8-1). The tablet and 1st display are not externally powered, (meaning, they have no source of power outside of USB PD). The 2nd display has an external supply attached which may either be a USB PD based supply or some other form of external supply. When the displays are connected as shown, the power adapter attached to the 2nd display is able to power both the 1st display and the tablet. In this case the 2nd display will indicate the presence of the wall wart, to the 1st display, by setting its "Externally Powered" bit. The 1st display will then in turn indicate the presence of an external supply to the tablet by setting its "Externally Powered" bit. Power is transmitted through the system to all devices, provided that there is sufficient power available from the external supply.



Figure 8-1 Example of daisy chained displays



### 8.2.6.2 Battery Supplies

When monitored, and a USB interface is supported, the battery state shall be reported to the System Policy Manager using the USB interface.

A simplified algorithm is detailed below to ensure that battery powered devices will get charge from non-battery powered devices when possible, and also to ensure that devices do not constantly Power Role Swap back and forth. When two devices are connected that are not Externally Powered, they should define their own policies so as to prevent constant Power Role Swapping.

This algorithm uses the “Externally Powered” bit (see Sections 6.4.1.2.3.3 and 6.4.1.3.1.3), thus the decisions are based on the availability of an external supply, not the full capabilities of a system or device or product.

Rules:

1. Provider/Consumers using external sources (“Externally powered” bit set) shall always deny Power Role Swap requests from Consumer/Providers not using external sources (“Externally Powered” bit cleared).
2. Provider/Consumers not using external sources (“Externally Powered” bit cleared) shall always accept a Power Role Swap request from a Consumer/Provider using external power sources (“Externally Powered” bit set) unless the requester is not able to provide the requirements of the present Provider/Consumer.

### 8.2.7 Interface to the Policy Engine

The Device Policy Manager shall maintain an interface to the Policy Engine for each Port in the device.

#### 8.2.7.1 Device Policy Manager in a Provider

The Device Policy Manager in a Provider shall also provide the following functions to the Policy Engine:

- Inform the Policy Engine of changes in cable/ device attachment status for a given cable.
- Inform the Policy Engine whenever the Source capabilities available for a Port change.
- Evaluate requests from an attached Consumer and provide responses to the Policy Engine.
- Respond to requests for power supply transitions from the Policy Engine.

- Indication to Policy Engine when power supply transitions are complete.
- Maintain a Power Reserve for devices operating on a Port at less than maximum power.

#### 8.2.7.2 Device Policy Manager in a Consumer

The Device Policy Manager in a Consumer shall also provide the following functions to the Policy Engine:

- Inform the Policy Engine of changes in cable/device attachment status.
- Inform the Policy Engine whenever the power requirements for a Port change.
- Evaluate Source capabilities and provide suitable responses:
  - Request from offered capabilities
  - Indicate whether additional power is required
- Respond to requests for Sink transitions from the Policy Engine.

#### 8.2.7.3 Device Policy Manager in a Provider/Consumer

The Device Policy Manager in a Provider/Consumer shall provide the following functions to the Policy Engine:

- Provider Device Policy Manager
- Consumer Device Policy Manager
- Interface for the Policy Engine to request power supply transitions from Source to Sink and vice versa.
- Indications to Policy Engine during Power Role Swap transitions.

#### 8.2.7.4 Device Policy Manager in a Provider/Consumer Dead Battery handling

The Device Policy Manager in a Provider/Consumer with a battery should also provide:

- Detection and handling of back powering in the case of a Type-A to Type-B Dead Battery (see Section 4.1.1).

In this scenario a Provider/Consumer shall:

- Detect that  $V_{BUS}$  is present and that its battery is dead
- Switch to Consumer role without using a *PR\_Swap* Message
- Use  $V_{BUS}$  to power the USB Power Delivery communications

#### 8.2.7.5 Device Policy Manager in a Consumer/Provider

The Device Policy Manager in a Consumer/Provider shall provide the following functions to the Policy Engine:

- Consumer Device Policy Manager
- Provider Device Policy Manager
- Interface for the Policy Engine to request power supply transitions from Sink to Source and vice versa.
- Indications to Policy Engine during Power Role Swap transitions.

#### 8.2.7.6 Device Policy Manager in a Consumer/Provider Dead Battery handling

The Device Policy Manager in a Consumer/Provider shall also provide:

- Detection and handling of back powering in the case of a Dead Battery (see Section 4.1).

In this scenario a Consumer/Provider shall:

- Detect that a Provider/Consumer with a Dead Battery capable of being back powered is attached
- Check that  $V_{BUS}$  is not present.
- Temporarily output *vSafeDB* on  $V_{BUS}$  to provide power to the Provider/Consumer so that it can send a Bit Stream.
- Detect Bit Stream sent by Provider/Consumer.
- Switch to the Provider role without using a *PR\_Swap* Message and output *vSafe5V* on  $V_{BUS}$  that will power the Provider/Consumer's PD communications.

## 8.3 Policy Engine

### 8.3.1 Introduction

There is one Policy Engine instance ~~per SOP\*~~ per Port that interacts with the Device Policy Manager in order to implement the present Local Policy for that particular Port. This section includes:

- Message sequences for various operations
- A state diagram of the Policy Engine for a Source Port
- A state diagram of the Policy Engine for a Sink Port
- A state diagram of the Policy Engine for Dual-Role Ports
- State diagrams for handling Soft Reset
- State diagrams for handling *Ping* Messages
- State diagrams for the Provider/Consumer and Consumer/Provider dead-battery/back-powering case
- State diagrams for Hard Reset
- State diagrams for BIST

### 8.3.2 Message Sequence Diagrams

The Device Policy Engine drives the Message sequences and responses based on both the expected Message sequences and the present Local Policy. This section contains sequence diagrams that highlight some of the more interesting transactions. It is by no means a complete summary of all possible combinations, but is illustrative in nature.

#### 8.3.2.1 Basic Message Exchange

Figure 8-2 Basic Message Exchange (Successful) below illustrates how a Message is sent. Note that the sender may be either a Source or Sink while the receiver may be either a Sink or Source. The basic Message sequence is the same. It starts when the Message Sender's Protocol Layer at the behest of its Policy Engine forms a Message that it passes to the Physical Layer.

Figure 8-2 Basic Message Exchange (Successful)

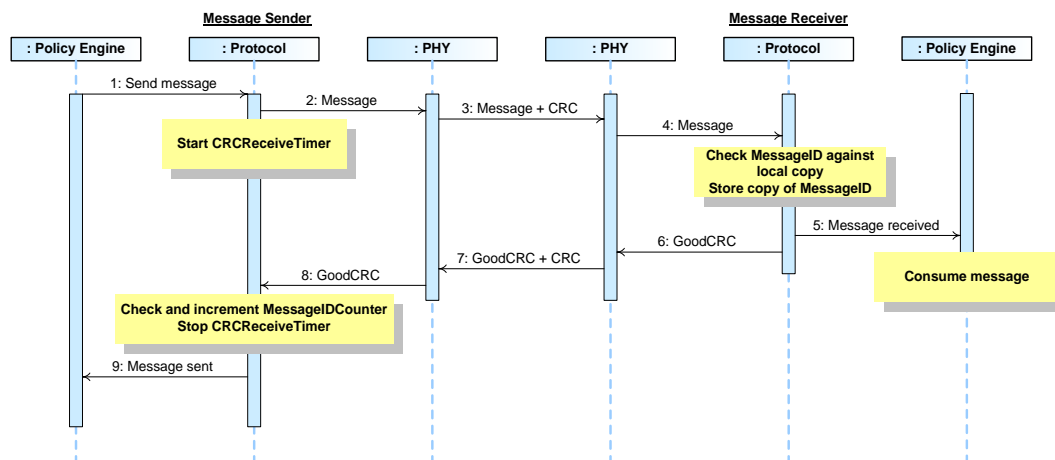


Table 8-1 Basic Message Flow

Step	Message Sender	Message Receiver
1	Policy Engine directs Protocol Layer to send a Message.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends a CRC and sends the Message.	Physical Layer receives the Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Message to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer forwards the received Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it to the Physical Layer.
7	Physical Layer receives the Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer. Protocol Layer checks and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> .	
9	Protocol Layer informs the Policy Engine that the Message was successfully sent.	

8.3.2.2 Errors in Basic Message flow

There are various points during the Message flow where failures in communication or other issues can occur. Figure 8-3 is an annotated version of Figure 8-2 indicating at which points issues may occur.

Figure 8-3 Basic Message flow indicating possible errors

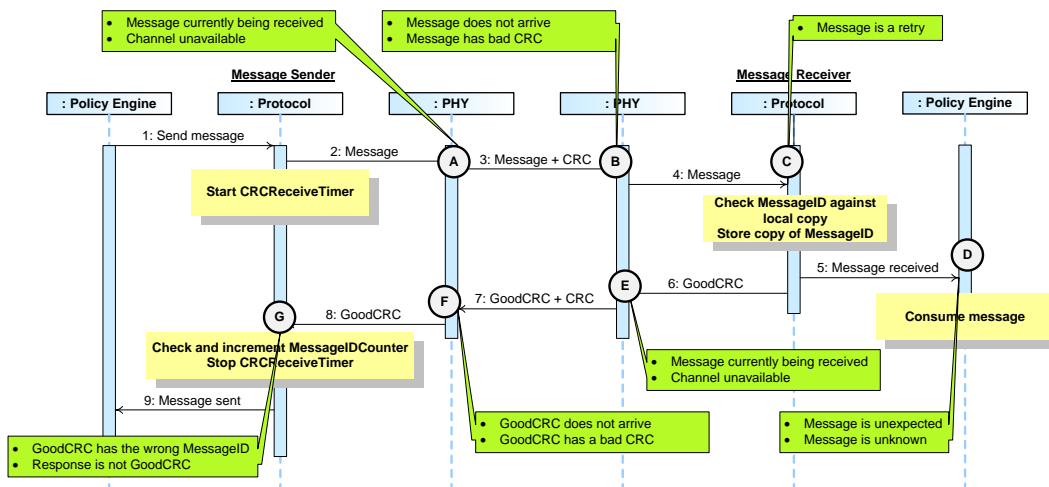


Table 8-2 Potential issues in Basic Message Flow

Point	Possible issues
A	<ol style="list-style-type: none"> <li>1. There is an incoming Message on the channel meaning that the PHY Layer is unable to send. In this case the outgoing Message is removed from the queue and the incoming Message processed.</li> <li>2. Due to some sort of noise on the line it is not possible to transmit. In this case the outgoing Message is discarded by the PHY Layer. Retransmission is via the Protocol Layer's normal mechanism.</li> </ol>
B	<ol style="list-style-type: none"> <li>1. Message does not arrive at the Physical Layer due to noise on the channel.</li> <li>2. Message arrives but has been corrupted and has a bad CRC.</li> </ol> <p>There is no Message to passed up to the Protocol Layer on the receiver which means a <i>GoodCRC</i> Message is not sent. This leads to a <i>CRCReceiveTimer</i> timeout in the Message Sender.</p>
C	<ol style="list-style-type: none"> <li>1. <i>MessageID</i> of received Message matches stored <i>MessageID</i> so this is a retry. Message is not passed up to the Policy Engine.</li> </ol>
D	<ol style="list-style-type: none"> <li>1. Policy Engine receives a known Message that it was not expecting.</li> <li>2. Policy Engine receives an unknown (unrecognised) Message.</li> </ol> <p>These cases are errors in the protocol which leads to the generation of a <i>Soft_Reset</i> Message.</p>
E	Same as point A but at the Message Receiver side.
F	<ol style="list-style-type: none"> <li>1. <i>GoodCRC</i> Message response does not arrive Message Sender side due to the noise on the channel.</li> <li>2. <i>GoodCRC</i> Message response arrives but has a bad CRC.</li> </ol> <p>A <i>GoodCRC</i> Message is not received by the Message Sender's Protocol Layer. This leads to a <i>CRCReceiveTimer</i> timeout in the Message Sender.</p>
G	<ol style="list-style-type: none"> <li>1. <i>GoodCRC</i> Message is received but does contain the same <i>MessageID</i> as the transmitted Message.</li> <li>2. A Message is received but it is not a <i>GoodCRC</i> Message (similar case to that of an unexpected or unknown Message but this time detected in the Protocol Layer).</li> </ol> <p>Both of these issues indicate errors in receiving an expected <i>GoodCRC</i> Message which will lead to a <i>CRCReceiveTimer</i> timeout in the Protocol Layer and a subsequent retry (except for communications with Cable Plugs).</p>

Figure 8-4 illustrates one of these cases; the basic Message flow with a retry due to a bad CRC at the Message Receiver. It starts when the Message Sender's Protocol Layer at the behest of its Policy Engine forms a Message that it passes to the Physical Layer. The Protocol Layer is responsible for retries on a "3:n strikes and you are out" basis (*nRetryCount*).

Field Code Changed

Figure 8-4 Basic Message Flow with Bad CRC followed by a Retry

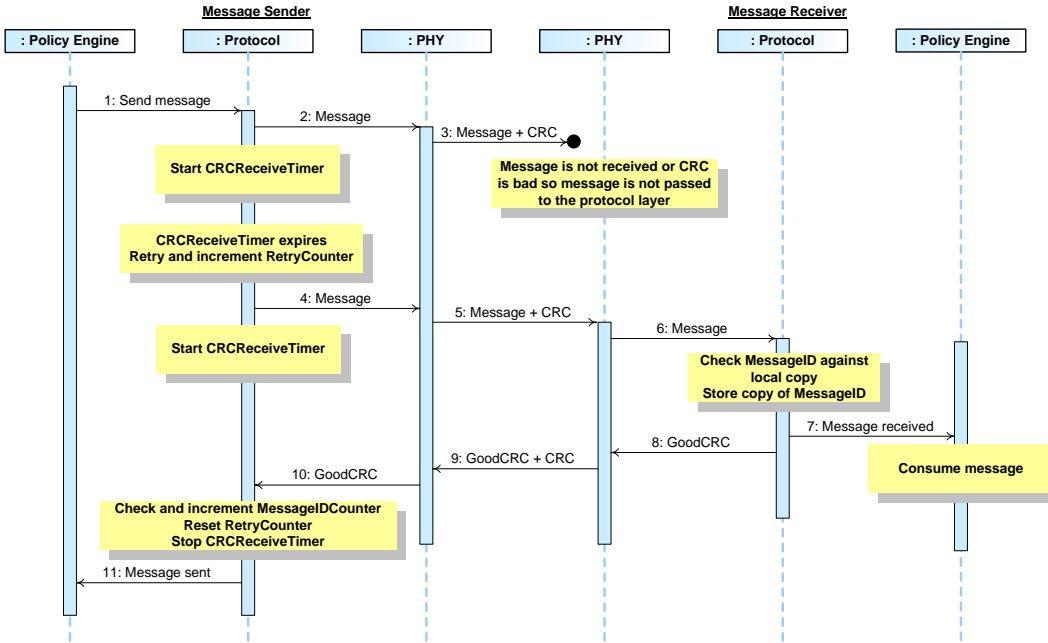


Table 8-3 Basic Message Flow with CRC failure

Step	Message Sender	Message Receiver
1	Policy Engine directs Protocol Layer to send a Message.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends a CRC and sends the Message.	Physical Layer receives no Message or a Message with an incorrect CRC. Nothing is passed to Protocol Layer.
4	Since no response is received, the <i>CRCReceiveTimer</i> will expire and trigger the first retry by the Protocol Layer. The <i>RetryCounter</i> is incremented. Protocol Layer passes the Message to the Physical Layer. Starts <i>CRCReceiveTimer</i> .	
5	Physical Layer appends a CRC and sends the Message.	Physical Layer receives the Message and checks the CRC to verify the Message.
6		Physical Layer removes the CRC and forwards the Message to the Protocol Layer.
7		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer forwards the received Message information to the Policy Engine that consumes it.
8		Protocol Layer generates a <i>GoodCRC</i> Message and passes it to the Physical Layer.

Step	Message Sender	Message Receiver
9	Physical Layer receives the Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
10	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
11	Protocol Layer verifies the <i>MessageID</i> , stops <i>CRCReceiveTimer</i> and resets the <i>RetryCounter</i> . Protocol Layer informs the Policy Engine that the Message was successfully sent.	

### 8.3.2.3 Power Negotiation

Figure 8-5 illustrates an example of a successful Message flow during Power Negotiation. The negotiation goes through 5 distinct phases:

- The Source sends out its power capabilities in a *Source\_Capabilities* Message.
- The Sink evaluates these capabilities and in the request phase selects one power level by sending a *Request* Message.
- The Source evaluates the request and accepts the request with an *Accept* Message.
- The Source transitions to the new power level and then informs the Sink by sending a *PS\_RDY* Message.
- The Sink starts using the new power level.

Figure 8-5 Successful Power Negotiation

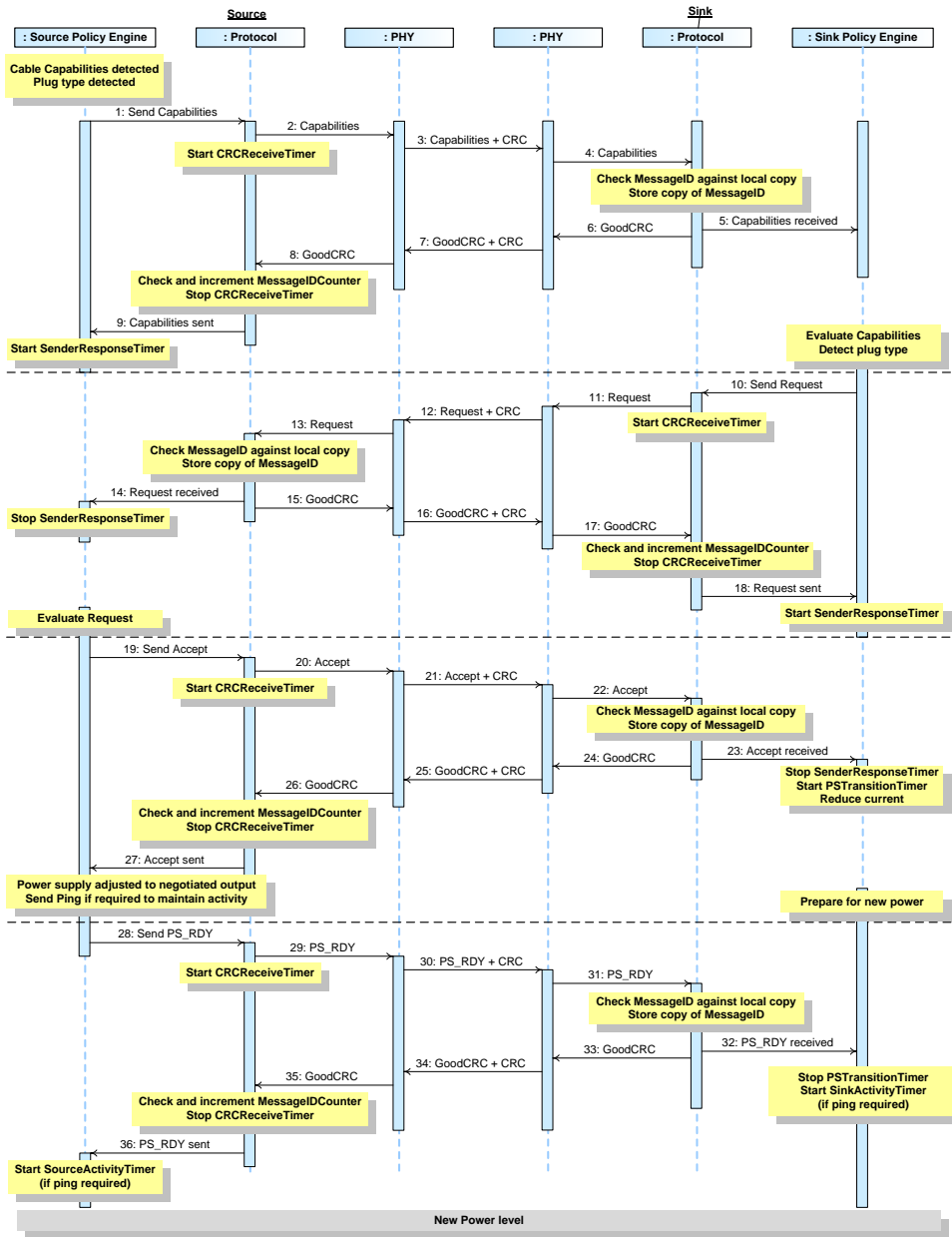




Table 8-4 below provides a detailed explanation of what happens at each labeled step in Figure 8-5 above.

Table 8-4 Steps for a successful Power Negotiation

Step	Source	Sink
1	The Cable Capabilities or Plug Type are detected if these are not already known (see Section 4.4). Policy Engine directs the Protocol Layer to send a <i>Source_Capabilities</i> Message that represents the power supply's present capabilities.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>Source_Capabilities</i> Message.	Physical Layer receives the <i>Source_Capabilities</i> Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>Source_Capabilities</i> Message to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Source_Capabilities</i> Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Source_Capabilities</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .	
10		Policy Engine evaluates the <i>Source_Capabilities</i> Message sent by the Source, detects the plug type if this is necessary (see Section 4.4) and selects which power it would like. It tells the Protocol Layer to form the data (e.g. Power Data Object) that represents its Request into a Message.
11		Protocol Layer creates the <i>Request</i> Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	Physical Layer receives the <i>Request</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>Request</i> Message.
13	Physical Layer removes the CRC and forwards the <i>Request</i> Message to the Protocol Layer.	
14	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the Request information to the Policy Engine. Policy Engine stops <i>SenderResponseTimer</i> .	
15	The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.	

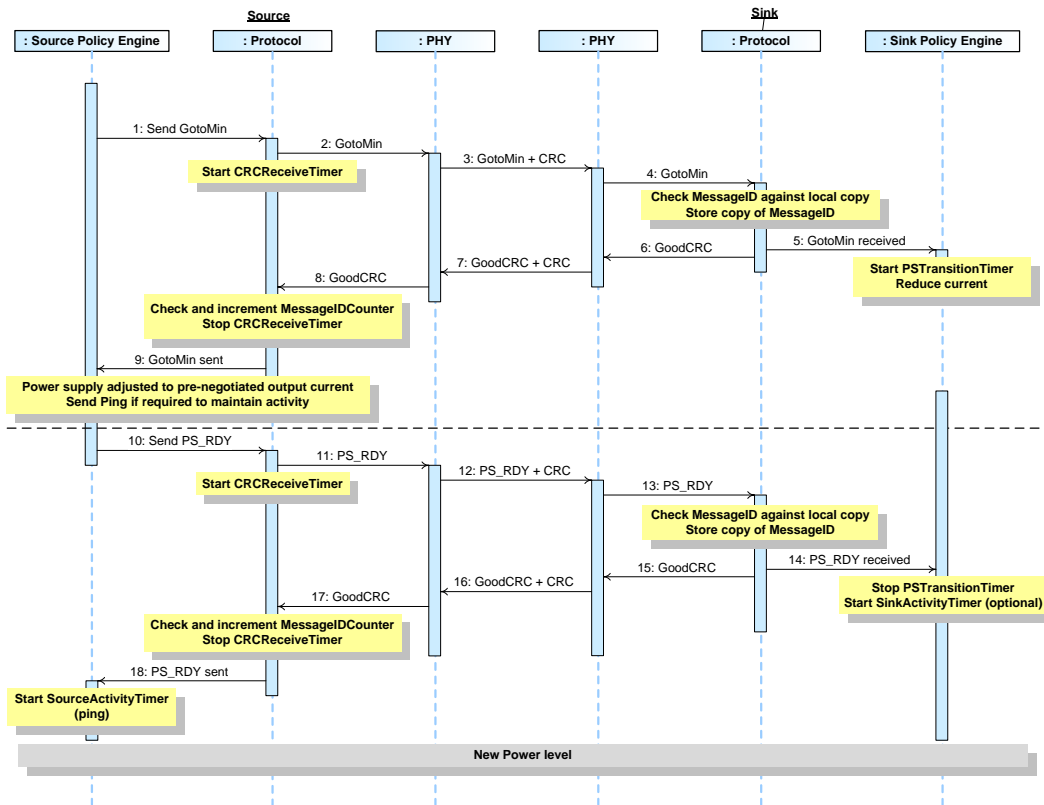
Step	Source	Sink
16	Physical Layer appends CRC and sends the Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		The protocol Layer verifies and increments the <i>MessageIDCounter</i> . It informs the Policy Engine that the <i>Request</i> Message was successfully sent. The Protocol Layer stops the <i>CRCReceiveTimer</i> . The Policy Engine starts <i>SenderResponseTimer</i> .
19	Policy Engine evaluates the <i>Request</i> Message sent by the Sink and decides if it can meet the request. It tells the Protocol Layer to form an <i>Accept</i> Message.	
20	The Protocol Layer forms the <i>Accept</i> Message that is passed to the Physical Layer and starts the <i>CRCReceiveTimer</i> .	
21	Physical Layer appends CRC and sends the <i>Accept</i> Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
22		Physical Layer forwards the <i>Accept</i> Message to the Protocol Layer.
23		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that an <i>Accept</i> Message has been received. The Policy Engine stops <i>SenderResponseTimer</i> , starts the <i>PSTransitionTimer</i> and reduces its current draw. The Device Policy Manager prepares the Power supply for transition to the new power level.
24		The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.
25	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the Message.
26	Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer. The Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops the <i>CRCReceiveTimer</i> .	
27	The Protocol Layer informs the Policy Engine that an <i>Accept</i> Message was successfully sent.	
<u>power supply Adjusts its Output to the Negotiated Value</u>		
28	The Device Policy Manager informs the Policy Engine that the power supply has settled at the new operating condition and tells the Protocol Layer to send a <i>PS_RDY</i> Message. If the time taken to settle exceeds <i>tSourceActivity</i> then a <i>Ping</i> Message will be sent.	
29	The Protocol Layer forms the <i>PS_RDY</i> Message and starts the <i>CRCReceiveTimer</i> .	
30	Physical Layer appends CRC and sends the <i>PS_RDY</i> Message.	Physical Layer receives the <i>PS_RDY</i> Message and compares the CRC it calculated with the one sent to verify the Message.

Step	Source	Sink
31		Physical Layer forwards the <i>PS_RDY</i> Message to the Protocol Layer.
32		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that a RS_RDY has been received. The Policy Engine stops the <i>PSTransitionTimer</i> and starts the <i>SinkActivityTimer</i> to monitor for <i>Ping</i> Message timeout if required (see Section 6.3.5).
33		The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.
34	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the Message.
35	Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer. The Protocol Layer verifies and increments the <i>MessageIDCounter</i> . Stops the <i>CRCReceiveTimer</i> .	
36	The Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent. The Policy Engine starts the <i>SourceActivityTimer</i> in order to start pinging if required (see Section 6.3.5).	

### 8.3.2.4 Reclaiming Power with GotoMin Message

This is an example of a GotoMin operation. Figure 8-6 shows the Messages as they flow across the bus and within the devices to accomplish the GotoMin.

Figure 8-6 Successful GotoMin operation



The table below provides a detailed explanation of what happens at each labeled step in Figure 8-6 above.

Table 8-5 Steps for a GotoMin Negotiation

Step	Source	Sink
1	Policy Engine tells the Protocol Layer to form a <i>GotoMin</i> Message.	
2	The Protocol Layer forms the <i>GotoMin</i> Message that is passed to the Physical Layer and starts the <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>GotoMin</i> Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer forwards the <i>GotoMin</i> Message to the Protocol Layer.

Step	Source	Sink
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that a <i>GotoMin</i> Message has been received. The Policy starts the <i>PSTransitionTimer</i> and reduces its current draw. The Policy Engine prepares the Power supply for transition to the new power level.
6		The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.
7	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the Message.
8	Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer. The Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops the <i>CRCReceiveTimer</i> .	
9	The Protocol Layer informs the Policy Engine that a <i>GotoMin</i> Message was successfully sent.	
power supply Adjusts its Output to the Negotiated Value		
10	Policy Engine sees the power supply has settled at the new operating condition and tells the Protocol Layer to send a <i>PS_RDY</i> Message. If the time taken to settle exceeds <i>tSourceActivity</i> then a <i>Ping</i> Message will be sent (if required see Section 6.3.5).	
11	The Protocol Layer forms the <i>PS_RDY</i> Message and starts the <i>CRCReceiveTimer</i> .	
12	Physical Layer appends CRC and sends the <i>PS_RDY</i> Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
13		Physical Layer forwards the <i>PS_RDY</i> Message to the Protocol Layer.
14		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that a <i>PS_RDY</i> Message has been received. The Policy Engine stops the <i>PSTransitionTimer</i> and optionally starts the <i>SinkActivityTimer</i> to monitor for <i>Ping</i> Message timeout (if required see Section 6.3.5).
15		The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.
16	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the Message.
17	Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer. The Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops the <i>CRCReceiveTimer</i> .	
18	The Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent. The Policy Engine starts the <i>SourceActivityTimer</i> in order to start ping (if required see Section 6.3.5).	

### 8.3.2.5 Soft Reset

This is an example of a Soft Reset operation. Figure 8-7 shows the Messages as they flow across the bus and within the devices to accomplish the Soft Reset.

Figure 8-7 Soft Reset

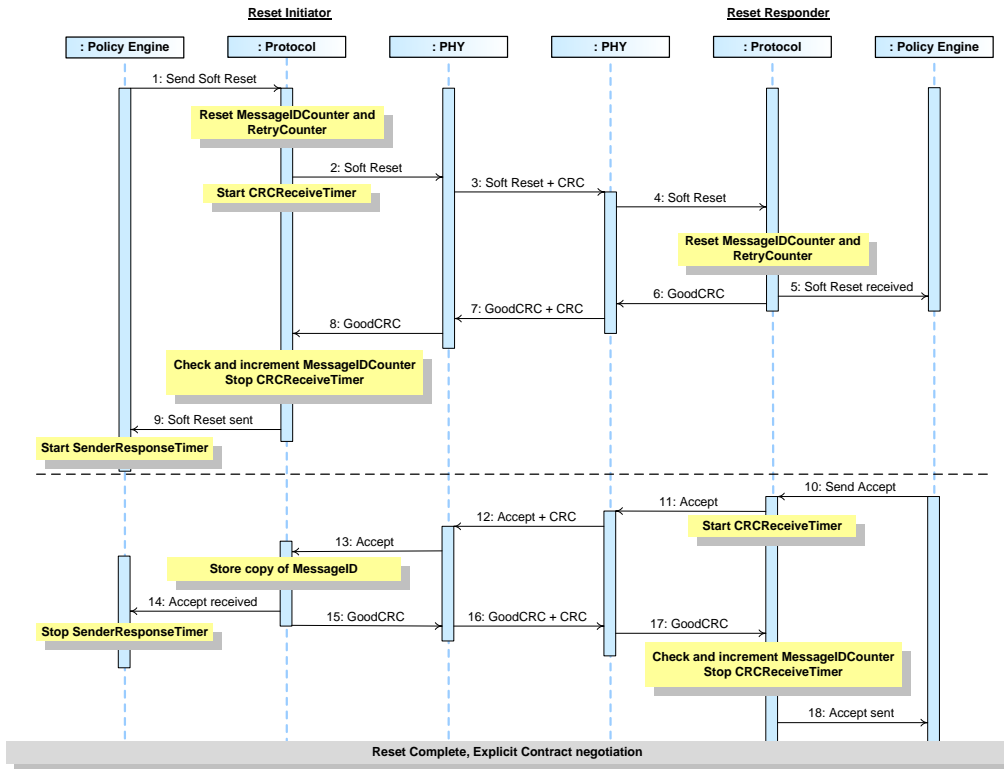


Table 8-6 below provides a detailed explanation of what happens at each labeled step in Figure 8-7 above.

Table 8-6 Steps for a Soft Reset

Step	Reset Initiator	Reset Responder
1	The Policy Engine directs the Protocol Layer to generate a <i>Soft_Reset</i> Message to request a Soft Reset.	
2	Protocol Layer resets <i>MessageIDCounter</i> and <i>RetryCounter</i> . Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>Soft_Reset</i> Message.	Physical Layer receives the <i>Soft_Reset</i> Message and compares the CRC it calculated with the one sent to verify the Message.

Step	Reset Initiator	Reset Responder
4		Physical Layer removes the CRC and forwards the <i>Soft_Reset</i> Message to the Protocol Layer.
5		Protocol Layer does not check the <i>MessageID</i> in the incoming Message and resets <i>MessageIDCounter</i> and <i>RetryCounter</i> . The Protocol Layer forwards the received <i>Soft_Reset</i> Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Soft_Reset</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .	
10		Policy Engine tells the Protocol Layer to form an <i>Accept</i> Message.
11		Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Message.
13	Protocol Layer stores the <i>MessageID</i> of the incoming Message.	
14	The Protocol Layer forwards the received <i>Accept</i> Message information to the Policy Engine that consumes it.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent.
	The reset is complete and protocol communication can restart. <u>Port Partners perform an Explicit Contract negotiation to re-synchronise their state machines.</u>	

### 8.3.2.6 Hard Reset

The following sections describe the steps required for a USB Power Delivery Hard Reset. The Hard Reset returns the operation of the USB Power Delivery to default role and operating voltage/current. During the Hard Reset USB Power Delivery PHY Layer communications shall be disabled preventing communication between the Port partners.

Note: Hard Reset, in this case, is applied to the USB Power Delivery capability of an individual Port on which the Hard Reset is requested. A side effect of the Hard Reset is that it might reset other functions on the Port such as USB.

#### 8.3.2.6.1 ProviderSource Initiated Hard Reset

This is an example of a Hard Reset operation when initiated by a ProviderSource. Figure 8-8 shows the Messages as they flow across the bus and within the devices to accomplish the Hard Reset.



Figure 8-8 **ProviderSource** initiated Hard Reset

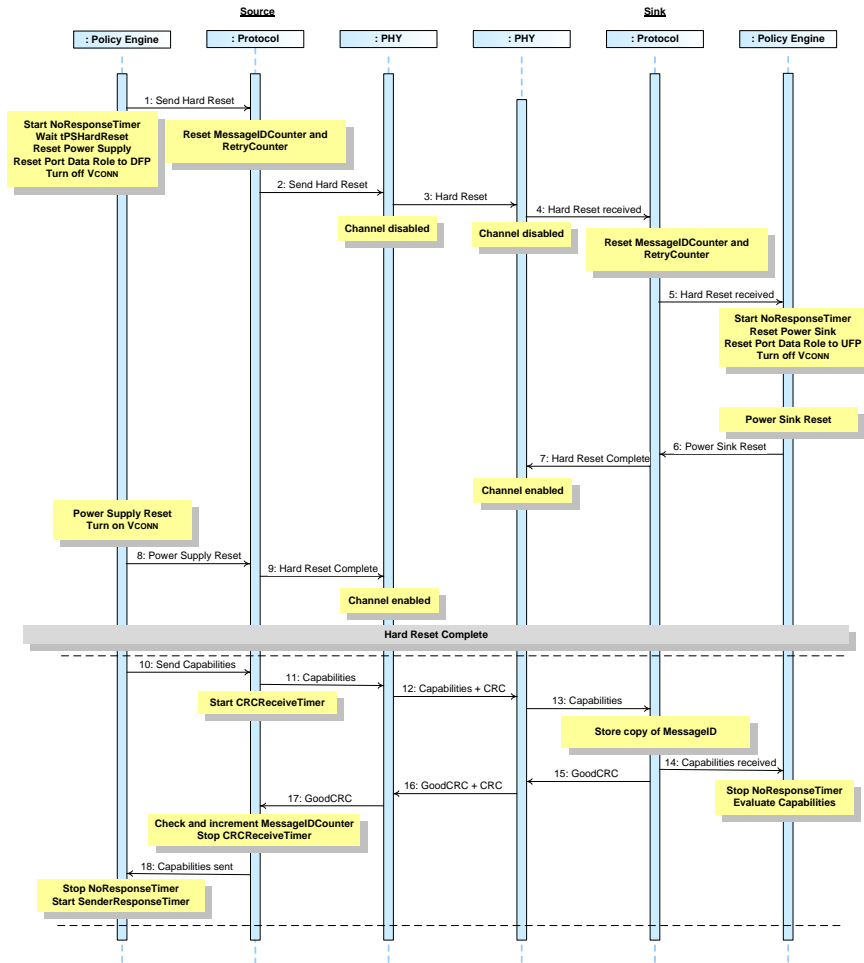


Table 8-7 Steps for Source initiated Hard Reset

Step	Provider:Source	Consumer:Sink
1	The Policy Engine directs the Protocol Layer to generate <b>Hard Reset</b> Signaling. The Policy Engine starts the <b>NoResponseTimer</b> and <b>directs requests</b> the <b>Device Policy Manager to reset the power supply to reset to USB Default Operation. If this is a Type-C connector the Policy Engine requests the Device Policy Manager to reset the Port Data Role to DFP and to turn off VCONN if this is on.</b>	
2	Protocol Layer resets <b>MessageIDCounter</b> and <b>RetryCounter</b> . Protocol Layer requests the Physical Layer send <b>Hard Reset</b> Signaling.	
3	Physical Layer sends <b>Hard Reset</b> Signaling and then disables the PHY Layer communications channel for transmission and reception.	Physical Layer receives the <b>Hard Reset</b> Signaling and disables the PHY Layer communications channel for transmission and reception.
4		Physical Layer informs the Protocol Layer of the Hard Reset. Protocol Layer resets <b>MessageIDCounter</b> and <b>RetryCounter</b> .
5		The Protocol Layer informs the Policy Engine of the Hard Reset. The Policy Engine starts the <b>NoResponseTimer</b> and <b>directs requests</b> the <b>Device Policy Manager to reset the Power Sink to reset to default operation. If this is a Type-C connector the Policy Engine requests the Device Policy Manager to reset the Port Data Role to UFP and to turn off VCONN if this is on.</b>
6		The Power Sink returns to default operation. The Policy Engine informs the Protocol Layer that the Power Sink has been reset.
7		The Protocol Layer informs the PHY Layer that the Hard Reset is complete. The PHY Layer enables the PHY Layer communications channel for transmission and reception.
8	The power supply is reset to default operation. <b>If this is a Type-C connector VCONN is turned on.</b> The Policy Engine informs the Protocol Layer that the power supply has been reset.	
9	The Protocol Layer informs the PHY Layer that the Hard Reset is complete. The PHY Layer enables the PHY Layer communications channel for transmission and reception.	
	The reset is complete and protocol communication can restart.	
10	Policy Engine directs the Protocol Layer to send a <b>Source_Capabilities</b> Message that represents the power supply's present capabilities.	
11	Protocol Layer creates the Message and passes to Physical Layer. Starts <b>CRCReceiveTimer</b> .	
12	Physical Layer appends CRC and sends the <b>Source_Capabilities</b> Message.	Physical Layer receives the <b>Source_Capabilities</b> Message and checks the CRC to verify the Message.

Step	Provider:Source	Consumer:Sink
13		Physical Layer removes the CRC and forwards the <i>Source_Capabilities</i> Message to the Protocol Layer.
14		Protocol Layer stores the <i>MessageID</i> of the incoming Message. The Protocol Layer forwards the received <i>Source_Capabilities</i> Message information to the Policy Engine that consumes it. The Policy Engine stops the <i>NoResponseTimer</i> .
15		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
16	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
17	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Source_Capabilities</i> Message was successfully sent. Policy Engine stops the <i>NoResponseTimer</i> and starts the <i>SenderResponseTimer</i> .	
	USB Power Delivery communication is re-established.	

### 8.3.2.6.2 ConsumerSink Initiated Hard Reset

This is an example of a Hard Reset operation when initiated by a ConsumerSink. Figure 8-9 shows the Messages as they flow across the bus and within the devices to accomplish the Hard Reset.

Figure 8-9 ConsumerSink Initiated Hard Reset

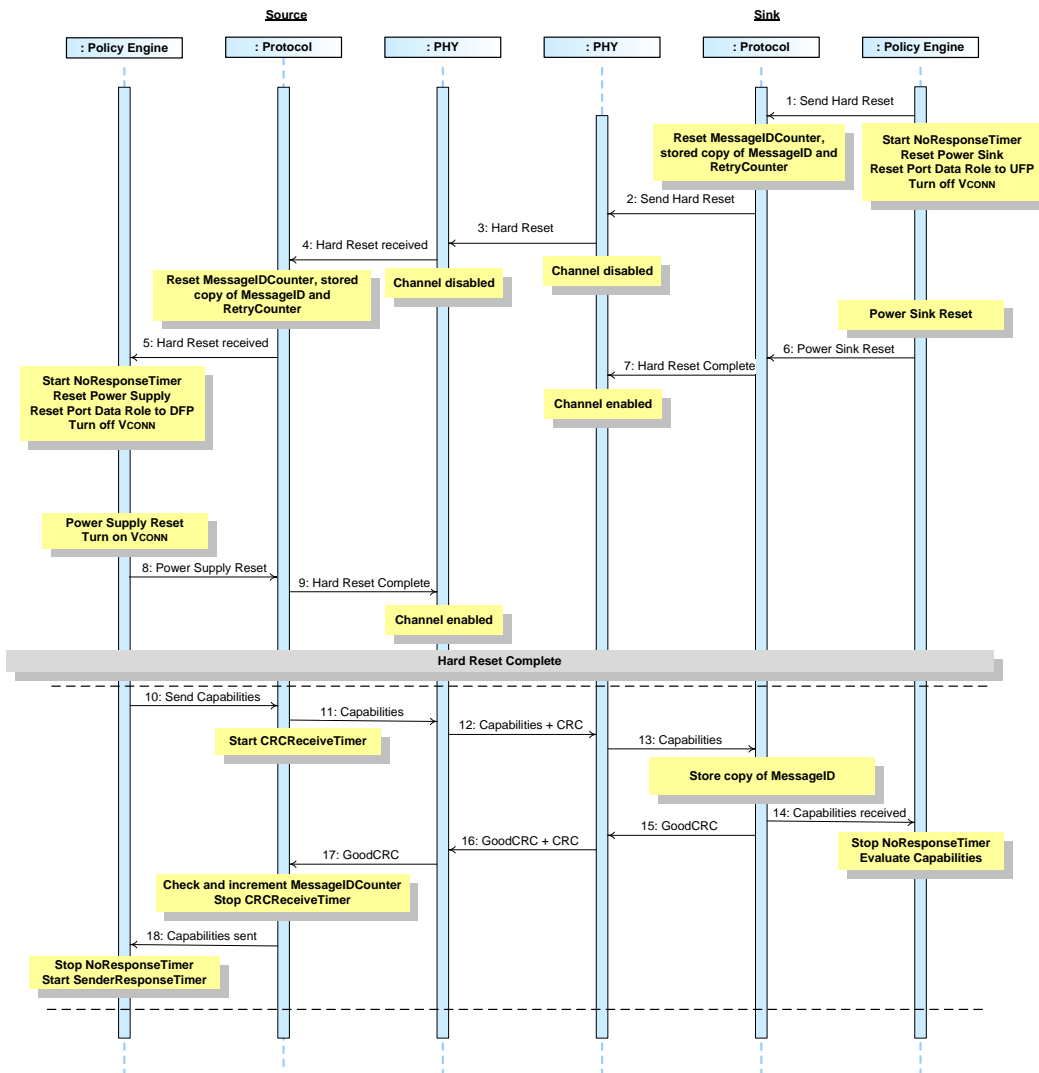


Table 8-8 Steps for Sink initiated Hard Reset

Step	Provider:Source	Consumer:Sink
1		The Policy Engine directs the Protocol Layer to generate <b>Hard Reset</b> Signaling. The Policy Engine starts the <b>NoResponseTimer</b> and <del>directs requests</del> the <b>Power Sink Device Policy Manager</b> to reset <u>the power supply</u> to USB Default Operation. <u>If this is a Type-C connector the Policy Engine requests the Device Policy Manager to reset the Port Data Role to UFP and to turn off VCONN if this is on.</u>
2		Protocol Layer resets <b>MessageIDCounter</b> , stored copy of <b>MessageID</b> and <b>RetryCounter</b> . Protocol Layer requests the Physical Layer send <b>Hard Reset</b> Signaling.
3	Physical Layer receives the <b>Hard Reset</b> Signaling and disables the PHY Layer communications channel for transmission and reception.	Physical Layer sends the <b>Hard Reset</b> Signaling and then disables the PHY Layer communications channel for transmission and reception.
4	Physical Layer informs the Protocol Layer of the Hard Reset. Protocol Layer resets <b>MessageIDCounter</b> , stored copy of <b>MessageID</b> and <b>RetryCounter</b> .	
5	The Protocol Layer Informs the Policy Engine of the Hard Reset. The Policy Engine starts the <b>NoResponseTimer</b> and requests the <del>power supply to reset to USB Default Operation</del> <b>Device Policy Manager to reset the Power Sink to default operation</b> . <u>If this is a Type-C connector the Policy Engine requests the Device Policy Manager to reset the Port Data Role to DFP and to turn off VCONN if this is on.</u>	
6		The Power Sink returns to USB Default Operation. The Policy Engine informs the Protocol Layer that the Power Sink has been reset.
7		The Protocol Layer informs the PHY Layer that the Hard Reset is complete. The PHY Layer enables the PHY Layer communications channel for transmission and reception.
8	The power supply is reset to USB Default Operation. <u>If this is a Type-C connector VCONN is turned on.</u> The Policy Engine informs the Protocol Layer that the power supply has been reset.	
9	The Protocol Layer informs the PHY Layer that the Hard Reset is complete. The PHY Layer enables the PHY Layer communications channel for transmission and reception.	
	The reset is complete and protocol communication can restart.	
10	Policy Engine directs the Protocol Layer to send a <b>Source_Capabilities</b> Message that represents the power supply's present capabilities.	
11	Protocol Layer creates the Message and passes to Physical Layer. Starts <b>CRCReceiveTimer</b> .	
12	Physical Layer appends CRC and sends the <b>Source_Capabilities</b> Message.	Physical Layer receives the <b>Source_Capabilities</b> Message and checks the CRC to verify the Message.

Step	Provider:Source	Consumer:Sink
13		Physical Layer removes the CRC and forwards the <i>Source_Capabilities</i> Message to the Protocol Layer.
14		Protocol Layer stores the <i>MessageID</i> of the incoming Message. The Protocol Layer forwards the received <i>Source_Capabilities</i> Message information to the Policy Engine that consumes it. The Policy Engine stops the <i>NoResponseTimer</i> .
15		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
16	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
17	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Source_Capabilities</i> Message was successfully sent. Policy Engine stops the <i>NoResponseTimer</i> and starts the <i>SenderResponseTimer</i> .	
USB Power Delivery communication is re-established.		

### 8.3.2.6.3 ProviderSource Initiated Hard Reset – ConsumerSink Long Reset

This is an example of a Hard Reset operation when initiated by a **ProviderSource**. In this example the **ConsumerSink** is slow responding to the reset causing the **ProviderSource** to send multiple **Source\_Capabilities** Messages before it receives a **GoodCRC** Message. Figure 8-10 shows the Messages as they flow across the bus and within the devices to accomplish the Hard Reset.

Figure 8-10 **ProviderSource** initiated reset - **ConsumerSink** long reset

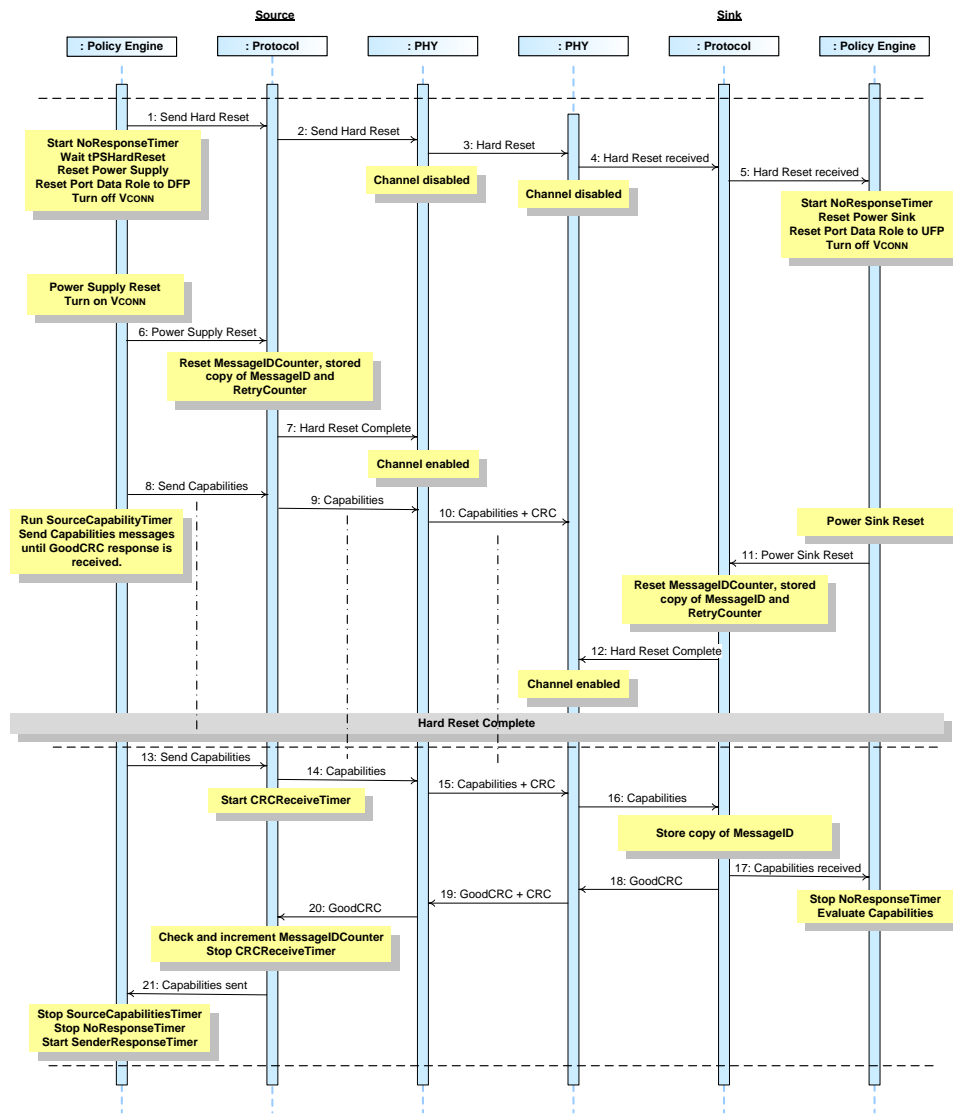


Table 8-9 Steps for Source initiated Hard Reset – Sink long reset

Step	Provider:Source	Consumer:Sink
1	The Policy Engine directs the Protocol Layer to generate <i>Hard Reset</i> Signaling. The Policy Engine starts the <i>NoResponseTimer</i> and <del>directs requests</del> the <i>Device Policy Manager to reset the power supply to reset to USB Default Operation. If this is a Type-C connector the Policy Engine requests the Device Policy Manager to reset the Port Data Role to DFP and to turn off VCONN if this is on.</i>	
2	Protocol Layer resets <i>MessageIDCounter</i> , stored copy of <i>MessageID</i> and <i>RetryCounter</i> . Protocol Layer requests the Physical Layer send <i>Hard Reset</i> Signaling.	
3	Physical Layer sends the <i>Hard Reset</i> Signaling and then disables the PHY Layer communications channel for transmission and reception.	Physical Layer receives the <i>Hard Reset</i> Signaling and disables the PHY Layer communications channel for transmission and reception.
4		Physical Layer informs the Protocol Layer of the Hard Reset. Protocol Layer resets <i>MessageIDCounter</i> , stored copy of <i>MessageID</i> and <i>RetryCounter</i> .
5		The Protocol Layer Informs the Policy Engine of the Hard Reset. The Policy Engine starts the <i>NoResponseTimer</i> and requests the <del>Power Sink to reset to USB Default Operation</del> <i>Device Policy Manager to reset the Power Sink to default operation. If this is a Type-C connector the Policy Engine requests the Device Policy Manager to reset the Port Data Role to UFP and to turn off VCONN if this is on.</i>
6	The power supply is reset to USB Default Operation. <del>If this is a Type-C connector VCONN is turned on.</del> The Policy Engine informs the Protocol Layer that the power supply has been reset.	
7	The Protocol Layer informs the PHY Layer that the Hard Reset is complete. The PHY Layer enables the PHY Layer communications channel for transmission and reception.	
	The reset is complete and protocol communication can restart.	
8	Policy Engine directs the Protocol Layer to send a <i>Source_Capabilities</i> Message that represents the power supply's present capabilities. Policy Engine starts the <i>SourceCapabilityTimer</i> . The <i>SourceCapabilityTimer</i> times out one or more times until a <i>GoodCRC</i> Message response is received.	
9	Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
10	Physical Layer appends CRC and sends the <i>Source_Capabilities</i> Message.	Note: <i>Source_Capabilities</i> Message not received since channel is disabled.
11		The Power Sink returns to USB Default Operation. The Policy Engine informs the Protocol Layer that the Power Sink has been reset.



Step	ProviderSource	ConsumerSink
12		The Protocol Layer informs the PHY Layer that the Hard Reset is complete. The PHY Layer enables the PHY Layer communications channel for transmission and reception.
	The reset is complete and protocol communication can restart.	
13	Policy Engine directs the Protocol Layer to send a <i>Source_Capabilities</i> Message that represents the power supply's present capabilities. Starts the <i>SourceCapabilityTimer</i> .	
14	Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
15	Physical Layer appends CRC and sends the <i>Source_Capabilities</i> Message.	Physical Layer receives the <i>Source_Capabilities</i> Message and checks the CRC to verify the Message.
16		Physical Layer removes the CRC and forwards the <i>Source_Capabilities</i> Message to the Protocol Layer.
17		Protocol Layer stores the <i>MessageID</i> of the incoming Message. The Protocol Layer forwards the received <i>Source_Capabilities</i> Message information to the Policy Engine that consumes it. The Policy Engine stops the <i>NoResponseTimer</i> .
18		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
19	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
20	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
21	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Source_Capabilities</i> Message was successfully sent. Policy Engine stops the <i>SourceCapabilityTimer</i> , stops the <i>NoResponseTimer</i> and starts the <i>SenderResponseTimer</i> .	
	USB Power Delivery communication is re-established.	

### 8.3.2.7 Type-A/B specific Message Sequence Diagrams

#### 8.3.2.7.1 Type-A/B Power Role Swap

##### 8.3.2.7.1.1 Type-A/B Source Initiated Power Role Swap without subsequent Power Negotiation

This is an example of a successful Power Role Swap operation initiated by a Type-A or Type-B Dual-Role Port which initially, at the start of this Message sequence, is acting as a Source. It does not include any subsequent Power Negotiation which is required in order to establish an Explicit Contract (see previous section for the details of a Power Negotiation).

There are four distinct phases to the Power Role Swap negotiation:

1. A *PR\_Swap* Message is sent.
2. An *Accept* Message in response to the *PR\_Swap* Message.
3. The original Source sets its power output to *vSafe0V* and sends a *PS\_RDY* Message when it gets there.
4. The new Source then sets its power output to *vSafe5V* and sends a *PS\_RDY* Message when it is ready to supply power.

Figure 8-11 shows the Messages as they flow across the bus and within the devices to accomplish the Power Role Swap sequence.

Figure 8-11 Type-A or Type-B Successful Power Role Swap Sequence Initiated by the Source

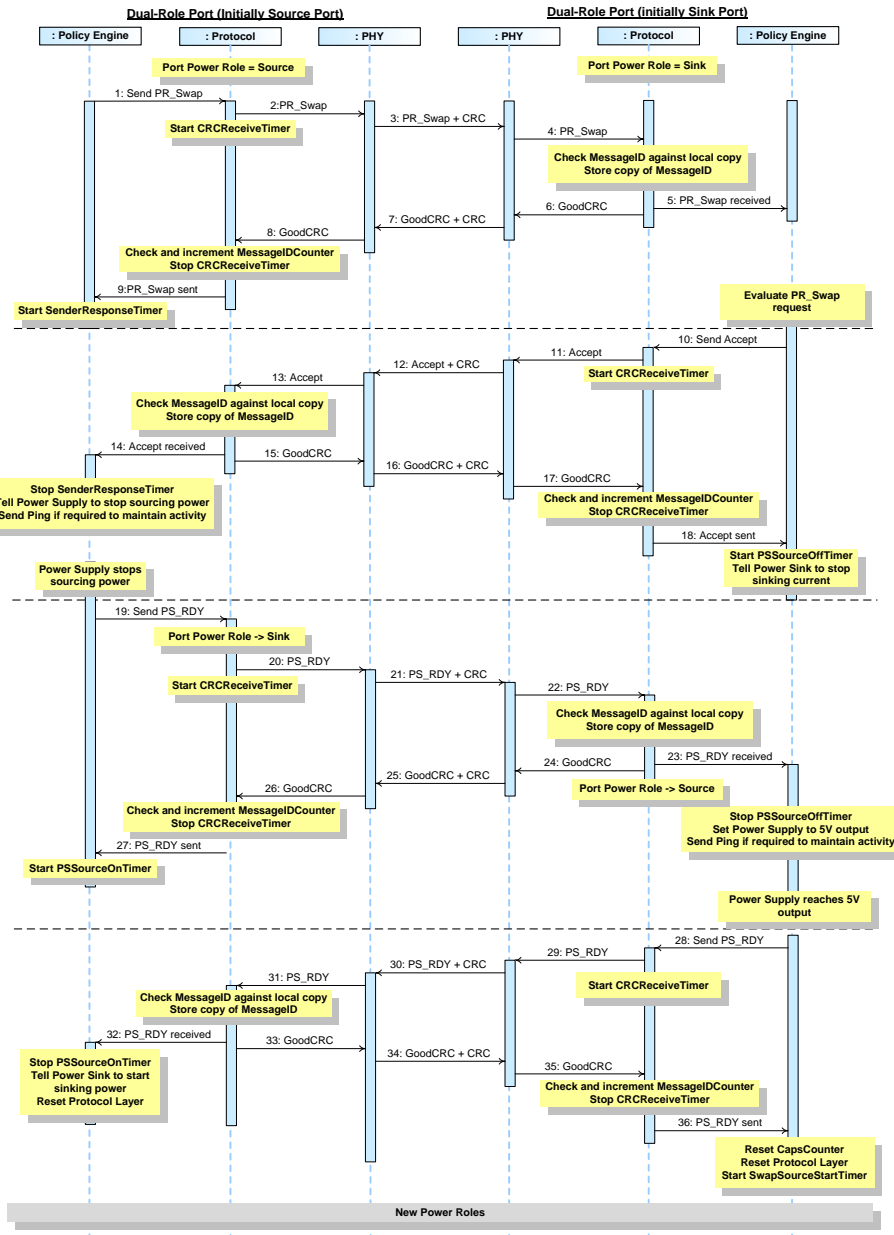


Table 8-10 below provides a detailed explanation of what happens at each labeled step in Figure 8-11 above.

Table 8-10 Steps for a Successful Type-A or Type-B Source Initiated Power Role Swap Sequence

Step	Dual-RolePort (initially Source Port)	Dual-Role Port (initially Sink Port)
1	<i>Port Power Role</i> starts as Source. Policy Engine directs the Protocol Layer to send a <i>PR_Swap</i> Message.	<i>Port Power Role</i> starts as Sink.
2	Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>PR_Swap</i> Message.	Physical Layer receives the <i>PR_Swap</i> Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>PR_Swap</i> Message to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PR_Swap</i> Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PR_Swap</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .	
10		Policy Engine evaluates the <i>PR_Swap</i> Message sent by the Source and decides that it is able and willing to do the Power Role Swap. It tells the Protocol Layer to form an <i>Accept</i> Message.
11		Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the <i>Accept</i> Message.	Physical Layer appends a CRC and sends the <i>Accept</i> Message.
13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PR_Swap</i> Message information to the Policy Engine that consumes it.	
14	The Policy Engine requests its power supply to stop supplying power and stops the <i>SenderResponseTimer</i> . If the time taken to stop supplying power exceeds <i>tSourceActivity</i> then a <i>Ping</i> Message will be sent (if required see Section 6.3.5).	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.

Step	Dual-RolePort (initially Source Port)	Dual-Role Port (initially Sink Port)
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent. The Policy Engine starts the <i>PSSourceOffTimer</i> and tells the power supply to stop sinking current.
19	The Policy Engine determines its power supply is no longer supplying $V_{BUS}$ , it directs the Protocol Layer to generate a <i>PS_RDY</i> Message, with the <i>Port Power Role</i> bit in the <i>Message</i> Header set to "Sink", to tell its Port Partner that it can begin to Source $V_{BUS}$ .	
20	Protocol Layer sets the <i>Port Power Role</i> bit in the <i>Message</i> Header set to "Sink", creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
21	Physical Layer appends CRC and sends the <i>PS_RDY</i> Message.	Physical Layer receives the <i>PS_RDY</i> Message and checks the CRC to verify the Message.
22		Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.
23		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PS_RDY</i> Message information to the Policy Engine that consumes it. The Policy Engine stops the <i>PSSourceOffTimer</i> and starts switching the power supply to <i>vSafe5V</i> Source operation. If the time taken to start supplying power exceeds <i>tSourceActivity</i> then a <i>Ping</i> Message will be sent (if required see Section 6.3.5).
24		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
25	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
26	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
27	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent. Policy Engine starts <i>PSSourceOnTimer</i> .	
28		Policy Engine, when its power supply is ready to supply power, tells the Protocol Layer to form a <i>PS_RDY</i> Message. The <i>Port Power Role</i> bit used in this and subsequent Message Headers is now set to "Source".
29		Protocol Layer creates the <i>PS_RDY</i> Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
30	Physical Layer receives the <i>PS_RDY</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>PS_RDY</i> Message.
31	Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.	

Step	Dual-RolePort (initially Source Port)	Dual-Role Port (initially Sink Port)
32	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PS_RDY</i> Message information to the Policy Engine that consumes it.	
33	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
34	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.. The Policy Engine stops the <i>PSSourceOnTimer</i> , informs the power supply it can now Sink power and resets the Protocol Layer.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
35		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
36		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent. The Policy Engine resets the <i>CapsCounter</i> , resets the Protocol Layer and starts the <i>SwapSourceStartTimer</i> which must timeout before sending any <i>Source_Capabilities</i> Messages.
	The Power Role Swap is complete, the roles have been reversed and the Port Partners are free to negotiate for more power.	

#### 8.3.2.7.1.2 Type-A/B Sink Initiated Power Role Swap without subsequent Power Negotiation

This is an example of a successful Power Role Swap operation initiated by a Type-A or Type-B Dual-Role Port which initially, at the start of this Message sequence, is acting as a Sink. It does not include any subsequent Power Negotiation which is required in order to establish an Explicit Contract (see Section 8.3.2.3).

There are four distinct phases to the Power Role Swap negotiation:

- 1) A *PR\_Swap* Message is sent.
- 2) An *Accept* Message in response to the *PR\_Swap* Message.
- 3) The original Source sets its power output to *vSafe0V* and sends a *PS\_RDY* Message when it gets there.
- 4) The new Source then sets its power output to *vSafe5V* and sends a *PS\_RDY* Message when it is ready to supply power.

Figure 8-12 shows the Messages as they flow across the bus and within the devices to accomplish the Power Role Swap.

Figure 8-12 Type-A or Type-B Successful Power Role Swap Sequence Initiated by the Sink

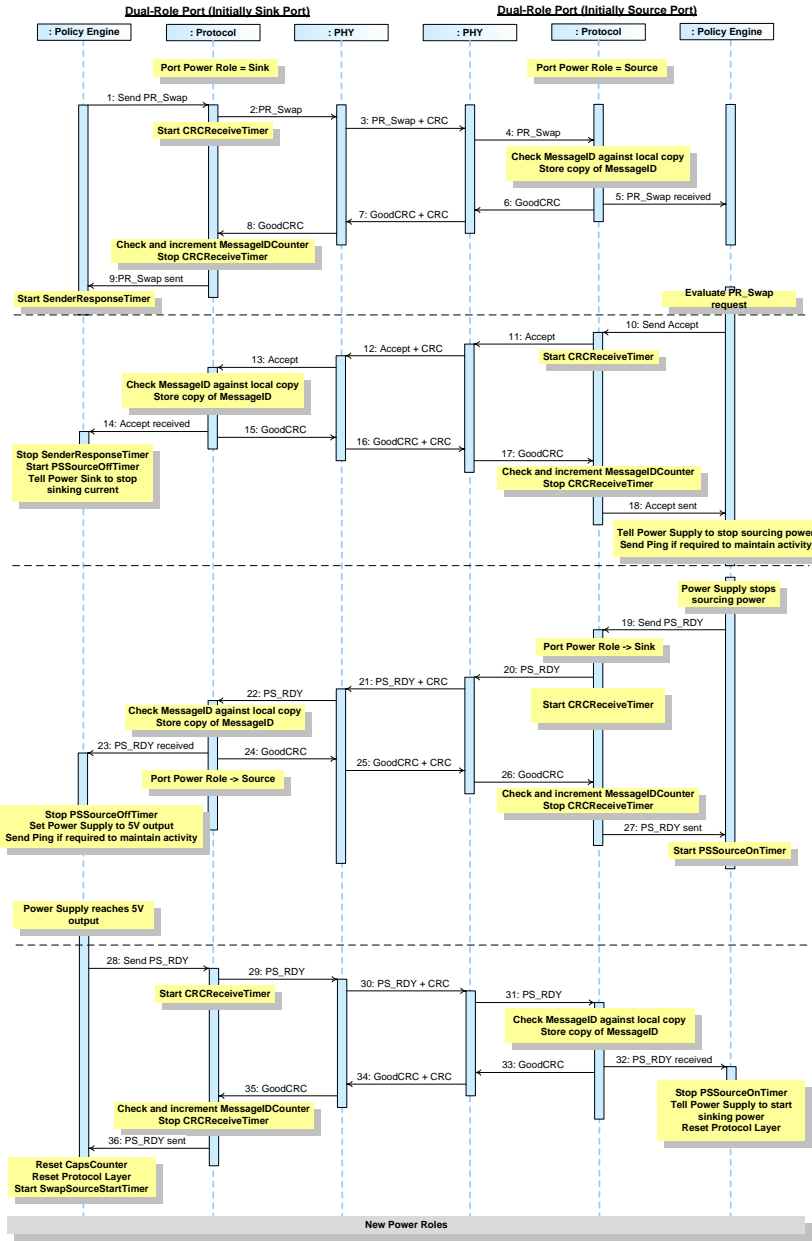


Table 8-11 Steps for a Successful Type-A or Type-B Sink Initiated Power Role Swap Sequence

Step	Dual-Role Port (initially Sink Port)	Dual-Role Port (initially Source Port)
1	<i>Port Power Role</i> starts as Sink. Policy Engine directs the Protocol Layer to send a <i>PR_Swap</i> Message.	<i>Port Power Role</i> starts as Source.
2	Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>PR_Swap</i> Message.	Physical Layer receives the <i>PR_Swap</i> Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>PR_Swap</i> Message to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PR_Swap</i> Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PR_Swap</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .	
10		Policy Engine evaluates the <i>PR_Swap</i> Message sent by the Sink and decides that it is able and willing to do the Power Role Swap. It tells the Protocol Layer to form an <i>Accept</i> Message.
11		Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the <i>Accept</i> Message.	Physical Layer appends a CRC and sends the <i>Accept</i> Message.
13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PR_Swap</i> Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>SenderResponseTimer</i> , starts the <i>PSSourceOffTimer</i> and tells the power supply to stop sinking current.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.



Step	Dual-Role Port (initially Sink Port)	Dual-Role Port (initially Source Port)
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent. The Policy Engine tells the power supply to stop supplying power. If the time taken to stop supplying power exceeds <i>tSourceActivity</i> then a <i>Ping</i> Message will be sent (if required see Section 6.3.5).
19		The Policy Engine determines its power supply is no longer supplying $V_{BUS}$ , it directs the Protocol Layer to generate a <i>PS_RDY</i> Message, with the <i>Port Power Role</i> bit in the <i>Message</i> Header set to "Sink", to tell its Port Partner that it can begin to source $V_{BUS}$ .
20		Protocol Layer sets the <i>Port Power Role</i> bit in the <i>Message</i> Header set to "Sink", creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
21	Physical Layer receives the <i>PS_RDY</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>PS_RDY</i> Message.
22	Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.	
23	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PS_RDY</i> Message information to the Policy Engine that consumes it. The Policy Engine stops the <i>PSSourceOffTimer</i> and starts switching the power supply to <i>vSafe5V</i> Source operation. If the time taken to start supplying power exceeds <i>tSourceActivity</i> then a <i>Ping</i> Message will be sent (if required see Section 6.3.5).	
24	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
25	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.
26		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
27		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent. Policy Engine starts <i>PSSourceOnTimer</i> .
28	Policy Engine, when its power supply is ready to supply power, tells the Protocol Layer to form a <i>PS_RDY</i> Message. The <i>Port Power Role</i> bit used in this and subsequent Message Headers is now set to "Source".	
29	Protocol Layer creates the <i>PS_RDY</i> Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
30	Physical Layer appends a CRC and sends the <i>PS_RDY</i> Message.	Physical Layer receives the <i>PS_RDY</i> Message and compares the CRC it calculated with the one sent to verify the Message.

Step	Dual-Role Port (initially Sink Port)	Dual-Role Port (initially Source Port)
31		Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.
32		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PS_RDY</i> Message information to the Policy Engine that consumes it. The Policy Engine stops the <i>PSSourceOnTimer</i> , informs the power supply that it can start consuming power and optionally starts the <i>SinkActivityTimer</i> to check for <i>Ping</i> Messages.
33		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
34	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message. The Policy Engine stops the <i>PSSourceOnTimer</i> , informs the power supply it can now Sink power and resets the Protocol Layer.
35	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> to the Protocol Layer.	
36	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent. The Policy Engine resets the <i>CapsCounter</i> , resets the Protocol Layer and starts the <i>SwapSourceStartTimer</i> which must timeout before sending any <i>Source_Capabilities</i> Messages.	
	The Power Role Swap is complete, the roles have been reversed and the Port Partners are free to negotiate for more power.	

### 8.3.2.7.1.3 Type-A/B Source Initiated Hard Reset (Power Role Swapped)

This is an example of a Hard Reset operation when initiated by a Type-A or Type-B Dual-Role device which is currently Power Role Swapped and initially acting as a Source at the start of this Message sequence. In this example both Dual-Role devices return to their original roles after the Hard Reset. Figure 8-13 shows the Messages as they flow across the bus and within the devices to accomplish the Hard Reset.

Figure 8-13 Type-A or Type-B Source initiated Hard Reset (Power Role Swapped)

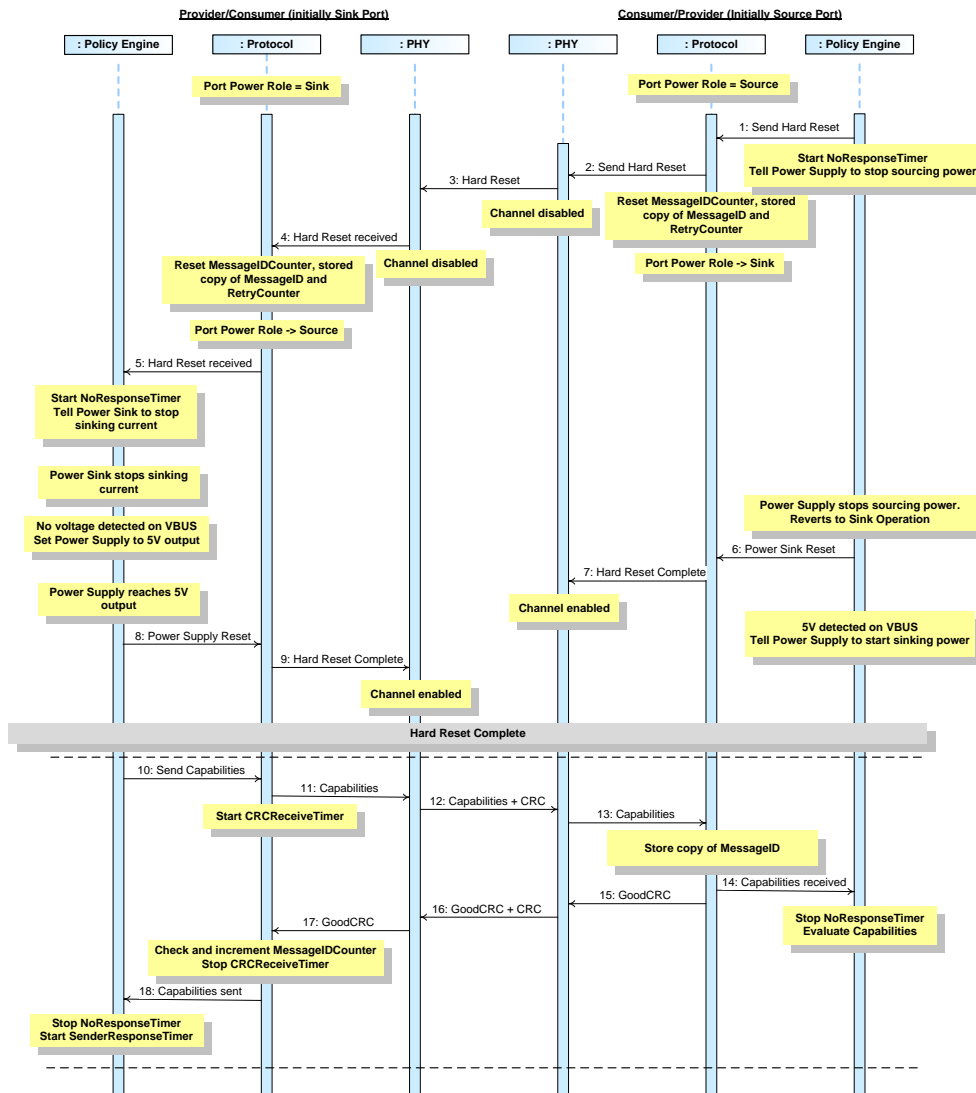


Table 8-12 Steps for Type-A or Type-B Source initiated Hard Reset (Power Role Swapped)

Step	Provider/Consumer (InitiallySink Port)	Consumer/Provider (InitiallySource Port)
1	<i>Port Power Role</i> starts as Sink.	<i>Port Power Role</i> starts as Source. The Policy Engine directs the Protocol Layer to generate <i>Hard Reset</i> Signaling. The Policy Engine starts the <i>NoResponseTimer</i> and directs the power supply to stop sourcing power.
2		Protocol Layer resets <i>MessageIDCounter</i> , stored copy of <i>MessageID</i> and <i>RetryCounter</i> . <i>Port Power Role</i> reverts to Sink. Protocol Layer requests the Physical Layer send <i>Hard Reset</i> Signaling.
3	Physical Layer receives the <i>Hard Reset</i> Signaling and disables the PHY Layer communications channel for transmission and reception.	Physical Layer sends the <i>Hard Reset</i> Signaling and then disables the PHY Layer communications channel for transmission and reception.
4	Physical Layer informs the Protocol Layer of the Hard Reset. Protocol Layer resets <i>MessageIDCounter</i> , stored copy of <i>MessageID</i> and <i>RetryCounter</i> . <i>Port Power Role</i> reverts to Source.	
5	The Protocol Layer Informs the Policy Engine of the Hard Reset. The Policy Engine starts the <i>NoResponseTimer</i> and directs the Sink to stop sinking power.	
6		The power supply stops sourcing power on $V_{BUS}$ and reverts to Sink operation. The Policy Engine informs the Protocol Layer that the Sink has been reset.
7		The Protocol Layer informs the PHY Layer that the Hard Reset is complete. The PHY Layer enables the PHY Layer communications channel for transmission and reception. When <i>vSafe5V</i> is detected on $V_{BUS}$ the Policy Engine directs the Sink to start Sinking power.
8	The Sink stops sinking power. When there is no voltage detected on $V_{BUS}$ the Policy Engine requests the Source to go to <i>vSafe5V</i> output. When the Source reaches <i>vSafe5V</i> the Policy Engine informs the Protocol Layer that the power supply has been reset.	
9	The Protocol Layer informs the PHY Layer that the Hard Reset is complete. The PHY Layer enables the PHY Layer communications channel for transmission and reception.	
The reset is complete and protocol communication can restart.		
10	Policy Engine directs the Protocol Layer to send a <i>Source_Capabilities</i> Message that represents the power supply's present capabilities.	
11	Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
12	Physical Layer appends CRC and sends the <i>Source_Capabilities</i> Message.	Physical Layer receives the <i>Source_Capabilities</i> Message and checks the CRC to verify the Message.

Step	Provider/Consumer (InitiallySink Port)	Consumer/Provider (InitiallySource Port)
13		Physical Layer removes the CRC and forwards the <i>Source_Capabilities</i> Message to the Protocol Layer.
14		Protocol Layer stores the <i>MessageID</i> of the incoming Message. The Protocol Layer forwards the received <i>Source_Capabilities</i> Message information to the Policy Engine that consumes it. The Policy Engine stops the <i>NoResponseTimer</i> .
15		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
16	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
17	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Source_Capabilities</i> Message was successfully sent. Policy Engine stops the <i>NoResponseTimer</i> and starts the <i>SenderResponseTimer</i> .	
USB Power Delivery communication is re-established.		

### 8.3.2.7.1.4 Type-A/B Sink Initiated Hard Reset (Power Role Swapped)

This is an example of a Hard Reset operation when initiated by a Type-A or Type-B Dual-Role device which is currently Power Role Swapped and initially acting as a Sink at the start of this Message sequence. In this example both Dual-Role devices revert to their original roles after the Hard Reset. Figure 8-14 shows the Messages as they flow across the bus and within the devices to accomplish the Hard Reset.

Figure 8-14 Type-A or Type-B Sink Initiated Hard Reset (Power Role Swapped)

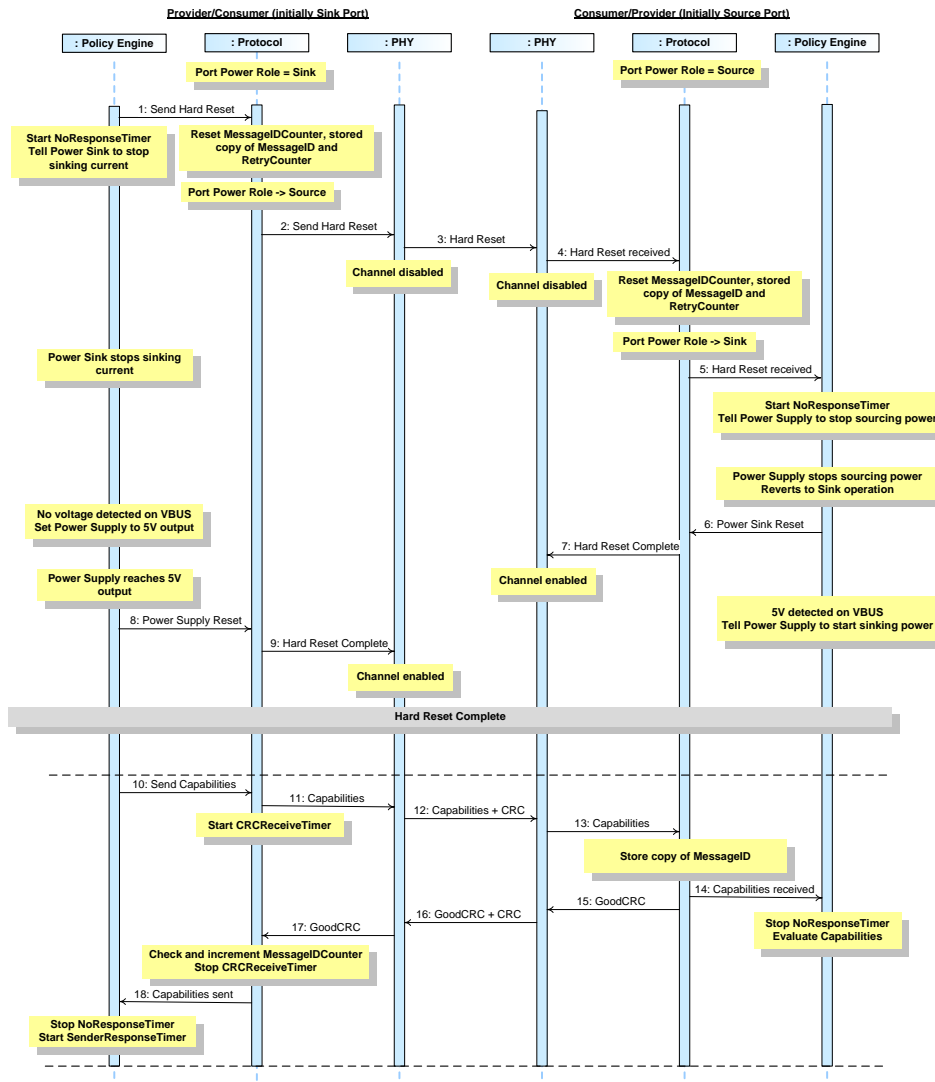




Table 8-13 Steps for Type-A or Type-B Sink initiated Hard Reset (Power Role Swapped)

Step	Provider/Consumer (InitiallySink Port)	Consumer/Provider (InitiallySource Port)
1	<i>Port Power Role</i> starts as Sink. The Policy Engine directs the Protocol Layer to generate <i>Hard Reset</i> Signaling. The Policy Engine starts the <i>NoResponseTimer</i> and directs the Sink to stop sinking power.	<i>Port Power Role</i> starts as Source.
2	Protocol Layer resets <i>MessageIDCounter</i> , stored copy of <i>MessageID</i> and <i>RetryCounter</i> . <i>Port Power Role</i> reverts to Source. Protocol Layer requests the Physical Layer send <i>Hard Reset</i> Signaling.	
3	Physical Layer sends the <i>Hard Reset</i> Signaling and then disables the PHY Layer communications channel for transmission and reception.	Physical Layer receives the <i>Hard Reset</i> Signaling and disables the PHY Layer communications channel for transmission and reception.
4		Physical Layer informs the Protocol Layer of the Hard Reset. Protocol Layer resets <i>MessageIDCounter</i> , stored copy of <i>MessageID</i> and <i>RetryCounter</i> . <i>Port Power Role</i> reverts to Sink.
5		The Protocol Layer Informs the Policy Engine of the Hard Reset. The Policy Engine starts the <i>NoResponseTimer</i> and directs the Source to stop sourcing power.
6		The power supply stops sourcing power on $V_{BUS}$ and reverts to Sink operation. The Policy Engine informs the Protocol Layer that the Sink has been reset.
7		The Protocol Layer informs the PHY Layer that the Hard Reset is complete. The PHY Layer enables the PHY Layer communications channel for transmission and reception.
8	The Sink stops sinking power. When there is no voltage detected on $V_{BUS}$ the Policy Engine requests the Source to go to <i>vSafe5V</i> output. When the Source reaches <i>vSafe5V</i> the Policy Engine informs the Protocol Layer that the power supply has been reset.	When <i>vSafe5V</i> is detected on $V_{BUS}$ the Policy Engine directs the Sink to start Sinking power.
9	The Protocol Layer informs the PHY Layer that the Hard Reset is complete. The PHY Layer enables the PHY Layer communications channel for transmission and reception.	
	The reset is complete and protocol communication can restart.	
10	Policy Engine directs the Protocol Layer to send a <i>Source_Capabilities</i> Message that represents the power supply's present capabilities.	
11	Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
12	Physical Layer appends CRC and sends the <i>Source_Capabilities</i> Message.	Physical Layer receives the <i>Source_Capabilities</i> Message and checks the CRC to verify the Message.
13		Physical Layer removes the CRC and forwards the <i>Source_Capabilities</i> Message to the Protocol Layer.



Step	Provider/Consumer (InitiallySink Port)	Consumer/Provider (InitiallySource Port)
14		Protocol Layer stores the <i>MessageID</i> of the incoming Message. The Protocol Layer forwards the received <i>Source_Capabilities</i> Message information to the Policy Engine that consumes it. The Policy Engine stops the <i>NoResponseTimer</i> .
15		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
16	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
17	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Source_Capabilities</i> Message was successfully sent. Policy Engine stops the <i>NoResponseTimer</i> and starts the <i>SenderResponseTimer</i> .	
	USB Power Delivery communication is re-established.	

### 8.3.2.8 Type-C specific Message Sequence Diagrams

#### 8.3.2.8.1 Type-C Power Role Swap

##### 8.3.2.8.1.1 Type-C Source Initiated Power Role Swap without subsequent Power Negotiation

This is an example of a successful Power Role Swap operation initiated by a [USBType-C 1.0] Port which initially, at the start of this Message sequence, is acting as a Source and therefore has  $R_p$  pulled up on its CC wire. It does not include any subsequent Power Negotiation which is required in order to establish an Explicit Contract (see previous section for the details of a Power Negotiation).

There are four distinct phases to the Power Role Swap negotiation:

1. A *PR\_Swap* Message is sent.
2. An *Accept* Message in response to the *PR\_Swap* Message.
3. The original Source sets its power output to *vSafe0V*, then asserts  $R_d$  and sends a *PS\_RDY* Message when this process is complete.
4. The new Source asserts  $R_p$ , then sets its power output to *vSafe5V* and sends a *PS\_RDY* Message when it is ready to supply power.

Figure 8-15 shows the Messages as they flow across the bus and within the devices to accomplish the Power Role Swap sequence.

Figure 8-15 Type-C Successful Power Role Swap Sequence Initiated by the Source

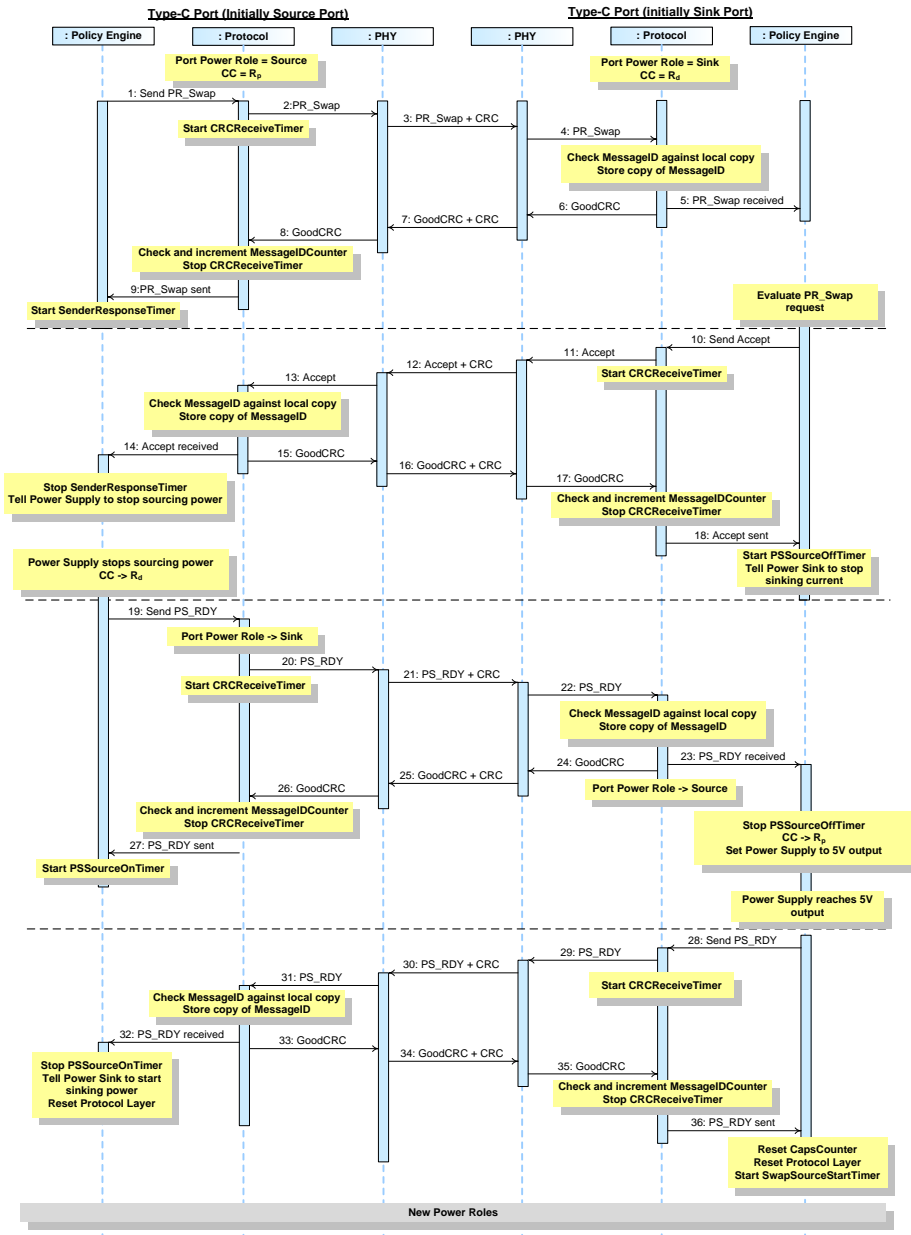


Table 8-10 below provides a detailed explanation of what happens at each labeled step in Figure 8-15 above.

Table 8-14 Steps for a Successful Type-C Source Initiated Power Role Swap Sequence

Step	Type-C Port (initially Source Port)	Type-C Port (initially Sink Port)
1	The [USBType-C1.0] Port has <i>Port Power Role</i> set to Source and the $R_p$ pull up on its CC wire. Policy Engine directs the Protocol Layer to send a <i>PR_Swap</i> Message.	The [USBType-C1.0] Port has <i>Port Power Role</i> set to Sink with the $R_d$ pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>PR_Swap</i> Message.	Physical Layer receives the <i>PR_Swap</i> Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>PR_Swap</i> Message to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PR_Swap</i> Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PR_Swap</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .	
10		Policy Engine evaluates the <i>PR_Swap</i> Message sent by the Source and decides that it is able and willing to do the Power Role Swap. It tells the Protocol Layer to form an <i>Accept</i> Message.
11		Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the <i>Accept</i> Message.	Physical Layer appends a CRC and sends the <i>Accept</i> Message.
13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PR_Swap</i> Message information to the Policy Engine that consumes it.	
14	The Policy Engine requests its power supply to stop supplying power and stops the <i>SenderResponseTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.

Step	Type-C Port (initially Source Port)	Type-C Port (initially Sink Port)
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent. The Policy Engine starts the <i>PSSourceOffTimer</i> and tells the power supply to stop sinking current.
19	The Policy Engine determines its power supply is no longer supplying $V_{BUS}$ . The Policy Engine requests the Device Policy Manager to assert the $R_d$ pull down on the CC wire. The Policy Engine then directs the Protocol Layer to generate a <i>PS_RDY</i> Message, with the <i>Port Power Role</i> bit in the <i>Message</i> Header set to "Sink", to tell its Port Partner that it can begin to Source $V_{BUS}$ .	
20	Protocol Layer sets the <i>Port Power Role</i> bit in the <i>Message</i> Header set to "Sink", creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
21	Physical Layer appends CRC and sends the <i>PS_RDY</i> Message.	Physical Layer receives the <i>PS_RDY</i> Message and checks the CRC to verify the Message.
22		Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.
23		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PS_RDY</i> Message information to the Policy Engine that consumes it. The Policy Engine stops the <i>PSSourceOffTimer</i> , directs the Device Policy Manager to apply the $R_p$ pull up and then starts switching the power supply to <i>vSafe5V</i> Source operation.
24		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
25	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
26	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
27	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent. Policy Engine starts <i>PSSourceOnTimer</i> .	
28		Policy Engine, when its power supply is ready to supply power, tells the Protocol Layer to form a <i>PS_RDY</i> Message. The <i>Port Power Role</i> bit used in this and subsequent Message Headers is now set to "Source".
29		Protocol Layer creates the <i>PS_RDY</i> Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
30	Physical Layer receives the <i>PS_RDY</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>PS_RDY</i> Message.
31	Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.	

Step	Type-C Port (initially Source Port)	Type-C Port (initially Sink Port)
32	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PS_RDY</i> Message information to the Policy Engine that consumes it.	
33	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
34	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.. The Policy Engine stops the <i>PSSourceOnTimer</i> , informs the power supply it can now Sink power and resets the Protocol Layer.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
35		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
36		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent. The Policy Engine resets the <i>CapsCounter</i> , resets the Protocol Layer and starts the <i>SwapSourceStartTimer</i> which must timeout before sending any <i>Source_Capabilities</i> Messages.
	The Power Role Swap is complete, the roles have been reversed and the Port Partners are free to negotiate for more power.	

#### 8.3.2.8.1.2 Type-C Sink Initiated Power Role Swap without subsequent Power Negotiation

This is an example of a successful Power Role Swap operation initiated by a [USBType-C 1.0] Port which initially, at the start of this Message sequence, is acting as a Sink and therefore has  $R_d$  pulled down its CC wire. It does not include any subsequent Power Negotiation which is required in order to establish an Explicit Contract (see Section 8.3.2.3).

There are four distinct phases to the Power Role Swap negotiation:

- 5) A *PR\_Swap* Message is sent.
- 6) An *Accept* Message in response to the *PR\_Swap* Message.
- 7) The original Source sets its power output to *vSafe0V*, then asserts  $R_d$  and sends a *PS\_RDY* Message when this process is complete.
- 8) The new Source asserts  $R_p$ , then sets its power output to *vSafe5V* and sends a *PS\_RDY* Message when it is ready to supply power.

Figure 8-16 shows the Messages as they flow across the bus and within the devices to accomplish the Power Role Swap.

Figure 8-16 Type-C Successful Power Role Swap Sequence Initiated by the Type-C Sink

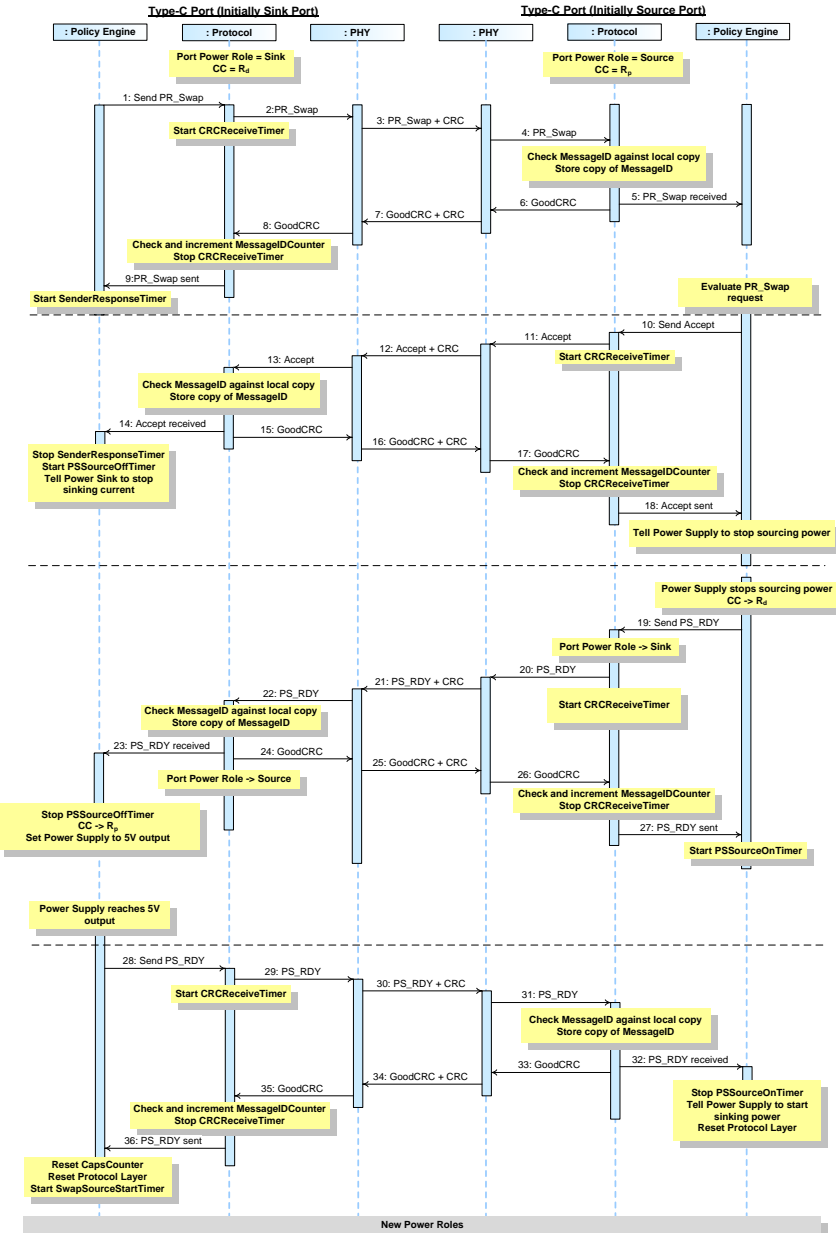




Table 8-15 Steps for a Successful Type-C Sink Initiated Power Role Swap Sequence

Step	Type-C Port (initially Sink Port)	Type-C Port (initially Source Port)
1	The [USBType-C1.0] Port has <i>Port Power Role</i> set to Sink with the R <sub>A</sub> pull down on its CC wire. Policy Engine directs the Protocol Layer to send a <i>PR_Swap</i> Message.	The [USBType-C1.0] Port has <i>Port Power Role</i> set to Source and the R <sub>P</sub> pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>PR_Swap</i> Message.	Physical Layer receives the <i>PR_Swap</i> Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>PR_Swap</i> Message to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PR_Swap</i> Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PR_Swap</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .	
10		Policy Engine evaluates the <i>PR_Swap</i> Message sent by the Sink and decides that it is able and willing to do the Power Role Swap. It tells the Protocol Layer to form an <i>Accept</i> Message.
11		Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the <i>Accept</i> Message.	Physical Layer appends a CRC and sends the <i>Accept</i> Message.
13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PR_Swap</i> Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>SenderResponseTimer</i> , starts the <i>PSSourceOffTimer</i> and tells the power supply to stop sinking current.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.

Step	Type-C Port (initially Sink Port)	Type-C Port (initially Source Port)
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent. The Policy Engine tells the power supply to stop supplying power.
19		The Policy Engine determines its power supply is no longer supplying $V_{BUS}$ . The Policy Engine requests the Device Policy Manager to assert the $R_d$ pull down on the CC wire. The Policy Engine then directs the Protocol Layer to generate a <i>PS_RDY</i> Message, with the <i>Port Power Role</i> bit in the <i>Message</i> Header set to "Sink", to tell its Port Partner that it can begin to Source $V_{BUS}$ .
20		Protocol Layer sets the <i>Port Power Role</i> bit in the <i>Message</i> Header set to "Sink", creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
21	Physical Layer receives the <i>PS_RDY</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>PS_RDY</i> Message.
22	Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.	
23	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PS_RDY</i> Message information to the Policy Engine that consumes it. The Policy Engine stops the <i>PSSourceOffTimer</i> , directs the Device Policy Manager to apply the $R_p$ pull up and then starts switching the power supply to <i>vSafe5V</i> Source operation.	
24	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
25	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.
26		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
27		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent. Policy Engine starts <i>PSSourceOnTimer</i> .
28	Policy Engine, when its power supply is ready to supply power, tells the Protocol Layer to form a <i>PS_RDY</i> Message. The <i>Port Power Role</i> bit used in this and subsequent Message Headers is now set to "Source".	
29	Protocol Layer creates the <i>PS_RDY</i> Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
30	Physical Layer appends a CRC and sends the <i>PS_RDY</i> Message.	Physical Layer receives the <i>PS_RDY</i> Message and compares the CRC it calculated with the one sent to verify the Message.

Step	Type-C Port (initially Sink Port)	Type-C Port (initially Source Port)
31		Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.
32		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PS_RDY</i> Message information to the Policy Engine that consumes it. The Policy Engine stops the <i>PSSourceOnTimer</i> , informs the power supply that it can start consuming power.
33		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
34	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message. The Policy Engine stops the <i>PSSourceOnTimer</i> , informs the power supply it can now Sink power and resets the Protocol Layer.
35	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> to the Protocol Layer.	
36	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent. The Policy Engine resets the <i>CapsCounter</i> , resets the Protocol Layer and starts the <i>SwapSourceStartTimer</i> which must timeout before sending any <i>Source_Capabilities</i> Messages.	
	The Power Role Swap is complete, the roles have been reversed and the Port Partners are free to negotiate for more power.	

### 8.3.2.8.2 Type-C Data Role Swap

#### 8.3.2.8.2.1 Type-C Data Role Swap, Initiated by UFP Operating as Sink

This is an example of Message sequence between a Port Pair both utilizing the Type-C connector.

Figure 8-17 shows an example sequence between a [USBType-C 1.0] DRP, which is initially a UFP (Device) and a Sink ( $R_d$  asserted), and a [USBType-C 1.0] DRP which is initially a DFP (Host) and a Source ( $R_p$  asserted). A Data Role Swap is initiated by the UFP. During the process the Port Partners maintain their operation as either a Source or a Sink (power and  $R_p/R_d$  remain constant) but exchange data roles between DFP (Host) and UFP (Device).

Figure 8-17 Type-C Data Role Swap, UFP operating as Sink initiates

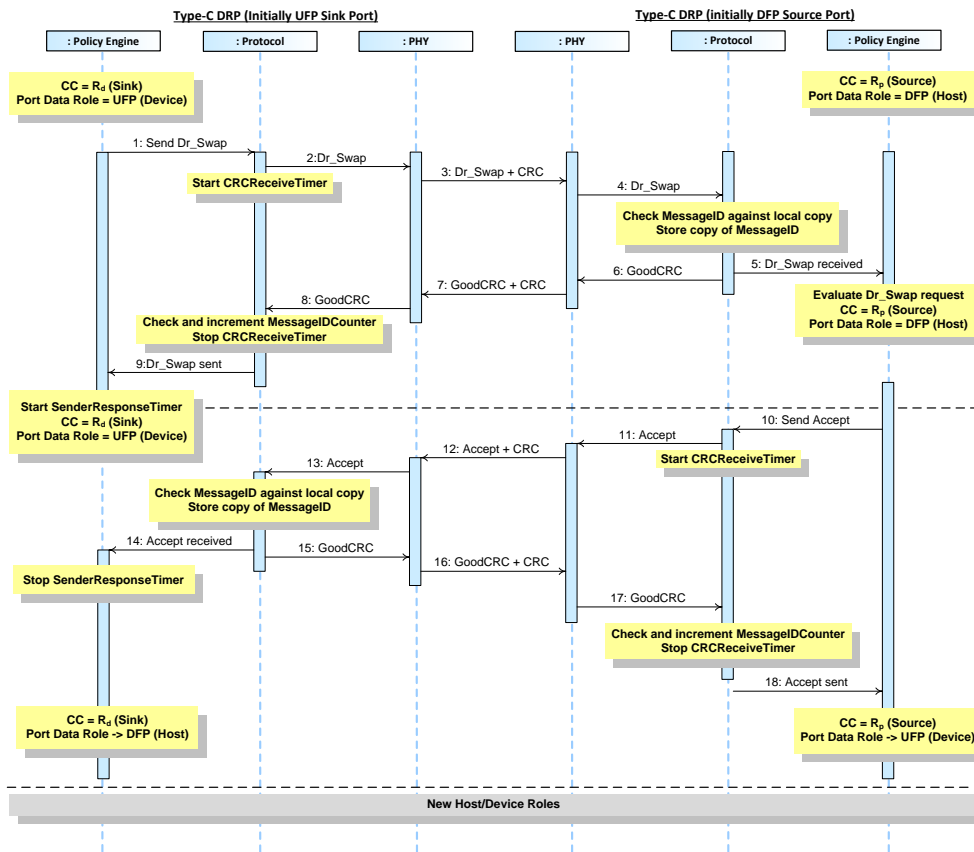


Table 8-10 below provides a detailed explanation of what happens at each labeled step in Figure 8-17 above.

Table 8-16 Steps for Type-C Data Role Swap, UFP operating as Sink initiates

Step	Type-C DRP (initially UFP Sink Port)	Type-C DRP (initially DFP Source Port)
1	Port starts as a UFP (Device) operating as a Sink with $R_d$ asserted and <i>Port Data Role</i> set to UFP. The Policy Engine directs the Protocol Layer to send a <i>DR_Swap</i> Message.	Port starts as a DFP (Host) operating as Source with $R_p$ asserted and <i>Port Data Role</i> set to DFP.
2	Protocol Layer creates the <i>DR_Swap</i> Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>DR_Swap</i> Message.	Physical Layer receives the <i>DR_Swap</i> Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>DR_Swap</i> Message to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>DR_Swap</i> Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>DR_Swap</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .	
10		Policy Engine evaluates the <i>DR_Swap</i> Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form an <i>Accept</i> Message.
11		Protocol Layer creates the <i>Accept</i> Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	Physical Layer receives the <i>Accept</i> Message and checks the CRC to verify the Message.	Physical Layer appends a CRC and sends the <i>Accept</i> Message.
13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Accept</i> Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>SenderResponseTimer</i> ; requests that Data Role is changed from UFP (Device) to DFP (Host). The Power Delivery role is now a DFP (Host), with <i>Port Data Role</i> set to DFP, still operating as a Sink ( $R_d$ asserted).	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.

Step	Type-C DRP (initially UFP Sink Port)	Type-C DRP (initially DFP Source Port)
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18	<u>The Policy Engine requests that Data Role is changed from UFP (Device) to DFP (Host).</u> <u>The Power Delivery role is now a DFP (Host), with Port Data Role set to DFP, still operating as a Sink (R<sub>d</sub> asserted).</u>	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent. The Policy Engine requests that the Data Role is changed to UFP (Device), with <i>Port Data Role</i> set to UFP and continues supplying power as a Source (R <sub>p</sub> asserted).
	The Data Role Swap is complete, the data roles have been reversed while maintaining the direction of power flow.	

### 8.3.2.8.2.2 Type-C Data Role Swap, Initiated by UFP Operating as Source

This is an example of Message sequence between a Port Pair both utilizing the Type-C connector.

Figure 8-18 shows an example sequence between a [USBType-C.1.0] DRP, which is initially a UFP (Device) and a Source ( $R_p$  asserted), and a [USBType-C.1.0] DRP which is initially a DFP (Host) and a Sink ( $R_d$  asserted). A Data Role Swap is initiated by the UFP. During the process the Port Partners maintain their operation as either a Source or a Sink (power and  $R_p/R_d$  remain constant) but exchange data roles between DFP (Host) and UFP (Device).

Figure 8-18 Type-C Data Role Swap, UFP operating as Source initiates

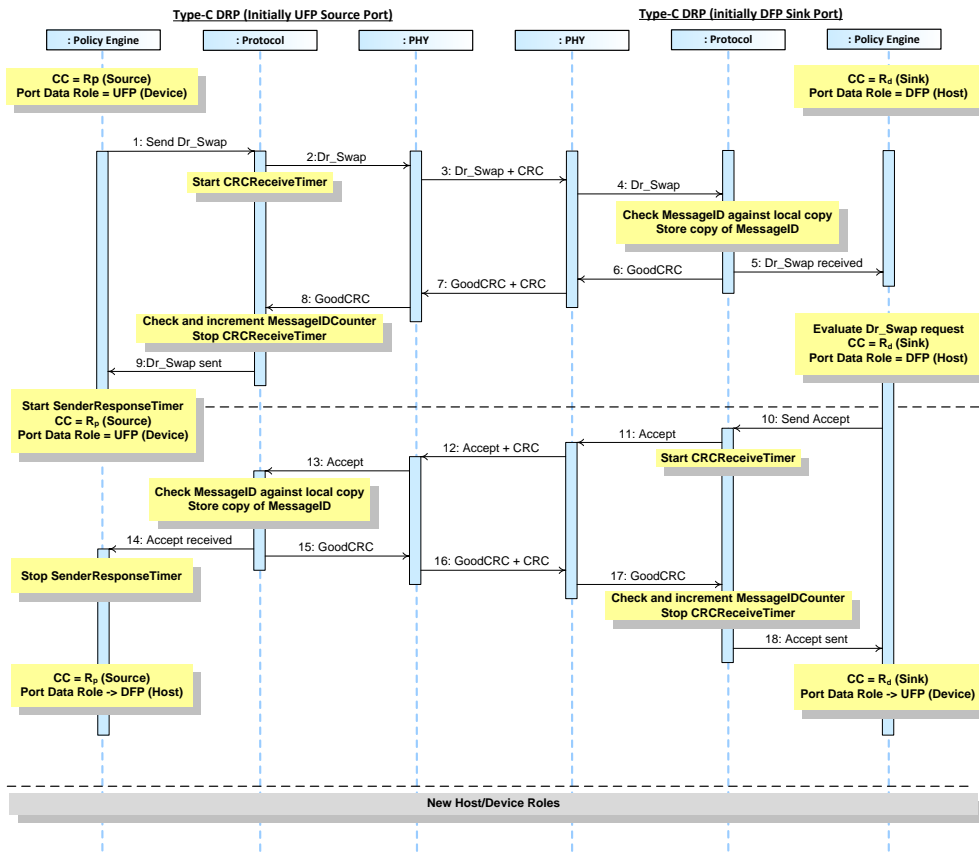


Table 8-17 below provides a detailed explanation of what happens at each labeled step in Figure 8-18 above.

Table 8-17 Steps for Type-C Data Role Swap, UFP operating as Source initiates

Step	Type-C DRP (initially UFP Source Port)	Type-C DRP (initially DFP Sink Port)
1	Port starts as a UFP (Device) operating as Source with $R_p$ asserted and <i>Port Data Role</i> set to UFP. The Policy Engine directs the Protocol Layer to send a <i>DR_Swap</i> Message.	Port starts as a DFP (Host) operating as a Sink with $R_d$ asserted and <i>Port Data Role</i> set to DFP.
2	Protocol Layer creates the <i>DR_Swap</i> Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>DR_Swap</i> Message.	Physical Layer receives the <i>DR_Swap</i> Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>DR_Swap</i> Message to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>DR_Swap</i> Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>DR_Swap</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .	
10		Policy Engine evaluates the <i>DR_Swap</i> Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form an <i>Accept</i> Message.
11		Protocol Layer creates the <i>Accept</i> Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	Physical Layer receives the <i>Accept</i> Message and checks the CRC to verify the Message.	Physical Layer appends a CRC and sends the <i>Accept</i> Message.
13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Accept</i> Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>SenderResponseTimer</i> ; <del>requests that Data Role is changed from UFP (Device) to DFP (Host).</del> The Power Delivery role is now a DFP (Host), and <i>Port Data Role</i> set to DFP, and continues supplying power as a Source ( $R_p$ asserted).	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.



Step	Type-C DRP (initially UFP Source Port)	Type-C DRP (initially DFP Sink Port)
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18	<u>The Policy Engine requests that Data Role is changed from UFP (Device) to DFP (Host). The Power Delivery role is now a DFP (Host), and Port Data Role set to DFP, and continues supplying power as a Source (R<sub>p</sub> asserted).</u>	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent. The Policy Engine requests that the Data Role is changed to UFP (Device), with <i>Port Data Role</i> set to UFP and still operating as a Sink (R <sub>p</sub> asserted).
	The Data Role Swap is complete, the data roles have been reversed while maintaining the direction of power flow.	

### 8.3.2.8.2.3 Type-C Data Role Swap, Initiated by DFP Operating as Source

This is an example of Message sequence between a Port Pair both utilizing the Type-C connector.

Figure 8-19 shows an example sequence between a [USBType-C.1.0] DRP, which is initially a UFP (Device) and a Sink ( $R_d$  asserted), and a [USBType-C.1.0] DRP which is initially a DFP and a Source ( $R_p$  asserted). A Data Role Swap is initiated by the DFP. During the process the Port Partners maintain their operation as either a Source or a Sink (power and  $R_p/R_d$  remain constant) but exchange data roles between DFP (Host) and UFP (Device).

Figure 8-19 Type-C Data Role Swap, DFP operating as Source initiates

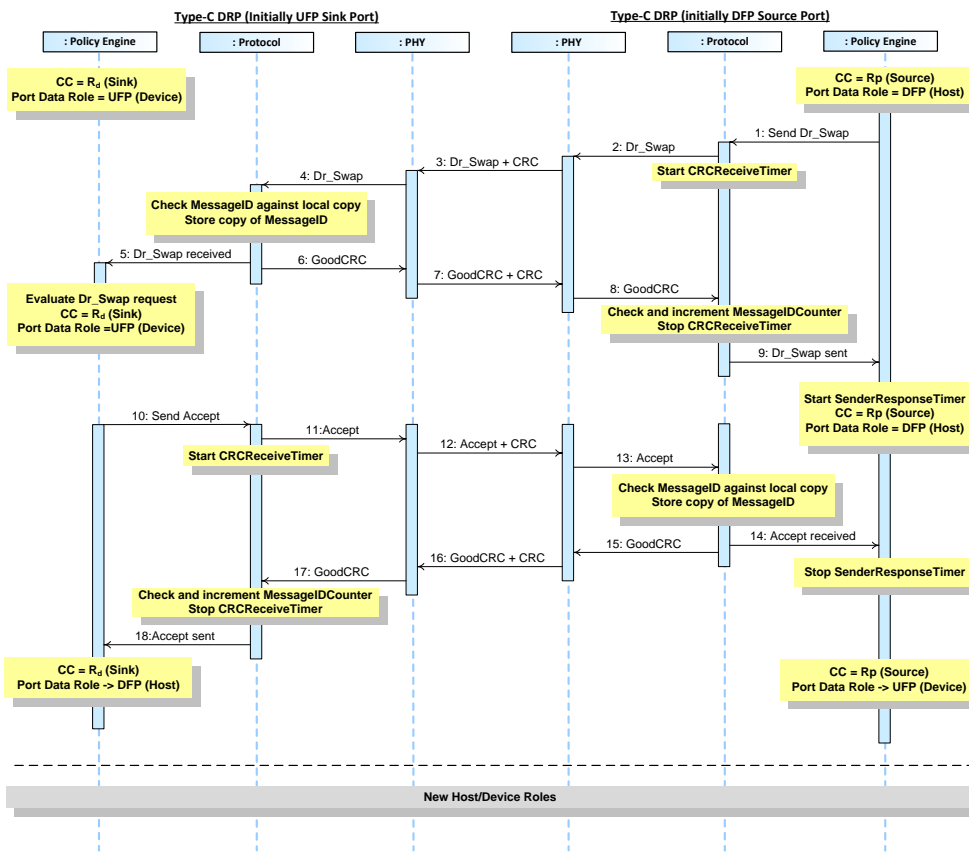


Table 8-18 below provides a detailed explanation of what happens at each labeled step in Figure 8-19 above.

Table 8-18 Steps for Type-C Data Role Swap, DFP operating as Source initiates

Step	Type-C DRP (initially UFP Sink Port)	Type-C DRP (initially DFP Source Port)
1	Port starts as a UFP (Device) operating as a Sink with $R_d$ asserted and <i>Port Data Role</i> set to UFP.	Port starts as a DFP (Host) operating as Source with $R_p$ asserted and <i>Port Data Role</i> set to DFP. The Policy Engine directs the Protocol Layer to send a <i>DR_Swap</i> Message.
2		Protocol Layer creates the <i>DR_Swap</i> Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
3	Physical Layer receives the <i>DR_Swap</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>DR_Swap</i> Message.
4	Physical Layer removes the CRC and forwards the <i>DR_Swap</i> Message to the Protocol Layer.	
5	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>DR_Swap</i> Message information to the Policy Engine that consumes it.	
6	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
7	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.
8		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
9		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>DR_Swap</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .
10	Policy Engine evaluates the <i>DR_Swap</i> Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form an <i>Accept</i> Message.	
11	Protocol Layer creates the <i>Accept</i> Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
12	Physical Layer appends a CRC and sends the <i>Accept</i> Message.	Physical Layer receives the <i>Accept</i> Message and checks the CRC to verify the Message.
13		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Accept</i> Message information to the Policy Engine that consumes it.
14		The Policy Engine stops the <i>SenderResponseTimer</i> ; requests that Data Role is changed from DFP (Host) to UFP (Device). The Power Delivery role is now a UFP (Device), with <i>Port Data Role</i> set to UFP, and continues supplying power as a Source ( $R_p$ asserted).
15		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
16	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.

Step	Type-C DRP (initially UFP Sink Port)	Type-C DRP (initially DFP Source Port)
17	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent. . The Policy Engine requests that the Data Role is changed to DFP (Host), with <i>Port Data Role</i> set to DFP, still operating as a Sink ( $R_d$ asserted).	<u>The Policy Engine requests that Data Role is changed from DFP (Host) to UFP (Device).</u> <u>The Power Delivery role is now a UFP (Device), with Port Data Role set to UFP, and continues supplying power as a Source (<math>R_p</math> asserted).</u>
	The Data Role Swap is complete, the data roles have been reversed while maintaining the direction of power flow.	

### 8.3.2.8.2.4 Type-C Data Role Swap, Initiated by DFP Operating as Sink

This is an example of Message sequence between a Port Pair both utilizing the Type-C connector.

Figure 8-20 shows an example sequence between a [USBType-C.1.0] DRP, which is initially a UFP (Device) and a Source ( $R_p$  asserted), and a [USBType-C.1.0] DRP which is initially a DFP (Host) and a Sink ( $R_d$  asserted). A Data Role Swap is initiated by the DFP. During the process the Port Partners maintain their operation as either a Source or a Sink (power and  $R_p/R_d$  remain constant) but exchange data roles between DFP (Host) and UFP (Device).

Figure 8-20 Type-C Data Role Swap, DFP operating as Sink initiates

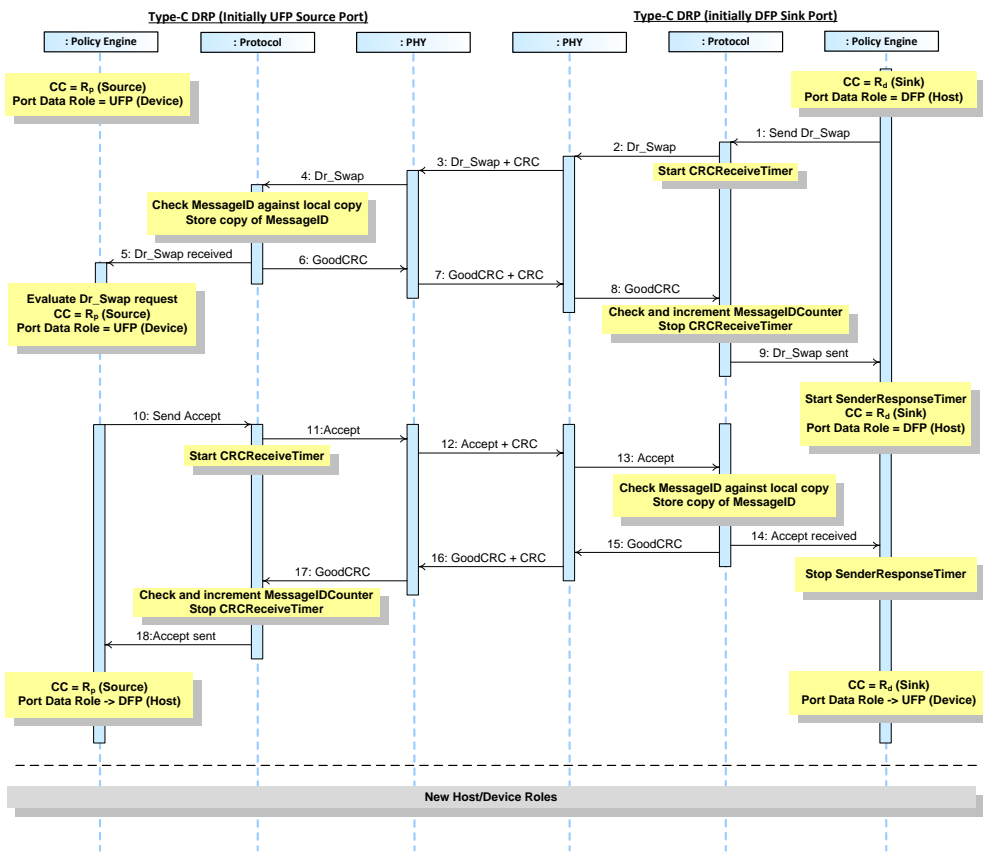


Table 8-19 below provides a detailed explanation of what happens at each labeled step in Figure 8-20 above.

Table 8-19 Steps for Type-C Data Role Swap, DFP operating as Sink initiates

Step	Type-C DRP (initially UFP Source Port)	Type-C DRP (initially DFP Sink Port)
1	Port starts as a UFP (Device) operating as Source with $R_p$ asserted and <i>Port Data Role</i> set to UFP.	Port starts as a DFP (Host) operating as a Sink with $R_d$ asserted and <i>Port Data Role</i> set to DFP. The Policy Engine directs the Protocol Layer to send a <i>DR_Swap</i> Message.
2		Protocol Layer creates the <i>DR_Swap</i> Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
3	Physical Layer receives the <i>DR_Swap</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>DR_Swap</i> Message.
4	Physical Layer removes the CRC and forwards the <i>DR_Swap</i> Message to the Protocol Layer.	
5	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>DR_Swap</i> Message information to the Policy Engine that consumes it.	
6	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
7	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.
8		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
9		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>DR_Swap</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .
10	Policy Engine evaluates the <i>DR_Swap</i> Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form an <i>Accept</i> Message.	
11	Protocol Layer creates the <i>Accept</i> Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
12	Physical Layer appends a CRC and sends the <i>Accept</i> Message.	Physical Layer receives the <i>Accept</i> Message and checks the CRC to verify the Message.
13		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Accept</i> Message information to the Policy Engine that consumes it.
14		The Policy Engine stops the <i>SenderResponseTimer</i> ; requests that Data Role is changed from DFP (Host) to UFP (Device). The Power Delivery role is now a UFP (Device), with <i>Port Data Role</i> set to UFP, still operating as a Sink ( $R_d$ asserted).
15		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
16	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.

Step	Type-C DRP (initially UFP Source Port)	Type-C DRP (initially DFP Sink Port)
17	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent. The Policy Engine requests that the Data Role is changed to DFP (Host), with <i>Port Data Role</i> set to DFP, and continues supplying power as a Source ( $R_P$ asserted).	<u>The Policy Engine requests that Data Role is changed from DFP (Host) to UFP (Device).</u> <u>The Power Delivery role is now a UFP (Device), with Port Data Role set to UFP, still operating as a Sink (<math>R_d</math> asserted).</u>
	The Data Role Swap is complete, the data roles have been reversed while maintaining the direction of power flow.	

### 8.3.2.8.3 Type-C VCONN

#### 8.3.2.8.3.1 Type-C DFP to UFP VCONN Source Swap

This is an example of Message sequence between a Port Pair both utilizing the Type-C connector.

Figure 8-21 shows an example sequence between a Type-C DFP and UFP, where the DFP is initially supplying VCONN and then tells the UFP to supply VCONN. During the process the Port Partners, keep their role as DFP or UFP, maintain their operation as either a Source or a Sink (power remains constant) but exchange the VCONN Source from the DFP to the UFP.

Figure 8-21 Type-C DFP to UFP VCONN Source Swap

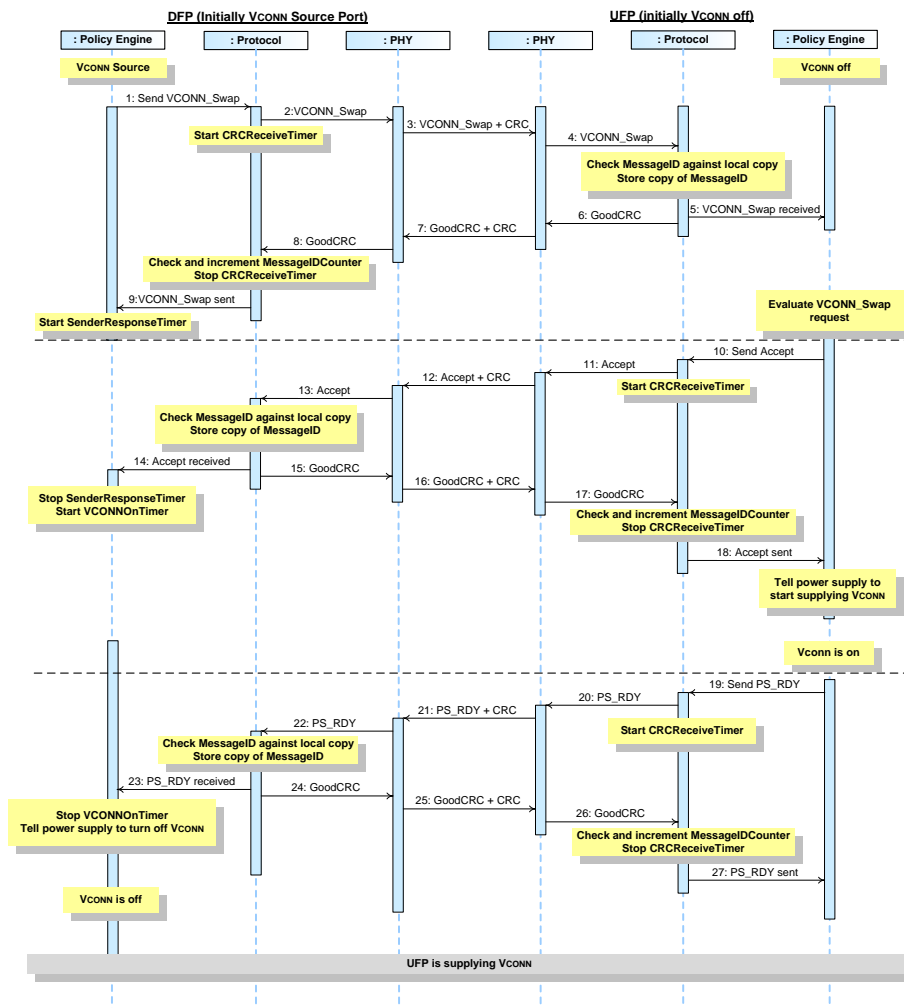




Table 8-20 below provides a detailed explanation of what happens at each labeled step in Figure 8-21 above.

Table 8-20 Steps for Type-C DFP to UFP VCONN Source Swap

Step	DFP (initially VCONN Source)	UFP (Initially VCONN off)
1	The DFP starts as the VCONN Source. The Policy Engine directs the Protocol Layer to send a <i>VCONN_Swap</i> Message.	The UFP starts with VCONN off.
2	Protocol Layer creates the <i>VCONN_Swap</i> Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>VCONN_Swap</i> Message.	Physical Layer receives the <i>VCONN_Swap</i> Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>VCONN_Swap</i> Message to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>VCONN_Swap</i> Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>VCONN_Swap</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .	
10		Policy Engine evaluates the <i>VCONN_Swap</i> Message sent by the DFP and decides that it is able and willing to do the VCONN Swap. It tells the Protocol Layer to form an <i>Accept</i> Message.
11		Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	Physical Layer receives the <i>Accept</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>Accept</i> Message.
13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Accept</i> Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>SenderResponseTimer</i> and starts the <i>VCONNOnTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.

Step	DFP (initially VCONN Source)	UFP (Initially VCONN off)
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent. The Policy Engine asks the Device Policy Manager to turn on VCONN.
19		The Device Policy Manager informs the Policy Engine that its power supply is supplying VCONN. The Policy Engine directs the Protocol Layer to generate a <i>PS_RDY</i> Message to tell the DFP it can turn off VCONN.
20		Protocol Layer creates the <i>PS_RDY</i> Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
21	Physical Layer receives the <i>PS_RDY</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>PS_RDY</i> Message.
22	Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.	
23	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PS_RDY</i> Message information to the Policy Engine that consumes it.	
24	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
25	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message. The Policy Engine stops the <i>VCONNOnTimer</i> , and tells the power supply to stop sourcing VCONN.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
26		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
27	VCONN is off.	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent.
	The UFP is now the VCONN Source and the DFP has VCONN turned off.	

### 8.3.2.8.3.2 Type-C UFP to DFP VCONN Source Swap

This is an example of Message sequence between a Port Pair both utilizing the Type-C connector.

Figure 8-22 shows an example sequence between a Type-C DFP and UFP, where the UFP is initially supplying VCONN and then the DFP tells the UFP that it will become the VCONN Source. During the process the Port Partners, keep their role as DFP or UFP, maintain their operation as either a Source or a Sink (power remains constant) but exchange the VCONN Source from the UFP to the DFP.

Figure 8-22 Type-C UFP to DFP VCONN Source Swap

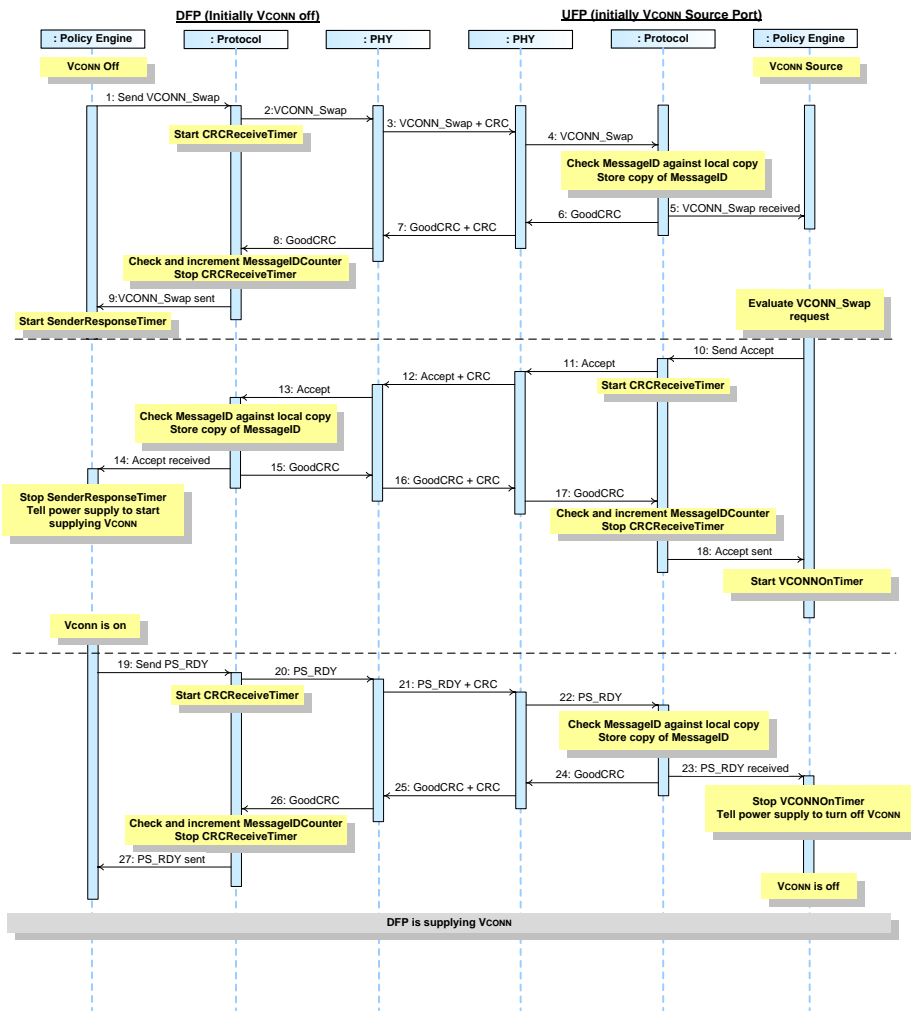


Table 8-21 below provides a detailed explanation of what happens at each labeled step in Figure 8-22 above.

Table 8-21 Steps for Type-C UFP to DFP VCONN Source Swap

Step	DFP	UFP
1	The DFP starts with VCONN off. The Policy Engine directs the Protocol Layer to send a <i>VCONN_Swap</i> Message.	The UFP starts as the VCONN Source.
2	Protocol Layer creates the <i>VCONN_Swap</i> Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>VCONN_Swap</i> Message.	Physical Layer receives the <i>VCONN_Swap</i> Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>VCONN_Swap</i> Message to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>VCONN_Swap</i> Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>VCONN_Swap</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .	
10		Policy Engine evaluates the <i>VCONN_Swap</i> Message sent by the DFP and decides that it is able and willing to do the VCONN Swap. It tells the Protocol Layer to form an <i>Accept</i> Message.
11		Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	Physical Layer receives the <i>Accept</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>Accept</i> Message.
13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Accept</i> Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>SenderResponseTimer</i> . The Policy Engine tells the Device Policy Manger to turn on VCONN.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.

Step	DFP	UFP
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent. The Policy Engine <del>tells</del> <i>starts</i> the <i>VCONNOnTimer</i> <del>power supply to turn on VCONN.</del>
19	The Device Policy Manager tells the Policy Engine that its power supply is supplying VCONN. The Policy Engine directs the Protocol Layer to generate a <i>PS_RDY</i> Message to tell the UFP it can turn off VCONN.	
20	Protocol Layer creates the <i>PS_RDY</i> Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
21	Physical Layer appends a CRC and sends the <i>PS_RDY</i> Message.	Physical Layer receives the <i>PS_RDY</i> Message and compares the CRC it calculated with the one sent to verify the Message.
22		Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.
23		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PS_RDY</i> Message information to the Policy Engine that consumes it.
24		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
25	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message. The Policy Engine stops the <i>VCONNOnTimer</i> , and tells the power supply to stop sourcing VCONN.
26	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
27	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent.	VCONN is off.
The DFP is now the VCONN Source and the UFP has VCONN turned off.		

### 8.3.2.9 Structured VDM

#### 8.3.2.9.1 DFP to UFP Discover Identity

This is an example of Message sequence between a Port Pair both utilizing the Type-C connector.

Figure 8-23 shows an example sequence between a Type-C DFP and UFP, where both Port Partners are in an Explicit Contract and the DFP attempts to discover identity information from the UFP.

Figure 8-23 DFP to UFP Discover Identity

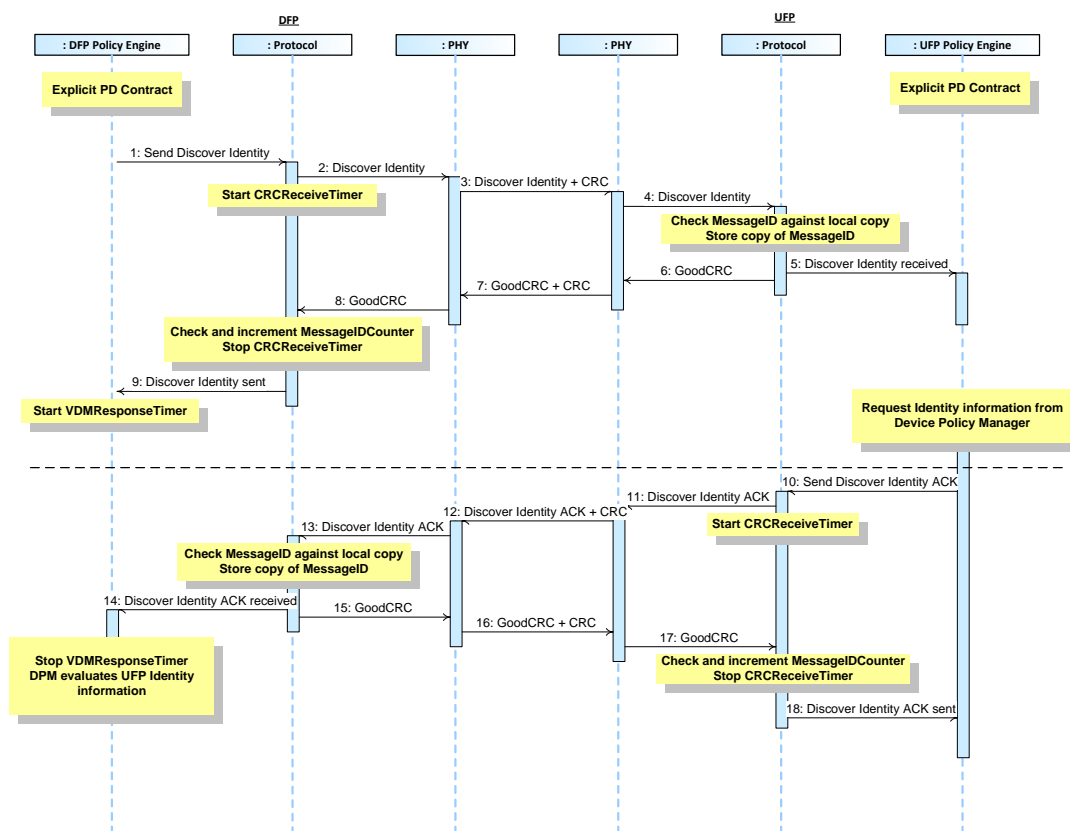


Table 8-22 below provides a detailed explanation of what happens at each labeled step in Figure 8-23 above.

Table 8-22 Steps for DFP to UFP Discover Identity

Step	DFP	UFP
1	The DFP has an Explicit Contract. The Policy Engine directs the Protocol Layer to send a <i>Discover Identity</i> Command request.	The UFP has an Explicit Contract.
2	Protocol Layer creates the <i>Discover Identity</i> Command request and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	

Step	DFP	UFP
3	Physical Layer appends CRC and sends the <i>Discover Identity</i> Command request.	Physical Layer receives the <i>Discover Identity</i> Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>Discover Identity</i> Command request to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Discover Identity</i> Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover Identity</i> Command request was successfully sent. Policy Engine starts the <i>VDMResponseTimer</i> .	
10		Policy Engine requests the identity information from the Device Policy Manager. The Policy Engine tells the Protocol Layer to form a <i>Discover Identity</i> Command ACK response.
11		Protocol Layer creates the <i>Discover Identity</i> Command ACK response and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	Physical Layer receives the <i>Discover Identity</i> Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>Discover Identity</i> Command ACK response.
13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Discover Identity</i> Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>VDMResponseTimer</i> and passed the Identity information to the Device Policy Manager for evaluation.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover Identity</i> Command ACK response was successfully sent.

### 8.3.2.9.2 Source Port to Cable Plug Discover Identity

This is an example of Message sequence between a Source and a Cable Plug both utilizing the Type-C connector.

Figure 8-24 shows an example sequence between a Type-C Source and a Cable Plug, where the Source attempts to discover identity information from the Cable Plug prior to establishing an Explicit Contract with its Port Partner.

Figure 8-24 Source Port to Cable Plug Discover Identity

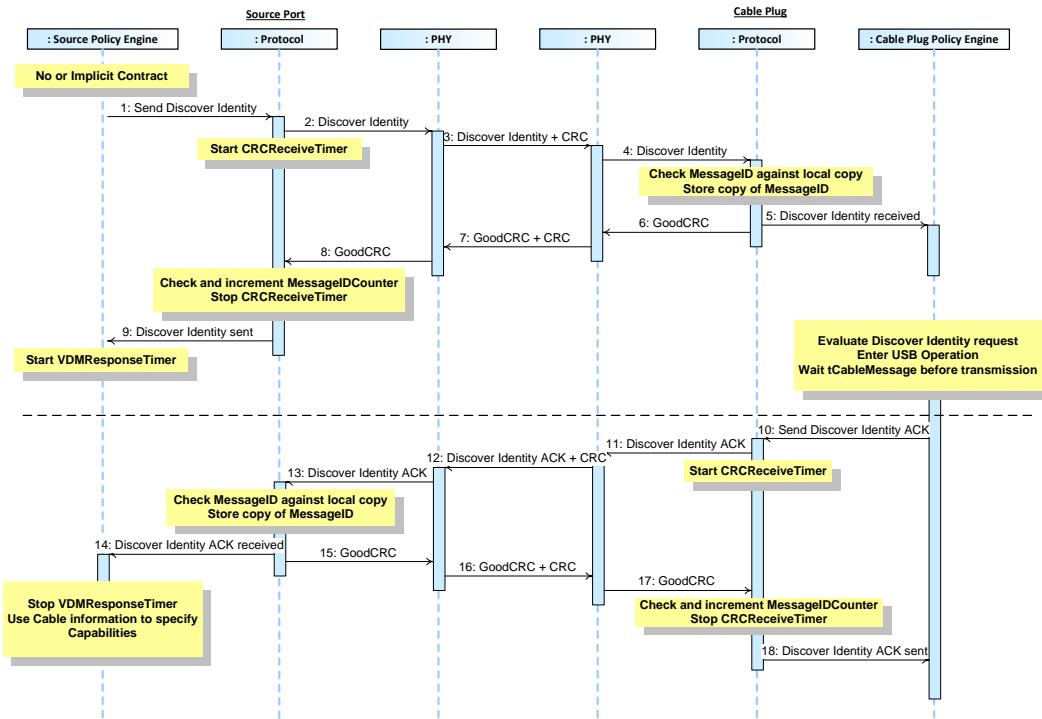


Table 8-23 below provides a detailed explanation of what happens at each labeled step in Figure 8-24 above.

Table 8-23 Steps for Source Port to Cable Plug Discover Identity

Step	Source Port	Cable Plug
1	The Source has no Contract or an Implicit Contract with its Port Partner. The Policy Engine directs the Protocol Layer to send a <i>Discover Identity</i> Command request.	
2	Protocol Layer creates the <i>Discover Identity</i> Command request and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>Discover Identity</i> Command request.	Physical Layer receives the <i>Discover Identity</i> Command request and checks the CRC to verify the Message.



Step	Source Port	Cable Plug
4		Physical Layer removes the CRC and forwards the <i>Discover Identity</i> Command request to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Discover Identity</i> Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover Identity</i> Command request was successfully sent. Policy Engine starts the <i>VDMResponseTimer</i> .	
10		Policy Engine requests the identity information from the Device Policy Manager. <i>.tCableMessage after the GoodCRC Message was sent</i> the Policy Engine tells the Protocol Layer to form a <i>Discover Identity</i> Command ACK response.
11		Protocol Layer creates the <i>Discover Identity</i> Command ACK response and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	Physical Layer receives the <i>Discover Identity</i> Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>Discover Identity</i> Command ACK response.
13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Discover Identity</i> Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>VDMResponseTimer</i> and passes the identity information to the Device Policy Manager for evaluation.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18	The Source uses the Cable Plug information as input to its offered capabilities.	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent.

### 8.3.2.9.3 DFP to Cable Plug Discover Identity

This is an example of Message sequence between a DFP and a Cable Plug both utilizing the Type-C connector.

Figure 8-25 shows an example sequence between a Type-C DFP and a Cable Plug, where the DFP attempts to discover identity information from the Cable Plug.

Figure 8-25 DFP to Cable Plug Discover Identity

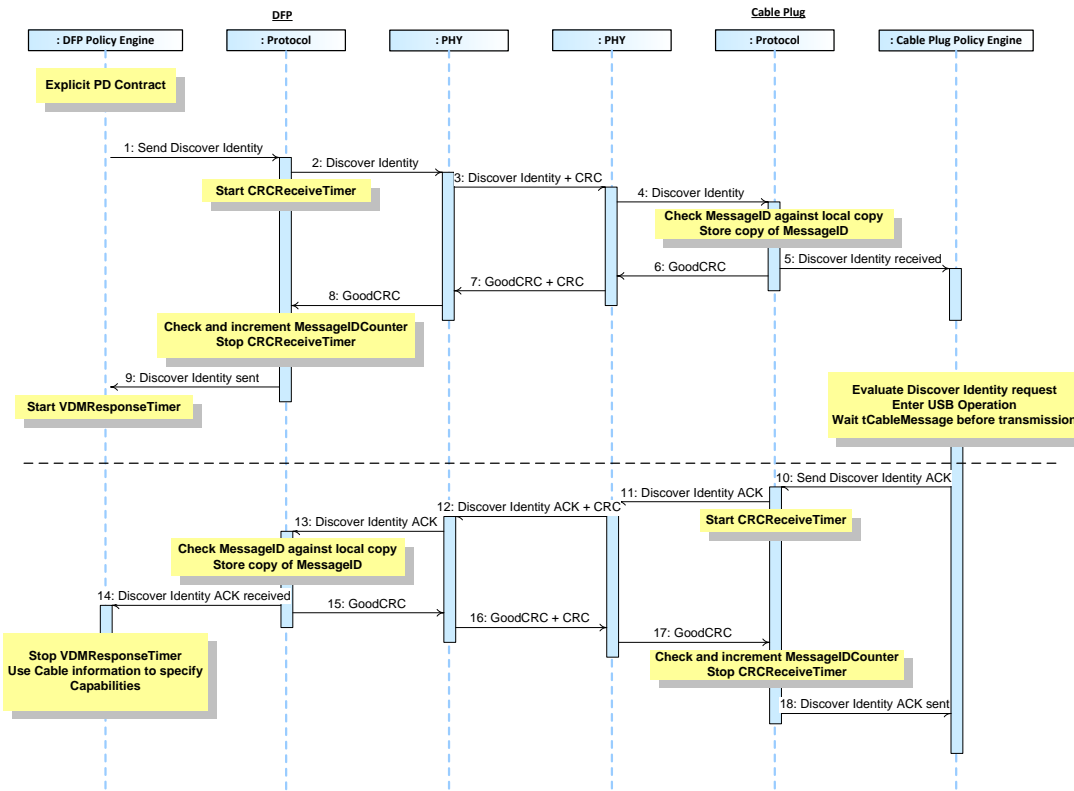


Table 8-24 below provides a detailed explanation of what happens at each labeled step in Figure 8-25 above.

Table 8-24 Steps for DFP to Cable Plug Discover Identity

Step	DFP	Cable Plug
1	The DFP has an Explicit Contract with its Port Partner. The Policy Engine directs the Protocol Layer to send a <i>Discover Identity</i> Command request.	
2	Protocol Layer creates the <i>Discover Identity</i> Command request and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	

Step	DFP	Cable Plug
3	Physical Layer appends CRC and sends the <i>Discover Identity</i> Command request.	Physical Layer receives the <i>Discover Identity</i> Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>Discover Identity</i> Command request to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Discover Identity</i> Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover Identity</i> Command request was successfully sent. Policy Engine starts the <i>VDMResponseTimer</i> .	
10		Policy Engine requests the identity information from the Device Policy Manager. <i>CableMessage after the GoodCRC Message was sent</i> the Policy Engine tells the Protocol Layer to form a <i>Discover Identity</i> Command ACK response.
11		Protocol Layer creates the <i>Discover Identity</i> Command ACK response and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	Physical Layer receives the <i>Discover Identity</i> Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>Discover Identity</i> Command ACK response.
13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Discover Identity</i> Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>Discover Identity</i> Command ACK response and passes the identity information to the Device Policy Manager for evaluation.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.

Step	DFP	Cable Plug
18	The DFP when acting as a Source uses the Cable Plug information as input to its offered capabilities.	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent.

#### 8.3.2.9.4 DFP to UFP Enter Mode

This is an example of Message sequence between a DFP and a UFP both utilizing the Type-C connector.

Figure 8-26 shows an example sequence between a Type-C DFP and a UFP, where the DFP attempts to discover supported SVIDs, then Modes before selecting and entering a Mode.

Figure 8-26 DFP to UFP Enter Mode

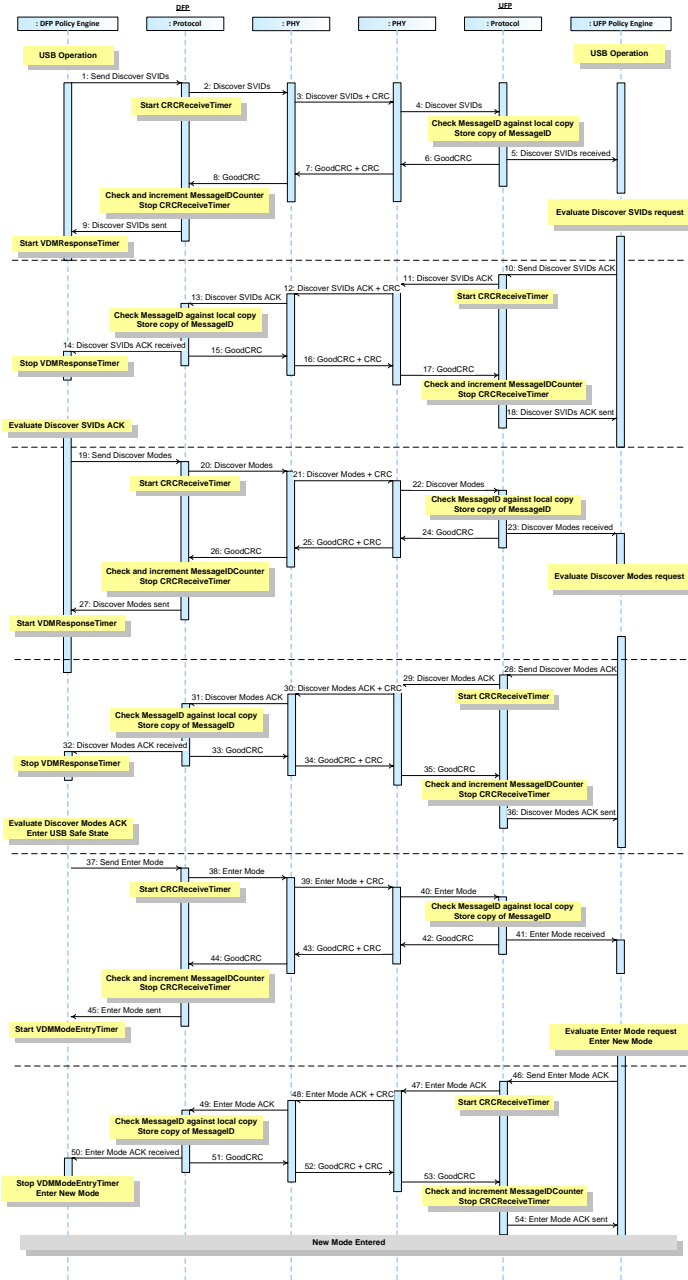


Table 8-25 below provides a detailed explanation of what happens at each labeled step in Figure 8-26 above.

Table 8-25 Steps for DFP to UFP Enter Mode

Step	DFP	UFP
1	The DFP has an Explicit Contract. The Device Policy Manager requests the Policy Engine to discover SVID information. The Policy Engine directs the Protocol Layer to send a <i>Discover SVIDs</i> Command request.	The UFP has an Explicit Contract.
2	Protocol Layer creates the <i>Discover SVIDs</i> Command request and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>Discover SVIDs</i> Command request.	Physical Layer receives the <i>Discover SVIDs</i> Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>Discover SVIDs</i> Command request to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Discover SVIDs</i> Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover SVIDs</i> Command request was successfully sent. Policy Engine starts the <i>VDMResponseTimer</i> .	
10		Policy Engine requests the SVID information from the Device Policy Manager. The Policy Engine tells the Protocol Layer to form a <i>Discover SVIDs</i> Command ACK response.
11		Protocol Layer creates the <i>Discover SVIDs</i> Command ACK response and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	Physical Layer receives the <i>Discover SVIDs</i> Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>Discover SVIDs</i> Command ACK response.
13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Discover SVIDs</i> Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>VDMResponseTimer</i> and passes the SVIDs information to the Device Policy Manager for evaluation.	

Step	DFP	UFP
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover SVIDs</i> Command ACK response was successfully sent.
19	The Device Policy Manager requests the Policy Engine to discover Mode information for a given SVID. The Policy Engine directs the Protocol Layer to send a <i>Discover Modes</i> Command request.	
20	Protocol Layer creates the <i>Discover Modes</i> Command request and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
21	Physical Layer appends CRC and sends the <i>Discover Modes</i> Command request.	Physical Layer receives the <i>Discover Modes</i> Command request and checks the CRC to verify the Message.
22		Physical Layer removes the CRC and forwards the <i>Discover Modes</i> Command request to the Protocol Layer.
23		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Discover Modes</i> Command request information to the Policy Engine that consumes it.
24		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
25	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
26	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
27	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover Modes</i> Command request was successfully sent. Policy Engine starts the <i>VDMResponseTimer</i> .	
28		Policy Engine requests the Mode information from the Device Policy Manager. The Policy Engine tells the Protocol Layer to form a <i>Discover Modes</i> Command ACK response.
29		Protocol Layer creates the <i>Discover Modes</i> Command ACK response and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
30	Physical Layer receives the <i>Discover Modes</i> Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>Discover Modes</i> Command ACK response.



Step	DFP	UFP
31	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Discover Modes</i> Command ACK response information to the Policy Engine that consumes it.	
32	The Policy Engine stops the <i>VDMResponseTimer</i> and passes the Modes information to the Device Policy Manager for evaluation.	
33	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
34	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
35		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
36		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover Modes</i> Command ACK response was successfully sent.
37	The DFP goes to USB Safe State. The Device Policy Manager requests the Policy Engine to enter a Mode. The Policy Engine directs the Protocol Layer to send an <i>Enter Mode</i> Command request.	
38	Protocol Layer creates the <i>Enter Mode</i> Command request and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
39	Physical Layer appends CRC and sends the <i>Enter Mode</i> Command request.	Physical Layer receives the <i>Enter Mode</i> Command request and checks the CRC to verify the Message.
40		Physical Layer removes the CRC and forwards the <i>Enter Mode</i> Command request to the Protocol Layer.
41		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Enter Mode</i> Command request information to the Policy Engine that consumes it.
42		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
43	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
44	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
45	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Enter Mode</i> Command request was successfully sent. Policy Engine starts the <i>VDMModeEntryTimer</i> .	
46		Policy Engine requests the Device Policy Manager to enter the new Mode. The Policy Engine tells the Protocol Layer to form an <i>Enter Mode</i> Command ACK response.

Step	DFP	UFP
47		Protocol Layer creates the <i>Enter Mode</i> Command ACK response and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
48	Physical Layer receives the <i>Enter Mode</i> Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>Enter Mode</i> Command ACK response.
49	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Enter Mode</i> Command ACK response information to the Policy Engine that consumes it.	
50	The Policy Engine stops the <i>VDMModeEntryTimer</i> and requests the Device Policy Manager to enter the new Mode.	
51	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
52	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
53		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
54		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Enter Mode</i> Command ACK response was successfully sent.
DPPDFP and UFP are operating in the new Mode		

### 8.3.2.9.5 DFP to UFP Exit Mode

This is an example of Message sequence between a DFP and a UFP both utilizing the Type-C connector.

Figure 8-27 shows an example sequence between a Type-C DFP and a UFP, where the DFP commands the UFP to exit the Active Mode.

Figure 8-27 DFP to UFP Exit Mode

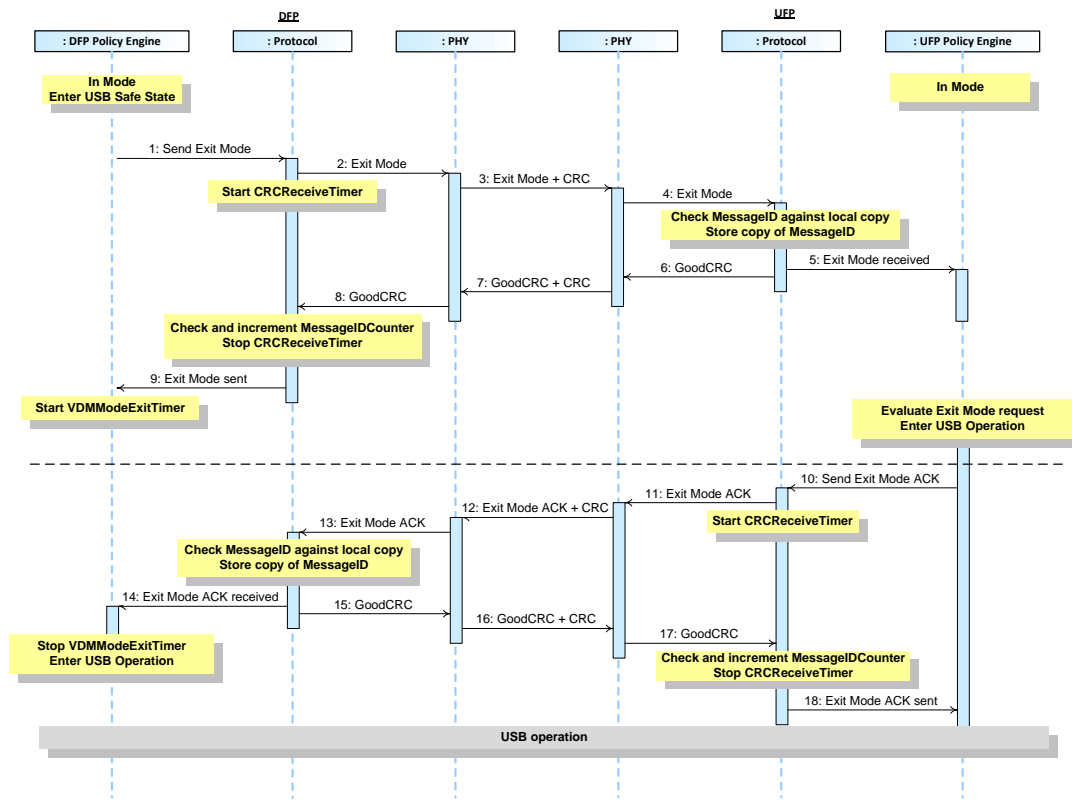


Table 8-26 below provides a detailed explanation of what happens at each labeled step in Figure 8-27 above.

Table 8-26 Steps for DFP to UFP Exit Mode

Step	DFP	UFP
1	The DFP is in a Mode and then enters USB Safe State. The Policy Engine directs the Protocol Layer to send a <i>Exit Mode</i> Command request.	The UFP is in a Mode.
2	Protocol Layer creates the <i>Exit Mode</i> Command request and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>Exit Mode</i> Command request.	Physical Layer receives the <i>Exit Mode</i> Command request and checks the CRC to verify the Message.

Step	DFP	UFP
4		Physical Layer removes the CRC and forwards the <i>Exit Mode</i> Command request to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Exit Mode</i> Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Exit Mode</i> Command request was successfully sent. Policy Engine starts the <i>VDMModeExitTimer</i> .	
10		Policy Engine requests the Device Policy Manager to enter USB operation. The Policy Engine tells the Protocol Layer to form an <i>Exit Mode</i> Command ACK response.
11		Protocol Layer creates the <i>Exit Mode</i> Command ACK response and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	Physical Layer receives the <i>Exit Mode</i> Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>Exit Mode</i> Command ACK response.
13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Exit Mode</i> Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>VDMModeExitTimer</i> and requests the Device Policy Manager to enter USB Operation.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Exit Mode</i> Command ACK response was successfully sent.
Both DFP and UFP are in USB Operation		

#### 8.3.2.9.6 DFP to Cable Plug Enter Mode

This is an example of Message sequence between a DFP and a Cable Plug both utilizing the Type-C connector.

Figure 8-28 shows an example sequence between a Type-C DFP and a Cable Plug, where the DFP attempts to discover supported SVIDs, then Modes before selecting and entering a Mode.

Figure 8-28 DFP to Cable Plug Enter Mode

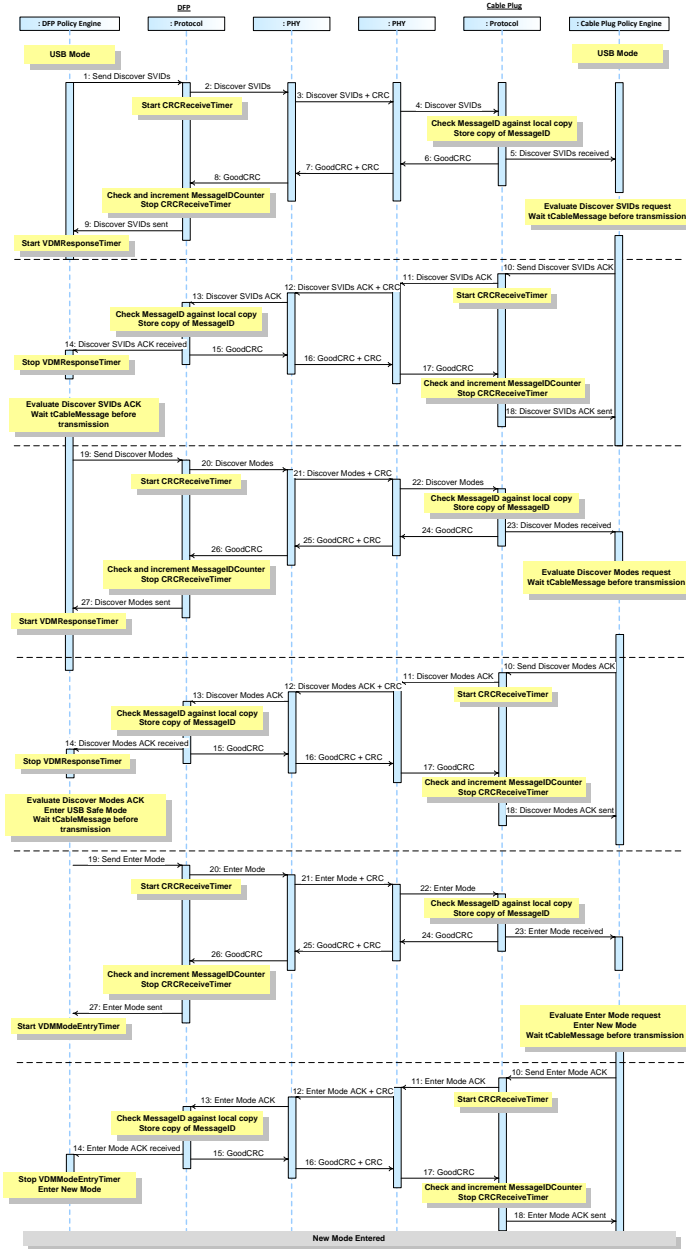


Table 8-27 below provides a detailed explanation of what happens at each labeled step in Figure 8-28 above.

Table 8-27 Steps for DFP to Cable Plug Enter Mode

Step	DFP	Cable Plug
1	The DFP has an Explicit Contract. The Device Policy Manager requests the Policy Engine to discover SVID information. The Policy Engine directs the Protocol Layer to send a <i>Discover SVIDs</i> Command request.	The UFP has an Explicit Contract.
2	Protocol Layer creates the <i>Discover SVIDs</i> Command request and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>Discover SVIDs</i> Command request.	Physical Layer receives the <i>Discover SVIDs</i> Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>Discover SVIDs</i> Command request to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Discover SVIDs</i> Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover SVIDs</i> Command request was successfully sent. Policy Engine starts the <i>VDMResponseTimer</i> .	
10		Policy Engine requests the SVID information from the Device Policy Manager. <i>CableMessage after the GoodCRC Message was sent</i> the Policy Engine tells the Protocol Layer to form a <i>Discover SVIDs</i> Command ACK response.
11		Protocol Layer creates the <i>Discover SVIDs</i> Command ACK response and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	Physical Layer receives the <i>Discover SVIDs</i> Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>Discover SVIDs</i> Command ACK response.
13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Discover SVIDs</i> Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>VDMResponseTimer</i> and passes the SVIDs information to the Device Policy Manager for evaluation.	

Step	DFP	Cable Plug
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover SVIDs</i> Command ACK response was successfully sent.
19	The Device Policy Manager requests the Policy Engine to discover Mode information for a given SVID. <i>tCableMessage after the GoodCRC Message was sent</i> the Policy Engine directs the Protocol Layer to send a <i>Discover Modes</i> Command request.	
20	Protocol Layer creates the <i>Discover Modes</i> Command request and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
21	Physical Layer appends CRC and sends the <i>Discover Modes</i> Command request.	Physical Layer receives the <i>Discover Modes</i> Command request and checks the CRC to verify the Message.
22		Physical Layer removes the CRC and forwards the <i>Discover Modes</i> Command request to the Protocol Layer.
23		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Discover Modes</i> Command request information to the Policy Engine that consumes it.
24		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
25	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
26	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
27	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover Modes</i> Command request was successfully sent. Policy Engine starts the <i>VDMResponseTimer</i> .	
28		Policy Engine requests the Mode information from the Device Policy Manager. <i>tCableMessage after the GoodCRC Message was sent</i> the Policy Engine tells the Protocol Layer to form a <i>Discover Modes</i> Command ACK response.
29		Protocol Layer creates the <i>Discover Modes</i> Command ACK response and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .



Step	DFP	Cable Plug
30	Physical Layer receives the <i>Discover Modes</i> Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>Discover Modes</i> Command ACK response.
31	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Discover Modes</i> Command ACK response information to the Policy Engine that consumes it.	
32	The Policy Engine stops the <i>VDMResponseTimer</i> and passes the Modes information to the Device Policy Manager for evaluation.	
33	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
34	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
35		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
36		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover Modes</i> Command ACK response was successfully sent.
37	The DFP goes to USB Safe State. The Device Policy Manager requests the Policy Engine to enter a Mode. <i>CableMessage after the GoodCRC Message was sent</i> the Policy Engine directs the Protocol Layer to send an <i>Enter Mode</i> Command request.	
38	Protocol Layer creates the <i>Enter Mode</i> Command request and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
39	Physical Layer appends CRC and sends the <i>Enter Mode</i> Command request.	Physical Layer receives the <i>Enter Mode</i> Command request and checks the CRC to verify the Message.
40		Physical Layer removes the CRC and forwards the <i>Enter Mode</i> Command request to the Protocol Layer.
41		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Enter Mode</i> Command request information to the Policy Engine that consumes it.
42		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
43	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
44	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
45	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Enter Mode</i> Command request was successfully sent. Policy Engine starts the <i>VDMModeEntryTimer</i> .	

Step	DFP	Cable Plug
46		Policy Engine requests the Device Policy Manager to enter the new Mode. <i>CableMessage</i> after the <i>GoodCRC Message</i> was sent the Policy Engine tells the Protocol Layer to form an <i>Enter Mode</i> Command ACK response.
47		Protocol Layer creates the <i>Enter Mode</i> Command ACK response and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
48	Physical Layer receives the <i>Enter Mode</i> Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>Enter Mode</i> Command ACK response.
49	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Enter Mode</i> Command ACK response information to the Policy Engine that consumes it.	
50	The Policy Engine stops the <i>VDMModeEntryTimer</i> and requests the Device Policy Manager to enter the new Mode.	
51	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
52	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
53		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
54		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Enter Mode</i> Command ACK response was successfully sent.
	<i>BPFDFP</i> and UFP are operating in the new Mode	

### 8.3.2.9.7 DFP to Cable Plug Exit Mode

This is an example of Message sequence between a DFP and a Cable Plug both utilizing the Type-C connector.

Figure 8-29 shows an example sequence between a Type-C DFP and a Cable Plug, where the DFP commands the Cable Plug to exit an Active Mode.

Figure 8-29 DFP to Cable Plug Exit Mode

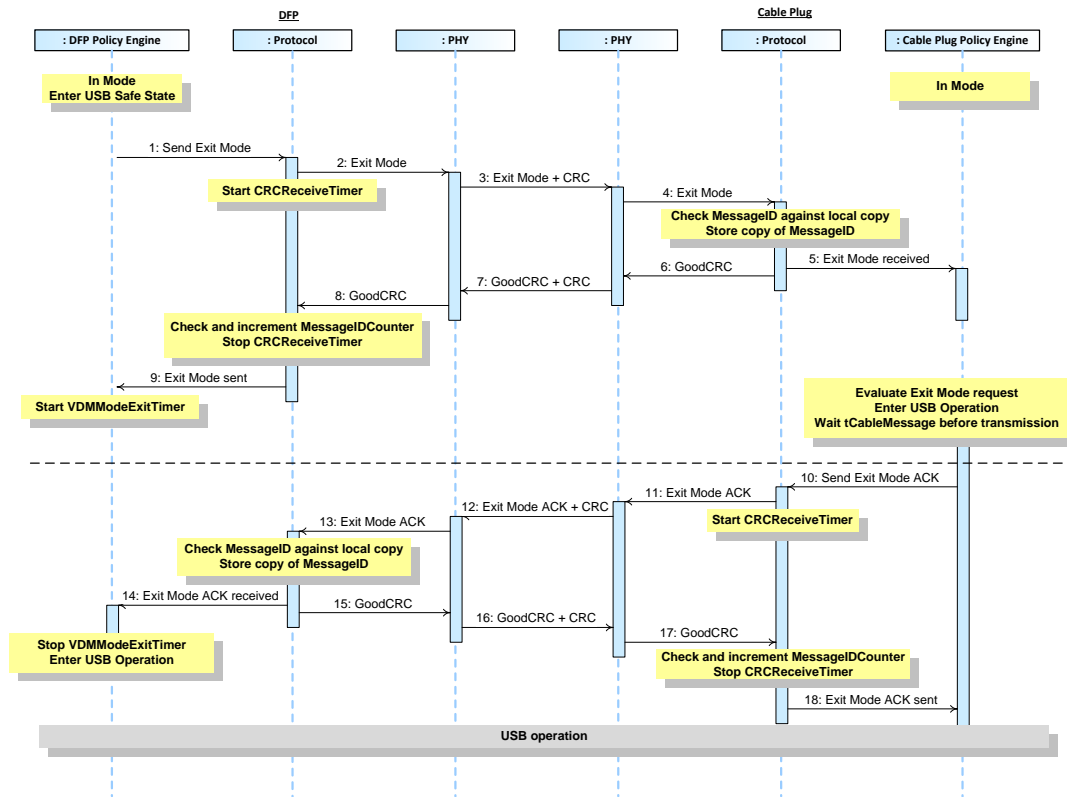


Table 8-28 below provides a detailed explanation of what happens at each labeled step in Figure 8-29 above.

Table 8-28 Steps for DFP to Cable Plug Exit Mode

Step	DFP	Cable Plug
1	The DFP is in a Mode and then enters USB Safe State. The Policy Engine directs the Protocol Layer to send a <i>Exit Mode</i> Command request.	The Cable Plug is in a Mode.
2	Protocol Layer creates the <i>Exit Mode</i> Command request and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>Exit Mode</i> Command request.	Physical Layer receives the <i>Exit Mode</i> Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>Exit Mode</i> Command request to the Protocol Layer.

Step	DFP	Cable Plug
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Exit Mode</i> Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Exit Mode</i> Command request was successfully sent. Policy Engine starts the <i>VDMModeExitTimer</i> .	
10		Policy Engine requests the Device Policy Manager to enter USB operation. <i>tCableMessage after the GoodCRC Message was sent</i> the Policy Engine tells the Protocol Layer to form an <i>Exit Mode</i> Command ACK response.
11		Protocol Layer creates the <i>Exit Mode</i> Command ACK response and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	Physical Layer receives the <i>Exit Mode</i> Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>Exit Mode</i> Command ACK response.
13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Exit Mode</i> Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>VDMModeExitTimer</i> and requests the Device Policy Manager to enter USB Operation.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Exit Mode</i> Command ACK response was successfully sent.
Both DFP and Cable Plug are in USB Operation		

### 8.3.2.9.8 UFP to DFP Attention

This is an example of Message sequence between a DFP and a UFP both utilizing the Type-C connector.

Figure 8-30 shows an example sequence between a Type-C DFP and a UFP, where the UFP requests attention from the DFP.

Figure 8-30 UFP to DFP Attention

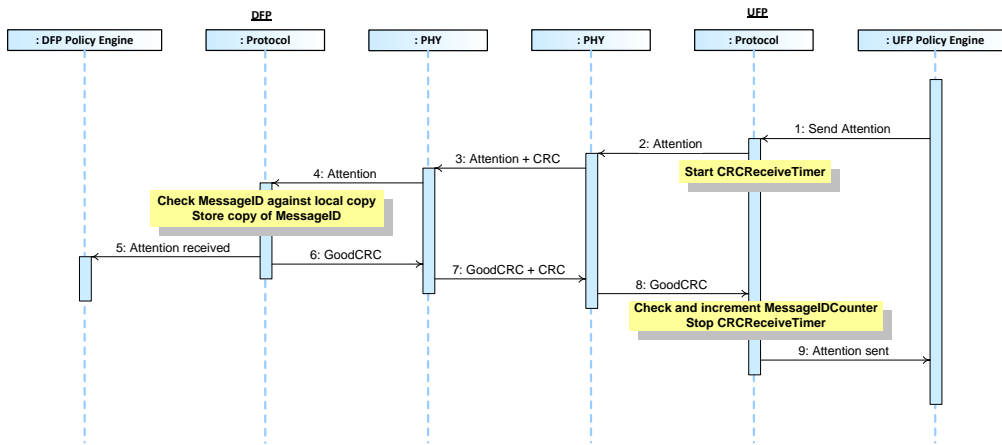


Table 8-29 below provides a detailed explanation of what happens at each labeled step in Figure 8-30 above.

Table 8-29 Steps for UFP to DFP Attention

Step	DFP	UFP
1		The Device Policy Manager requests attention. The Policy Engine tells the Protocol Layer to form an <b>Attention</b> Command request.
2		Protocol Layer creates the <b>Attention</b> Command request and passes to Physical Layer. Starts <b>CRCReceiveTimer</b> .
3	Physical Layer receives the <b>Attention</b> Command request and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <b>Attention</b> Command request.
4	Protocol Layer checks the <b>MessageID</b> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <b>Attention</b> Command request information to the Policy Engine that consumes it.	
5	The Policy Engine informs the Device Policy Manager	
6	Protocol Layer generates a <b>GoodCRC</b> Message and passes it Physical Layer.	
7	Physical Layer appends a CRC and sends the <b>GoodCRC</b> Message.	Physical Layer receives <b>GoodCRC</b> Message and compares the CRC it calculated with the one sent to verify the Message.
8		Physical Layer removes the CRC and forwards the <b>GoodCRC</b> Message to the Protocol Layer.

Step	DFP	UFP
9		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Attention</i> Command request was successfully sent.

### 8.3.2.9.9 Cable Plug to DFP Attention

This is an example of message sequence between a DFP and a Cable Plug both utilizing the Type-C connector. This shows an example sequence between a Type-C DFP and a Cable Plug, where the Cable Plug requests attention from the DFP.

**Figure – Cable Plug to DFP Attention**

below provides a detailed explanation of what happens at each labeled step in above.

**Table – Steps for Cable Plug to DFP Attention**

Step	DFP	Cable Plug
1		The Device Policy Manager requests attention. The Policy Engine tells the Protocol Layer to form an <b>Attention</b> Command request.
2		Protocol Layer creates the <b>Attention</b> Command request and passes to Physical Layer. Starts <b>CRCReceiveTimer</b> .
3	Physical Layer receives the <b>Attention</b> Command request and compares the CRC it calculated with the one sent to verify the message.	Physical Layer appends a CRC and sends the <b>Attention</b> Command request.
4	Protocol Layer checks the <b>MessageID</b> in the incoming message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <b>Attention</b> Command request information to the Policy Engine that consumes it.	
5	The Policy Engine informs the Device Policy Manager	
6	Protocol Layer generates a <b>GoodCRC</b> message and passes it Physical Layer.	
7	Physical Layer appends a CRC and sends the <b>GoodCRC</b> message.	Physical Layer receives <b>GoodCRC</b> message and compares the CRC it calculated with the one sent to verify the message.
8		Physical Layer removes the CRC and forwards the <b>GoodCRC</b> message to the Protocol Layer.
9		Protocol Layer verifies and increments the <b>MessageIDCounter</b> and stops <b>CRCReceiveTimer</b> . Protocol Layer informs the Policy Engine that the <b>Attention</b> Command request was successfully sent.

### 8.3.2.10 Built in Self-Test (BIST)

#### 8.3.2.10.1 BIST Receiver Mode

This is an example of a BIST Receiver Mode test between a Tester and a Unit Under Test (UUT).

Figure 8-31 shows the Messages as they flow across the bus and within the devices. This test verifies that the UUT can receive data with a performance according this specification.

Figure 8-31 BIST Receiver Mode test

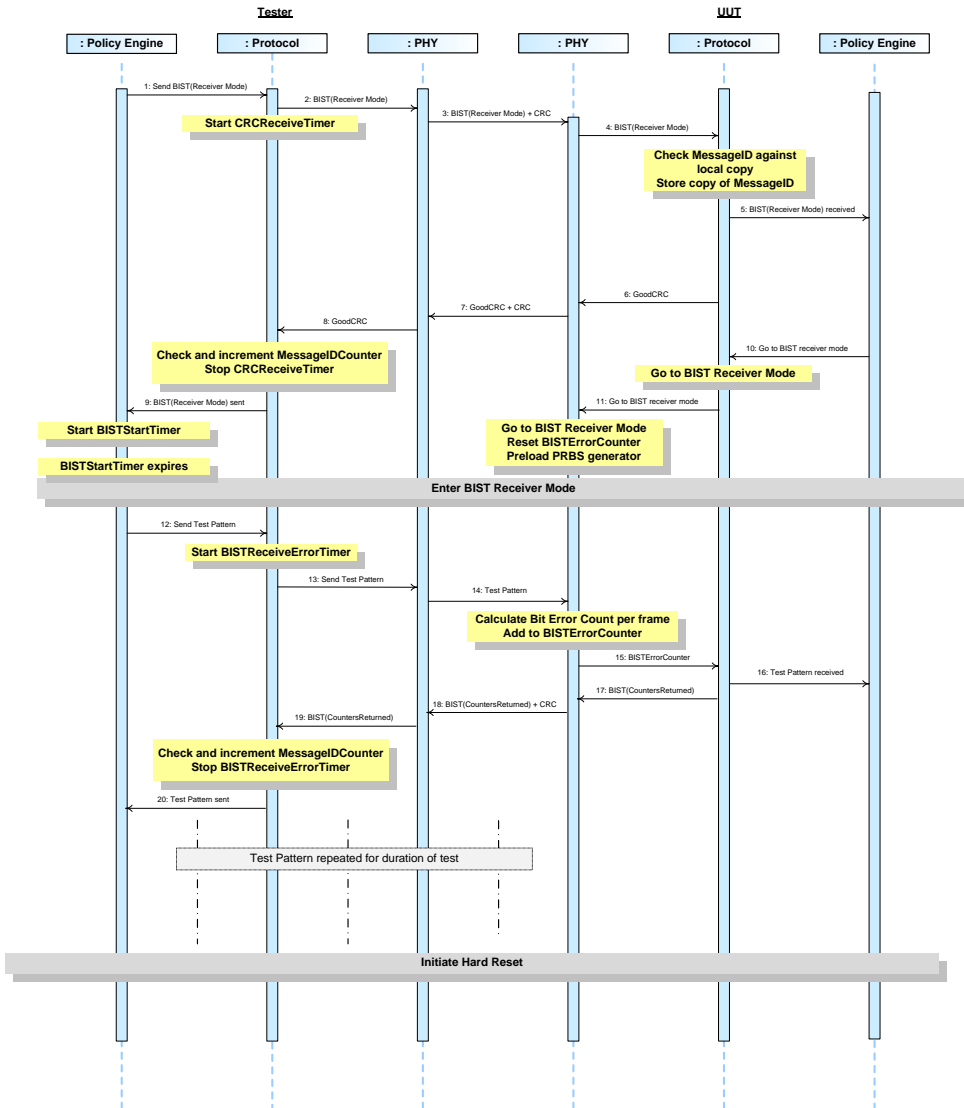




Table 8-30 Steps for BIST Receiver Mode test

Step	Tester	UUT
1	The Policy Engine directs the Protocol Layer to generate a <i>BIST</i> Message with <i>BIST</i> Data Object of <i>BIST Receiver Mode</i> to put the UUT into BIST receiver mode.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>BIST</i> Message.	Physical Layer receives the <i>BIST</i> Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>BIST</i> Message to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>BIST</i> Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>BIST</i> Message was successfully sent. The Policy Engine initializes and runs the <i>BISTStartTimer</i> .	
10		Policy Engine tells Protocol Layer to go into Receiver Test Mode. The Protocol Layer goes to BIST receiver mode.
11		Protocol Layer tells Physical Layer to go into BIST Receiver Mode. The Physical Layer goes to BIST Receiver Mode, resets the <i>BISTErrorCounter</i> and preloads the PRBS generator.
	UUT enters BIST Receiver Mode	
12	After the <i>BISTStartTimer</i> has expired the Policy Engine directs the Protocol Layer to generate a packet containing the next Test Frame.	
13	Protocol Layer creates the Message and passes to Physical Layer. Starts <i>BISTReceiveErrorTimer</i> .	
14	Physical Layer does not append a CRC and sends the Test Frame.	Physical Layer receives the Test Frame. The Physical Layer calculates the Bit Error Count for the Test Frame and adds this to the <i>BISTErrorCounter</i> .
15		Physical Layer forwards the <i>BISTErrorCounter</i> value to the Protocol Layer.
16		The Protocol Layer informs the Policy Engine that a Test Frame has been received.
17		Protocol Layer generates a <i>BIST</i> Message with <i>BIST</i> Data Object of <i>Returned BIST Counters</i> and passes it Physical Layer.
18	Physical Layer receives <i>BIST</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>BIST</i> Message.

Step	Tester	UUT
19	Physical Layer removes the CRC and forwards the <i>BIST</i> Message to the Protocol Layer.	
20	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>BISTReceiveErrorTimer</i> . Protocol Layer informs the Policy Engine that the Test Frame was successfully sent.	
	Steps 12 to 20 are repeated until <i>Hard Reset</i> Signaling is generated at which point the UUT returns to normal operation.	

### 8.3.2.10.2 BIST Transmit mode

This is an example of a BIST Transmit Mode test between a Tester and a UUT. Figure 8-32 shows the Messages as they flow across the bus and within the devices. This test verifies that the UUT transmitter can transmit with sufficient quality (see Section 6.4.3.2).

Figure 8-32 BIST Transmit Mode test

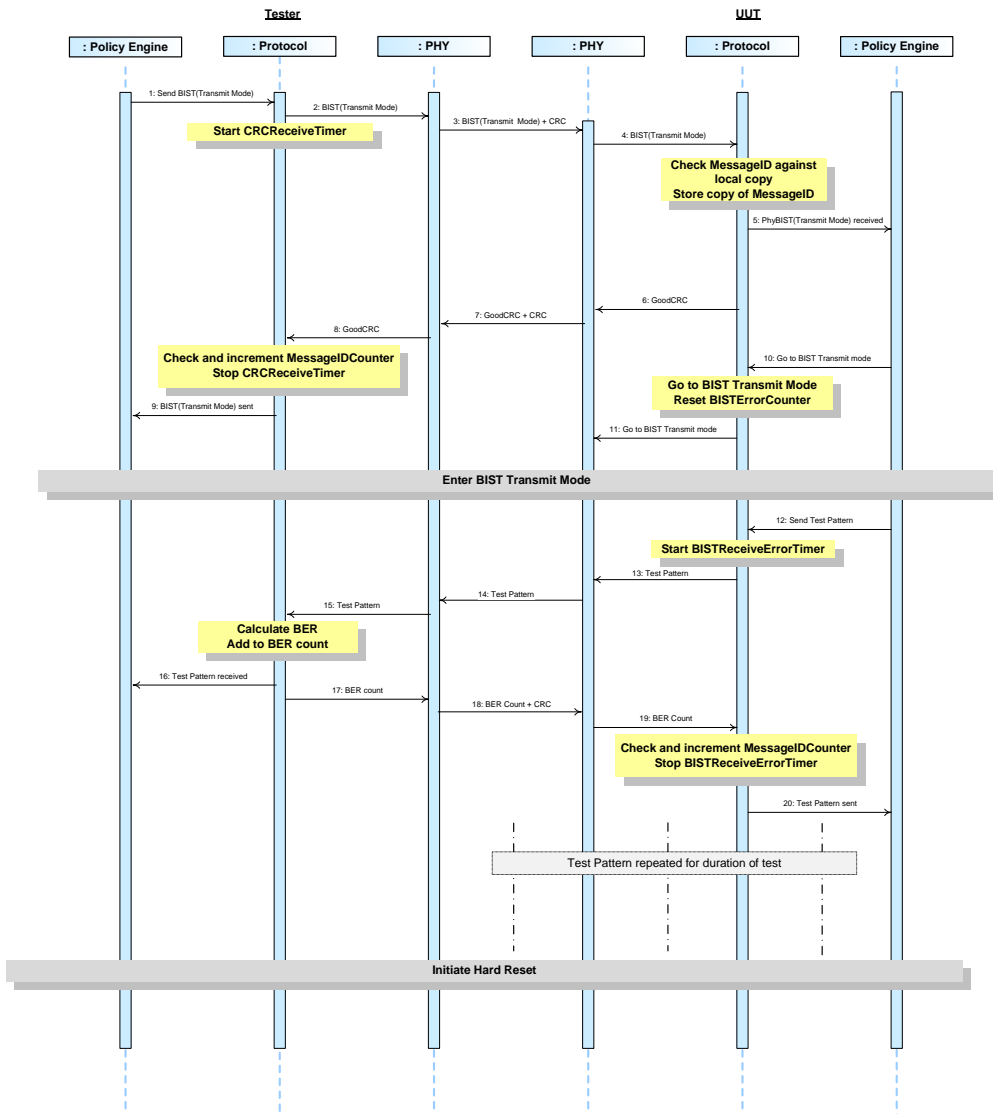


Table 8-31 Steps for BIST Transmit Mode test

Step	Tester	UUT
1	The Policy Engine directs the Protocol Layer to generate a <i>BIST</i> Message, with <i>BIST</i> Data Object of <i>BIST Transmit Mode</i> , to put the UUT into BIST Transmit Mode.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>BIST</i> Message.	Physical Layer receives the <i>BIST</i> Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>BIST</i> Message to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>BIST</i> Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>BIST</i> Message was successfully sent.	
10		Policy Engine tells Protocol Layer to go into BIST Transmit Mode. The Policy Engine goes to BIST Transmit mode and resets the <i>BISTErrorCounter</i> .
11		Protocol Layer tells Physical Layer to go into BIST Transmit Mode.
	UUT enters BIST Transmit Mode	
12		The Policy Engine directs the Protocol Layer to generate a packet containing the next Test Frame.
13		Protocol Layer creates the Test Frame and passes to Physical Layer. Starts <i>BISTReceiveErrorTimer</i> .
14	Physical Layer receives the Test Frame.	Physical Layer does not append a CRC and sends the Test Frame.
15	Physical Layer forwards the Test Frame to the Protocol Layer.	
16	The Protocol Layer forwards the received Test Frame information to the Policy Engine that consumes it.	
17	Protocol Layer generates a <i>BIST</i> Message with a <i>BIST</i> Data Object of <i>Returned BIST Counters</i> and passes it Physical Layer.	
18	Physical Layer appends CRC and sends the <i>BIST</i> Message.	Physical Layer receives the <i>BIST</i> Message and checks the CRC to verify the <i>BIST</i> Message.
19		Physical Layer removes the CRC and forwards the <i>BIST</i> Message to the Protocol Layer.

Step	Tester	UUT
20		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>BISTRceiveErrorTimer</i> . Protocol Layer informs the Policy Engine that the Test Frame was successfully sent.
	Steps 12 to 20 are repeated until <i>Hard Reset</i> Signaling is generated at which point the UUT returns to normal operation.	

### 8.3.2.10.3 BIST Test Patterns

In addition to the BIST transmit and receive Test Frames there are various Test Patterns which the UUT can be made to send continuously in order to perform measurements on the transmission spectrum, eye pattern etc. (see Section 5.9.1). The following is an example of a *BIST Eye Pattern* test between a Tester and a UUT but the sequence applies equally to other continuous Test Patterns. When the UUT is connected to the Tester the sequence below is executed.

Figure 8-33 shows the Messages as they flow across the bus and within the devices. This test verifies the eye pattern and the spectrum of the transmitted signal.

- 1) Connection is established and stable.
- 2) Tester sends a *BIST* Message with a *BIST Eye Pattern* *BIST* Data Object.
- 3) UUT answers with a *GoodCRC* Message.
- 4) UUT starts sending the Test Pattern.
- 5) Operator does the measurements.
- 6) Operator restarts the UUT. Note: that the method of operator restart for the UUT is outside the scope of this specification. This is assumed to be some mechanism whereby the operator restores the UUT to normal PD operation.

Refer to Sections 6.4.3.4 through 0.

Figure 8-33 BIST Eye Pattern Test

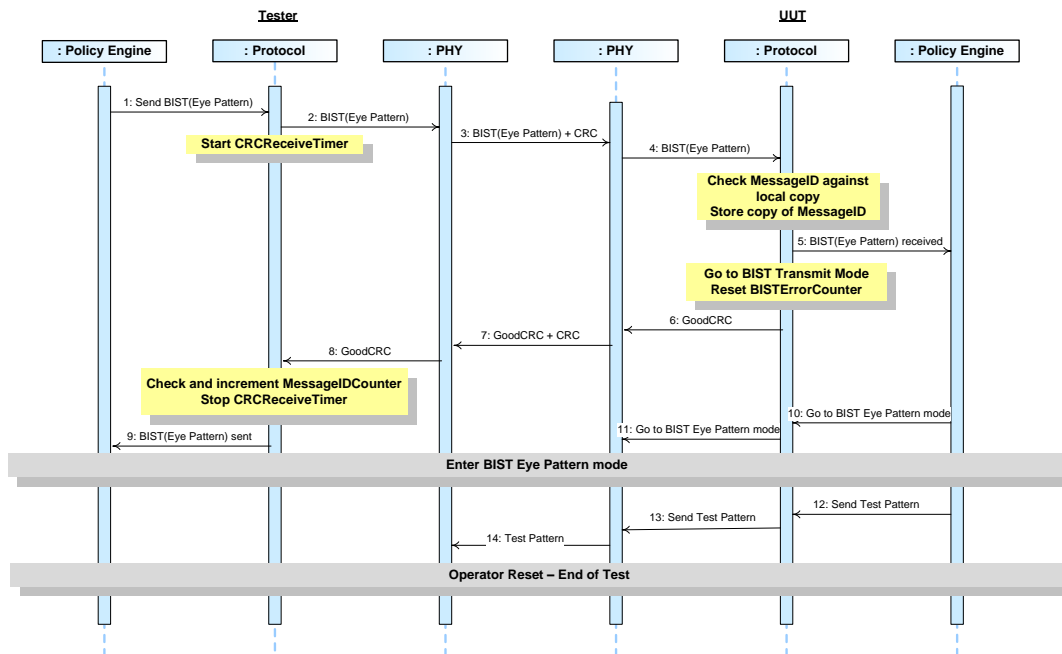


Table 8-32 Steps for BIST Eye Pattern Test

Step	Tester	UUT
1	The Policy Engine directs the Protocol Layer to generate a <i>BIST</i> Message, with a <i>BIST</i> Data Object of <i>BIST Eye Pattern</i> , to put the UUT into BIST Eye Pattern Test.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	Physical Layer appends CRC and sends the <i>BIST</i> Message.	Physical Layer receives the <i>BIST</i> Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>BIST</i> Message to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>BIST</i> Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>BIST</i> Message was successfully sent.	
10		Policy Engine tells Protocol Layer to go into BIST Eye Pattern Test Mode. The Policy Engine goes to BIST Eye Pattern Test Mode.
11		Protocol Layer tells Physical Layer to go into BIST Eye Pattern Test Mode.
	UUT enters BIST Eye Pattern Test Mode	
12		The Policy Engine directs the Protocol Layer to start generation of the Test Pattern.
13		Protocol Layer directs the PHY Layer to generate the Test Pattern.
14	Physical Layer receives the Test Pattern stream.	Physical Layer generates a continuous Test Pattern stream.
	Operator reset of UUT terminates the test	

### 8.3.3 State Diagrams

#### 8.3.3.1 Introduction to state diagrams used in Chapter 8

The state diagrams defined in Section 8.3.3 are normative and shall define the operation of the Power Delivery Policy Engine. Note that these state diagrams are not intended to replace a well written and robust design.

Figure 8-34 Outline of States

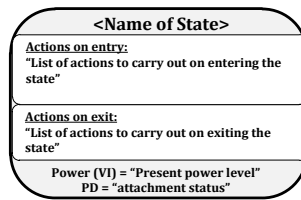


Figure 8-34 shows an outline of the states defined in the following sections. At the top there is the name of the state. This is followed by “Actions on entry” a list of actions carried out on entering the state. If there are also “Actions on exit” a list of actions carried out on exiting the state then these are listed as well; otherwise this box is omitted from the state. At the bottom the status of PD is listed:

- “Power” which indicates the present output power for a Source Port or input power for a Sink Port.
- “PD” which indicates the present attachment status either “attached”, “unattached”, or “unknown”.

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 “&”.

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 (e.g. Source Port or Sink Port state diagrams). Figure 8-35 indicates how such references are made. The reference is indicated with a hatched box. The box contains the name of the state and whether the state is a DFP or UFP. It has also been necessary to indicate conditional entry to either Source Port or Sink Port state diagrams. This is achieved by the use of a bulleted list indicating the pre-conditions (see example in Figure 8-36). It is also possible that the entry and return states are the same. Figure 8-37 indicates a state reference where each referenced state corresponds to either the entry state or the exit state.

Figure 8-35 References to states

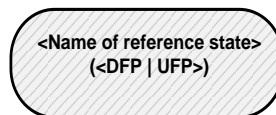


Figure 8-36 References to states Example of state reference with conditions

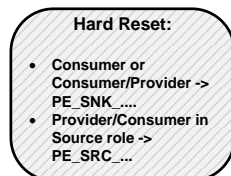
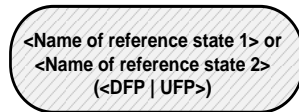




Figure 8-37 Example of state reference with conditions

~~Figure— Example of state reference with~~ the same entry and exit



Timers are included in many of the states. Timers are initialized (set to their starting condition) and run (timer is counting) in the particular state it is referenced. As soon as the state is exited then the timer is no longer active. Where the timers continue to run outside of the state (such as the *NoResponseTimer*), this is called out in the text. Timeouts of the timers are listed as conditions on state transitions.

Conditions listed on state transitions will come from one of three sources and, when there is a conflict, should be serviced in the following order:

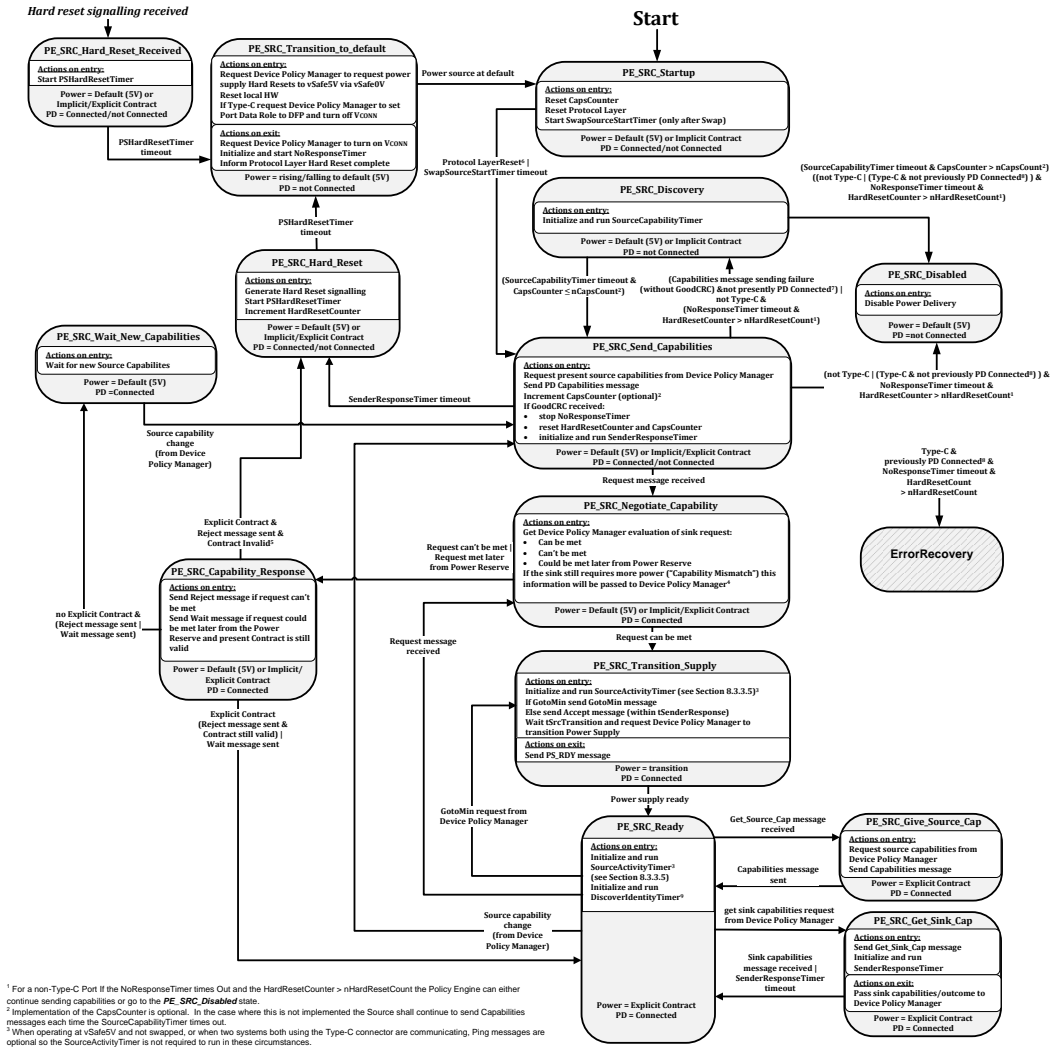
1. Message and related indications passed up to the Policy Engine from the Protocol Layer (Message sent, Message received etc.)
2. Events triggered within the Policy Engine e.g. timer timeouts.
3. Information and requests coming from the Device Policy manager relating either to Local Policy, or to other modules which the Device Policy Manager controls such as power supply and Cable Detection.

Note: The following state diagrams are not intended to cover all possible corner cases that may be encountered. For example where an outgoing Message is discarded, due to an incoming Message by the Protocol Layer (see Section 6.8.2.2) it will be necessary for the higher layers of the system to handle a retry of the Message sequence that was being initiated, after first handling the incoming Message.

### 8.3.3.2 Policy Engine Source Port State Diagram

Figure 8-38 below shows the state diagram for the Policy Engine in a Source Port. The following sections describe operation in each of the states.

Figure 8-38 Source Port Policy Engine state diagram



<sup>1</sup> For a non-Type-C Port if the NoResponseTimer times Out and the HardResetCounter > nHardResetCount the Policy Engine can either continue sending capabilities or go to the **PE\_SRC\_Disabled** state.  
<sup>2</sup> Implementation of the CapsCounter is optional. In the case where this is not implemented the Source shall continue to send Capabilities messages each time the SourceCapabilityTimer times out.  
<sup>3</sup> When operating at vSafeV and not swapped, or when two systems both using the Type-C connector are communicating, Ping messages are optional so the SourceActivityTimer is not required to run in these circumstances.  
<sup>4</sup> Since the Sink is required to make a valid request from the offered capabilities the expected transition is via "Request can be met" unless the Source capabilities have changed since the last offer.  
<sup>5</sup> "Contract invalid" means that the previously negotiated Voltage and Current values are no longer included in the Source's new Capabilities. If the Sink fails to make a valid Request in this case then Power Delivery operation is no longer possible and Power Delivery mode is exited with a Hard Reset.  
<sup>6</sup> After a Power Swap the new Source is required to wait an additional tSwapSourceStart before sending Source Capabilities. This delay is not required when first starting up a system.  
<sup>7</sup> PD Connected is defined as a situation when the Port Partners are actively communicating. The Port Partners remain PD Connected after a Swap until there is a transition to Disabled or the connector is able to identify a disconnect (Type-C, Type-A with insert detect, Micro-AB).  
<sup>8</sup> Port Partners are no longer PD Connected after a Hard Reset but for Type-C connections consideration needs to be given as to whether there has been a PD Connection while the Ports have been attached to prevent unnecessary Type-C disconnects.  
<sup>9</sup> The DiscoverIdentifyTimer is run when this is a DFP and a PD Connection with a Cable Plug needs to be established i.e. no GoodCRC has yet been received in response to a DiscoverIdentify message.

#### 8.3.3.2.1 PE\_SRC\_Startup state

**PE\_SRC\_Startup** shall be the starting state for a Source Policy Engine either on power up or after a Hard Reset. On entry to this state the Policy Engine shall reset the **CapsCounter** and reset the Protocol Layer. Note that resetting the Protocol Layer will also reset the **MessageIDCounter** and stored **MessageID** (see Section 6.8.2.3).

The Policy Engine shall transition to the **PE\_SRC\_Send\_Capabilities** state:

- When the Protocol Layer reset has completed if the **PE\_SRC\_Startup** state was entered due to the system first starting up.
- When the **SwapSourceStartTimer** times out if the **PE\_SRC\_Startup** state was entered as the result of a Power Role Swap.

Note: Providers or Provider/Consumers with an insertion detection mechanism (Type-C, Standard ~~AAA~~ insertion detect or Micro-A ID pin or Attach Detection Protocol see [USBOTG 2.0]) and without a plug attached shall remain in the **PE\_SRC\_Startup** state, without sending any **Source\_Capabilities** Messages until a plug is attached.

#### 8.3.3.2.2 PE\_SRC\_Discovery state

On entry to the **PE\_SRC\_Discovery** state the Policy Engine shall initialize and run the **SourceCapabilityTimer** in order to trigger sending a **Source\_Capabilities** Message.

The Policy Engine shall transition to the **PE\_SRC\_Send\_Capabilities** state when:

- The **SourceCapabilityTimer** times out and **CapsCounter**  $\leq$  **nCapsCount**.

The Policy Engine may optionally go to the **PE\_SRC\_Disabled** state when:

- ~~• This is not a Type-C Port.~~
- ~~• The Port Partners are not presently PD Connected~~
- And the **SourceCapabilityTimer** times out
- And **CapsCounter**  $>$  **nCapsCount**.

~~The Policy Engine shall go to the **PE\_SRC\_Disabled** state when:~~

- ~~• The Port is not a Type-C Port~~
- ~~• Or the Port is a Type-C Port and the Port Partners have not been PD Connected (the Type-C Port remains attached to a Port it has not had a PD Connection with during this attachment)~~
- And the **NoResponseTimer** times out
- And the **HardResetCounter**  $>$  **nHardResetCount**

~~–Note in the **PE\_SRC\_Disabled** state the attached device is assumed to be unresponsive. The Policy Engine operates as if the device is unattached until such time as a detach/reattach is detected.~~

The Policy Engine shall go to the **ErrorRecovery** state when:

- ~~• This The Port is a Type-C Port.~~
- ~~• And the Port Partners have previously been PD Connected (the Type-C Port remains attached to a Port it has had a PD Connection with during this attachment)~~
- And the **NoResponseTimer** times out.
- And the **HardResetCounter**  $>$  **nHardResetCount**.

#### 8.3.3.2.3 PE\_SRC\_Send\_Capabilities state

~~A capabilities change event from the Device Policy Manager while in any state when connected shall trigger the Policy Engine to enter the state. There is also a transition~~Note: this state may be entered from the **PE\_SRC\_Soft\_Reset** state.

On entry to the **PE\_SRC\_Send\_Capabilities** state the Policy Engine shall request the present Port capabilities from the Device Policy Manager. The Policy Engine shall then request the Protocol Layer to send a **Source\_Capabilities** Message containing these capabilities and increment the **CapsCounter** (if implemented).

If a **GoodCRC** Message is received then the Policy Engine shall:

- Stop the *NoResponseTimer* .
- Reset the *HardResetCounter* and *CapsCounter* to zero. Note that the *HardResetCounter* shall only be set to zero in this state and at power up; its value shall be maintained during a Hard Reset.
- Initialize and run the *SenderResponseTimer*.

Once a *Source\_Capabilities* Message has been received and acknowledged by a *GoodCRC* Message, the Sink is required to then send a *Request* Message within *tSenderResponse*.

The Policy Engine shall transition to the *PE\_SRC\_Negotiate\_Capability* state when:

- A *Request* Message is received from the Sink.

The Policy Engine shall transition to the *PE\_SRC\_Discovery* state when:

- The Protocol Layer indicates that the Message has not been sent and we are presently not Connected. This is part of the Capabilities sending process whereby successful Message sending indicates connection to a PD Sink Port.

The Policy Engine shall transition to the *PE\_SRC\_Hard\_Reset* state when:

- The *SenderResponseTimer* times out. In this case a transition back to USB Default Operation is required.

When:

- ~~This~~The Port is not a Type-C Port
- Or the Port is a Type-C Port and the Port Partners have not been PD Connected (the Type-C Port remains attached to a Port it has not had a PD Connection with during this attachment)
- And the *NoResponseTimer* times out
- And the *HardResetCounter* > *nHardResetCount*

The Policy Engine shall do one of the following:

- Transition to the *PE\_SRC\_Discovery* state.
- Transition to the *PE\_SRC\_Disabled* state.

Note that in either case the attached device is assumed to be unresponsive. The Policy Engine should operate as if the device is unattached until such time as a detach/reattach is detected.

The Policy Engine shall go to the *ErrorRecovery* state when:

- ~~This~~The Port is a Type-C Port;
- And the Port Partners have previously been PD Connected (the Type-C Port remains attached to a Port it has had a PD Connection with during this attachment)
- And the *NoResponseTimer* times out.
- And the *HardResetCounter* > *nHardResetCount*.

#### 8.3.3.2.4 PE\_SRC\_Negotiate\_Capability state

On entry to the *PE\_SRC\_Negotiate\_Capability* state the Policy Engine shall ask the Device Policy Manager to evaluate the Request from the attached Sink. The response from the Device Policy Manager shall be one of the following:

- The Request can be met.
- The Request cannot be met
- The Request could be met later from the Power Reserve.

The Policy Engine shall transition to the *PE\_SRC\_Transition\_Supply* state when:

- The Request can be met.

The Policy Engine shall transition to the *PE\_SRC\_Capability\_Response* state when:

- The Request cannot be met.
- Or the Request can be met later from the Power Reserve.

#### 8.3.3.2.5 PE\_SRC\_Transition\_Supply state

The Policy Engine shall be in the **PE\_SRC\_Transition\_Supply** state while the power supply is transitioning from one power to another.

On entry to the **PE\_SRC\_Transition\_Supply** state, the Policy Engine shall initialize and run the **SourceActivityTimer** (see Section 8.3.3.6 for details of **Ping** messaging for Source Ports), request the Protocol Layer to either send a **GotoMin** Message (if this was requested by the Device Policy Manager) or otherwise an **Accept** Message, wait for **tSrcTransition**, and inform the Device Policy Manager that it shall transition the power supply to the Requested power level. Note: that if the power supply is currently operating at the requested power no change will be necessary.

On exit from the **PE\_SRC\_Transition\_Supply** state the Policy Engine shall request the Protocol Layer to send a **PS\_RDY** Message.

The Policy Engine shall transition to the **PE\_SRC\_Ready** state when:

- The Device Policy Manager informs the Policy Engine that the power supply is ready.

#### 8.3.3.2.6 PE\_SRC\_Ready state

In the **PE\_SRC\_Ready** state the PD Source shall operating at a stable power with no ongoing negotiation. It shall respond to requests from the Sink, events from the Device Policy Manager and shall send out **Ping** Messages to maintain the PD link if required (see Section 6.3.5).

On entry to the **PE\_SRC\_Ready** state the Policy Engine shall initialize and run the **SourceActivityTimer** (see Section 8.3.3.6 for details of **Ping** messaging for Source Ports). If this is a DFP which needs to establish communication with a Cable Plug, the DFP shall initialize and run the **DiscoverIdentityTimer** (no **GoodCRC** Message response yet received to **Discover Identity Message**).

The Policy Engine shall transition to the **PE\_SRC\_Send\_Capabilities** state when:

- The Device Policy Manager indicates that Source Capabilities have changed.

The Policy Engine shall transition to the **PE\_SRC\_Transition\_Supply** state when:

- A **GotoMin** request is received from the Device Policy Manager for the attached Device to go to minimum power.

The Policy Engine shall transition to the **PE\_SRC\_Give\_Source\_Cap** state when:

- A **Get\_Source\_Cap** Message is received from the Sink.

The Policy Engine shall transition to the **PE\_SRC\_Get\_Sink\_Cap** state when:

- The Device Policy Manager asks for the Sink's capabilities.

#### 8.3.3.2.7 PE\_SRC\_Disabled state

In the **PE\_SRC\_Disabled** state the PD Source supplies default power and is unresponsive to USB Power Delivery messaging, but not to **Hard Reset** Signaling.

#### 8.3.3.2.8 PE\_SRC\_Capability\_Response state

The Policy Engine shall enter the **PE\_SRC\_Capability\_Response** state if there is a Request received from the Sink that cannot be met based on the present capabilities. When the present Contract is not within the present capabilities it is regarded as invalid and a Hard Reset will be triggered.

On entry to the **PE\_SRC\_Capability\_Response** state the Policy Engine shall request the Protocol Layer to send one of the following:

- **Reject** Message – if the request cannot be met or the present Contract is invalid.
- **Wait** Message – if the request could be met later from the Power Reserve. A **Wait** Message shall not be sent if the present Contract is invalid.

The Policy Engine shall transition to the *PE\_SRC\_Ready* state when:

- There is an Explicit Contract and
- A *Reject* Message has been sent and the present Contract is still valid or
- A *Wait* Message has been sent.

The Policy Engine shall transition to the *PE\_SRC\_Hard\_Reset* state when:

- There is an Explicit Contract and
- The *Reject* Message has been sent and the present Contract is invalid (i.e. the Sink had to request a new value so instead we will return to USB Default Operation).

The Policy Engine shall transition to the *PE\_SRC\_Wait\_New\_Capabilities* state when:

- There is no Explicit Contract and
- A *Reject* Message has been sent or
- A *Wait* Message has been sent.

Note: The Policy Engine of a Consumer/Provider, acting as the Source, transitions to the *PE\_SNK\_Hard\_Reset* state as described in the Section 8.3.3.6.1.4.

#### 8.3.3.2.9 PE\_SRC\_Hard\_Reset state

The Policy Engine shall transition from any state to the *PE\_SRC\_Hard\_Reset* state when

- The *NoResponseTimer* times out and the *HardResetCounter*  $\leq$  *nHardResetCount* or
- The Device Policy Manager requests a Hard Reset.

On entry to the *PE\_SRC\_Hard\_Reset* state the Policy Engine shall request the generation of *Hard Reset* Signaling by the PHY Layer, initialize and run the *PSHardResetTimer* and increment the *HardResetCounter*. Note that the *NoResponseTimer* shall continue to run in every state until it is stopped or times out.

The Policy Engine shall transition to the *PE\_SRC\_Transition\_to\_default* state when:

- The *PSHardResetTimer* times out.
- ~~PE\_SRC\_Hard\_Reset is complete.~~

#### 8.3.3.2.10 ~~PE\_SRC\_Transition\_to\_default\_Received~~ state

The Policy Engine shall transition from any state to the ~~*PE\_SRC\_Hard\_Reset\_Received*~~ state when:

- *Hard Reset* Signaling is detected.

On entry to the *PE\_SRC\_Hard\_Reset\_Received* state the Policy Engine shall initialize and run the *PSHardResetTimer*.

The Policy Engine shall transition to the *PE\_SRC\_Transition\_to\_default* state when:

- The *PSHardResetTimer* times out.

#### 8.3.3.2.11 PE\_SRC\_Transition\_to\_default state

On entry to the *PE\_SRC\_Transition\_to\_default* state the Policy Engine shall:

- indicate to the Device Policy Manager that the power supply shall *Hard Reset* (see Section 7.1.6 ~~transition to default~~). ~~The Policy Engine shall then -~~
- request a reset of the local hardware -
- for a Type-C connector shall request the Device Policy Manager to set the Port Data Role to DFP and turn off *VCONN*.

On exit from the *PE\_SRC\_Transition\_to\_default* state the Policy Engine shall:

- for a Type-C connector shall request the Device Policy Manager to turn on *VCONN*

- initialize and run the *NoResponseTimer* and ~~inform the Protocol Layer that the Hard Reset is complete~~. Note that the *NoResponseTimer* shall continue to run in every state until it is stopped or times out.
- ~~inform the Protocol Layer that the Hard Reset is complete.~~

The Policy Engine shall transition to the *PE\_SRC\_Startup* state when:

- The Device Policy Manager indicates that the power supply has reached the default level.

#### ~~8.3.3.2.11~~ 8.3.3.2.12 PE\_SRC\_Give\_Source\_Cap state

On entry to the *PE\_SRC\_Give\_Source\_Cap* state the Policy Engine shall request the Device Policy Manager for the current system capabilities. The Policy Engine shall then request the Protocol Layer to send a *Source\_Capabilities* Message containing these capabilities.

The Policy Engine shall transition to the *PE\_SRC\_Ready* state when:

- The *Source\_Capabilities* Message has been successfully sent.

#### ~~8.3.3.2.12~~ 8.3.3.2.13 PE\_SRC\_Get\_Sink\_Cap state

In this state the Policy Engine, due to a request from the Device Policy Manager, shall request the capabilities from the attached Sink.

On entry to the *PE\_SRC\_Get\_Sink\_Cap* state the Policy Engine shall request the Protocol Layer to send a *Get\_Sink\_Cap* Message in order to retrieve the Sink's capabilities. The Policy Engine shall then start the *SenderResponseTimer*.

On exit from the *PE\_SRC\_Get\_Sink\_Cap* state the Policy Engine shall inform the Device Policy Manager of the outcome (capabilities or response timeout).

The Policy Engine shall transition to the *PE\_SRC\_Ready* state when:

- A *Sink\_Capabilities* Message is received.
- Or *SenderResponseTimer* times out.

#### 8.3.3.2.14 PE\_SRC\_Wait\_New\_Capabilities

In this state the Policy Engine has been unable to negotiate an Explicit Contract and is waiting either new Capabilities from the Device Policy Manager.

The Policy Engine shall transition to the *PE\_SRC\_Send\_Capabilities* state when:

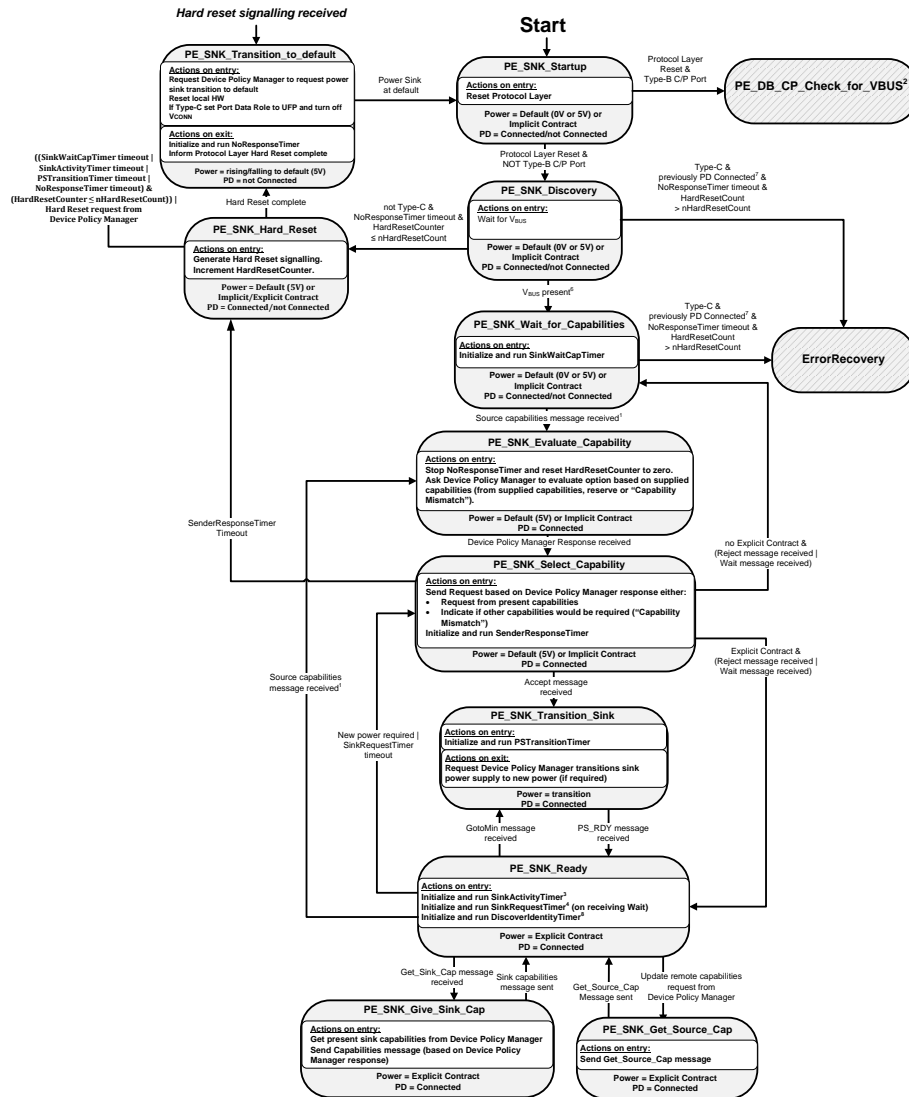
- The Device Policy Manager indicates that Source Capabilities have changed.

### 8.3.3.3 Policy Engine Sink Port State Diagram

Figure 8-39 below shows the state diagram for the Policy Engine in a Sink Port. The following sections describe operation in each of the states.

Figure 8-39 Sink Port state diagram





<sup>1</sup> Source capabilities messages received in states other than PE\_SNK\_Wait\_For\_Capabilities and PE\_SNK\_READY constitute a Protocol Error.  
<sup>2</sup> The NoResponseTimer will be stopped upon entry to the PE\_DB\_CP\_Check\_for\_VBUS state.  
<sup>3</sup> The SinkActivityTimer shall not be run when operating at vSafe5V or when two systems using the Type-C connector are communicating, since Ping messages are optional.  
<sup>4</sup> The SinkRequestTimer should not be stopped if a Ping message is received in the PE\_SNK\_Ready state since it represents the maximum time between requests after a Wait which is not reset by a Ping message.  
<sup>5</sup> For Type-C connectors error recovery steps can be taken at this point which are defined in the USB Type-C specification and are outside the scope of this specification.  
<sup>6</sup> During a Hard Reset the Source voltage will transition to vSafe0V and then transition to vSafe5V. Sinks need to ensure that Vbus present is not indicated until after the Source has completed the Hard Reset process by detecting both of these transitions.  
<sup>7</sup> PD Connected is defined as a situation when the Port Partners have exchanged a Message and GoodCRC response. The Port Partners remain PD Connected after a Swap or the connector is able to identify a disconnect (Type-C, Type-A with insert detect, Micro-AB).  
<sup>8</sup> The DiscoverIdentityTimer is run when this is a DFP and a PD Connection with a Cable Plug needs to be established i.e. no GoodCRC has yet been received in response to a DiscoverIdentity message.

#### 8.3.3.3.1 PE\_SNK\_Startup state

**PE\_SNK\_Startup** shall be the starting state for a Sink Policy Engine either on power up or after a Hard Reset. On entry to this state the Policy Engine shall reset the Protocol Layer. Note that resetting the Protocol Layer will also reset the **MessageIDCounter** and stored **MessageID** (see Section 6.8.2.3).

Once the reset process completes, the Policy Engine shall transition to the **PE\_SNK\_Discovery** state for a Consumer only and to the **PE\_DB\_CP\_Check\_for\_VBUS** state for a Type-B Consumer/Provider.

#### 8.3.3.3.2 PE\_SNK\_Discovery state

In the **PE\_SNK\_Discovery** state the Sink Policy Engine waits for  $V_{BUS}$  to be present.

The Policy Engine shall transition to the **PE\_SNK\_Wait\_for\_Capabilities** state when:

- The Device Policy Manager indicates that  $V_{BUS}$  has been detected.

The Policy Engine shall transition to the **ErrorRecovery** state when:

- The Port is a Type-C connector and
- The Port Partners have previously been PD Connected (the Type-C Port remains attached to a Port it has had a PD Connection with during this attachment) and
- There has been a **NoResponseTimer** timeout and
- The **HardResetCounter** > **nHardResetCount**.

The Policy Engine shall transition to the **PE\_SNK\_Hard\_Reset** state when:

- The Port is not a Type-C connector and
- There has been a **NoResponseTimer** timeout and
- The **HardResetCounter** ≤ **nHardResetCount**.

#### 8.3.3.3.3 PE\_SNK\_Wait\_for\_Capabilities state

On entry to the **PE\_SNK\_Wait\_for\_Capabilities** state the Policy Engine shall initialize and start the **SinkWaitCapTimer**.

The Policy Engine shall transition to the **PE\_SNK\_Evaluate\_Capability** state when:

- A **Source\_Capabilities** ~~Source capabilities are~~ **Message** is received.

The Policy Engine shall transition to the **ErrorRecovery** state when:

The Policy Engine shall transition to the **ErrorRecovery** ~~This~~ state when:

- The Port is a Type-C connector and
- The Port Partners have previously been PD Connected (the Type-C Port remains attached to a Port it has had a PD Connection with during this attachment) and
- There has been a **NoResponseTimer** timeout and
- The **HardResetCounter** > **nHardResetCount**.

When the **SinkWaitCapTimer** times out, the Policy Engine will perform a Hard Reset.

#### 8.3.3.3.4 PE\_SNK\_Evaluate\_Capability state

Whenever the Policy Engine receives a message (except in the ~~and~~ states or during a Soft Reset) it shall transition to the **PE\_SNK\_Evaluate\_Capability** state is first entered when the Sink receives its first **Source\_Capabilities Message** from the Source. At this point the Sink knows that it is attached to and communicating with a PD capable Source.

On entry to the **PE\_SNK\_Evaluate\_Capability** state the Policy Engine shall request the Device Policy Manager to evaluate the supplied Source capabilities based on Local Policy. The Device Policy Manager shall indicate to the Policy Engine which new power level is required:

- A selection from the present offered capabilities is to be made.
- Capability mismatch; offered power does not meet the device's requirements.

The Policy Engine shall transition to the **PE\_SNK\_Select\_Capability** state when:

- A response is received from the Device Policy Manager.

#### 8.3.3.3.5 PE\_SNK\_Select\_Capability state

On entry to the **PE\_SNK\_Select\_Capability** state the Policy Engine shall request the Protocol Layer to send a response Message, based on the evaluation from the Device Policy Manager. The Message shall be one of the following:

- A Request from the offered Source Capabilities.
- A Request from the offered Source Capabilities with an indication that another power level would be preferred ("Capability Mismatch" bit set).

The Policy Engine shall initialize and run the **SenderResponseTimer**.

The Policy Engine shall transition to the **PE\_SNK\_Transition\_Sink** state when:

- An **Accept** Message is received from the Source.

The Policy Engine shall transition to the **PE\_SNK\_Wait\_for\_Capabilities** state when:

- There is no Explicit Contract in place and
- A **Reject** Message is received from the Source or
- A **Wait** Message is received from the Source.

The Policy Engine shall transition to the **PE\_SNK\_Ready** state when:

- There is an Explicit Contract in place and
- A **Reject** Message is received from the Source-or
- A **Wait** Message is received from the Source.

The Policy Engine shall transition to the **PE\_SNK\_Hard\_Reset** state when:

- A **SenderResponseTimer** timeout occurs.

Note: The Policy Engine of the Provider/Consumer, acting as the Sink, transitions to the **PE\_SRC\_Hard\_Reset** state as described in the Section 8.3.3.6.1.3.

#### 8.3.3.3.6 PE\_SNK\_Transition\_Sink state

On entry to the **PE\_SNK\_Transition\_Sink** state the Policy Engine shall initialize and run the **PSTransitionTimer** (timeout will lead to a Hard Reset see Section 8.3.3.3.8 and shall then request the Device Policy Manager to transition the Sink's power supply to the new power level. Note that if there is no power level change the Device Policy Manager should not affect any change to the power supply.

On exit from the **PE\_SNK\_Transition\_Sink** state the Policy Engine shall request the Device Policy Manager to transition the Sink's power supply to the new power level.

The Policy Engine shall transition to the **PE\_SNK\_Ready** state when:

- A **PS\_RDY** Message is received from the Source.

#### 8.3.3.3.7 PE\_SNK\_Ready state

In the **PE\_SNK\_Ready** state the PD Sink shall be operating at a stable power level with no ongoing negotiation. It shall respond to requests from the Source, events from the Device Policy Manager and may monitor for **Ping** Messages to maintain the PD link.

On entry to the **PE\_SNK\_Ready** state as the result of a wait the Policy Engine should do the following:

- Initialize and run the **SinkRequestTimer**.

On entry to the *PE\_SNK\_Ready* state the Policy Engine shall do the following:

- Initialize and run the *SinkActivityTimer* (see Section 6.5.3.2 for exceptions).

If this is a DFP which needs to establish communication with a Cable Plug, the DFP shall:

- Initialize and run the *DiscoverIdentityTimer* (no *GoodCRC* Message response yet received to *Discover Identity* Message).

The Policy Engine shall transition to the *PE\_SNK\_Evaluate\_Capability* state when:

- A *Source\_Capabilities* Message is received.

The Policy Engine shall transition to the *PE\_SNK\_Select\_Capability* state when:

- A new power level is requested by the Device Policy Manager.
- A *SinkRequestTimer* timeout occurs.

The Policy Engine shall transition to the *PE\_SNK\_Transition\_Sink* state when:

- A *GotoMin* Message is received.

The Policy Engine shall transition to the *PE\_SNK\_Hard\_Reset* state when:

- A *SinkActivityTimer* timeout occurs.

The Policy Engine shall transition back to the *PE\_SNK\_Ready* state when:

- A *Ping* Message is received. Note this should not cause the *SinkRequestTimer* to be reinitialized.

The Policy Engine shall transition to the *PE\_SNK\_Give\_Sink\_Cap* state when:

- A *Get\_Sink\_Cap* Message is received from the Protocol Layer.

The Policy Engine shall transition to the *PE\_SNK\_Get\_Source\_Cap* state when:

- The Device Policy Manager requests an update of the remote Source's capabilities.

#### 8.3.3.3.8 PE\_SNK\_Hard\_Reset state

The Policy Engine shall transition to the *PE\_SNK\_Hard\_Reset* state from any state when:

- ((*SinkWaitCapTimer* timeout |
- *SinkActivityTimer* timeout |
- *PSTransitionTimer* timeout |
- *NoResponseTimer* timeout) &
- (*HardResetCounter* ≤ *nHardResetCount*)) |
- Hard Reset request from Device Policy Manager

Note: if the *NoResponseTimer* times out and the *HardResetCounter* is greater than *nHardResetCount* the Sink shall assume that the Source is non-responsive.

Note: The *nHardResetCount* is reset on a power cycle or detach.

On entry to the *PE\_SNK\_Hard\_Reset* state the Policy Engine shall request the generation of *Hard Reset* Signaling by the PHY Layer and increment the *HardResetCounter*.

The Policy Engine shall transition to the *PE\_SNK\_Transition\_to\_default* state when:

- The Hard Reset is complete.

#### 8.3.3.3.9 PE\_SNK\_Transition\_to\_default state

The Policy Engine shall transition from any state to *PE\_SNK\_Transition\_to\_default* state when:

- **Hard Reset** Signaling is detected.

When **Hard Reset** Signaling is ~~detected or message sending in the Protocol Layer fails after retries (i.e. a message is not received) or transmitted~~ then the Policy Engine shall transition from any state to **PE\_SNK\_Transition\_to\_default**. This state can also be entered from the **PE\_SNK\_Hard\_Reset** state.

On entry to the **PE\_SNK\_Transition\_to\_default** state the Policy Engine shall:

- indicate to the Device Policy Manager that the Sink shall transition to default
- ~~The Policy Engine shall then~~ request a reset of the local hardware
- ~~for a Type-C connector shall request that the Port Data Role is set to UFP.~~

On exit from the **PE\_SNK\_Transition\_to\_default** state the Policy Engine shall initialize and run the **NoResponseTimer** and inform the Protocol Layer that the Hard Reset is complete. Note that the **NoResponseTimer** shall continue to run in every state until it is stopped or times out.

The Policy Engine shall transition to the **PE\_SNK\_Startup** state when:

- The Device Policy Manager indicates that the Sink has reached the default level.

#### 8.3.3.3.10 PE\_SNK\_Give\_Sink\_Cap state

On entry to the **PE\_SNK\_Give\_Sink\_Cap** state the Policy Engine shall request the Device Policy Manager for the current system capabilities. The Policy Engine shall then request the Protocol Layer to send a **Sink\_Capabilities** Message containing these capabilities.

The Policy Engine shall transition to the **PE\_SNK\_Ready** state when:

- The **Sink\_Capabilities** Message has been successfully sent.

#### 8.3.3.3.11 PE\_SNK\_Get\_Source\_Cap state

In the **PE\_SNK\_Get\_Source\_Cap** state the Policy Engine, due to a request from the Device Policy Manager, shall request the capabilities from the attached Source.

On entry to the **PE\_SNK\_Get\_Source\_Cap** state the Policy Engine shall request the Protocol Layer to send a **Get\_Source\_Cap** Message in order to retrieve the Source's capabilities.

The Policy Engine shall transition to the **PE\_SNK\_Ready** state when:

- The **Get\_Source\_Cap** Message is sent.

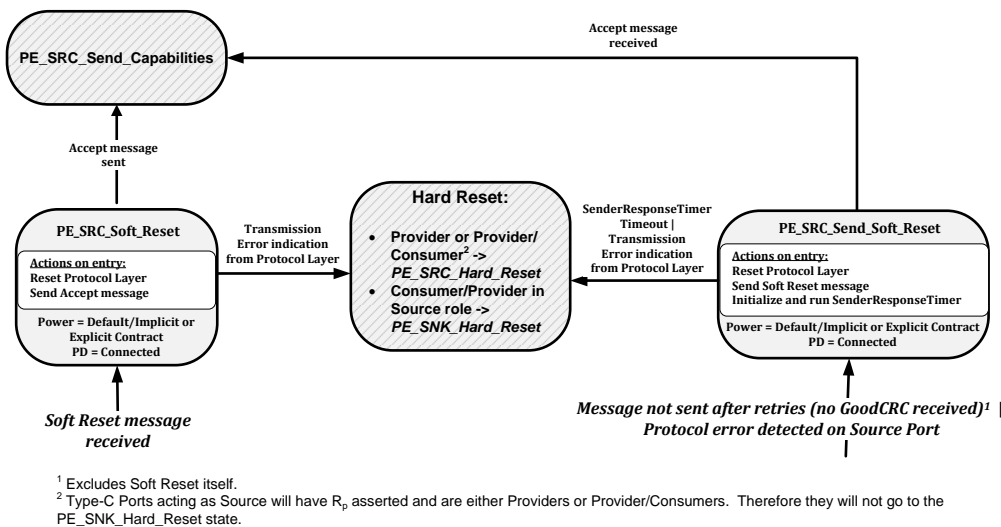
### 8.3.3.4 Soft Reset State Diagrams

#### 8.3.3.4.1 Source Port Soft Reset State Diagram

~~Figure 8-40 below shows the state diagram for the Policy Engine in a Source Port when performing a Soft Reset. The following sections describe operation in each of its Port Partner. The following sections describe operation in each of the states.~~

~~Figure 8-40 the states.~~

Figure- Source Port Soft Reset Diagram



#### 8.3.3.4.1.1 PE\_SRC\_Send\_Soft\_Reset state

The **PE\_SRC\_Send\_Soft\_Reset** state shall be entered from any state when a Protocol Error is detected by the Protocol Layer (see Section 6.7.1) or when a Message has not been sent after retries to the Sink. The main exceptions to this rule are when:

- The source is in the **PE\_SRC\_Send\_Capabilities** state, there is a **Source\_Capabilities** Message sending failure (without GoodCRC) and the source is not presently attached (as indicated in Figure 8-38). In this case, the **PE\_SRC\_Discovery** state is entered (see Section 8.3.3.2.3).
- During a Power Role Swap when the power supply is in transition (see Sections 8.3.3.6.3.1 and 8.3.3.6.3.2). In this case a hard reset will be triggered directly.
- During a Data Role Swap when the DFP/UFP roles are changing. In this case Type-C error recovery will be triggered directly.

~~Note that there are corner cases that are not shown in the defined state diagrams that could be handled without generating a Protocol Error.~~

On entry to the **PE\_SRC\_Send\_Soft\_Reset** state the Policy Engine shall request the Protocol Layer to perform a Soft Reset, then shall send a **Soft\_Reset** Message to the Sink, and initialize and run the **SenderResponseTimer**.

The Policy Engine shall transition to the **PE\_SRC\_Send\_Capabilities** state when:

- An **Accept** Message has been received.

The Policy Engine shall transition to the **PE\_SRC\_Hard\_Reset** or **PE\_SNK\_Hard\_Reset** state depending on its default role as either a Source or Sink Port (see Section 6.7.2) when:

- A **SenderResponseTimer** timeout occurs.
- Or the Protocol Layer indicates that a transmission error has occurred.

The decision as to whether to go to **PE\_SRC\_Hard\_Reset** or **PE\_SNK\_Hard\_Reset** shall depend on the type of device:

- The Source Port in a Provider or Provider/Consumer shall go to **PE\_SRC\_Hard\_Reset**.

- The Source Port in a **Type-B** Consumer/Provider shall go to **PE\_SNK\_Hard\_Reset** i.e. revert to USB Default Operation as Sink Port.

**8.3.3.4.1.2** PE\_SRC\_Soft\_Reset state

The **PE\_SRC\_Soft\_Reset** state shall be entered from any state when a **Soft\_Reset** Message is received from the Protocol Layer.

On entry to the **PE\_SRC\_Soft\_Reset** state the Policy Engine shall reset the Protocol Layer and shall then request the Protocol Layer to send an **Accept** Message.

The Policy Engine shall transition to the **PE\_SRC\_Send\_Capabilities** state (see Section 8.3.3.2.3) when:

- The **Accept** Message has been sent.

The Policy Engine shall transition to the **PE\_SRC\_Hard\_Reset** or **PE\_SNK\_Hard\_Reset** state depending on its default role as either a Source or Sink Port (see Section 6.7.2) when:

- The Protocol Layer indicates that a transmission error has occurred.

The decision as to whether to go to **PE\_SRC\_Hard\_Reset** or **PE\_SNK\_Hard\_Reset** shall depend on the type of device:

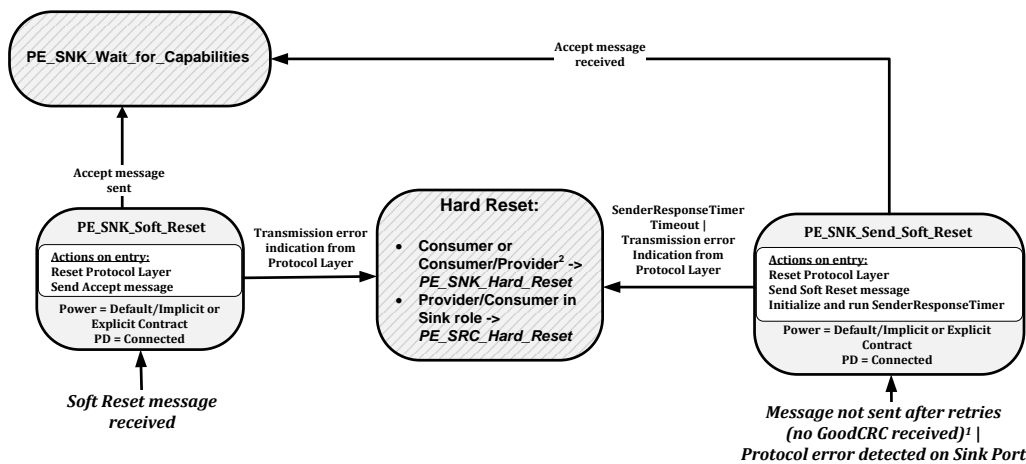
- The Source Port in a Provider or Provider/Consumer shall go to **PE\_SRC\_Hard\_Reset**.
- The Source Port in a Consumer/Provider shall go to **PE\_SNK\_Hard\_Reset** i.e. revert to USB Default Operation as Sink Port.

**8.3.3.4.2** Sink Port Soft Reset State Diagram

**Figure 8-41** below shows the state diagram for the Policy Engine in a Sink Port when performing a Soft Reset. ~~The following sections describe operation in each of the states.~~

~~**Figure 8-** of its Port Partner. The following sections describe operation in each of the states.~~

**Figure 8-41 Sink Port Soft Reset Diagram**



<sup>1</sup> Excludes Soft Reset itself.

<sup>2</sup> Type-C Ports acting as Sinks will have R<sub>s</sub> asserted and are either Consumers or Consumer/Providers. Therefore they will not go to the PE\_SRC\_Hard\_Reset state.

#### 8.3.3.4.2.1 PE\_SNK\_Send\_Soft\_Reset state

The *PE\_SNK\_Send\_Soft\_Reset* state shall be entered from any state when a Protocol Error is detected by the Protocol Layer (see Section 6.7.1) or when a Message has not been sent after retries *to the Source*. The main exceptions to this rule are when:

- During a Power Role Swap when the power supply is in transition (see Sections 8.3.3.6.3.1 and 8.3.3.6.3.2). In this case a hard reset will be triggered directly.
- During a Data Role Swap when the DFP/UFP roles are changing. In this case Type-C error recovery will be triggered directly.

~~Note that there are corner cases that are not shown in the defined state diagrams that could be handled without generating a protocol error.~~

On entry to the *PE\_SNK\_Send\_Soft\_Reset* state the Policy Engine shall request the Protocol Layer to perform a Soft Reset, then shall send a *Soft\_Reset* Message to the Source, and initialize and run the *SenderResponseTimer*.

The Policy Engine shall transition to the *PE\_SNK\_Wait\_for\_Capabilities* state when:

- An *Accept* Message has been received.

The Policy Engine shall transition to the *PE\_SRC\_Hard\_Reset* or *PE\_SNK\_Hard\_Reset* state when:

- A *SenderResponseTimer* timeout occurs.
- Or the Protocol Layer indicates that a transmission error has occurred.

The decision as to whether to go to *PE\_SRC\_Hard\_Reset* or *PE\_SNK\_Hard\_Reset* shall depend on the type of device:

- The *SourceSink* Port in a ~~Provider or Provider~~/Consumer shall go to *PE\_SRC\_Hard\_Reset*.
- The *SourceSink* Port in a ~~Consumer or~~ Consumer/Provider shall go to *PE\_SNK\_Hard\_Reset* i.e. revert to USB Default Operation as Sink Port.

#### 8.3.3.4.2.2 PE\_SNK\_Soft\_Reset state

The *PE\_SNK\_Soft\_Reset* state shall be entered from any state when a *Soft\_Reset* Message is received from the Protocol Layer.

On entry to the *PE\_SNK\_Soft\_Reset* state the Policy Engine shall reset the Protocol Layer and shall then request the Protocol Layer to send an *Accept* Message.

The Policy Engine shall transition to the *PE\_SNK\_Wait\_for\_Capabilities* state when:

- The *Accept* Message has been sent.

The Policy Engine shall transition to the *PE\_SRC\_Hard\_Reset* or *PE\_SNK\_Hard\_Reset* state depending on its default role as either a Source or Sink Port (see Section 6.7.2) when:

- The Protocol Layer indicates that a transmission error has occurred.

The decision as to whether to go to *PE\_SRC\_Hard\_Reset* or *PE\_SNK\_Hard\_Reset* shall depend on the type of device:

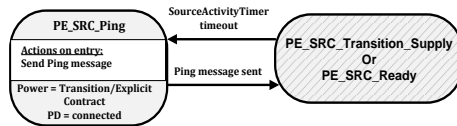
- The *SourceSink* Port in a ~~Type-A Provider or Provider~~/Consumer shall go to *PE\_SRC\_Hard\_Reset*.
- The *SourceSink* Port in a ~~Consumer or~~ Consumer/Provider shall go to *PE\_SNK\_Hard\_Reset* i.e. revert to USB Default Operation as Sink Port.

#### 8.3.3.5 Source Port Ping State Diagram

Figure 8-42 shows the state diagram for a *Ping* Message from a Source Port. Note: Pings are optional under certain operating conditions (see Section 6.3.5).



Figure 8-42 Source Port Ping State Diagram



### 8.3.3.5.1 PE\_SRC\_Ping state

On entry to the *PE\_SRC\_Ping* state (from the *PE\_SRC\_Transition\_Supply* or *PE\_SRC\_Ready* states) the Policy Engine shall request the Protocol Layer to send a *Ping* Message.

The Policy Engine shall transition back to the previous state (*PE\_SRC\_Transition\_Supply* or *PE\_SRC\_Ready*) state (see Figure 8-38) when:

- The *Ping* Message has been successfully sent.

On re-entry to the *PE\_SRC\_Transition\_Supply* or *PE\_SRC\_Ready* states the Policy Engine shall not perform any of the “Actions on Entry” except for initializing and running the *SourceActivityTimer*.

### 8.3.3.6 Dual-Role Port State Diagrams

Dual-Role Ports that combine Source and Sink capabilities shall comprise Source and Sink Policy Engine state machines. In addition they shall have the capability to perform a Power Role Swap from the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* states and shall return to USB Default Operation on a Hard Reset.

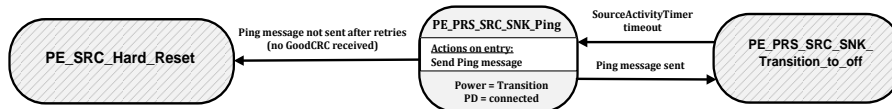
#### 8.3.3.6.1 Type-A/B Dual-Role State Diagrams

The State Diagrams in this section shall apply to all Type-A/B Dual-Role Ports.

##### 8.3.3.6.1.1 Type-A/B Dual-Role (initially Source Port) Ping State Diagram

Figure 8-43 shows the state diagram for a Dual-Role Port which is initially a Source Port. Note: Pings are optional under certain operating conditions (see Section 6.3.5).

Figure 8-43 Dual-Role (initially Source Port) Ping State Diagram



##### 8.3.3.6.1.1.1 PE\_PRS\_SRC\_SNK\_Ping state

The Policy Engine shall transition to the *PE\_PRS\_SRC\_SNK\_Ping* state, from the *PE\_PRS\_SRC\_SNK\_Transition\_to\_off* state, due to a *SourceActivityTimer* timeout.

On entry to the *PE\_PRS\_SRC\_SNK\_Ping* state the Policy Engine shall request the Protocol Layer to send a *Ping* Message.

The Policy Engine shall transition back to the *PE\_PRS\_SRC\_SNK\_Transition\_to\_off* state (see Figure 8-51) when:

- The *Ping* Message has been successfully sent.

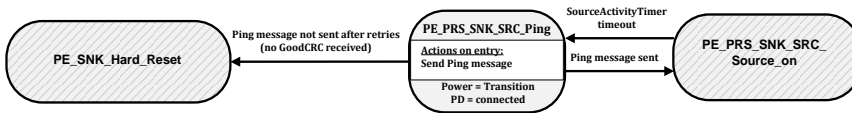
The Policy Engine shall transition to the *PE\_SRC\_Hard\_Reset* when:

- The *Ping* Message has not been sent after retries (a *GoodCRC* Message has not been received). A soft reset shall not be initiated in this case.

### 8.3.3.6.1.2 Type-A/B Dual-Role (initially Sink Port) Ping State Diagram

Figure 8-44 shows the state diagram for a Dual-Role Port which is initially a Sink Port. Note: Pings are optional under certain operating conditions (see Section 6.3.5).

Figure 8-44 Dual-Role (initially Sink Port) Ping State Diagram



#### 8.3.3.6.1.2.1 PE\_PRS\_SNK\_SRC\_Ping state

The Policy Engine shall transition to the *PE\_PRS\_SNK\_SRC\_Ping* state, from the *PE\_PRS\_SNK\_SRC\_Source\_on* state, due to a *SourceActivityTimer* timeout.

On entry to the *PE\_PRS\_SNK\_SRC\_Ping* state from the *PE\_PRS\_SNK\_SRC\_Source\_on* state the Policy Engine shall request the Protocol Layer to send a *Ping* Message.

The Policy Engine shall transition back to the *PE\_PRS\_SNK\_SRC\_Source\_on* state (see Figure 8-52) when:

- The *Ping* Message has been successfully sent.

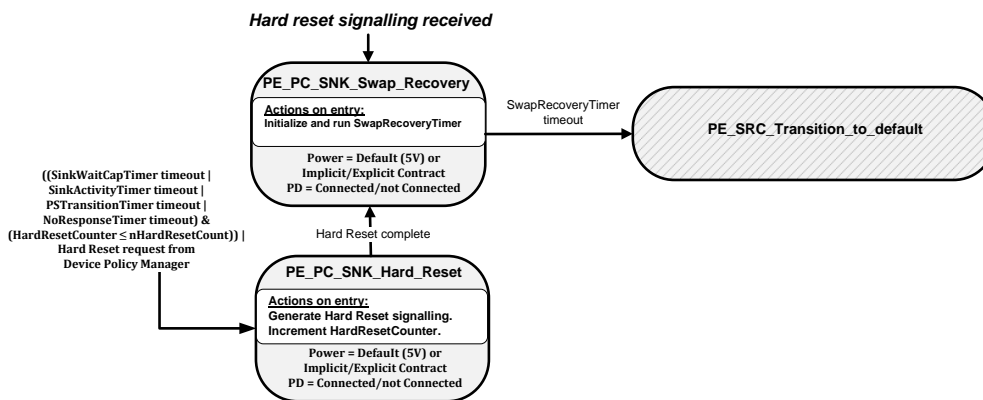
The Policy Engine shall transition to the *PE\_SNK\_Hard\_Reset* state when:

- The *Ping* Message has not been sent after retries (a *GoodCRC* Message has not been received). A soft reset shall not be initiated in this case.

### 8.3.3.6.1.3 Type-A/B Hard Reset of Policy Engine in a Provider/Consumer in Sink Role

Figure 8-45 shows the state diagram in the case where a Provider/Consumer with a Port operating in Sink Role is required to perform a Hard Reset.

Figure 8-45 State Diagram for Hard Reset of P/C in Sink Role



### 8.3.3.6.1.3.1 PE\_PC\_SNK\_Hard\_Reset state

The Policy Engine shall transition to the **PE\_PC\_SNK\_Hard\_Reset** state for a Provider/Consumer Port in Sink Role from any state when:

- ((*SinkWaitCapTimer* timeout | *SinkActivityTimer* timeout | *PSTransitionTimer* timeout | *NoResponseTimer* timeout) & (*HardResetCounter* ≤ *nHardResetCount*)) |
- Hard Reset request from Device Policy Manager

The Policy Engine shall transition to the **PE\_PC\_SNK\_Swap\_Recovery** state when:

- The Hard Reset is complete.

### 8.3.3.6.1.3.2 PE\_PC\_SNK\_Swap\_Recovery state

The Policy Engine shall transition to the **PE\_PC\_SNK\_Swap\_Recovery** state from any state when:

- **Hard Reset** Signaling is received.

On entry to the **PE\_PC\_SNK\_Swap\_Recovery** state the Policy Engine shall initialize and run the *SwapRecoveryTimer*.

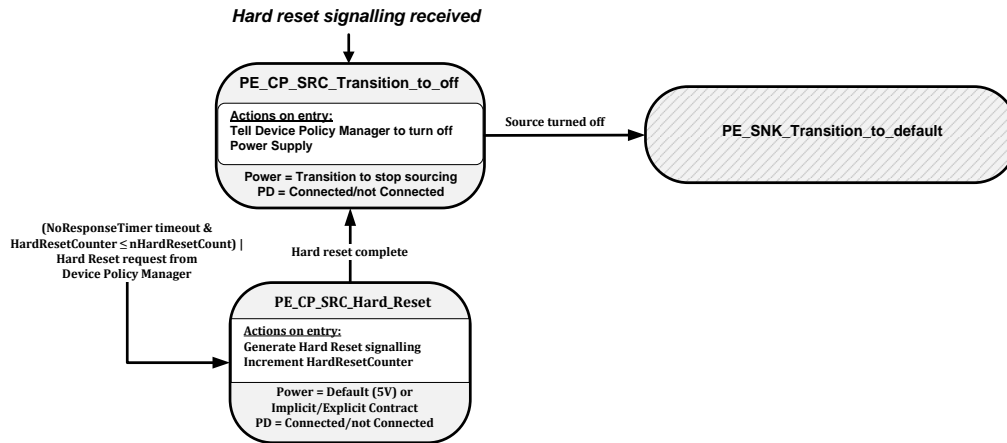
The Policy Engine shall transition to the **PE\_SRC\_Transition\_to\_default** state for a Source Port when:

- The *SwapRecoveryTimer* times out.

### 8.3.3.6.1.4 Type-A/B Hard Reset of Policy Engine in a Consumer/Provider in Source Role

Figure 8-46 shows the state diagram in the case where a Consumer/Provider with a Port operating in Source Role is required to perform a Hard Reset.

Figure 8-46 State Diagram for the Hard Reset of a C/P in Source Role



### 8.3.3.6.1.4.1 PE\_CP\_SRC\_Hard\_Reset state

The Protocol Engine shall transition from any state to the **PE\_CP\_SRC\_Hard\_Reset** state for a Consumer/Provider in Source Role when

- The *NoResponseTimer* times out and the *HardResetCounter* ≤ *nHardResetCount* or

- The Device Policy Manager requests a Hard Reset.

The Policy Engine shall transition to the *PE\_CP\_SRC\_Transition\_to\_off* state when:

- The Hard Reset is complete.

#### 8.3.3.6.1.4.2 PE\_CP\_SRC\_Transition\_to\_off state

The Policy Engine shall transition from any state to the *PE\_CP\_SRC\_Transition\_to\_off* state for a Consumer/Provider in Source Role when:

- *Hard Reset* Signaling is detected.

On entry to the *PE\_CP\_SRC\_Transition\_to\_off* state the Policy Engine shall tell the Device Policy Manager to turn off the power supply.

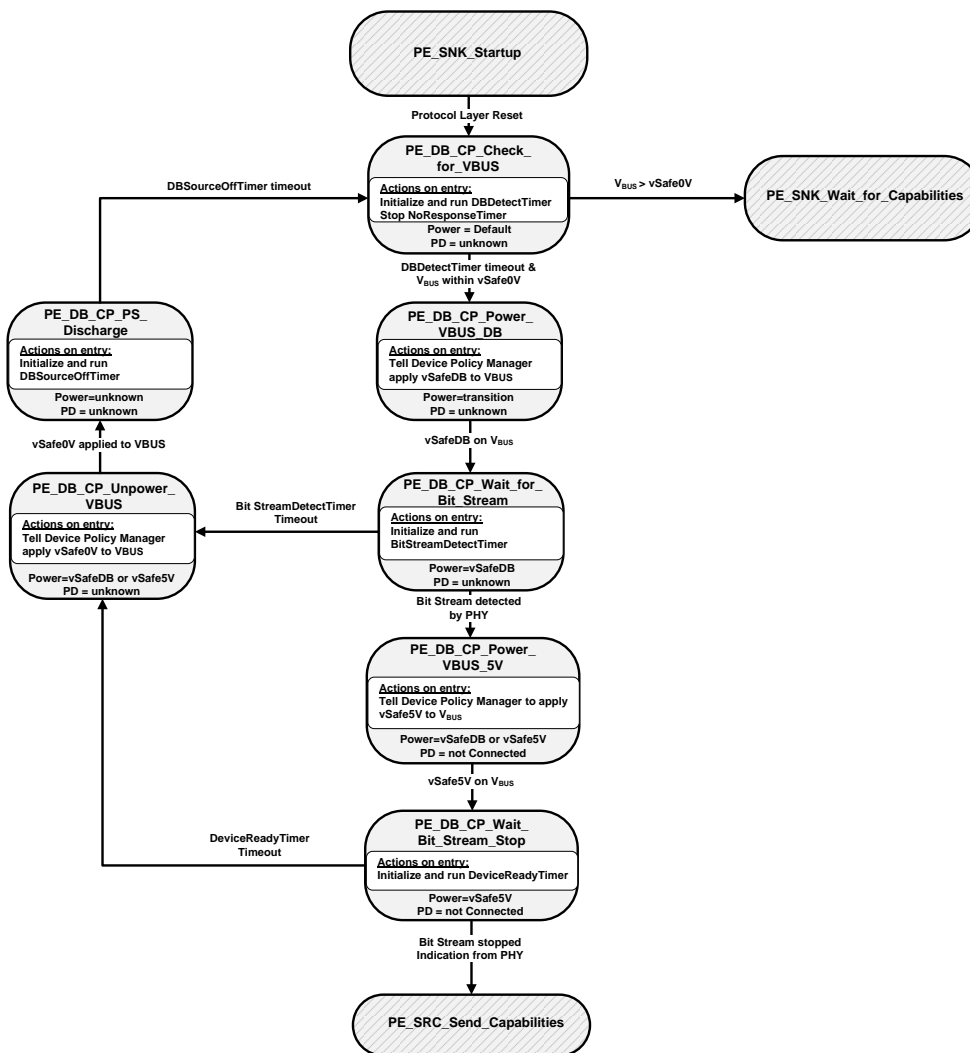
The Policy Engine shall transition to the *PE\_SNK\_Transition\_to\_default* when:

- The power supply has been turned off.

### 8.3.3.6.1.5 Type-A/B Consumer/Provider Dead Battery/Power Loss State Diagram

Figure 8-47 shows the additional state diagram required for a Consumer/Provider to handle Dead Battery detection. After the Consumer/Provider Policy Engine has transitioned to the *PE\_SRC\_Send\_Capabilities* state, its subsequent state operation shall conform to that of a Consumer/Provider which has completed a Power Role Swap (see Section 8.3.3.6.3.2). The Consumer/Provider has effectively undergone a Power Role Swap without the requirement of protocol negotiation. The Consumer/Provider will not respond to received *Source\_Capabilities* Messages until it transitions to the *PE\_SNK\_Wait\_for\_Capabilities* state.

Figure 8-47 Consumer/Provider Dead Battery/Power Loss State Diagram



#### 8.3.3.6.1.5.1 PE\_DB\_CP\_Check\_for\_VBUS state

The Policy Engine for a Consumer/Provider shall initially start in the *PE\_SNK\_Startup* state. Once the Protocol Layer has been reset it shall transition to the *PE\_DB\_CP\_Check\_for\_VBUS* state.

On entry to the *PE\_DB\_CP\_Check\_for\_VBUS* state the Policy Engine shall initialize and run the *DBDetectTimer* and stop the *NoResponseTimer*.

The Policy Engine shall transition to the *PE\_SNK\_Wait\_for\_Capabilities* state when:

- $V_{BUS}$  is greater than *vSafe0V*.

The Policy Engine shall transition to the *PE\_DB\_CP\_Power\_VBUS\_DB* state when:

- The *DBDetectTimer* has timed out and
- $V_{BUS}$  is within *vSafe0V*.

#### 8.3.3.6.1.5.2 PE\_DB\_CP\_Power\_VBUS\_DB state

On entry to the *PE\_DB\_CP\_Power\_VBUS\_DB* state the Policy Engine shall tell the Device Policy Manager to apply *vSafeDB* to  $V_{BUS}$ .

The Policy Engine shall transition to the *PE\_DB\_CP\_Wait\_For\_Bit\_Stream* state when:

- *vSafeDB* is on  $V_{BUS}$ .

#### 8.3.3.6.1.5.3 PE\_DB\_CP\_Wait\_For\_Bit\_Stream state

On entry to the *PE\_DB\_CP\_Wait\_For\_Bit\_Stream* state the Policy Engine shall initialize and run the *BitStreamDetectTimer*.

The Policy Engine shall transition to the *PE\_DB\_CP\_Power\_VBUS\_5V* state when:

- The PHY Layer indicates that Bit Stream signaling has been received.

The Policy Engine shall transition to the *PE\_DB\_CP\_Unpower\_VBUS* state when:

- The *BitStreamDetectTimer* times out.

#### 8.3.3.6.1.5.4 PE\_DB\_CP\_Power\_VBUS\_5V state

On entry to the *PE\_DB\_CP\_Power\_VBUS\_5V* state the Policy Engine shall request the Device Policy Manager to apply *vSafe5V* to  $V_{BUS}$ .

The Policy Engine shall transition to the *PE\_DB\_CP\_Wait\_Bit\_Stream\_Stop* state when:

- *vSafe5V* is present on  $V_{BUS}$ .

#### 8.3.3.6.1.5.5 PE\_DB\_CP\_Wait\_Bit\_Stream\_Stop state

On entry to the *PE\_DB\_CP\_Wait\_Bit\_Stream\_Stop* state the Policy Engine shall initialize and run the *DeviceReadyTimer*.

The Policy Engine shall transition to the *PE\_SRC\_Send\_Capabilities* state when:

- An indication is received from the PHY Layer that the Bit Stream has stopped.

The Policy Engine shall transition to the *PE\_DB\_CP\_Unpower\_VBUS* state when:

- The *DeviceReadyTimer* times out.

#### 8.3.3.6.1.5.6 PE\_DB\_CP\_Unpower\_VBUS state

On entry to the *PE\_DB\_CP\_Unpower\_VBUS* state the Policy Engine shall tell the Device Policy Manager to apply *vSafe0V* to  $V_{BUS}$ .

The Policy Engine shall transition to the *PE\_DB\_CP\_PS\_Discharge* state when:

- *vSafe0V* has been applied to  $V_{BUS}$ .

Note: the intention of applying *vSafe0V* is that the Consumer/Provider will utilize the same mechanism to unpower  $V_{BUS}$  as it uses to remove  $V_{BUS}$  power during a Power Role Swap.

#### 8.3.3.6.1.5.7 PE\_DB\_CP\_PS\_Discharge state

On entry to the *PE\_DB\_CP\_PS\_Discharge* state the Policy Engine shall initialize and run the *DBSourceOffTimer*.

The Policy Engine shall transition to the *PE\_DB\_CP\_Check\_for\_VBUS* state when:

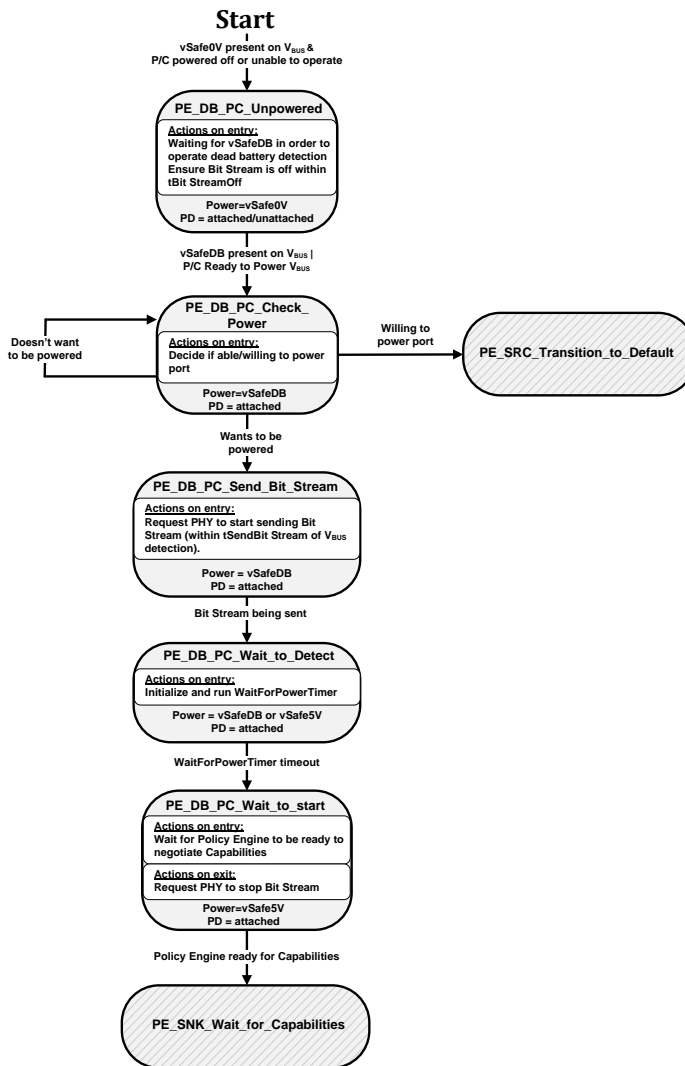
- The *DBSourceOffTimer* times out.

Note: the *DBSourceOffTimer* is used to ensure that the Consumer/Provider is not powering  $V_{BUS}$  when it proceeds to check the voltage on  $V_{BUS}$ . This assumes that the discharge of  $V_{BUS}$  follows the same process as when removing power during a Power Role Swap.

### 8.3.3.6.1.6 Type-A/B Provider/Consumer Dead Battery/Power Loss State Diagram

Figure 8-48 shows the additional state diagram required for a BFSK Provider/Consumer to handle Dead Battery detection. The Provider/Consumer is assumed to startup in a state where it is either powered off or is unable to power its Port (e.g. due to a Dead Battery). If the Provider/Consumer is powered on and has sufficient power to power its Port it should startup as a Source Port.

Figure 8-48 BFSK Provider/Consumer Dead Battery/Power Loss State Diagram





#### 8.3.3.6.1.6.1 PE\_DB\_PC\_Unpowered state

The **PE\_DB\_PC\_Unpowered** state is the startup state for a Provider/Consumer at power up when either there is no power to the Provider/Consumer (in this case there may be no physical “state” as such) or when the Provider/Consumer has some sort of power supply but is inactive.

The **PE\_DB\_PC\_Unpowered** state shall be entered from any state when:

- $V_{BUS}$  is within **vSafe0V** and
- The Provider/Consumer is powered off or has insufficient power to operate.

On entry to the **PE\_DB\_PC\_Unpowered** state the Policy Engine shall wait for **vSafeDB** to appear on  $V_{BUS}$  in order to start the Dead Battery detection process. If a Bit Stream is currently being transmitted then this shall be stopped within **tBitStreamOff** of **vSafe0V** appearing on  $V_{BUS}$ .

The Policy Engine shall transition to the **PE\_DB\_PC\_Check\_Power** state when:

- **vSafeDB** is present on  $V_{BUS}$ , or
- The Provider/Consumer is ready to power  $V_{BUS}$

#### 8.3.3.6.1.6.2 PE\_DB\_PC\_Check\_Power state

On entry to the **PE\_DB\_PC\_Check\_Power** state the Policy Engine shall decide whether it is able and willing to supply power to the Port.

The Policy Engine shall transition to the **PE\_SRC\_Transition\_to\_default** state when:

- It is willing to power  $V_{BUS}$ .

The Policy Engine shall transition to the **PE\_DB\_PC\_Send\_Bit\_Stream** state when:

- It wants to be powered by the Consumer/Provider.

The Policy Engine shall stay in the **PE\_DB\_PC\_Check\_Power** state when:

- The Provider/Consumer does not want to either power the Port or be powered.

#### 8.3.3.6.1.6.3 PE\_DB\_PC\_Send\_Bit\_Stream state

On entry to the **PE\_DB\_PC\_Send\_Bit\_Stream** state the Policy Engine shall request the PHY Layer to start sending the Bit Stream (see Section 4.1.1).

The Policy Engine shall transition to the **PE\_DB\_PC\_Wait\_to\_Detect** state when:

- The Bit Stream is being sent.

#### 8.3.3.6.1.6.4 PE\_DB\_PC\_Wait\_to\_Detect state

On entry to the **PE\_DB\_PC\_Wait\_to\_Detect** state the Policy Engine shall initialize and run the **WaitForPowerTimer** to allow the Consumer/Provider time to detect the Bit Stream and apply power to  $V_{BUS}$ .

The Policy Engine shall transition to the **PE\_DB\_PC\_Wait\_to\_Start** state when:

- The DeadBatteryTimer times out.

#### 8.3.3.6.1.6.5 PE\_DB\_PC\_Wait\_to\_Start state

On entry to the **PE\_DB\_PC\_Wait\_to\_Start** state the Policy Engine shall wait until it is ready to negotiate Capabilities.

On exit from the **PE\_DB\_PC\_Wait\_to\_Start** state the Policy Engine shall request the PHY Layer to stop sending the Bit Stream.

The Policy Engine shall transition to the **PE\_SNK\_Wait\_for\_Capabilities** state when:

- The Policy Engine is ready to negotiate Capabilities.

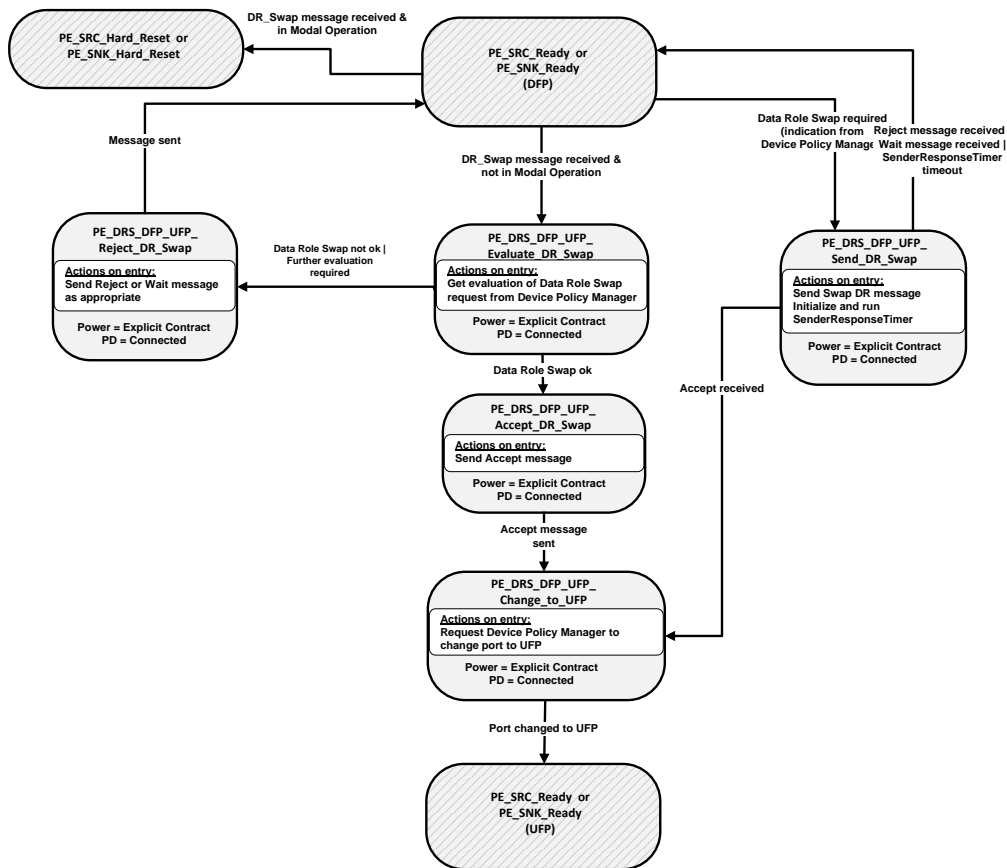
### 8.3.3.6.2 Type-C ~~DRP~~DR\_Swap State Diagrams

The State Diagrams in this section shall apply to all Dual-Role Ports that are [USBTyPe-C.1.0] DRPs.

#### 8.3.3.6.2.1 Type-C ~~DRP~~Policy Engine in DFP to UFP Data Role Swap State Diagram

Figure 8-49 shows the additional state diagram required to perform a Data Role Swap from Type-C DFP to UFP operation and the changes that shall be followed for error and Hard Reset handling. ~~If successful a Provider or Provider/Consumer will become a Consumer or Consumer/Provider.~~

Figure 8-49: ~~Dual-Role Port in C-~~ Type-C DFP to UFP Data Role Swap State Diagram



##### 8.3.3.6.2.1.1 PE\_SRC\_Ready or PE\_SNK\_Ready state

The Data Role Swap process shall start only from either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state where power is stable.

The Policy Engine shall transition to the *PE\_DRS\_DFP\_UFP\_Evaluate\_DR\_Swap* state when:

- A *DR\_Swap* Message is received and
- There are no Active Modes (not in Modal Operation).

The Policy Engine shall transition to either the *PE\_SRC\_Hard\_Reset* or *PE\_SNK\_Hard\_Reset* states when:

- A *DR\_Swap* Message is received and
- There are one or more Active Modes (Modal Operation).

The Policy Engine shall transition to the *PE\_DRS\_DFP\_UFP\_Send\_DR\_Swap* state when:

- The Device Policy Manager indicates that a Data Role Swap is required.

**8.3.3.6.2.1.2** PE\_DRS\_DFP\_UFP\_Evaluate\_DR\_Swap state

On entry to the *PE\_DRS\_DFP\_UFP\_Evaluate\_DR\_Swap* state the Policy Engine shall ask the Device Policy Manager whether a Data Role Swap can be made.

The Policy Engine shall transition to the *PE\_DRS\_DFP\_UFP\_Accept\_DR\_Swap* state when:

- The Device Policy Manager indicates that a Data Role Swap is ok.

The Policy Engine shall transition to the *PE\_DRS\_DFP\_UFP\_Reject\_DR\_Swap* state when:

- The Device Policy Manager indicates that a Data Role Swap is not ok.
- Or further evaluation of the Data Role Swap request is needed.

**8.3.3.6.2.1.3** PE\_DRS\_DFP\_UFP\_Accept\_DR\_Swap state

On entry to the *PE\_DRS\_DFP\_UFP\_Accept\_DR\_Swap* state the Policy Engine shall request the Protocol Layer to send an *Accept* Message.

The Policy Engine shall transition to the *PE\_DRS\_DFP\_UFP\_Change\_to\_UFP* state when:

- The *Accept* Message has been sent.

**8.3.3.6.2.1.4** PE\_DRS\_DFP\_UFP\_Change\_to\_UFP state

On entry to the *PE\_DRS\_DFP\_UFP\_Change\_to\_UFP* state the Policy Engine shall request the Device Policy Manager to change the Port from a DFP to a UFP.

The Policy Engine shall transition to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state when:

- The Device Policy Manager indicates that the Type-C Port has been changed to a UFP.

**8.3.3.6.2.1.5** PE\_DRS\_DFP\_UFP\_Send\_DR\_Swap state

On entry to the *PE\_DRS\_DFP\_UFP\_Send\_DR\_Swap* state the Policy Engine shall request the Protocol Layer to send a *DR\_Swap* Message and shall start the *SenderResponseTimer*.

On exit from the *PE\_DRS\_DFP\_UFP\_Send\_DR\_Swap* state the Policy Engine shall stop the *SenderResponseTimer*.

The Policy Engine shall continue as a DFP and shall transition to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state when:

- A *Reject* Message is received.
- Or a *Wait* Message is received.
- Or the *SenderResponseTimer* times out.

The Policy Engine shall transition to the *PE\_DRS\_DFP\_UFP\_Change\_to\_UFP* state when:

- An *Accept* Message is received.

#### 8.3.3.6.2.1.6 PE\_DRS\_DFP\_UFP\_Reject\_DR\_Swap state

On entry to the *PE\_DRS\_DFP\_UFP\_Reject\_DR\_Swap* state the Policy Engine shall request the Protocol Layer to send:

- A *Reject* Message if the device is unable to perform a Data Role Swap at this time.
- A *Wait* Message if further evaluation of the Data Role Swap request is required. Note: in this case it is expected that one of the Port Partners will send a *DR\_Swap* Message at a later time (see Section 6.3.12.3).

The Policy Engine shall continue as a DFP and shall transition to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state when:

- The *Reject* or *Wait* Message has been sent.

#### ~~8.3.3.6.2.1.7 ErrorRecovery state~~

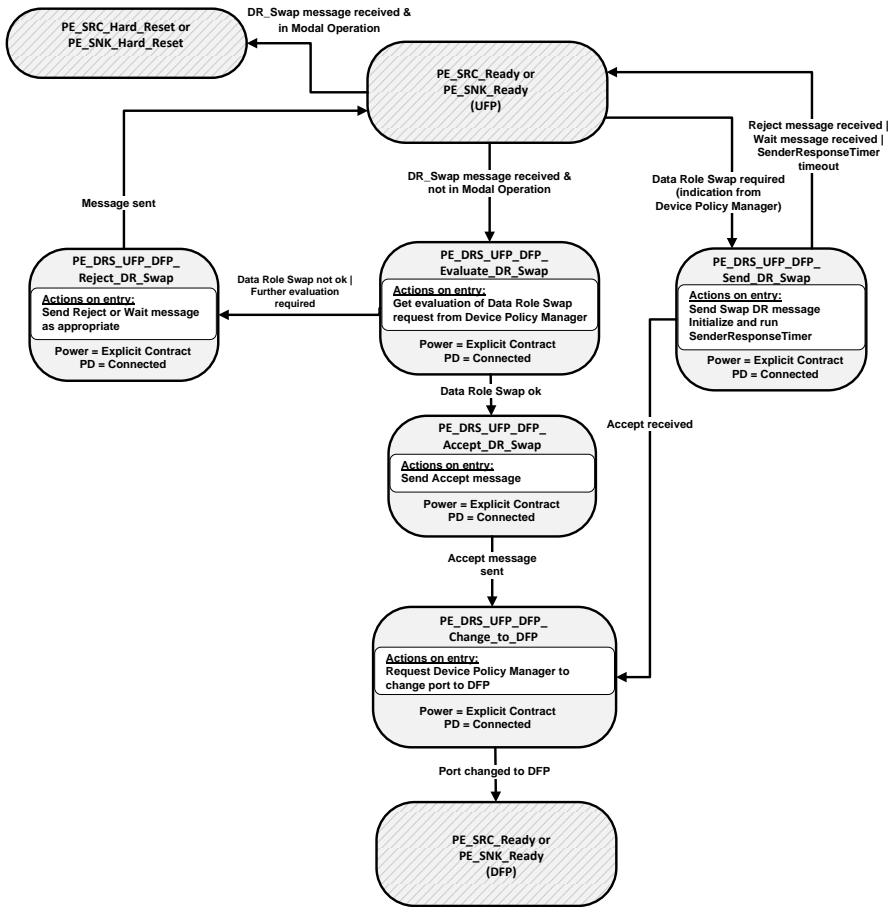
~~The Policy Engine shall transition to the state (see Section ) when:~~

- ~~• A protocol error is detected by the Protocol Layer or~~
- ~~• A message has not been sent after retries or~~
- ~~• signaling is received.~~

### 8.3.3.6.2.2 Type-C DRP Policy Engine in UFP to DFP Data Role Swap State Diagram

Figure 8-50 shows the additional state diagram required to perform a Data Role Swap from Type-C DRP UFP to DFP operation and the changes that shall be followed for error and Hard Reset handling. ~~If successful a Consumer or Consumer/Provider will become a Provider or Provider/Consumer.~~

Figure 8-50: ~~Dual Role Port in C:~~ Type-C UFP to DFP Data Role Swap State Diagram



#### 8.3.3.6.2.2.1 PE\_SRC\_Ready or PE\_SNK\_Ready state

The Data Role Swap process shall start only from the either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state where power is stable.

The Policy Engine shall transition to the *PE\_DRS\_UFP\_DFP\_Evaluate\_DR\_Swap* state when:

- A *DR\_Swap* Message is received: and
- There are no Active Modes (not in Modal Operation).

The Policy Engine shall transition to either the *PE\_SRC\_Hard\_Reset* or *PE\_SNK\_Hard\_Reset* states when:

- A *DR\_Swap* Message is received and
- There are one or more Active Modes (Modal Operation).

The Policy Engine shall transition to the *PE\_DRS\_UFP\_DFP\_Send\_DR\_Swap* state when:

- The Device Policy Manager indicates that a Data Role Swap is required.

**8.3.3.6.2.2.2** PE\_DRS\_UFP\_DFP\_Evaluate\_DR\_Swap state

On entry to the *PE\_DRS\_UFP\_DFP\_Evaluate\_DR\_Swap* state the Policy Engine shall ask the Device Policy Manager whether a Data Role Swap can be made.

The Policy Engine shall transition to the *PE\_DRS\_UFP\_DFP\_Accept\_DR\_Swap* state when:

- The Device Policy Manager indicates that a Data Role Swap is ok.

The Policy Engine shall transition to the *PE\_DRS\_UFP\_DFP\_Reject\_DR\_Swap* state when:

- The Device Policy Manager indicates that a Data Role Swap is not ok.
- Or further evaluation of the Data Role Swap request is needed.

**8.3.3.6.2.2.3** PE\_DRS\_UFP\_DFP\_Accept\_DR\_Swap state

On entry to the *PE\_DRS\_UFP\_DFP\_Accept\_DR\_Swap* state the Policy Engine shall request the Protocol Layer to send an *Accept* Message.

The Policy Engine shall transition to the *PE\_DRS\_UFP\_DFP\_Change\_to\_DFP* state when:

- The *Accept* Message has been sent.

**8.3.3.6.2.2.4** PE\_DRS\_UFP\_DFP\_Change\_to\_DFP state

On entry to the *PE\_DRS\_UFP\_DFP\_Change\_to\_DFP* state the Policy Engine shall request the Device Policy Manager to change the Port from a UFP to a DFP.

The Policy Engine shall transition to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state when:

- The Device Policy Manager indicates that the Type-C Port has been changed to a DFP.

**8.3.3.6.2.2.5** PE\_DRS\_UFP\_DFP\_Send\_DR\_Swap state

On entry to the *PE\_DRS\_UFP\_DFP\_Send\_DR\_Swap* state the Policy Engine shall request the Protocol Layer to send a *DR\_Swap* Message and shall start the *SenderResponseTimer*.

On exit from the *PE\_DRS\_UFP\_DFP\_Send\_DR\_Swap* state the Policy Engine shall stop the *SenderResponseTimer*.

The Policy Engine shall continue as a UFP and shall transition to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state when:

- A *Reject* Message is received.
- Or a *Wait* Message is received.
- Or the *SenderResponseTimer* times out.

The Policy Engine shall transition to the *PE\_DRS\_UFP\_DFP\_Change\_to\_DFP* state when:

- An *Accept* Message is received.

**8.3.3.6.2.2.6** PE\_DRS\_UFP\_DFP\_Reject\_DR\_Swap state

On entry to the *PE\_DRS\_UFP\_DFP\_Reject\_DR\_Swap* state the Policy Engine shall request the Protocol Layer to send:

- A *Reject* Message if the device is unable to perform a Data Role Swap at this time.
- A *Wait* Message if further evaluation of the Data Role Swap request is required. Note: in this case it is expected that one of the Port Partners will send a *DR\_Swap* Message at a later time (see Section 6.3.12.3).

The Policy Engine shall continue as a UFP and shall transition to the either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state when:

- The *Reject* or *Wait* Message has been sent.

~~8.3.3.6.2.2.7 ErrorRecovery state~~

~~The Policy Engine shall transition to the state (see Section ) when:~~

- ~~• A protocol error is detected by the Protocol Layer or~~
- ~~• A message has not been sent after retries or~~
- ~~• signaling is received.~~

#### 8.3.3.6.3 Common Dual-Role Port State Diagrams

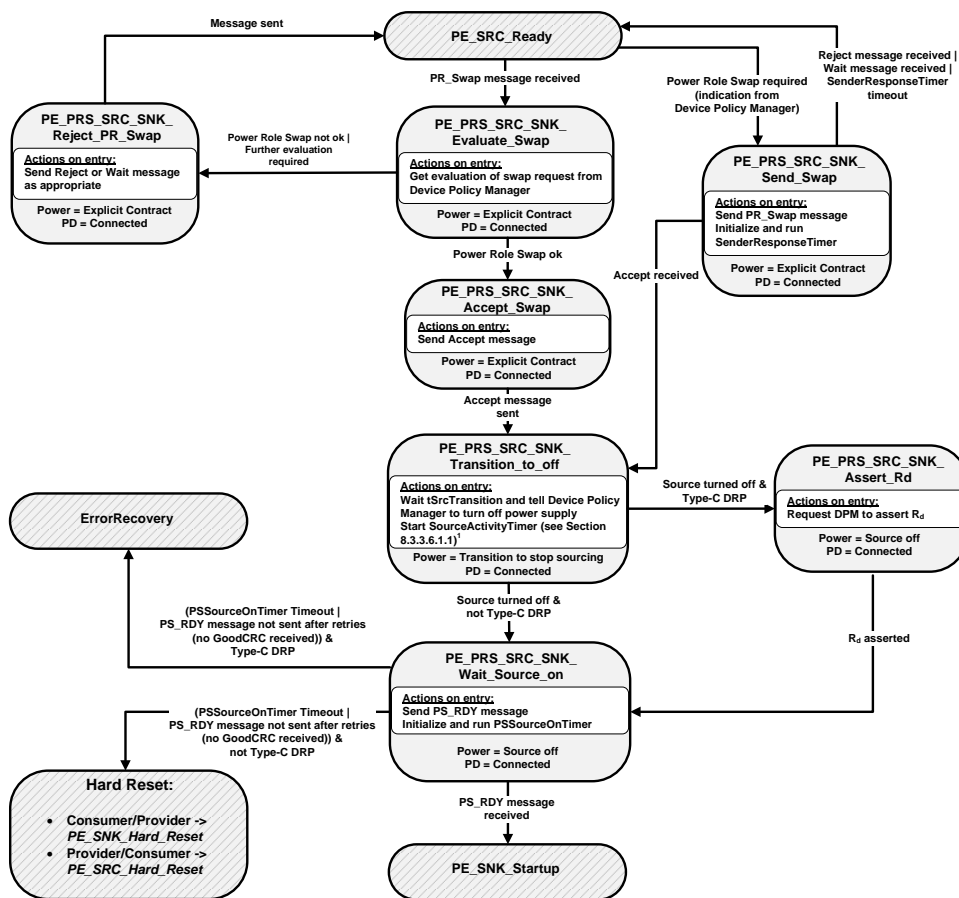
The State Diagrams in this section shall apply to all Dual-Role Ports: both Type-A/B and [*USBType-C1.0*] DRP.

### 8.3.3.6.3.1 Policy Engine in Source to Sink Power Role Swap State Diagram

Dual-Role Ports that combine Source and Sink capabilities shall comprise Source and Sink Policy Engine state machines. In addition they shall have the capability to do a Power Role Swap from the *PE\_SRC\_Ready* state and shall return to USB Default Operation on a Hard Reset.

Figure 8-51 shows the additional state diagram required to perform a Power Role Swap from Source to Sink roles and the changes that shall be followed for error and Hard Reset handling.

Figure 8-51: Dual-Role Port in Source to Sink Power Role Swap State Diagram



<sup>1</sup> When operating at vSafe5V and not swapped, or when two systems both using the Type-C connector are communicating, Ping messages are optional so the SourceActivityTimer is not required to run in these circumstances.

#### 8.3.3.6.3.1.1 PE\_SRC\_Ready state

The Power Role Swap process shall start only from the *PE\_SRC\_Ready* state where power is stable.



The Policy Engine shall transition to the *PE\_PRS\_SRC\_SNK\_Evaluate\_PR\_Swap* state when:

- A *PR\_Swap* Message is received.

The Policy Engine shall transition to the *PE\_PRS\_SRC\_SNK\_Send\_PR\_Swap* state when:

- The Device Policy Manager indicates that a Power Role Swap is required.

#### 8.3.3.6.3.1.2 PE\_PRS\_SRC\_SNK\_Evaluate\_Swap state

On entry to the *PE\_PRS\_SRC\_SNK\_Evaluate\_PR\_Swap* state the Policy Engine shall ask the Device Policy Manager whether a Power Role Swap can be made.

The Policy Engine shall transition to the *PE\_PRS\_SRC\_SNK\_Accept\_PR\_Swap* state when:

- The Device Policy Manager indicates that a Power Role Swap is ok.

The Policy Engine shall transition to the *PE\_PRS\_SRC\_SNK\_Reject\_PR\_Swap* state when:

- The Device Policy Manager indicates that a Power Role Swap is not ok.
- Or further evaluation of the Power Role Swap request is needed.

#### 8.3.3.6.3.1.3 PE\_PRS\_SRC\_SNK\_Accept\_Swap state

On entry to the *PE\_PRS\_SRC\_SNK\_Accept\_PR\_Swap* state the Policy Engine shall request the Protocol Layer to send an *Accept* Message.

The Policy Engine shall transition to the *PE\_PRS\_SRC\_SNK\_Transition\_to\_off* state when:

- The *Accept* Message has been sent.

#### 8.3.3.6.3.1.4 PE\_PRS\_SRC\_SNK\_Transition\_to\_off state

On entry to the *PE\_PRS\_SRC\_SNK\_Transition\_to\_off* state the Policy Engine shall wait *tSrcTransition* and then request the Device Policy Manager to turn off the Source and shall initialize and run the *SourceActivityTimer* (see Section 8.3.3.6.1.1.1 for use of *Ping* messaging for Dual-Role Ports which are initially Source Ports).

The Policy Engine shall transition to the *PE\_PRS\_SRC\_SNK\_Wait\_Source\_on* state when:

- The Device Policy Manager indicates that the Source has been turned off and
- The Port is not a [*USBType-C1.0*] DRP

The Policy Engine shall transition to the *PE\_PRS\_SRC\_SNK\_Assert\_Rd* state when:

- The Device Policy Manager indicates that the Source has been turned off and
- The Port is a [*USBType-C1.0*] DRP

#### 8.3.3.6.3.1.5 PE\_PRS\_SRC\_SNK\_Assert\_Rd

On entry to the *PE\_PRS\_SRC\_SNK\_Assert\_Rd* state the Policy Engine shall request the Device Policy Manager to change the resistor asserted on the CC wire from  $R_p$  to  $R_d$ .

The Policy Engine shall transition to the *PE\_PRS\_SRC\_SNK\_Wait\_Source\_on* state when:

- The Device Policy Manager indicates that  $R_d$  is asserted.

#### 8.3.3.6.3.1.6 PE\_PRS\_SRC\_SNK\_Wait\_Source\_offon state

On entry to the *PE\_PRS\_SRC\_SNK\_Wait\_Source\_on* state the Policy Engine shall request the Protocol Layer to send a *PS\_RDY* Message and shall start the *PSSourceOnTimer*.

On exit from the Source off state the Policy Engine shall stop the *PSSourceOnTimer*.

The Policy Engine shall transition to the *PE\_SNK\_Startup* when:

- A *PS\_RDY* Message is received indicating that the remote Source is now supplying power.

The Policy Engine shall transition to the *PE\_SRC\_Hard\_Reset* or *PE\_SNK\_Hard\_Reset* state depending on its default role as either a Source or Sink Port (see Section 6.7.2) when:

- The Port is not a [USBType-C 1.0] DRP and
- The *PSSourceOnTimer* times out or
- The *PS\_RDY* Message is not sent after retries (a *GoodCRC* Message has not been received). **Note:** a soft reset shall not be initiated in this case.

The decision as to whether to go to *PE\_SRC\_Hard\_Reset* or *PE\_SNK\_Hard\_Reset* shall depend on the type of device:

- The Source Port in a Provider or Provider/Consumer shall go to *PE\_SRC\_Hard\_Reset*.
- The Source Port in a Consumer/Provider shall go to *PE\_SNK\_Hard\_Reset* i.e. revert to USB Default Operation as Sink Port.

The Policy Engine shall transition to the *ErrorRecovery* state when:

- The Port is a [USBType-C 1.0] DRP and
- The *PSSourceOnTimer* times out or
- The *PS\_RDY* Message is not sent after retries (a *GoodCRC* Message has not been received). **Note:** a soft reset shall not be initiated in this case.

~~The decision as to whether to go to *PE\_SRC\_Hard\_Reset* or *PE\_SNK\_Hard\_Reset* shall depend on the type of device:~~

- ~~• The Source Port in a Provider or Provider/Consumer shall go to *PE\_SRC\_Hard\_Reset*.~~
- ~~• The Source Port in a Consumer/Provider shall go to *PE\_SNK\_Hard\_Reset* i.e. revert to USB Default Operation as Sink Port.~~

#### 8.3.3.6.3.1.7 PE\_PRS\_SRC\_SNK\_Send\_Swap state

On entry to the *PE\_PRS\_SRC\_SNK\_Send\_PR\_Swap* state the Policy Engine shall request the Protocol Layer to send a *PR\_Swap* Message and shall start the *SenderResponseTimer*.

On exit from the *PE\_PRS\_SRC\_SNK\_Send\_PR\_Swap* state the Policy Engine shall stop the *SenderResponseTimer*.

The Policy Engine shall transition to the *PE\_SRC\_Ready* state when:

- A *Reject* Message is received.
- Or a *Wait* Message is received.
- Or the *SenderResponseTimer* times out.

The Policy Engine shall transition to the *PE\_PRS\_SRC\_SNK\_Transition\_to\_off* state when:

- An *Accept* Message is received.

#### 8.3.3.6.3.1.8 PE\_PRS\_SRC\_SNK\_Reject\_Swap state

On entry to the *PE\_PRS\_SRC\_SNK\_Reject\_PR\_Swap* state the Policy Engine shall request the Protocol Layer to send:

- A *Reject* Message if the device is unable to perform a Power Role Swap at this time.
- A *Wait* Message if further evaluation of the Power Role Swap request is required. **Note:** in this case it is expected that one of the Port Partners will send a *PR\_Swap* Message at a later time (see Section 6.3.12.2).

The Policy Engine shall transition to the *PE\_SRC\_Ready* when:

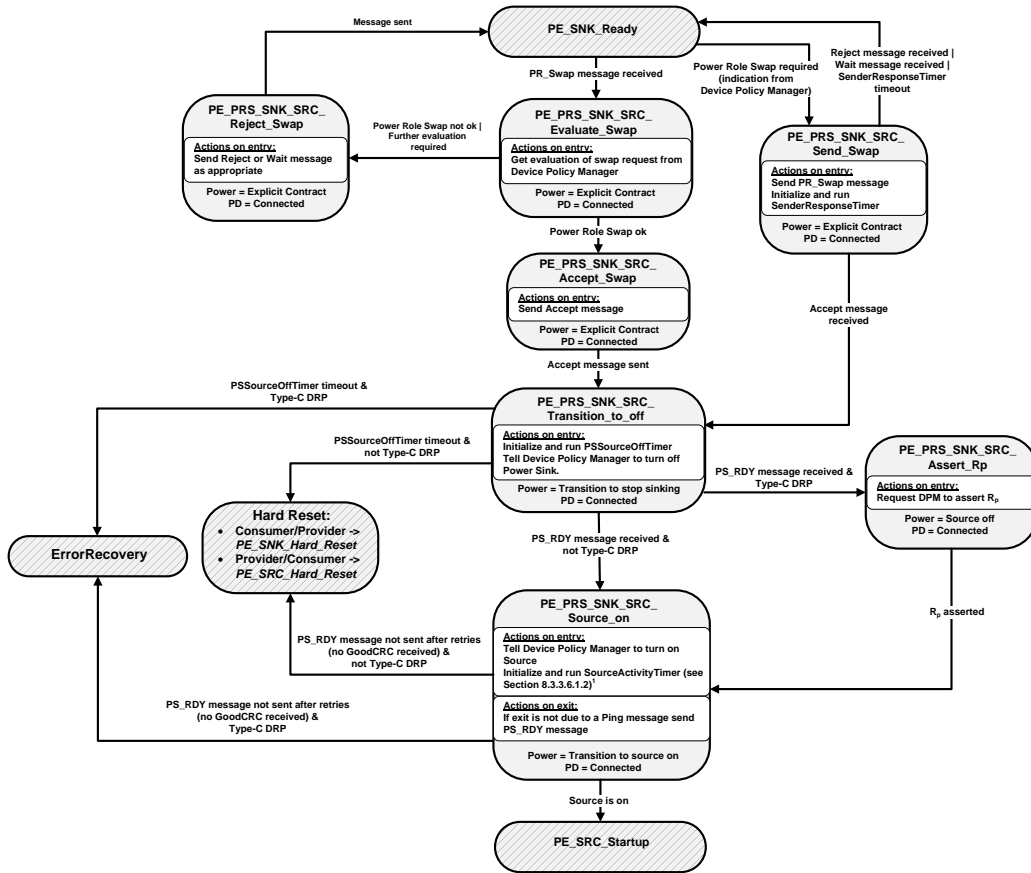
- The *Reject* or *Wait* Message has been sent.

#### 8.3.3.6.3.2 Policy Engine in Sink to Source Power Role Swap State Diagram

Dual-Role Ports that combine Sink and Source capabilities shall comprise Sink and Source Policy Engine state machines. In addition they shall have the capability to do a Power Role Swap from the *PE\_SNK\_Ready* state and shall return to USB Default Operation on a Hard Reset.

Figure 8-52 shows the additional state diagram required to perform a Power Role Swap from Sink to Source roles and the changes that shall be followed for error and Hard Reset handling.

Figure 8-52: Dual-role Port in Sink to Source Power Role Swap State Diagram



<sup>1</sup> When operating at vSafe5V and not swapped, or when two systems both using the Type-C connector are communicating, Ping messages are optional so the SourceActivityTimer is not required to run in these circumstances.

### 8.3.3.6.3.2.1 PE\_SNK\_Ready state

The Power Role Swap process shall start only from the **PE\_SNK\_Ready** state where power is stable.

The Policy Engine shall transition to the **PE\_PRS\_SNK\_SRC\_Evaluate\_PR\_Swap** state when:

- A **PR\_Swap** Message is received.

The Policy Engine shall transition to the **PE\_PRS\_SNK\_SRC\_Send\_PR\_Swap** state when:

- The Device Policy Manager indicates that a Power Role Swap is required.

#### 8.3.3.6.3.2.2 PE\_PRS\_SNK\_SRC\_Evaluate\_Swap state

On entry to the *PE\_PRS\_SNK\_SRC\_Send\_PR\_Swap* state the Policy Engine shall ask the Device Policy Manager whether a Power Role Swap can be made.

The Policy Engine shall transition to the *PE\_PRS\_SNK\_SRC\_Accept\_PR\_Swap* state when:

- The Device Policy Manager indicates that a Power Role Swap is ok.

The Policy Engine shall transition to the *PE\_PRS\_SNK\_SRC\_Reject\_PR\_Swap* state when:

- The Device Policy Manager indicates that a Power Role Swap is not ok.

#### 8.3.3.6.3.2.3 PE\_PRS\_SNK\_SRC\_Accept\_Swap state

On entry to the *PE\_PRS\_SNK\_SRC\_Accept\_PR\_Swap* state the Policy Engine shall request the Protocol Layer to send an *Accept* Message.

The Policy Engine shall transition to the *PE\_PRS\_SNK\_SRC\_Transition\_to\_off* state when:

- The *Accept* Message has been sent.

#### 8.3.3.6.3.2.4 PE\_PRS\_SNK\_SRC\_Transition\_to\_off state

On entry to the *PE\_PRS\_SNK\_SRC\_Transition\_to\_off* state the Policy Engine shall initialize and run the *PSSourceOffTimer* and then request the Device Policy Manager to turn off the Sink.

The Policy Engine shall transition to the *PE\_SRC\_Hard\_Reset* or *PE\_SNK\_Hard\_Reset* state depending on its default role as either a Source or Sink Port (see Section 6.7.2) when:

- The *PSSourceOffTimer* times out. ~~and~~

The ~~decision as to whether to go to or shall depend on the type of device:~~

- ~~The Source Port in is not a [USBType-C 1.0] Provider or Provider/Consumer shall go to - DRP.~~

The decision as to whether to go to *PE\_SRC\_Hard\_Reset* or *PE\_SNK\_Hard\_Reset* shall depend on the type of device:

- The Source Port in a Provider or Provider/Consumer shall go to *PE\_SRC\_Hard\_Reset*.
- The Source Port in a Consumer/Provider shall go to *PE\_SNK\_Hard\_Reset* i.e. revert to USB Default Operation as Sink Port.

The ~~Source Port in a Consumer/Provider~~ Policy Engine shall ~~go~~ transition to the *ErrorRecovery* state when:

- ~~The *PSSourceOffTimer* times out and~~
- ~~The Port is a [USBType-C 1.0] i.e. revert to USB Default Operation as Sink Port. DRP.~~

The Policy Engine shall transition to the *PE\_PRS\_SNK\_SRC\_Source\_on* state when:

- A *PS\_RDY* Message is received and
- This is not a [USBType-C 1.0] DRP

The Policy Engine shall transition to the *PE\_PRS\_SNK\_SRC\_Assert\_Rp* state when:

- A *PS\_RDY* Message is received and
- ~~This~~The Port is a [USBType-C 1.0] DRP

#### 8.3.3.6.3.2.5 PE\_PRS\_SNK\_SRC\_Assert\_Rp state

On entry to the *PE\_PRS\_SNK\_SRC\_Assert\_Rp* state the Policy Engine shall request the Device Policy Manager to change the resistor asserted on the CC wire from R<sub>d</sub> to R<sub>p</sub>.

The Policy Engine shall transition to the *PE\_PRS\_SNK\_SRC\_Source\_on* state when:

- The Device Policy Manager indicates that R<sub>d</sub> is asserted.

#### 8.3.3.6.3.2.6 PE\_PRS\_SNK\_SRC\_Source\_on state

On entry to the **PE\_PRS\_SNK\_SRC\_Source\_on** state the Policy Engine shall request the Device Policy Manager to turn on the Source and shall initialize and run the **SourceActivityTimer** (see Section 8.3.3.6.1.2.1 for details of **Ping** messaging for Dual-Role ports which are initially Sink Ports).

On exit from the **PE\_PRS\_SNK\_SRC\_Source\_on** state (except if the exit is due to a **SourceActivityTimer** timeout) the Policy Engine shall send a **PS\_RDY** Message.

The Policy Engine shall transition to the **PE\_SRC\_Startup** state when:

- The Source Port has been turned on.

The Policy Engine shall transition to the **PE\_SRC\_Hard\_Reset** or **PE\_SNK\_Hard\_Reset** state depending on its default role as either a Source or Sink Port (see Section 6.7.2) when:

- The Port is not a [USBType-C 1.0] DRP and
- The **PS\_RDY** Message is not sent after retries (a **GoodCRC** Message has not been received). A soft reset shall not be initiated in this case.

The decision as to whether to go to **PE\_SRC\_Hard\_Reset** or **PE\_SNK\_Hard\_Reset** shall depend on the type of device:

- The Source Port in a Provider or Provider/Consumer shall go to **PE\_SRC\_Hard\_Reset**.
- The Source Port in a Consumer/Provider shall go to **PE\_SNK\_Hard\_Reset** i.e. revert to USB Default Operation as Sink Port.

The Policy Engine shall transition to the **ErrorRecovery** state when:

- The Port is a [USBType-C 1.0] DRP and
- The **PS\_RDY** Message is not sent after retries (a **GoodCRC** Message has not been received). A soft reset shall not be initiated in this case.

#### 8.3.3.6.3.2.7 PE\_PRS\_SNK\_SRC\_Send\_Swap state

On entry to the **PE\_PRS\_SNK\_SRC\_Send\_PR\_Swap** state the Policy Engine shall request the Protocol Layer to send a **PR\_Swap** Message and shall initialize and run the **SenderResponseTimer**.

The Policy Engine shall transition to the **PE\_SNK\_Ready** state when:

- A **Reject** Message is received.
- Or a **Wait** Message is received.
- Or the **SenderResponseTimer** times out.

The Policy Engine shall transition to the **PE\_PRS\_SNK\_SRC\_Transition\_to\_off** state when:

- An **Accept** Message is received.

#### 8.3.3.6.3.2.8 PE\_PRS\_SNK\_SRC\_Reject\_Swap state

On entry to the **PE\_PRS\_SNK\_SRC\_Reject\_PR\_Swap** state the Policy Engine shall request the Protocol Layer to send:

- A **Reject** Message if the device is unable to perform a Power Role Swap at this time.
- A **Wait** Message if further evaluation of the Power Role Swap request is required. Note: in this case it is expected that one of the Port Partners will send a **PR\_Swap** Message at a later time (see Section 6.3.12.2).

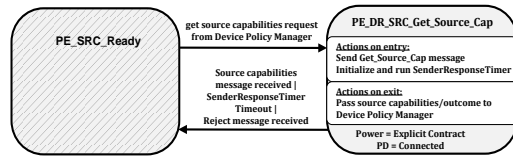
The Policy Engine shall transition to the **PE\_SNK\_Ready** state when:

- The **Reject** or **Wait** Message has been sent.

#### 8.3.3.6.3.3 Dual-Role (Source Port) Get Source Capabilities State Diagram

Figure 8-53 shows the state diagram for a Dual-Role device, presently operating as a Source, on receiving a request from the Device Policy Manager to get the Port Partner's Source capabilities. See also Section 6.4.1.1.3.

Figure 8-53 Dual-Role (Source) Get Source Capabilities diagram



### 8.3.3.6.3.3.1 PE\_DR\_SRC\_Get\_Source\_Cap state

The Policy Engine shall transition to the *PE\_DR\_SRC\_Get\_Source\_Cap* state, from the *PE\_SRC\_Ready* state, due to a request to get the remote source capabilities from the Device Policy Manager.

On entry to the *PE\_DR\_SRC\_Get\_Source\_Cap* state the Policy Engine shall send a *Get\_Source\_Cap* Message and initialize and run the *SenderResponseTimer*.

On exit from the *PE\_DR\_SRC\_Get\_Source\_Cap* state the Policy Engine shall inform the Device Policy Manager of the outcome (capabilities or response timeout).

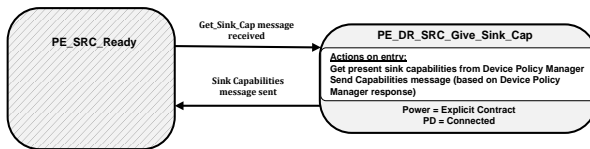
The Policy Engine shall transition back to the *PE\_SRC\_Ready* state (see Figure 8-38) when:

- A *Source\_Capabilities* Message is received
- Or *SenderResponseTimer* times out
- Or a *Reject* Message is received

### 8.3.3.6.3.4 Dual-Role (Source Port) Give Sink Capabilities State Diagram

Figure 8-54 shows the state diagram for a Dual-Role device, presently operating as a Source, on receiving a *Get\_Sink\_Cap* Message. See also Section 6.4.1.1.3.

Figure 8-54 Dual-Role (Source) Give Sink Capabilities diagram



### 8.3.3.6.3.4.1 PE\_DR\_SRC\_Give\_Sink\_Cap state

The Policy Engine shall transition to the *PE\_DR\_SRC\_Give\_Sink\_Cap* state, from the *PE\_SRC\_Ready* state, when a *Get\_Sink\_Cap* Message is received.

On entry to the *PE\_DR\_SRC\_Give\_Sink\_Cap* state the Policy Engine shall request the present capabilities from the Device Policy Manager and then send a *Sink\_Capabilities* Message based on these capabilities.

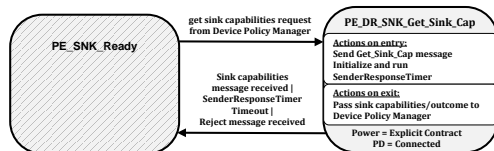
The Policy Engine shall transition back to the *PE\_SRC\_Ready* state (see Figure 8-38) when:

- The *Sink\_Capabilities* Message has been successfully sent.

### 8.3.3.6.3.5 Dual-Role (Sink Port) Get Sink Capabilities State Diagram

Figure 8-55 shows the state diagram for a Dual-Role device, presently operating as a Sink, on receiving a request from the Device Policy Manager to get the Port Partner's Sink capabilities. See also Section 6.4.1.1.3.

Figure 8-55 Dual-Role (Sink) Get Sink Capabilities State Diagram



8.3.3.6.3.5.1 PE\_DR\_SNK\_Get\_Sink\_Cap state

The Policy Engine shall transition to the *PE\_DR\_SNK\_Get\_Sink\_Cap* state, from the *PE\_SNK\_Ready* state, due to a request to get the remote source capabilities from the Device Policy Manager.

On entry to the *PE\_DR\_SNK\_Get\_Sink\_Cap* state the Policy Engine shall send a *Get\_Sink\_Cap* Message and initialize and run the *SenderResponseTimer*.

On exit from the *PE\_DR\_SNK\_Get\_Sink\_Cap* state the Policy Engine shall inform the Device Policy Manager of the outcome (capabilities or response timeout).

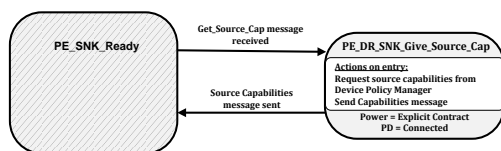
The Policy Engine shall transition back to the *PE\_SNK\_Ready* state (see Figure 8-39 and Figure 8-44) when:

- A *Source\_Capabilities* Message is received
- Or *SenderResponseTimer* times out
- Or a *Reject* Message is received

8.3.3.6.3.6 Dual-Role (Sink Port) Give Source Capabilities State Diagram

Figure 8-56 shows the state diagram for a Dual-Role device, presently operating as a Sink, on receiving a *Get\_Source\_Cap* Message. See also Section 6.4.1.1.3.

Figure 8-56 Dual-Role (Sink) Give Source Capabilities State Diagram



8.3.3.6.3.6.1 PE\_DR\_SNK\_Give\_Source\_Cap state

The Policy Engine shall transition to the *PE\_DR\_SNK\_Give\_Source\_Cap* state, from the *PE\_SNK\_Ready* state, when a *Get\_Source\_Cap* Message is received.

On entry to the *PE\_DR\_SNK\_Give\_Source\_Cap* state the Policy Engine shall request the present capabilities from the Device Policy Manager and then send a *Source\_Capabilities* Message based on these capabilities.

The Policy Engine shall transition back to the *PE\_SNK\_Ready* state (see Figure 8-39 and Figure 8-44) when:

- The *Source\_Capabilities* Message has been successfully sent.

8.3.3.7 Type-C VCONN Swap State Diagrams

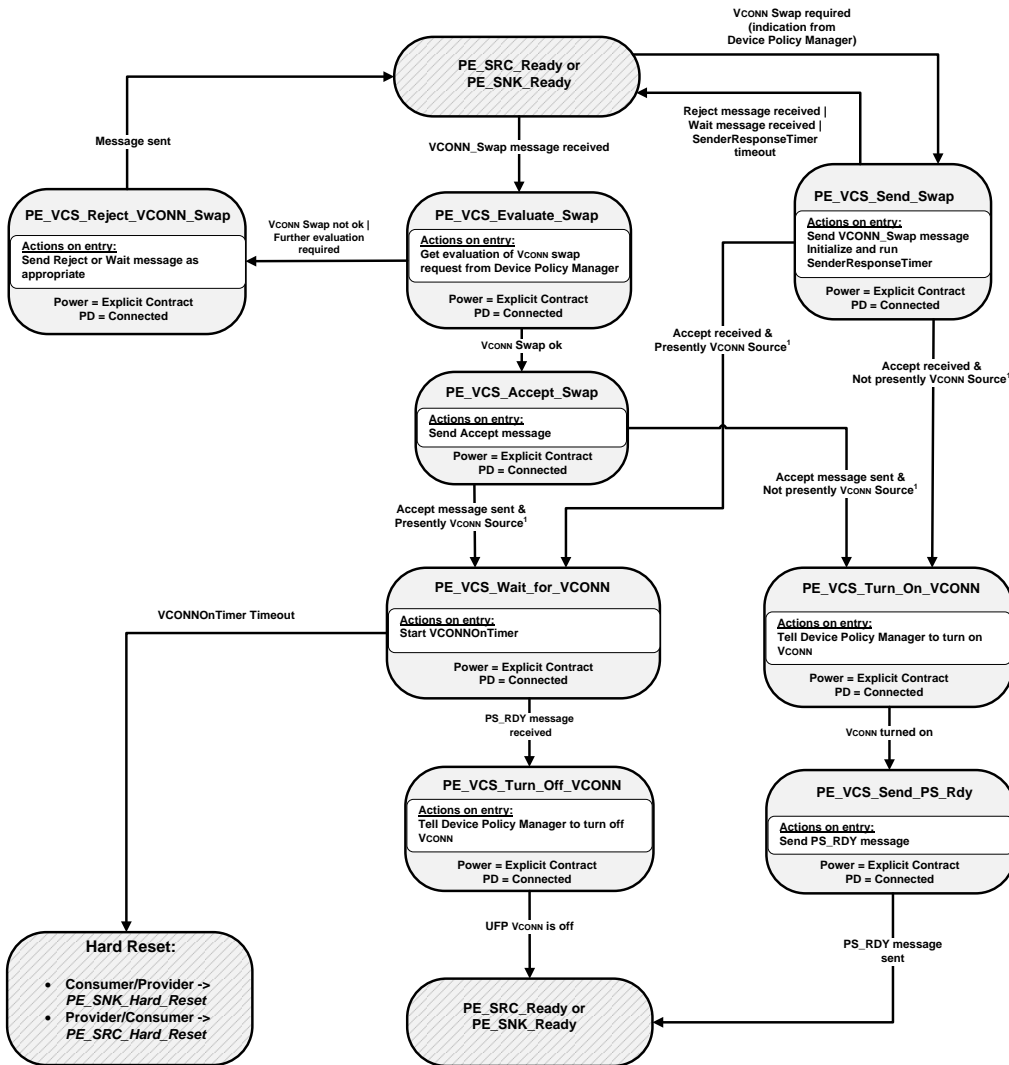
The State Diagrams in this section shall apply to [USBType-C 1.0] Ports that supply VCONN. Figure 8-57

#### 8.3.3.7.1 ~~Type-C DFP VCONN Swap State Diagram~~

shows the state operation for a ~~Downstream Facing Type-C Port (DFP)~~ on ~~initiating sending or receiving a~~ VCONN Swap ~~request~~.



Figure 8-57 ~~DFP~~ VCONN Swap State Diagram



1. A Port is presently the VCONN Source if it has the responsibility for supplying VCONN even if VCONN has been turned off.

8.3.3.7.1.1 PE\_VCS\_DFP\_Send\_Swap state

The *PE\_VCS\_Send\_Swap* state is entered from either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a DFP when the Policy Engine receives a request from the Device Policy Manager to perform a VCONN Swap.

On entry to the **PE\_VCS\_Send\_Swap** state the Policy Engine shall send a **VCONN\_Swap** Message and start the **SenderResponseTimer**.

Field Code Changed

The Policy Engine shall transition to the **PE\_VCS\_Wait\_For\_VCONN** state when:

Field Code Changed

- An **Accept** Message is received and
- DFP current has VCONN turned on.

The Policy Engine shall transition to the **PE\_VCS\_Turn\_On\_VCONN** state when:

- An **Accept** Message is received and
- DFP current has VCONN turned off.

The Policy Engine shall transition back to either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a DFP when:

- A **Reject** Message is received or
- A **Wait** Message is received or
- The **SenderResponseTimer** times out.

#### **8.3.3.7.1.2** PE\_VCS\_DFP\_Wait\_for\_UFP\_VCONN state Evaluate\_Swap

~~On entry to the state the Policy Engine shall start the~~

The

PE_VCS_Evaluate_Swap
----------------------

8.3.3.7.1.2
-------------

~~Policy Engine shall transition to the state when:~~

- ~~• A message is received.~~

~~The Policy Engine shall transition to either the or state when:~~

- ~~• The times out.~~

#### ~~8.3.3.7.1.3~~ PE\_VCS\_DFP\_Turn\_Off\_VCONN state

~~On entry to the state the Policy Engine shall tell the Device Policy Manager to turn off VCONN.~~

~~The Policy Engine shall transition back to the either the or state for a DFP when:~~

- ~~• The DFP's VCONN is off.~~

#### ~~8.3.3.7.1.4~~ PE\_VCS\_DFP\_Turn\_On\_VCONN state

~~On entry to the state the Policy Engine shall tell the Device Policy Manager to turn on VCONN.~~

~~The Policy Engine shall transition to the state when:~~

- ~~• The DFP's VCONN is on.~~

#### ~~8.3.3.7.1.5~~ PE\_VCS\_DFP\_Send\_PS\_Rdy state

~~On entry to the the Policy Engine shall send a message.~~

~~The Policy Engine shall transition back to the either the or state for a DFP when:~~

- ~~• The message has been sent.~~

#### ~~8.3.3.7.2~~ Type C UFP VCONN Swap State Diagram

~~shows the state operation for an Upstream Facing Port (UFP) on receiving a VCONN Swap request.~~

**Figure – UFP VCONN Swap State Diagram**

**8.3.3.7.2.1 PE\_VCS\_UFP\_Evaluate\_Swap**

The state is entered from either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a UFP when the Policy Engine receives a *VCONN\_Swap* Message.

On entry to the

PE_VCS_Evaluate_Swap	8.3.3.7.1.2
----------------------	-------------

state the Policy Engine shall request the Device Policy Manager for an evaluation of the VCONN Swap request.

The Policy Engine shall transition to the *PE\_VCS\_Accept\_Swap* state when:

- The Device Policy Manager indicates that a VCONN Swap is ok.

The Policy Engine shall transition to the *PE\_VCS\_Reject\_Swap* state when:

- The Device Policy Manager indicates that a VCONN Swap is **not** ok or
- The Device Policy Manager indicates that a VCONN Swap cannot be done at this time.

**8.3.3.7.2.2 PE\_VCS\_UFP\_Accept\_Swap**

On entry to the *PE\_VCS\_Accept\_Swap* state the Policy Engine shall send an *Accept* Message.

The Policy Engine shall transition to the *PE\_VCS\_Wait\_For\_VCONN* state when:

- The *Accept* Message has been sent and
- The UFP's VCONN is on.

The Policy Engine shall transition to the *PE\_VCS\_Turn\_On\_VCONN* state when:

- The *Accept* Message has been sent and
- The UFP's VCONN is off.

**8.3.3.7.2.3 PE\_VCS\_UFP\_Reject\_Swap**

On entry to the *PE\_VCS\_Reject\_Swap* state the Policy Engine shall request the Protocol Layer to send:

- A *Reject* Message if the device is unable to perform a VCONN Swap at this time.
- A *Wait* Message if further evaluation of the VCONN Swap request is required. Note: in this case it is expected that the DFP will send a *VCONN\_Swap* Message at a later time.

The Policy Engine shall transition back to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state when:

- The *Reject* or *Wait* Message has been sent.

**8.3.3.7.2.4 PE\_VCS\_UFP\_Wait\_for\_DFP\_VCONN**

On entry to the *PE\_VCS\_Wait\_For\_VCONN* state the Policy Engine shall start the *VCONNOnTimer*.

The Policy Engine shall transition to the *PE\_VCS\_Turn\_Off\_VCONN* state when:

- A *PS\_RDY* Message is received.

The Policy Engine shall transition to either the *PE\_SRC\_Hard\_Reset* or *PE\_SNK\_Hard\_Reset* state when:

- The *VCONNOnTimer* times out.

### 8.3.3.7.2.5 8.3.3.7.1.6 PE\_VCS\_UFP\_Turn\_Off\_VCONN

On entry to the **PE\_VCS\_Turn\_Off\_VCONN** state the Policy Engine shall tell the Device Policy Manager to turn off VCONN. The Policy Engine shall transition back to either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a UFP when:

- The UFP's VCONN is off.

### 8.3.3.7.2.6 8.3.3.7.1.7 PE\_VCS\_UFP\_Turn\_On\_VCONN

On entry to the **PE\_VCS\_Turn\_On\_VCONN** state the Policy Engine shall tell the Device Policy Manager to turn on VCONN. The Policy Engine shall transition to the **PE\_VCS\_Send\_Ps\_Rdy** state when:

- The UFP's VCONN is on.

### 8.3.3.7.2.7 8.3.3.7.1.8 PE\_VCS\_UFP\_Send\_PS\_Rdy

On entry to the **PE\_VCS\_Send\_Ps\_Rdy** state the Policy Engine shall send a **PS\_RDY** Message.

The Policy Engine shall transition back to either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a UFP when:

- The **PS\_RDY** Message has been sent.

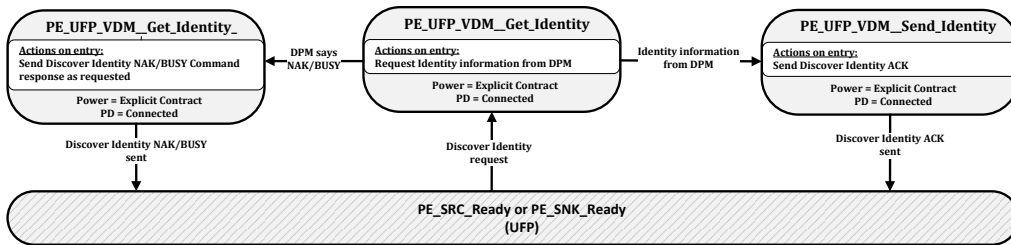
## 8.3.3.8 UFP Structured VDM State Diagrams

The State Diagrams in this section shall apply to all **Ports/UFPs** that support structured VDMs.

### 8.3.3.8.1 UFP Structured VDM Discover Identity State Diagram

Figure 8-58 shows the state diagram for a UFP in response to a **Discover Identity** operation for an Upstream Facing Port (UFP) on receiving a Structured VDM message Command.

Figure 8-58 UFP Structured VDM Discover Identity State Diagram



#### 8.3.3.8.1.1 PE\_UFP\_VDM\_Get\_Identity state

The Policy Engine transitions to the **PE\_UFP\_VDM\_Get\_Identity** state from either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a UFP when:

- A Structured VDM **Discover Identity** Command request is received from the DFP.

On entry to the **PE\_UFP\_VDM\_Get\_Identity** state the UFP shall request identity information from the Device Policy Manager.

The Policy Engine shall transition to the **PE\_UFP\_VDM\_Send\_Identity** state when:

- Identity information is received from the Device Policy Manager.

The Policy Engine shall transition to the *PE\_UFP\_VDM\_Get\_Identity\_NAK* state when:

- The Device Policy Manager indicates that the response to the Discover Identity request is NAK or BUSY.

#### 8.3.3.8.1.2 PE\_UFP\_VDM\_Send\_Identity state

On entry to the *PE\_UFP\_VDM\_Send\_Identity* state the UFP shall send the Structured VDM *Discover Identity* ACK Command response.

The Policy Engine shall transition to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a UFP when:

- The Structured VDM *Discover Identity* ACK Command response has been sent.

#### 8.3.3.8.1.3 PE\_UFP\_VDM\_Get\_Identity\_NAK

On entry to the *PE\_UFP\_VDM\_Get\_Identity\_NAK* state the Policy Engine shall send a Structured VDM *Discover Identity* NAK or BUSY Command response as indicated by the Device Policy Manager.

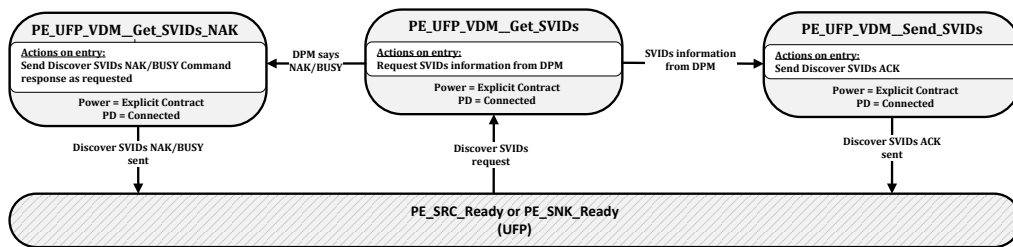
The Policy Engine shall transition to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a UFP when:

- The Structured VDM *Discover Identity* NAK or BUSY Command response has been sent.

#### 8.3.3.8.2 UFP Structured VDM Discover SVIDs State Diagram

Figure 8-59 shows the state diagram for a UFP in response to a *Discover SVIDs* Command.

Figure 8-59 UFP Structured VDM Discover SVIDs State Diagram



#### 8.3.3.8.2.1 PE\_UFP\_VDM\_Get\_SVIDs state

The Policy Engine transitions to the *PE\_UFP\_VDM\_Get\_SVIDs* state from either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a UFP when:

- A Structured VDM *Discover SVIDs* Command request is received from the DFP.

On entry to the *PE\_UFP\_VDM\_Get\_SVIDs* state the UFP shall request SVIDs information from the Device Policy Manager.

The Policy Engine shall transition to the *PE\_UFP\_VDM\_Send\_SVIDs* state when:

- SVIDs information is received from the Device Policy Manager.

The Policy Engine shall transition to the *PE\_UFP\_VDM\_Get\_SVIDs\_NAK* state when:

- The Device Policy Manager indicates that the response to the Discover Identity request is NAK or BUSY.

#### 8.3.3.8.2.2 PE\_UFP\_VDM\_Send\_SVIDs state

On entry to the *PE\_UFP\_VDM\_Send\_SVIDs* state the UFP shall send the Structured VDM *Discover SVIDs* ACK Command response.

The Policy Engine shall transition to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a UFP when:

- The Structured VDM *Discover SVIDs* ACK Command ~~request~~response has been sent.

### 8.3.3.8.2.3 PE\_UFP\_VDM\_Get\_SVIDs\_NAK

On entry to the *PE\_UFP\_VDM\_Get\_SVIDs\_NAK* state the Policy Engine shall send a Structured VDM *Discover SVIDs NAK* or *BUSY* Command response as indicated by the Device Policy Manager.

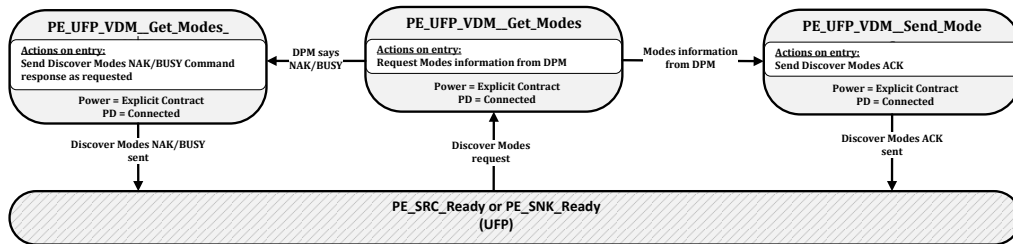
The Policy Engine shall transition to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a UFP when:

- The Structured VDM *Discover SVIDs NAK* or *BUSY* Command response has been sent.

### 8.3.3.8.3 UFP Structured VDM Discover Modes State Diagram

Figure 8-60 shows the state diagram for a UFP in response to a *Discover Modes Command*.

Figure 8-60 UFP Structured VDM Discover Modes State Diagram



### 8.3.3.8.1.58.3.3.8.3.1 PE\_UFP\_VDM\_Get\_Modes state

The Policy Engine transitions to the *PE\_UFP\_VDM\_Get\_Modes* state from either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a UFP when:

- A Structured VDM *Discover Modes* Command request is received from the DFP.

On entry to the *PE\_UFP\_VDM\_Get\_Modes* state the UFP shall request Modes information from the Device Policy Manager.

The Policy Engine shall transition to the *PE\_UFP\_VDM\_Send\_Modes* state when:

- Modes information is received from the Device Policy Manager.

The Policy Engine shall transition to the *PE\_UFP\_VDM\_Get\_Modes\_NAK* state when:

- The Device Policy Manager indicates that the response to the *Discover Identity* request is *NAK* or *BUSY*.

### 8.3.3.8.1.68.3.3.8.3.2 PE\_UFP\_VDM\_Send\_Modes state

On entry to the *PE\_UFP\_VDM\_Send\_Modes* state the UFP shall send the Structured VDM *Discover Modes* ACK Command response.

The Policy Engine shall transition to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a UFP when:

- The Structured VDM *Discover Modes* ACK Command response has been sent.

### 8.3.3.8.3.3 PE UFP VDM Get Modes NAK

On entry to the **PE\_CBL\_Get\_Modes\_NAK** state the Policy Engine shall send a Structured VDM **Discover Modes\_NAK** or **BUSY** Command response as indicated by the Device Policy Manager.

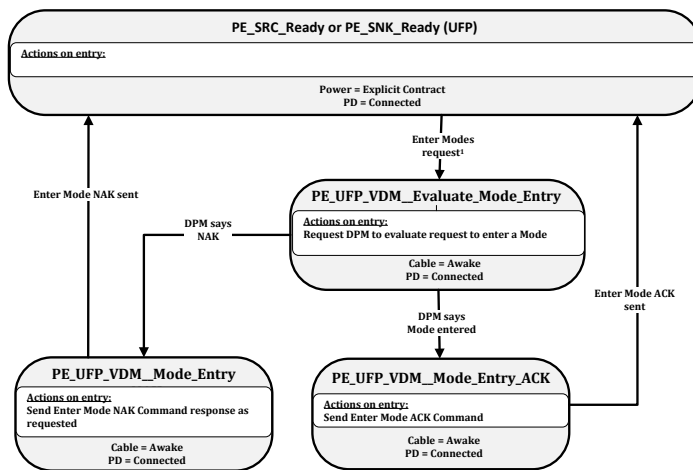
The Policy Engine shall transition to either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a UFP when:

- The Structured VDM **Discover Modes\_NAK** or **BUSY** Command response has been sent.

### 8.3.3.8.4 UFP Structured VDM Enter Mode State Diagram

Figure 8-61 shows the state diagram for a UFP in response to an **Enter Mode Command**.

Figure 8-61 UFP Structured VDM Enter Mode State Diagram



### 8.3.3.8.4.1 PE\_UFP\_VDM\_Evaluate\_Mode\_Entry state

The Policy Engine transitions to the **PE\_UFP\_VDM\_Evaluate\_Mode\_Entry** state from either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a UFP when:

- A Structured VDM **Enter Mode** Command request is received from the DFP.

On Entry to the **PE\_UFP\_VDM\_Evaluate\_Mode\_Entry** state the Policy Engine shall request the Device Policy Manager to evaluate the **Enter Mode** Command request and enter the Mode indicated in the Command request if the request is acceptable.

The Policy Engine shall transition to the **PE\_UFP\_VDM\_Mode\_Entry\_ACK** state when:

- The Device Policy Manager indicates that the Mode has been entered.

The Policy Engine shall transition to the **PE\_UFP\_VDM\_Mode\_Entry\_NAK** state when:

- The Device Policy Manager indicates that the ~~Command~~ response to the ~~CommandMode~~ request is NAK.

### 8.3.3.8.1-108.3.3.8.4.2 PE\_UFP\_VDM\_Mode\_Entry\_ACK state

On entry to the **PE\_UFP\_VDM\_Mode\_Entry\_ACK** state the Policy Engine shall send a Structured VDM *Enter Mode* ACK Command response.

The Policy Engine shall transition to either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a UFP when:

- The Structured VDM *Enter Mode* ACK Command response has been sent.

### 8.3.3.8.1-108.3.3.8.4.3 PE\_UFP\_VDM\_Mode\_Entry\_NAK state

On entry to the **PE\_UFP\_VDM\_Mode\_Entry\_NAK** state the Policy Engine shall send a Structured VDM *Enter Mode* NAK Command response as indicated by the Device Policy Manager.

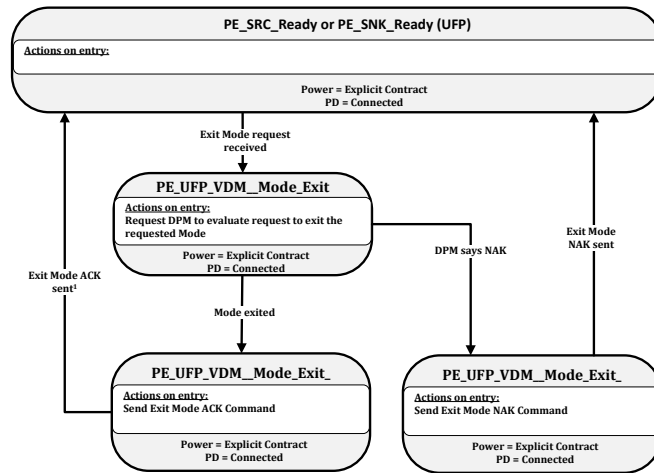
The Policy Engine shall transition to either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a UFP when:

- The Structured VDM *Enter Mode* NAK Command response has been sent.

### 8.3.3.8.5 UFP Structured VDM Exit Mode State Diagram

Figure 8-62 shows the state diagram for a UFP in response to an *Exit Mode* Command.

Figure 8-62 UFP Structured VDM Exit Mode State Diagram



<sup>1</sup> The UFP is required to be in USB operation or USB Safe State at this point.

### 8.3.3.8.1-108.3.3.8.5.1 PE\_UFP\_VDM\_Mode\_Exit state

The Policy Engine transitions to the **PE\_UFP\_VDM\_Mode\_Exit** state from either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a UFP when:

- A Structured VDM *Exit Mode* Command request is received from the DFP.

On entry to the **PE\_UFP\_VDM\_Mode\_Exit** state the Policy Engine shall request the Device Policy Manager to exit the Mode indicated in the Command.

The Policy Engine shall transition to the **PE\_UFP\_VDM\_Mode\_Exit\_ACK** state when:



- ~~The indicated~~The Device Policy Manger indicates that the Mode has been exited.

The Policy Engine shall transition to the **PE UFP VDM Mode Exit NAK** state when:

- ~~The Device Policy Manager indicates that the Command response to the Exit Mode~~**PE\_UFP\_VDM\_Command request is NAK.**

~~8.3.3.8.1.11~~**8.3.3.8.5.2** **PE\_CBL\_Mode\_Exit\_ACK** state

On entry to the **PE\_UFP\_VDM\_Mode\_Exit\_ACK** state the Policy Engine shall send a Structured VDM **Exit Mode ACK** Command response.

The Policy Engine shall transition to either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a UFP when:

- The Structured VDM **Exit Mode ACK** Command response has been sent.

**8.3.3.8.5.3** **PE\_UFP\_VDM\_Mode\_Exit\_NAK** state

~~On entry to the PE\_UFP\_VDM\_Mode\_Exit\_NAK state the Policy Engine shall send a Structured VDM Exit Mode NAK Command response as indicated by the Device Policy Manager.~~

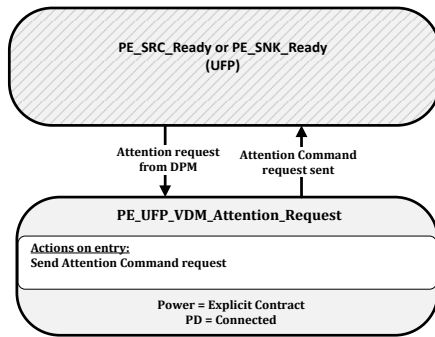
~~The Policy Engine shall transition to either the either the PE\_SRC\_Ready or PE\_SNK\_Ready state for a UFP when:~~

- ~~The Structured VDM Exit Mode~~**UFP\_NAK** Command response has been sent.

~~8.3.3.8.28~~**8.3.3.8.6** **UFP Structured VDM Attention State Diagram**

Figure 8-70 shows the state diagram for a UFP when sending an **Attention** Command request.

**Figure 8-63 UFP VDM Attention State Diagram**



~~8.3.3.8.2.18~~**8.3.3.8.6.1** **PE\_UFP\_VDM\_Attention\_Request**

The Policy Engine transitions to the **PE\_UFP\_VDM\_Attention\_Request** state from either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a ~~DFF~~**UFP** when:

- When the Device Policy Manager requests attention from the DFP.

On entry to the **PE\_UFP\_VDM\_Attention\_Request** state the Policy Engine shall send an **Attention** Command request.

The Policy Engine shall transition to either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a ~~DFF~~**UFP** when:

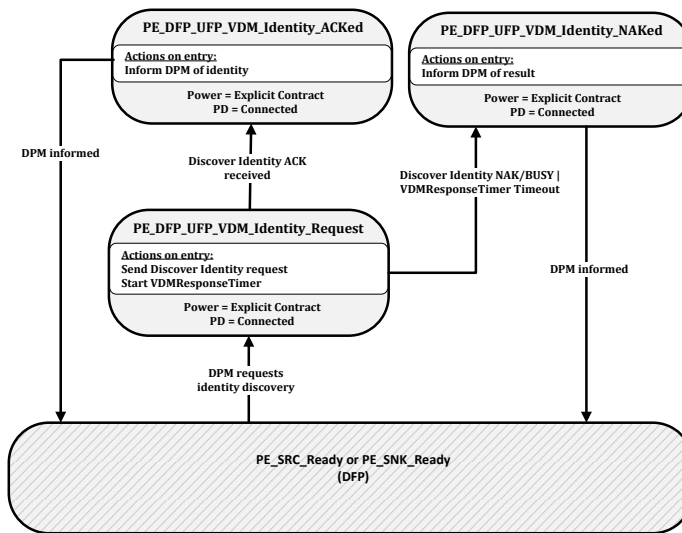
- The **Attention** Command request has been sent.

### 8.3.3.9 DFPDFP Structured VDM State Diagrams

#### 8.3.3.9.1 DFP to UFP Structured VDM Discover Identity State Diagram

Figure 8-64 shows the state diagram for a DFP when discovering the identity of a UFP.

Figure 8-64 DFP to UFP VDM Discover Identity State Diagram



#### 8.3.3.9.1.1 PE\_DFP\_VDM\_Identity\_Request state

The Policy Engine transitions to the *PE\_DFP\_UFP\_VDM\_Identity\_Request* state from either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a DFP when:

- The Device Policy Manager requests the discovery of the identity of the Port Partner or a Cable Plug.

On entry to the *PE\_DFP\_UFP\_VDM\_Identity\_Request* state the Policy Engine shall send a Structured VDM *Discover Identity* Command request and shall start the *VDMResponseTimer*.

The Policy Engine shall transition to the *PE\_DFP\_UFP\_VDM\_Identity\_ACKed* state when:

- A Structured VDM *Discover Identity* ACK Command response is received.

The Policy Engine shall transition to the *PE\_DFP\_UFP\_VDM\_Identity\_NAKed* state when:

- A Structured VDM *Discover Identity* NAK or BUSY Command response is received or
- The *VDMResponseTimer* times out

#### 8.3.3.9.1.2 PE\_DFP\_VDM\_Identity\_ACKed state

~~The times out.~~

#### 8.3.3.9.1.1.1 PE\_DFP\_VDM\_Identity\_ACKed state

On entry to the *PE\_DFP\_UFP\_VDM\_Identity\_ACKed* state the Policy Engine shall inform the Device Policy Manager of the Identity information.

The Policy Engine shall transition to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a DFP when:

- The Device Policy Manager has been informed.

~~8.3.3.8.3~~ **8.3.3.9.1.3** PE\_DFP\_VDM\_Identity\_NAKed state

On entry to the **PE\_DFP\_VDM\_Identity\_NAKed** state the Policy Engine shall inform the Device Policy Manager of the result (NAK, BUSY or timeout).

~~The Policy Engine shall transition to either the or state for a DFP when:~~

- ~~• The Device Policy Manager has been informed.~~

~~8.3.3.8.4~~ **DFP VDM Discover SVIDs State Diagram**

~~shows the state diagram for a DFP when discovering SVIDs.~~

~~Figure – DFP VDM Discover SVIDs State Diagram~~

~~8.3.3.8.4.1~~ **PE\_DFP\_VDM\_SVIDs\_Request state**

~~The Policy Engine transitions to the state from either the or state for a DFP when:~~

- ~~• The Device Policy Manager requests the discovery of the SVIDs of the Port Partner or a Cable Plug.~~

~~On entry to the state the Policy Engine shall send a Structured VDM Command request and shall start the.~~

~~The Policy Engine shall transition to the state when:~~

- ~~• A Structured VDM ACK Command response is received.~~

~~The Policy Engine shall transition to the state when:~~

- ~~• A Structured VDM NAK or BUSY Command response is received or~~
- ~~• The times out.~~

~~8.3.3.8.4.2~~ **PE\_DFP\_VDM\_SVIDs\_ACKed state**

~~On entry to the state the Policy Engine shall inform the Device Policy Manager of the SVIDs information.~~

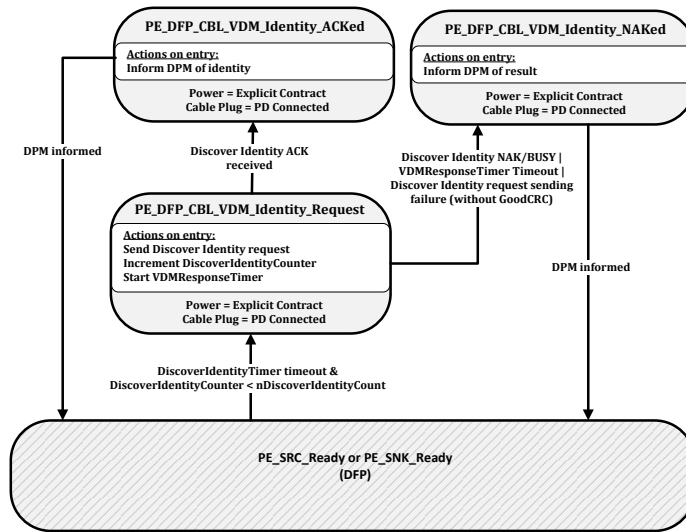
The Policy Engine shall transition to either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a DFP when:

- The Device Policy Manager has been informed.

**8.3.3.9.2 DFP to Cable Plug Structured VDM Discover Identity State Diagram**

Figure 8-64 shows the state diagram for a DFP when discovering the identity of a Cable Plug.

**Figure 8-65 DFP VDM Discover Identity State Diagram**



**8.3.3.8.4.38.3.3.9.2.1 PE\_DFP\_VDM\_SVIDs\_NAKedIdentity\_Request state**

The Policy Engine transitions to the **PE DFP CBL VDM Identity Request** state from either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a DFP when:

- The **DiscoverIdentityTimer** times out and
- The **DiscoverIdentityCounter** is less than **nDiscoverIdentityCount**.

On entry to the **PE DFP CBL VDM Identity Request** state the Policy Engine shall send a Structured VDM **Discover Identity** Command request, shall increment the **DiscoverIdentityCounter** and shall start the **VDMResponseTimer**.

The Policy Engine shall transition to the **PE DFP CBL VDM Identity ACKed** state when:

- A Structured VDM **Discover Identity ACK** Command response is received.

The Policy Engine shall transition to the **PE DFP CBL VDM Identity NAKed** state when:

- A Structured VDM **Discover Identity NAK** or **BUSY** Command response is received or
- The **VDMResponseTimer** times out or
- The Structured VDM **Discover Identity** Command request Message sending fails (no **GoodCRC** Message received after retries).

**8.3.3.9.2.2 PE\_DFP\_VDM\_Identity\_ACKed state**

On entry to the **PE DFP CBL VDM Identity ACKed** state the Policy Engine shall inform the Device Policy Manager of the **result (NAK, BUSY or timeout) Identity information**.

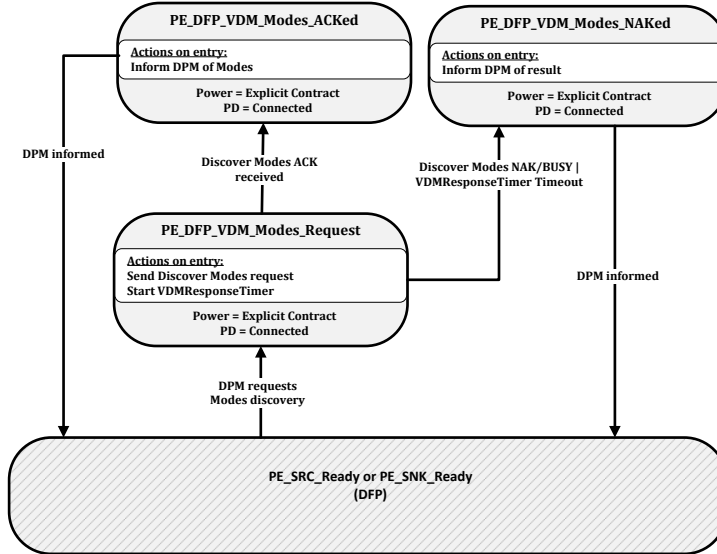
The Policy Engine shall transition to either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a DFP when:

- The Device Policy Manager has been informed.

**8.3.3.8.5 DFP VDM Discover Modes State Diagram**

~~shows the state diagram for a DFP when discovering Modes.~~

**Figure 8-66 DFP VDM Discover Modes State Diagram**



**8.3.3.9.5.18.3.3.9.2.3 PE\_DFP\_VDM\_Modes\_Request/Identity\_NAKed state**

The Policy Engine transitions to the state from either the or state for a DFP when:

- The Device Policy Manager requests the discovery of the Modes of the Port Partner or a Cable Plug.

On entry to the state the Policy Engine shall send a Structured VDM Command request and shall start the

The Policy Engine shall transition to the state when:

- A Structured VDM ACK Command response is received.

The Policy Engine shall transition to the state when:

- A Structured VDM NAK or BUSY Command response is received or
- The VDMResponseTimer times out PE\_DFP\_CBL\_VDM\_Identity\_NAKed.

**8.3.3.9.5.2 PE\_DFP\_VDM\_Modes\_ACKed state**

On entry to the state the Policy Engine shall inform the Device Policy Manager of the Modes information result (NAK, BUSY or timeout).

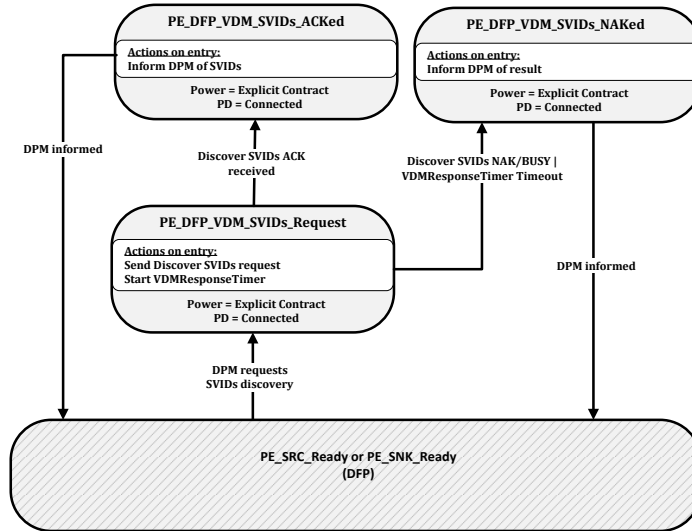
The Policy Engine shall transition to either the PE\_SRC\_Ready or PE\_SNK\_Ready state for a DFP when:

- The Device Policy Manager has been informed.

**8.3.3.9.3 DFP Structured VDM Discover SVIDs State Diagram**

Figure 8-66 shows the state diagram for a DFP when discovering SVIDs.

**Figure 8-66 DFP VDM Discover SVIDs State Diagram**



**8.3.3.9.3.1 PE\_DFP\_VDM\_Modes\_NAKedSVIDs\_Request state**

The Policy Engine transitions to the *PE\_DFP\_VDM\_SVIDs\_Request* state from either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a DFP when:

- The Device Policy Manager requests the discovery of the SVIDs of the Port Partner or a Cable Plug.

On entry to the *PE\_DFP\_VDM\_SVIDs\_Request* state the Policy Engine shall send a Structured VDM *Discover SVIDs* Command request and shall start the *VDMResponseTimer*.

The Policy Engine shall transition to the *PE\_DFP\_VDM\_SVIDs\_ACKed* state when:

- A Structured VDM *Discover SVIDs ACK* Command response is received.

The Policy Engine shall transition to the *PE\_DFP\_VDM\_SVIDs\_NAKed* state when:

- A Structured VDM *Discover SVIDs NAK* or *BUSY* Command response is received or
- The *VDMResponseTimer* times out.

**8.3.3.9.3.2 PE\_DFP\_VDM\_SVIDs\_ACKed state**

On entry to the *PE\_DFP\_VDM\_SVIDs\_ACKed* state the Policy Engine shall inform the Device Policy Manager of the result (NAK, BUSY or timeout)-SVIDs information.

The Policy Engine shall transition to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a DFP when:

- The Device Policy Manager has been informed.

**8.3.3.9.3.3 PE\_DFP\_VDM\_SVIDs\_NAKed state**

On entry to the *PE\_DFP\_VDM\_SVIDs\_NAKed* state the Policy Engine shall inform the Device Policy Manager of the result (NAK, BUSY or timeout).

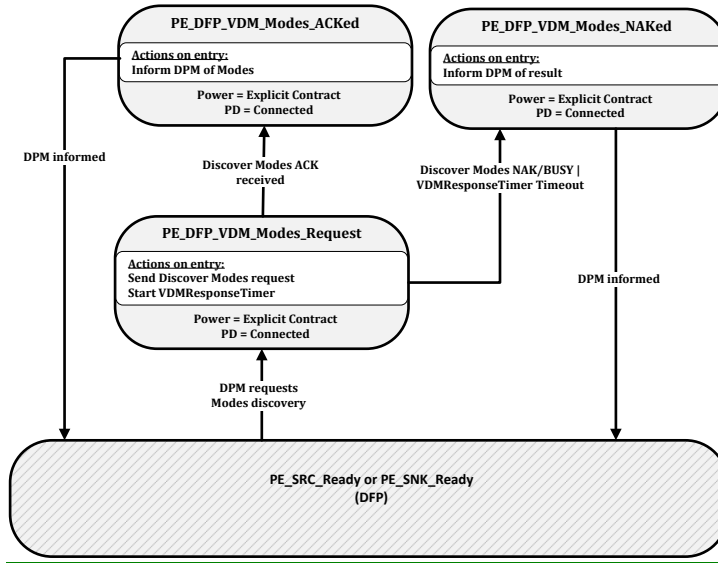
The Policy Engine shall transition to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a DFP when:

- The Device Policy Manager has been informed.

#### 8.3.3.9.4 DFP Structured VDM Discover Modes State Diagram

Figure 8-67 shows the state diagram for a DFP when discovering Modes.

Figure 8-67 DFP VDM Discover Modes State Diagram



##### 8.3.3.9.4.1 PE DFP VDM Modes Request state

The Policy Engine transitions to the *PE\_DFP\_VDM\_Modes\_Request* state from either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a DFP when:

- The Device Policy Manager requests the discovery of the Modes of the Port Partner or a Cable Plug.

On entry to the *PE\_DFP\_VDM\_Modes\_Request* state the Policy Engine shall send a Structured VDM *Discover Modes* Command request and shall start the *VDMResponseTimer*.

The Policy Engine shall transition to the *PE\_DFP\_VDM\_Modes\_ACKed* state when:

- A Structured VDM *Discover Modes* ACK Command response is received.

The Policy Engine shall transition to the *PE\_DFP\_VDM\_Modes\_NAKed* state when:

- A Structured VDM *Discover Modes* NAK or BUSY Command response is received or
- The *VDMResponseTimer* times out.

##### 8.3.3.9.4.2 PE DFP VDM Modes ACKed state

On entry to the *PE\_DFP\_VDM\_Modes\_ACKed* state the Policy Engine shall inform the Device Policy Manager of the Modes information.

The Policy Engine shall transition to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a DFP when:

- The Device Policy Manager has been informed.

### 8.3.3.9.4.3 PE DFP VDM Modes NAKed state

On entry to the *PE\_DFP\_VDM\_Modes\_NAKed* state the Policy Engine shall inform the Device Policy Manager of the result (NAK, BUSY or timeout).

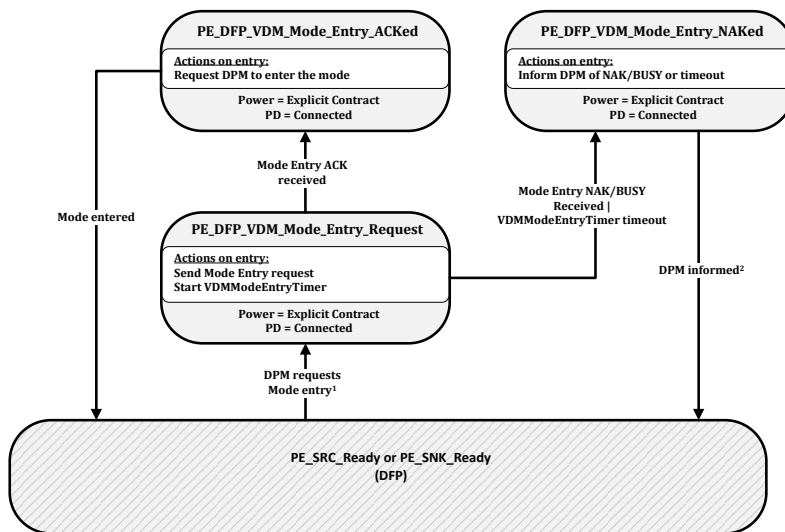
The Policy Engine shall transition to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a DFP when:

- The Device Policy Manager has been informed.

### 8.3.3.9.5 DFP Structured VDM Mode Entry State Diagram

Figure 8-68 shows the state operation for a DFP when entering a Mode.

Figure 8-68 DFP VDM Mode Entry State Diagram



<sup>1</sup> The Device Policy Manager is required to have place the system into USB Safe State before issuing this request when entering Modal operation.

<sup>2</sup> The Device Policy Manager is required to return the system to USB operation if not in Modal operation at this point.

### 8.3.3.9.5.1 PE\_DFP\_VDM\_Mode\_Entry\_Request state

The Policy Engine transitions to the *PE\_DFP\_VDM\_Mode\_Entry\_Request* state from either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a DFP when:

- The Device Policy Manager requests that the Port Partner or a Cable Plug enter a Mode.

On entry to the *PE\_DFP\_VDM\_Mode\_Entry\_Request* state the Policy Engine shall send a Structured VDM *Enter Mode* Command request and shall start the *VDMModeEntryTimer*.

The Policy Engine shall transition to the *PE\_DFP\_VDM\_Mode\_Entry\_ACKed* state when:

- A Structured VDM *Enter Mode* ACK Command response is received.

The Policy Engine shall transition to the *PE\_DFP\_VDM\_Mode\_Entry\_NAKed* state when:

- A Structured VDM *Enter Mode* NAK or BUSY Command response is received or
- The *VDMModeEntryTimer* times out.



~~8.3.3.8.6.28~~ 8.3.3.9.5.2 PE\_DFP\_VDM\_Mode\_Entry\_ACKed state

On entry to the **PE\_DFP\_VDM\_Mode\_Entry\_ACKed** state the Policy Engine shall request the Device Policy Manager to enter the Mode.

The Policy Engine shall transition to either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a DFP when:

- ~~The Mode has been entered.~~
- ~~The Device Policy Manager has been informed.~~

~~8.3.3.8.6.28~~ 8.3.3.9.5.3 PE\_DFP\_VDM\_Mode\_Entry\_NAKed state

On entry to the **PE\_DFP\_VDM\_Mode\_Entry\_NAKed** state the Policy Engine shall inform the Device Policy Manager of the result (NAK, BUSY or timeout).

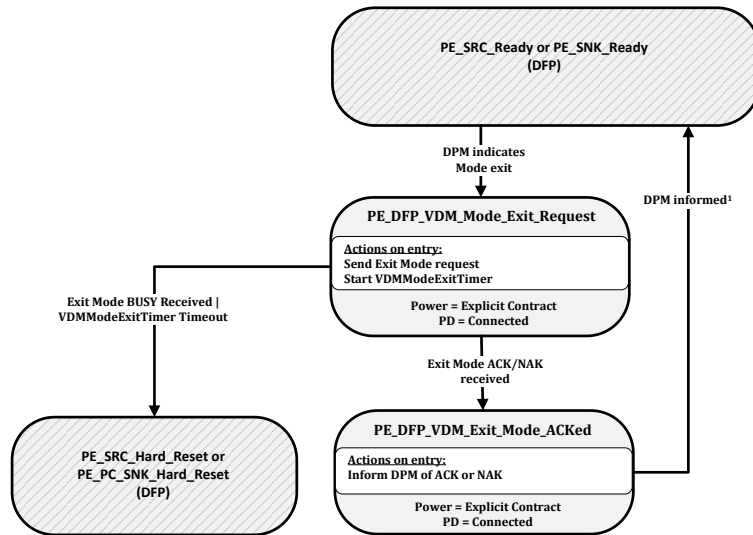
The Policy Engine shall transition to either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a DFP when:

- The Device Policy Manager has been informed.

~~8.3.3.8.78~~ 8.3.3.9.6 DFP Structured VDM Mode Exit State Diagram

Figure 8-69 shows the state diagram for a DFP when exiting a Mode.

Figure 8-69 DFP VDM Mode Exit State Diagram



<sup>1</sup> The Device Policy Manager is required to return the system to USB operation at this point when exiting Modal Operation.

~~8.3.3.8.7.18~~ 8.3.3.9.6.1 PE\_DFP\_VDM\_Mode\_Exit\_Request state

The Policy Engine transitions to the **PE\_DFP\_VDM\_Mode\_Exit\_Request** state from either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state for a DFP when:

- The Device Policy Manager requests that the Port Partner or a Cable Plug exit a Mode.

On entry to the *PE\_DFP\_VDM\_Mode\_Exit\_Request* state the Policy Engine shall send a Structured VDM *Exit Mode* Command request and shall start the *VDMModeExitTimer*.

The Policy Engine shall transition to the *PE\_DFP\_VDM\_Mode\_Exit\_ACKed* state when:

- A Structured VDM *Exit Mode* ACK or NAK Command response is received.

The Policy Engine shall transition to either the *PE\_SRC\_Hard\_Reset* or *PE\_PC\_SNK\_Hard\_Reset* state for a DFP when:

- A Structured VDM *Exit Mode* NAK or BUSY Command response is received or
- The *VDMModeExitTimer* times out.

#### ~~8.3.3.8.7~~ 8.3.3.9.6.2 PE\_DFP\_VDM\_DFP\_Mode\_Exit\_ACKed state

On Exit to the *PE\_DFP\_VDM\_Mode\_Exit\_ACKed* state the Policy Engine shall request the Device Policy Manager to exit the Mode.

The Policy Engine shall transition to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a DFP when:

- ~~The Device Policy Manager has been informed.~~

#### ~~8.3.3.8.8~~ Source Startup VDM Discover Identity Diagram

~~shows the state diagram for Source discovery of identity information during the startup sequence.~~

#### ~~Figure – Source Startup VDM Discover Identity Diagram~~

#### ~~8.3.3.8.8.1~~ PE\_SRC\_VDM\_Identity\_Request state

~~The Policy Engine transitions to the state from the state when:~~

- ~~The Device Policy Manager requests the discovery of the identity of the Port Partner or a Cable Plug.~~

~~On entry to the state the Policy Engine shall send a Structured VDM Command request and shall start the~~

~~The Policy Engine shall transition to the state when:~~

- ~~A Structured VDM ACK Command response is received.~~

~~The Policy Engine shall transition to the state when:~~

- ~~A Structured VDM NAK or BUSY Command response is received or~~

~~• The VDMResponseTimer times out.~~

#### ~~8.3.3.8.8.2~~ PE\_SRC\_VDM\_Identity\_ACKed state

~~On entry to the state the Policy Engine shall inform the Device Policy Manager of the Identity information.~~

~~The Policy Engine shall transition to the state when:~~

- ~~The Device Policy Manager has been informed.~~

#### ~~8.3.3.8.8.3~~ PE\_SRC\_VDM\_Identity\_NAKed state

~~On entry to the state the Policy Engine shall inform the Device Policy Manager of the result (NAK, BUSY or timeout).~~

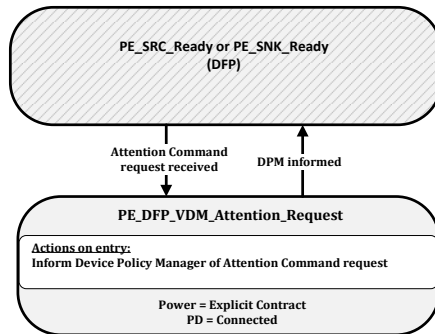
~~The Policy Engine shall transition to the state when:~~

- The Device Policy Manager has been informed.

8.3.3.9.8.3.3.9.7 ~~\_\_\_\_\_~~ DFP **Structured** VDM Attention State Diagram

Figure 8-70 shows the state diagram for a DFP when receiving an *Attention* Command request.

Figure 8-70 DFP VDM Attention State Diagram



8.3.3.9.9.18.3.3.9.7.1 ~~\_\_\_\_\_~~ PE\_DFP\_VDM\_Attention\_Request

The Policy Engine transitions to the *PE\_DFP\_VDM\_Attention\_Request* state from either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a DFP when:

- An *Attention* Command request is received.

On entry to the *PE\_DFP\_VDM\_Attention\_Request* state the Policy Engine shall inform the Device Policy Manager of the attention request.

The Policy Engine shall transition to either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* state for a DFP when:

- The Device Policy Manager has been informed.

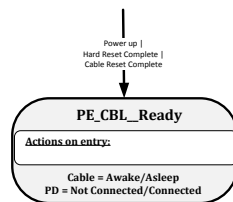
8.3.3.9.8.3.3.10 ~~\_\_\_\_\_~~ **USB to USB** Cable **Plug Related** State Diagrams

The State Diagrams in this section shall apply to all Cable Plugs that support Structured VDMs.

8.3.3.9.18.3.3.10.1 ~~\_\_\_\_\_~~ **General** Cable **VDM** Plug **Cable Ready** State Diagram

Figure 8-71 shows the **general** **Cable Ready** state diagram for a Cable Plug ~~in response to Discover and Mode related requests.~~

Figure 8-71 ~~General~~ Cable **Ready** VDM State Diagram



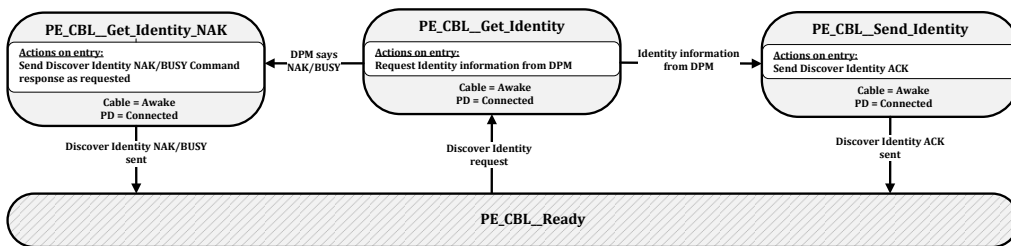
### ~~8.3.3.9.1.18~~ 8.3.3.10.1.1 PE\_CBL\_Ready state

The *PE\_CBL\_Ready* state [shown in the following sections](#) is the normal operational state for a Cable Plug and where it starts after power up or a Hard/Cable Reset.

#### 8.3.3.10.2 Cable Plug Structured VDM Discover Identity State Diagram

Figure 8-72 shows the state diagram for a Cable Plug in response to a *Discover Identity Command*.

Figure 8-72 Cable Plug Structured VDM Discover Identity State Diagram



### ~~8.3.3.9.1.28~~ 8.3.3.10.2.1 PE\_CBL\_Get\_Identity state

The Policy Engine transitions to the *PE\_CBL\_Get\_Identity* state from the *PE\_CBL\_Ready* state when:

- A Structured VDM *Discover Identity* Command request is received from the DFP.

On entry to the *PE\_CBL\_Get\_Identity* state the Cable shall request identity information from the Device Policy Manager.

The Policy Engine shall transition to the *PE\_CBL\_Send\_Identity* state when:

- Identity information is received from the Device Policy Manager.

The Policy Engine shall transition to the *PE\_CBL\_Get\_Identity\_NAK* state when:

- The Device Policy Manager indicates that the response to the Discover Identity request is NAK or BUSY.

### ~~8.3.3.9.1.38~~ 8.3.3.10.2.2 PE\_CBL\_Send\_Identity state

On entry to the *PE\_CBL\_Send\_Identity* state the Cable shall send the Structured VDM *Discover Identity* ACK Command response.

The Policy Engine shall transition to the *PE\_CBL\_Ready* state when:

- The Structured VDM *Discover Identity* ACK Command response has been sent.

#### 8.3.3.10.2.3 PE\_CBL\_Get\_Identity\_NAK

On entry to the *PE\_CBL\_Get\_Identity\_NAK* state the Policy Engine shall send a Structured VDM *Discover Identity\_NAK* or *BUSY* Command response as indicated by the Device Policy Manager.

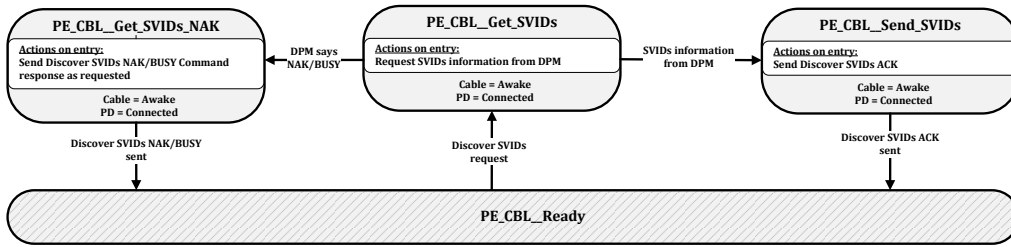
The Policy Engine shall transition to the *PE\_CBL\_Ready* state when:

- The Structured VDM *Discover Identity\_NAK* or *BUSY* Command response has been sent.

### 8.3.3.10.3 Cable Plug Structured VDM Discover SVIDs State Diagram

Figure 8-73 shows the state diagram for a Cable Plug in response to a *Discover SVIDs* Command.

Figure 8-73 Cable Plug Structured VDM Discover SVIDs State Diagram



#### 8.3.3.9.1.48.3.3.10.3.1 PE\_CBL\_Get\_SVIDs state

The Policy Engine transitions to the *PE\_CBL\_Get\_SVIDs* state from the *PE\_CBL\_Ready* state when:

- A Structured VDM *Discover SVIDs* Command request is received from the DFP.

On entry to the *PE\_CBL\_Get\_SVIDs* state the Cable shall request SVIDs information from the Device Policy Manager.

The Policy Engine shall transition to the *PE\_CBL\_Send\_SVIDs* state when:

- SVIDs information is received from the Device Policy Manager.

The Policy Engine shall transition to the *PE\_CBL\_Get\_SVIDs\_NAK* state when:

- The Device Policy Manager indicates that the response to the Discover Identity request is NAK or BUSY.

#### 8.3.3.9.1.58.3.3.10.3.2 PE\_CBL\_Send\_SVIDs state

On entry to the *PE\_CBL\_Send\_SVIDs* state the Cable shall send the Structured VDM *Discover SVIDs* ACK Command response.

The Policy Engine shall transition to the *PE\_CBL\_Ready* state when:

- The Structured VDM *Discover SVIDs* ACK Command response has been sent.

#### 8.3.3.10.3.3 PE\_CBL\_Get\_SVIDs\_NAK

On entry to the *PE\_CBL\_Get\_SVIDs\_NAK* state the Policy Engine shall send a Structured VDM *Discover SVIDs NAK* or *BUSY* Command response as indicated by the Device Policy Manager.

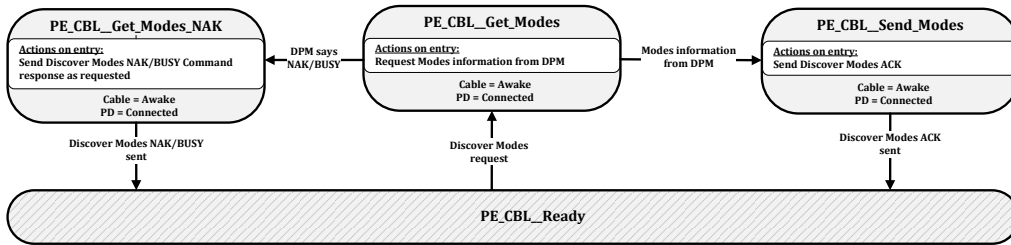
The Policy Engine shall transition to the *PE\_CBL\_Ready* state when:

- The Structured VDM *Discover SVIDs NAK* or *BUSY* Command response has been sent.

### 8.3.3.10.4 Cable Plug Structured VDM Discover Modes State Diagram

Figure 8-74 shows the state diagram for a Cable Plug in response to a *Discover Modes* Command.

**Figure 8-74 Cable Plug Structured VDM Discover Modes State Diagram**



**8.3.3.9.1.6 8.3.3.10.4.1 PE\_CBL\_Get\_Modes state**

The Policy Engine transitions to the *PE\_CBL\_Get\_Modes* state from the *PE\_CBL\_Ready* state when:

- A Structured VDM *Discover Modes* Command request is received from the DFP.

On entry to the *PE\_CBL\_Get\_Modes* state the Cable shall request Modes information from the Device Policy Manager.

The Policy Engine shall transition to the *PE\_CBL\_Send\_Modes* state when:

- Modes information is received from the Device Policy Manager.

The Policy Engine shall transition to the *PE\_CBL\_Get\_Modes\_NAK* state when:

- The Device Policy Manager indicates that the response to the Discover Identity request is NAK or BUSY.

**8.3.3.9.1.7 8.3.3.10.4.2 PE\_CBL\_Send\_Modes state**

On entry to the *PE\_CBL\_Send\_Modes* state the Cable shall send the Structured VDM *Discover Modes* ACK Command response.

The Policy Engine shall transition to the *PE\_CBL\_Ready* state when:

- The Structured VDM *Discover Modes* ACK Command response has been sent.

**8.3.3.10.4.3 PE\_CBL\_Get\_Modes\_NAK**

On entry to the *PE\_CBL\_Get\_Modes\_NAK* state the Policy Engine shall send a Structured VDM *Discover Modes NAK* or *BUSY* Command response as indicated by the Device Policy Manager.

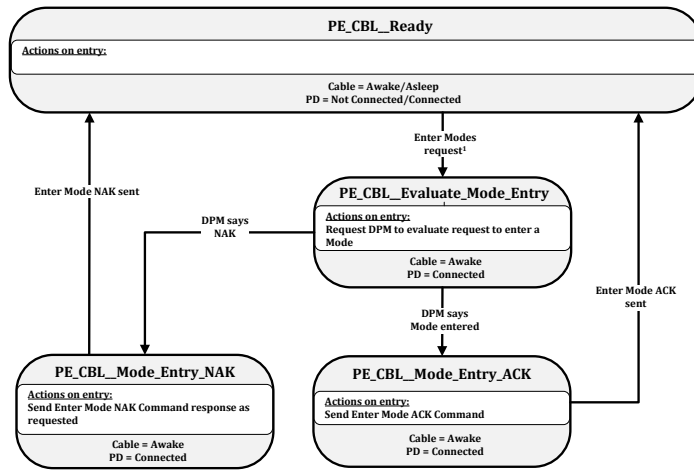
The Policy Engine shall transition to the *PE\_CBL\_Ready* state when:

- The Structured VDM *Discover Modes NAK* or *BUSY* Command response has been sent.

**8.3.3.10.5 Cable Plug Structured VDM Enter Mode State Diagram**

Figure 8-75 shows the state diagram for a Cable Plug in response to an *Enter Mode* Command.

**Figure 8-75 CablePlug Structured VDM Enter Mode State Diagram**



<sup>1</sup> The Cable is required to be in USB operation or USB Safe State at this point.

**8.3.3.9.1.108.3.3.10.5.1 PE\_CBL\_Evaluate\_Mode\_Entry state**

The Policy Engine transitions to the **PE\_CBL\_Evaluate\_Mode\_Entry** state from the **PE\_CBL\_Ready** state when:

- A Structured VDM **Enter Mode** Command request is received from the DFP.

On Entry to the **PE\_CBL\_Evaluate\_Mode\_Entry** state the Policy Engine shall request the Device Policy Manager to evaluate the **Enter Mode** Command request and enter the Mode indicated in the Command request if the request is acceptable.

The Policy Engine shall transition to the **PE\_CBL\_Mode\_Entry\_ACK** state when:

- The Device Policy Manager indicates that the Mode has been entered.

The Policy Engine shall transition to the **PE\_CBL\_Mode\_Entry\_NAK** state when:

- The Device Policy Manager indicates that the response to the Mode request is NAK.

**8.3.3.9.1.108.3.3.10.5.2 PE\_CBL\_Mode\_Entry\_ACK state**

On entry to the **PE\_CBL\_Mode\_Entry\_ACK** state the Policy Engine shall send a Structured VDM **Enter Mode** ACK Command response.

The Policy Engine shall transition to the **PE\_CBL\_Ready** state when:

- The Structured VDM **Enter Mode** ACK Command response has been sent.

**8.3.3.9.1.108.3.3.10.5.3 PE\_CBL\_Mode\_Entry\_NAK state**

On entry to the **PE\_CBL\_Mode\_Entry\_NAK** state the Policy Engine shall send a Structured VDM **Enter Mode** NAK Command response as indicated by the Device Policy Manager.

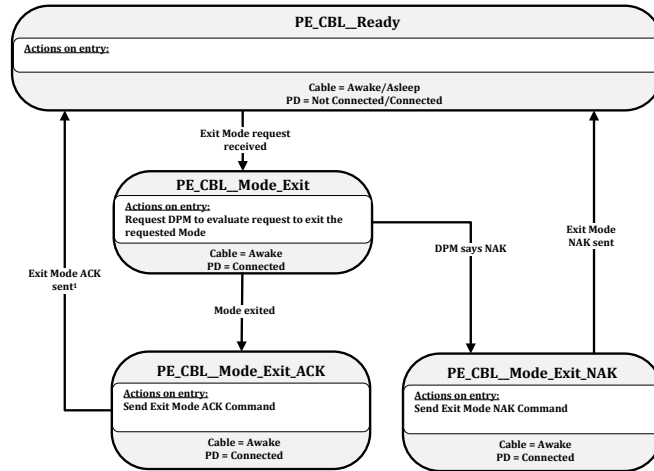
The Policy Engine shall transition to the **PE\_CBL\_Ready** state when:

- The Structured VDM **Enter Mode** NAK Command response has been sent.

### 8.3.3.10.6 Cable Plug Structured VDM Exit Mode State Diagram

Figure 8-76 shows the state diagram for a Cable Plug in response to an *Exit Mode Command*.

Figure 8-76 CablePlug Structured VDM Exit Mode State Diagram



<sup>1</sup> The Cable is required to be in USB operation or USB Safe State at this point.

#### 8.3.3.9.1.118.3.3.10.6.1 PE\_CBL\_Mode\_Exit state

The Policy Engine transitions to the *PE\_CBL\_Mode\_Exit* state from the *PE\_CBL\_Ready* state when:

- A Structured VDM *Exit Mode* Command request is received from the DFP.

On entry to the *PE\_CBL\_Mode\_Exit* state the Policy Engine shall request the Device Policy Manager to exit the Mode indicated in the Command.

The Policy Engine shall transition to the *PE\_CBL\_Mode\_Exit\_ACK* state when:

- The ~~indicated~~ Device Policy Manger indicates that the Mode has been exited.

The Policy Engine shall transition to the *PE\_CBL\_Mode\_Exit\_NAK* state when:

- The Device Policy Manager indicates that the Command response to the *Exit Mode Command* request is NAK.

#### 8.3.3.9.1.128.3.3.10.6.2 PE\_CBL\_Mode\_Exit\_ACK state

On entry to the *PE\_CBL\_Mode\_Exit\_ACK* state the Policy Engine shall send a Structured VDM *Exit Mode* ACK Command response.

The Policy Engine shall transition to the *PE\_CBL\_Ready* state when:

- The Structured VDM *Exit Mode* ACK Command response has been sent.



### 8.3.3.10.6.3 PE UFP VDM Mode Exit NAK state

On entry to the **PE\_CBL\_Mode\_Exit\_NAK** state the Policy Engine shall send a Structured VDM **Exit Mode NAK Command** response as indicated by the Device Policy Manager.

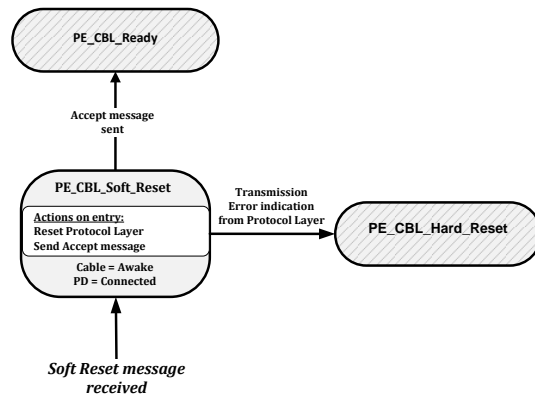
The Policy Engine shall transition to the **PE\_CBL\_Ready** state when:

- The Structured VDM **Exit Mode NAK Command** response has been sent.

### 8.3.3.9.2.18.3.3.10.7 Cable Plug Soft Reset State Diagram

Figure 8-77 shows the Cable Plug state diagram on reception of a **Soft\_Reset** Message.

Figure 8-77 Cable Plug Soft Reset State Diagram



### 8.3.3.9.2.18.3.3.10.7.1 PE\_CBL\_Soft\_Reset state

The **PE\_CBL\_Soft\_Reset** state shall be entered from any state when a Reset Message is received from the Protocol Layer.

On entry to the **PE\_CBL\_Soft\_Reset** state the Policy Engine shall reset the Protocol Layer in the Cable Plug and shall then request the Protocol Layer to send an **Accept** Message.

The Policy Engine shall transition to the **PE\_CBL\_Ready** state when:

- The **Accept** Message has been sent.

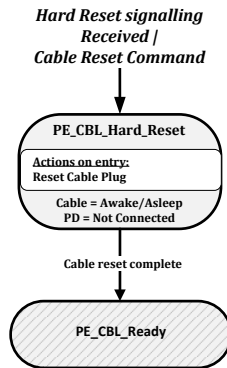
The Policy Engine shall transition to the **PE\_CBL\_Hard\_Reset** state when:

- The Protocol Layer indicates that a transmission error has occurred.

### 8.3.3.9.2.18.3.3.10.8 Cable Plug Hard Reset State Diagram

Figure 8-78 shows the Cable Plug state diagram for a Hard Reset or Cable Reset.

Figure 8-78 Cable Plug Hard Reset State Diagram



~~8.3.3.9.3~~ 8.3.3.10.8.1 PE\_CBL\_Hard\_Reset state

The **PE\_CBL\_Hard\_Reset** state shall be entered from any state when either **Hard Reset** Signaling or **Cable Reset** Signaling is detected.

On entry to the **PE\_CBL\_Hard\_Reset** state the Policy Engine shall reset the Cable Plug (equivalent to a power cycle).

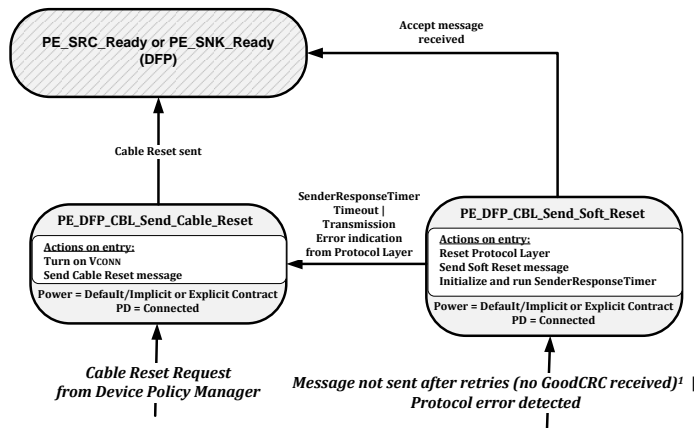
The Policy Engine shall transition to the **PE\_CBL\_Ready** state when:

- The Cable Plug reset is complete.

8.3.3.10.9 DFP Soft Reset or Cable Reset of a Cable Plug State Diagram

Figure 8-82 below shows the state diagram for the Policy Engine in a DFP when performing a Soft Reset or Cable Reset of a Cable Plug. The following sections describe operation in each of the states.

**Figure 8-79 DFP Soft Reset or Cable Reset of a Cable Plug State Diagram**



<sup>1</sup> Excludes Soft Reset itself.

**8.3.3.10.9.1 PE DFP CBL Send Soft Reset state**

The **PE DFP CBL Send Soft Reset** state shall be entered from any state when a Protocol Error is detected by the Protocol Layer (see Section 6.7.1) when a Message has not been sent after retries while communicating with a Cable Plug or whenever the Device Policy Manager directs a Soft Reset.

On entry to the **PE DFP CBL Send Soft Reset** state the Policy Engine shall request the Protocol Layer to perform a Soft Reset, then shall send a **Soft Reset** Message to the Cable Plug, and initialize and run the **SenderResponseTimer**.

The Policy Engine shall transition to either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state, depending on the DFP's Power Role, when:

- An **Accept** Message has been received.

The Policy Engine shall transition to the **PE DFP CBL Send Cable Reset** state when:

- A **SenderResponseTimer** timeout occurs
- Or the Protocol Layer indicates that a transmission error has occurred.

**8.3.3.10.9.2 PE DFP CBL Send Cable Reset state**

The **PE DFP CBL Send Cable Reset** state shall be entered from any state when the Device Policy Manager requests a Cable Reset.

On entry to the **PE DFP CBL Send Cable Reset** state the Policy Engine shall request the Protocol Layer to send **Cable Reset Signaling**.

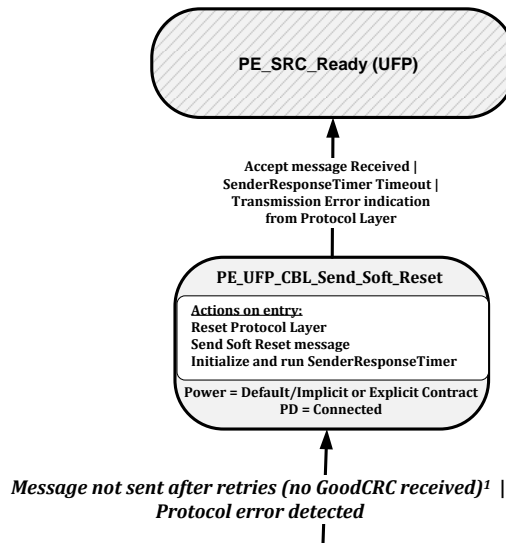
The Policy Engine shall transition to either the **PE\_SRC\_Ready** or **PE\_SNK\_Ready** state, depending on the DFP's Power Role, when:

- **Cable Reset** Signaling has been sent.

### 8.3.3.10.10 UFP Source Soft Reset of a Cable Plug State Diagram

Figure 8-82 below shows the state diagram for the Policy Engine in a UFP Source, prior to an Explicit Contract, when performing a Soft Reset of a Cable Plug. The following sections describe operation in each of the states.

Figure 8-80 UFP Source Soft Reset of a Cable Plug State Diagram



<sup>1</sup> Excludes Soft Reset itself.

#### 8.3.3.10.10.1 PE UFP CBL Send Soft Reset state

The **PE UFP CBL Send Soft Reset** state shall be entered from any state when a Protocol Error is detected by the Protocol Layer, when a Message has not been sent after retries while communicating with a Cable Plug or whenever the Device Policy Manager directs a Soft Reset.

Note that there are corner cases that are not shown in the defined state diagrams that could be handled without generating a Protocol Error.

On entry to the **PE UFP CBL Send Soft Reset** state the Policy Engine shall request the Protocol Layer to perform a Soft Reset, then shall send a **Soft\_Reset** Message to the Cable Plug, and initialize and run the **SenderResponseTimer**.

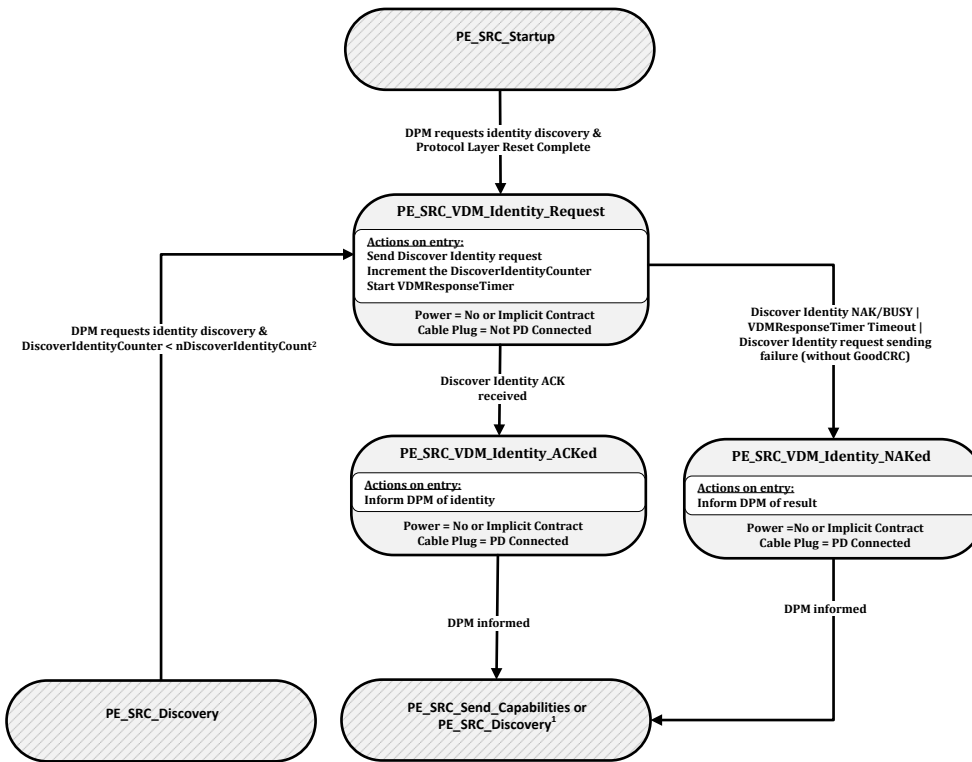
The Policy Engine shall transition to the **PE\_SRC\_Ready** state when:

- An **Accept** Message has been received
- Or a **SenderResponseTimer** timeout occurs
- Or the Protocol Layer indicates that a transmission error has occurred

### 8.3.3.10.11 Source Startup Structured VDM Discover Identity of a Cable Plug State Diagram

Figure 8-81 shows the state diagram for Source discovery of identity information from a Cable Plug during the startup sequence.

**Figure 8-81 Source Startup Structured VDM Discover Identity State Diagram**



1 If the Discover Identity message is being sent at startup then the Policy Engine will subsequently transition to the PE\_SRC\_Capabilities state as normal. Otherwise the Policy Engine will transition to the PE\_SRC\_Discovery state.  
 2 The SourceCapabilitiesTimer continues to run during the states defined in this diagram even though there has been an exit from the PE\_SRC\_Discovery state. This ensures that Source Capabilities are sent out at a regular rate.

**8.3.3.10.11.1 PE\_SRC VDM Identity Request state**

The Policy Engine shall transition to the **PE\_SRC VDM Identity Request** state from the **PE\_SRC\_Startup** state when:

- The Device Policy Manager requests the discovery of the identity of the Cable Plug.

Even though there has been a transition out of the **PE\_SRC\_Discovery** state the **SourceCapabilityTimer** shall continue to run during the states shown in Figure 8-81 and shall not be initialized on re-entry to **PE\_SRC\_Discovery**.

The Policy Engine shall transition to the **PE\_SRC VDM Identity Request** state from the **PE\_SRC\_Discovery** state when:

- The Device Policy Manager requests the discovery of the identity of the Cable Plug and
- The **DiscoverIdentityCounter** < **nDiscoverIdentityCount**.

On entry to the **PE\_SRC VDM Identity Request** state the Policy Engine shall send a Structured VDM **Discover Identity** Command request, shall increment the **DiscoverIdentityCounter** and shall start the **VDMResponseTimer**.

The Policy Engine shall transition to the **PE\_SRC VDM Identity ACKed** state when:

- A Structured VDM *Discover Identity ACK* Command response is received.

The Policy Engine shall transition to the *PE\_SRC\_VDM\_Identity\_NAKed* state when:

- A Structured VDM *Discover Identity NAK* or *BUSY* Command response is received or
- The *VDMResponseTimer* times out or
- The Structured VDM *Discover Identity* Command request Message sending fails (no *GoodCRC* Message received after retries).

#### 8.3.3.10.11.2 PE\_SRC\_VDM\_Identity\_ACKed state

On entry to the *PE\_SRC\_VDM\_Identity\_ACKed* state the Policy Engine shall inform the Device Policy Manager of the Identity information.

The Policy Engine shall transition back to either the *PE\_SRC\_Send\_Capabilities* or *PE\_SRC\_Discovery* state when:

- The Device Policy Manager has been informed.

#### 8.3.3.10.11.3 PE\_SRC\_VDM\_Identity\_NAKed state

On entry to the *PE\_SRC\_VDM\_Identity\_NAKed* state the Policy Engine shall inform the Device Policy Manager of the result (NAK, BUSY or timeout).

The Policy Engine shall transition back to either the *PE\_SRC\_Send\_Capabilities* or *PE\_SRC\_Discovery* state when:

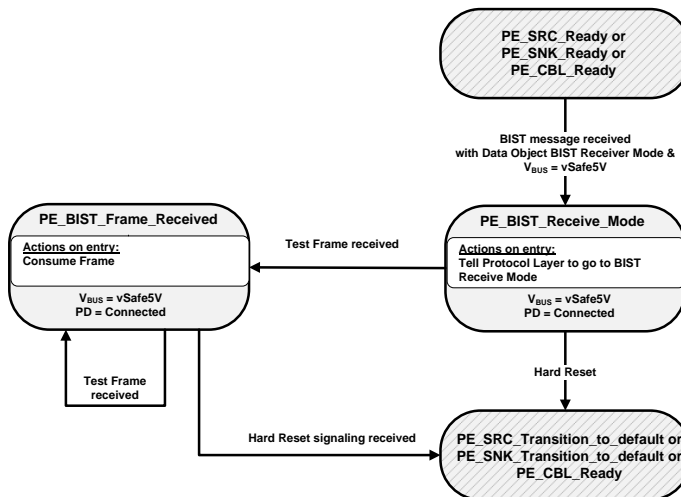
The Device Policy Manager has been informed.

### 8.3.3.10.3.3.11 BIST State diagrams

#### 8.3.3.10.18.3.3.11.1 BIST Receive Mode State Diagram

Figure 8-82 shows the state diagram required by a UUT, which can be either a Source or Sink or Cable Plug, when operating in BIST Receive Mode. Transitions to *PE\_BIST\_Receive\_Mode* state shall be from either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* or *PE\_CBL\_Ready* state. See Section 5.9.9 for which tests are applicable to BFSK and BMC.

Figure 8-82 BIST Receive Mode State Diagram



~~8.3.3.10.1.18~~.3.3.11.1.1 PE\_BIST\_Receive\_Mode state

The Source ~~Sink~~ or ~~Sink Cable Plug~~ shall enter the *PE\_BIST\_Receive\_Mode* state from either the *PE\_SRC\_Ready* ~~or~~, *PE\_SNK\_Ready* or *PE\_CBL\_Ready* state when:

- A *BIST* Message is received with a *BIST Receiver Mode BIST* Data Object and  $V_{BUS}$  is at *vSafe5V*.

On entry to the *PE\_BIST\_Receive\_Mode* state the Policy Engine shall tell the Protocol Layer to go to BIST Receive Mode.

The Policy Engine shall transition to the *PE\_BIST\_Frame\_Received* state when:

- A Test Frame is received.

The Policy Engine shall transition to either the *PE\_SRC\_Transition\_to\_default* state ~~or~~, *PE\_SNK\_Transition\_to\_default* state or *PE\_CBL\_Ready state* (as appropriate) when:

- *Hard Reset* Signaling is received.

~~8.3.3.10.1.28~~.3.3.11.1.2 PE\_BIST\_Frame\_Received state

On entry to the *PE\_BIST\_Frame\_Received* state the Policy Engine shall consume a BIST Transmit Test Frame if one has been received.

The Policy Engine shall transition back to the *PE\_BIST\_Frame\_Received* state when:

- The BIST Receive Test Frame has been received.

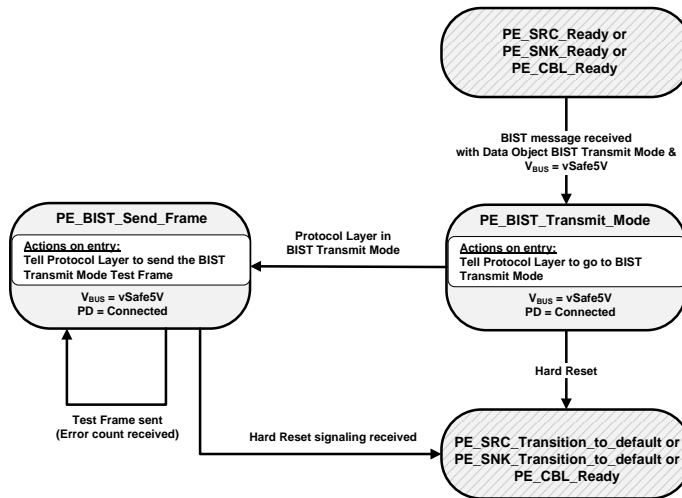
The Policy Engine shall transition to either the *PE\_SRC\_Transition\_to\_default* state ~~or~~, *PE\_SNK\_Transition\_to\_default* state or *PE\_CBL\_Ready state* (as appropriate) when:

- *Hard Reset* Signaling is received.

### 8.3.3.10.2.18.3.3.11.2 BIST Transmit Mode State Diagram

Figure 8-83 shows the state diagram required by a UUT that can be either a Source or Sink or Cable Plug, when operating in BIST Transmit Mode. Transitions to *PE\_BIST\_Transmit\_Mode* state shall be from either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* or *PE\_CBL\_Ready* state. See Section 5.9.9 for which tests are applicable to BFSK and BMC.

Figure 8-83 BIST Transmit Mode State Diagram



#### 8.3.3.10.2.18.3.3.11.2.1 PE\_BIST\_Transmit\_Mode state

The Source, Sink or Sink Cable Plug shall enter the *PE\_BIST\_Transmit\_Mode* state from either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* or *PE\_CBL\_Ready* state when:

- A *BIST* Message is received with a *BIST Transmit Mode BIST* Data Object and  $V_{BUS}$  is at *vSafe5V*.

On entry to the *PE\_BIST\_Transmit\_Mode* state the Policy Engine shall tell the Protocol Layer to go to BIST Transmit Mode.

The Policy Engine shall transition to the *PE\_BIST\_Send\_Frame* state when:

- The Protocol Layer is in BIST Transmit Mode.

The Policy Engine shall transition to either the *PE\_SRC\_Transition\_to\_default* state or *PE\_SNK\_Transition\_to\_default* state or *PE\_CBL\_Ready* state (as appropriate) when:

- Hard Reset* Signaling is received.

#### 8.3.3.10.2.18.3.3.11.2.2 PE\_BIST\_Send\_Frame state

On entry to the *PE\_BIST\_Send\_Frame* state the Policy Engine shall tell the Protocol Layer to send the next BIST Transmit Test Frame in the PRBS sequence.

The Policy Engine shall transition back to the *PE\_BIST\_Send\_Frame* state when:

- The BIST Transmit Test Frame has been sent (a *BIST* Message with a *BISTErrorCounter BIST* Data Object has been received).



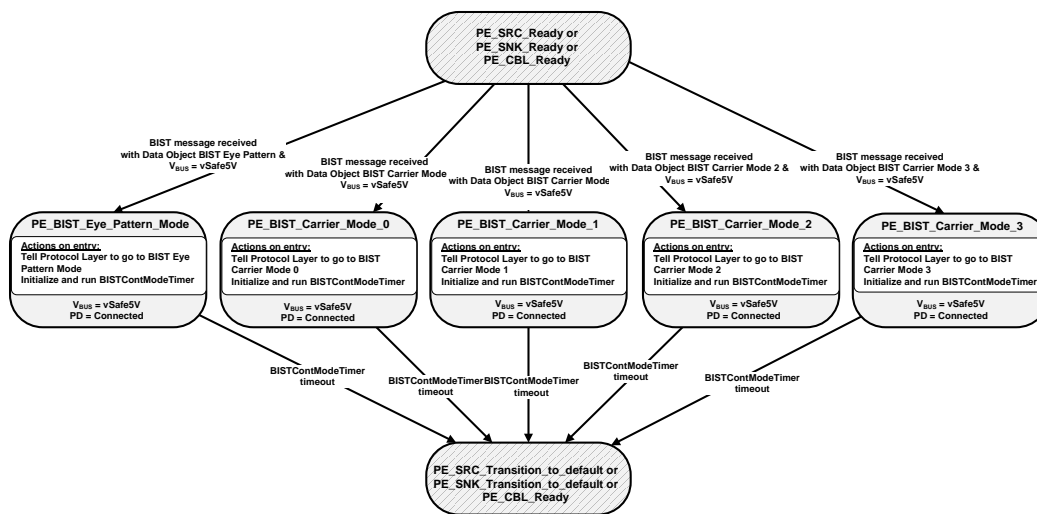
The Policy Engine shall transition to either the *PE\_SRC\_Transition\_to\_default* state or *PE\_SNK\_Transition\_to\_default* state or *PE\_CBL\_Ready\_state* (as appropriate) when:

- *Hard Reset* Signaling is received.

8.3.3.10.3.11.3 BIST Carrier Mode and Eye Pattern State Diagram

Figure 8-84 shows the state diagram required by a UUT, which can be either a Source or Sink or Cable Plug, when operating in *BIST Eye Pattern*, *BIST Carrier Mode 0*, *BIST Carrier Mode 1*, *BIST Carrier Mode 2* or *BIST Carrier Mode 3*. Transitions to any of the Test Mode states shall be from either the *PE\_SRC\_Ready* or *PE\_SNK\_Ready* or *PE\_CBL\_Ready* states. See Section 5.9.9 state for which tests are applicable to BFSK and BMC.

Figure 8-84 BIST Carrier Mode and Eye Pattern State Diagram



8.3.3.10.3.18.3.3.11.3.1 BIST Eye Pattern

The Source, Sink or Sink Cable Plug shall enter the *PE\_BIST\_Eye\_Pattern\_Mode* state from either the *PE\_SRC\_Ready*, *PE\_SNK\_Ready* or *PE\_CBL\_Ready* state when:

- A *BIST* Message is received with a *BIST Eye Pattern BIST* Data Object and
- $V_{BUS}$  is at *vSafe5V*.

On entry to the *PE\_BIST\_Eye\_Pattern\_Mode* state the Policy Engine shall tell the Protocol Layer to go to BIST Eye Pattern Test Mode and shall initialize and run the *BISTContModeTimer*.

The Policy Engine shall transition to either the *PE\_SRC\_Transition\_to\_default* state or *PE\_SNK\_Transition\_to\_default* state or *PE\_CBL\_Ready\_state* (as appropriate) when:

- ~~The *BISTContModeTimer* There is an operator reset of the UUT.~~
- times out.

8.3.3.10.3.28.3.3.11.3.2 BIST Carrier Mode 0

The Source, Sink or Sink Cable Plug shall enter the *PE\_BIST\_Carrier\_Mode\_0* state from either the *PE\_SRC\_Ready*, *PE\_SNK\_Ready* or *PE\_CBL\_Ready* state when:

Field Code Changed

Field Code Changed

- A *BIST* Message is received with a *BIST Carrier Mode 0 BIST* Data Object and
- $V_{BUS}$  is at *vSafe5V*.

On entry to the *PE\_BIST\_Carrier\_Mode\_0* state the Policy Engine shall tell the Protocol Layer to go to BIST Carrier ~~Parity~~ Mode 0 and shall initialize and run the *BISTContModeTimer*.

The Policy Engine shall transition to either the *PE\_SRC\_Transition\_to\_default* state ~~or~~, *PE\_SNK\_Transition\_to\_default* state or *PE\_CBL\_Ready\_state* (as appropriate) when:

- ~~The *BISTContModeTimer* There is an operator reset of the UUT.~~
- times out.

#### ~~8.3.3.10.3.38~~ 8.3.3.11.3.3 BIST Carrier Mode 1

The Source, *Sink* or *SinkCable Plug* shall enter the *PE\_BIST\_Carrier\_Mode\_1* state from either the *PE\_SRC\_Ready*, *PE\_SNK\_Ready* or *PE\_CBL\_Ready* state when:

Field Code Changed

- A *BIST* Message is received with a *BIST Carrier Mode 1 BIST* Data Object and
- $V_{BUS}$  is at *vSafe5V*.

On entry to the *PE\_BIST\_Carrier\_Mode\_1* state the Policy Engine shall tell the Protocol Layer to go to BIST Carrier ~~Parity~~ Mode 1 and shall initialize and run the *BISTContModeTimer*.

The Policy Engine shall transition to either the *PE\_SRC\_Transition\_to\_default* state ~~or~~, *PE\_SNK\_Transition\_to\_default* state or *PE\_CBL\_Ready\_state* (as appropriate) when:

- ~~The *BISTContModeTimer* There is an operator reset of the UUT.~~
- times out.

#### ~~8.3.3.10.3.48~~ 8.3.3.11.3.4 BIST Carrier Mode 2

The Source, *Sink* or *SinkCable Plug* shall enter the *PE\_BIST\_Carrier\_Mode\_2* state from either the *PE\_SRC\_Ready*, *PE\_SNK\_Ready* or *PE\_CBL\_Ready* state when:

Field Code Changed

- A *BIST* Message is received with a *BIST Carrier Mode 2 BIST* Data Object and
- $V_{BUS}$  is at *vSafe5V*.

On entry to the *PE\_BIST\_Carrier\_Mode\_2* state the Policy Engine shall tell the Protocol Layer to go to BIST Carrier ~~Parity~~ Mode 2 and shall initialize and run the *BISTContModeTimer*.

The Policy Engine shall transition to either the *PE\_SRC\_Transition\_to\_default* state ~~or~~, *PE\_SNK\_Transition\_to\_default* state or *PE\_CBL\_Ready\_state* (as appropriate) when:

- ~~The *BISTContModeTimer* There is an operator reset of the UUT.~~
- times out.

#### ~~8.3.3.10.3.58~~ 8.3.3.11.3.5 BIST Carrier Mode 3

The Source, *Sink* or *SinkCable Plug* shall enter the *PE\_BIST\_Carrier\_Mode\_3* state from either the *PE\_SRC\_Ready*, *PE\_SNK\_Ready* or *PE\_CBL\_Ready* state when:

Field Code Changed

- A *BIST* Message is received with a *BIST Carrier Mode 3 BIST* Data Object and
- $V_{BUS}$  is at *vSafe5V*.

On entry to the *PE\_BIST\_Carrier\_Mode\_3* state the Policy Engine shall tell the Protocol Layer to go to BIST Carrier Mode 3 and shall initialize and run the *BISTContModeTimer*.

The Policy Engine shall transition to either the *PE\_SRC\_Transition\_to\_default* state ~~or~~, *PE\_SNK\_Transition\_to\_default* state or *PE\_CBL\_Ready\_state* (as appropriate) when:

- ~~The *BISTContModeTimer* There is an operator reset of the UUT.~~

- times out.

### ~~8.3.3.11~~8.3.3.12 **Type-C Referenced States**

This section contains states cross-referenced from the [\[USBType-C 1.0\]](#) specification.

#### ~~8.3.3.11.1~~8.3.3.12.1 **ErrorRecovery state**

The **ErrorRecovery** state is used to electronically disconnect Port Partners using the Type-C connector. The **ErrorRecovery** state shall be entered when there are errors on Type-C Ports which cannot be recovered by Hard Reset. The **ErrorRecovery** state shall map to Type-C ErrorRecovery state operation as defined in the [\[USBType-C 1.0\]](#) specification, including any other state transitions mandated in cases where Type-C ErrorRecovery is not supported.

On entry to the **ErrorRecovery** state the Contract and PD Connection shall be ended.

On exit from the **ErrorRecovery** state a new Explicit Contract should be established once the Port Partners have re-connected over the CC wire.

8.3.3.128.3.3.13 Policy Engine States

Table 8-33 lists the states used by the various state machines.

Table 8-33 Policy Engine States

State name	Section
<b>Source Port</b>	
<i>PE_SRC_Startup</i>	8.3.3.2.1
<i>PE_SRC_Discovery</i>	8.3.3.2.2
<i>PE_SRC_Send_Capabilities</i>	8.3.3.2.3
<i>PE_SRC_Negotiate_Capability</i>	8.3.3.2.4
<i>PE_SRC_Transition_Supply</i>	8.3.3.2.5
<i>PE_SRC_Ready</i>	8.3.3.2.6
<i>PE_SRC_Disabled</i>	8.3.3.2.7
<i>PE_SRC_Capability_Response</i>	8.3.3.2.8
<i>PE_SRC_Hard_Reset</i>	8.3.3.2.9
<i>PE_SRC_Hard_Reset_Received</i>	⊘
<i>PE_SRC_Transition_to_default</i>	8.3.3.2.11
<i>PE_SRC_Give_Source_Cap</i>	8.3.3.2.12
<i>PE_SRC_Get_Sink_Cap</i>	8.3.3.2.13
<i>PE_SRC_Wait_New_Capabilities</i>	8.3.3.2.14
<b>Sink Port</b>	
<i>PE_SNK_Startup</i>	8.3.3.3.1
<i>PE_SNK_Discovery</i>	8.3.3.3.2
<i>PE_SNK_Wait_for_Capabilities</i>	8.3.3.3.3
<i>PE_SNK_Evaluate_Capability</i>	8.3.3.3.4
<i>PE_SNK_Select_Capability</i>	8.3.3.3.5
<i>PE_SNK_Transition_Sink</i>	8.3.3.3.6
<i>PE_SNK_Ready</i>	8.3.3.3.7
<i>PE_SNK_Hard_Reset</i>	8.3.3.3.8
<i>PE_SNK_Transition_to_default</i>	8.3.3.3.9
<i>PE_SNK_Give_Sink_Cap</i>	8.3.3.3.10
<i>PE_SNK_Get_Source_Cap</i>	8.3.3.3.11
<b>Source Port Soft Reset</b>	
<i>PE_SRC_Send_Soft_Reset</i>	8.3.3.4.1.1
<i>PE_SRC_Soft_Reset</i>	8.3.3.4.1.2
<b>Sink Port Soft Reset</b>	
<i>PE_SNK_Send_Soft_Reset</i>	8.3.3.4.2.1
<i>PE_SNK_Soft_Reset</i>	8.3.3.4.2.2
<b>Source Port Ping</b>	
<i>PE_SRC_Ping</i>	8.3.3.5.1

State name	Section
<b>Type-A/B Dual-Role (initially Source Port) Ping</b>	
<i>PE_PRS_SRC_SNK_Ping</i>	8.3.3.6.1.1.1
<b>Type-A/B Dual-Role (initially Sink Port) Ping</b>	
<i>PE_PRS_SNK_SRC_Ping</i>	8.3.3.6.1.2.1
<b>Type-A/B Hard Reset of P/C in Sink Role</b>	
<i>PE_PC_SNK_Hard_Reset</i>	8.3.3.6.1.3.1
<i>PE_PC_SNK_Swap_Recovery</i>	8.3.3.6.1.3.2
<b>Type-A/B Hard Reset of C/P in Source Role</b>	
<i>PE_CP_SRC_Hard_Reset</i>	8.3.3.6.1.4.1
<i>PE_CP_SRC_Transition_to_off</i>	8.3.3.6.1.4.2
<b>Type-A/B C/P Dead Battery/Power Loss</b>	
<i>PE_DB_CP_Check_for_VBUS</i>	8.3.3.6.1.5.1
<i>PE_DB_CP_Power_VBUS_DB</i>	8.3.3.6.1.5.2
<i>PE_DB_CP_Wait_For_Bit_Stream</i>	8.3.3.6.1.5.3
<i>PE_DB_CP_Power_VBUS_5V</i>	8.3.3.6.1.5.4
<i>PE_DB_CP_Wait_Bit_Stream_Stop</i>	8.3.3.6.1.5.5
<i>PE_DB_CP_Unpower_VBUS</i>	8.3.3.6.1.5.6
<i>PE_DB_CP_PS_Discharge</i>	8.3.3.6.1.5.7
<b>Type-A/B P/C Dead Battery/Power Loss</b>	
<i>PE_DB_PC_Unpowered</i>	8.3.3.6.1.6.1
<i>PE_DB_PC_Check_Power</i>	8.3.3.6.1.6.2
<i>PE_DB_PC_Send_Bit_Stream</i>	8.3.3.6.1.6.3
<i>PE_DB_PC_Wait_to_Detect</i>	8.3.3.6.1.6.4
<i>PE_DB_PC_Wait_to_Start</i>	8.3.3.6.1.6.5
<b>Type-C DFP to UFP Data Role Swap</b>	
<i>PE_DRS_DFP_UFP_Evaluate_DR_Swap</i>	8.3.3.6.2.1.2
<i>PE_DRS_DFP_UFP_Accept_DR_Swap</i>	8.3.3.6.2.1.3
<i>PE_DRS_DFP_UFP_Change_to_UFP</i>	8.3.3.6.2.1.4
<i>PE_DRS_DFP_UFP_Send_DR_Swap</i>	8.3.3.6.2.1.5
<i>PE_DRS_DFP_UFP_Reject_DR_Swap</i>	8.3.3.6.2.1.6
<b>Type-C UFP to DFP Data Role Swap</b>	
<i>PE_DRS_UFP_DFP_Evaluate_DR_Swap</i>	8.3.3.6.2.2.2
<i>PE_DRS_UFP_DFP_Accept_DR_Swap</i>	8.3.3.6.2.2.3
<i>PE_DRS_UFP_DFP_Change_to_DFP</i>	8.3.3.6.2.2.4
<i>PE_DRS_UFP_DFP_Send_DR_Swap</i>	8.3.3.6.2.2.5
<i>PE_DRS_UFP_DFP_Reject_DR_Swap</i>	8.3.3.6.2.2.6
<b>Source to Sink Power Role Swap</b>	
<i>PE_PRS_SRC_SNK_Evaluate_PR_Swap</i>	8.3.3.6.3.1.2
<i>PE_PRS_SRC_SNK_Accept_PR_Swap</i>	8.3.3.6.3.1.3
<i>PE_PRS_SRC_SNK_Transition_to_off</i>	8.3.3.6.3.1.4

State name	Section
<i>PE_PRS_SRC_SNK_Assert_Rd</i>	8.3.3.6.3.1.5
<i>PE_PRS_SRC_SNK_Wait_Source_offon</i>	8.3.3.6.3.1.6
<i>PE_PRS_SRC_SNK_Send_PR_Swap</i>	0
<i>PE_PRS_SRC_SNK_Reject_PR_Swap</i>	8.3.3.6.3.1.8
<b>Sink to Source Power Role Swap</b>	
<i>PE_PRS_SNK_SRC_Evaluate_PR_Swap</i>	8.3.3.6.3.2.2
<i>PE_PRS_SNK_SRC_Accept_PR_Swap</i>	8.3.3.6.3.2.3
<i>PE_PRS_SNK_SRC_Transition_to_off</i>	8.3.3.6.3.2.4
<i>PE_PRS_SNK_SRC_Assert_Rp</i>	
<i>PE_PRS_SNK_SRC_Source_on</i>	8.3.3.6.3.2.5
<i>PE_PRS_SNK_SRC_Send_PR_Swap</i>	8.3.3.6.3.2.7
<i>PE_PRS_SNK_SRC_Reject_PR_Swap</i>	8.3.3.6.3.2.8
<b>Dual-Role Source Port Get Source Capabilities</b>	
<i>PE_DR_SRC_Get_Source_Cap</i>	8.3.3.6.3.3.1
<b>Dual-Role Source Port Give Sink Capabilities</b>	
<i>PE_DR_SRC_Give_Sink_Cap</i>	8.3.3.6.3.4.1
<b>Dual-Role Sink Port Get Sink Capabilities</b>	
<i>PE_DR_SNK_Get_Sink_Cap</i>	8.3.3.6.3.5.1
<b>Dual-Role Sink Port Give Source Capabilities</b>	
<i>PE_DR_SNK_Give_Source_Cap</i>	8.3.3.6.3.6.1
<b>Type-C <del>DFF</del>-VCONN Swap</b>	
<i>PE_VCS_DFF_Send_Swap</i>	8.3.3.7.1.1
<i>PE_VCS_DFF_Wait_for_UFP_VCONN</i>	<del>8.3.3.7.1.2</del>
<i>PE_VCS_DFF_Turn_Off_VCONN</i>	<del>8.3.3.7.1.3</del>
<i>PE_VCS_DFF_Turn_On_VCONN</i>	<del>8.3.3.7.1.4</del>
<i>PE_VCS_DFF_Send_PS_Rdy</i>	<del>8.3.3.7.1.5</del>
<b>Type-C <del>UFP</del>-VCONN Swap</b>	
<i>PE_VCS_UFP_Evaluate_Swap</i>	8.3.3.7.1.2
<i>PE_VCS_UFP_Accept_Swap</i>	8.3.3.7.1.3
<i>PE_VCS_UFP_Reject_Swap</i>	8.3.3.7.1.4
<i>PE_VCS_UFP_Wait_For_DFF_VCONN</i>	8.3.3.7.1.5
<i>PE_VCS_UFP_Turn_Off_VCONN</i>	8.3.3.7.1.6
<i>PE_VCS_UFP_Turn_On_VCONN</i>	8.3.3.7.1.7
<i>PE_VCS_UFP_Send_Ps_Rdy</i>	8.3.3.7.1.8
<b>UFP <u>Structured</u> VDM</b>	
<b><u>UFP Structured VDM Discovery Identity</u></b>	
<i>PE_UFP_VDM_Get_Identity</i>	8.3.3.8.1.1
<i>PE_UFP_VDM_Send_Identity</i>	8.3.3.8.1.2
<i>PE_UFP_VDM_Get_Identity_NAK</i>	8.3.3.8.1.3

State name	Section
<b><u>UFP Structured VDM Discovery SVIDs</u></b>	
<i>PE_UFP_VDM_Get_SVIDs</i>	8.3.3.8.2.1
<i>PE_UFP_VDM_Send_SVIDs</i>	8.3.3.8.2.2
<i>PE_UFP_VDM_Get_SVIDs_NAK</i>	8.3.3.8.2.3
<b><u>UFP Structured VDM Discovery Modes</u></b>	
<i>PE_UFP_VDM_Get_Modes</i>	8.3.3.8.3.1
<i>PE_UFP_VDM_Send_Modes</i>	8.3.3.8.3.2
<i>PE_UFP_VDM_Get_Modes_NAK</i>	8.3.3.8.3.3
<b><u>UFP Structured VDM Enter Mode</u></b>	
<i>PE_UFP_VDM_Evaluate_Mode_Entry</i>	8.3.3.8.4.1
<i>PE_UFP_VDM_Mode_Entry_ACK</i>	8.3.3.8.4.2
<i>PE_UFP_VDM_Mode_Entry_NAK</i>	8.3.3.8.4.3
<b><u>UFP Structured VDM Exit Mode</u></b>	
<i>PE_UFP_VDM_Mode_Exit</i>	8.3.3.8.5.1
<i>PE_UFP_VDM_Mode_Exit_ACK</i>	8.3.3.8.5.2
<i>PE_UFP_VDM_Mode_Exit_NAK</i>	8.3.3.8.5.3
<b><u>UFP Structured VDM Attention</u></b>	
<i>PE_UFP_VDM_Attention_Request</i>	8.3.3.8.6.1
<b><u>DFP Structured VDM</u></b>	
<b><u>DFP to UFP Structured VDM Discover Identity</u></b>	
<i>PE_DFP_UFP_VDM_Identity_Request</i>	8.3.3.9.1.1
<i>PE_DFP_UFP_VDM_Identity_ACKed</i>	8.3.3.9.1.2
<i>PE_DFP_UFP_VDM_Identity_NAKed</i>	8.3.3.9.1.3
<b><u>DFP to Cable Plug Structured VDM Discover Identity</u></b>	
<i>PE_DFP_CBL_VDM_Identity_Request</i>	8.3.3.9.2.1
<i>PE_DFP_CBL_VDM_Identity_ACKed</i>	8.3.3.9.2.2
<i>PE_DFP_CBL_VDM_Identity_NAKed</i>	1.1.1.1.1
<b><u>DFP Structured VDM Discover SVIDs</u></b>	
<i>PE_DFP_VDM_SVIDs_Request</i>	8.3.3.9.3.1
<i>PE_DFP_VDM_SVIDs_ACKed</i>	8.3.3.9.3.2
<i>PE_DFP_VDM_SVIDs_NAKed</i>	8.3.3.9.3.3
<b><u>DFP Structured VDM Discover Modes</u></b>	
<i>PE_DFP_VDM_Modes_Request</i>	8.3.3.9.4.1
<i>PE_DFP_VDM_Modes_ACKed</i>	8.3.3.9.4.2
<i>PE_DFP_VDM_Modes_NAKed</i>	8.3.3.9.4.3
<b><u>DFP Structured VDM Mode Entry</u></b>	
<i>PE_DFP_VDM_Mode_Entry_Request</i>	8.3.3.9.5.1
<i>PE_DFP_VDM_Mode_Entry_ACKed</i>	8.3.3.9.5.2
<i>PE_DFP_VDM_Mode_Entry_NAKed</i>	☐
<b><u>DFP Structured VDM Mode Exit</u></b>	
<i>PE_DFP_VDM_Mode_Exit_Request</i>	8.3.3.9.6.1

State name	Section
<i>PE_DFP_VDM_Mode_Exit_ACKed</i>	8.3.3.9.6.2
<b>Source Startup VDM Discover Identity</b>	
<i>PE_SRC_VDM_Identity_Request</i>	<del>8.3.3.8.8.1</del>
<i>PE_SRC_VDM_Identity_ACKed</i>	<del>8.3.3.8.8.2</del>
<i>PE_SRC_VDM_Identity_NAKed</i>	<del>8.3.3.8.8.3</del>
<b>DFP Structured VDM Attention</b>	
<i>PE_DFP_VDM_Attention_Request</i>	8.3.3.9.7.1
<b>USB to USB Cable Plug Related</b>	
<b>Cable Ready</b>	
<i>PE_CBL_Ready</i>	8.3.3.10.1.1
<b>Discover Identity</b>	
<i>PE_CBL_Get_Identity</i>	8.3.3.10.2.1
<i>PE_CBL_Send_Identity</i>	8.3.3.10.2.2
<i>PE_CBL_Get_Identity_NAK</i>	8.3.3.10.2.3
<b>Discover SVIDs</b>	
<i>PE_CBL_Get_SVIDs</i>	8.3.3.10.3.1
<i>PE_CBL_Send_SVIDs</i>	8.3.3.10.3.2
<i>PE_CBL_Get_SVIDs_NAK</i>	8.3.3.10.3.3
<b>Discover Modes</b>	
<i>PE_CBL_Get_Modes</i>	8.3.3.10.4.1
<i>PE_CBL_Send_Modes</i>	8.3.3.10.4.2
<i>PE_CBL_Get_Modes_NAK</i>	8.3.3.10.4.3
<b>Mode Entry</b>	
<i>PE_CBL_Evaluate_Mode_Entry</i>	8.3.3.10.5.1
<i>PE_CBL_Mode_Entry_ACK</i>	8.3.3.10.5.2
<i>PE_CBL_Mode_Entry_NAK</i>	8.3.3.10.5.3
<b>Mode Exit</b>	
<i>PE_CBL_Mode_Exit</i>	8.3.3.10.6.1
<i>PE_CBL_Mode_Exit_ACK</i>	8.3.3.10.6.2
<i>PE_CBL_Mode_Exit_NAK</i>	
<b>Cable Soft Reset</b>	
<i>PE_CBL_Soft_Reset</i>	8.3.3.10.7.1
<b>Cable Hard Reset</b>	
<i>PE_CBL_Hard_Reset</i>	8.3.3.10.8.1
<b>DFP Soft Reset or Cable Reset</b>	
<i>PE_DFP_CBL_Send_Soft_Reset</i>	8.3.3.10.9.1
<i>PE_DFP_CBL_Send_Cable_Reset</i>	8.3.3.10.9.2
<b>UFP Source Soft Reset</b>	
<i>PE_UFP_CBL_Send_Soft_Reset</i>	8.3.3.10.10



State name	Section
<b>Source Startup Structured VDM Discover Identity</b>	
<i>PE_SRC_VDM_Identity_Request</i>	8.3.3.10.11.1
<i>PE_SRC_VDM_Identity_ACKed</i>	8.3.3.10.11.2
<i>PE_SRC_VDM_Identity_NAKed</i>	8.3.3.10.11.3
<b>BIST Receive Mode</b>	
<i>PE_BIST_Receive_Mode</i>	8.3.3.11.1.1
<i>PE_BIST_Frame_Received</i>	8.3.3.11.1.2
<b>BIST Transmit Mode</b>	
<i>PE_BIST_Transmit_Mode</i>	8.3.3.11.2.1
<i>PE_BIST_Send_Frame</i>	8.3.3.11.2.2
<b>BIST Carrier Mode and Eye Pattern</b>	
<i>PE_BIST_Eye_Pattern_Mode</i>	8.3.3.11.3.1
<i>PE_BIST_Carrier_Mode_0</i>	8.3.3.11.3.2
<i>PE_BIST_Carrier_Mode_1</i>	8.3.3.11.3.3
<i>PE_BIST_Carrier_Mode_2</i>	8.3.3.11.3.4
<i>PE_BIST_Carrier_Mode_3</i>	8.3.3.11.3.5
<b>Type-C referenced states</b>	
<i>ErrorRecovery</i>	8.3.3.12.1

## 9. System Policy

### 9.1 Overview

This chapter describes the requirements of the USB Power Delivery Specification's System Policy and Status mechanisms for devices with data connections (e.g. D+/D- and or SSTx+/- and SSRx+/-). The Policies themselves are not described here; these are left to the implementers of the relevant products and systems to define. The aim of these mechanisms is to support a System USB Power Delivery Policy Manager (SPM) which:

- Provides feedback of system power status to the end user as and when appropriate
- Provides co-ordination of system power resources based on a system wide policy when enabled
- Exposes relevant policy settings to the end user
- Allows a device to report its capabilities in relation to the power it draws

All PD Capable USB (PDUSB) Devices shall report themselves as self-powered devices (over USB) when plugged into a PD capable Port even if they are entirely powered from  $V_{BUS}$ . However, there are some differences between PD and [\[USB 2.0\]](#) / [\[USB 3.1\]](#); for example, the presence of  $V_{BUS}$  alone does not mean that the device (Consumer) moves from the USB Attached state to the USB Powered state. Similarly the removal of  $V_{BUS}$  alone does not move the device (Consumer) from any of the USB states to the Attached state. See Section 9.1.2 for details.

PDUSB Devices shall follow the PD requirements when it comes to suspend (see Section 6.4.1.2.3.2), configured, and operational power. The PD requirements when the device is configured or operational are defined in this section (see Table 9-4). Note that the power requirements reported in the PD Consumer Port descriptor of the device shall override the power draw reported in the *bMaxPower* field in the configuration descriptor. A PDUSB Device shall report zero in the *bMaxPower* field after successfully negotiating a mutually agreeable Contract and shall disconnect and re-enumerate when it switches operation back to operating in standard [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USBType-C 1.0\]](#) or [\[BC 1.2\]](#). When operating in [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USBType-C 1.0\]](#) or [\[BC 1.2\]](#) mode it shall report its power draw via the *bMaxPower* field.

As shown in Figure 9-1, each Provider and Consumer will have their own Local Policies which operate between directly connected ports. An example of a typical PD system is shown in Figure 9-1. This example consists of a Provider, Consumer/Providers and Consumers connected together in a tree topology. Between directly connected devices there is both a flow of Power and also Communication consisting of both Status and Control information.

Figure 9-1 Example PD Topology

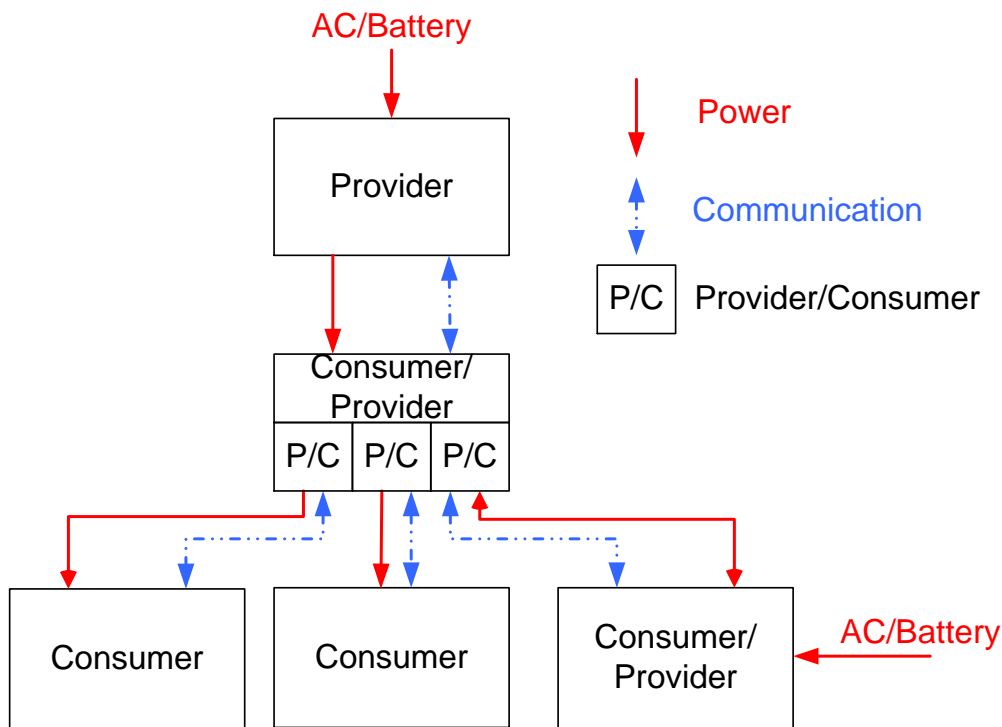
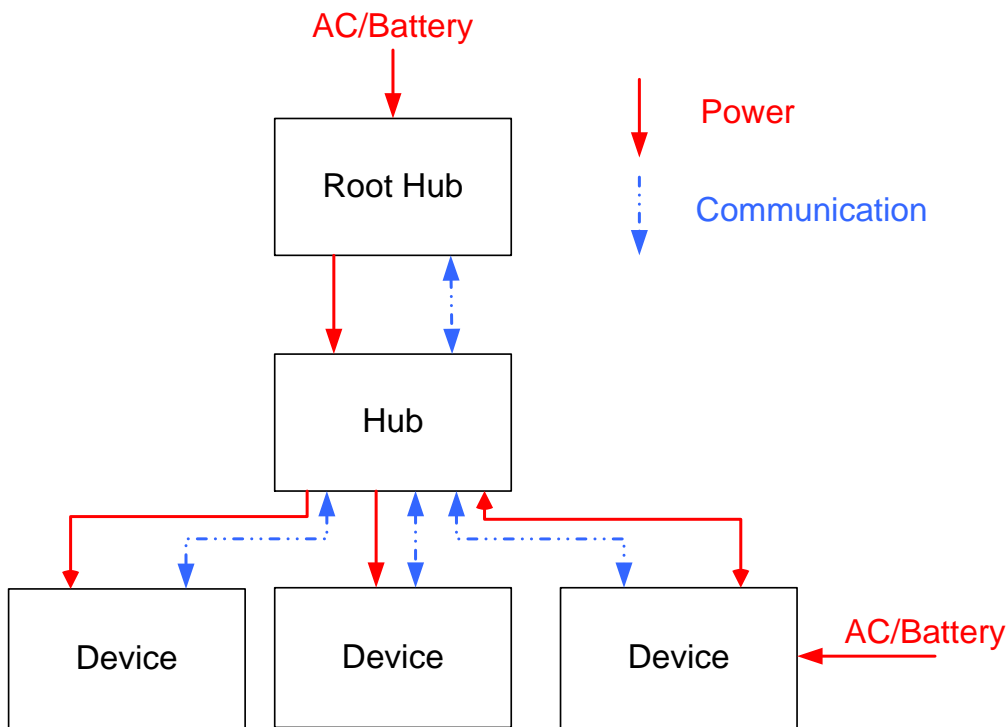


Figure 9-2 shows how this same topology can be mapped to USB. In a USB based system, policy is managed by the host and communication of system level policy information is via standard USB data line communication. This is a separate mechanism to the USB Power Delivery  $V_{BUS}$  protocol which is used to manage Local Policy. When USB data line communication is used, status information and control requests are passed directly between the System Policy Manager (SPM) on the host and the Provider or Consumer.

Status information comes from a Provider or Consumer to the SPM so it can better manage the resources on the host and provide feedback to the end user. Control information comes from the SPM to the Provider or Consumer allowing the SPM to manage resources at a system level and expose relevant settings to the end user.

Real systems will be a mixture of devices which in terms of power management support may have implemented PD, [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USBTtype-C 1.0\]](#) or [\[BC 1.2\]](#) or they may even just be non-compliant Power Sucking Devices. The level of communication of system status to the SPM will therefore not necessarily be comprehensive. The aim of the System Policies and Status mechanisms is to provide a mechanism whereby each connected entity in the system provides as much information as possible on the status of itself and, in the case of hubs, its downstream ports. This will enable the PM to build up as comprehensive a picture of the system as possible.

Figure 9-2 Mapping of PD Topology to USB



Information which will be communicated to the SPM is as follows:

- Versions of Type-C Current, PD and BC supported
- Capabilities of each Provider/Consumer including a per Port summary for Providers
- Current operational state of each Port e.g. Standard, Type-C Current, BC, PD and negotiated power level
- Status of AC or Battery Power for each PDUSB Device in the system

The SPM can negotiate with Providers or Consumers in the system in order to request a different Local Policy, or to request the amount of power to be delivered by the Provider to the Consumer. Any change in Local Policy could trigger a renegotiation of the Contract, using USB Power Delivery protocols, between a directly connected Provider and Consumer. A change in how much power is to be delivered will, for example, cause a renegotiation.

### 9.1.1 PDUSB Device and Hub Requirements

All PDUSB Devices shall return all relevant descriptors mentioned in this chapter. PDUSB Hubs shall also support a PD notification capability as defined in this chapter. In addition, a PDUSB Hubs shall support the capability to return the PD specific capabilities of a device that is attached to any of its downstream ports. PDUSB Hubs shall support Local Policy Mode and shall report the appropriate status information in response to any status requests from the System Policy Manager when operating in Local Policy Mode. In addition PDUSB Hubs may support Hybrid and/or Intrusive Policy Modes (see Section 9.4.5). Note that in a USB Hub, the notifications are sent over the USB Hub notification endpoint.

### 9.1.2 Mapping to USB Device States

As mentioned in Section 9.1 a PDUSB Device reports itself as a self-powered device. However, the device shall determine whether or not it is in the USB Attached or USB Powered states as described in Figure 9-3, Figure 9-4 and Figure 9-5. All other USB states of the PDUSB Device shall be as described in Chapter 9 of [\[USB 2.0\]](#) and [\[USB 3.1\]](#).

Figure 9-3 shows how a PDUSB Device determines when to transition from the USB Attached to the USB Powered state. Figure 9-3 [also shows Dead Battery operation for a Type-A to Type-B connection \(see Section 4.1.1\). Type-C Dead Battery operation does not require special handling since the default state at attach or after a Hard Reset is that the USB Device is a Sink.](#)

Figure 9-3 USB Attached to USB Powered State Transition

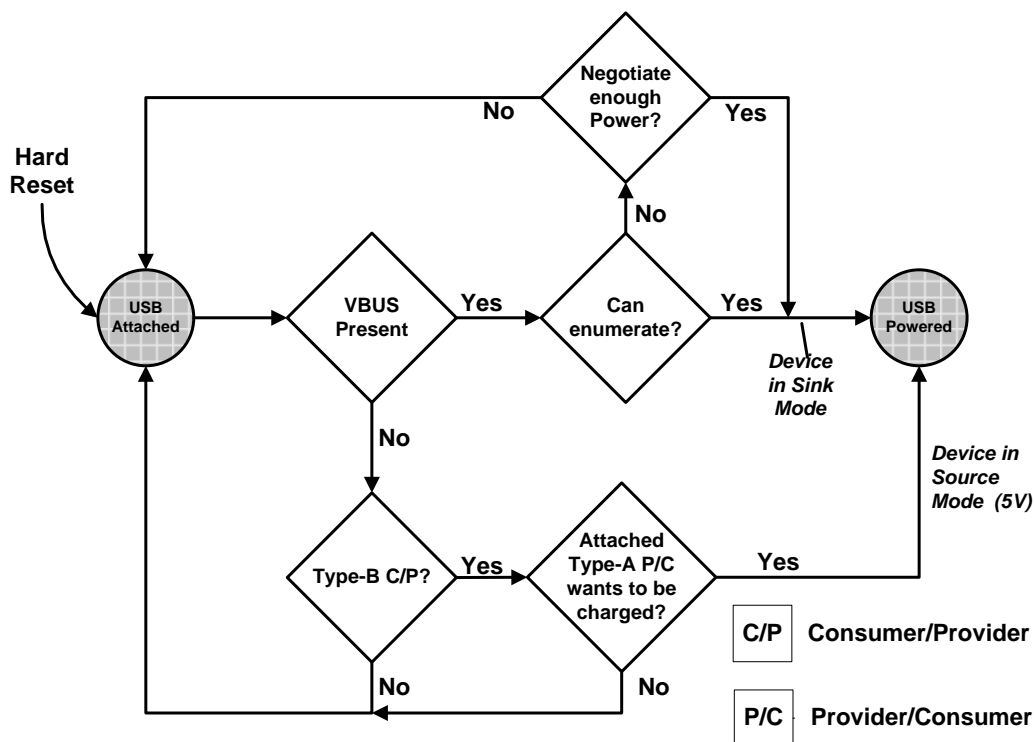


Figure 9-4 shows how a PDUSB Device determines when to transition from the USB Powered state to the USB Attached state when the device is a Consumer. A PDUSB Device determines that it is performing a Power Role Swap as described in Sections 8.3.2.7.1.1 and 8.3.2.7.1.2. See Section 7.1.6 for additional information on device behavior during Hard Resets.

Figure 9-4 Any USB State to USB Attached State Transition (When operating as a Consumer)

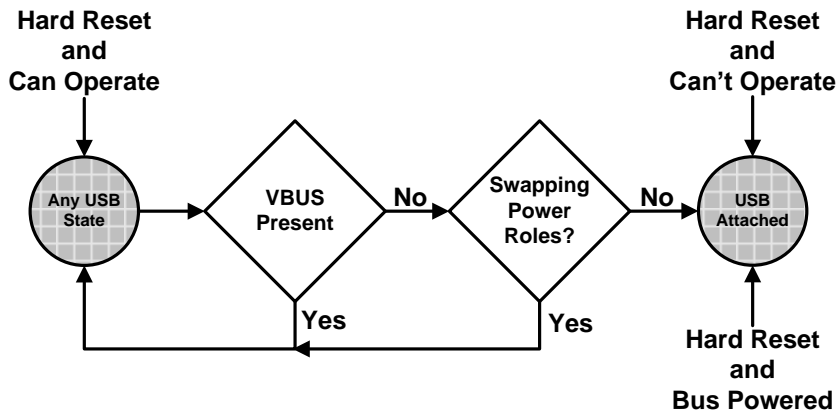


Figure 9-5 shows how a PDUSB Device determines when to transition from the USB Powered state to the USB Attached state when the device is a Provider.

Figure 9-5 Any USB State to USB Attached State Transition (When operating as a Provider)

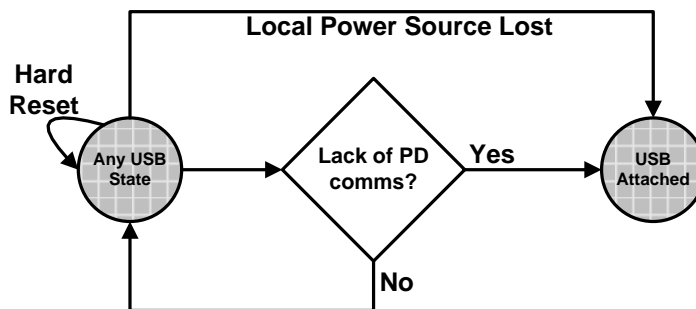
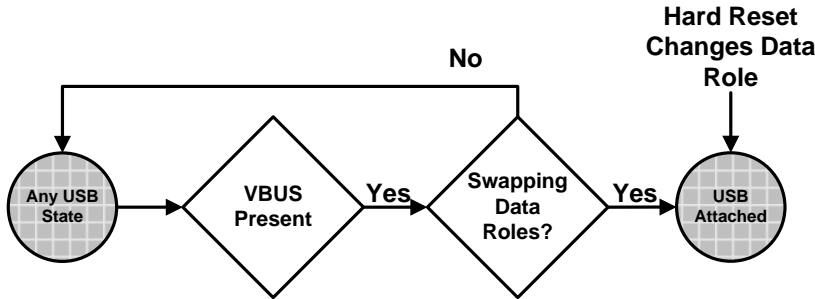


Figure 9-6 shows how a PDUSB Device using the Type-C connector determines when to transition from the USB Powered state to the USB Attached state after a Data Role Swap has been performed i.e. it has just changed from operation as a PDUSB Host to operation as a PDUSB Device. The Data Role Swap is described in Section 0. A Hard Reset will also return a Sink acting as a PDUSB Host to PDUSB Device operation as described in Section 0. See Section 7.1.6 for additional information on device behavior during Hard Resets.

**Figure 9-6 Any USB State to USB Attached State Transition (After a Type-C Data Role Swap)**

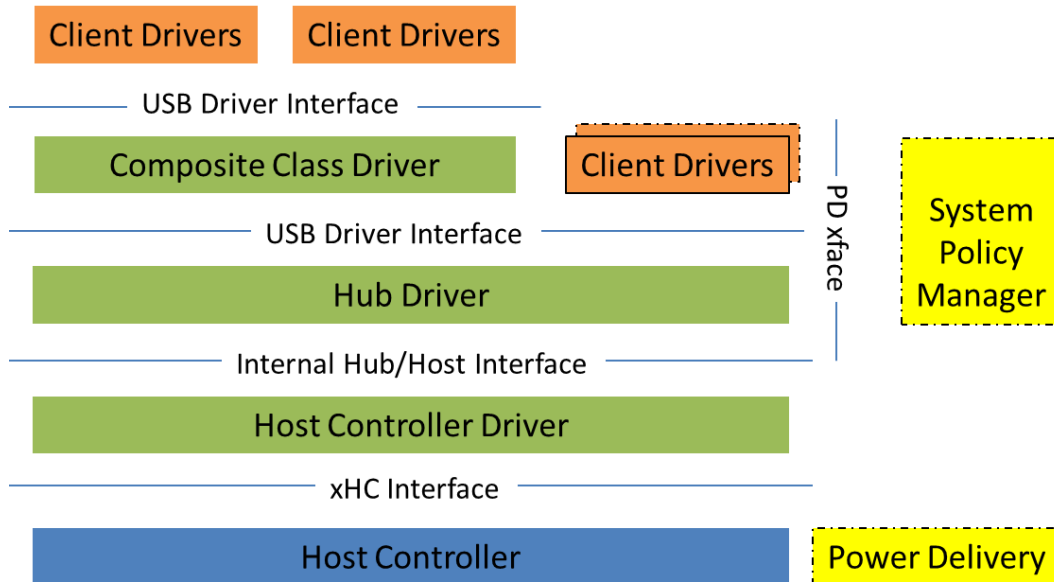


### 9.1.3 PD Software Stack

Figure 9-7 gives an example of the software stack on a PD aware OS. In this stack we are using the example of a system with an xHCI based controller. The USB Power Delivery hardware may or may not be a part of the xHC. The software that controls the USB Power Delivery hardware is called the System Policy Manager (SPM) in this specification.

It is expected that the SPM will expose an interface that can be used by an updated hub driver and updated client drivers that wish to interact with PDUSB Hubs and PDUSB Peripherals respectively.

**Figure 9-7 Software stack on a PD aware OS**



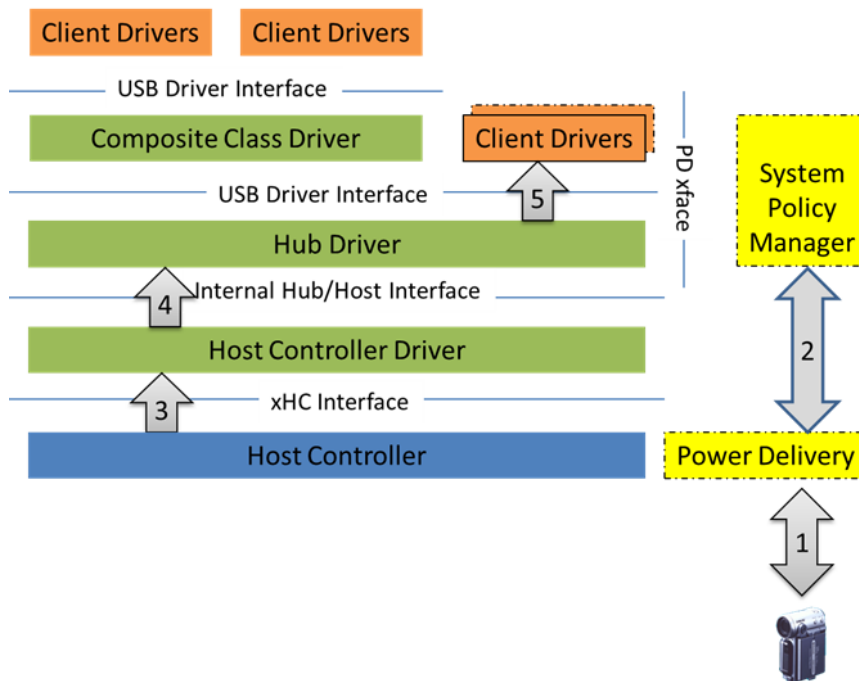
### 9.1.4 PDUSB Device Enumeration

As described earlier, a PDUSB Device acts as a self-powered device with some caveats with respect to how it transitions from the USB Attached state to USB Powered state. Figure 9-8 gives a high level overview of the enumeration steps involved due to this change. A PDUSB Device will first (Step1) interact with the Power Delivery hardware and the Local Policy manager to determine whether or not it can get sufficient power to enumerate/operate. Note: PD is likely to have established a Contract prior to enumeration. The SPM will be notified (Step 2) of the result of this negotiation between the Power Delivery hardware and the PDUSB Device. It may request changes to the Local Policy manager if it deems it necessary. After successfully negotiating a mutually agreeable Contract the device will signal a connect to the xHC. The standard USB enumeration process (Steps 3, 4 and 5) is then followed to load the appropriate driver for the function(s) that the PDUSB Device exposes.

It is the responsibility of the SPM to ensure that the total amount of power supplied to a PDUSB Device is sufficient to run as many of the functions exposed by the device as software needs to operate at the same time.

Note that the interfaces between the SPM and USB Power Delivery blocks in Figure 9-7 are outside scope of this specification.

Figure 9-8 Enumeration of a PDUSB Device



If a PDUSB Device cannot perform its intended function with the amount of power that it can get from the Port it is connected to then the host system should display a Message (on a PD aware OS) about the failure to provide sufficient power to the device. In addition the device shall follow the requirements listed in Section 8.2.5.2.1.



## 9.2 PD Class Specific Descriptors

A PDUSB Device shall return all relevant descriptors mentioned in this section.

The device shall return its capability descriptors as part of the device's Binary Object Store (BOS) descriptor set. Table 9-1 lists the type of PD device capabilities.

Table 9-1 USB Power Delivery Type Codes

Capability Code	Value	Description
<i>POWER_DELIVERY_CAPABILITY</i>	06H	Defines the various PD Capabilities of this device
<i>BATTERY_INFO_CAPABILITY</i>	07H	Provides information on each battery supported by the device
<i>PD_CONSUMER_PORT_CAPABILITY</i>	08H	The Consumer characteristics of a Port on the device
<i>PD_PROVIDER_PORT_CAPABILITY</i>	09H	The provider characteristics of a Port on the device

### 9.2.1 USB Power Delivery Capability Descriptor

Table 9-2 USB Power Delivery Capability Descriptor

Offset	Field	Size	Value	Description												
0	<i>bLength</i>	1	Number	Size of descriptor												
1	<i>bDescriptorType</i>	1	Constant	DEVICE CAPABILITY Descriptor type												
2	<i>bDevCapabilityType</i>	1	Constant	Capability type: <i>POWER_DELIVERY_CAPABILITY</i>												
3	<i>bReserved</i>	1	Reserved	Shall be set to zero.												
4	<i>bmAttributes</i>	4	Bitmap	<p>Bitmap encoding of supported device level features. A value of one in a bit location indicates a feature is supported; a value of zero indicates it is not supported. Encodings are:</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Reserved. Shall be set to zero.</td> </tr> <tr> <td>1</td> <td>Battery Charging. This bit shall be set to one to indicate this device supports the Battery Charging Specification as per the value reported in the <i>bcdBCVersion</i> field.</td> </tr> <tr> <td>2</td> <td>USB Power Delivery. This bit shall be set to one to indicate this device supports the USB Power Delivery Specification as per the value reported in the <i>bcdPDVersion</i> field.</td> </tr> <tr> <td>3</td> <td>Provider. This bit shall be set to one to indicate this device is capable of providing power. This field is only valid if Bit 2 is set to one.</td> </tr> <tr> <td>4</td> <td>Consumer. This bit shall be set to one</td> </tr> </tbody> </table>	Bit	Description	0	Reserved. Shall be set to zero.	1	Battery Charging. This bit shall be set to one to indicate this device supports the Battery Charging Specification as per the value reported in the <i>bcdBCVersion</i> field.	2	USB Power Delivery. This bit shall be set to one to indicate this device supports the USB Power Delivery Specification as per the value reported in the <i>bcdPDVersion</i> field.	3	Provider. This bit shall be set to one to indicate this device is capable of providing power. This field is only valid if Bit 2 is set to one.	4	Consumer. This bit shall be set to one
Bit	Description															
0	Reserved. Shall be set to zero.															
1	Battery Charging. This bit shall be set to one to indicate this device supports the Battery Charging Specification as per the value reported in the <i>bcdBCVersion</i> field.															
2	USB Power Delivery. This bit shall be set to one to indicate this device supports the USB Power Delivery Specification as per the value reported in the <i>bcdPDVersion</i> field.															
3	Provider. This bit shall be set to one to indicate this device is capable of providing power. This field is only valid if Bit 2 is set to one.															
4	Consumer. This bit shall be set to one															

Offset	Field	Size	Value	Description														
				<p>to indicate that this device is a consumer of power. This field is only valid if Bit 2 is set to one.</p> <p>5 This bit shall be set to 1 to indicate that this device supports the feature <i>CHARGING_POLICY</i>. Note that supporting the <i>CHARGING_POLICY</i> feature does not require a BC or PD mechanism to be implemented.</p> <p>6 USB Type-C Current. This bit shall be set to one to indicate this device supports power capabilities defined in the USB Type-C Specification as per the value reported in the bcdUSBTypeCVersion field</p> <p>7 Reserved. Shall be set to zero.</p> <p>15:8 bmPowerSource. At least one of the following bits 8, 9 and 14 shall be set to indicate which power sources are supported.</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>8</td> <td>AC Supply</td> </tr> <tr> <td>9</td> <td>Battery</td> </tr> <tr> <td>10</td> <td>Other</td> </tr> <tr> <td>13:11</td> <td>NumBatteries. This field shall only be valid when the Battery field is set to one and shall be used to report the number of batteries in the device.</td> </tr> <tr> <td>14</td> <td>Uses V<sub>BUS</sub></td> </tr> <tr> <td>15</td> <td>Reserved and shall be set to zero.</td> </tr> </tbody> </table> <p>31:16 Reserved and shall be set to zero.</p>	Bit	Description	8	AC Supply	9	Battery	10	Other	13:11	NumBatteries. This field shall only be valid when the Battery field is set to one and shall be used to report the number of batteries in the device.	14	Uses V <sub>BUS</sub>	15	Reserved and shall be set to zero.
Bit	Description																	
8	AC Supply																	
9	Battery																	
10	Other																	
13:11	NumBatteries. This field shall only be valid when the Battery field is set to one and shall be used to report the number of batteries in the device.																	
14	Uses V <sub>BUS</sub>																	
15	Reserved and shall be set to zero.																	
8	bmProviderPorts	2	Bitmap	<p>The bit corresponding to the Port shall be set to one to indicate that this device is capable of providing power on that Port. (Either BC 1.2, USB Type-C Current or PD)</p> <p>Bit zero refers to the UFP of the device. Bits one through fifteen are only valid for hubs and refers to the downstream ports of the hub.</p>														

Offset	Field	Size	Value	Description
10	<i>bmConsumerPorts</i>	2	Bitmap	The bit corresponding to the Port shall be set to one to indicate that this device is capable of consuming power on that Port. (Either BC 1.2, USB Type-C Current or PD).  Bit zero refers to the UFP of the device. Bits one through fifteen are only valid for hubs and refers to the downstream ports of the hub.
12	<i>bcdBCVersion</i>	2	BCD	Battery Charging Specification Release Number in Binary-Coded Decimal (e.g., V1.20 is 120H). This field shall only be valid if the device indicates that it supports BC in the <i>bmAttributes</i> field.
14	<i>bcdPDVersion</i>	2	BCD	USB Power Delivery Specification Release Number in Binary-Coded Decimal. This field shall only be valid if the device indicates that it supports PD in the <i>bmAttributes</i> field.
16	<i>bcdUSBTypeCVersion</i>	2	BCD	USB Type-C Specification Release Number in Binary-Coded Decimal. This field shall only be valid if the device indicates that it supports USB Type-C in the <i>bmAttributes</i> field.

### 9.2.2 Battery Info Capability Descriptor

A PDUSB Device shall support this capability descriptor if it reported that one of its power sources was a Battery in the *bmPowerSource* field in its Power Deliver Capability Descriptor. It shall return one Battery Info Descriptor per battery it supports.

Table 9-3 Battery Info Capability Descriptor

Offset	Field	Size	Value	Description
0	<i>bLength</i>	1	Number	Size of descriptor
1	<i>bDescriptorType</i>	1	Constant	DEVICE CAPABILITY Descriptor type
2	<i>bDevCapabilityType</i>	1	Constant	Capability type: <b>BATTERY_INFO_CAPABILITY</b>
3	<i>iBattery</i>	1	Index	Index of string descriptor shall contain the user friendly name for this battery.
4	<i>iSerial</i>	1	Index	Index of string descriptor shall contain the Serial Number String for this battery.
5	<i>iManufacturer</i>	1	Index	Index of string descriptor shall contain the name of the Manufacturer for this battery.
6	<i>bBatteryId</i>	1	Number	Value shall be used to uniquely identify this battery in status Messages.
7	<i>bReserved</i>	1	Number	Reserved and shall be set to zero.
8	<i>dwChargedThreshold</i>	4	mWh	Shall contain the Battery Charge value above which this battery is considered to be fully charged but not necessarily "topped off."
12	<i>dwWeakThreshold</i>	4	mWh	Shall contain the minimum charge level of this battery such that above this threshold, a device can be assured of being able to power up successfully (see Battery Charging 1.2).

Offset	Field	Size	Value	Description
16	<i>dwBatteryDesignCapacity</i>	4	mWh	Shall contain the design capacity of the battery.
20	<i>dwBatteryLastFullchargeCapacity</i>	4	mWh	Shall contain the maximum capacity of the battery when fully charged.

### 9.2.3 PD Consumer Port Capability Descriptor

A PDUSB Device shall support this capability descriptor if it reported that one, or more, of its ports are a Consumer Port, as described in the *bmConsumerPorts* field in its Power Deliver Capability Descriptor. It shall return one PD Consumer Port Capability descriptor per Port that is a Consumer. The order of the descriptors shall be Port order number (low to high).

A PDUSB Peripheral shall have at most one Consumer or Consumer/Provider Port. A PDUSB Hub may have a maximum of sixteen Consumer capable ports (Consumer, Consumer/Provider or Provider/Consumer).

Table 9-4 PD Consumer Port Descriptor

Offset	Field	Size	Value	Description										
0	<i>bLength</i>	1	Number	Size of descriptor										
1	<i>bDescriptorType</i>	1	Constant	DEVICE CAPABILITY Descriptor type										
2	<i>bDevCapabilityType</i>	1	Constant	Capability type: <i>PD_CONSUMER_PORT_CAPABILITY</i>										
3	<i>bReserved</i>	1	Number	Reserved and shall be set to zero.										
4	<i>bmCapabilities</i>	2	Bitmap	Capability: This field shall indicate the specification the Consumer Port will operate under. <table border="1" data-bbox="662 730 1029 932"> <thead> <tr> <th>Bit</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Battery Charging (BC)</td> </tr> <tr> <td>1</td> <td>USB Power Delivery (PD)</td> </tr> <tr> <td>2</td> <td>USB Type-C Current</td> </tr> <tr> <td>15:3</td> <td>Reserved and shall be set to zero.</td> </tr> </tbody> </table>	Bit	Description	0	Battery Charging (BC)	1	USB Power Delivery (PD)	2	USB Type-C Current	15:3	Reserved and shall be set to zero.
Bit	Description													
0	Battery Charging (BC)													
1	USB Power Delivery (PD)													
2	USB Type-C Current													
15:3	Reserved and shall be set to zero.													
6	<i>wMinVoltage</i>	2	Number	Shall contain the minimum voltage in 50mV units that this Consumer is capable of operating at.										
8	<i>wMaxVoltage</i>	2	Number	Shall contain the maximum voltage in 50mV units that this Consumer is capable of operating at.										
10	<i>wReserved</i>	2	Number	Reserved and shall be set to zero.										
12	<i>dwMaxOperatingPower</i>	4	Number	Shall contain the maximum power in 10mW units this Consumer can draw when it is in a steady state operating mode.										
16	<i>dwMaxPeakPower</i>	4	Number	Shall contain the maximum power in 10mW units this Consumer can draw for a short duration of time ( <i>dwMaxPeakPowerTime</i> ) before it falls back into a steady state.										
20	<i>dwMaxPeakPowerTime</i>	4	Number	Shall contain the time in 100ms units that this Consumer can draw peak current. A device shall set this field to 0xFFFF if this value is unknown.										

#### 9.2.4 PD Provider Port Capability Descriptor

A PDUSB Device shall support this capability descriptor if it reported that one or more of its ports is a Provider Port, as described in the *bmProviderPorts* field in its Power Deliver Capability Descriptor. It shall return one PD Provider Port Capability descriptor per Port that is a Provider. The order of the descriptors will be Port order number (low to high).

A PDUSB Peripheral shall have at most one Provider Port. A PDUSB Hub can have a maximum of sixteen Provider ports.

Table 9-5 PD Provider Port Descriptor

Offset	Field	Size	Value	Description										
0	<i>bLength</i>	1	Number	Size of descriptor										
1	<i>bDescriptorType</i>	1	Constant	DEVICE CAPABILITY Descriptor type										
2	<i>bDevCapabilityType</i>	1	Constant	Capability type: <i>PD_PROVIDER_PORT_CAPABILITY</i>										
3	<i>bReserved</i>	1	Number	Reserved and shall be set to zero.										
4	<i>bmCapabilities</i>	2	Bitmap	<p>This field shall indicate the specification the Provider Port will operation under.</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Battery Charging (BC)</td> </tr> <tr> <td>1</td> <td>USB Power Delivery (PD)</td> </tr> <tr> <td>2</td> <td>USB Type-C Current</td> </tr> <tr> <td>15:3</td> <td>Reserved. Shall be set to zero.</td> </tr> </tbody> </table>	Bit	Description	0	Battery Charging (BC)	1	USB Power Delivery (PD)	2	USB Type-C Current	15:3	Reserved. Shall be set to zero.
Bit	Description													
0	Battery Charging (BC)													
1	USB Power Delivery (PD)													
2	USB Type-C Current													
15:3	Reserved. Shall be set to zero.													
6	<i>bNumOfPDObjects</i>	1	Number	Shall indicate the number of Power Data Objects.										
7	<i>bReserved</i>	1	Number	Reserved and shall be set to zero.										
8	<i>wPowerDataObject1</i>	4	Bitmap	Shall contain the first Power Data Object supported by this Provider Port. See Section 6.4.1 for details of the Power Data Objects.										
...	...	...	...	...										
N+4	<i>wPowerDataObjectN</i>	4	Bitmap	Shall contain the 2 <sup>nd</sup> and subsequent Power Data Objects supported by this Provider Port. See Section 6.4.1 for details of the Power Data Objects.										

### 9.3 PD Class Specific Requests and Events

A PDUSB Hub that is compliant to this specification shall support the PD specific requests and events detailed in this section irrespective of whether the PDUSB Hub is a Power Provider, a Power Consumer, or both.

A PDUSB Device that is compliant to this specification shall support the battery related requests if it has a battery.

The events shall be sent to the system software in response to changes on the PDUSB Hub ports that are not a direct result of a request from system software. These notifications shall be sent over the Interrupt endpoint of a PDUSB Hub.

Note that a PDUSB Hub shall only send these notifications, or respond to PD requests (except for GetBatteryStatus); if it has been made aware that the SPM is active on the host as detailed in Section 9.4.4.

#### 9.3.1 Class-specific Requests

The PD class defines requests to which PDUSB Devices shall respond as outlined in Table 9-6. All valid requests in Table 9-6 shall be implemented by PDUSB Devices.

Table 9-6 PD Class Requests

Request	bmRequestType	bRequest	wValue	wIndex	wLength	Data
ClearPortPDFeature <sup>1</sup>	00100011B	CLEAR_FEATURE	Feature Selector	Port Number	Two	Clear Change Mask
GetBatteryStatus <sup>2</sup>	1000000B	<i>GET_BATTERY_STATUS</i>	Zero	Battery ID	Eight	Battery Status
GetPartnerPDO <sup>1</sup>	10100011B	<i>GET_PARTNER_PDO</i>	Zero or One	Port Number	Variable	Power Data Objects
GetPortPDStatus <sup>1</sup>	10100011B	GET_STATUS	One	Port Number	Eight	PD Status
SetPDFeature <sup>2</sup>	0000000B	SET_FEATURE	Feature Selector	Feature Specific	Zero	None
SetPortPDFeature <sup>1</sup>	00100011B	SET_FEATURE	Feature Selector	Port Number	Zero	None
SetPortPDOS <sup>1</sup>	00100011B	<i>SET_PDO</i>	Zero	Port Number	Variable	Provider Power Data Objects
GetVDM <sup>1</sup>	10100011B	<i>GET_VDM</i>	Zero	Port Number	Variable	VDM
SendVDM <sup>1</sup>	00100011B	<i>SEND_VDM</i>	SOP/SOP'/SOP''	Port Number	Variable	VDM

<sup>1</sup>Requests valid for PDUSB Hubs only.

<sup>2</sup>Requests valid for PDUSB Hubs and PDUSB Peripherals.

Table 9-7 gives the bRequest values for commands that are not listed in the hub/device framework chapters of [\[USB 2.0\]](#), [\[USB 3.1\]](#).

Table 9-7 PD Class Request Codes

bRequest	Value
<i>GET_PARTNER_PDO</i>	20
<i>GET_BATTERY_STATUS</i>	21
<i>SET_PDO</i>	22

<b>bRequest</b>	<b>Value</b>
<i>GET_VDM</i>	23
<i>SEND_VDM</i>	24

Table 9-8 gives the valid feature selectors for the PD class. Refer to Sections 9.4.1, 9.4.5 and 9.4.6 for a description of the features.

**Table 9-8 PD Class Feature Selectors**

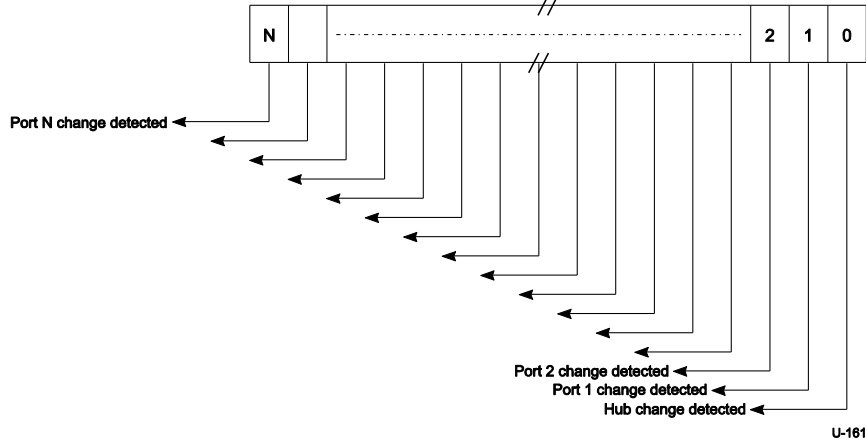
<b>Feature Selector</b>	<b>Recipient</b>	<b>Value</b>
<i>BATTERY_WAKE_MASK</i>	Device	40
<i>OS_IS_PD_AWARE</i>	Device	41
<i>POLICY_MODE</i>	Device	42
<i>PR_SWAP</i>	Port	43
<i>GOTO_MIN</i>	Port	44
<i>RETURN_POWER</i>	Port	45
<i>ACCEPT_PD_REQUEST</i>	Port	46
<i>REJECT_PD_REQUEST</i>	Port	47
<i>PORT_PD_RESET</i>	Port	48
<i>C_PORT_PD_CHANGE</i>	Port	49
<i>CABLE_PD_RESET</i>	Port	50
<i>CHARGING_POLICY</i>	Device	54

### 9.3.2 PDUSB Hub Event Reporting

PDUSB Hubs shall report events back to the system via the “Port N change detected” bit in the PDUSB Hub Status Change bitmap returned via the PDUSB Hub notification endpoint. ‘N’ refers to the Port number of the Port on which a PD change has occurred.



Figure 9-9 Hub Status Change



In order to indicate that a PD change occurred, a PDUSB Hub compliant to the PD class shall set Bit 15 *C\_PORT\_PD\_CHANGE* in the Port Change field when queried using the standard Get Port Status request.

A PDUSB Hub shall send these notifications for all downstream ports irrespective of whether it is a Consumer or Provider on that Port. For example, if a PDUSB Peripheral is a Provider on one of the downstream ports and it starts a negotiation to change the amount of power it will deliver – the PDUSB Hub sends a Port Change notification to the SPM. However, if the PDUSB Hub loses all power (e.g. the power Provider is unplugged) then it is sufficient if the PDUSB Hub disconnects from the downstream Port it is attached to without sending any notifications.

## 9.4 PDUSB Hub and PDUSB Peripheral Device Requests

### 9.4.1 ClearPortPDFeature (PDUSB Hub)

This request resets a value reported in the PDUSB Hub status. This request is only valid for hubs.

bmRequestType	bRequest	WValue	wIndex	wLength	Data
00100011B	CLEAR_FEATURE	Feature Selector	Port Number	Two	Clear Change Mask

The Port number shall be a valid Port number for that PDUSB Hub, greater than zero. The Port field is located in bits 7..0 of the *wIndex* field.

Refer to Table 9-8 for the feature selector definitions that shall apply to the Port as a recipient. If the feature selector is associated with a status change, the Clear Change Mask shall define the status changes that are being acknowledged by the SPM. Only when all the status changes in PD Status Change field (see Table 9-11) are acknowledged shall the hub clear the **C\_PORT\_PD\_CHANGE** bit in the Port's Port Change field. The Clear Change Mask is defined in Table 9-9.

Table 9-9 Clear Change Mask

Bit	Change Acknowledged when bit is set
0	Reserved
1	External supply.
2	Port operation mode
3	Supported Provider Capabilities
4	Negotiated power level
5	New power direction
6	PD Reset Complete
7	Insertion Detect
8	PD Capable Cable
9	Request Rejected
10	Reserved
11	Hybrid Mode Request
12	Power Role Swap Completed
13	VDM Received
14	Reserved
15	Error

It shall be a Request Error if *wValue* is not a feature selector listed in Table 9-8, or if *wIndex* specifies a Port that does not exist, or if *wLength* is not as specified above.

If the PDUSB Hub is not configured, the PDUSB Hub's response to this request is undefined.

### 9.4.2 GetBatteryStatus (PDU Hub/Peripheral Device)

This request returns the current status of the battery in a PDU Hub/Peripheral.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
1000000B	GET_BATTERY_STATUS	Zero	Battery ID	Eight	Battery Status

The PDU Hub/Peripheral shall return the Battery Status of the Battery identified by the value of *wIndex* field.

Every PDU Device that has a battery shall return its Battery Status when queried with this request. For Providers or Consumers with multiple batteries, the status of each battery shall be reported per battery.

Table 9-10 Battery Status Structure

Offset	Field	Size	Value	Description																				
0	<i>bBatteryAttributes</i>	1	Number	<p>Shall indicate whether a battery is installed and whether this is charging or discharging.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>There is no battery</td> </tr> <tr> <td>1</td> <td>The battery is charging</td> </tr> <tr> <td>2</td> <td>The battery is discharging</td> </tr> <tr> <td>3</td> <td>The battery is neither discharging nor charging</td> </tr> <tr> <td>255-4</td> <td>Reserved and shall not be used</td> </tr> </tbody> </table>	Value	Description	0	There is no battery	1	The battery is charging	2	The battery is discharging	3	The battery is neither discharging nor charging	255-4	Reserved and shall not be used								
Value	Description																							
0	There is no battery																							
1	The battery is charging																							
2	The battery is discharging																							
3	The battery is neither discharging nor charging																							
255-4	Reserved and shall not be used																							
1	<i>bBatterySOC</i>	1	Number	Shall indicate the Battery State of Charge given as percentage value from Battery Remaining Capacity.																				
2	<i>bBatteryStatus</i>	1	Number	<p>If a battery is present shall indicate the present status of the battery.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No error</td> </tr> <tr> <td>1</td> <td>Battery required and not present</td> </tr> <tr> <td>2</td> <td>Battery non-chargeable/wrong chemistry</td> </tr> <tr> <td>3</td> <td>Over-temp shutdown</td> </tr> <tr> <td>4</td> <td>Over-voltage shutdown</td> </tr> <tr> <td>5</td> <td>Over-current shutdown</td> </tr> <tr> <td>6</td> <td>Fatigued battery</td> </tr> <tr> <td>7</td> <td>Unspecified error</td> </tr> <tr> <td>255-8</td> <td>Reserved and shall not be used</td> </tr> </tbody> </table>	Value	Meaning	0	No error	1	Battery required and not present	2	Battery non-chargeable/wrong chemistry	3	Over-temp shutdown	4	Over-voltage shutdown	5	Over-current shutdown	6	Fatigued battery	7	Unspecified error	255-8	Reserved and shall not be used
Value	Meaning																							
0	No error																							
1	Battery required and not present																							
2	Battery non-chargeable/wrong chemistry																							
3	Over-temp shutdown																							
4	Over-voltage shutdown																							
5	Over-current shutdown																							
6	Fatigued battery																							
7	Unspecified error																							
255-8	Reserved and shall not be used																							

Offset	Field	Size	Value	Description										
3	<i>bRemoteWakeCapStatus</i>	1	Bitmap	<p>If the device supports remote wake, then the device shall support Battery Remote wake events. The default value for the Remote wake events shall be turned off (set to zero) and can be enable/disabled by the host as required. If set to one the device shall generate a wake event when a change of status occurs. See Section 9.4.5 for more details.</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Battery present event</td> </tr> <tr> <td>1</td> <td>Charging flow</td> </tr> <tr> <td>2</td> <td>Battery error</td> </tr> <tr> <td>7:3</td> <td>Reserved and shall be set to zero</td> </tr> </tbody> </table>	Bit	Description	0	Battery present event	1	Charging flow	2	Battery error	7:3	Reserved and shall be set to zero
Bit	Description													
0	Battery present event													
1	Charging flow													
2	Battery error													
7:3	Reserved and shall be set to zero													
4	<i>wRemainingOperatingTime</i>	2	Number	<p>Shall contain the operating time (in minutes) until the Weak Battery threshold is reached, based on Present Battery Strength and the device's present operational power needs. Note: this value shall exclude any additional power received from charging.</p> <p>A battery that is not capable of returning this information shall return a value of 0xFFFF.</p>										
6	<i>wRemainingChargeTime</i>	2	Number	<p>Shall contain the remaining time (in minutes) until the Charged Battery threshold is reached based on Present Battery Strength, charging power and the device's present operational power needs. Value shall only be valid if the Charging Flow is "Charging".</p> <p>A battery that is not capable of returning this information shall return a value of 0xFFFF.</p>										

If *wValue* or *wLength* are not as specified above, then the behavior of the PDUSB Device is not specified.

If *wIndex* refers to a Battery that does not exist, then the PDUSB Device shall respond with a Request Error.

If the PDUSB Device is not configured, the PDUSB Hub's response to this request is undefined.

### 9.4.3 GetPortPartnerPDOObjects (PDUSB Hub)

This request returns the Provider/Consumer PD Objects of the PDUSB Device attached to the downstream Port on a PDUSB Hub.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100011B	<i>GET_PARTNER_PDO</i>	Zero or one	Port Number	Variable	Power Data Objects

The *wIndex* field shall indicate the Port number to which a PDUSB Device is attached. The Port number shall be a valid Port number for that PDUSB Hub, greater than zero.

When this request is received by the PDUSB Hub it shall return the Consumer PDOs (if *wValue* is set to zero) or Provider PDOs (if *wValue* is set to one) for the PDUSB Device connected on the downstream Port indicated by the *wIndex* field. The PDUSB Hub shall use PD mechanisms (see Section 6.3.8) to retrieve the Power Data Objects from the PDUSB Device.

The *wLength* field shall specify the number of bytes to return. The maximum length of this field is a function of the maximum number of PDOs a PDUSB Device can return as defined in Section 6.2.1.2. If the device on the Partner Port does not support PD, then the PDUSB Hub shall respond with a Request Error.

If the PDUSB Hub is not configured, the PDUSB Hub's response to this request is undefined.

#### 9.4.4 GetPortPDStatus (PDUSB Hub)

This request returns the PD Status of the specified Port on a PDUSB Hub.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100011B	GET_STATUS	One	Port Number	Eight	PD Status

The *wIndex* field shall indicate the PD capable Port that is being queried. The Port number shall be a valid Port number for that PDUSB Hub, greater than zero.

If *wValue* or *wLength* are not as specified above, then the behavior of the PDUSB Hub is not specified.

If a Port is specified that does not exist, then the PDUSB Hub shall respond with a Request Error. The PD status changes on the Port are indicated in the PD Status Change field. The SPM can acknowledge the Port Change notifications via the Clear Port PD Feature (See Section 9.4.1).

If the PDUSB Hub is not configured, the PDUSB Hub's response to this request is undefined.

A GetPDStatus () request to a PDUSB Hub shall return the information as shown in Table 9-11.

Table 9-11 Port PD Status

Bit	Description																				
0:15	<p><b>PD Status Change:</b> A one in the bit position shall indicate the types of status changes that have occurred on the Port.</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Change Indicated</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Reserved and shall not be used.</td> </tr> <tr> <td>1</td> <td>External supply. When set, the <b>Supply</b> field shall indicate the current status of the supply.</td> </tr> <tr> <td>2</td> <td>Port operation mode. When set the <b>Port Operation Mode</b> field shall indicate the current operational mode of the Port.</td> </tr> <tr> <td>3</td> <td>Supported Provider Capabilities. When set, the SPM shall get the updated <b>Power Data Objects</b> from by retrieving the BOS Descriptor from the PDUSB Hub.</td> </tr> <tr> <td>4</td> <td>Negotiated power level. When set, the <b>Request Data Object</b> field shall indicate the newly negotiated power level if the PDUSB Hub was operating in Non-Intrusive mode. If the PDUSB Hub is working in intrusive mode then, when set, the <b>Request Data Object</b> field shall indicate the power level that the Port partner wants to operate at.</td> </tr> <tr> <td>5</td> <td>New power direction. When set, this field shall indicate that the power direction has changed if the PDUSB Hub was operating in Non-Intrusive mode. If the PDUSB Hub is working in intrusive mode then, when set, this field shall indicate the Port partner wants to perform a Power Role Swap.</td> </tr> <tr> <td>6</td> <td>PD Reset Complete. When set, this field shall indicate that a PD Hard Reset or a Cable Reset has completed. When this bit is set, then the other bits in the <b>PD Status Change</b> field shall report their change status as if the PDUSB Hub was working in Non-Intrusive mode.</td> </tr> <tr> <td>7</td> <td>Insertion Detect. When set, the <b>Insertion Detect</b> field shall indicate the current status of whether a cable is plugged into this Port.</td> </tr> <tr> <td>8</td> <td>PD Capable Cable. When set, the <b>PD Capable Cable</b> field shall indicate the current status of</td> </tr> </tbody> </table>	Bit	Change Indicated	0	Reserved and shall not be used.	1	External supply. When set, the <b>Supply</b> field shall indicate the current status of the supply.	2	Port operation mode. When set the <b>Port Operation Mode</b> field shall indicate the current operational mode of the Port.	3	Supported Provider Capabilities. When set, the SPM shall get the updated <b>Power Data Objects</b> from by retrieving the BOS Descriptor from the PDUSB Hub.	4	Negotiated power level. When set, the <b>Request Data Object</b> field shall indicate the newly negotiated power level if the PDUSB Hub was operating in Non-Intrusive mode. If the PDUSB Hub is working in intrusive mode then, when set, the <b>Request Data Object</b> field shall indicate the power level that the Port partner wants to operate at.	5	New power direction. When set, this field shall indicate that the power direction has changed if the PDUSB Hub was operating in Non-Intrusive mode. If the PDUSB Hub is working in intrusive mode then, when set, this field shall indicate the Port partner wants to perform a Power Role Swap.	6	PD Reset Complete. When set, this field shall indicate that a PD Hard Reset or a Cable Reset has completed. When this bit is set, then the other bits in the <b>PD Status Change</b> field shall report their change status as if the PDUSB Hub was working in Non-Intrusive mode.	7	Insertion Detect. When set, the <b>Insertion Detect</b> field shall indicate the current status of whether a cable is plugged into this Port.	8	PD Capable Cable. When set, the <b>PD Capable Cable</b> field shall indicate the current status of
Bit	Change Indicated																				
0	Reserved and shall not be used.																				
1	External supply. When set, the <b>Supply</b> field shall indicate the current status of the supply.																				
2	Port operation mode. When set the <b>Port Operation Mode</b> field shall indicate the current operational mode of the Port.																				
3	Supported Provider Capabilities. When set, the SPM shall get the updated <b>Power Data Objects</b> from by retrieving the BOS Descriptor from the PDUSB Hub.																				
4	Negotiated power level. When set, the <b>Request Data Object</b> field shall indicate the newly negotiated power level if the PDUSB Hub was operating in Non-Intrusive mode. If the PDUSB Hub is working in intrusive mode then, when set, the <b>Request Data Object</b> field shall indicate the power level that the Port partner wants to operate at.																				
5	New power direction. When set, this field shall indicate that the power direction has changed if the PDUSB Hub was operating in Non-Intrusive mode. If the PDUSB Hub is working in intrusive mode then, when set, this field shall indicate the Port partner wants to perform a Power Role Swap.																				
6	PD Reset Complete. When set, this field shall indicate that a PD Hard Reset or a Cable Reset has completed. When this bit is set, then the other bits in the <b>PD Status Change</b> field shall report their change status as if the PDUSB Hub was working in Non-Intrusive mode.																				
7	Insertion Detect. When set, the <b>Insertion Detect</b> field shall indicate the current status of whether a cable is plugged into this Port.																				
8	PD Capable Cable. When set, the <b>PD Capable Cable</b> field shall indicate the current status of																				

Bit	Description														
	whether a PD Capable Cable is plugged into this Port.														
9	Request Rejected. This bit shall be only valid if the PDUSB Hub is operating in Non-Intrusive mode. When set, this field shall indicate one of two conditions: <ul style="list-style-type: none"> <li>the PDUSB Hub received a request to change the power level (Negotiated power level (Bit 4) bit in this field shall be set) that was rejected or</li> <li>the PDUSB Hub received a request to perform a Power Role Swap (New power direction bit (Bit 5) in this field shall be set) that was rejected.</li> </ul>														
10	Reserved and shall be set to zero.														
11	Hybrid Mode Request. This bit shall only be set by the PDUSB Hub when it is operating in Hybrid mode. If this bit is set, then bits 4 and 5 ( <b>Negotiated power level</b> and <b>New power direction</b> ) shall be interpreted as if the hub was operating in Intrusive mode.														
12	Power Role Swap Completed. This shall be set when the hub completes a Power Role Swap requested by the SPM. The SPM can determine if the Swap request was successful or unsuccessful based on the <b>PD Direction</b> bit.														
13	VDM Received. When set, the SPM shall use the Get VDMs command (see section 9.4.8) to get further details on this notification.														
14	Reserved and shall be set to zero.														
15	Error. When set, this field shall indicate that an unknown error has occurred on the Port.														
16	<p><b>Supply:</b> This field shall indicate the type of supply that is attached to this Port. This field shall map directly to the "Externally Powered" bit provided as part of the Source or Sink Capabilities Message (see Section 6.4.1).</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No Supply</td> </tr> <tr> <td>1</td> <td>Supply Present</td> </tr> </tbody> </table>	Value	Meaning	0	No Supply	1	Supply Present								
Value	Meaning														
0	No Supply														
1	Supply Present														
17:19	<p><b>Port Operation Mode:</b> The field shall indicate the current mode of operation of the Port.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No Consumer</td> </tr> <tr> <td>1</td> <td>Legacy</td> </tr> <tr> <td>2</td> <td>BC</td> </tr> <tr> <td>3</td> <td>PD</td> </tr> <tr> <td>4</td> <td>Type-C Current</td> </tr> <tr> <td>5-7</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Meaning	0	No Consumer	1	Legacy	2	BC	3	PD	4	Type-C Current	5-7	Reserved
Value	Meaning														
0	No Consumer														
1	Legacy														
2	BC														
3	PD														
4	Type-C Current														
5-7	Reserved														
20	<p><b>Insertion Detect:</b> The field shall indicate the current status of whether a cable is plugged into this Port.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No cable detected</td> </tr> <tr> <td>1</td> <td>Cable detected</td> </tr> </tbody> </table>	Value	Meaning	0	No cable detected	1	Cable detected								
Value	Meaning														
0	No cable detected														
1	Cable detected														
21	<b>PD Capable Cable:</b> The field shall indicate the current status of whether a PD Capable cable is plugged into this Port.														

Bit	Description	
	<b>Value</b>	<b>Meaning</b>
	0	PD Capable Cable not detected
	1	PD Capable Cable detected
22	<b>PD Direction.</b> The field shall indicate whether the Port is operating as a Consumer or provider.	
	<b>Value</b>	<b>Meaning</b>
	0	Port is operating as a Consumer
	1	Port is operating as a provider
23:31	Reserved. Shall be set to zero.	
32:63	<b>Request Data Object:</b> This field shall return the currently negotiated power level if the hub was operating in Non-Intrusive mode. If the PDUSB Hub is working in intrusive mode then this field shall indicate the power level that the Port partner wants to operate at. See Table 6-13 for additional information on the contents of this data structure.	

#### 9.4.5 SetPDFeature (PDUSB Hub/Peripheral Device)

This request sets the value requested in the PDUSB Hub/Peripheral.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0000000B	SET_FEATURE	Feature Selector	Feature Specific	Zero	None

Setting a feature enables that feature or starts a process associated with that feature; see Table 9-8 for the feature selector definitions. Features that may be set with this request are:

- **BATTERY\_WAKE\_MASK**
- **OS\_IS\_PD\_AWARE** (Only valid for PDUSB Hubs)
- **POLICY\_MODE** (Only valid for PDUSB Hubs)
- **CHARGING\_POLICY**

##### 9.4.5.1 BATTERY\_WAKE\_MASK Feature Selector

When the feature selector is set to **BATTERY\_WAKE\_MASK**, then the *wIndex* field is structured as shown in the following table.

Table 9-12 Battery Wake Mask

Bit	Description
0	<b>Battery Present:</b> When this bit is set then the PDUSB Device shall generate a wake event if it detects that a battery has been attached.
1	<b>Charging Flow:</b> When this bit is set then the PDUSB Device shall generate a wake event if it detects that a battery switched from charging to discharging or vice versa.
2	<b>Battery Error:</b> When this bit is set then the PDUSB Device shall generate a wake event if the battery has detected an error condition.
15:3	Reserved and shall not be used.

The SPM may Enable or Disable the wake events associated with one or more of the above events by using this feature.

If the PDUSB Hub is not configured, the PDUSB Hub's response to this request is undefined.

#### 9.4.5.2 OS\_IS\_PD\_AWARE Feature Selector

When the feature selector is set to *OS\_IS\_PD\_AWARE*, then if the *wIndex* field is set to one it informs the PDUSB Hub that the SPM is active on the host. If the *wIndex* field is set to zero then it informs the PDUSB Hub that the SPM is not active on the host. The default state for this feature shall be zero. The PDUSB Hub shall respond to all PD requests (except GetBatteryStatus) or send any PD notifications only after the *OS\_IS\_PD\_AWARE* feature is set.

This is a valid command for the PDUSB Hub/Peripheral Device in the Address or Configured USB states.

#### 9.4.5.3 POLICY\_MODE Feature Selector

When the feature selector is set to *POLICY\_MODE*, the *wIndex* field shall be set to one of the values in Table 9-13.

Table 9-13 Policy Mode Encoding

Value	Description
00H	Local: Status Reporting only (Default)
01H	Intrusive: All requests are sent to the SPM
02H	Hybrid: Send only requests that cannot be accommodated to the SPM
03H-FFFFH	Reserved and shall not be used

If the Policy Mode is set to Intrusive, then the PDUSB Hub shall be set to operate in Intrusive mode. If the Policy Mode is set to Local or Hybrid, the hub shall be set to operate in Non-Intrusive mode.

When the Policy Mode is set to Intrusive then the PDUSB Hub shall pass any requests that need policy decisions to be made to the SPM and use PD mechanisms to delay the completion of PD Requests (e.g. using the Wait Message) over PD. When the Policy Mode is set to Hybrid then only requests which cannot be accommodated by the PDUSB Hub shall be sent to the SPM.

Some requests are only valid when the PDUSB Hub is operating in Intrusive mode. The requests that are valid in Intrusive mode are indicated as such in the description for those requests. In addition notifications that operate differently in Intrusive mode are also called out in the description of those notifications.

It shall be a Request Error if *wValue* is not a feature selector listed in Table 9-8 or if *wIndex* is not set to a value as specified above.

If the PDUSB Hub is not configured, the PDUSB Hub's response to this request is undefined.

#### 9.4.5.4 CHARGING\_POLICY Feature Selector

When the feature selector is set to *CHARGING\_POLICY*, the *wIndex* field shall be set to one of the values defined in Table 9-14. If the device is using USB Type-C Current above the default value or is using PD then this feature setting has no effect and the rules for power levels specified in the [\[USBType-C 1.0\]](#) or USB PD specifications shall apply.

Table 9-14 Charging Policy Encoding

Value	Description
00H	The device shall follow the default current limits as defined in the USB 2.0 or USB 3.1 specification, or as negotiated through other USB mechanisms such as BC. This is the default value.



Value	Description
01H	The Device may draw additional power during the unconfigured and suspend states for the purposes of charging.  For charging the device itself, the device shall limit its current draw to the higher of these two values: <ul style="list-style-type: none"> <li>• ICCHPF as defined in the USB 2.0 or USB 3.1 specification, regardless of its USB state.</li> <li>• Current limit as negotiated through other USB mechanisms such as BC.</li> </ul>
02H	The Device may draw additional power during the unconfigured and suspend states for the purposes of charging.  For charging the device itself, the device shall limit its current draw to the higher of these two values: <ul style="list-style-type: none"> <li>• ICCLPF as defined in the USB 2.0 or USB 3.1 specification, regardless of its USB state.</li> <li>• Current limit as negotiated through other USB mechanisms such as BC.</li> </ul>
03H	The device shall not consume any current for charging the device itself regardless of its USB state.
04H-FFFFH	Reserved and shall not be used

This is a valid command for the PDUSB Hub/Peripheral in the Address or Configured USB states. Further, it is only valid if the device reports a USB PD capability descriptor in its BOS descriptor and Bit 6 of the bmAttributes in that descriptor is set to 1. The device will go back to the wIndex default value of 0 whenever it is reset.

#### 9.4.6 SetPortPDFeature (PDUSB Hub)

This request sets the feature for a Port in a PDUSB Hub. This request shall only be valid when the PDUSB Hub is operating in Intrusive mode. If the PDUSB Hub is not operating in Intrusive mode, the PDUSB Hub's response to this request is undefined. If the PDUSB Hub cannot satisfy a request then it shall respond with a Request Error to the SPM.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100011B	SET_FEATURE	Feature Selector	Port Number	Zero	None

The *wIndex* field shall indicate the PD capable Port that is being queried. The Port number shall be a valid Port number for that hub, greater than zero.

Setting a feature enables that feature or starts a process associated with that feature; see Table 9-8 for the feature selector definitions. Features that may be set with this request are:

- **PR\_SWAP**
- **GOTO\_MIN**
- **RETURN\_POWER**
- **ACCEPT\_PD\_REQUEST**
- **REJECT\_PD\_REQUEST**
- **PORT\_PD\_RESET**
- **CABLE\_PD\_RESET**

When operating in intrusive mode the PDUSB Hub shall respond with a *Wait* Message when it receives any *Request* Message.

When the feature selector is set to **PR\_SWAP** then the PDUSB Hub shall initiate a Power Role Swap with the Port partner on the specified Port. See Section 7.3.9 and Section 7.3.10 for details on the *PR\_Swap* Message.

When the feature selector is set to **GOTO\_MIN** and the downstream Port is operating as a Provider, then the PDUSB Hub shall send the *GotoMin* Message to the Port partner on the specified Port. See Section 6.3.2 for details on the *GotoMin* Message.

When the feature selector is set to **RETURN\_POWER** and the downstream Port is operating as a Provider, then the PDUSB Hub shall return the power that it borrowed from the PD Device on that Port.

When the feature selector is set to **ACCEPT\_PD\_REQUEST** then the PDUSB Hub shall accept the request that was previously made by the Port partner on this downstream Port.

When the feature selector is set to **REJECT\_PD\_REQUEST** then the PDUSB Hub shall reject the request that was previously made by the Port partner on this downstream Port.

When the feature selector is set to **PORT\_PD\_RESET**, the Port shall be sent a Hard Reset. Once the Hard Reset is complete, as described in Section 0, the hub shall set the **C\_PORT\_PD\_CHANGE** bit in the Port change field.

When the feature selector is set to **CABLE\_PD\_RESET**, the hub shall send a Cable Reset to the cable attached on that Port. Once the Cable Reset is complete, as described in Section 6.7.3, the hub shall set the **C\_PORT\_PD\_CHANGE** bit in the Port change field.

It shall be a Request Error if *wValue* is not a feature selector listed in Table 9-8, or if *wIndex* specifies a Port that does not exist or if the Port is operating as a Consumer, or if *wLength* is not as specified above.

If the PDUSB Hub is not configured, the PDUSB Hub's response to this request is undefined.

#### 9.4.7 SetPortPDOs (PDUSB Hub)

This request sets the PDOs that a Port in a PDUSB Hub shall send to its Port Partner if the Port on the PDUSB Hub is operating as a Provider. Setting the PDO shall cause the Provider to send an updated **Source\_Capabilities** Message. The PDOs sent shall constitute a subset of the capabilities supported by the PDUSB Hub as defined in the PD Provider Port Capability Descriptor (see Section 9.2.4). When any PDO set exceeds the capabilities of the PDUSB Hub this request shall be rejected.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100011B	<b>SET_PDO</b>	Zero	Port Number	Variable	Provider Power Data Objects

The *wIndex* field shall indicate the PD capable Port that is being queried. The Port number shall be a valid Port number for that hub, greater than zero.

The *wLength* field shall be a multiple of 4 and shall include all the PDOs (see Section 6.4.1) that the Port may send to its Port partner when sending its PD capabilities. The Port shall send the PDOs in the order that they are included in this request. It is the responsibility of the PDUSB Hub to verify that the contents of the PDOs sent by the SPM are valid for this Port. A PDUSB Hub shall respond with a Request Error if the PDOs sent by the SPM are not valid for the Port specified.

It shall be a Request Error if *wValue* is not set to zero or if *wIndex* specifies a Port that does not exist, or if *wLength* is not as specified above.

If the Port on the PDUSB Hub is not operating as a Provider, the PDUSB Hub's response to this request is undefined.

If the PDUSB Hub is not configured, the PDUSB Hub's response to this request is undefined.

#### 9.4.8 GetVDM (PDUSB Hub)

This request instructs the PDUSB Hub to return the VDMs that it received from a cable or device attached to a Port in a PDUSB Hub. The response to each request shall contain only one VDM. VDMs shall be returned in the order they were received by the Port. If there are no VDMs to return for the Port the PDUSB Hub shall return a zero length packet.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100011B	<b>GET_VDM</b>	Zero	Port Number	Variable	VDM

The *wIndex* field shall indicate the PD capable Port that is being queried. The Port number shall be a valid Port number for that PDUSB Hub, greater than zero.

The *wLength* field shall be a multiple of 4 and shall include all the VDOs that the PDUSB Hub received on that Port (see Section 6.4.4).

It shall be a Request Error if *wIndex* specifies a Port that does not exist, or if *wValue* or *wLength* is not as specified above.

If the PDUSB Hub is not configured, the PDUSB Hub's response to this request is undefined.

#### 9.4.9 SendVDM (PDUSB Hub)

This request instructs the PDUSB Hub to send a VDM to the one of three possible recipients (SOP, SOP' or SOP'') attached to a Port in a PDUSB Hub.

<b>bmRequestType</b>	<b>bRequest</b>	<b>wValue</b>	<b>wIndex</b>	<b>wLength</b>	<b>Data</b>
00100011B	<i>SEND_VDM</i>	SOP/SOP'/SOP''	Port Number	Variable	VDM

The *wIndex* field shall indicate the PD capable Port that is being queried. The Port number shall be a valid Port number for that PDUSB Hub, greater than zero.

The *wValue* field indicates the recipient. If the value is 0 it shall be sent to its Port partner. If the value is 1 it shall be sent to SOP'. If the value is 2 it shall be sent to SOP''. No other values in *wValue* are allowed.

The *wLength* field shall be a multiple of 4 and shall include one VDM (see Section 6.4.4) that the Port shall send to the recipient indicated by *wValue*.

It shall be a Request Error if *wIndex* specifies a Port that does not exist, or if *wValue* or *wLength* is not as specified above.

If the PDUSB Hub is not configured, the PDUSB Hub's response to this request is undefined.

## A. Power Profiles

Power Profiles are optional normative. They define a standardized set of voltages at several current ranges that are offered by USB Power Delivery Sources. The profiles are defined for Sources only.

The Power Profiles are optional but are intended to provide a finite set of power levels to:

- Limit the number of voltages and current combinations a Source has to supply
- Provide a well-defined set of voltage and current combinations from which a Sink can choose
- Provide a selection of power ranging from 10W to 100W in approximately 2x steps
- Limit the number of valid combinations

### A.1 Profile Definitions

There are the following profiles based on Fixed Supply Objects:

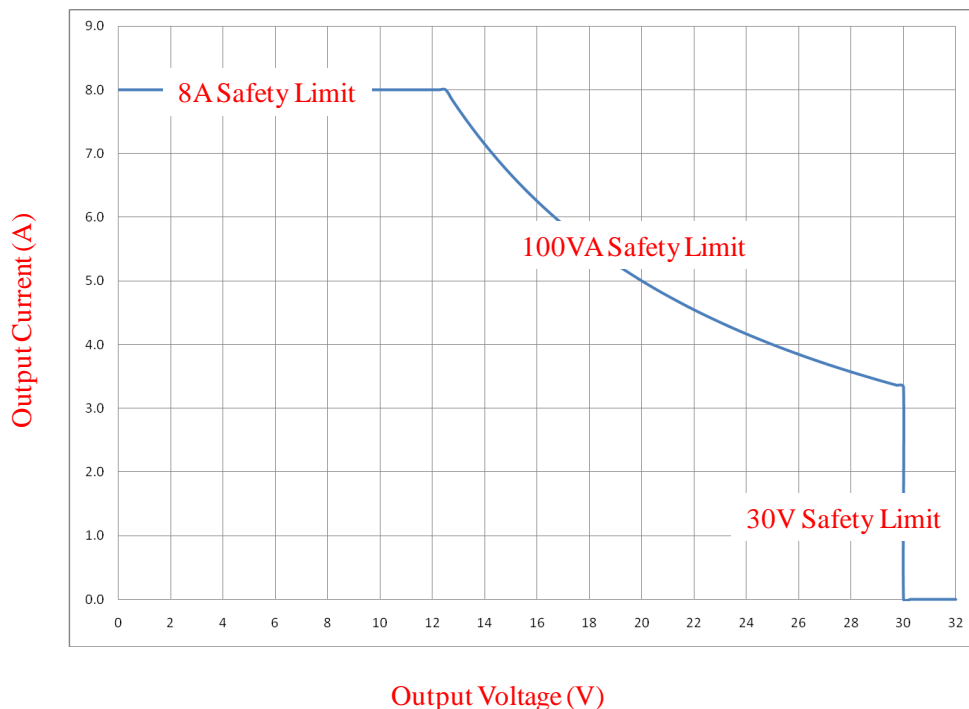
- Profile 0 reserved
- Profile 1 capable of supplying at least 5V @ 2.0A
- Profile 2 ports are capable of supplying at least 5V @ 2.0A, 12V @ 1.5A.
- Profile 3 ports are capable of supplying at least 5V @ 2.0A, 12V @ 3A.
- Profile 4 ports are capable of supplying at least 5V @ 2.0A, 12V and 20V at 3A respectively.
- Profile 5 ports are capable of supplying at least 5V @ 2.0A, 12V and 20V at 5A respectively.

Power Profiles are defined to overlap such that a Device that requires a Profile 2 Source will operate equally well when connected to a Profile 2 or any higher Profile Source.

Sources may have additional capabilities. For example a Source might advertise 5V @ 3.0A, 12V and 15V at 1.5A respectively. It is a Profile 2 Source because it meets the Profile 2 requirements to supply 5V @ 2.0A and 12V @ 1.5A. The fact that it can also supply 5V @ 3.0A and 15V will have no effect on a Device that wants a Profile 2 source.

Profile 5 Sources that are capable of 100W operation are subject to various worldwide safety standards. In order to meet the most common safety standards, the continuous output power cannot exceed 100W and the continuous output current cannot exceed 5A. The industry is well versed in meeting the safety requirements for such power sources (e.g. Wall Warts). Refer to Figure A-1 for an interpretation of the safety requirements imposed by IEC/UL60950 Table 2B.

Figure A-1 Interpretation of UL60950



## A.2 Voltage Selection Rationale

Voltages used by the profiles were not picked randomly; this section describes the rationale behind the choices.

5V is the USB default that must be supported to provide interoperability with existing Devices. However, higher voltages are needed to provide more power through USB connectors because of their current carrying capability.

12V was selected because it is very common in PCs and many other systems. The current limitation of the Micro USB Connector family is 3A for the enhanced PD version. The use of 12V with 3A provides sufficient power to charge tablets in the 20-30W range that use the Micro USB Connector.

20V was selected because larger systems, such as notebooks, often have 4 lithium cells in series and require charging voltages in the 18-20V range. A sampling of systems showed that chargers for these systems were typically 19.5-20V. Typical systems had chargers that supplied between 60W and 100W with the exception of a few very high-end performance systems that were well over 100W. The 5A current limit of the PD enhanced Standard-A and Standard-B connectors, with 20V allows up to 100W to be delivered to charge this class of Devices.

## A.3 Relevance of Profiles to Sink Ports

Sinks do not have Power Profiles per se. A Sink Port only need to be compatible with the voltage and current offered in one or more Power Profiles. This can lead to some interesting cases where Dual-Role ports are involved (see Appendix 0 for examples). A dual-role Port may offer one Profile when operating as a Source and request power from another Profile when operating as a Sink. For example, a tablet might offer Profile 1 (5V @ 2.0A) when operating as a Source, but request power (12V @ 3A) supplied by a Profile 3 Source when operating as a Sink. Another example is a

notebook computer with a Profile 2 Source Port it also uses as a Sink for charging which requires it be supplied with 20V @ 5A from a Profile 5 Source.

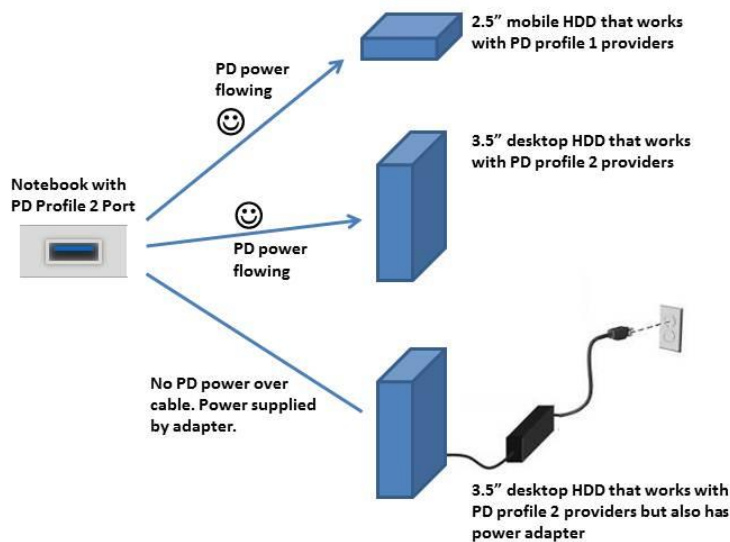
## A.4 System Level Examples

The following examples are provided to help illustrate some typical system use cases. Examples using a notebook and HDDs have been selected but the principles apply equally to other types of USB Power Delivery devices (e.g. printers, scanners, tablets, phones, mp3 players etc.).

### A.4.1 Notebook and HDD Examples

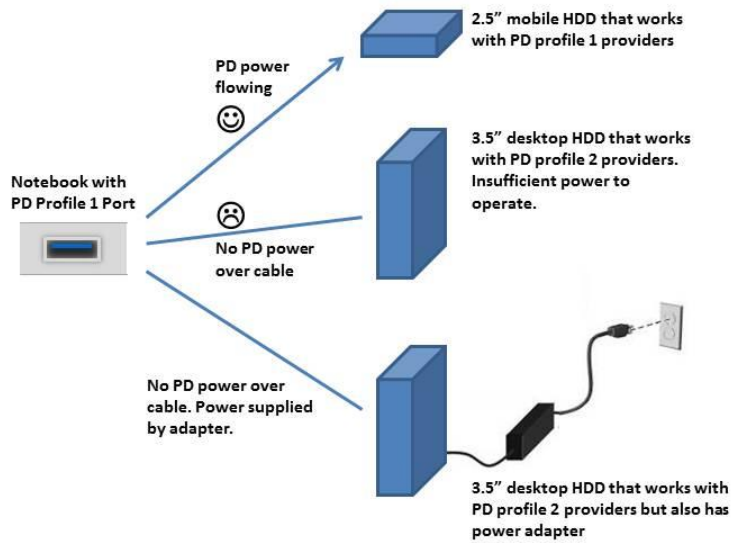
In the following two examples, the user has varying experiences attaching his hard disk-drive to a PD Port on his notebook computer.

Figure A-2 Notebook is Capable of Meeting User's Requirements



The notebook shown in Figure A-2 is able to supply up to PD profile 2. The larger HDD is able to successfully operate without a separate power adapter since it gets adequate power from the PD Port. In the case of the HDD with separate power adapter plugged in, the HDD should not draw power from the notebook's Port.

Figure A-3 Notebook is not Capable of Meeting User's Requirements

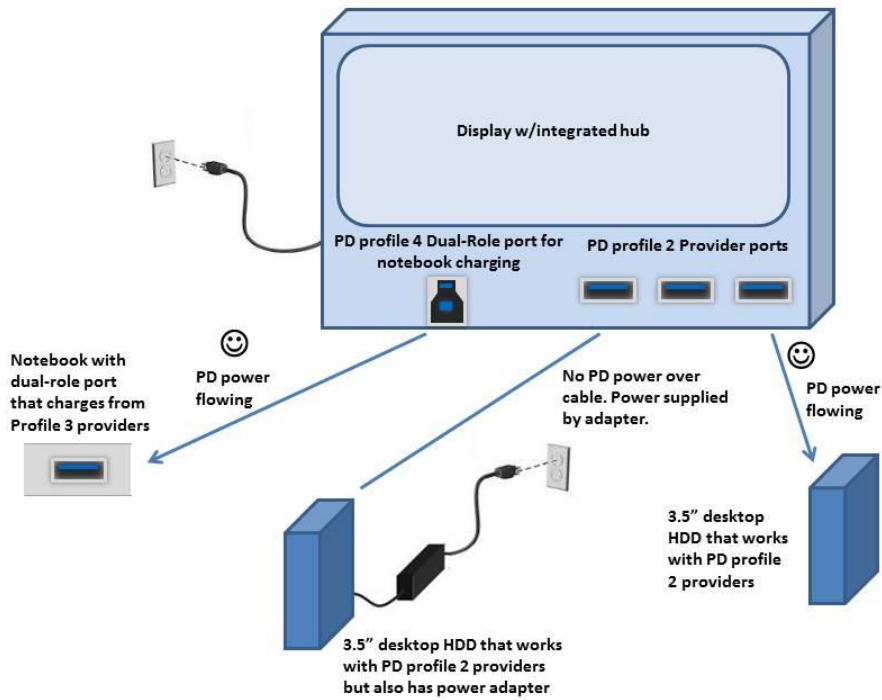


Note that in the case shown in Figure A-3, the experience for the user is not unlike their experience today. USB HDDs that attempt to rely only on USB bus power only will often fail because not all notebooks on the market are capable of supplying adequate current.

#### A.4.2 Display with Hub Example

The following example is for a display dock where the integrated hub also provides power and at least one of the hub ports is of high enough power capacity to power the host platform.

Figure A-4 Display with Integrated Hub Capable of Meeting User's Requirements





## B. CRC calculation

### B.1 C code example

```
//
// USB PD CRC Demo Code.
//
#include <stdio.h>

int crc;

//-----

void crcBits(int x, int len) {

    const int poly = 0x04C11DB6; //spec 04C1 1DB7h
    int newbit,newword,r1_crc;

    for(int i=0;i<len;i++) {

        newbit = ((crc>>31) ^ ((x>>i)&1)) & 1;
        if(newbit) newword=poly; else newword=0;
        r1_crc = (crc<<1) | newbit;
        crc = r1_crc ^ newword;
        printf("%2d newbit=%d, x>>i=0x%x, crc=0x%x\n",i,newbit, (x>>i),crc);
    }
}

int crcWrap(int c){

    int ret = 0;
    int j, bit;

    c = ~c;
    printf("~crc=0x%x\n",c);

    for(int i=0;i<32;i++) {
        j = 31-i;

        bit = (c>>i) & 1;
        ret |= bit<<j;
    }
}
```

```
    }
    return ret;
}

//-----

int main(){

    int txCrc=0,rxCrc=0,residue=0,data;

    printf("using packet data 0x%x\n", data=0x0101);

    crc = 0xffffffff;
    crcBits(data,16);
    txCrc = crcWrap(crc);

    printf("crc=0x%x, txCrc=0x%x\n",crc,txCrc);

    printf("received packet after decode= 0x%x, 0x%x\n",data,txCrc);

    crc = 0xffffffff;
    crcBits(data,16);
    rxCrc = crcWrap(crc);

    printf("Crc of the received packet data is (of course) =0x%x\n",rxCrc);

    printf("continue by running the transmit crc through the crc\n");
    crcBits(rxCrc,32);

    printf("Now the crc residue is 0x%x\n",crc);

    printf("should be 0xc704dd7b\n");

}
```

## B.2 Table showing the full calculation over one Message

Function	Nibble	Symbol	Bits	CRC register transmitter	CRC register receiver	bit nr.	Function	Nibble	Symbol	Bits	CRC register transmitter	CRC register receiver	bit nr.									
P r e a m b l e			0	FFFFFFF	FFFFFFF	1	GoodCRC Header #0101	#1	#09	1	FFFFFFF	FFFFFFF	85									
			1	FFFFFFF	FFFFFFF	2				0	FFFFFFF	FFFFFFF	86									
			0	FFFFFFF	FFFFFFF	3				0	F83EE24B	FFFFFFF	87									
			1	FFFFFFF	FFFFFFF	4				1	F28CD921	FFFFFFF	88									
			0	FFFFFFF	FFFFFFF	5				0	E18BAFF5	FFFFFFF	89									
			1	FFFFFFF	FFFFFFF	6				0	E18BAFF5	FFFFFFF	90									
			0	FFFFFFF	FFFFFFF	7				1	C7B0425D	FFFFFFF	91									
			1	FFFFFFF	FFFFFFF	8				1	8BA1990D	F83EE24B	92									
			0	FFFFFFF	FFFFFFF	9				1	13822FAD	F28CD921	93									
			1	FFFFFFF	FFFFFFF	10				1	27045F5A	E18BAFF5	94									
			0	FFFFFFF	FFFFFFF	11				1	27045F5A	E18BAFF5	95									
			1	FFFFFFF	FFFFFFF	12				0	4AC9A303	C7B0425D	96									
			0	FFFFFFF	FFFFFFF	13				0	95934606	8BA1990D	97									
			1	FFFFFFF	FFFFFFF	14				1	2FE7918B	13822FAD	98									
			0	FFFFFFF	FFFFFFF	15				0	5FCF2376	27045F5A	99									
			1	FFFFFFF	FFFFFFF	16				0	5FCF2376	27045F5A	100									
			0	FFFFFFF	FFFFFFF	17				1	B9E46EC	4AC9A303	101									
			1	FFFFFFF	FFFFFFF	18				1	7BF0906F	95934606	102									
			0	FFFFFFF	FFFFFFF	19				1	F7F820DE	2FE7918B	103									
			1	FFFFFFF	FFFFFFF	20				1	EB375C0B	5FCF2376	104									
			0	FFFFFFF	FFFFFFF	21				0	EB375C0B	5FCF2376	105									
			1	FFFFFFF	FFFFFFF	22				1	EB375C0B	BF9E46EC	106									
			0	FFFFFFF	FFFFFFF	23				0	EB375C0B	7BF0906F	107									
			1	FFFFFFF	FFFFFFF	24				0	EB375C0B	F7F820DE	108									
			0	FFFFFFF	FFFFFFF	25				1	EB375C0B	EB375C0B	109									
			1	FFFFFFF	FFFFFFF	26				0	EB375C0B	EB375C0B	110									
			0	FFFFFFF	FFFFFFF	27				0	EB375C0B	D04FAS41	111									
			1	FFFFFFF	FFFFFFF	28				1	EB375C0B	A19E56F5	112									
			0	FFFFFFF	FFFFFFF	29				0	EB375C0B	47F0805D	113									
			1	FFFFFFF	FFFFFFF	30				1	EB375C0B	88A7D0D	114									
			0	FFFFFFF	FFFFFFF	31				1	EB375C0B	88A7D0D	115									
			1	FFFFFFF	FFFFFFF	32				0	EB375C0B	1285E7AD	116									
			0	FFFFFFF	FFFFFFF	33				1	EB375C0B	21AAD2ED	117									
			1	FFFFFFF	FFFFFFF	34				0	EB375C0B	4355A50A	118									
			0	FFFFFFF	FFFFFFF	35				1	EB375C0B	86A84BB4	119									
			1	FFFFFFF	FFFFFFF	36				1	EB375C0B	86A84BB4	120									
			0	FFFFFFF	FFFFFFF	37				0	EB375C0B	0D569788	121									
			1	FFFFFFF	FFFFFFF	38				0	EB375C0B	16C3367	122									
			0	FFFFFFF	FFFFFFF	39				1	EB375C0B	3CDB66CE	123									
			1	FFFFFFF	FFFFFFF	40				0	EB375C0B	7980C9C	124									
			0	FFFFFFF	FFFFFFF	41				1	EB375C0B	7980C9C	125									
			1	FFFFFFF	FFFFFFF	42				1	EB375C0B	F7A0868F	126									
			0	FFFFFFF	FFFFFFF	43				0	EB375C0B	EB8010A9	127									
			1	FFFFFFF	FFFFFFF	44				1	EB375C0B	D3C13CE5	128									
			0	FFFFFFF	FFFFFFF	45				0	EB375C0B	A343647D	129									
			1	FFFFFFF	FFFFFFF	46				0	EB375C0B	A343647D	130									
			0	FFFFFFF	FFFFFFF	47				1	EB375C0B	4686C8FA	131									
			1	FFFFFFF	FFFFFFF	48				0	EB375C0B	80D091F4	132									
			0	FFFFFFF	FFFFFFF	49				1	EB375C0B	141D23E8	133									
			1	FFFFFFF	FFFFFFF	50				1	EB375C0B	343647D0	134									
			0	FFFFFFF	FFFFFFF	51				1	EB375C0B	343647D0	135									
			1	FFFFFFF	FFFFFFF	52				0	EB375C0B	686C8FA0	136									
			0	FFFFFFF	FFFFFFF	53				1	EB375C0B	D0D91F40	137									
			1	FFFFFFF	FFFFFFF	54				1	EB375C0B	A1823E80	138									
			0	FFFFFFF	FFFFFFF	55				1	EB375C0B	43647D00	139									
			1	FFFFFFF	FFFFFFF	56				0	EB375C0B	43647D00	140									
			0	FFFFFFF	FFFFFFF	57				0	EB375C0B	8209E7B7	141									
			1	FFFFFFF	FFFFFFF	58				1	EB375C0B	0413CF6E	142									
			0	FFFFFFF	FFFFFFF	59				0	EB375C0B	0CE8836B	143									
			1	FFFFFFF	FFFFFFF	60				1	EB375C0B	100C1961	144									
			0	FFFFFFF	FFFFFFF	61				1	EB375C0B	100C1961	145									
			1	FFFFFFF	FFFFFFF	62				0	EB375C0B	3A1836C2	146									
			0	FFFFFFF	FFFFFFF	63				1	EB375C0B	70E17033	147									
			1	FFFFFFF	FFFFFFF	64				1	EB375C0B	E1E2E066	148									
			0	FFFFFFF	FFFFFFF	65				0	EB375C0B	C704DD7B	149									
S O P	Sync1 (#18)	0	FFFFFFF	FFFFFFF	66	Note: CRC transmitter is calculated over data bytes only, in casu marked nibbles, and calculation results are available one (bit-) clock later	#0D	#0D	0	FFFFFFF	FFFFFFF	149										
		0	FFFFFFF	FFFFFFF	67																	
		1	FFFFFFF	FFFFFFF	68																	
	Sync1 (#18)	1	FFFFFFF	FFFFFFF	69				Note: CRC receiver is calculated over data bytes and received CRC bytes, in casu marked nibbles, and calculation results are available five (bit-) clocks later	#0D	#0D	0	FFFFFFF	FFFFFFF	149							
		0	FFFFFFF	FFFFFFF	70																	
		0	FFFFFFF	FFFFFFF	71																	
	Sync1 (#18)	1	FFFFFFF	FFFFFFF	72							Fixed residual	#0D	#0D	0	FFFFFFF	FFFFFFF	149				
		1	FFFFFFF	FFFFFFF	73																	
		1	FFFFFFF	FFFFFFF	74																	
	Sync1 (#18)	0	FFFFFFF	FFFFFFF	75																	
		0	FFFFFFF	FFFFFFF	76																	
		0	FFFFFFF	FFFFFFF	77																	
	Sync1 (#18)	1	FFFFFFF	FFFFFFF	78																	
1		FFFFFFF	FFFFFFF	79																		
1		FFFFFFF	FFFFFFF	80																		
Sync2 (#11)	0	FFFFFFF	FFFFFFF	81																		
	0	FFFFFFF	FFFFFFF	82																		
	0	FFFFFFF	FFFFFFF	83																		
			1	FFFFFFF	FFFFFFF	84																

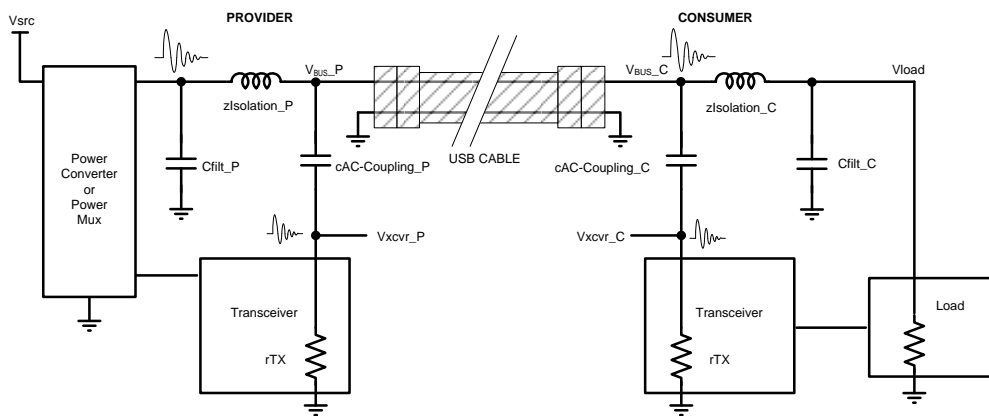
## C. Power Implementation Considerations

This section has been added as an informative guide for implementers of the PD specification. Expansion of USB Power Delivery to higher voltage and power levels with its AC coupled communication over  $V_{BUS}$  requires new design considerations to the power delivery system. Many of these may not have been as important in legacy USB systems, but should be considered when implementing the PD specification. Component labels within this Appendix are specific to the Appendix unless otherwise stated.

### C.1 Managing Isolation Impedance (BFSK)

The isolation impedance outlined in the PD Specification Section 5.8.2.2 is used to block specific frequency bands and pass others in order to provide a communication path on the  $V_{BUS}$  conductor. This impedance will likely be inductive in nature (for example a  $1\mu\text{H}$  inductor see Table 5-17). The following section describes potential problems that may arise due to this impedance. Measurement techniques to validate conducted noise will be discussed.

Figure C-1 Typical System Electrical Model



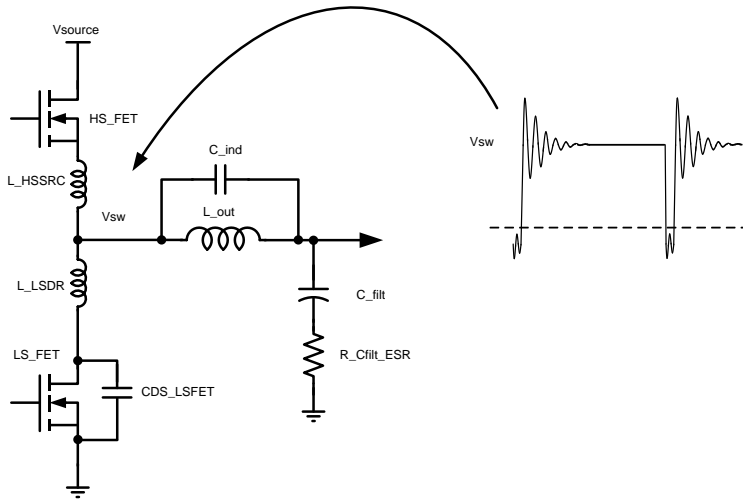
#### C.1.1 In-band fCarrier Spurious Noise

The output stage of a switch mode power converter can typically produce high frequency noise. This noise is differentiated from fundamental switching ripple. The noise tends to be a high frequency damped sinusoid occurring regularly at the converter power switch transitions. It is likely that some implementations will produce noise that coincides with the  $f_{Carrier}$  frequency band of the transceivers.

Figure C-2 shows parasitic elements within a basic synchronous buck power stage that can generate high frequency conducted noise which may interfere with the PD Transceiver. A basic synchronous buck stage is shown that is comprised of two power MOSFETs, HS\_FET and LS\_FET, an inductor,  $L_{out}$ , and output filter capacitor,  $C_{filt}$ . Also shown are the parasitic elements that create damped sinusoid noise that may be in contention with  $f_{Carrier}$ .  $L_{HSSRC}$ ,  $L_{LSDRV}$ , and  $CDS_{LSFET}$  are typically responsible for creating resonant noise on the VSW node. Other topologies may have different parasitic elements to consider. This waveform can couple to the output through  $C_{in}$  and/or circuit board parasitics and impose a voltage across  $R_{Cfilr\_ESR}$ . This section is meant to help implementers identify and resolve noise problems in the design process before submitting PD designs to compliance testing.

### C.1.2 Spurious Noise Test Setup and Calibration

Figure C-2 Typical Synchronous Buck Power Stage with Parasitics



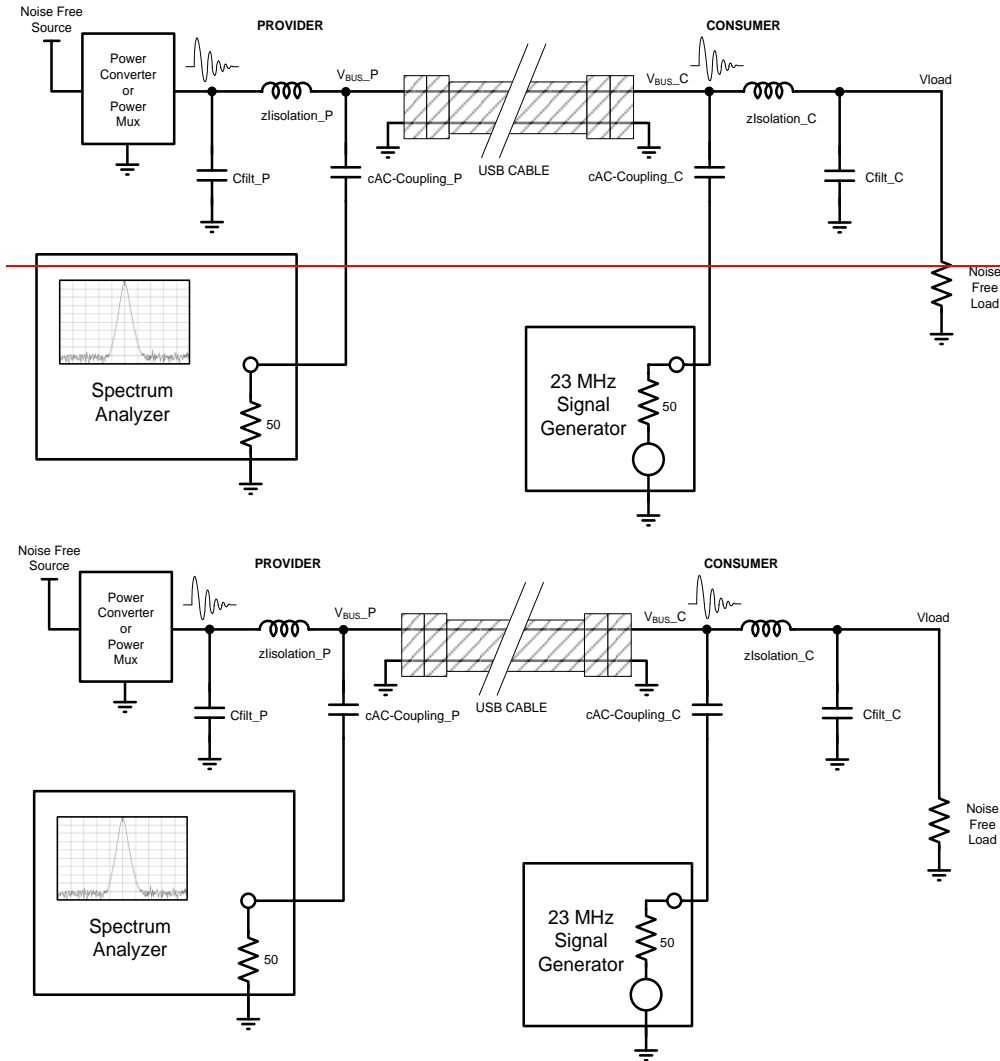
Measurement of in-band spurious noise from the power converter involves setting up measurement equipment with the correct scales and impedances. The process involves calibrating a known carrier signal in a test setup containing the correct source and load impedances with the power converter connected, but not operational. Once the test setup is calibrated, the operational noise can be evaluated. In order to simplify this test and to match industry standard communication test equipment, 50 Ohm terminations and dBm scaled measurements will be used.

Measurement of in-band spurious noise will require the use of:

- An RF signal generator capable of providing a  $v_{TX}$  nominal (150mVrms),  $f_{Carrier}$  nominal (23MHz) carrier sine wave into a 50 ohm load.
- An oscilloscope with at least 200MHz Bandwidth.
- A spectrum analyzer with better than -80dBm measurement capability and 2MHz/div scale capability at 23MHz center frequency setting.
- A low noise environment with less than -60dBm of RF noise at 13 to 33MHz.

Figure C-3 shows the test setup which includes both the Provider and Consumer to be connected using PD USB cables. During calibration the power converter and load impedances should be in place with the power converter turned off.

Figure C-3 Spurious Noise Measurement Test Setup



- 1) Using appropriately matched RF cabling configure the test setup as shown in Figure C-3 set the termination for both the signal generator and the spectrum analyzer to 50 ohms.
- 2) Set the signal generator to  $((V_{TX} \text{ minimum} \times 1.414) \times 2)$  Peak to Peak ( $\sim 282\text{mV}$ ) at  $f_{Carrier}$  nominal (23MHz). Confirm the level using an oscilloscope.
- 3) Set the spectrum analyzer to  $f_{Carrier}$  nominal (23MHz) Center Frequency,  $f_{Carrier}$  nominal  $\pm 10\text{MHz}$  span and set the Resolution Bandwidth (RBW) to 10 KHz.
- 4) Confirm that the measured level on the Spectrum Analyzer is approximately -7 to -13 dBm (dBMilliwatts on 50 Ohms). Record the Baseline level (dBM Baseline).

- 5) Confirm that no spurious noise levels are present above -60dBm. If the noise floor is higher, it may be necessary to test in an environment with EMI shielding from radio noise sources.
- 6) Turn on the power converter and turn off the *fCarrier* nominal signal from the signal generator.
- 7) Confirm that the noise floor has not increased beyond (dBm\_Baseline - *snrSrc*). Reference Figure 7-8 for in-band noise. The (dBm\_Baseline - *snrSrc*) represents the 0dB level in Figure 7-8. Confirm that all measured noise falls below the curve.

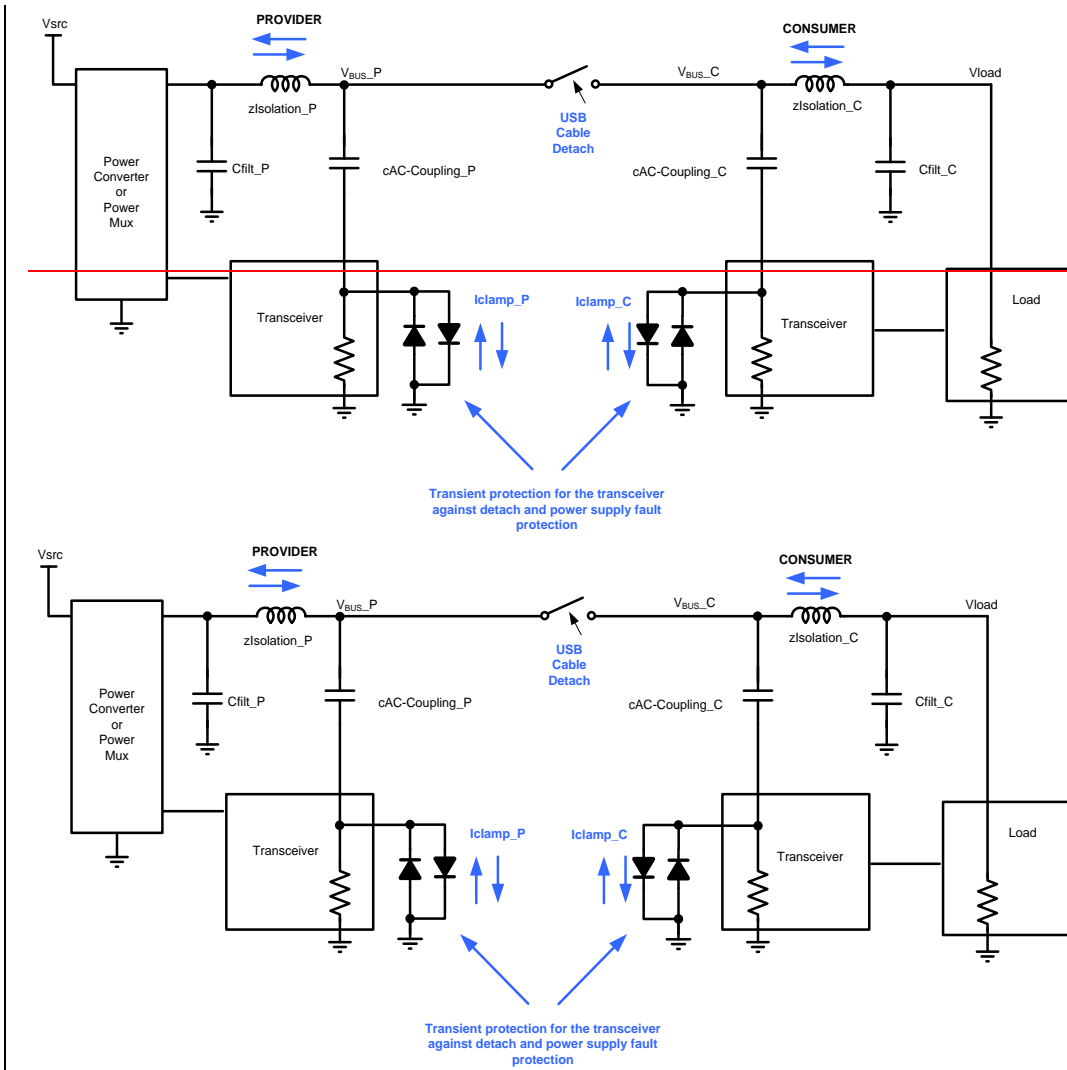
The same measurement can be made from the perspective of the Sink using a quiet source. In this case the frequency generator is placed on the Provider side and the signal generator is placed on the Consumer side. The levels would be compared to Figure 7-11 in this case.

\* Noise levels should be validated across expected line and load conditions including expected combinations of USB Cable types and lengths.

## C.2 Connector Detach Transients

The presence of inductive elements *z*isolation\_P and *z*isolation\_C will cause transient voltages to be presented to the transceiver inputs. This section describes this behavior and some protection methods that can be considered.

Figure C-4 Current Transients when Cable/Load Removed



As shown in Figure C-4, equal current is flowing in both isolation elements (*z*isolation\_P and *z*isolation\_C) within the Provider and Consumer just prior to detach. Inductive elements resist changes in current and will force voltage transients on  $V_{BUS\_P}$  and  $V_{BUS\_C}$  terminals just following detach. These high  $dv/dt$  rate voltages will AC couple through the coupling capacitors *cAC-Coupling\_P* and *cAC-Coupling\_C*. A positive going voltage transient will be presented on



the Provider side and a negative going voltage transient will be presented on the Consumer side. Clamping elements should be included or the transceiver should be capable of absorbing the energy of the attach and detach events to prevent damage to the transceiver.

Figure C-5 Isolation Inductor Energy versus Load

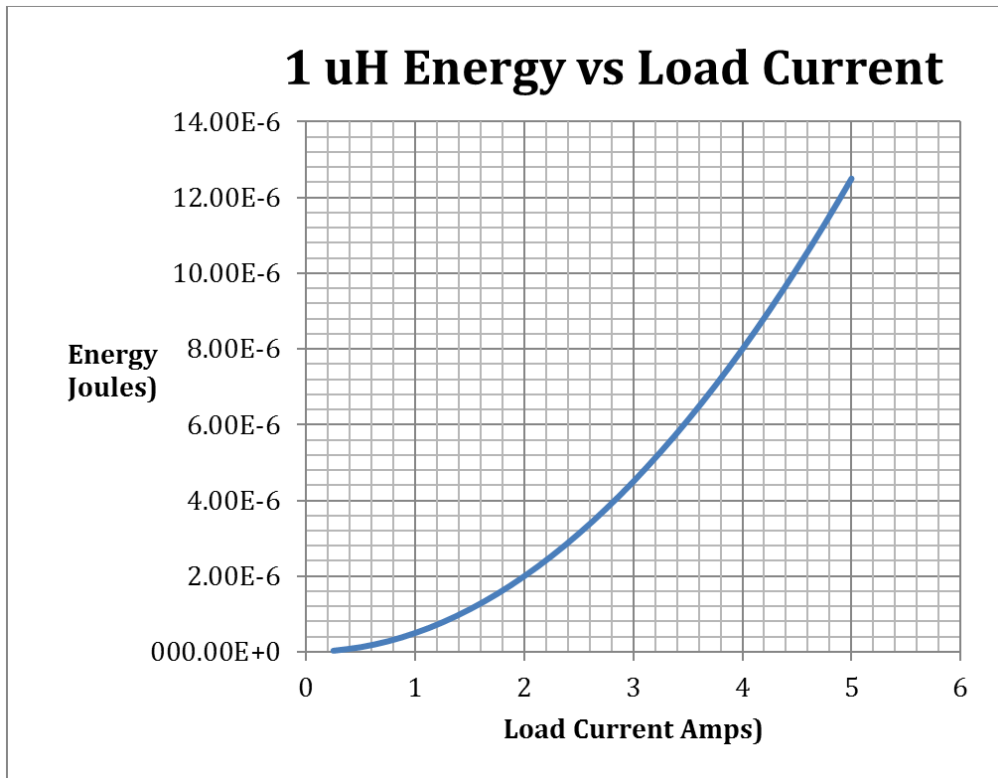


Figure C-5 shows the total energy that could be delivered during a detach event with the example 1µH inductor where:

$$Energy = \frac{LI^2}{2}$$

An external or integrated clamp should be implemented to absorb this energy and limit the applied voltage at the transceiver input.

### C.3 Closed Loop Stability Effects

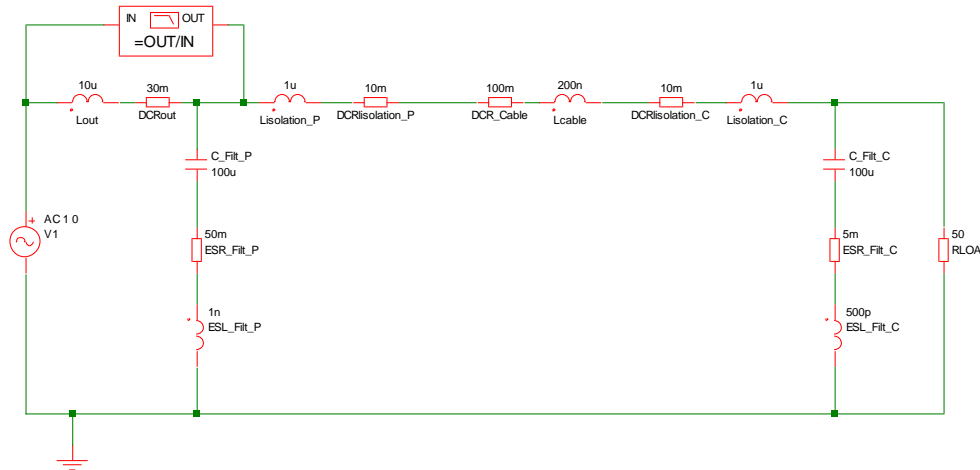
Addition of the isolation inductor as well as cable inductance will affect the small signal power stage response of a switch mode power converter PD implementation.

#### C.3.1 Basic Power Stage Small Signal AC Model

Figure C-6 shows a simplified diagram of the small signal power stage AC response including the parasitic elements that should be considered. Power stage response refers to the voltage regulation system around which a feedback

loop will be applied. This model does not include modulator gain or dc gain in the power stage as it is only meant to display exemplary parasitic poles and zeroes in the typical system.

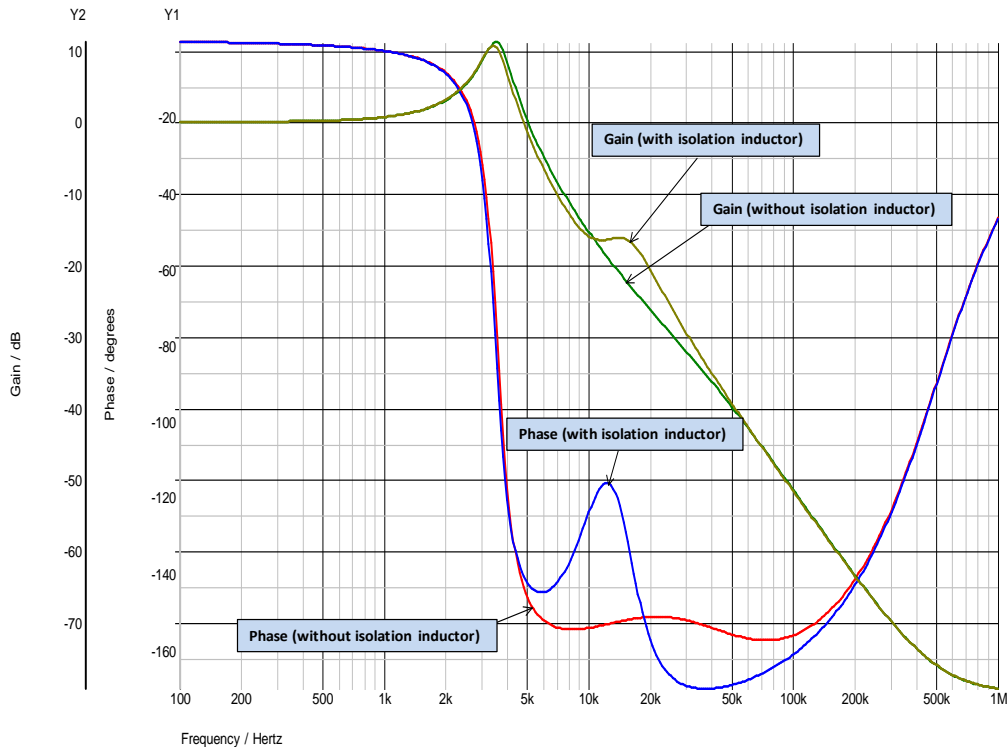
Figure C-6 Simplified Small Signal AC Model



The dominant pole of the power converter is set by  $L_{out}$  and  $C_{Filt\_P}$  assuming the value of  $L_{out}$  is larger than  $(z_{lisolation\_P} + L_{cable} + z_{lisolation\_C})$ . Parasitic ESR (Equivalent Series Resistance) and ESL (Equivalent Series Inductance) form high frequency zeroes in the power stage response gain and phase. Phase distortion is introduced into the power stage response from  $(z_{lisolation\_P} + L_{cable} + z_{lisolation\_C})$  interacting with  $C_{Filt\_C}$ . Secondary high frequency poles and zeroes can degrade phase margin in the feedback loop of the power converter. These effects need to be modeled and accounted for when designing the feedback loop for stability and transient performance. Figure C-7 shows the resulting phase and gain impacts of the parasitics using the model in Figure C-6.

These traces compare the power stage phase and gain with and without isolation inductance. In this case, a phase boost is introduced which is not necessarily bad for compensation; however it is important for the designer to be aware of these effects to maintain predictable stability and transient response.

Figure C-7 Power Stage Phase And Gain with and without Isolation Inductors



It is important to note that worst case conditions tend to occur at:

- Light load (high Q Condition)
- Load capacitance highly tuned to series isolation inductance (high Q Condition)
- Main power converter LC pole ( $L_{out}$ ,  $C_{Filt\_P}$ ) is selected too close to the parasitic LC ( $(z_{Isolation\_P} + z_{Isolation\_C})$ ,  $C_{Filt\_C}$ )

It is also important to note that selection of a  $C_{Filt\_C}$  capacitor network that has parasitic ESR large enough to detune the dominant pole will help reduce the negative phase effects.

### C.3.2 Feedback Past Isolation Inductor

It may be desirable to compensate the voltage loop past the Provider isolation inductor ( $z_{Isolation\_P}$  in order to eliminate the IR voltage drop due to the DC Resistance of the inductor. This further complicates the power stage response curve by placing the inductors reactance in series with the feedback path. Figure C-7 shows the same simple model with feedback measured past the isolation inductor.

Figure C-8 Simplified Small Signal AC Model (Feedback before and after Inductor zIsolation\_P)

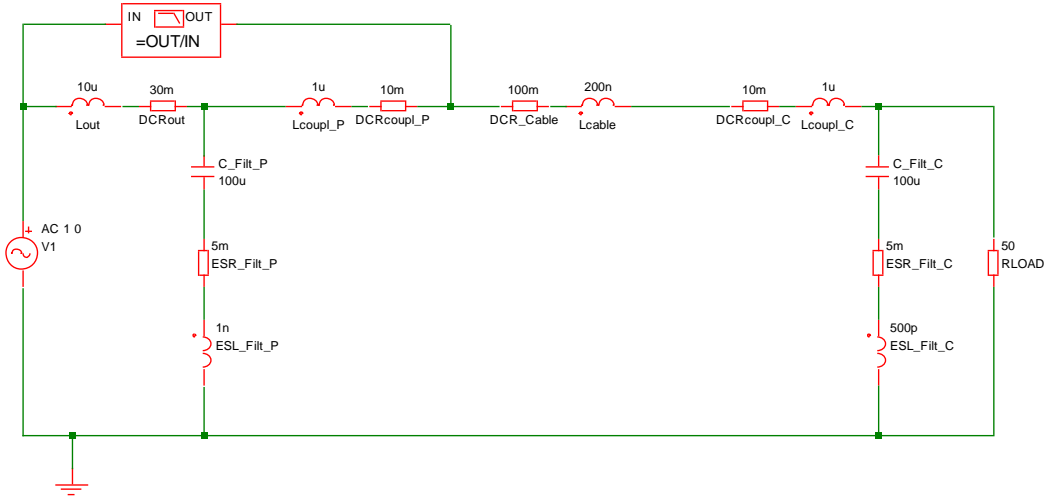
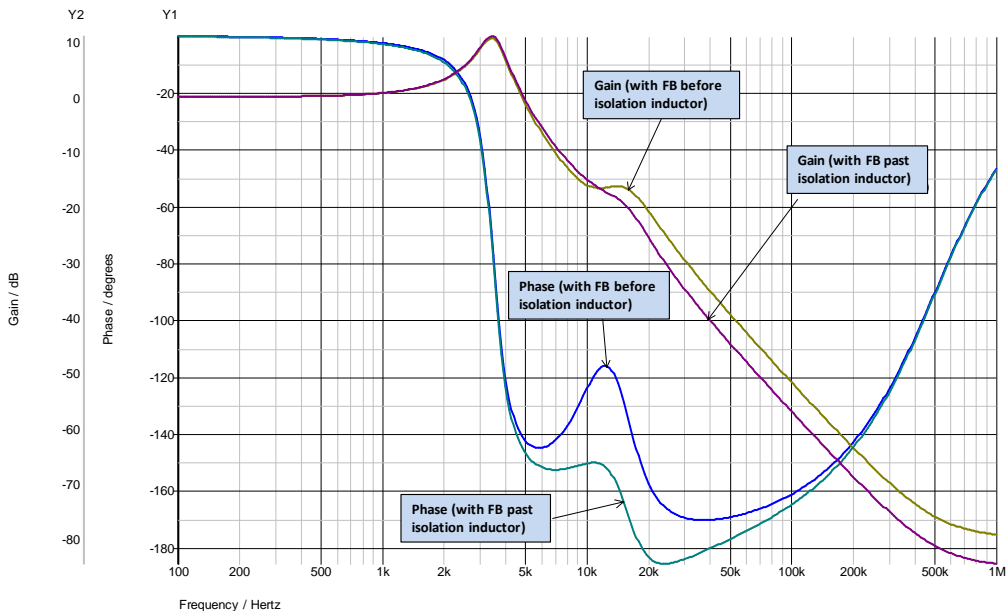


Figure C-8 compares power stage response with compensation taken before and after the *zIsolation\_P* Provider isolation inductor. As can be seen in in Figure C-9, the phase and gain impacts are larger. A pole is created by the *zIsolation\_P* inductor that forces a phase reduction beyond 180 degrees which will have to be accounted for in the compensation network.

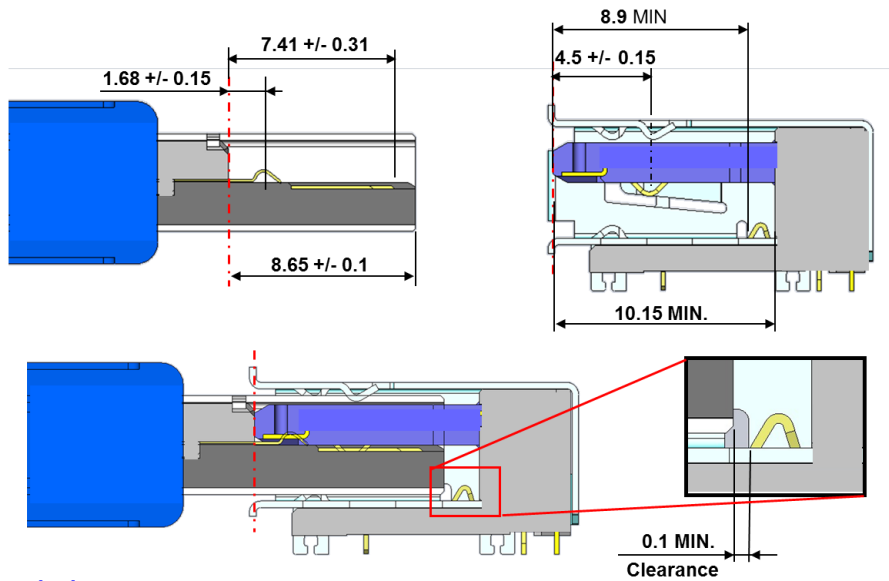
Figure C-9 Simplified Small Signal AC Model (Feedback before and after Inductor zIsolation\_P)



## D. Standard-A Mating Illustrations

The following illustrations show the mating configurations between PD and non-PD Standard-A connectors. All dimensions are in mm.

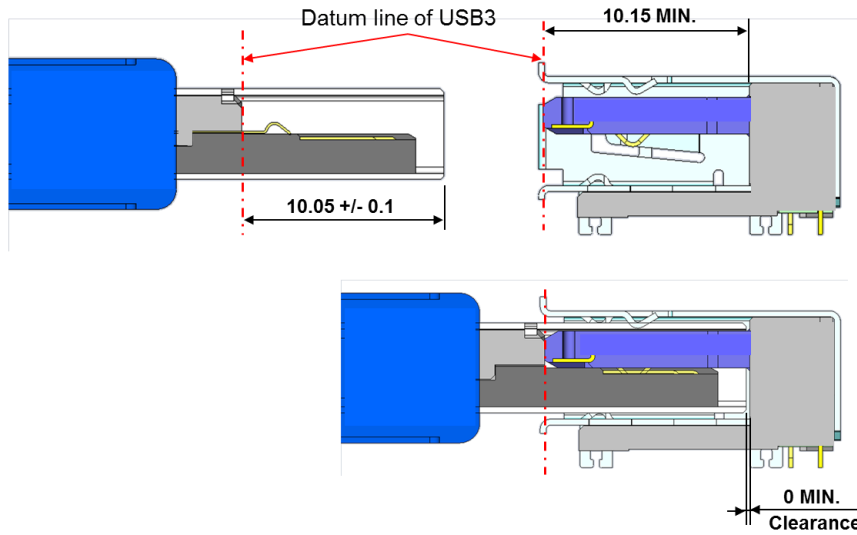
Figure D-1 USB 3.1 Standard-A Plug with USB 2.0 PD or 3.1 PD Standard-A Receptacle



### Conclusion

- 1) Plug does not contact to detect pins, then is identified as legacy USB3 plug.
- 2) The minimum clearance would be 0.1mm in the worst case which includes angled insertion case.

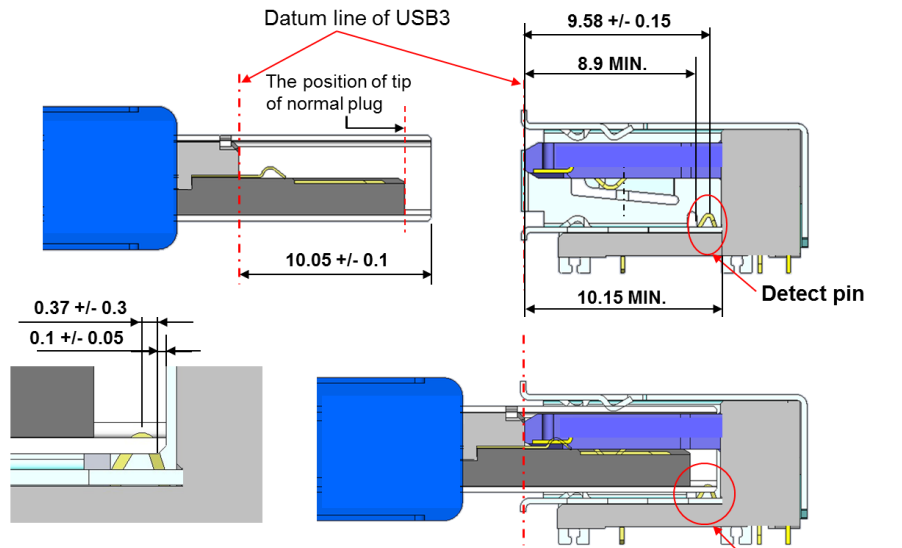
Figure D-2 USB 3.1 PD Standard-A Plug with USB 2.0 or 3.1 Standard-A Receptacle



**Conclusion**

- 1) Legacy receptacle does not have detect pins, so PD plug will work as normal USB3 plug.
- 2) Wipe length will be same as legacy USB combination since there would be clearance (0 MIN.) between receptacle and plug after completing mating.

Figure D-3 USB 2.0 PD or 3.1 PD Standard-A plug with USB 2.0 PD or 3.1 PD Standard-A receptacle

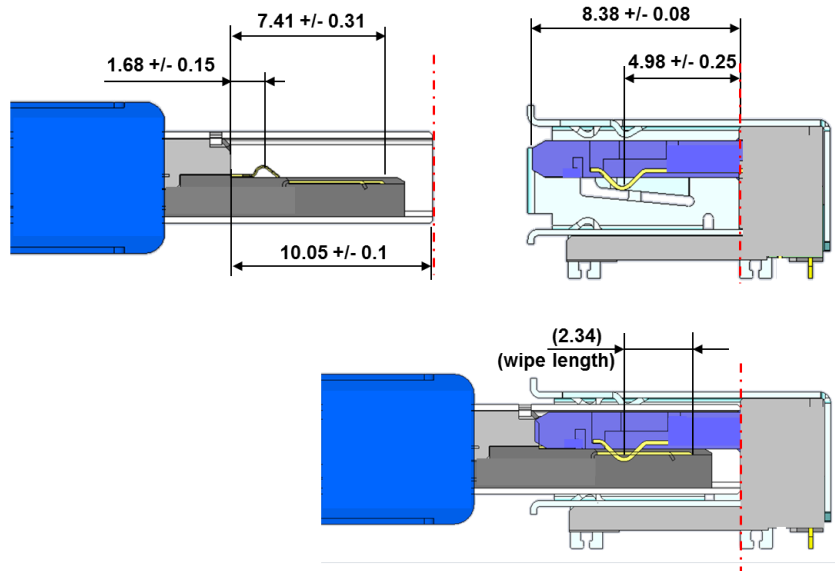


**Conclusion**

- 1) Connectors are identified as PD connectors due to connecting with detect pin.
- 2) Wipe length of detect pin is  $0.37 \pm 0.3$ mm in normal mating condition.

Tip of plug shell will contact detect pin.

Figure D-4 USB 2.0 PD or 3.1 PD Standard-A Plug with USB 2.0 Standard-A Receptacle

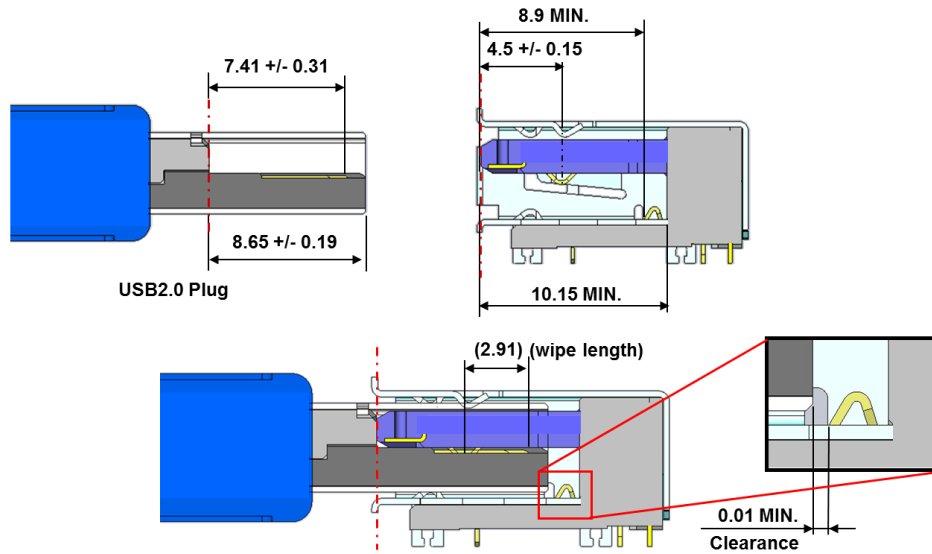


**Conclusion**

USB2 legacy receptacle does not have detect pins, so PD plug will work as normal USB2 plug.



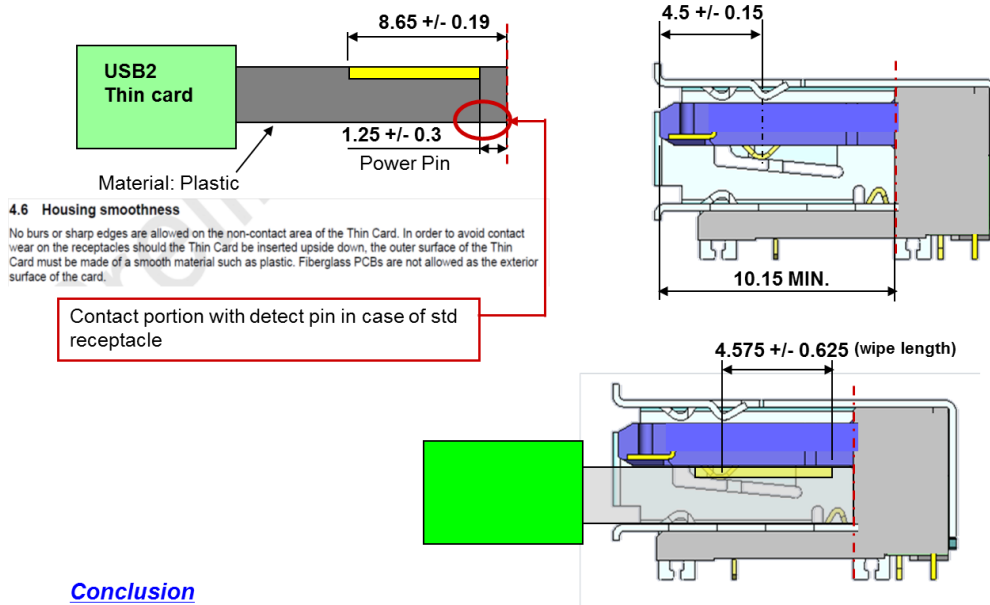
Figure D-5 USB 2.0 Standard-A Plug with USB 2.0 or USB3.1 PD Standard-A Receptacle



**Conclusion**

- 1) USB2 legacy plug does not contact to detect pins, so plug is identified as legacy plug then perform as USB2.
- 2) There would be 0.33mm clearance. The minimum clearance would be 0.01mm in the worst case which includes angled insertion case.

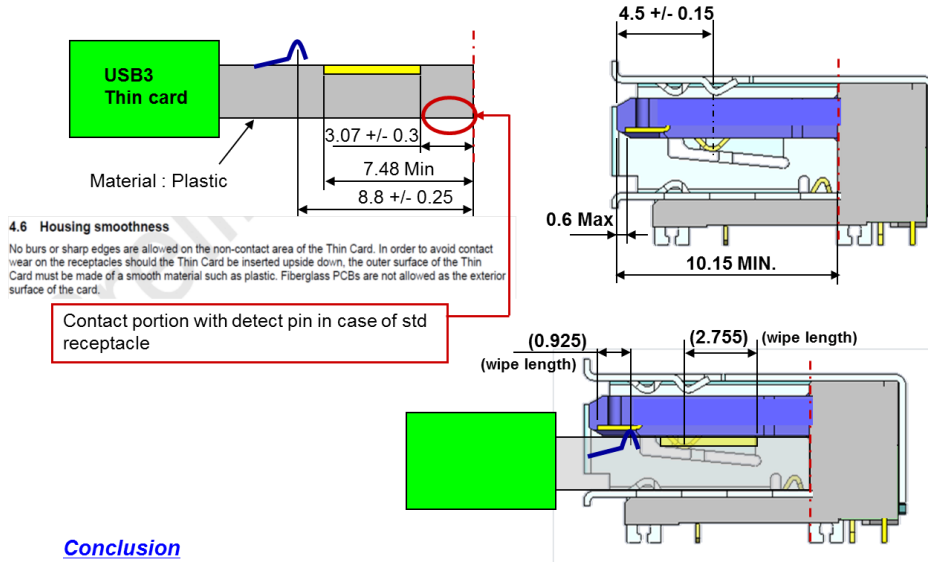
Figure D-6 USB 2.0 Thin Card with USB 2.0 PD or 3.1 PD Standard-A Receptacle



**Conclusion**

- 1) USB2 thin card would be identified as thin card due to discontinuity with detect pin.
- 2) Detect pin would be displaced in case of standard receptacle. However it will not make electric connection as long as contact portion of thin card is insulator.

Figure D-7 USB 3.1 Thin Card with USB 2.0 PD or 3.1 PD Standard-A Receptacle



**Conclusion**

- 1) USB3 thin card would be identified as thin card due to discontinuity with detect pin.
- 2) Detect pin would be displaced in case of standard receptacle. However it will not make electric connection as long as contact portion of thin card is insulator.

## E. Physical Layer Informative Material

This section contains informative material about the Physical Layer.

## E.1 Squelch Budget

Figure E-1 Squelch Budget

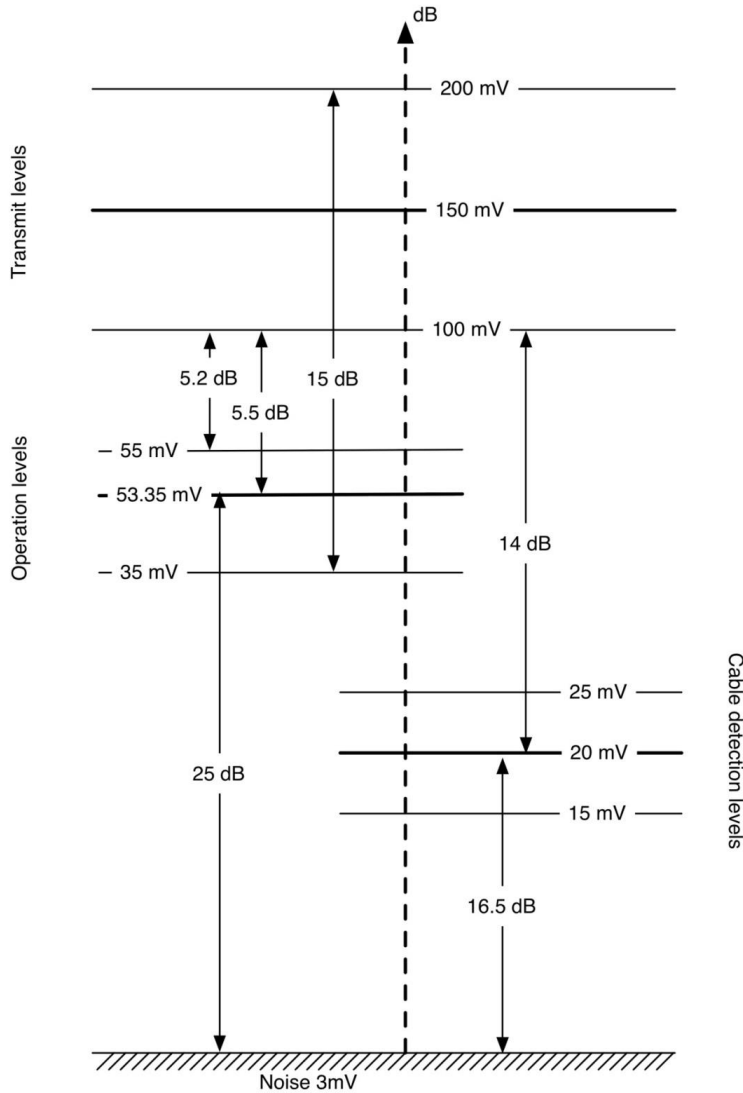


Figure E-1 shows the signal levels in mV (rms) needed to illustrate the squelch budget for normal operation (*vSquelchOperating*) and cable-type detection operation (*vSquelchDetecting*). The transmission level of the signal is between 100mV and 200mV, or at least 26.5dB above a 3mV noise floor. During normal operation the nominal SNR is 25dB, packets with a signal level below 55mV may be discarded, and packets with a signal level below 35mV are discarded. This allows for at least 5.2dB of attenuation in the channel and at most 15dB. During plug-type detection

the nominal SNR is 16.5dB, a signal level below 25mV may be discarded, and a signal level below 15mV must be discarded. This allows for at least 12dB of attenuation in the channel and at most 22.5dB.

## F. PD Message Sequence Examples

The following examples are intended to show how the Device Policy Manager may operate and the sequence of Power Delivery messaging which will result. The aim of this section is to inform implementer's how some of the mechanisms detailed in this specification may be applied; it does not contain any normative requirements.

In these illustrations all A-Ports are assumed to implement the optional cold-socket functionality.  $V_{BUS}$  is only applied when an A-plug is inserted. In all cases Providers indicate all of the power they have available in their Capabilities and Consumers only take the power they need and give back what they cannot use. In the case of Hubs this means only taking sufficient power for the ports with A-plugs inserted.

All ports are assumed to be Enhanced SuperSpeed capable, with a default operating voltage of 5V and a unit load of 150mA. This 0.75W is assumed to be enough power to enable an externally powered device to maintain communication over USB and is enough to allow such a device to enumerate but not operate until more power is negotiated.

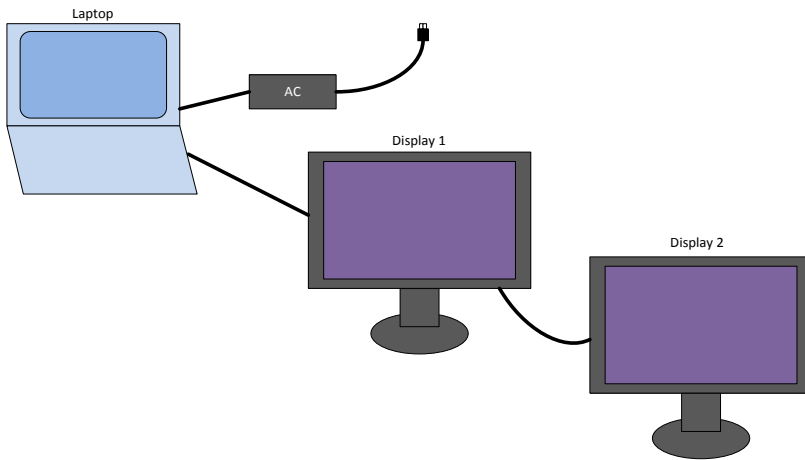
Although the Hubs in these illustrations support Power Delivery on both their UFPs and DFPs this is only one possible Hub implementation.

HDDs are assumed to spin up immediately after they are attached. This follows the typical operation of current systems.

Ideal power transmission is assumed so that there are no power losses through a device; in practice these would need to be taken into account when requesting power.

### F.1 External power is supplied downstream

Figure F-1 External Power supplied downstream



Configuration:

1. Laptop with an AC supply. AC supply provides sufficient power to charge the laptop and, in addition, to provide up to 60W downstream via its Profile 4 Enhanced SuperSpeed Port.

2. Display 1 and Display 2 each require 30W to operate. Display 1 contains a Hub allowing Display 2 to be connected to Display 1. In USB suspend each Display will power down but can maintain USB connection using the PD power provided.

**Table F-1 External power is supplied downstream**

Step	Laptop	Display 1	Display 2	Device Policy Manager	PD Power (W)
<b>Display 1</b>					
1	Connected to wall supply	Unattached	Unattached		0
2	Display 1 attached, V <sub>BUS</sub> powered.	Attached, drawing 5V@150mA.	Unattached		0.75
3	Set of Source Capabilities sent including: 5V@2A (10W), 12V@3A (36W) and 20V@3A (60W). The externally powered and USB suspend bits are set.	Source Capabilities received	Unattached	Laptop determines its Source Capabilities based on its needs and the presence of a wall supply.	0.75
4	Request received	Requests 20V@1.5A (30W) from laptop	Unattached	Display 1 knows it needs 20v@1.5A (30W) for its own operation, evaluates the supplied capabilities and determines that this is available.	0.75
5	Sends Accept	Accept received	Unattached	Waiting for PS_RDY before drawing additional power.	0.75
6	Sends PS_RDY	PS_RDY received. Starts drawing 20V@1.5A. Display 1 turns on and starts operating.	Unattached	Laptop evaluates the request, finds that it can meet this and so sends an accept.	30
<b>Display 2</b>					
7	Powering Display 1	Detects attach	Attached, no V <sub>BUS</sub>		30



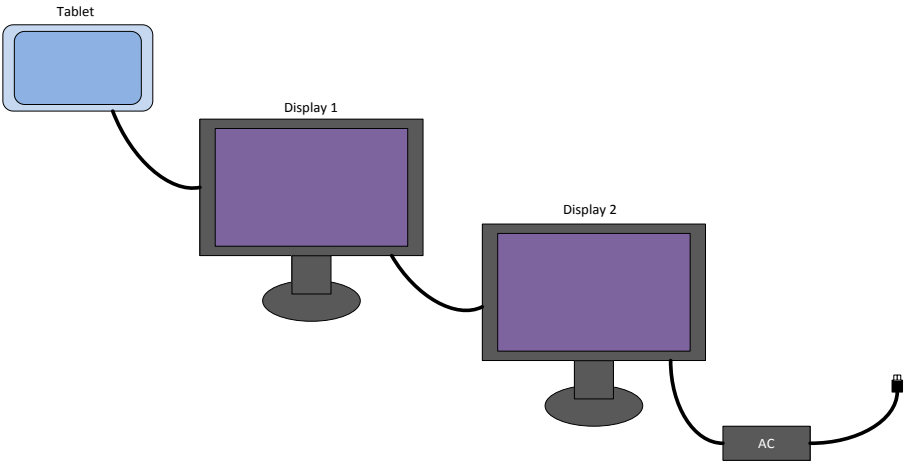
Step	Laptop	Display 1	Display 2	Device Policy Manager	PD Power (W)
8	Request received	Display 1 requests 20V@1.73A (34.6W) from Laptop.	Attached, no V <sub>BUS</sub>	Display 1 detects attach and requests additional 4.5W of power for USB 3.1 Port.	30
9	Sends Accept	Accept received.	Attached, no V <sub>BUS</sub>		34.6
10	Sends PS_RDY	PS_RDY received	Attached, no V <sub>BUS</sub>		
11		Powers V <sub>BUS</sub>	Attached, drawing 5V@150mA.		34.6
12		Sends out Source Capabilities including: 5V@0.9A to Display 2. The externally powered and USB suspend bits are set.	Source Capabilities received	Display 1 has 4.5W to allocate to Display 1. This is offered as a standard USB 3.1 Port.	34.6
13		Request received	Display 2 requests 5V@0.15A but indicates a Capability Mismatch. Display 2 remains off.	Display 2 decides it can manage to run its USB/PD function with 1 unit load but needs more power to function as a display.	34.6
14		Sends Accept	Accept received		34.6
15		Sends PS_RDY	PS_RDY received	Display 2 indicates a capability mismatch to the user.	34.6
16		Get Sink Capabilities sent	Get Sink Capabilities received	Display 1 needs to assess the capability mismatch by first determining what Display 2 actually needs.	34.6
17		Sink Capabilities received	Display 2 returns Sink Capabilities indicating operation at 20V@1.5A.		34.6

Step	Laptop	Display 1	Display 2	Device Policy Manager	PD Power (W)
18	Request received	Display 1 requests 20V@3A (60W) from Laptop.		Display1 now knows what Display 2 needs and requests the additional power from the laptop.	34.6
19	Sends Accept	Accept received.			34.6
20	Sends PS_RDY	PS_RDY received		An additional 30W is now available to Display 1 to offer to Display 2.	60
21		Sends out Source Capabilities including: 5V@0.9A and 20V@1.5A to Display 2. The externally powered and USB suspend bits are set.	Source Capabilities received	Now that Display 1 can power Display 2 correctly this power is offered by Display 1 via a new capabilities Message.	60
22		Request received	Display 2 requests 20V@1.5A.		60
23		Sends Accept	Accept received	Display 1 determines that the request by Display 2 is within the offered capabilities so the request is accepted.	60
24		Sends PS_RDY. Drawing 20V@3A from laptop.	PS_RDY received. Starts drawing 20V@1.5A, turns on and starts operating.	Display 2 now has the power it needs and can start working.	60
<b>USB Suspend</b>					
25	Laptop OS goes into suspend (S3), $V_{BUS}$ remains on but USB bus is also suspended.	Display 1 turns off but draws 50mW, 25mW to maintain PDU Hub functions. The additional 25mW is used to supply the Port used by Display 2.	Display 2 turns off but draws 25mW to maintain USB/PD functions.	No changes in Contract. This is a power reduction purely based on the USB state.	60

Step	Laptop	Display 1	Display 2	Device Policy Manager	PD Power (W)
26	Laptop OS wakes up. USB is woken up.	Display 1 turns on and returns to drawing 20V@3A.	Display 2 turns on and returns to drawing 20V@1.5A.	No changes in PD Contract. This purely relates to USB bus state.	60

F.2 External power is supplied upstream

Figure F-2 External Power supplied upstream



Configuration:

1. Tablet with no AC supply. Tablet is a USB host and can use 12V@0.2A (1W) during normal operation and up to 12V@1A (12W) in order to charge.
2. Display 1 and Display 2 each require 30W to operate. Display 1 contains a Hub allowing Display 2 to be connected to Display 1. In USB suspend each Display will power down but can maintain USB connection using the PD power provided.
3. Display 2 has an AC supply connected. AC supply provides sufficient power to power Display 2 and, in addition, to provide up to 60W upstream via a Profile 4 Port.

Table F-2 External power is supplied downstream

Step	Tablet	Display 1	Display 2	Device Policy Manager	PD Power (W)
<b>Display 1 - Dead Battery</b>					
1	Unattached	Unattached	Connected to the wall supply and outputs vSafeDB every 10-15s on its UFP		0

Step	Tablet	Display 1	Display 2	Device Policy Manager	PD Power (W)
2		Attached to Display 2	Display 1 attached		0
3		Dead battery negotiation	Dead battery negotiation		0
4		Attached to Display 2, drawing 5V@150mA (0.75W)	Providing 1 unit load to Display 1.		0.75
5		Source Capabilities received	Display2 sends out a set of capabilities including: 5V@2A (10W), 12V@3A (36W) and 20V@3A (60W). The externally powered and USB suspend bits are set.	Based on the capabilities of the wall supply and its own needs Display 2 calculates what it can offer upstream.	0.75
6		Display 1 requests 20V@1.5A (30W) from Display 2.	Request received	Display 1 knows it needs 30W to operate so it requests this amount.	0.75
7		Accept received	Sends Accept	Display 2 accepts the offer since it is within its capabilities.	0.75
8		PS_RDY received. Display 1 starts drawing power and turns on.	Sends PS_RDY	Display 2 indicates its power supply is ready to offer the power.	30
<b>Tablet – Power Role Swap</b>					
9	Tablet is attached to Display 1.	Attached, V <sub>BUS</sub> powered.			30
10	Tablet sends out a set of capabilities including: 5V@0.5A (2.5W). The externally powered bit cleared and USB suspend bit set.	Capabilities received			30

Step	Tablet	Display 1	Display 2	Device Policy Manager	PD Power (W)
11	Request received	Display 1 requests 5V@0A from the Tablet. The externally powered and Dual-Role Power bits are set.		Display 1 has external power providing everything it needs so it does not request any more.	30
12	Sends Accept	Accept received.		No power has been requested from the Tablet so the tablet has no reason to Reject this.	30
13	Sends PS_RDY	PS_RDY received.		Table completes the Explicit Contract by sending PS_RDY.	30
14	Get Sink Capabilities received.	Sends Get Sink Capabilities		Display 1 has access to an external supply so it needs to check whether the Tablet upstream, which has no external supply, could use some power. Display 1 also knows that there is excess capacity, based on the last capabilities it received, which it is not currently using from Display 2.	30
15	The Tablet returns Sink Capabilities indicating that it is a Dual-Role and that it can use 12V@0.2A (2.4W) as a Sink.	Sink Capabilities received			30
16		Display 1 requests 20V@1.62A (32.4W) from Display 2.	Request received		30
17		Accept received	Sends Accept	Request is within the available power so Display 2 sends an accept.	30

Step	Tablet	Display 1	Display 2	Device Policy Manager	PD Power (W)
18		PS_RDY received	Sends PS_RDY	Display 2 indicates that the power supply is ready to supply the power.	32.4
19	PR_Swap received	Requests PR_Swap from Tablet.		Display 1 now offers to provide power to the Tablet by initiating a Power Role Swap.	32.4
20	Accept sent. Tablet turns off its V <sub>BUS</sub> supply.	Accept received.		Tablet is happy to accept a Power Role Swap from any device offering it power.	32.4
21	Send PS_RDY	PS_RDY received. Display 1 turns on its V <sub>BUS</sub> supply		Tablet indicates that its supply has been turned off.	32.4
22	PS_RDY received.	PS_RDY sent.		Display 1 indicates that its power supply is ready so the Tablet starts drawing power.	32.4
23	Source Capabilities received	Display 1 sends out a set of capabilities to the Tablet including: 5V@0.48A (2.4W), 12V@0.2A (2.4W) and 20V@0.12 (2.4W). The externally powered and USB suspend bits are set.			32.4
24	The Tablet requests 12V@0.2A.	Request received.		Tablet can now request the power it needs.	32.4
25	Accept received	Accept sent		Power is within the capabilities of Display 1 so it accepts the request.	32.4

Step	Tablet	Display 1	Display 2	Device Policy Manager	PD Power (W)
26	PS_RDY received. The Tablet starts drawing 12V@0.2A.	PS_RDY sent		Display 1 indicates that its power supply is ready so the tablet starts drawing the power.	32.4
<b>Tablet – Charge</b>					
27	Tablet requests 12V@0.2A (2.4W) from Display 1. The Tablet needs to charge and so sets the Capability Mismatch bit and the No USB Suspend bit.	Request received.		Tablet needs to charge but the power offered is not sufficient. Since Display 1 claims to have an external supply the Tablet will try to get more power using the Capability Mismatch Flag.	32.4
28	Accept received	Accept sent		A valid request has been made so Display 1 accepts the request.	32.4
29	PS_RDY received	PS_RDY received		Tablet indicates a capability mismatch to the user.	32.4
30	Get Sink Capabilities received.	Get Sink Capabilities sent		Due to the Capability Mismatch Flag Display 1 requests Sink Capabilities from the Tablet.	32.4
31	The Tablet returns Sink capabilities containing: 12V@1A (12W). The externally powered bit is cleared.	Sink Capabilities received			32.4

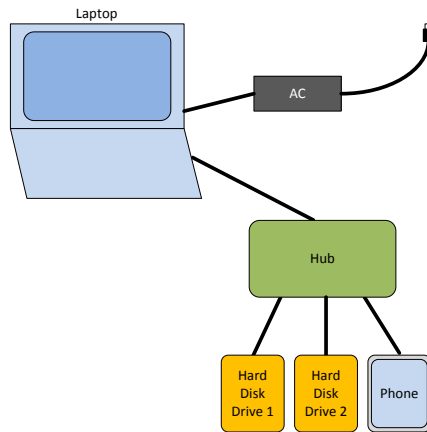
Step	Tablet	Display 1	Display 2	Device Policy Manager	PD Power (W)
32		Display 1 requests 20V@2.1A (42W) from Display 2. The No Suspend Bit is set to reflect the request from the Tablet.	Request received	Since the Tablet requires an additional 12W of power, and Display 1 knows that this is available from Display 2 based on the last Capabilities received so it requests it. In addition the Request from the Tablet indicated that it wanted No Suspend so this is reflected upwards.	32.4
33		Accept received	Sends Accept	Display 2 has 42W available and so accepts the request.	42
34		PS_RDY received	Sends PS_RDY	Display 2 completes the Explicit Contract but at this point has not accepted that power can be drawn during suspend.	42
35		Source Capabilities received	Display2 sends out a new set of capabilities including: 5V@2A (10W), 12V@3A (36W) and 20V@3A (60W). The externally powered and USB suspend bits is now set to zero.	Based on the capabilities of the wall supply and its own needs Display 2 calculates what it can offer upstream. It decides that it can continue to supply the power even during USB suspend and so resets the USB suspend bit.	42



Step	Tablet	Display 1	Display 2	Device Policy Manager	PD Power (W)
36		Display 1 requests 20V@2.1A (42W) from Display 2. The No Suspend Bit is set to reflect the request from the Tablet.	Request received	Display 1 repeats its request since a new set of Capabilities have been sent out.	42
37		Accept received	Sends Accept	Display 2 has 42W available, even during suspend, and so accepts the request.	42
38		PS_RDY received	Sends PS_RDY	Display 2 completes the Explicit Contract.	42
39	Capabilities received	Display 1 sends out a set of capabilities to the Tablet including: 5V@0.5A (2.5W), 12V@1A (12W) and 20V@0.6A (12W). The externally powered bit is set and USB suspend bit is cleared.		Display 1 now has the additional power available and so offers this to the Tablet.	42
40	Tablet requests 12V@1A (12W) from Display 1.	Request received.		Tablet is being offered the power it needs to charge and so the Tablet requests this from Display 1.	42
41	Accept received	Sends Accept		Request is within the available Display 1's available power and so it accepts the request.	42
42	PS_RDY received. Tablet starts drawing 12V@1A from Display 1 and starts to charge.	Sends PS_RDY		Display 1 indicates its supply is ready to supply power.	42

### F.3 Giving back power

Figure F-3 Giving Back Power



Configuration:

1. Laptop with an AC supply. AC supply provides sufficient power to charge the laptop and, in addition, to provide up to 60W downstream via a Profile 4 Port.
2. A Hub with 4 downstream ports which initially provides 1 unit load (150mA) per Port plus 1 unit load for its internal functions.
3. Two Hard Disk Drives both of which require 20V@0.5A (10W) to spin up and 20V@0.25A (5W) while being accessed.
4. A phone which uses 5V@2A (10W) to charge and can give back all of this power when requested.

Table F-3 Giving back power

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
<b>Connect Hub</b>					
1	Connected to wall supply	Unattached	Unattached		Default
2	Hub is attached	Attached, $V_{BUS}$ powered			Default

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
3	Laptop sends out a set of capabilities including: 5V@2A (10W), 12V@3A (36W), 20V@3A (60W). The externally powered and USB suspend bits are set.	Source Capabilities received		Laptop sends out details of all available power via external supply	Default
4	The Hub requests 5V@0.15A. This is the power for the Hubs internal operation.	Request received		Hub needs 1 unit load for its own operation and so requests this amount.	Default
5	Send Accept	Accept received		Laptop evaluates request and it is within its available power.	0.75
6	Send PS_RDY	PS_RDY received. Starts to draw 5V@0.15A		Laptop indicates that its power supply is ready.	0.75
<b>Connect Hard Disk Drive 1</b>					
7		Attached detected.	Hard Disk Drive 1 is attached to one of the downstream ports of the Hub.		0.75
8	Request received	The Hub requests 5V@0.3A (1.5W) from the Laptop.		Hub needs 0.75W for its own operation plus 0.75W for USB communication on one Port.	0.75
9	Accept sent	Accept received		Request is within available power so the laptop accepts.	1.5
10	PS_RDY sent	PS_RDY received		Laptop indicates that its power supply is ready	1.5

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
11		Hub turns on $V_{BUS}$ and sends out a set of capabilities to Hard Disk Drive 1 including: 5V@0.15A. The externally powered and USB suspend bits are set.	Source Capabilities received		1.5
12		Request received	Hard Disk Drive 1 requests 5V@0.15A from the Hub.	Hard Disk Drive 1 only needs one unit load when not operating so requests this.	1.5
13		Accept sent	Accept received	Request is within available power so the Hub accepts.	1.5
14		PS_RDY sent	PS_RDY received. The Hard Disk Drive starts drawing 1 unit load 5V@0.15A.	Laptop indicates its power supply is ready and the Hard Disk Drive starts drawing power.	1.5
<b>Hard Disk Drive 1 spin up</b>					
15		Request received	Hard Disk Drive 1 requests 5V@0.15A from the Hub but sets the Capability Mismatch bit.	Hard Disk Drive 1 needs 20V@0.5A to spin up but this is not available so it re-requests the available power flagging a capability mismatch.	1.5
16		Accept sent	Accept received	Request is within available power so the Hub accepts.	1.5
17		PS_RDY sent	PS_RDY received	Hard Disk Drive 1 indicates a capability mismatch to the user.	1.5

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
18		The Hub requests the Sink Capabilities from Hard Disk Drive 1.	Get Sink Capabilities received	Due to the Capability Mismatch the Hub needs to determine what Hard Disk Drive 1 actually needs	1.5
19		Sink Capabilities received	Hard Disk Drive 1 returns capabilities indicating that it requires 20V@0.5A.		1.5
20	Request received	The Hub requests 20V@0.54A (10.8W) from the Laptop.		The Hub evaluates that it now needs 0.75W for the Hub and 10W for Hard Disk Drive 1.	1.5
21	Accept sent	Accept received		Power request from the Hub is within the Laptop's capabilities so the Laptop accepts the request.	10.8
22	PS_RDY sent	PS_RDY received		Laptop completes the Explicit Contract.	10.8
23		Hub sends out a set of capabilities to Hard Disk Drive 1 including: 5V@0.5A and 20V@0.5A. The externally powered and USB suspend bits are set.	Source Capabilities received	Hub now offers Hard Disk Drive 1 what it needs.	10.8
24		Request received	Hard Disk Drive 1 requests 20V@0.5A operating current and indicates 20V@0.5A maximum current.	Hard Disk Drive 1 is operating at its maximum current to spin up so sets operating current = maximum current.	10.8
25		Accept sent	Accept received	Request is within the Hubs capabilities so it accepts.	10.8

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
26		PS_RDY sent	PS_RDY received. Hard Disk Drive 1 starts to draw 20V@0.5A and spins up.	Hub indicates its power supply is ready so Hard Disk Drive 1 starts to draw power.	10.8
27		Request received	Once spun up Hard Disk Drive 1 requests 20V@0.25A operating current and 20V@0.5A maximum current.	Hard Disk Drive 1 is operating at a lower current so sets operating current < maximum current.	10.8
28		Accept sent	Accept received	The Hub will maintain a Power Reserve of 20V@0.25A (5W) for Hard Disk Drive 1 in addition to the 20V@0.25A (5W) it is currently using.	10.8
29		PS_RDY sent	PS_RDY received	Hub completes the Explicit Contract.	10.8
<b>Hard Disk Drive 2 spin up</b>					
30		Attach detected	Hard Disk Drive 2 is attached to one of the downstream ports of the Hub.		10.8
31	Request received	The Hub requests 20V@0.58A (11.6W) from the Laptop.		The Hub needs 0.75W for itself, 0.75W for USB communication on one Port, 5W for Hard Disk Drive 1 operation and 5W for the Power Reserve.	10.8

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
32	Accept sent	Accept received		Power request from the Hub is within the Laptop's capabilities so it accepts the request.	10.8
33	PS_RDY sent	PS_RDY received		Laptop indicates its power supply is ready.	11.6
34		Hub sends out a set of capabilities to Hard Disk Drive 2 including: 5V@0.15A. The externally powered and USB suspend bits are set.	Source Capabilities received by Hard Disk Drive 2	Hub offers Hard Disk Drive 2 enough power to enumerate.	11.6
35		Request received	Hard Disk Drive 2 requests 5V@0.15A from the Hub.		11.6
36		Accept sent to Hard Disk Drive 2	Accept received by Hard Disk Drive 2	Request is within available capabilities so the Hub accepts	11.6
37		PS_RDY sent to Hard Disk Drive 2.	PS_RDY received. Hard Disk Drive 2 starts drawing 5V@0.15A.	Hard Disk Drive 2 takes the power that it needs	11.6
<b>Phone charge</b>					
38		Attach detected	The phone is attached to one of the downstream ports of the Hub.		11.6

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
39	Request received	The Hub Requests 20V@0.62A (12.4W) from the Laptop.		The Hub needs 0.75W for itself, 1.5W for USB communications on two ports (Hard Disk Drive 1 and the Phone), 5W for Hard Disk Drive 1 operation and 5W for the Power Reserve.	11.6
40	Accept sent	Accept received		Request is within available capabilities so the Laptop accepts	12.4
41	PS_RDY sent	PS_RDY received		Laptop indicates that its power supply is ready.	12.4
42		The Hub powers $V_{BUS}$ and sends out a set of capabilities to the Phone including: 5V@0.15A. The externally powered and USB suspend bits are set.	Source Capabilities received by the Phone	The Hub offers the Phone 1 unit load to enumerate.	12.4
43		Request received from the Phone	The Phone requests 5V@0.15A from the Hub but sets the Capability Mismatch bit.	The Phone would like to charge and so indicates this fact through the Capability Mismatch bit.	12.4
44		Accept sent	Accept received	Request is within available capabilities so the Hub accepts	12.4
45		PS_RDY sent	PS_RDY received	Hub indicates that its power supply is ready	12.4



Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
46		The Hub requests the Sink Capabilities from the phone.	Get Sink Capabilities received by the Phone	Due to the Capability Mismatch the Hub needs to determine what the Phone actually needs	12.4
47		Sink Capabilities received from the Phone	The Phone returns capabilities indicating that it requires 5V@2A.	Phone returns the Capabilities it needs to charge	12.4
48	Request received	The Hub Requests 20V@1.12A (22.4W) from the Laptop.		The Hub needs 0.75W for itself, 0.75W for Hard Disk Drive 2, 10W for the phone, 5W for Hard Disk Drive 1 operation and 5W for the Power Reserve.	12.4
49	Accept sent	Accept received		Request is within available capabilities so the Laptop accepts	12.4
50	PS_RDY sent	PS_RDY received		Laptop indicates that its power supply is ready.	22.4
51		The Hub sends out a set of capabilities to the Phone including: 5V@2A. The externally powered and USB suspend bits are set.	Source Capabilities received by the Phone	The Hub now has the power that the Phone needs and so sends out a new set of Capabilities.	22.4
52		Request received from the Phone	The Phone requests 5V@2A from the Hub and sets the No USB Suspend bit since it needs to charge constantly. It sets the GiveBack flag and sets the Minimum Operating Current to 5V@0A.	The Phone requests the power it needs to charge. It asks for the USB Suspend requirement to be removed.	22.4

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
53		Accept sent to the Phone	Accept received by the Phone		22.4
54		PS_RDY sent to the phone.	PS_RDY received by the phone. Phone starts to charge 5V@2A but has to follow USB Suspend rules		22.4
55	Request received	The Hub Requests 20V@0.83A (16.6W) from the Laptop but sets the No USB Suspend bit.		The Hub needs 0.75W for itself, 0.75W for Hard Disk Drive 2, 10W for the phone (includes the Power Reserve of 5W), and 5W for Hard Disk Drive 1 operation. It requests for USB Suspend rule to be removed.	22.4
56	Accept sent	Accept received		Request is within available capabilities so the Laptop accepts. Note that the request for No Suspend has not been acted on by the Laptop. USB Suspend rules apply until the Laptop sends out new Source Capabilities with the USB Suspend bit cleared.	22.4
57	PS_RDY sent	PS_RDY received		Laptop indicates that its power supply is ready.	16.6

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
<b>Hard Disk Drive 2 spin up</b>					
58		Request received from Hard Disk Drive 2	Hard Disk Drive 2 requests 5V@0.15A from the Hub but sets the Capability Mismatch bit.	Hard Disk Drive 2 needs more power to spin up and so indicates a Capability Mismatch	16.6
59		Accept sent	Accept received	The request is within its capabilities so the Hub accepts.	16.6
60		PS_RDY sent	PS_RDY received	The Hub indicates that its power supply is ready.	16.6
61		The Hub requests the Sink Capabilities from Hard Disk Drive 2.	Get Sink Capabilities received by Hard Disk Drive 2	Due to the Capability Mismatch the Hub has to determine what Hard Disk Drive 2 needs	16.6
62		Sink Capabilities received	Hard Disk Drive 2 returns capabilities indicating that it requires 20V@0.5A maximum current.		16.6
63		The Hub instructs the Phone to Goto Minimum operation.	Goto Min received by the Phone	Hub assess that there is additional power available from the Phone and so tells it to Goto Min. In this case it is reallocating the Phone's Charging power as the Power Reserve for the Hard Disk Drives.	16.6
64			The Phone drops to zero current draw.		16.6
65		PD_RDY sent	PS_RDY received.	Hub indicates that its power supply has changed to the new level.	16.6

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
66	Request received	The Hub Requests 20V@1.04A (20.8W) from the Laptop		The Hub has an additional 10W from the Phone but needs 5W more to maintain its Power Reserve. The Hub needs 0.75W for itself, 10W for Hard Disk Drive 2, 5W for the Power Reserve, 5W for Hard Disk Drive 1 operation.	16.6
67	Accept sent	Accept received		Request is within available capabilities so the Laptop accepts.	16.6
68	PS_RDY sent	PS_RDY received		Laptop indicates that its power supply is ready.	20.8
69		Hub sends out a set of capabilities to Hard Disk Drive 2 including: 5V@0.5A and 20V@0.5A. The externally powered and USB suspend bits are set.	Source Capabilities received by Hard Disk Drive 2	The Hub now has the power that Hard Disk Drive 2 needs so it sends out new Capabilities.	20.8
70		Request received from Hard Disk Drive 2	Hard Disk Drive 2 requests 20V@0.5A operating current and 20V@0.5A.	Hard Disk Drive 2 requests what it needs to spin up.	20.8
71		Accept sent to Hard Disk Drive 2	Accept received by Hard Disk Drive 2	The Hub assesses that the request is within its Capabilities so it accepts.	20.8
72		PS_RDY sent.	PS_RDY sent. Hard Disk Drive 2 starts to draw 20V@0.5A and spins up.		20.8

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
73		Request received from Hard Disk Drive 2	Once spun up Hard Disk Drive 2 requests 20V@0.25A operating current and 20V@0.5A maximum current.	Hard Disk Drive 2 no longer needs the additional power so it gives back what it does not need.	20.8
74		Accept sent to Hard Disk Drive 2	Accept received by Hard Disk Drive 2	The Hub assesses that the request is within its Capabilities so it accepts.	20.8
75		PS_RDY sent to Hard Disk Drive 2.	PS_RDY received by Hard Disk Drive 2.	The Hub indicates that its power supply is ready.	20.8
76		The Hub sends out a set of capabilities to the Phone including: 5V@2A. The externally powered bit is set and the USB suspend bit is set.	Source Capabilities received by the Phone	The Hub now has the power available to charge the phone so it sends out new Capabilities	20.8
77		Request received from the Phone	The Phone requests 5V@2A operating current from the Hub and sets the No USB Suspend bit since it needs to charge constantly. It sets the GiveBack flag and sets the Minimum Operating Current to 5V@0A.	The Phone requests the power it needs to charge. It asks for the USB Suspend requirement to be removed.	20.8
78		Accept sent to the Phone	Accept received by the Phone	The Hub assesses that the request is within its Capabilities so it accepts but maintains USB Suspend rules.	20.8

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
79		PS_RDY sent to the Phone.	PS_RDY received by the Phone. The phone starts to draw 5V@2A but has to follow USB Suspend.	The Hub has allocated 0.75W for itself, 5W for Hard Disk Drive 2, 10W for the Phone (including 5W for the Power Reserve), and 5W for Hard Disk Drive 1 operation.	20.8

## G. VDM Command Examples

### G.1 Discover Identity Example

#### G.1.1 Discover Identity Command request

Table G-1 below shows the contents of the key fields in the Message Header and VDM header for an Initiator sending a *Discover Identity* Command request.

Table G-1 Discover Identity Command request from Initiator Example

Bit(s)	Field	Value
<b>Message Header</b>		
15	Reserved	0
14..12	<i>Number of Data Objects</i>	1 (VDM Header)
11..9	<i>MessageID</i>	0..7
8	<i>Port Power Role</i>	0 or 1
7..6	<i>Specification Revision</i>	01b
5..4	Reserved	0
3..0	<i>Message Type</i>	1111b (Vendor Defined Message)
<b>VDM Header</b>		
B31..16	Standard or Vendor ID (SVID)	0xFF00 ( <i>PD SID</i> )
B15	VDM Type	1 (Structured VDM)
B14..13	Structured VDM Version	00b (Version 1.0)
B12..11	Reserved	00b
B10..8	Object Position	000b
B7..6	Command Type	00b (Initiator)
B5	Reserved	0
B4..0	Command <sup>1</sup>	1 ( <i>Discover Identity</i> )

#### G.1.2 Discover Identity Command response – Active Cable

Table G-2 shows the contents of the key fields in the Message Header and VDM header for a Responder returning VDOs in response to a *Discover SVIDs* Command request. In this illustration, the responder is an active Gen2 cable which supports Modal Operation.

Table G-2 Discover Identity Command response from Active Cable Responder Example

Bit(s)	Field	Value
<b>Message Header</b>		
15	Reserved	0
14..12	<i>Number of Data Objects</i>	4 (VDM Header + ID Header <i>VDO</i> + Cer Stat VDO + Cable VDO)
11..9	<i>MessageID</i>	0..7
8	<i>Cable Plug</i>	<del>0 or 1</del>
7..6	<i>Specification Revision</i>	01b
5..4	Reserved	0
3..0	<i>Message Type</i>	1111b (Vendor Defined Message)

Bit(s)	Field	Value
<b>VDM Header</b>		
B31..16	Standard or Vendor ID (SVID)	0xFF00 ( <i>PD SID</i> )
B15	VDM Type	1 (Structured VDM)
B14..13	Structured VDM Version	00b (Version 1.0)
B12..11	Reserved	00b
B10..8	Object Position	000b
B7..6	Command Type	01b (Responder ACK)
B5	Reserved	0
B4..0	Command	2 ( <i>Discover Identity</i> )
<b>ID Header VDO</b>		
B31	Data Capable as USB Host	0 (not data capable as a Host)
B30	Data Capable as a USB Device	0 (not data capable as a Device)
B29..27	Product Type	100b (Active Cable)
B26	Modal Operation Supported	1 (supports Modes)
B25..16	Reserved. Shall be set to zero.	0
B15..0	16-bit unsigned integer. USB Vendor ID	USB-IF assigned VID for this cable vendor
<b>Cert Stat VDO</b>		
B31..20	Reserved, shall be set to zero.	0
B19..0	20-bit unsigned integer	USB-IF assigned TID for this cable
<b>Product VDO</b>		
B31..16	16-bit unsigned integer. USB Product ID	Product ID assigned by the cable vendor
B15..0	16-bit unsigned integer. bcdDevice	Device version assigned by the cable vendor
<b>Cable VDO (returned for Product Type "Active Cable")</b>		
B31..28	Cable HW Version	Cable HW version number (vendor defined)
B27..24	Cable Firmware Version	Cable FW version number (vendor defined)
B23..20	Reserved	0
B19..18	Type-C to Type-A/B/C	10b (Type-C)
B17	Type-C to Plug/Receptacle	0 (Plug)
B16..13	Cable Latency	0001b (<10ns (~1m))
B12..11	Cable Termination Type	11b (Both ends Active, VCONN required)
B10	SSTX1 Directionality Support	0 (Fixed)
B9	SSTX2 Directionality Support	0 (Fixed)
B8	SSRX1 Directionality Support	0 (Fixed)
B7	SSRX2 Directionality Support	0 (Fixed)
B6..5	V <sub>BUS</sub> Current Handling Capability	01b (3A)
B4	V <sub>BUS</sub> through cable	1 (Yes)
B3	SOP" controller present?	1 (SOP" controller present)
B2..0	USB Superspeed Signaling Support	010b ( <i>[USB 3.1]</i> Gen1 and Gen2)

### G.1.3 Discover Identity Command response – Hub

Table G-2 shows the contents of the key fields in the Message Header and VDM header for a Responder returning VDOs in response to a *Discover SVIDs* Command request. In this illustration, the responder is a Hub



Table G-3 Discover Identity Command response from **Active-CableHub** Responder Example

Bit(s)	Field	Value
<b>Message Header</b>		
15	Reserved	0
14..12	<i>Number of Data Objects</i>	4 (VDM Header + ID Header <b>VDO</b> + Cer Stat VDO + Product VDO)
11..9	<i>MessageID</i>	0..7
8	<i>Port Power Role</i>	0 or 1
7..6	<i>Specification Revision</i>	01b
5..4	Reserved	0
3..0	<i>Message Type</i>	1111b (Vendor Defined Message)
<b>VDM Header</b>		
B31..16	Standard or Vendor ID (SVID)	0xFF00 ( <i>PD SID</i> )
B15	VDM Type	1 (Structured VDM)
B14..13	Structured VDM Version	00b (Version 1.0)
B12..11	Reserved	00b
B10..8	Object Position	000b
B7..6	Command Type	01b (Responder ACK)
B5	Reserved	0
B4..0	Command	2 ( <i>Discover Identity</i> )
<b>ID Header <b>VDO</b></b>		
B31	Data Capable as USB Host	0 (not data capable as a Host)
B30	Data Capable as a USB Device	1 (data capable as a Device)
B29..27	Product Type	001b (Hub)
B26	Modal Operation Supported	0 (doesn't support Modes)
B25..16	Reserved. Shall be set to zero.	0
B15..0	16-bit unsigned integer. USB Vendor ID	USB-IF assigned VID for this hub vendor
<b>Cer Stat VDO</b>		
B31..20	Reserved, shall be set to zero.	0
B19..0	20-bit unsigned integer	USB-IF assigned TID for this hub
<b>Product VDO</b>		
B31..16	16-bit unsigned integer. USB Product ID	Product ID assigned by the hub vendor
B15..0	16-bit unsigned integer. bcdDevice	Device version assigned by the hub vendor

## G.2 Discover SVIDs Example

### G.2.1 Discover SVIDs Command request

Table G-4 below shows the contents of the key fields in the Message Header and VDM header for an Initiator sending a *Discover SVIDs* Command request.

Table G-4 Discover SVIDs Command request from Initiator Example

Bit(s)	Field	Value
<b>Message Header</b>		
15	Reserved	0
14..12	<i>Number of Data Objects</i>	1 (VDM Header)
11..9	<i>MessageID</i>	0..7
8	<i>Port Power Role</i>	0 or 1
7..6	<i>Specification Revision</i>	01b
5..4	Reserved	0
3..0	<i>Message Type</i>	1111b (Vendor Defined Message)
<b>VDM Header</b>		
B31..16	Standard or Vendor ID (SVID)	0xFF00 ( <i>PD SID</i> )
B15	VDM Type	1 (Structured VDM)
B14..13	Structured VDM Version	00b (Version 1.0)
B12..11	Reserved	00b
B10..8	Object Position	000b
B7..6	Command Type	00b (Initiator)
B5	Reserved	0
B4..0	Command <sup>1</sup>	2 ( <i>Discover SVIDs</i> )

### G.2.1 Discover SVIDs Command response

Table G-5 shows the contents of the key fields in the Message Header and VDM Header for a Responder returning VDOs in response to a *Discover SVIDs* Command request. In this illustration, the value 3 in the Message Header indicates that one VDO containing the supported SVIDs would be returned followed by a terminating VDO. Note that the last VDO returned (the terminator of the transmission) contains zero value SVIDs. If a SVID value is zero, it is ~~ignored~~not used.

Table G-5 Discover SVIDs Command response from Responder Example

Bit(s)	Field	Value
<b>Message Header</b>		
15	Reserved	0
14..12	<i>Number of Data Objects</i>	3 (VDM Header + 2*VDO)
11..9	<i>MessageID</i>	0..7
8	<i>Port Power Role</i>	0 or 1
7..6	<i>Specification Revision</i>	01b
5..4	Reserved	0
3..0	<i>Message Type</i>	1111b (Vendor Defined Message)
<b>VDM Header</b>		
B31..16	Standard or Vendor ID (SVID)	0xFF00 ( <i>PD SID</i> )

Bit(s)	Field	Value
B15	VDM Type	1 (Structured VDM)
B14..13	Structured VDM Version	00b (Version 1.0)
B12..11	Reserved	00b (Reserved)
B10..8	Object Position	000b
B7..6	Command Type	01b (Responder ACK)
B5	Reserved	0
B4..0	Command	2 ( <i>Discover SVIDs</i> )
<b>VDO 1</b>		
B31..16	SVID 0	SVID value
B15..0	SVID 1	SVID value
<b>VDO 2</b>		
B31..16	SVID 2	0x0000
B15..0	SVID 3	0x0000

## G.3 Discover Modes Example

### G.3.1 Discover Modes Command request

Table G-6 shows the contents of the key fields in the Message Header and VDM header for an Initiator sending a *Discover Modes* Command request. The Initiator of the *Discover Modes* Command sequence sends a Message Header with the *Number of Data Objects* field set to 1 followed by a VDM Header with the Command Type (B7..6) set to zero indicating the Command is from an Initiator and the Command (B4..0) is set to 3 indicating Mode discovery.

Table G-6 Discover Modes Command request from Initiator Example

Bit(s)	Field	Value
<b>Message Header</b>		
15	Reserved	0
14..12	<i>Number of Data Objects</i>	1 (VDM Header)
11..9	<i>MessageID</i>	0..7
8	<i>Port Power Role</i>	0 or 1
7..6	<i>Specification Revision</i>	01b
5..4	Reserved	0
3..0	<i>Message Type</i>	1111b (Vendor Defined Message)
<b>VDM Header</b>		
B31..16	Standard or Vendor ID (SVID)	SVID for which Modes are being requested
B15	VDM Type	1 (Structured VDM)
B14..13	Structured VDM Version	00b (Version 1.0)
B12..11	Reserved	00b
B10..8	Object Position	000b
B7..6	Command Type	00b (Initiator)
B5	Reserved	0
B4..0	Command <sup>1</sup>	3 ( <i>Discover Modes</i> )

### G.3.2 Discover Modes Command response

The Responder to the *Discover Modes* Command request returns a Message Header with the *Number of Data Objects* field set to a value of 1 to 7 (the actual value is the number of Mode objects plus one) followed by a VDM Header with the Message Source (B5) set to 1 indicating the Command is from a Responder and the Command (B4..0) set to 2 indicating the following objects describe the Modes the device supports. 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 Standard.

Table G-7 shows the contents of the key fields in the Message Header and VDM Header for a Responder returning VDOs in response to a *Discover Modes* Command request. In this illustration, the value 2 in the Message Header indicates that the device is returning one VDO describing the Mode it supports. It is possible for a Responder to report up to six different Modes.

Table G-7 Discover Modes Command response from Responder Example

Bit(s)	Field	Value
<b>Message Header</b>		
15	Reserved	0
14..12	<i>Number of Data Objects</i>	2 (VDM Header + 1 Mode VDO)

Bit(s)	Field	Value
11..9	<i>MessageID</i>	0..7
8	<i>Port Power Role</i>	0 or 1
7..6	<i>Specification Revision</i>	01b
5..4	Reserved	0
3..0	<i>Message Type</i>	1111b (Vendor Defined Message)
<b>VDM Header</b>		
B31..16	Standard or Vendor ID (SVID)	SVID for which Modes were requested
B15	VDM Type	1 (Structured VDM)
B14..13	Structured VDM Version	00b (Version 1.0)
B12..11	Reserved	00b
B10..8	Object Position	000b
B7..6	Command Type	01b (Responder ACK)
B5	Reserved	0
B4..0	Command	3 ( <i>Discover Modes</i> )
<b>Mode VDO</b>		
B31..0	Mode 1	Standard or Vendor defined Mode value

## G.4 Enter Mode Example

### G.4.1 Enter Mode Command request

The Initiator of the *Enter Mode* Command request sends a Message Header with the *Number of Data Objects* field set to 1 followed by a VDM Header with the Message Source (B5) set to zero indicating the Command is from an Initiator and the Command (B4..0) set to 4 to request the Responder to enter its mode of operation and sets the Object Position field to the desired functional VDO based on its offset as received during Discovery.

Table G-8 shows the contents of the key fields in the Message Header and VDM Header for an Initiator sending an *Enter Mode* Command request.

Table G-8 Enter Mode Command request from Initiator Example

Bit(s)	Field	Value
<b>Message Header</b>		
15	Reserved	0
14..12	<i>Number of Data Objects</i>	1 (VDM Header)
11..9	<i>MessageID</i>	0..7
8	<i>Port Power Role</i>	0 or 1
7..6	<i>Specification Revision</i>	01b
5..4	Reserved	0
3..0	<i>Message Type</i>	1111b (Vendor Defined Message)
<b>VDM Header</b>		
B31..16	Standard or Vendor ID (SVID)	SVID for the Mode being entered
B15	VDM Type	1 (Structured VDM)
B14..13	Structured VDM Version	00b (Version 1.0)
B12..11	Reserved	00b
B10..8	Object Position	001b (a one in this field indicates a request to enter the first Mode in list returned by Discover Modes)
B7..6	Command Type	00b (Initiator)
B5	Reserved	0
B4..0	<i>Command</i>	4 ( <i>Enter Mode</i> )

### G.4.2 Enter Mode Command response

The Responder that is the target of the *Enter Mode* Command request shall send a Message Header with the *Number of Data Objects* field set to a value of 1 followed by a VDM Header with the Command Source (B5) set to 1 indicating the response is from a Responder and the Command (B4..0) set to 4 indicating the Responder has entered the Mode and is ready to operate.

Table G-9 shows the contents of the key fields in the Message Header and VDM Header for a Responder sending an *Enter Mode* Command response with an ACK.

Table G-9 Enter Mode Command response from Responder Example

Bit(s)	Field	Value
<b>Message Header</b>		
15	Reserved	0
14..12	<i>Number of Data Objects</i>	1 (VDM Header)

Bit(s)	Field	Value
11..9	<i>MessageID</i>	0..7
8	<i>Port Power Role</i>	0 or 1
7..6	<i>Specification Revision</i>	01b
5..4	Reserved	0
3..0	<i>Message Type</i>	1111b (Vendor Defined Message)
<b>VDM Header</b>		
B31..16	Standard or Vendor ID (SVID)	SVID for the Mode entered
B15	VDM Type	1 (Structured VDM)
B14..13	Structured VDM Version	00b (Version 1.0)
B12..11	Reserved	00b
B10..8	Object Position	001b (offset of the Mode entered)
B7..6	Command Type	01b (Responder ACK)
B5	Reserved	0
B4..0	Command	4 ( <i>Enter Mode</i> )

## G.5 — Exit Mode Example

### G.5.1G.4.1 ~~Exit~~Enter Mode Command request with additional VDO

The Initiator of the *Enter Mode* Command request sends a Message Header with the *Number of Data Objects* ~~Number of Data Objects~~ field set to 42 indicating an additional VDO followed by a VDM Header with the Message Source (B5) set to zero indicating the Command is from an Initiator and the Command (B4..0) set to 54 to request the Responder to ~~exit~~enter its mode of operation and sets the Object Position field to the desired functional VDO based on its offset as received during Discovery.

Table G-8 shows the contents of the key fields in the Message Header and VDM Header for an Initiator sending an *Enter Mode* Command request with an additional VDO.

**Table G-10 Enter Mode Command request from Initiator Example**

Bit(s)	Field	Value
<b>Message Header</b>		
15	<del>Reserved</del>	<u>0</u>
14..12	<i>Number of Data Objects</i>	<u>1 (VDM Header)</u>
11..9	<i>MessageID</i>	<u>0..7</u>
8	<i>Port Power Role</i>	<u>0 or 1</u>
7..6	<i>Specification Revision</i>	<u>01b</u>
5..4	<del>Reserved</del>	<u>0</u>
3..0	<i>Message Type</i>	<u>1111b (Vendor Defined Message)</u>
<b>VDM Header</b>		
B31..16	<del>Standard or Vendor ID (SVID)</del>	<u>SVID for the Mode being entered</u>
B15	<del>VDM Type</del>	<u>1 (Structured VDM)</u>
B14..13	<del>Structured VDM Version</del>	<u>00b (Version 1.0)</u>
B12..11	<del>Reserved</del>	<u>00b</u>
B10..8	<del>Object Position</del>	<u>001b (a one in this field indicates a request to enter the first Mode in list returned by Discover Modes)</u>
B7..6	<del>Command Type</del>	<u>00b (Initiator)</u>
B5	<del>Reserved</del>	<u>0</u>

<b>Bit(s)</b>	<b>Field</b>	<b>Value</b>
<u>B4..0</u>	<u>Command</u>	<u>4 (Enter Mode)</u>
<b><u>Including optional Mode specific VDO</u></b>		
<u>B31..0</u>	<u>Mode specific</u>	



## G.5 Exit Mode Example

### G.5.1 Exit Mode Command request

The Initiator of the *Exit Mode* Command request sends a Message Header with the *Number of Data Objects* field set to 1 followed by a VDM Header with the Message Source (B5) set to zero indicating the Command is from an Initiator and the Command (B4..0) set to 5 to request the Responder to exit its Mode of operation.

Table G-11 shows the contents of the key fields in the Message Header and VDM header for an Initiator sending an *Exit Mode* Command request.

Table G-11 Exit Mode Command request from Initiator Example

Bit(s)	Field	Value
<b>Message Header</b>		
15	Reserved	0
14..12	<i>Number of Data Objects</i>	1 (VDM Header)
11..9	<i>MessageID</i>	0..7
8	<i>Port Power Role</i>	0 or 1
7..6	<i>Specification Revision</i>	01b
5..4	Reserved	0
3..0	<i>Message Type</i>	1111b (Vendor Defined Message)
<b>VDM Header</b>		
B31..16	Standard or Vendor ID (SVID)	SVID for the Mode being exited
B15	VDM Type	1 (Structured VDM)
B14..13	Structured VDM Version	00b (Version 1.0)
B12..11	Reserved	00b
B10..8	Object Position	001b (identifies the previously entered Mode by its Object Position that is to be exited)
B7..6	Command Type	00b (Initiator)
B5	Reserved	0
B4..0	<del>Command</del> Command	5 ( <i>Exit Mode</i> )

### G.5.2 Exit Mode Command response

The Responder that receives the *Exit Mode* Command request sends a Message Header with the *Number of Data Objects* field set to a value of 1 followed by a VDM Header with the Message Source (B5) set to 1 indicating the Command is from a Responder and the Command (B4..0) set to 5 indicating the Responder has exited the Mode and has returned to normal USB operation.

Table G-12 shows the contents of the key fields in the Message Header and VDM header for a Responder sending an *Exit Mode* Command ACK response.

Table G-12 Exit Mode Command response from Responder Example

Bit(s)	Field	Value
<b>Message Header</b>		
15	Reserved	0
14..12	<i>Number of Data Objects</i>	1 (VDM Header)
11..9	<i>MessageID</i>	0..7
8	<i>Port Power Role</i>	0 or 1
7..6	<i>Specification Revision</i>	01b

Bit(s)	Field	Value
5..4	Reserved	0
3..0	<i>Message Type</i>	1111b (Vendor Defined Message)
<b>VDM Header</b>		
B31..16	Standard or Vendor ID (SVID)	SVID for the Mode exited
B15	VDM Type	1 (Structured VDM)
B14..13	Structured VDM Version	00b (Version 1.0)
B12..11	Reserved	00b
B10..8	Object Position	001b (offset of the Mode to be exited)
B7..6	Command Type	01b (Responder ACK)
B5	Reserved	0
B4..0	Command	5 ( <i>Exit Mode</i> )

## G.6 Attention Example

### G.6.1 Attention Command request

The Initiator of the *Attention* Command request sends a Message Header with the *Number of Data Objects* field set to 1 followed by a VDM Header with the Message Source (B5) set to zero indicating the Command is from an Initiator and the Command (B4..0) set to 6 to request attention from the Responder.

Table G-11 shows the contents of the key fields in the Message Header and VDM header for an Initiator sending an *Attention* Command request.

Table G-13 Attention Command request from Initiator Example

Bit(s)	Field	Value
<b>Message Header</b>		
15	Reserved	0
14..12	<i>Number of Data Objects</i>	1 (VDM Header)
11..9	<i>MessageID</i>	0..7
8	<i>Port Power Role</i>	0 or 1
7..6	<i>Specification Revision</i>	01b
5..4	Reserved	0
3..0	<i>Message Type</i>	1111b (Vendor Defined Message)
<b>VDM Header</b>		
B31..16	Standard or Vendor ID (SVID)	SVID for which attention is being requested
B15	VDM Type	1 (Structured VDM)
B14..13	Structured VDM Version	00b (Version 1.0)
B12..11	Reserved	00b
B10..8	Object Position	001b (offset of the Mode requesting attention)
B7..6	Command Type	000b (Initiator)
B5	Reserved	0
B4..0	<i>Command</i>	6 ( <i>Attention</i> )

### G.6.2 Attention Command request with additional VDO

The Initiator of the *Attention* Command request sends a Message Header with the *Number of Data Objects* field set to 2 indicating an additional VDO followed by a VDM Header with the Message Source (B5) set to zero indicating the Command is from an Initiator and the Command (B4..0) set to 6 to request attention from the Responder.

Table G-11 shows the contents of the key fields in the Message Header and VDM header for an Initiator sending an *Attention* Command request with an additional VDO.

Table G-14 Attention Command request from Initiator with additional VDO Example

Bit(s)	Field	Value
<b>Message Header</b>		
15	Reserved	0
14..12	<i>Number of Data Objects</i>	2 (VDM Header + VDO)
11..9	<i>MessageID</i>	0..7
8	<i>Port Power Role</i>	0 or 1
7..6	<i>Specification Revision</i>	01b
5..4	Reserved	0

<b>Bit(s)</b>	<b>Field</b>	<b>Value</b>
<u>3..0</u>	<i>Message Type</i>	<u>1111b (Vendor Defined Message)</u>
<b>VDM Header</b>		
<u>B31..16</u>	<u>Standard or Vendor ID (SVID)</u>	<u>SVID for which attention is being requested</u>
<u>B15</u>	<u>VDM Type</u>	<u>1 (Structured VDM)</u>
<u>B14..13</u>	<u>Structured VDM Version</u>	<u>00b (Version 1.0)</u>
<u>B12..11</u>	<u>Reserved</u>	<u>00b</u>
<u>B10..8</u>	<u>Object Position</u>	<u>001b (offset of the Mode requesting attention)</u>
<u>B7..6</u>	<u>Command Type</u>	<u>000b (Initiator)</u>
<u>B5</u>	<u>Reserved</u>	<u>0</u>
<u>B4..0</u>	<u>Command</u>	<u>6 (Attention)</u>
<b>Including optional Mode specific VDO</b>		
<u>B31..0</u>	<u>Mode specific</u>	