

Universal Serial Bus Power Delivery Specification

Revision: ***2.0***
Version: ***1.3***
Release date: ***12 January 2017***

Editors

Bob Dunstan Renesas Electronics Corp.
Richard Petrie DisplayLink

Contributors

Charles Wang ACON, Advanced-Connectek, Inc.
Conrad Choy ACON, Advanced-Connectek, Inc.
Steve Sedio ACON, Advanced-Connectek, Inc.
Vicky Chuang ACON, Advanced-Connectek, Inc.
Joseph Scanlon Advanced Micro Devices
Caspar Lin Allion Labs, Inc.
Casper Lee Allion Labs, Inc.
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
Dave Conroy Apple
David Sekowski Apple
Girault Jones Apple
James Orr Apple
Jason Chung Apple
Jennifer Tsai Apple
Karl Bowers Apple
Keith Porthouse Apple
Matt Mora Apple
Paul Baker Apple
Reese Schreiber Apple
Sameer Kelkar Apple
Sasha Tietz Apple
Scott Jackson Apple
Sree Raman Apple
William Ferry Apple
Zaki Moussaoui Apple
Bernard Shyu Bizlink Technology, Inc.
Eric Wu Bizlink Technology, Inc.
Morphy Hsieh Bizlink Technology, Inc.
Shawn Meng Bizlink Technology Inc.
Tiffany Hsiao Bizlink Technology, Inc.
Weichung Ooi Bizlink Technology, Inc.
Michal Staworko Cadence Design Systems, Inc.
Alessandro Ingrassia Canova Tech
Andrea Colognese Canova Tech
Davide Ghedin Canova Tech
Matteo Casalin Canova Tech
Nicola Scantamburlo Canova Tech
Yi-Feng Lin Canyon Semiconductor
YuHung Lin Canyon Semiconductor

Anup Nayak	Cypress Semiconductor
Jagadeesan Raj	Cypress Semiconductor
Pradeep Bajpai	Cypress Semiconductor
Rushil Kadakia	Cypress Semiconductor
Steven Wong	Cypress Semiconductor
Subu Sankaran	Cypress Semiconductor
Sumeet Gupta	Cypress Semiconductor
Venkat Mandagulathar	Cypress Semiconductor
Adolfo Montero	Dell Inc.
Bruce Montag	Dell Inc.
Gary Verdun	Dell Inc.
Merle Wood	Dell Inc.
Mohammed Hijazi	Dell Inc.
Siddhartha Reddy	Dell Inc.
Dan Ellis	DisplayLink
Jason Young	DisplayLink
Kevin Jacobs	DisplayLink
Peter Burgers	DisplayLink
Richard Petrie	DisplayLink
Abel Astley	Ellisys
Chuck Trefts	Ellisys
Emmanuel Durin	Ellisys
Mario Pasquali	Ellisys
Tim Wei	Ellisys
Chien-Cheng Kuo	Etron Technology, Inc.
Jack Yang	Etron Technology, Inc.
Richard Crisp	Etron Technology, Inc.
Shyanjia Chen	Etron Technology, Inc.
TsungTa Lu	Etron Technology, Inc.
Christian Klein	Fairchild Semiconductor
Oscar Freitas	Fairchild Semiconductor
Souhib Harb	Fairchild Semiconductor
AJ Yang	Foxconn / Hon Hai
Fred Fons	Foxconn / Hon Hai
Steve Sedio	Foxconn / Hon Hai
Terry Little	Foxconn / Hon Hai
Bob McVay	Fresco Logic Inc.
Christopher Meyers	Fresco Logic Inc.
Tom Burton	Fresco Logic Inc.
Dian Kurniawan	Fresco Logic Inc.
Adam Rodriguez	Google Inc.
Alec Berg	Google Inc.
David Schneider	Google Inc.
Jim Guerin	Google Inc.
Juan Fantin	Google Inc.
Ken Wu	Google Inc.
Mark Hayter	Google Inc.
Nithya Jagannathan	Google Inc.
Todd Broch	Google Inc.
Vincent Palatin	Google Inc.

PD Chair/Device Policy Lead

Mike Engbretson	Granite River Labs	
Rajaraman V	Granite River Labs	
Alan Berkema	Hewlett Packard	
Lee Atkinson	Hewlett Packard	
Rahul Lakdawala	Hewlett Packard	
Robin Castell	Hewlett Packard	
Roger Benson	Hewlett Packard	
Ron Schooley	Hewlett Packard	
Suketu Partiwala	Hewlett Packard	
Vaibhav Malik	Hewlett Packard	
Walter Fry	Hewlett Packard	
Bob Dunstan	Intel Corporation	PD Chair/Protocol WG Lead
Brad Saunders	Intel Corporation	
Chee Lim Nge	Intel Corporation	
Christine Krause	Intel Corporation	
Dan Froelich	Intel Corporation	
David Harriman	Intel Corporation	
David Hines	Intel Corporation	
David Thompson	Intel Corporation	
Guobin Liu	Intel Corporation	
Harry Skinner	Intel Corporation	
Henrik Leegaard	Intel Corporation	
Jervis Lin	Intel Corporation	
John Howard	Intel Corporation	
Karthi Vadivelu	Intel Corporation	
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	
Jia Wei	Intersil 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)	
Brian Fetz	Keysight Technologies Inc.	
Babu Mailachalam	Lattice Semiconductor Corp	
Gianluca Mariani	Lattice Semiconductor Corp	
Joel Coplen	Lattice Semiconductor Corp	
Thomas Watz	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	
Tyler Joe	LeCroy Corporation	
Phil Jakes	Lenovo	
Dave Thompson	LSI Corporation	
Alan Kinningham	Luxshare-ICT	
Daniel Chen	Luxshare-ICT	
Josue Castillo	Luxshare-ICT	
Scott Shuey	Luxshare-ICT	
Chris Yokum	MCCI Corporation	
Geert Knapen	MCCI Corporation	
Terry Moore	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.	
Ronald Kunin	Microchip Technology Inc.	
Shannon Cash	Microchip Technology Inc.	
Anthony Chen	Microsoft Corporation	
Dave Perchlik	Microsoft Corporation	
David Voth	Microsoft Corporation	
Geoff Shew	Microsoft Corporation	
Jayson Kastens	Microsoft Corporation	
Kai Inha	Microsoft Corporation	
Marwan Kadado	Microsoft Corporation	
Michelle Bergeron	Microsoft Corporation	
Rahul Ramadas	Microsoft Corporation	
Randy Aull	Microsoft Corporation	
Shiu Ng	Microsoft Corporation	
Timo Toivola	Microsoft Corporation	
Toby Nixon	Microsoft Corporation	
Vivek Gupta	Microsoft Corporation	
Yang You	Microsoft Corporation	
Dan Wagner	Motorola Mobility Inc.	
Ben Crowe	MQP Electronics Ltd.	
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	
Dong Nguyen	NXP Semiconductors	
Guru Prasad	NXP Semiconductors	
Ken Jaramillo	NXP Semiconductors	
Krishnan TN	NXP Semiconductors	
Michael Joehren	NXP Semiconductors	
Robert de Nie	NXP Semiconductors	
Rod Whitby	NXP Semiconductors	
Vijendra Kuroodi	NXP Semiconductors	
Robert Heaton	Obsidian Technology	
Bryan McCoy	ON Semiconductor	
Christian Klein	ON Semiconductor	
Cor Voorwinden	ON Semiconductor	
Edward Berrios	ON Semiconductor	Power Supply WG Lead
Oscar Freitas	ON Semiconductor	
Tom Duffy	ON Semiconductor	
Craig Wiley	Parade Technologies Inc.	
Aditya Kulkarni	Power Integrations	
Rahul Joshi	Power Integrations	
Ricardo Pregiteer	Power Integrations	
Chris Sporck	Qualcomm, Inc.	
Craig Aiken	Qualcomm, Inc.	
George Paparrizos	Qualcomm, Inc.	
Giovanni Garcea	Qualcomm, Inc.	
James Goel	Qualcomm, Inc.	
Joshua Warner	Qualcomm, Inc.	
Narendra Mehta	Qualcomm, Inc.	
Terry Remple	Qualcomm, Inc.	
Will Kun	Qualcomm, Inc.	
Yoram Rimoni	Qualcomm, Inc.	
Atsushi Mitamura	Renesas Electronics Corp.	
Bob Dunstan	Renesas Electronics Corp.	
Dan Aoki	Renesas Electronics Corp.	
Kiichi Muto	Renesas Electronics Corp.	
Masami Katagiri	Renesas Electronics Corp.	
Nobuo Furuya	Renesas Electronics Corp.	
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.	
Toshifumi Yamaoka	Renesas Electronics Corp.	
Chunan Kuo	Richtek Technology Corporation	
Heinz Wei	Richtek Technology Corporation	
Tatsuya Irisawa	Ricoh Company Ltd.	
Akihiro Ono	Rohm Co. Ltd.	
Chris Lin	Rohm Co. Ltd.	
Hidenori Nishimoto	Rohm Co. Ltd.	
Kris Bahar	Rohm Co. Ltd.	

Manabu Miyata	Rohm Co. Ltd.	
Ruben Balbuena	Rohm Co. Ltd.	
Takashi Sato	Rohm Co. Ltd.	
Vijendra Kuroodi	Rohm Co. Ltd.	
Yusuke Kondo	Rohm Co. Ltd.	
Matti Kulmala	Salcomp Plc	
Toni Lehimo	Salcomp Plc	
Tong Kim	Samsung Electronics Co. Ltd.	
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	
Chin Chang	Semtech Corporation	
Kafai Leung	Silicon Laboratories, Inc.	
Abhishek Sardeshpande	SiliConch Systems Private Limited	
Jaswanth Ammineni	SiliConch Systems Private Limited	
Kaustubh Kumar	SiliConch Systems Private Limited	
Pavitra Balasubramanian	SiliConch Systems Private Limited	
Rakesh Polasa	SiliConch Systems Private Limited	
Vishnu Pusuluri	SiliConch Systems Private Limited	
John Sisto	SMSC	
Ken Gay	SMSC	
Mark Bohm	SMSC	
Richard Wahler	SMSC	
Shannon Cash	SMSC	
Tim Knowlton	SMSC	
William Chiechi	SMSC	
Bob Dunstan	Specwerkz	
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	
Christophe Lorin	ST-Microelectronics	
John Bloomfield	ST-Microelectronics	
Massimo Panzica	ST-Microelectronics	
Meriem Mersel	ST-Microelectronics	
Nathalie Ballot	ST-Microelectronics	
Pascal Legrand	ST-Microelectronics	
Patrizia Milazzo	ST-Microelectronics	
Richard O'Connor	ST-Microelectronics	
Zongyao Wen	Synopsys, Inc.	
Joan Marrinan	Tektronix	
Kimberley McKay	Teledyne-LeCroy	
Matthew Dunn	Teledyne-LeCroy	
Tony Minchell	Teledyne-LeCroy	
Anand Dabak	Texas Instruments	
Bill Waters	Texas Instruments	

Bing Lu	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	
Mike Campbell	Texas Instruments	
Scott Jackson	Texas Instruments	
Srinath Hosur	Texas Instruments	
Steven Tom	Texas Instruments	
Chris Yokum	Total Phase	
Brad Cox	Ventev Mobile	
Colin Vose	Ventev Mobile	
Dydron Lin	VIA Technologies, Inc.	
Fong-Jim Wang	VIA Technologies, Inc.	
Jay Tseng	VIA Technologies, Inc.	
Rex Chang	VIA Technologies, Inc.	
Terrance Shih	VIA Technologies, Inc.	
Jeng Cheng Liu	Weltrend Semiconductor	
Wayne Lo	Weltrend Semiconductor	
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
2.0	1.1	Including errata through 7-May 2015	7 May 2015
2.0	1.2	Including errata through 25-March-2016	25 March 2016
2.0	1.3	Including errata through 12-January-2017	12 January 2017

INTELLECTUAL PROPERTY DISCLAIMER

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

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

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

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

Copyright © 2010-2017 Apple Inc, Hewlett-Packard Company, Intel Corporation, Microsoft Corporation, Renesas, STMicroelectronics, and Texas Instruments

All rights reserved.

Table of Contents

Editors	2
Contributors	2
Revision History	9
INTELLECTUAL PROPERTY DISCLAIMER	10
Table of Contents	11
List of Tables	21
List of Figures	27
1. Introduction	34
1.1 Overview.....	34
1.2 Purpose.....	35
1.3 Scope.....	35
1.4 Conventions.....	35
1.4.1 Precedence.....	35
1.4.2 Keywords.....	35
1.4.3 Numbering.....	37
1.5 Related Documents	37
1.6 Terms and Abbreviations.....	37
1.7 Parameter Values.....	44
2. Overview	45
2.1 Introduction.....	45
2.2 Section Overview	46
2.3 USB Power Delivery Capable Devices	47
2.4 SOP* Communication.....	48
2.4.1 Introduction.....	48
2.4.2 SOP* Collision Avoidance	48
2.4.3 SOP Communication.....	48
2.4.4 SOP'/SOP'' Communication with Cable Plugs.....	48
2.5 Operational Overview.....	50
2.5.1 Source Operation.....	50

2.5.2	Sink Operation	53
2.5.3	Cable Plugs.....	55
2.6	Architectural Overview	56
2.6.1	Policy	58
2.6.2	Message Formation and Transmission	59
2.6.3	Collision Avoidance	59
2.6.4	Power supply.....	59
2.6.5	DFP/UFP	60
2.6.6	VCONN Source.....	60
2.6.7	Cable and Connectors.....	60
2.6.8	Interactions between Non-PD, BC and PD devices.....	61
2.6.9	Power Rules	61
3.	USB Type-A and USB Type-B Cable Assemblies and Connectors	62
3.1	Significant Features	62
3.1.1	Connectors.....	62
3.1.2	Compliant Cable Assemblies	64
3.1.3	USB Power Delivery Adapters (USB plug to USB receptacle)	65
3.1.4	Hardwired Captive PD Cable.....	65
3.1.5	Standard-A Insertion Detect	65
3.1.6	Standard-A PD Detect.....	65
3.1.7	Raw Cables.....	66
3.2	Connector Mating Interfaces	67
3.2.1	Standard-A Insertion Detect Mechanical Dimensions.....	67
3.2.2	USB PD Standard-A PD Detect Mechanical Requirement	67
3.2.3	USB 2.0 PD Standard-A Connector	68
3.2.4	USB 3.1 PD Standard-A Connector	72
3.2.5	USB 2.0 PD Standard-B Connector	76
3.2.6	USB 3.1 PD Standard-B Connector	79
3.3	Cable Assemblies.....	84
3.3.1	Non-marked Cable Assemblies	84
3.3.2	Marked Cable Assemblies.....	84
3.3.3	PD Cable Assembly Overmold Requirements.....	84
3.4	PD Cable Assembly Marking.....	86

3.4.1	Marker for PD Standard-A Connectors	86
3.4.2	Electronic Markers for Micro-A Plugs	86
3.4.3	Electronic Markers for PD Standard-B Plugs and Micro-B Plugs	87
3.5	USB PD Icon	88
3.6	USB Power Delivery Cable Requirements.....	89
3.6.1	Low Level Contact Resistance (EIA 364-23B).....	89
3.6.2	Open Circuit Resistance.....	89
3.6.3	Dielectric Strength (EIA 364-20)	89
3.6.4	Insulation Resistance (EIA 364-21).....	89
3.6.5	Contact Current Rating	90
3.6.6	Differential Crosstalk between V_{BUS} and the D+/D- Pair (EIA-360-90)	90
3.6.7	PD Cable Assembly Shielding Connectivity	90
3.6.8	PD Cable V_{BUS} Impedance.....	91
3.6.9	PD Cable Insertion Loss.....	91
3.6.10	PD Cable IR Drop Considerations	91
3.7	Electrical Parameters.....	93
4.	Electrical Requirements.....	94
4.1	Interoperability with other USB Specifications	94
4.2	Dead Battery Detection / Unpowered Port Detection	94
4.2.1	Type-A to USB Type-B Dead Battery Operation	94
4.2.2	Type-C to USB Type-C Dead Battery Operation.....	98
4.3	Cable IR Ground Drop (IR Drop)	98
4.4	A-Plug Insertion Detect.....	98
4.5	Cable Type Detection	98
4.5.1	Detecting Cabling Capabilities.....	98
4.5.2	Plug Type Determination.....	99
4.5.3	Detecting the PD Capabilities of the Standard-A Connector.....	100
4.5.4	Plug Type Detection except Standard-A.....	101
4.6	Low Power Devices using Micro-A Plug	102
4.7	Electrical Parameters.....	103
5.	Physical Layer.....	104
5.1	Physical Layer Overview	104
5.2	Physical Layer Functions.....	104

5.3	Symbol Encoding.....	105
5.4	Ordered Sets.....	106
5.5	Transmitted Bit Ordering	107
5.6	Packet Format.....	108
5.6.1	Packet Framing.....	108
5.6.2	CRC	110
5.6.3	Packet Detection Errors	112
5.6.4	Hard Reset.....	112
5.6.5	Cable Reset	113
5.7	Collision Avoidance.....	113
5.8	Physical Layer Signaling Schemes	114
5.8.1	Common Signaling Scheme Specifications	114
5.8.2	Binary Frequency Shift Keyed (BFSK) Signaling Scheme	116
5.8.3	Biphase Mark Coding (BMC) Signaling Scheme.....	124
5.8.4	Interoperability with BFSK and BMC.....	138
5.9	Built in Self-Test (BIST)	139
5.9.1	BIST PRBS Pattern	139
5.9.2	BIST Carrier Mode 0.....	140
5.9.3	BIST Carrier Mode 1.....	140
5.9.4	BIST Carrier Mode 2.....	140
5.9.5	BIST Carrier Mode 3.....	140
5.9.6	BIST Eye Pattern	140
5.9.7	BIST Test Data.....	141
5.9.8	BIST Parameters	141
5.9.9	BIST Test Applicability	141
6.	Protocol Layer	142
6.1	Overview.....	142
6.2	Messages.....	142
6.2.1	Message Construction	142
6.3	Control Message	145
6.3.1	GoodCRC Message.....	145
6.3.2	GotoMin Message.....	146
6.3.3	Accept Message	146

6.3.4	Reject Message.....	146
6.3.5	Ping Message	147
6.3.6	PS_RDY Message	147
6.3.7	Get_Source_Cap Message	147
6.3.8	Get_Sink_Cap Message.....	147
6.3.9	DR_Swap Message	147
6.3.10	PR_Swap Message.....	148
6.3.11	VCONN_Swap Message.....	149
6.3.12	Wait Message.....	149
6.3.13	Soft Reset Message.....	150
6.4	Data Message	150
6.4.1	Capabilities Message.....	151
6.4.2	Request Message.....	159
6.4.3	BIST Message.....	162
6.4.4	Vendor Defined Message	166
6.5	Timers	188
6.5.1	CRCReceiveTimer.....	188
6.5.2	SenderResponseTimer.....	188
6.5.3	Activity Timers	188
6.5.4	Capability Timers	189
6.5.5	Wait Timers and Times	190
6.5.6	Power Supply Timers	190
6.5.7	NoResponseTimer.....	191
6.5.8	BIST Timers.....	192
6.5.9	Power Role Swap Timers.....	193
6.5.10	Soft Reset Timers.....	193
6.5.11	Hard Reset Timers	193
6.5.12	Structured VDM Timers	194
6.5.13	VCONN Timers	195
6.5.14	tCableMessage.....	195
6.5.15	DiscoverIdentityTimer.....	195
6.5.16	Attention Timers.....	196
6.5.17	Time Values and Timers.....	196
6.6	Counters	199

6.6.1	MessageID Counter.....	199
6.6.2	Retry Counter.....	199
6.6.3	Hard Reset Counter.....	199
6.6.4	Capabilities Counter.....	200
6.6.5	BIST Error Counter.....	200
6.6.6	Discover Identity Counter.....	200
6.6.7	VDMBusyCounter.....	200
6.6.8	nAttentionCount.....	200
6.6.9	Counter Values and Counters.....	200
6.7	Reset.....	201
6.7.1	Soft Reset and Protocol Error.....	201
6.7.2	Hard Reset.....	203
6.7.3	Cable Reset.....	203
6.8	Message Discarding.....	204
6.9	State behavior.....	206
6.9.1	Introduction to state diagrams used in Chapter 6.....	206
6.9.2	State Operation.....	206
6.9.3	BIST Operation.....	215
6.9.4	List of Protocol Layer States.....	219
6.10	Message Applicability.....	220
6.10.1	Applicability of Control Messages.....	220
6.10.2	Applicability of Data Messages.....	221
6.10.3	Applicability of Structured VDM Commands.....	222
6.10.4	Applicability of Reset Signaling.....	223
7.	Power Supply.....	224
7.1	Source Requirements.....	224
7.1.1	Behavioral Aspects.....	224
7.1.2	Source Bulk Capacitance.....	224
7.1.3	Types of Sources.....	225
7.1.4	Positive Voltage Transitions.....	225
7.1.5	Negative Voltage Transitions.....	226
7.1.6	Response to Hard Resets.....	226
7.1.7	Changing the Output Power Capability.....	227

7.1.8	Robust Source Operation.....	227
7.1.9	Output Voltage Tolerance and Range	228
7.1.10	Charging and Discharging the Bulk Capacitance on V_{BUS}	229
7.1.11	Swap Standby for Sources.....	229
7.1.12	Source Peak Current Operation.....	230
7.1.13	BFSK over V_{BUS} Considerations for Sources.....	230
7.1.14	Application of V_{BUS} Slew Rate Limits.....	233
7.2	Sink Requirements.....	233
7.2.1	Behavioral Aspects	233
7.2.2	Sink Bulk Capacitance	233
7.2.3	Sink Standby	234
7.2.4	Suspend Power Consumption	234
7.2.5	Zero Negotiated Current.....	234
7.2.6	Transient Load Behavior	234
7.2.7	Swap Standby for Sinks	235
7.2.8	Sink Peak Current Operation	235
7.2.9	BFSK over V_{BUS} Considerations for Sinks	235
7.2.10	Robust Sink Operation.....	237
7.3	Transitions	239
7.3.1	Increasing the Current.....	240
7.3.2	Increasing the Voltage.....	242
7.3.3	Increasing the Voltage and Current.....	244
7.3.4	Increasing the Voltage and Decreasing the Current.....	246
7.3.5	Decreasing the Voltage and Increasing the Current.....	248
7.3.6	Decreasing the Current.....	250
7.3.7	Decreasing the Voltage	252
7.3.8	Decreasing the Voltage and the Current	254
7.3.9	Sink Requested Power Role Swap.....	256
7.3.10	Source Requested Power Role Swap.....	259
7.3.11	GotoMin Current Decrease.....	262
7.3.12	Source Initiated Hard Reset.....	264
7.3.13	Sink Initiated Hard Reset.....	266
7.3.14	Type-A/B Hard Reset after a Power Role Swap	268
7.3.15	Type-A to USB Type-B Dead Battery Operation	276

7.3.19	No change in Current or Voltage	278
7.4	Electrical Parameters	280
7.4.1	Source Electrical Parameters	280
7.4.2	Sink Electrical Parameters	282
7.4.3	Common Electrical Parameters	283
8.	Device Policy	284
8.1	Overview	284
8.2	Device Policy Manager	284
8.2.1	Capabilities	285
8.2.2	System Policy	285
8.2.3	Control of Source/Sink	285
8.2.4	Cable Detection	286
8.2.5	Managing Power Requirements	286
8.2.6	Use of “Unconstrained Power” bit with Batteries and AC supplies	288
8.2.7	Interface to the Policy Engine	290
8.3	Policy Engine	292
8.3.1	Introduction	292
8.3.2	Atomic Message Sequence Diagrams	292
8.3.3	State Diagrams	395
9.	States and Status Reporting	470
9.1	Overview	470
9.1.1	PDUSB Device and Hub Requirements	472
9.1.2	Mapping to USB Device States	472
9.1.3	PD Software Stack	475
9.1.4	PDUSB Device Enumeration	475
9.2	PD Specific Descriptors	477
9.2.1	USB Power Delivery Capability Descriptor	477
9.2.2	Battery Info Capability Descriptor	478
9.2.3	PD Consumer Port Capability Descriptor	479
9.2.4	PD Provider Port Capability Descriptor	479
9.3	PD Specific Requests and Events	481
9.3.1	PD Specific Requests	481
9.4	PDUSB Hub and PDUSB Peripheral Device Requests	482

9.4.1	GetBatteryStatus.....	482
9.4.2	SetPDFeature.....	483
10.	Power Rules.....	485
10.1	Introduction.....	485
10.2	Source Power Rules.....	485
10.2.1	Source Power Rule Considerations.....	485
10.2.2	Normative Voltages and Currents.....	485
10.2.3	Optional Voltages/Currents.....	488
10.2.4	Power sharing between ports.....	489
10.3	Sink Power Rules.....	489
10.3.1	Sink Power Rule Considerations.....	489
10.3.2	Normative Sink Rules.....	489
A.	Power Profiles – This section is Deprecated.....	490
B.	CRC calculation.....	491
B.1	C code example.....	491
B.2	Table showing the full calculation over one Message.....	493
C.	Power Implementation Considerations.....	494
C.1	Managing Isolation Impedance (BFSK).....	494
C.1.1	In-band fCarrier Spurious Noise.....	494
C.1.2	Spurious Noise Test Setup and Calibration.....	495
C.2	Connector Detach Transients.....	497
C.3	Closed Loop Stability Effects.....	498
C.3.1	Basic Power Stage Small Signal AC Model.....	498
C.3.2	Feedback Past Isolation Inductor.....	500
D.	Standard-A Mating Illustrations.....	502
E.	Physical Layer Informative Material.....	509
E.1	Squelch Budget.....	509
F.	PD Message Sequence Examples.....	511
F.1	External power is supplied downstream.....	511
F.2	External power is supplied upstream.....	515
F.3	Giving back power.....	522
G.	VDM Command Examples.....	534

G.1	Discover Identity Example	534
G.1.1	Discover Identity Command request	534
G.1.2	Discover Identity Command response – Active Cable	534
G.1.3	Discover Identity Command response – Hub	535
G.2	Discover SVIDs Example	537
G.2.1	Discover SVIDs Command request	537
G.2.1	Discover SVIDs Command response	537
G.3	Discover Modes Example	539
G.3.1	Discover Modes Command request	539
G.3.2	Discover Modes Command response	539
G.4	Enter Mode Example	541
G.4.1	Enter Mode Command request	541
G.4.2	Enter Mode Command response	541
G.4.1	Enter Mode Command request with additional VDO	542
G.5	Exit Mode Example	544
G.5.1	Exit Mode Command request	544
G.5.2	Exit Mode Command response	544
G.6	Attention Example	546
G.6.1	Attention Command request	546
G.6.2	Attention Command request with additional VDO	546

List of Tables

Table 1-1 Terms and Abbreviations	37
Table 3-1 Plugs Accepted By Receptacles	63
Table 3-2 USB 2.0 PD Standard-A Connector Pin Assignments	72
Table 3-3 USB 3.1 PD Standard-A Connector Pin Assignments	76
Table 3-4 USB 2.0 PD Standard-B Connector Pin Assignments.....	79
Table 3-5 USB 3.1 PD Standard-B Connector Pin Assignments.....	83
Table 3-6 USB PD Cable Assembly Overmold Maximum Dimensions	84
Table 3-7 Electrical Parameters	93
Table 4-1 Normal Dead Battery Operation	97
Table 4-2 Plug Type Determination.....	102
Table 4-3 Electrical Parameters	103
Table 4-4 Electrical Timers.....	103
Table 5-1 4b5b Symbol Encoding Table	105
Table 5-2 Ordered Sets.....	106
Table 5-3 Validation of Ordered Sets	106
Table 5-4 Data Size.....	107
Table 5-5 SOP ordered set.....	108
Table 5-6 SOP' ordered set.....	109
Table 5-7 SOP'' ordered set.....	109
Table 5-8 SOP'_Debug ordered set	110
Table 5-9 SOP''_Debug ordered set	110
Table 5-10 CRC-32 Mapping.....	111
Table 5-11 Hard Reset ordered set.....	112
Table 5-12 Cable Reset ordered set	113
Table 5-13 Common Normative Signaling Scheme Requirements.....	114
Table 5-14 Common Normative Signaling Scheme Requirements for Transmitter	114
Table 5-15 Common Normative Signaling Scheme Requirements for Receiver	114
Table 5-16 BFSK Common Normative Requirements.....	118
Table 5-17 BFSK Transceiver Isolation Impedance Normative Requirements.....	118
Table 5-18 BFSK Transmitter Normative Requirements	118
Table 5-19 BFSK Spectrum Mask Corners.....	121
Table 5-20 BFSK Receiver Normative Requirements	122
Table 5-21 BMC Tx Mask Definition, X Values.....	128
Table 5-22 BMC Tx Mask Definition, Y Values.....	128
Table 5-23 BMC Rx Mask Definition	133

Table 5-24 BMC Common Normative Requirements	135
Table 5-25 BMC Transmitter Normative Requirements	135
Table 5-26 BMC Receiver Normative Requirements.....	136
Table 5-27 Allowable Bit Errors vs. Number of Test Frames.....	140
Table 5-28 BIST Parameters	141
Table 5-29 BIST Mode support.....	141
Table 6-1 Message Header	143
Table 6-2 Control Message Types.....	145
Table 6-3 Data Message Types	151
Table 6-4 Power Data Object.....	152
Table 6-5 USB Type-A to USB Type-A Port Behavior	154
Table 6-6 Fixed Supply PDO - Source.....	154
Table 6-7 Fixed Power Source Peak Current Capability	156
Table 6-8 Variable Supply (non-Battery) PDO - Source.....	156
Table 6-9 Battery Supply PDO - Source.....	156
Table 6-10 Fixed Supply PDO - Sink	157
Table 6-11 Variable Supply (non-Battery) PDO - Sink	158
Table 6-12 Battery Supply PDO - Sink.....	159
Table 6-13 Fixed and Variable Request Data Object.....	159
Table 6-14 Fixed and Variable Request Data Object with GiveBack Support.....	159
Table 6-15 Battery Request Data Object	160
Table 6-16 Battery Request Data Object with GiveBack Support	160
Table 6-17 BIST Data Object.....	164
Table 6-18 Unstructured VDM Header.....	167
Table 6-19 Structured VDM Header	167
Table 6-20 Structured VDM Commands.....	168
Table 6-21 SVID Values.....	168
Table 6-22 Commands and Responses.....	170
Table 6-23 ID Header VDO	172
Table 6-24 Product Types (UFP).....	172
Table 6-25 Product Types (Cable Plug)	173
Table 6-26 Cert Stat VDO	173
Table 6-27 Product VDO.....	173
Table 6-28 Passive Cable VDO	174
Table 6-29 Active Cable VDO.....	176
Table 6-30 AMA VDO	177

Table 6-31 Discover SVIDs Responder VDO	179
Table 6-32 Time Values	196
Table 6-33 Timers	197
Table 6-34 Counter parameters	200
Table 6-35 Counters	201
Table 6-36 Response to an incoming Message	202
Table 6-37 Message discarding	205
Table 6-38 Protocol Layer States	219
Table 6-39 Applicability of Control Messages	220
Table 6-40 Applicability of Data Messages	221
Table 6-41 Applicability of Structured VDM Commands	222
Table 6-42 Applicability of Reset Signaling	223
Table 7-1 Noise Spectral Mask Corners	232
Table 7-2 Noise Spectral Mask Corners	236
Table 7-3 Sequence Description for Increasing the Current	241
Table 7-4 Sequence Description for Increasing the Voltage	243
Table 7-5 Sequence Diagram for Increasing the Voltage and Current	245
Table 7-6 Sequence Description for Increasing the Voltage and Decreasing the Current	247
Table 7-7 Sequence Description for Decreasing the Voltage and Increasing the Current	249
Table 7-8 Sequence Description for Decreasing the Current	251
Table 7-9 Sequence Description for Decreasing the Voltage	253
Table 7-10 Sequence Description for Decreasing the Voltage and the Current	255
Table 7-11 Sequence Description for a Sink Requested Power Role Swap	257
Table 7-12 Sequence Description for a Source Requested Power Role Swap	260
Table 7-13 Sequence Description for a GotoMin Current Decrease	263
Table 7-14 Sequence Description for a Source Initiated Hard Reset	265
Table 7-15 Sequence Description for a Sink Initiated Hard Reset	267
Table 7-16 Sequence Description for USB Type-B New Source Initiated Hard Reset and USB Type-A New Sink Receives Hard Reset Signaling	269
Table 7-17 Sequence Description for USB Type-B New Source Initiated Hard Reset and USB Type-A New Sink Does Not Receive Hard Reset Signaling	271
Table 7-18 Sequence Description for USB Type-A New Sink Initiated Hard Reset and USB Type-B New Source Receives Hard Reset Signaling	273
Table 7-19 Sequence Description for USB Type-A New Sink Initiated Hard Reset and USB Type-B New Source Does Not Receive Hard Reset Signaling	275
Table 7-20 Sequence Description for USB Type-A to USB Type-B Dead Battery Operation	277
Table 7-21 Sequence Description for no change in Current or Voltage	279

Table 7-22 Source Electrical Parameters.....	280
Table 7-23 Sink Electrical Parameters	282
Table 7-24 Common Source/Sink Electrical Parameters.....	283
Table 8-1 Basic Message Flow	293
Table 8-2 Potential issues in Basic Message Flow	294
Table 8-3 Basic Message Flow with CRC failure	295
Table 8-4 Interruptible and Non-interruptible AMS	296
Table 8-5 Steps for a successful Power Negotiation.....	299
Table 8-6 Steps for a GotoMin Negotiation	302
Table 8-7 Steps for a Soft Reset	304
Table 8-8 Steps for Source initiated Hard Reset.....	307
Table 8-9 Steps for Sink initiated Hard Reset	310
Table 8-10 Steps for Source initiated Hard Reset – Sink long reset.....	313
Table 8-11 Steps for a Successful USB Type-A or USB Type-B Source Initiated Power Role Swap Sequence.....	317
Table 8-12 Steps for a Successful USB Type-A or USB Type-B Sink Initiated Power Role Swap Sequence.....	321
Table 8-13 Steps for USB Type-A or USB Type-B Source initiated Hard Reset (Power Role Swapped)	325
Table 8-14 Steps for USB Type-A or USB Type-B Sink initiated Hard Reset (Power Role Swapped)	328
Table 8-15 Steps for a Successful USB Type-C Source Initiated Power Role Swap Sequence	332
Table 8-16 Steps for a Successful USB Type-C Sink Initiated Power Role Swap Sequence	337
Table 8-17 Steps for USB Type-C Data Role Swap, UFP operating as Sink initiates.....	341
Table 8-18 Steps for USB Type-C Data Role Swap, UFP operating as Source initiates	344
Table 8-19 Steps for USB Type-C Data Role Swap, DFP operating as Source initiates	347
Table 8-20 Steps for USB Type-C Data Role Swap, DFP operating as Sink initiates.....	350
Table 8-21 Steps for USB Type-C Source to Sink VCONN Source Swap.....	353
Table 8-22 Steps for USB Type-C Sink to Source VCONN Source Swap.....	356
Table 8-23 Steps for a Sink getting Source capabilities Sequence	358
Table 8-24 Steps for a Dual-Role Source getting Dual-Role Sink’s capabilities as a Source Sequence	360
Table 8-25 Steps for a Source getting Sink capabilities Sequence	362
Table 8-26 Steps for a Dual-Role Sink getting Dual-Role Source capabilities as a Sink Sequence	364
Table 8-27 Steps for DFP to UFP Discover Identity	366
Table 8-28 Steps for Source Port to Cable Plug Discover Identity.....	368
Table 8-29 Steps for DFP to Cable Plug Discover Identity	370
Table 8-30 Steps for DFP to UFP Enter Mode	373
Table 8-31 Steps for DFP to UFP Exit Mode.....	375
Table 8-32 Steps for DFP to Cable Plug Enter Mode.....	378
Table 8-33 Steps for DFP to Cable Plug Exit Mode	380

Table 8-34 Steps for UFP to DFP Attention.....	382
Table 8-35 Steps for BIST Receiver Mode test.....	385
Table 8-36 Steps for BIST Transmit Mode test	388
Table 8-37 Steps for BIST Eye Pattern Test.....	391
Table 8-38 Steps for BIST Test Data Test.....	393
Table 8-39 Policy Engine States.....	464
Table 9-1 USB Power Delivery Type Codes.....	477
Table 9-2 USB Power Delivery Capability Descriptor.....	477
Table 9-3 Battery Info Capability Descriptor	478
Table 9-4 PD Consumer Port Descriptor	479
Table 9-5 PD Provider Port Descriptor	479
Table 9-6 PD Requests	481
Table 9-7 PD Request Codes.....	481
Table 9-8 PD Feature Selectors.....	481
Table 9-9 Battery Status Structure.....	482
Table 9-10 Battery Wake Mask.....	483
Table 9-11 Charging Policy Encoding.....	484
Table 10-1 Considerations for Sources	485
Table 10-2 Normative Voltages and Currents	486
Table 10-3 Fixed Supply PDO – Source 5V.....	487
Table 10-4 Fixed Supply PDO – Source 9V.....	488
Table 10-5 Fixed Supply PDO – Source 15V.....	488
Table 10-6 Fixed Supply PDO – Source 20V.....	488
Table F-1 External power is supplied downstream.....	512
Table F-2 External power is supplied upstream.....	515
Table F-3 Giving back power.....	522
Table G-1 Discover Identity Command request from Initiator Example.....	534
Table G-2 Discover Identity Command response from Active Cable Responder Example.....	534
Table G-3 Discover Identity Command response from Hub Responder Example	535
Table G-4 Discover SVIDs Command request from Initiator Example	537
Table G-5 Discover SVIDs Command response from Responder Example.....	537
Table G-6 Discover Modes Command request from Initiator Example.....	539
Table G-7 Discover Modes Command response from Responder Example	539
Table G-8 Enter Mode Command request from Initiator Example.....	541
Table G-9 Enter Mode Command response from Responder Example	541
Table G-10 Enter Mode Command request from Initiator Example.....	542

Table G-11 Exit Mode Command request from Initiator Example.....	544
Table G-12 Exit Mode Command response from Responder Example.....	544
Table G-13 Attention Command request from Initiator Example.....	546
Table G-14 Attention Command request from Initiator with additional VDO Example	546

List of Figures

Figure 2-1 Logical Structure of USB Power Delivery Capable Devices	47
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 USB Power Delivery Communications Stack.....	56
Figure 2-5 USB Power Delivery Communication Over USB	57
Figure 2-6 High Level Architecture View.....	58
Figure 3-1 Standard-A Insertion Detect Schematic Representation.....	65
Figure 3-2 PD Standard-A No Plug Detection Circuit.....	66
Figure 3-3 Non-PD Plug Standard-A Detection Circuit.....	66
Figure 3-4 USB Thin Card Standard-A Detection Circuit.....	66
Figure 3-5 PD Plug Standard-A Detection Circuit	66
Figure 3-6 Insertion Detect Zone Mechanical Dimensions for the Standard-A Receptacle.....	67
Figure 3-7 PD Detect Plane Location Range for PD Standard-A Receptacles	68
Figure 3-8 PD Standard-A Plug Interface Dimensions.....	69
Figure 3-9 USB 2.0 PD Standard-A Receptacle Interface Dimensions	70
Figure 3-10 Reference Footprint for the USB 2.0 PD Standard-A Top Mount Single Receptacle (Informative)	71
Figure 3-11 USB 3.1 PD Standard-A Plug Interface Dimensions	73
Figure 3-12 Reference USB 3.1 PD Standard-A Receptacle Interface Dimensions (Informative)	74
Figure 3-13 Reference Footprint for the USB 3.1 PD Standard-A Top Mount Single Receptacle (Informative)	75
Figure 3-14 USB 2.0 PD Standard-B Plug Interface Dimensions	77
Figure 3-15 USB 2.0 PD Standard-B Receptacle Interface Dimensions.....	78
Figure 3-16 Reference Footprint for the USB 2.0 PD Standard-B Receptacle.....	79
Figure 3-17 USB 3.1 PD Standard-B Plug Interface Dimensions	80
Figure 3-18 USB 3.1 PD Standard-B Receptacle Interface Dimensions.....	81
Figure 3-19 Reference Footprint for the USB 3.1 PD Standard-B Receptacle.....	82
Figure 3-20 USB PD Cable Assembly Overmold Maximum Dimensions.....	85
Figure 3-21 Schematic of a Micro-A Plug Legacy Termination	86
Figure 3-22 Schematic of a Micro-A Plug Marker Indicating Low Power Capability.....	87
Figure 3-23 Schematic of a Micro-A PD Plug.....	87
Figure 3-24 Schematic of a B Plug Connector Marker Indicating 3A Capability	88
Figure 3-25 Schematic of a B Plug Connector Marker Indicating 5A Capability	88
Figure 3-26 Differential Near-End and Far-End Crosstalk Requirement between the D+/D- Pair and V_{BUS}	90
Figure 3-27 Voltage Drop Measurement	91
Figure 4-1 USB Type-A to USB Type-B Dead Battery / Unpowered Port Detection Flow.....	96
Figure 4-2 Plug Type Determination.....	100

Figure 4-3 Standard-A Plug PD Capabilities Flow.....	101
Figure 4-4 Plug Type Detection Circuit.....	101
Figure 5-1 Interpretation of ordered sets.....	106
Figure 5-2 Transmit Order for Various Sizes of Data.....	107
Figure 5-3 USB Power Delivery Packet Format.....	108
Figure 5-4 CRC 32 generation.....	111
Figure 5-5 Line format of Hard Reset.....	113
Figure 5-6 Line format of Cable Reset.....	113
Figure 5-7 Inter-Frame Gap Timings.....	115
Figure 5-8 Transmitter Block Diagram.....	116
Figure 5-9 Receiver Block Diagram.....	116
Figure 5-10 Channel Diagram (Cable Type Detection not shown).....	117
Figure 5-11 Eye diagram of BFSK Modulation.....	120
Figure 5-12 BFSK Transmit Spectral Mask, given in absolute terms relative to the maximum value of v_{TX}	121
Figure 5-13 Line Format of Bit Stream.....	123
Figure 5-14 BMC Example.....	124
Figure 5-15 BMC Transmitter Block Diagram.....	124
Figure 5-16 BMC Receiver Block Diagram.....	125
Figure 5-17 BMC Encoded Start of Preamble.....	125
Figure 5-18 Transmitting or Receiving BMC Encoded Frame Terminated by Zero with High-to-Low Last Transition.....	126
Figure 5-19 Transmitting or Receiving BMC Encoded Frame Terminated by One with High-to-Low Last Transition.....	126
Figure 5-20 Transmitting or Receiving BMC Encoded Frame Terminated by Zero with Low to High Last Transition.....	126
Figure 5-21 Transmitting or Receiving BMC Encoded Frame Terminated by One with Low to High Last Transition.....	127
Figure 5-22 BMC Tx 'ONE' Mask.....	127
Figure 5-23 BMC Tx 'ZERO' Mask.....	128
Figure 5-24 BMC Rx 'ONE' Mask when Sourcing Power.....	130
Figure 5-25 BMC Rx 'ZERO' Mask when Sourcing Power.....	131
Figure 5-26 BMC Rx 'ONE' Mask when Power neutral.....	131
Figure 5-27 BMC Rx 'ZERO' Mask when Power neutral.....	132
Figure 5-28 BMC Rx 'ONE' Mask when Sinking Power.....	132
Figure 5-29 BMC Rx 'ZERO' Mask when Sinking Power.....	133
Figure 5-30 Transmitter Load Model for BMC Tx from a Source.....	134
Figure 5-31 Transmitter Load Model for BMC Tx from a Sink.....	134
Figure 5-32 Transmitter diagram illustrating zDriver.....	136
Figure 5-33 Example Multi-Drop Configuration showing two DRPs.....	137
Figure 5-34 Example Multi-Drop Configuration showing a DFP and UFP.....	137

Figure 5-35 Example implementation of the BIST generator and checker	139
Figure 5-36 Test Frame	139
Figure 5-37 Test Data Frame	141
Figure 6-1 USB Power Delivery Packet Format including Control Message Payload	142
Figure 6-2 USB Power Delivery Packet Format including Data Message Payload	143
Figure 6-3 Example Capabilities Message with 2 Power Data Objects	151
Figure 6-4 BIST Message	163
Figure 6-5 Vendor Defined Message	166
Figure 6-6 Discover Identity Command response	171
Figure 6-7 Example Discover SVIDs response with 3 SVIDs	179
Figure 6-8 Example Discover SVIDs response with 4 SVIDs	179
Figure 6-9 Example Discover SVIDs response with 12 SVIDs followed by an empty response	179
Figure 6-10 Example Discover Modes response for a given SVID with 3 Modes	180
Figure 6-11 Successful Enter Mode sequence	181
Figure 6-12 Enter Mode sequence Interrupted by Source Capabilities and then Re-run	182
Figure 6-13 Unsuccessful Enter Mode sequence due to NAK	182
Figure 6-14 Exit Mode sequence	183
Figure 6-15 Attention Command request/response sequence	184
Figure 6-16 Command request/response sequence	184
Figure 6-17 Enter/Exit Mode Process	186
Figure 6-18 Vendor Defined Message interrupted by a Power Delivery Message	187
Figure 6-19 Outline of States	206
Figure 6-20 References to states	206
Figure 6-21 Protocol Layer Message transmission	207
Figure 6-22 Protocol layer Message reception	210
Figure 6-23 Hard/Cable Reset	212
Figure 6-24 BIST Transmitter Test	215
Figure 6-25 BIST Receiver Test	217
Figure 7-1 Placement of Source Bulk Capacitance	224
Figure 7-2 Transition Envelope for Positive Voltage Transitions	225
Figure 7-3 Transition Envelope for Negative Voltage Transitions	226
Figure 7-4 Source V_{BUS} Response to Hard Reset	227
Figure 7-5 Application of v_{SrcNew} and $v_{SrcValid}$ limits after $t_{SrcReady}$	229
Figure 7-6 Source Peak Current Overload	230
Figure 7-7 Noise Spectral Mask, given in absolute terms relative to the maximum value of v_{TX}	231
Figure 7-8 v_{SafeDB} Operating Region	232

Figure 7-9 Placement of Sink Bulk Capacitance.....	234
Figure 7-10 Noise Spectral Mask, given in absolute terms relative to the maximum value of v_{TX}	236
Figure 7-11 Transition Diagram for Increasing the Current.....	240
Figure 7-12 Transition Diagram for Increasing the Voltage.....	242
Figure 7-13 Transition Diagram for Increasing the Voltage and Current.....	244
Figure 7-14 Transition Diagram for Increasing the Voltage and Decreasing the Current.....	246
Figure 7-15 Transition Diagram for Decreasing the Voltage and Increasing the Current.....	248
Figure 7-16 Transition Diagram for Decreasing the Current.....	250
Figure 7-17 Transition Diagram for Decreasing the Voltage.....	252
Figure 7-18 Transition Diagram for Decreasing the Voltage and the Current.....	254
Figure 7-19 Transition Diagram for a Sink Requested Power Role Swap.....	256
Figure 7-20 Transition Diagram for a Source Requested Power Role Swap.....	259
Figure 7-21 Transition Diagram for a GotoMin Current Decrease.....	262
Figure 7-22 Transition Diagram for a Source Initiated Hard Reset.....	264
Figure 7-23 Transition Diagram for a Sink Initiated Hard Reset.....	266
Figure 7-24 Transition Diagram for USB Type-B New Source Initiated Hard Reset and USB Type-A New Sink Receives Hard Reset Signaling.....	268
Figure 7-25 Transition Diagram for USB Type-B New Source Initiated Hard Reset and USB Type-A New Sink Does Not Receive Hard Reset Signaling.....	270
Figure 7-26 Transition Diagram for USB Type-A New Sink Initiated Hard Reset and USB Type-B New Source Receives Hard Reset Signaling.....	272
Figure 7-27 Transition Diagram for USB Type-A New Sink Initiated Hard Reset and USB Type-B New Source Does Not Receive Hard Reset Signaling.....	274
Figure 7-28 USB Type-A to USB Type-B Transition Diagram for Dead Battery Operation.....	276
Figure 7-29 Transition Diagram for no change in Current or Voltage.....	278
Figure 8-1 Example of daisy chained displays.....	289
Figure 8-2 Basic Message Exchange (Successful).....	292
Figure 8-3 Basic Message flow indicating possible errors.....	293
Figure 8-4 Basic Message Flow with Bad CRC followed by a Retry.....	295
Figure 8-5 Successful Power Negotiation.....	298
Figure 8-6 Successful GotoMin operation.....	302
Figure 8-7 Soft Reset.....	304
Figure 8-8 Source initiated Hard Reset.....	306
Figure 8-9 Sink Initiated Hard Reset.....	309
Figure 8-10 Source initiated reset - Sink long reset.....	312
Figure 8-11 USB Type-A or USB Type-B Successful Power Role Swap Sequence Initiated by the Source.....	316
Figure 8-12 USB Type-A or USB Type-B Successful Power Role Swap Sequence Initiated by the Sink.....	320

Figure 8-13 USB Type-A or USB Type-B Source initiated Hard Reset (Power Role Swapped)	324
Figure 8-14 USB Type-A or USB Type-B Sink Initiated Hard Reset (Power Role Swapped)	327
Figure 8-15 USB Type-C Successful Power Role Swap Sequence Initiated by the Source	331
Figure 8-16 USB Type-C Successful Power Role Swap Sequence Initiated by the USB Type-C Sink	336
Figure 8-17 USB Type-C Data Role Swap, UFP operating as Sink initiates	340
Figure 8-18 USB Type-C Data Role Swap, UFP operating as Source initiates.....	343
Figure 8-19 USB Type-C Data Role Swap, DFP operating as Source initiates.....	346
Figure 8-20 USB Type-C Data Role Swap, DFP operating as Sink initiates	349
Figure 8-21 USB Type-C Source to Sink VCONN Source Swap	352
Figure 8-22 USB Type-C Sink to Source VCONN Source Swap	355
Figure 8-23 Sink Gets Source's Capabilities	358
Figure 8-24 Dual-Role Source Gets Dual-Role Sink's Capabilities as a Source	360
Figure 8-25 Source Gets Sink's Capabilities	362
Figure 8-26 Dual-Role Sink Gets Dual-Role Source's Capabilities as a Sink.....	364
Figure 8-27 DFP to UFP Discover Identity.....	366
Figure 8-28 Source Port to Cable Plug Discover Identity	368
Figure 8-29 DFP to Cable Plug Discover Identity	370
Figure 8-30 DFP to UFP Enter Mode	372
Figure 8-31 DFP to UFP Exit Mode.....	375
Figure 8-32 DFP to Cable Plug Enter Mode.....	377
Figure 8-33 DFP to Cable Plug Exit Mode	380
Figure 8-34 UFP to DFP Attention	382
Figure 8-35 BIST Receiver Mode test.....	384
Figure 8-36 BIST Transmit Mode test.....	387
Figure 8-37 BIST Eye Pattern Test	390
Figure 8-38 BIST Test Data Test.....	392
Figure 8-39 Outline of States.....	395
Figure 8-40 References to states.....	395
Figure 8-41 Example of state reference with conditions	395
Figure 8-42 Example of state reference with the same entry and exit	396
Figure 8-43 Source Port Policy Engine state diagram	397
Figure 8-44 Sink Port state diagram.....	403
Figure 8-45 Source Port Soft Reset and Protocol Error State Diagram	407
Figure 8-46 Sink Port Soft Reset and Protocol Error Diagram	409
Figure 8-47 Source Port Ping State Diagram.....	410
Figure 8-48 Dual-Role (initially Source Port) Ping State Diagram	411

Figure 8-49 Dual-Role (initially Sink Port) Ping State Diagram.....	411
Figure 8-50 State Diagram for Hard Reset of P/C in Sink Role.....	412
Figure 8-51 State Diagram for the Hard Reset of a C/P in Source Role.....	413
Figure 8-52 Consumer/Provider Dead Battery/Power Loss State Diagram	414
Figure 8-53 BFSK Provider/Consumer Dead Battery/Power Loss State Diagram.....	417
Figure 8-54: USB Type-C DFP to UFP Data Role Swap State Diagram	419
Figure 8-55: USB Type-C UFP to DFP Data Role Swap State Diagram	421
Figure 8-56: Dual-Role Port in Source to Sink Power Role Swap State Diagram.....	424
Figure 8-57: Dual-role Port in Sink to Source Power Role Swap State Diagram.....	427
Figure 8-58 Dual-Role (Source) Get Source Capabilities diagram	429
Figure 8-59 Dual-Role (Source) Give Sink Capabilities diagram	430
Figure 8-60 Dual-Role (Sink) Get Sink Capabilities State Diagram	430
Figure 8-61 Dual-Role (Sink) Give Source Capabilities State Diagram.....	431
Figure 8-62 VCONN Swap State Diagram.....	432
Figure 8-63 UFP Structured VDM Discover Identity State Diagram	434
Figure 8-64 UFP Structured VDM Discover SVIDs State Diagram.....	435
Figure 8-65 UFP Structured VDM Discover Modes State Diagram.....	436
Figure 8-66 UFP Structured VDM Enter Mode State Diagram	437
Figure 8-67 UFP Structured VDM Exit Mode State Diagram.....	438
Figure 8-68 UFP VDM Attention State Diagram.....	439
Figure 8-69 DFP to UFP VDM Discover Identity State Diagram.....	440
Figure 8-70 DFP VDM Discover Identity State Diagram	441
Figure 8-71 DFP VDM Discover SVIDs State Diagram.....	442
Figure 8-72 DFP VDM Discover Modes State Diagram.....	443
Figure 8-73 DFP VDM Mode Entry State Diagram.....	444
Figure 8-74 DFP VDM Mode Exit State Diagram	446
Figure 8-75 DFP VDM Attention State Diagram.....	447
Figure 8-76 Cable Ready VDM State Diagram.....	447
Figure 8-77 Cable Plug Structured VDM Discover Identity State Diagram.....	448
Figure 8-78 Cable Plug Structured VDM Discover SVIDs State Diagram	449
Figure 8-79 Cable Plug Structured VDM Discover Modes State Diagram	450
Figure 8-80 Cable Plug Structured VDM Enter Mode State Diagram	451
Figure 8-81 Cable Plug Structured VDM Exit Mode State Diagram	452
Figure 8-82 Cable Plug Soft Reset State Diagram.....	453
Figure 8-83 Cable Plug Hard Reset State Diagram	454
Figure 8-84 DFP Soft Reset or Cable Reset of a Cable Plug State Diagram	454

Figure 8-85 UFP Source Soft Reset of a Cable Plug State Diagram	456
Figure 8-86 Source Startup Structured VDM Discover Identity State Diagram	457
Figure 8-87 BIST Receive Mode State Diagram.....	458
Figure 8-88 BIST Transmit Mode State Diagram	460
Figure 8-89 BIST Carrier Mode and Eye Pattern State Diagram.....	461
Figure 9-1 Example PD Topology	471
Figure 9-2 Mapping of PD Topology to USB.....	472
Figure 9-3 USB Attached to USB Powered State Transition.....	473
Figure 9-4 Any USB State to USB Attached State Transition (When operating as a Consumer)	474
Figure 9-5 Any USB State to USB Attached State Transition (When operating as a Provider)	474
Figure 9-6 Any USB State to USB Attached State Transition (After a USB Type-C Data Role Swap).....	475
Figure 9-7 Software stack on a PD aware OS	475
Figure 9-8 Enumeration of a PDUUSB Device	476
Figure 10-1 Source Power Rule Illustration	486
Figure 10-2 Source Power Rule Example.....	487
Figure C-1 Typical System Electrical Model.....	494
Figure C-2 Typical Synchronous Buck Power Stage with Parasitics.....	495
Figure C-3 Spurious Noise Measurement Test Setup	496
Figure C-4 Current Transients when Cable/Load Removed.....	497
Figure C-5 Isolation Inductor Energy versus Load.....	498
Figure C-6 Simplified Small Signal AC Model	499
Figure C-7 Power Stage Phase And Gain with and without Isolation Inductors.....	500
Figure C-8 Simplified Small Signal AC Model (Feedback before and after Inductor $z_{Isolation_P}$)	501
Figure C-9 Simplified Small Signal AC Model (Feedback before and after Inductor $z_{Isolation_P}$)	501
Figure D-1 USB 3.1 Standard-A Plug with USB 2.0 PD or 3.1 PD Standard-A Receptacle.....	502
Figure D-2 USB 3.1 PD Standard-A Plug with USB 2.0 or 3.1 Standard-A Receptacle	503
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.....	504
Figure D-4 USB 2.0 PD or 3.1 PD Standard-A Plug with USB 2.0 Standard-A Receptacle.....	505
Figure D-5 USB 2.0 Standard-A Plug with USB 2.0 or USB3.1 PD Standard-A Receptacle	506
Figure D-6 USB 2.0 Thin Card with USB 2.0 PD or 3.1 PD Standard-A Receptacle	507
Figure D-7 USB 3.1 Thin Card with USB 2.0 PD or 3.1 PD Standard-A Receptacle	508
Figure E-1 Squelch Budget	509
Figure F-1 External Power supplied downstream.....	511
Figure F-2 External Power supplied upstream	515
Figure F-3 Giving Back Power	522

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 [\[USB BC 1.2\]](#).

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 spinning 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 can 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 [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USB Type-C 1.2\]](#) or [\[USB BC 1.2\]](#) 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.

For USB 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.
- [\[USBBC 1.2\]](#) mechanisms for supplying higher power (not mandated by this specification).
- [\[USB Type-C 1.2\]](#) mechanisms for supplying higher power

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

- 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 [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USB Type-C 1.2\]](#) and [\[USB BC 1.2\]](#) specifications. It addresses only the elements required to implement USB Power Delivery. It is targeted at power supply vendors, manufacturers of [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USB Type-C 1.2\]](#) and [\[USB BC 1.2\]](#) Platforms, Devices and cable assemblies.

Normative information is provided to allow interoperability of components designed to this specification. Informative information, when provided, illustrates possible design implementations.

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**

Discard, **Discards** and **Discarded** are equivalent keywords 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.

1.4.2.4 **Ignored**

Ignore, **Ignores** and **Ignored** are equivalent keywords indicating Messages or Message fields which, when received, **Shall** result in no special action by the receiver. An **Ignored** Message **Shall** only result in returning a **GoodCRC** Message to acknowledge Message receipt. A Message with an **Ignored** field **Shall** be processed normally except for any actions relating to the **Ignored** field.

1.4.2.5 **Invalid**

Invalid is a keyword when used in relation to a Packet indicates that the Packet's usage or fields fall outside of the defined specification usage. When **Invalid** is used in relation to an Explicit Contract it indicates that a previously established Explicit Contract which can no longer be maintained by the Source. When **Invalid** is used in relation to individual K-codes or K-code sequences indicates that the received Signaling falls outside of the defined specification.

1.4.2.6 **May**

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

1.4.2.1 **May Not**

May Not is a keyword that is the inverse of **May**. Indicates a choice to not implement a given feature with no implied preference.

1.4.2.2 **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.

1.4.2.3 **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.

1.4.2.4 **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.

1.4.2.5 **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.

1.4.2.6 Shall Not

Shall Not is a keyword that is the inverse of **Shall** indicating non-compliant operation.

1.4.2.7 Should

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

1.4.2.8 Should Not

Should Not is a keyword is the inverse of **Should**; equivalent to the phrase “it is recommended that implementations do not...”.

1.4.2.1 Valid

Valid is a keyword that is the inverse of **Invalid** indicating either a Packet, Signaling that fall within the defined specification or an Explicit Contract that can be maintained by the Source.

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

- **[USB 2.0]** – Universal Serial Bus Specification, Revision 2.0, plus ECN and Errata http://www.usb.org/developers/docs/usb20_docs/.
- **[USB 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.
- **[USB BC 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.
- **[USBCBridge 1.0]** – Universal Serial Bus Type-C Bridging specification version 1.0 **[USBPDCompliance]** – USB Power Delivery Compliance Plan version 1.0 http://www.usb.org/developers/docs/devclass_docs/.
- **[USBOTG 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/EJ37.pdf>.
- **[USB Type-C 1.2]** – USB Type-C Specification www.usb.org/developers/docs
- **[IEC 60958-1]** IEC 60958-1 Digital Audio Interface Part:1 General Edition 3.0 2008-09 www.iec.ch
- **[IEC 63002]** Draft CD for IEC 63002 Identification and Communication Interoperability Method for External DC 851 Power Supplies Used With Portable Computing Devices.

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]**, **[USB Type-C 1.2]** and **[USB 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 Discover Identity Command to determine its characteristics in addition to other Structured VDM Commands (Electronically Marked Cable see [USB Type-C 1.2]).
Active Mode	A Mode which has been entered and not exited.
Alternate Mode	As defined in [USB Type-C 1.2] . Equivalent to Mode in the PD Specification.

Term	Description
Alternate Mode Adapter (AMA)	A PDUSB Device which supports Alternate Modes as defined in [USB Type-C 1.2] . Note that since an AMA is a PDUSB Device it has a single UFP that is only addressable by SOP Packets.
Atomic Message Sequence (AMS)	A fixed sequence of Messages as defined in Section 8.3.2 typically starting and ending in one of the following states: <i>PE_SRC_Ready</i> , <i>PE_SNK_Ready</i> or <i>PE_CBL_Ready</i> . An AMS can be Interruptible or Non-interruptible.
Attach	Mechanical joining of the Port Pair by a cable.
Attached	USB Power Delivery ports which are mechanically joined with USB cable.
Battery	A power storage device residing behind a Port that can either be a source or sink of power.
Battery Supply	A power supply that directly applies the output of a Battery to V_{BUS} . This is exposed by the Battery Supply PDO (see Section 6.4.1.2.5)
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"> • Logic 0 is indicated by a frequency $f_{Carrier} - f_{Deviation}$. • Logic 1 is indicated by a frequency $f_{Carrier} + f_{Deviation}$.
Biphase Mark Coding (BMC)	Modification of Manchester coding where each zero has one transition and a one has two transitions (see [IEC 60958-1]).
BIST	Built In Self-Test – Power Delivery testing mechanism for the PHY Layer.
BIST Data Object (BDO)	Data Object used by <i>BIST</i> Messages.
BIST Mode	A BIST receiver or transmitter test mode enabled by a <i>BIST</i> Message.
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 can 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 but does restore V_{CONN} to its attachment state.
Cold Socket	A DFP receptacle Port that does not apply <i>vSafe5V</i> on V_{BUS} until a plug insertion Sink is detected Attached.
Command	Request and response pair defined as part of a Structured Vendor Defined Message (see Section 6.4.4.2)
Configuration Channel (CC)	Single wire used by the BMC PHY Layer Signaling Scheme (see [USB Type-C 1.2]).
Connected	USB Power Delivery ports that have exchanged a Message and a <i>GoodCRC</i> Message response using the USB Power Delivery protocol so that both Port Partners know they that each is PD Capable.
Consumer	The capability of a PD Port (typically a Device’s UFP) to sink power from the power conductor (e.g. V_{BUS}). This corresponds to a Type- USB Type-B Port or a Type- USB Type-C Port with R_d 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- USB Type-B Port or a Dual-Role Type- USB Type-C Port with R_d asserted on its CC Wire.
Continuous BIST Mode	A BIST Mode where the Port or Cable Plug being tested sends a continuous stream of test data.
Contract	An agreement on both power level and direction reached between a Port Pair. A Contract can be explicitly negotiated between the Port Pair or can 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 can only be altered in the case of a (re-)negotiation, Power Role Swap, Data Role Swap, Hard Reset or failure of the Source.
Control Message	A Message is defined as a Control Message when the <i>Number of Data Objects</i> field in the Message Header is set to 0. The Control Message consists only of a Message Header and a CRC.

Term	Description
Data Message	A Data Message consists of a Message Header followed by one or more Data Objects. Data Messages are easily identifiable because the <i>Number of Data Objects</i> field in the Message Header is a non-zero value.
Data Object	32 bit object which contains information specific to different types of Data Message. Power, Request, BIST and Vendor Data Objects are defined.
Data Role Swap	Process of exchanging the DFP (Host) and UFP (Device) roles between Port Partners using the <i>[USB Type-C 1.2]</i> connector.
Dead Battery	A device has a Dead Battery when the Battery in a device is unable to power its functions.
Detach	Mechanical unjoining of the Port Pair by removal of the cable.
Detached	USB Power Delivery ports which are no longer mechanically joined with USB cable.
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)	Indicates the Port's position in the USB topology which typically corresponds to a USB Host root Port or Hub downstream Port as defined in <i>[USB Type-C 1.2]</i> . At connection the Port defaults to operation as a USB Host (when USB Communication is supported) and Source. Typically a Type-A Port on a Device as defined in <i>[USB 2.0]</i> , <i>[USB 3.1]</i> or Type-C Port as defined in <i>[USB Type-C 1.2]</i> . The default Host and Source.
Dual-Role Data (DRD)	Capability of operating as either a DFP or UFP.
Dual-Role Data Port	A Port Capable of operating as a DRD.
Dual-Role Power (DRP)	Capability of operating as either a Source or Sink.
Dual-Role Power Device	A product containing one or more Dual-Role Power Ports that are capable of operating as either a Source or a Sink.
Dual-Role Power Port	A Consumer/Provider or Provider/Consumer capable Port that is a Port capable of operating as a DRP.
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.
Error Recovery	Error recovery process as defined in <i>[USB Type-C 1.2]</i> .
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 <i>Accept</i> Message in response to a <i>Request</i> Message sent by a Sink followed by a <i>PS_RDY</i> Message indicating that the power supply is ready; this corresponds to the <i>PE_SRC_Ready</i> state for a Source Policy Engine and the <i>PE_SNK_Ready</i> state for a Sink Policy Engine. The Explicit Contract can be altered through the re-negotiation process. All Port pairs, except for those involving low power devices, are required to make an Explicit Contract.
Fixed Battery	A Battery that is not easily removed or replaced by an end user e.g. requires a special tool to access or is soldered in.
Fixed Supply	A well-regulated fixed voltage power supply. This is exposed by the Fixed Supply PDO (see Section 6.4.1.2.3)
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 _{BUS} 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. For Type- USB Type-C connectors it restores both Port Partners to their default Data Roles and returns the V _{CONN} Source to the Source Port.
HDD	A Hard Disk Drive.

Term	Description
ID Header VDO	The VDO in a <i>Discover Identity</i> ACK Command immediately following the VDM Header. The ID Header VDO contains information corresponding to the Power Delivery Product.
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.
Initiator	The initial sender of a Command request in the form of a query.
Interrupt	<ol style="list-style-type: none"> 1. The act of returning to the appropriate ready state, on receiving a Protocol Error, in order to process the incoming Message. See Interruptible. 2. Also applies to cases where PD Communication cannot be continued such as in the case of Detach.
Interruptible	An AMS that, on receiving a Protocol Error, returns to the appropriate ready state in order to process the incoming Message is said to be Interruptible. Every AMS is Interruptible until the first Message in the AMS has been sent (a <i>GoodCRC</i> Message has been received). An AMS of Vendor Messages is Interruptible during the entire sequence.
IoC	The negotiated current value as defined in [IEC 63002].
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 can 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 <i>Optionally</i> can be changed by a System Policy Manager.
Low Power	State in which Sources powered by e.g. a single cell Li Battery that want to minimize the power they output over V_{BUS} 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.
Message	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.
Message Header	Every Message starts with a 16-bit Message Header containing basic information about the Message and the PD Port's Capabilities.
Messaging	Communication in the form of Messages as defined in Section 6.
Modal Operation	State where there are one or more Active Modes. Modal Operation ends when there are no longer any Active Modes.
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 [USB Type-C 1.2].
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	<p>This is the PD process whereby:</p> <ol style="list-style-type: none"> 1. The Source advertises its capabilities. 2. The Sink requests one of the advertised capabilities. 3. The Source acknowledges the request and alters its output to satisfy the request. <p>The result of the negotiation is a Contract for power delivery/consumption between the Port Pair.</p>

Term	Description
Non-interruptible	An AMS that, on receiving a Protocol Error, generates either a Soft Reset or Hard Reset. Any power related AMS is Non-interruptible once the first Message in the AMS has been sent (a <i>GoodCRC</i> Message has been received).
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 [<i>USB Type-C 1.2</i>]). 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	See Connected.
PD Power (PDP)	The output power of a Source, as specified by the manufacturer and expressed in Fixed Supply PDOs as defined in Section 10.
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_{BUS} 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 (PE)	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_{BUS} .
Power Consumer	See Consumer
Power Data Object (PDO)	Data Object used to expose a Source Port's power capabilities or a Sink's power requirements as part of a <i>Source_Capabilities</i> or <i>Sink_Capabilities</i> Message respectively. Fixed, Variable and Battery Power Data Objects are defined.
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
Power Reserve	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.
Power Role Swap	Process of exchanging the Source and Sink roles between Port Partners.
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.

Term	Description
Product Type	Product categorization returned as part of the <i>Discover Identity</i> Command.
Product Type VDO	VDO identifying a certain Product Type in the ID Header VDO of a <i>Discover Identity</i> Command.
Protocol Error	An unexpected Message during an Atomic Message Sequence. A Protocol Error during a Non-interruptible AMS will result in either a Soft Reset or a Hard Reset. A Protocol Error during an Interruptible AMS will result in a return to the appropriate ready state where the Message will be handled. An unexpected Message during an atomic Message sequence.
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_{BUS}). This corresponds to a Type- USB Type-A Port or a Type- USB Type-C Port with R_p 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- USB Type-A Port or a Dual-Role Type- USB Type-C Port with R_p asserted on its CC Wire.
Rd	Pull-down resistor on the USB Type-C CC wire used to indicate that the Port is a Sink (see [USB Type-C 1.2]).
Reattach	Attach of the Port Pair by a cable after a previous Detach.
Re-negotiation	A process wherein one of the Port Partners wants to alter the negotiated Contract.
Request Data Object (RDO)	Data Object used by a Sink Port to negotiate a Contact as a part of a <i>Request</i> Message.
Re-run	Start an Interruptible AMS again from the beginning after a Protocol Error.
Responder	The receiver of a Command request sent by an Initiator that replies with a Command response.
Rp	Pull-up resistor on the USB Type-C CC wire used to indicate that the Port is a Source (see [USB Type-C 1.2]).
Safe Operation	Sources must have the ability to tolerate <i>vSafe5V</i> applied by both Port Partners.
Signaling	A Preamble followed by an ordered set of four K-codes used to indicate a particular line symbol e.g. <i>Hard Reset</i> as defined in Section 5.4.
Signaling Scheme	Physical mechanism used to transmit bits. BMC and BFSK Signaling Schemes are defined in this specification.
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_{BUS} ; most commonly a Device.
Soft Reset	A process that resets the PD communications engine to its default state.
SOP Communication	Communication using SOP Packets also implies that a Message sequence is being followed.
SOP Packet	Any Power Delivery Packet which starts with an <i>SOP</i> .
SOP* Communication	Communication with a Cable Plug using SOP* Packets, also implies a Message sequence is being followed.
SOP* Packet	A term referring to any Power Delivery Packet starting with either <i>SOP</i> , <i>SOP'</i> or <i>SOP''</i> .
SOP' Communication	Communication with a Cable Plug using SOP' Packets, also implies that a Message sequence is being followed.
SOP' Packet	Any Power Delivery Packet which starts with an <i>SOP'</i> used to communicate with a Cable Plug.
SOP'' Communication	Communication with a Cable Plug using SOP'' Packets, also implies that a Message sequence is being followed.
SOP'' Packet	Any Power Delivery Packet which starts with an <i>SOP''</i> used to communicate with a Cable Plug when SOP' Packets are being used to communicate with the other Cable Plug.
Source	A role a Port is currently taking to supply power over V_{BUS} ; most commonly a Host or Hub DFP downstream port.
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".

Term	Description
Start of Packet (SOP)	K-code marker used to delineate the start of a packet. Three start of packet sequences are defined: <i>SOP</i> , <i>SOP'</i> and <i>SOP''</i> , with <i>SOP*</i> used to refer to all three in place of <i>SOP/SOP'/SOP''</i> .
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 (SPM)	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.
Test Frame	Frame consisting of a Preamble, <i>SOP*</i> , followed by test data (See Section 5.9).
Test Pattern	Continuous stream of test data in a given sequence (See Section 5.9)
Tester	The Tester is assumed to be a piece of test equipment that manages the BIST testing process of a PD UUT.
Unexpected Message	Message that a Port supports but has been received in an incorrect state.
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.
Unrecognized Message	Message that a Port does not understand e.g. a Message using a <i>Reserved</i> Message type, a Message defined by a higher specification Revision than the Revision this Port supports, or an Unstructured Message for which the VID is not recognized.
Unsupported Message	Message that a Port recognizes but does not support. This is a Message defined by the specification but which is not supported by this Port.
Upstream Facing Port (UFP)	Indicates the Port's position in the USB topology typically a Port on a Device as defined in [USB Type-C 1.2] . At connection the Port defaults to operation as a USB Device (when USB Communication is supported) and Sink. Typically a B Port on a Device as defined in [USB 2.0] , [USB 3.1] or Type-C Port as defined in [USB Type-C 1.2] . The default Device and Sink.
USB Attached State	Synonymous with the [USB 2.0] and [USB 3.1] 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 [USB 2.0] , [USB 3.1] , [USB Type-C 1.2] or [USB BC 1.2] .
USB Device	Either a hub or a peripheral device as defined in [USB 2.0] and [USB 3.1] .
USB Host	The host computer system where the USB host controller is installed as defined in [USB 2.0] and [USB 3.1] .
USB Powered State	Synonymous with the [USB 2.0] and [USB 3.1] definition of the powered state.
USB Safe State	State of the Type- USB Type-C connector when there are pins to be re-purposed (see [USB Type-C 1.2]) so they are not damaged by and do not cause damage to their Port Partner.
USB 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 USB Type- USB Type-A and USB Type- USB Type-B.
USB 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 USB Type-A and USB Type-B.
USB Type-C	Term used to refer to the USB Type-C connector plug or receptacle as defined in [USB Type-C 1.2] .
USB-IF PD SID (PD SID)	Standard ID allocated to this specification by the USB Implementer's Forum.
Variable Supply	A very poorly regulated power supply that is not a Battery. This is exposed by the Variable Supply PDO (see Section 6.4.1.3.2).
VCONN Powered Accessory	An accessory that is powered from VCONN to operate in a Mode (see [USB Type-C 1.2]).
VCONN Source	The USB Type-C Port responsible for sourcing VCONN.
V _{CONN} Swap	Process of exchanging the VCONN Source between Port Partners.
VDM Header	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).
Vendor Data Object (VDO)	Data Object used to send Vendor specific information as part of a <i>Vendor_Defined</i> Message.

Term	Description
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.
VI	Same as power (i.e. voltage * current = power)
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. USB Type-C connectors can support the CC wire as the communications channel and in addition can support V_{BUS} communication but not concurrently. USB Type-A and USB 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 [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USB Type-C 1.2\]](#) or [\[USB BC 1.2\]](#) 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 [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USB Type-C 1.2\]](#) or [\[USB BC 1.2\]](#) 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 USB 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 [\[USB Type-C 1.2\]](#) 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 can also include discover of SVIDs, Modes and the entering/exiting of Modes supported by the cable assembly.

2.2 Section Overview

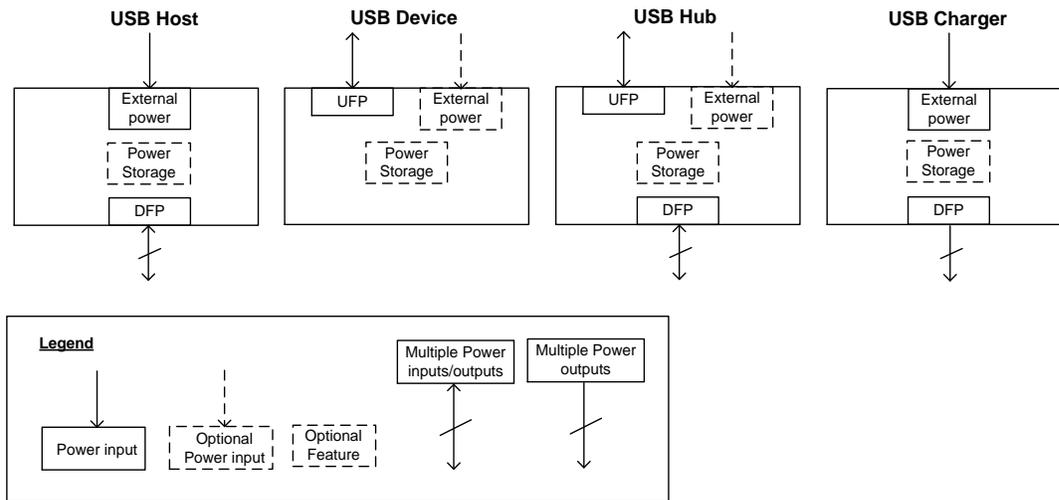
This specification contains the following sections:

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.
Section 10	Rated Output Power definitions for PD
Appendix A	<i>Deprecated</i>
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
 - **Optionally** source power (a Dual-Role Power Device).
 - **Optionally** communicate via USB.
 - Communicate using SOP Packets.
 - **Optionally** Communicate using SOP* Packets.
- DFPs that:
 - Source power
 - **Optionally** Sink power (a Dual-Role Power Device).
 - **Optionally** communicate via USB.
 - Communicate using SOP Packets
 - **Optionally** Communicate using SOP* Packets.
- A Source that can 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 can 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).
- A Vconn Source that:

- Can be either Port Partner, either the DFP/UFP or Source/Sink
- Powers the Cable Plug(s)

2.4 SOP* Communication

2.4.1 Introduction

The Start of Packet (or SOP) is used as an addressing scheme to identify whether the Communications were intended for one of the Port Partners (SOP Communication) or one of the Cable Plugs (SOP'/SOP'' Communication). SOP/SOP' and SOP'' are collectively referred to as SOP*. 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 could or could not be physically located in the plug.

The following sections describe how this addressing scheme operates for Port to Port and Port to Cable Plug Communication.

2.4.2 SOP* Collision Avoidance

Collision avoidance is coordinated by the DFP. Messages **Shall Not** be initiated by cables. Use of *tCableMessage* timing prevents collisions during PD Communications and enables the UFP time slots to communicate.

2.4.3 SOP Communication

SOP Communication is used for Port to Port communication between the Source and the Sink. SOP Communication is recognized by both Port Partners but not by any intervening Cable Plugs. SOP Communication takes priority over other SOP* Communications since it is critical to complete power related operations as soon as possible. Message sequences relating to power are also allowed to Interrupt other sequences to ensure that negotiation and control of power is given priority on the bus.

2.4.4 SOP'/SOP'' Communication with Cable Plugs

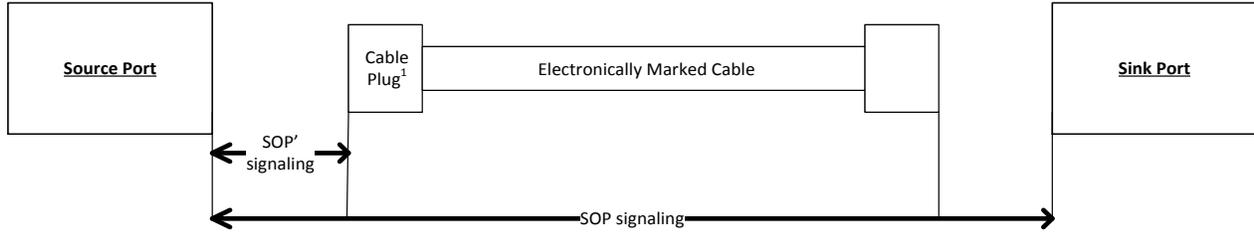
SOP' Communication is recognized by electronics in one Cable Plug (which could be Attached to either the UFP or DFP). SOP Communication between the Port Partners is not recognized by the Cable Plug. Note: that the term Cable Plug in the SOP' Communication case is used to represent a logical entity in the cable which is capable of PD Communication and which could or could 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.

All 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 from being blocked. For a product which does not recognize SOP/SOP' or SOP'' Packets, this will look like a non-idle channel, leading to 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 a Soft or Hard Reset.

When no 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 a Cable Plug using SOP' Packets in order to discover 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 **Discards** any SOP' Packets received.

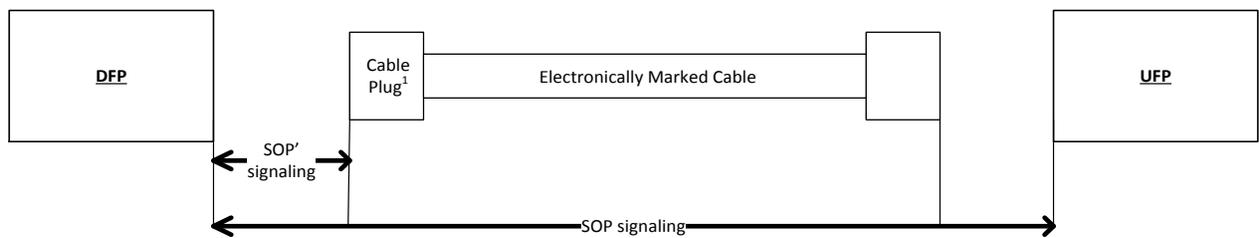
When an Explicit Contract is in place the DFP (either the Source or the Sink) can communicate with the Cable Plug(s) using SOP'/SOP'' Packets (see Figure 2-3). During this phase all communication with the Cable Plug is initiated and controlled by the DFP which acts to prevent conflicts between SOP* Packets.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

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



¹ Cable Plug can be physically attached to either the Source or Sink Port.

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



¹ Cable Plug can be physically attached to either the DFP or UFP.

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. At Attach the Source Port (the Port with Rp asserted see [USB Type-C 1.2]) is also the DFP and VCONN Source. At Attach the Sink Port (the Port with Rd asserted) is also the UFP and is not the VCONN Source.

The Source/Sink roles, DFP/UFP roles and VCONN Source role can all subsequently be swapped orthogonally to each other. A Port that supports both Source and Sink roles is called a Dual-Role Power Port (DRP). A USB Type-C Port that supports both DFP and UFP roles is called a Dual-Role Data Port (DRD).

When USB Communications Capability is supported in the DFP role then the Port will also be able to act as a USB Host. Similarly when USB Communications Capability is supported in the UFP role then the Port will also be able to act as a USB Device.

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 Source Operation

The Source 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.
 - For a Source-only Port the Source detects Sink Attachment.
 - For a DRP that toggles the Port becomes a Source Port on Attachment of a Sink
 - The Source then typically sets V_{BUS} to *vSafe5V*.
- Before PD Connection (no PD Connection or PD Contract):
 - Prior to sending *Source_Capabilities* Messages the Source can detect the type of cabling Attached and can 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 default capability of a USB Type-C cable is 3A, but SOP' Communication is used to discover other capabilities of the cable.
 - The Source periodically advertises its capabilities by sending *Source_Capabilities* Messages every *tTypeCSendSourceCap* or *tSendSourceCap*.
- Establishing PD Connection (no PD Connection or Contract):
 - Presence of a PD Capable Port Partner is detected either:
 - By receiving a *GoodCRC* Message in response to a *Source_Capabilities* Message.
 - By receiving *Hard Reset* Signaling.
- Establishing Explicit Contract (PD Connection but no Explicit Contract or Implicit Contract after a Power Role Swap):
 - The Source receives a *Request* Message from the Sink 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 does not generate SOP' or SOP'' Packets, is not required to detect SOP' or SOP'' Packets and **Discards** them.
- During PD Connection (Explicit Contract - **PE_SRC_Ready** State):
 - The Source processes and responds (if a response is required) to all Messages received and sends appropriate Messages whenever its Local Policy requires.
 - The Source informs the Sink whenever its capabilities change, by sending a **Source_Capabilities** Message.
 - The Source will always have Rp asserted on its CC wire.
 - When this Port is a DRP the Source can initiate or receive a request for the exchange of power roles. After the Power Role Swap this Port will be a Sink and an Implicit Contract will be in place until an Explicit Contract is negotiated immediately afterwards.
 - When this Port is a DRD the Source can initiate or receive a request for an exchange of data roles. After a Data Role Swap the DFP (Host) becomes a UFP (Device). The Port remains a Source and the VCONN Source role (or not) remains unchanged.
 - The Source can 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 Port remains a Source and DFP/UFP roles remain unchanged.
 - When the Source Port is also a DFP
 - The Source can initiate Unstructured or Structured VDMs.
 - The Source can control the entry and exiting of modes in the Sink and control Modal Operation using Structured VDMs.
 - If SOP Packets are received by the Source, during SOP' or SOP'' Communication, the SOP' or SOP'' Communication is immediately terminated (the Cable Plug times out and does not retry)
 - If the Source 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.
 - The Source can control the entry and exiting of modes in the Cable Plug(s) and control Modal Operation.
 - When the Source Port is part of a multi-port system:
- Will issue GotoMin requests when the power reserve is needed. Detach or Communications Failure
 - A Source detects plug Detach and takes V_{BUS} down to **vSafe5V** within **tSafe5V** and **vSafe0V** within **tSafe0V** (i.e. using USB Type-C Detach detection via CC).
 - When the Source detects the failure to receive a **GoodCRC** Message in response to a Message within **tReceive:**
 - Leads to a Soft Reset, within **tSoftReset** of the **CRCReceiveTimer** expiring.
 - If the soft reset process cannot be completed a Hard Reset will be issued within **tHardReset** of the **CRCReceiveTimer** to restore V_{BUS} to USB Default Operation within ~1-1.5s.
 - When the Source is also the VCONN Source, VCONN will also be power cycled during the Hard Reset.
 - Receiving no response to further attempts at communication is interpreted by the Source as an error (see Error handling).
 - Errors during power transitions will automatically lead to a Hard Reset in order to restore power to default levels.
- Error handling
 - 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. A Hard Reset:
 - 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.

- Restores the Port's data role to DFP.
 - When the Sink is the VCONN Source it removes VCONN then the Source Port is restored as the VCONN Source.
 - 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 *nHardResetCount* further Hard Resets are carried out before the Source performs additional Error Recovery steps, as defined in [USB Type-C 1.2], by entering the *ErrorRecovery* state.

2.5.2 Sink Operation

- At Attach (no PD Connection or Contract):
 - Sink detects Source Attachment through the presence of *vSafe5V*.
 - For a DRP that toggles the Port becomes a Sink Port on Attachment of a Source.
 - Once the Sink detects the presence of *vSafe5V* on V_{BUS} it waits for a *Source_Capabilities* Message indicating the presence of a PD capable Source.
 - If the Sink 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 Sink does not generate SOP' or SOP'' Packets, is not required to detect SOP' or SOP'' Packets and does not recognize them.
- Establishing PD Connection (no PD Connection or Contract):
 - The Sink receives a *Source_Capabilities* Message and responds with a *GoodCRC* Message.
 - The Sink does not generate SOP' or SOP'' Packets, is not required to detect SOP' or SOP'' Packets and *Discards* them.
- Establishing Explicit Contract (PD Connection but no Explicit Contract or Implicit Contract after a Power Role Swap):
 - The Sink receives a *Source_Capabilities* Message from the Source and responds with a *Request* Message. If this is a *Valid* request the Sink receives an *Accept* Message followed by a *PS_RDY* Message when the Source's power supply is ready to source power at the agreed level. At this point the Source and Sink have entered into an Explicit Contract.
 - The Sink Port can request one of the capabilities offered by the Source, even if this is the *vSafe5V* output offered by *[USB 2.0]*, *[USB 3.1]*, *[USB Type-C 1.2]* or *[USB 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 Sink does not generate SOP' or SOP'' Packets, is not required to detect SOP' or SOP'' Packets and *Discards* them.
- During PD Connection (Explicit Contract – *PE_SNK_Ready* state)
 - The Sink processes and responds (if a response is required) to all Messages received and sends appropriate Messages whenever its Local Policy requires.
 - A Sink whose power needs have changed indicates this to the Source with a new *Request* Message. The Sink Port can 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]*, *[USB Type-C 1.2]* or *[USB BC 1.2]*, in order to enable future power negotiation.
 - Not requesting any capability with a *Request* Message results in an error.
 - 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).
 - The Sink will always have R_d asserted on its CC wire.

- When this Port is a DRP the Sink can Initiate or receive a request for the exchange of data roles. After a Data Role Swap the DFP (Host becomes a UFP (Device)). The Port remains a Sink and VCONN Source role (or not) remains unchanged.
- The Sink can 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 Port remains a Sink and DFP/UFP roles remain unchanged.
- When the Sink Port is also a DFP
 - The Sink can initiate Unstructured or Structured VDMs.
 - The Sink can control the entry and exiting of modes in the Source and control Modal Operation using Structured VDMs.
- Detach or Communications Failure
 - 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 a Hard Rest, Power Role Swap or Fast Role Swap.
 - A Sink detects plug removal and discharges V_{BUS} .
 - When the Sink detects the failure to receive a *GoodCRC* Message in response to a Message within *tReceive*:
 - Leads to a Soft Reset, within *tSoftReset* of the *CRCReceiveTimer* expiring.
 - If the soft reset process cannot be completed a Hard Reset will be issued within *tHardReset* of the *CRCReceiveTimer* to restore V_{BUS} to USB Default Operation within ~1-1.5s.
 - Receiving no response to further attempts at communication is interpreted by the Sink as an error (see Error handling).
 - Errors during power transitions will automatically lead to a Hard Reset in order to restore power to default levels.
- Error handling
 - 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. Sink, DFP/UFP, VCONN Source) and does not cause an exit from Modal Operation.
 - Serious errors are handled by *Hard_Reset* Signaling issued by either Port Partner. A Hard Reset:
 - 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.
 - Restores the Port's data role to UFP.
 - When the Sink is the VCONN Source it removes VCONN then the Source Port is restored as the VCONN Source.
 - 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 *tTypeCSinkWaitCap*. 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.

2.5.3 Cable Plugs

- Cable Plugs are powered when VCONN is present but are not aware of the status of the Contract.
- Cable Plugs do not initiate message sequences and only respond to messages sent to them.
- Detach or Communications Failure:
 - Communications can be interrupted at any time
 - There is no communication timeout scheme between the DFP and Cable Plug
 - The Cable Plug is ready to respond to potentially repeated requests.
- Error handling:
 - The Cable Plug detects *Hard Reset* Signaling to determine that the Source and Sink have been reset and will need to reset itself (equivalent to a power cycle).
 - The Cable Plug cannot generate *Hard Reset* Signaling itself.
 - The Hard Reset process power cycles both V_{BUS} and VCONN so this is expected to reset the Cable Plugs by itself.
 - A Cable Plug detects *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.

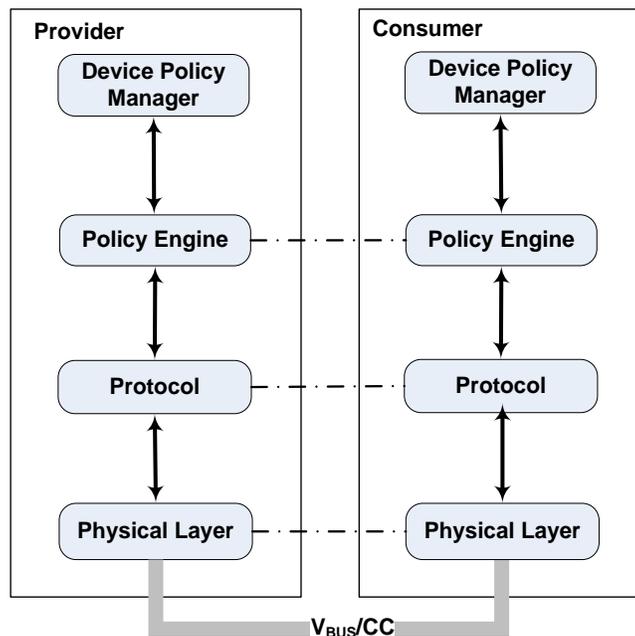
This section outlines the high level logical architecture of USB Power Delivery referenced throughout this specification. In practice various implementation options are possible based on many different possible types of PD device. PD devices can have many different configurations e.g. USB or non-USB 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 2-4 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 2-4 USB Power Delivery Communications Stack



Additionally USB Power Delivery devices which can operate as USB devices can communicate over USB (see Figure 2-5). 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 2-5 USB Power Delivery Communication Over USB

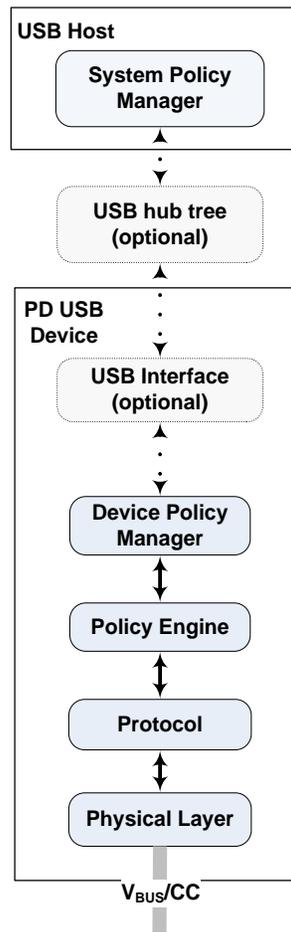


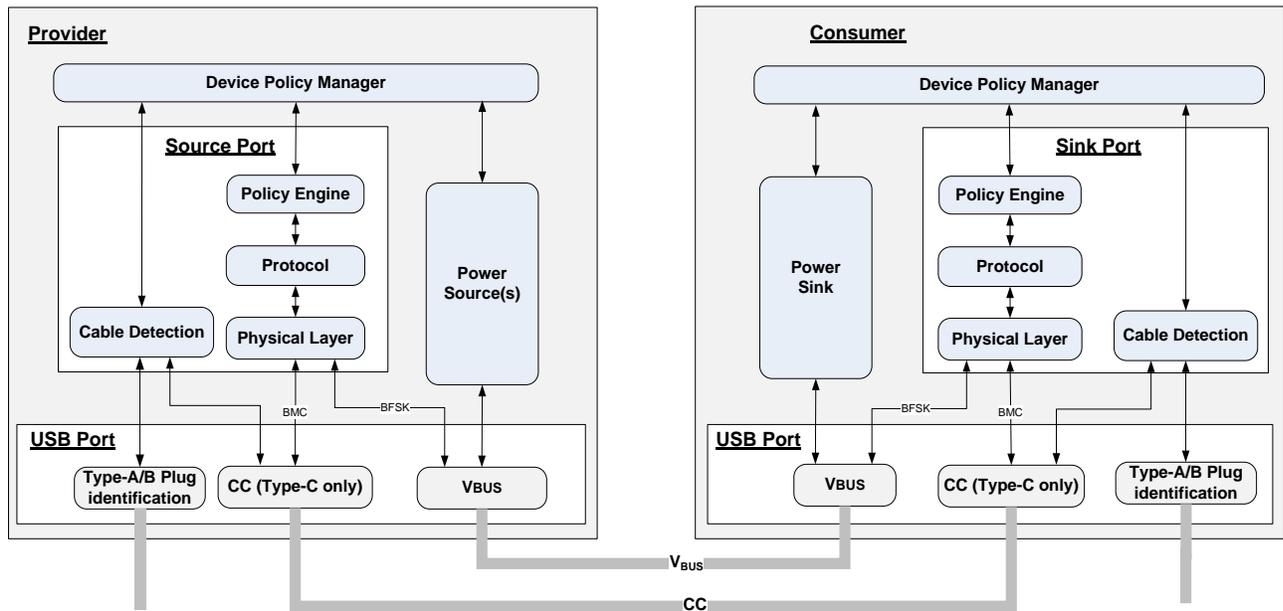
Figure 2-6 shows the logical blocks between two Attached PD ports. In addition to the communication stack described above there are also:

- For a Provider or Dual-Role Power Device: one or more **Sources** providing power to one or more ports.
- For a Consumer or Dual-Role Power Device: a **Sink** consuming power.
- A **Cable Detection** module (see Section 4.5) that:
 - Detects presence of V_{BUS} for Sink Ports
 - Identifies the type of PD cable Attached
- USB Power Delivery uses of either USB Type-A and USB Type-B **PD Cabling** is defined in Section 3 or standard cabling defined in [\[USB 2.0\]](#), [\[USB 3.1\]](#), and [\[USB Type-C 1.2\]](#).

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 2-6 illustrates a Provider and a Consumer. Dual-Role Power 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 2-6 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 can 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 can asynchronously request the Provider's present capabilities and can 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 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 Collision Avoidance

The DFP coordinates PD communications in order to avoid collisions.

2.6.4 Power supply

2.6.4.1 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.

2.6.4.2 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.

2.6.4.3 Dual-Role Power Ports

Dual-Role ports can be either a Dual-Role Power Device meaning that they have the ability to operate as either a Source or a Sink. A Provider/Consumer is a Port that can operate as either a Source Port (default) or a Sink Port. A Consumer/Provider 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.

2.6.4.4 Dead Battery or Lost Power Detection

[\[USB Type-C 1.2\]](#) defines mechanisms intended to communicate with and charge a Provider/Consumer with a Dead Battery. A Provider/Consumer can also use this mechanism to get power when they are disconnected from their normal power source. All Consumer/Providers support this mechanism.

2.6.5 DFP/UFP

2.6.5.1 Downstream Facing Port (DFP)

The Downstream Facing Port or DFP is equivalent in the USB topology to the USB A-Port. The DFP will also correspond to the USB Host but only if USB Communication is supported while acting as a DFP. Products such as Wall Warts can be a DFP while not having USB Communication capability. The DFP also acts as the bus master when controlling alternate mode operation.

2.6.5.2 Upstream Facing Port (UFP)

The Upstream Facing Port or UFP is equivalent in the USB topology to the USB B-Port. The UFP will also correspond to the USB Device but only if USB Communication is supported while acting as a UFP. Products which charge can be a UFP while not having USB Communication capability.

2.6.5.3 Dual-Role Data Ports

Dual-Role Data Ports have the ability to operate as either a DFP or a UFP and to swap between the two roles using Data Role Swap. Note that products can be Dual-Role Data Ports without being Dual-Role Power ports i.e. they can switch logically between DFP and UFP roles even if they are Source-only or Sink-only Ports.

2.6.6 VCONN Source

One Port, initially the Source Port, is the VCONN Source. The Cable Plugs use this supply to determine which Cable Plug is SOP'. The responsibility for sourcing VCONN can be swapped between the Source and Sink Ports in a make before break fashion to ensure that the Cable Plugs continue to be powered. To ensure reliable communication with the Cable Plugs prior to a Power Role Swap or Data Role Swap each Port needs to ensure that it is the VCONN Source if it needs to communicate with the Cable Plugs after the swap.

2.6.7 Cable and Connectors

2.6.7.1 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 can 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 [\[USB Type-C 1.2\]](#) specifications.

For USB Type-A and USB 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 USB Type-A and USB Type-B PD capable cables.

For USB Type-C connectors, PD uses the certified USB cables and associated detection mechanisms as defined in [\[USB Type-C 1.2\]](#).

2.6.8 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 [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USB BC 1.2\]](#) or [\[USB Type-C 1.2\]](#) specifications are applied.

There are two primary cases to consider:

- The Host (DFP/Source) 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 [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USB BC 1.2\]](#) or [\[USB Type-C 1.2\]](#) specifications.
- The Device (UFP/Sink) is non-PD and as such will not see any advertisements and therefore will not respond. The Host (DFP/Source) will continue to supply *vSafe5V* to V_{BUS} as specified in the [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USB BC 1.2\]](#) or [\[USB Type-C 1.2\]](#) specifications.

2.6.9 Power Rules

Power Rules define voltages and current ranges that are offered by USB Power Delivery Sources and used by a USB Power Delivery Sink for a given value of PD Power. See Section 10 for further details.

3. USB Type-A and USB 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 [\[USB BC 1.2\]](#) 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 USB Type-A and USB 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 USB Type-C connectors are defined in [\[USB Type-C 1.2\]](#) 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 [\[USB BC 1.2\]](#).

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 can result in unacceptable V_{GND_DROP} , restricting USB 2.0 signaling capability.

3.1.4 Hardwired Captive PD Cable

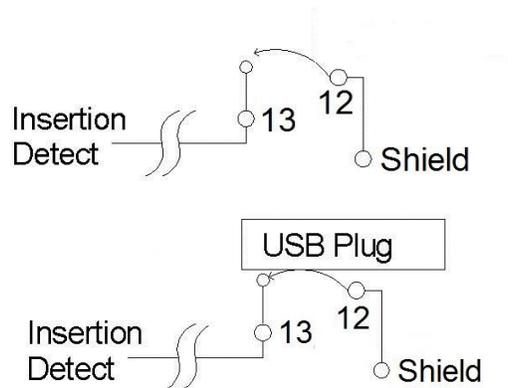
A hardwired captive PD cable assembly has a single PD plug at one end and a single hardwired (non-removable) connection at the other end. A hardwired captive PD cable assembly is PD compliant if it meets the requirements of this specification at the connector end cable marking or marking detection, negotiation 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 Section 3.6.1, Section 3.6.2 and Section 4.2.2 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

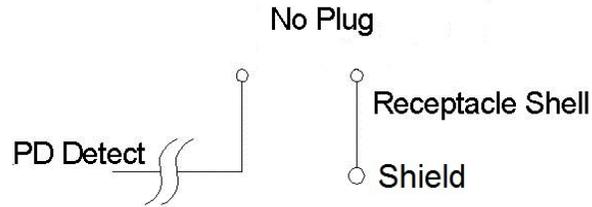


Figure 3-3 Non-PD Plug Standard-A Detection Circuit

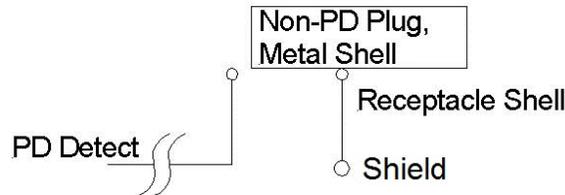


Figure 3-4 USB Thin Card Standard-A Detection Circuit

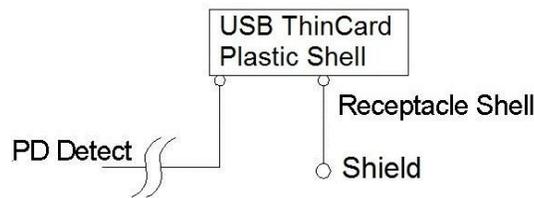
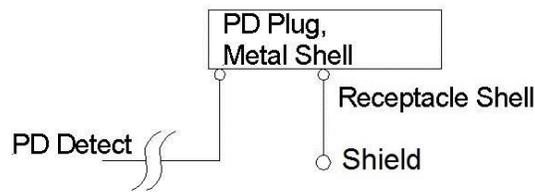


Figure 3-5 PD Plug Standard-A Detection Circuit



See Appendix D for mechanical details of different connector mating situations. See Section 3.6.1 and Section 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.

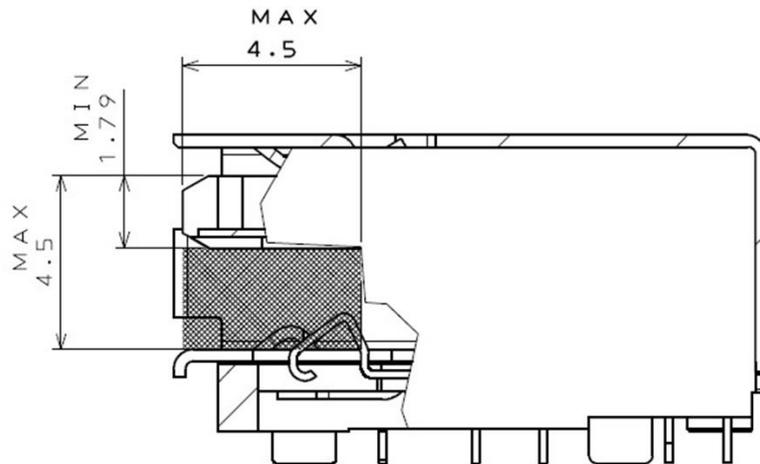
3.2 Connector Mating Interfaces

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

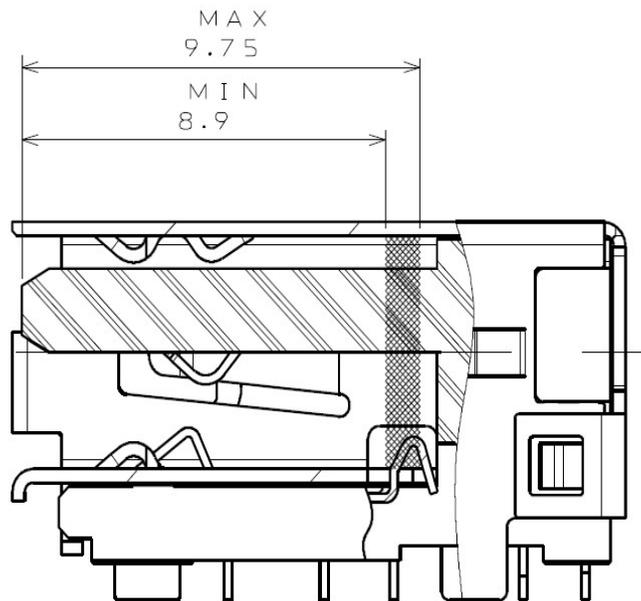


3.2.2 USB PD Standard-A PD Detect Mechanical Requirement

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.

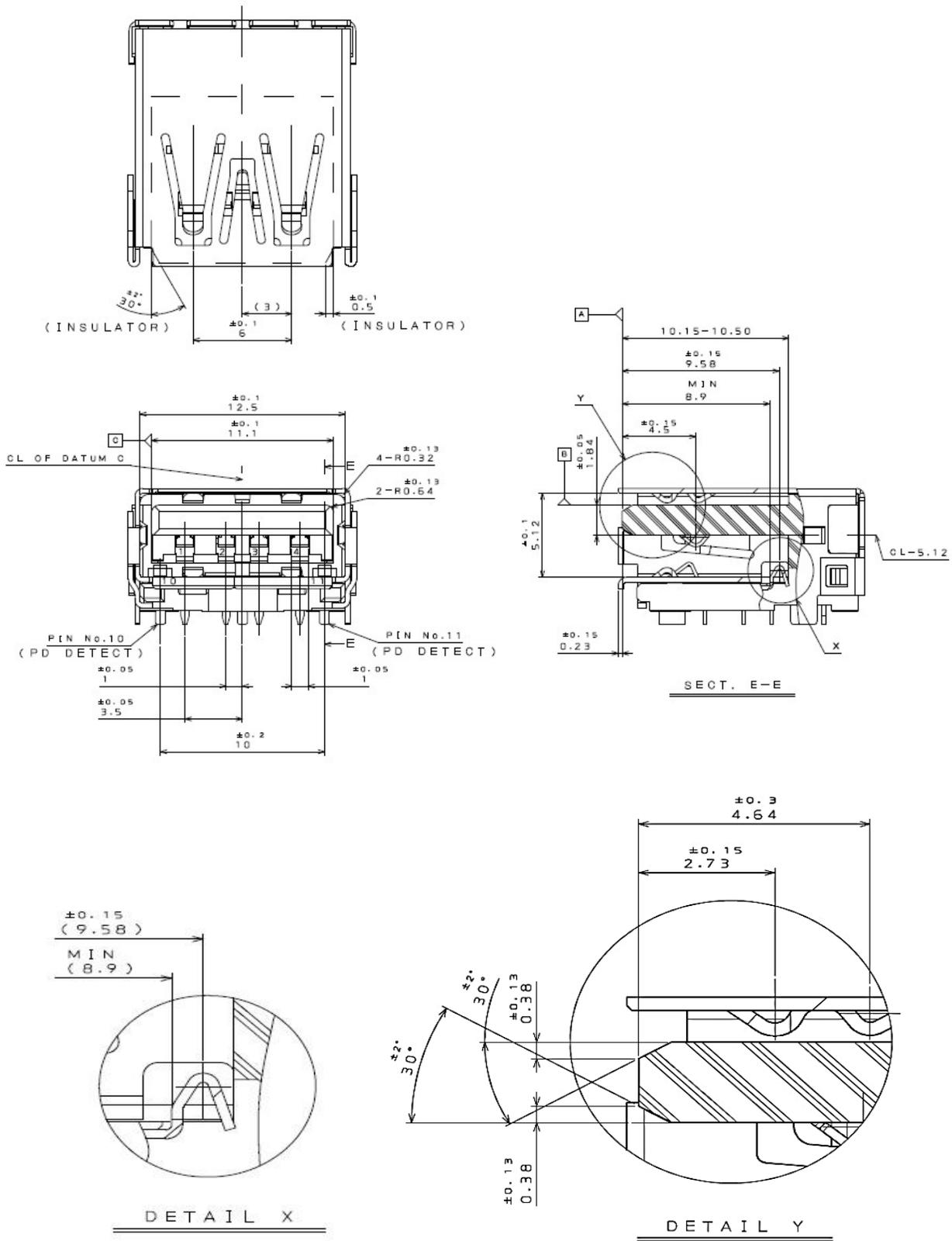
Figure 3-9 defines the **Normative** mechanical requirements the USB 2.0 PD Standard-A receptacle that govern the mating interoperability and informative dimensions for solder tail locations, a sample Insertion Detect implementation, and a sample PD Detect implementation. Dimensions associated with solder 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 could 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 3-9 USB 2.0 PD Standard-A Receptacle Interface Dimensions



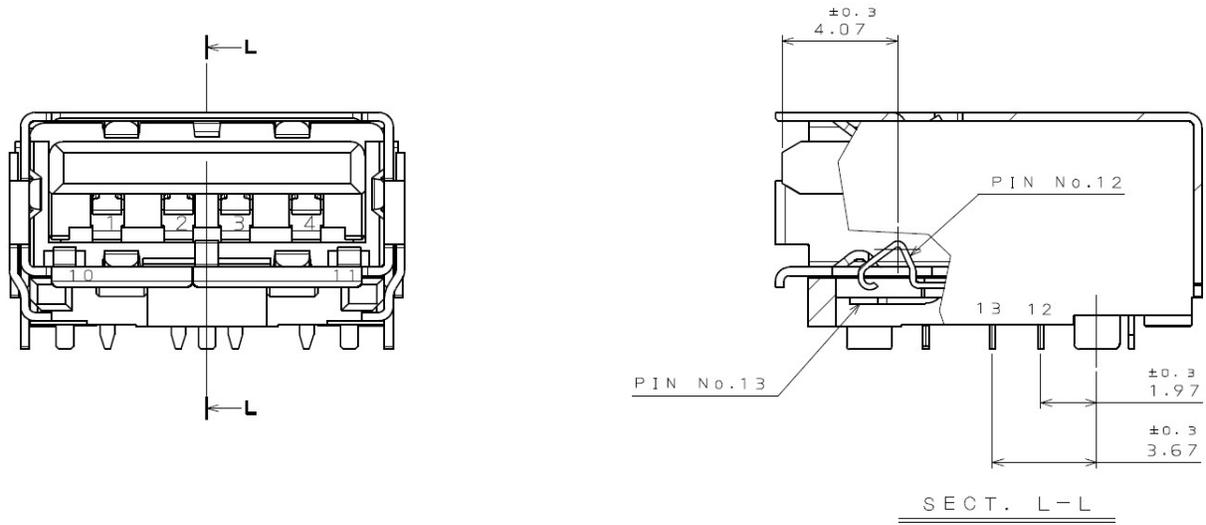
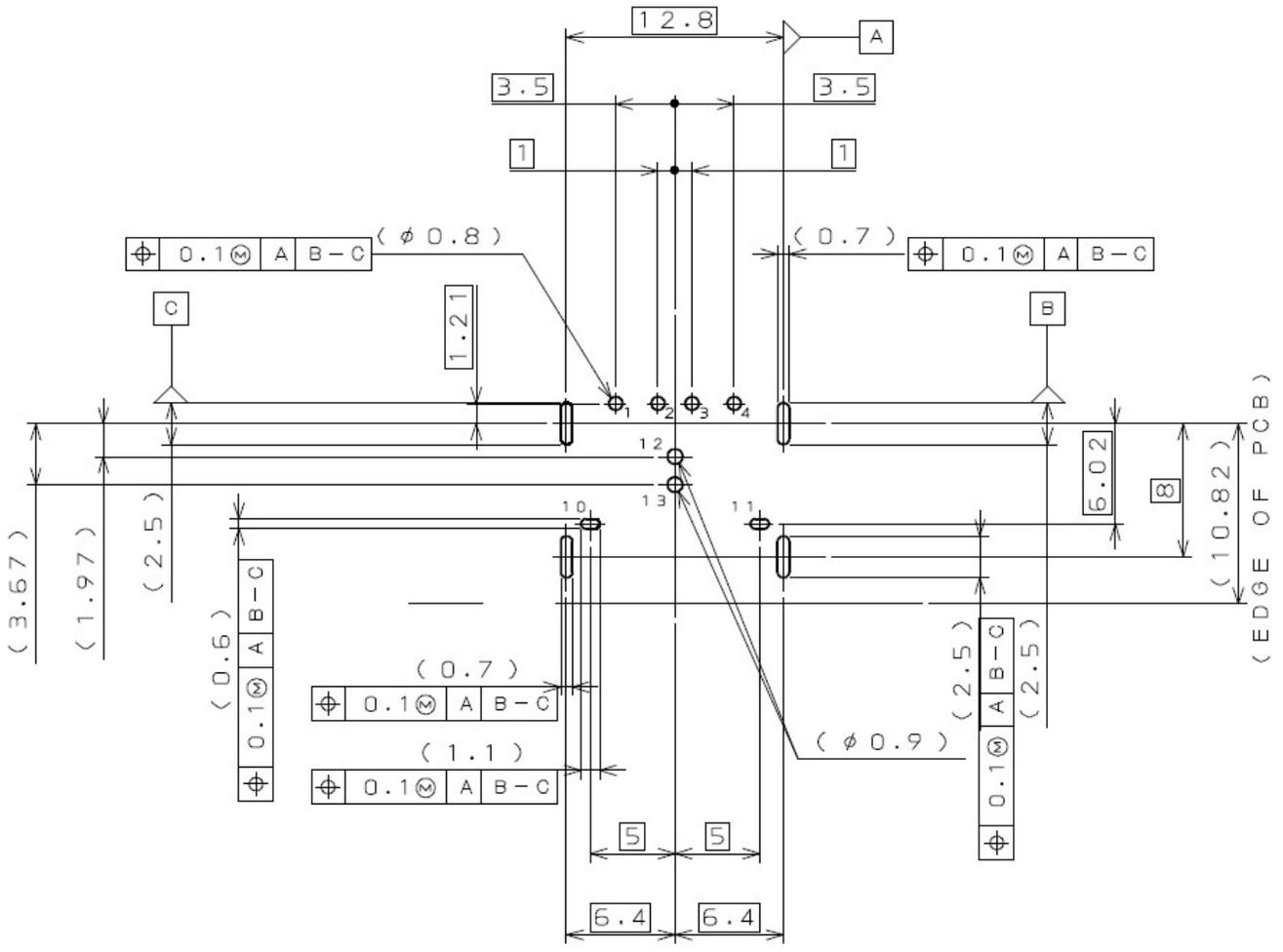


Figure 3-10 Reference Footprint for the USB 2.0 PD Standard-A Top Mount Single Receptacle (Informative)



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 ¹	Signal Name	Description	Mating Sequence
1	V _{BUS}	Power	Third
2	D-	Differential pair as defined in [USB 2.0]	Fourth
3	D+		
4	GND	Ground for power return	Third
10 ²	PD DETECT 1	Contact in PD receptacle to detect a PD plug	Last
11 ²	PD DETECT 2	Contact in PD receptacle to detect a PD plug	Last
12 ³ ,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: Implementation of PD DETECT shall include:</p> <ol style="list-style-type: none"> Either pin 10 or pin 11. Both pin 10 and pin 11. <p>Note 3: Pin 12, if present, shall be connected to Shield.</p>			

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

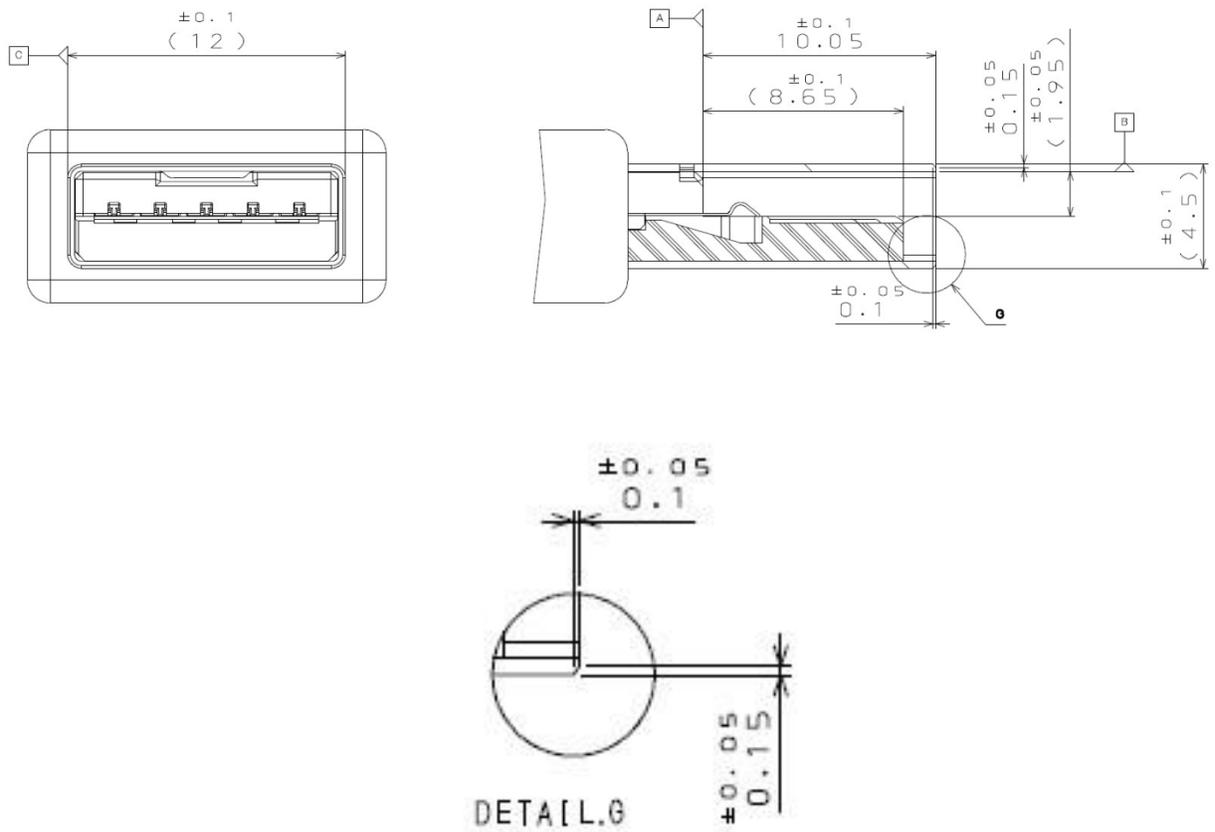


Figure 3-12 Reference USB 3.1 PD Standard-A Receptacle Interface Dimensions (Informative)

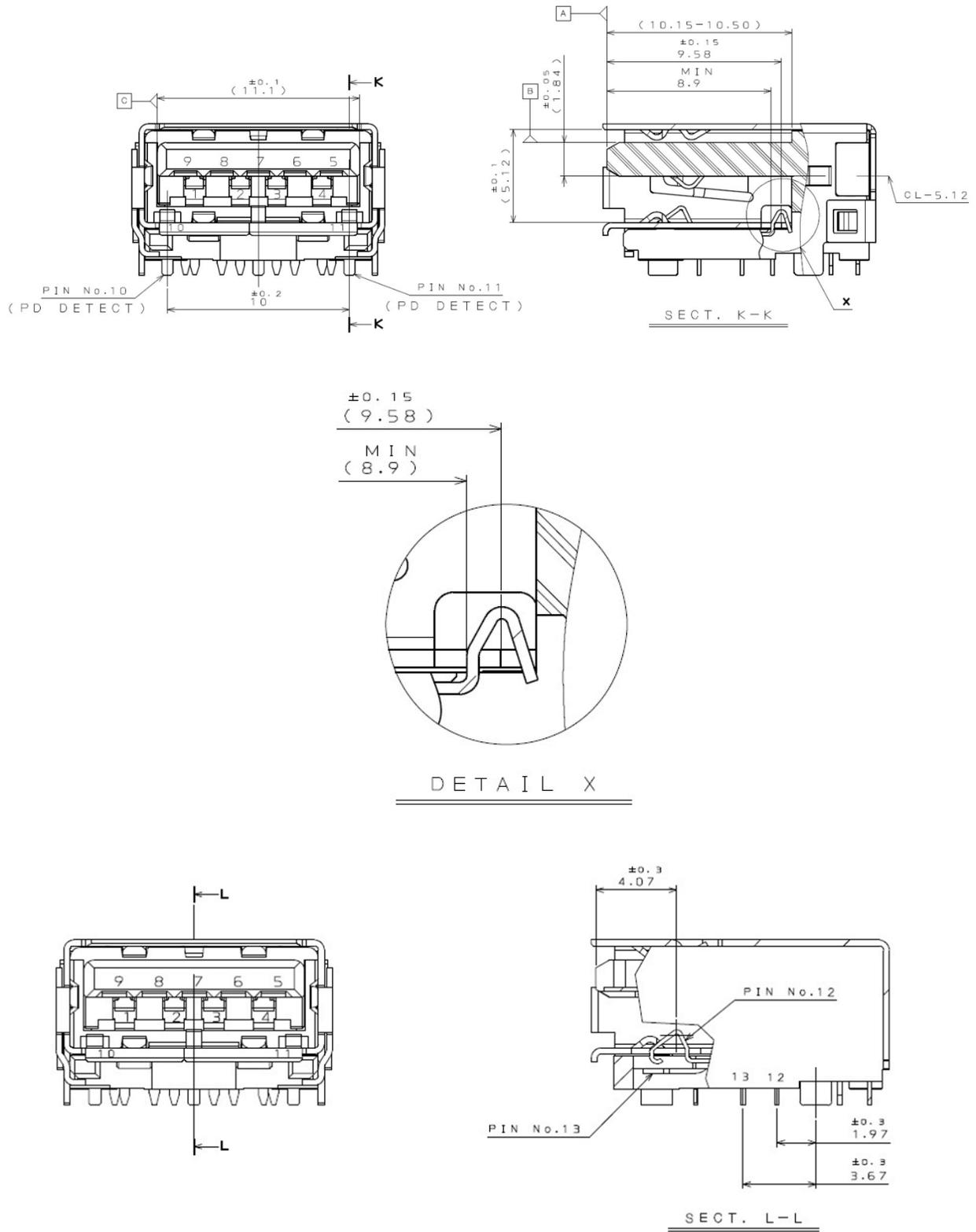
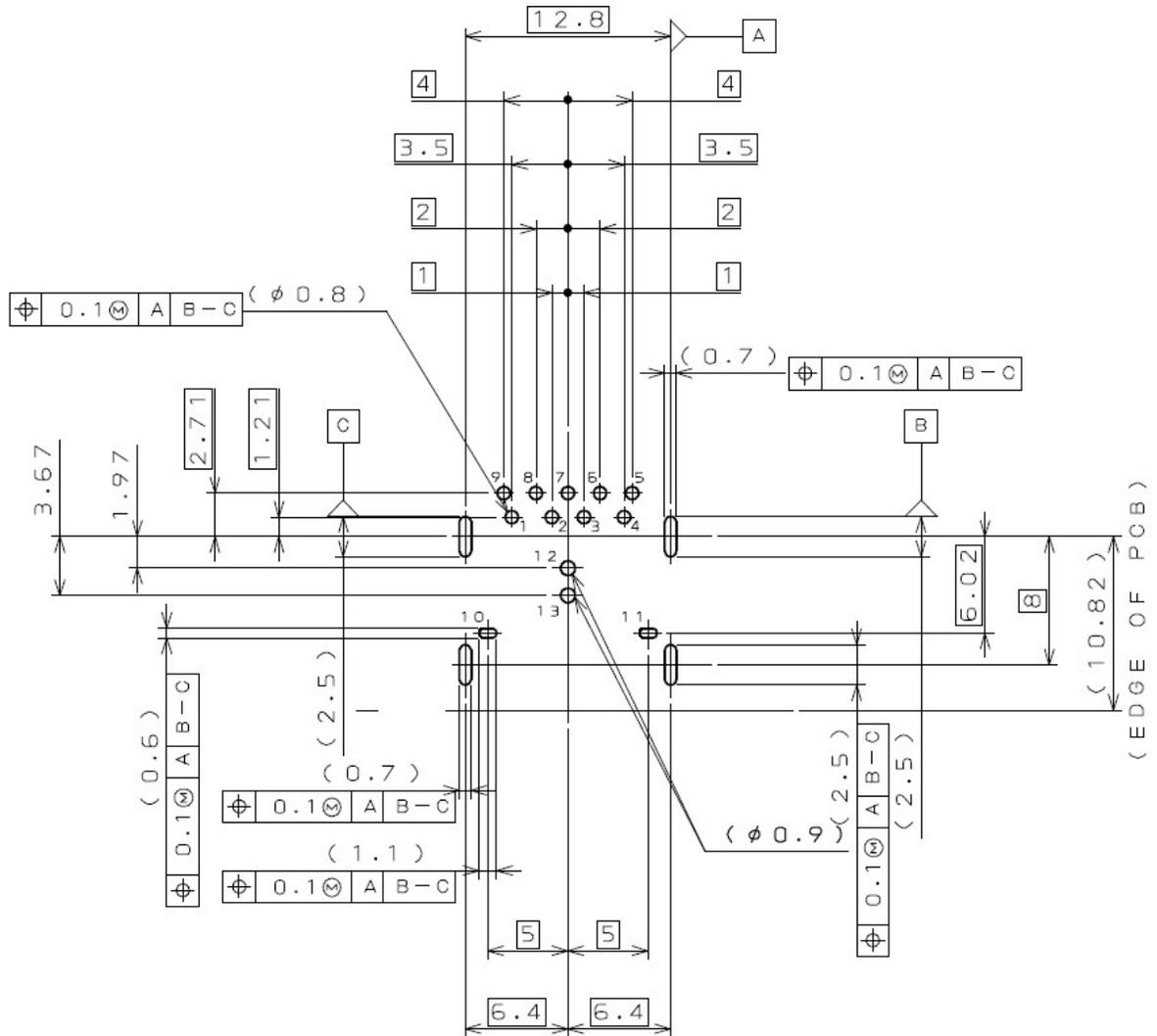


Figure 3-13 Reference Footprint for the USB 3.1 PD Standard-A Top Mount Single Receptacle (Informative)



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 ¹	Signal Name ²	Description	Mating Sequence
1	V _{BUS}	Power	Third
2	D-	Differential pair as defined in [USB 2.0]	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	Fifth
9	StdA_SSTX+		
10 ³	PD DETECT 1	Contact in PD receptacle to detect a PD plug	
11 ³	PD DETECT 2	Contact in PD receptacle to detect a PD plug	Last
12 ⁴ , 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"> a) Either pin 10 or pin 11. b) Both pin 10 and pin 11. <p>Note 4: Pin 12, if present, shall be connected to Shield.</p>			

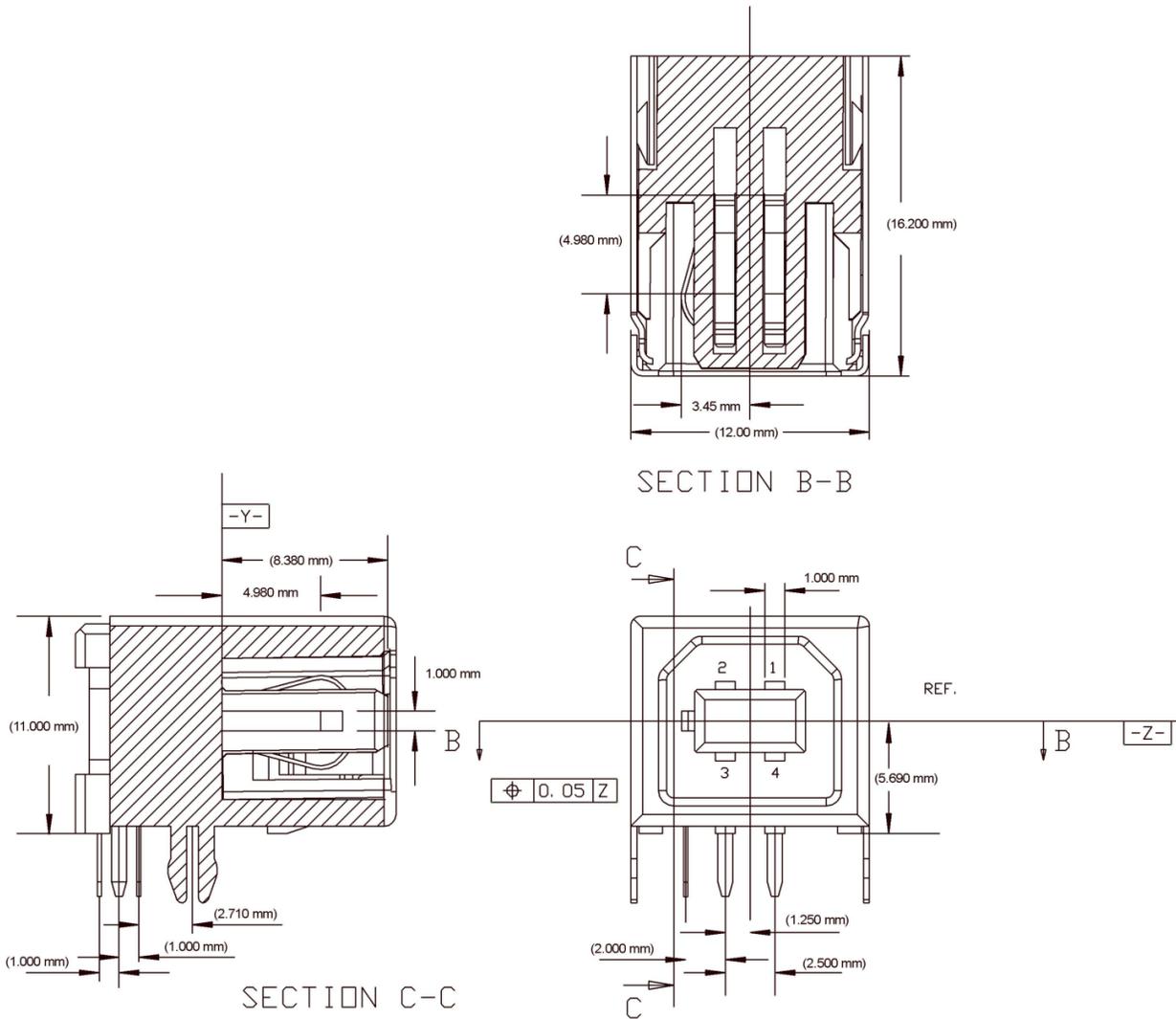
The physical location of the pins in the connector is illustrated in Figure 3-12. Note: pins 1 to 4 are referred to as the USB 2.0 pins, while pins 5 to 9 are referred to as the Enhanced SuperSpeed pins.

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



GENERAL TOLERANCE IS +/- 0.10
UNLESS OTHERWISE SPECIFIED

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

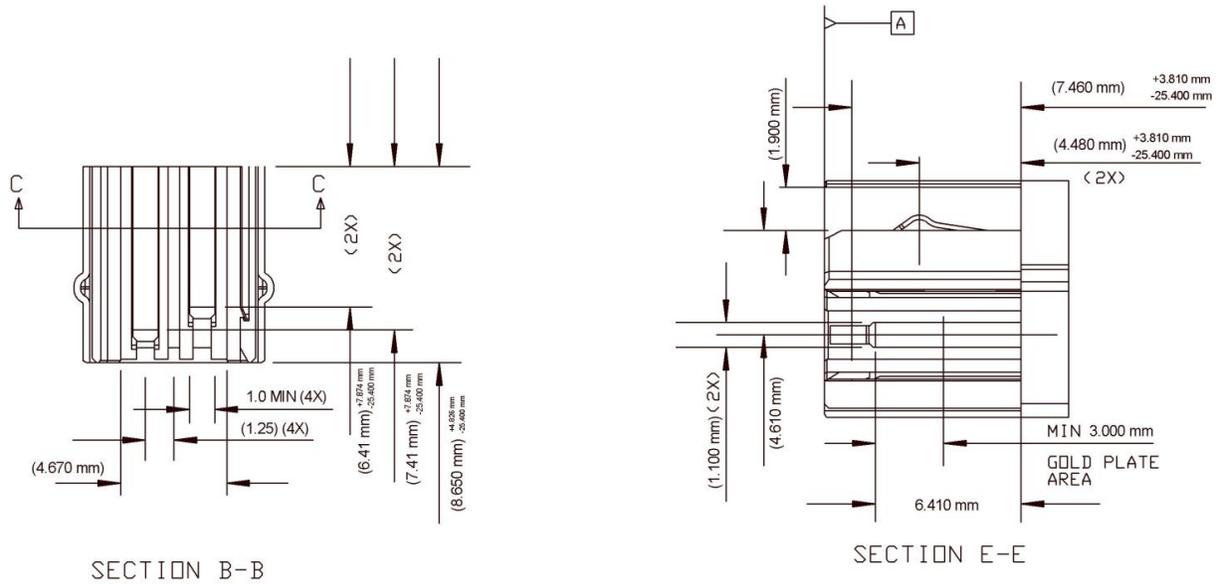
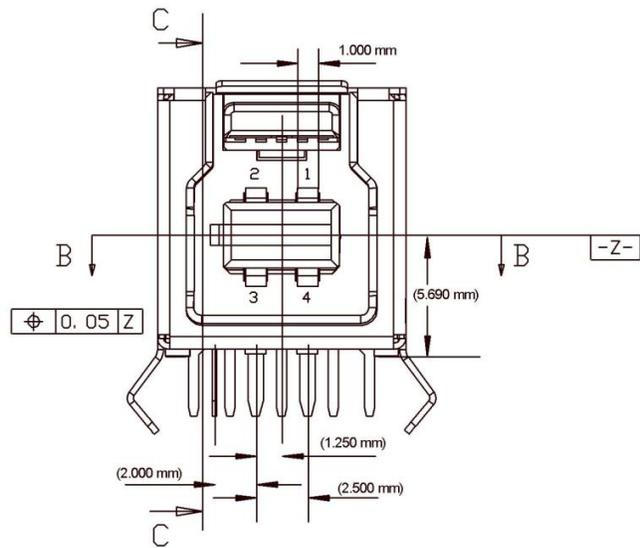
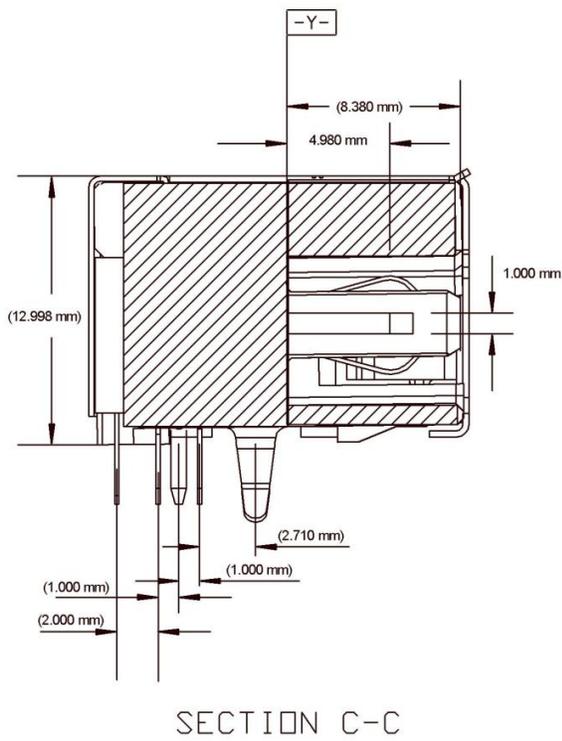
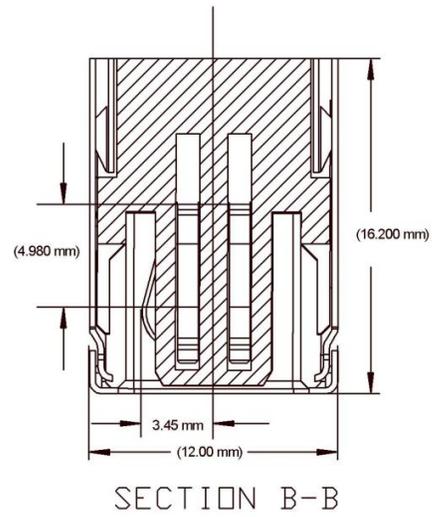


Figure 3-18 USB 3.1 PD Standard-B Receptacle Interface Dimensions



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 ¹	Signal Name ²	Description	Mating Sequence
1	V _{BUS}	Power	Second
2	D-	Differential pair as defined in [USB 2.0]	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 ³	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 [USB 3.1] 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 [\[USB BC 1.2\]](#).

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.

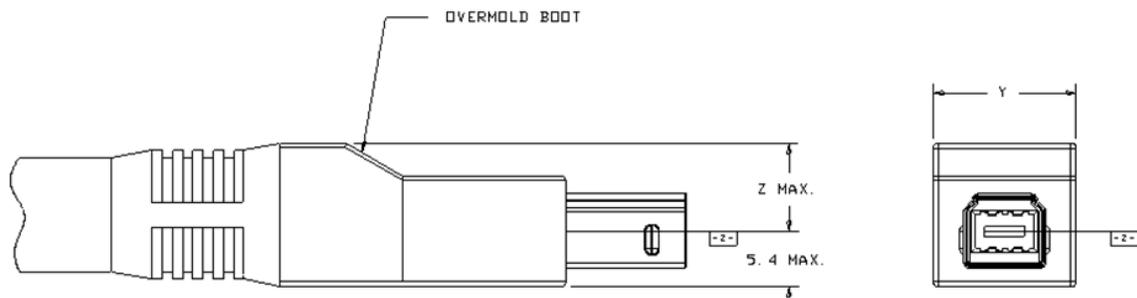
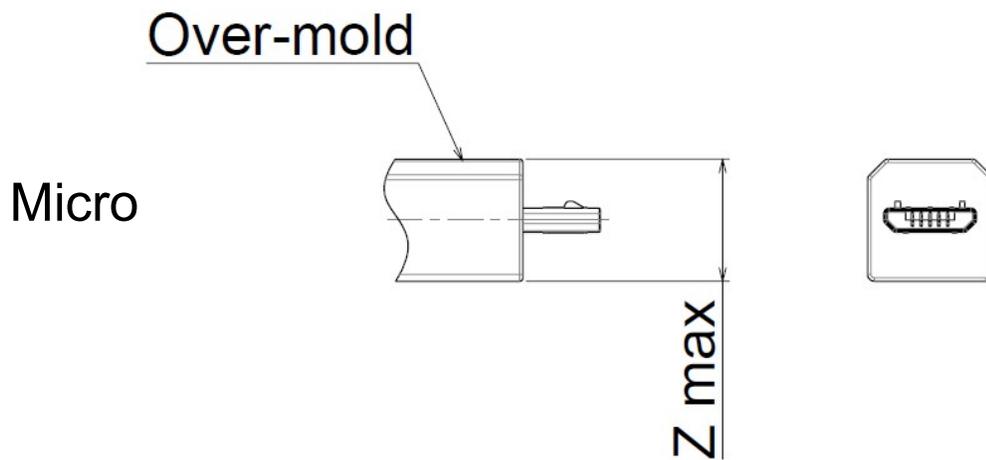
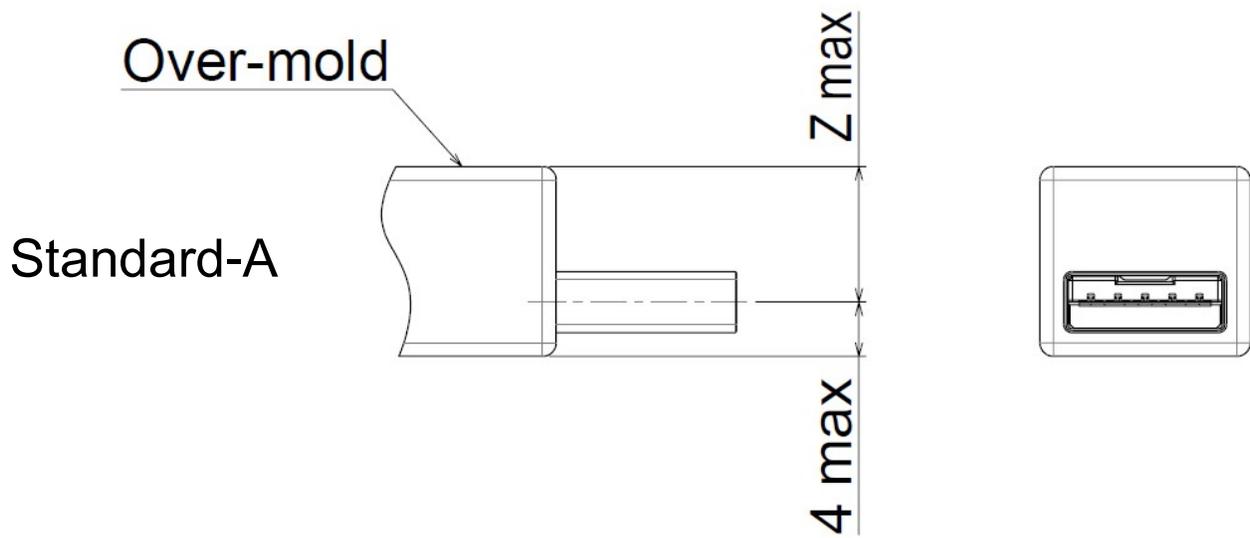
3.3.3 PD Cable Assembly Overmold Requirements

Depending on the PD cable assembly current rating and length, the wire gauge to meet the IR drop defined in 3.6.9 could 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 can result in mechanical interferences (e.g., could 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.

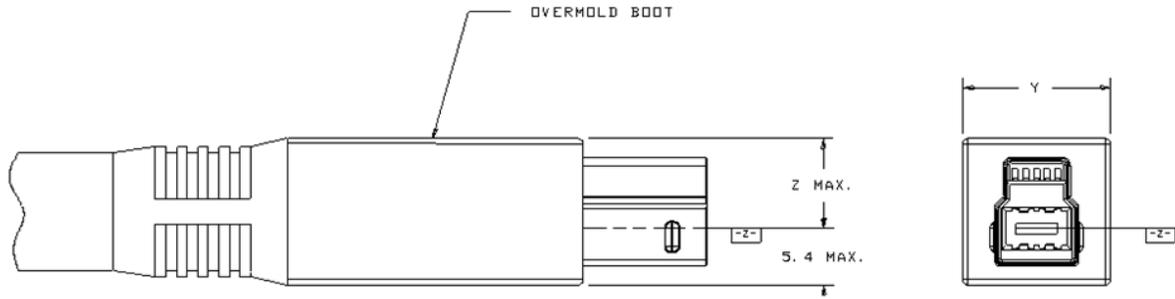
Table 3-6 USB PD Cable Assembly Overmold Maximum Dimensions

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

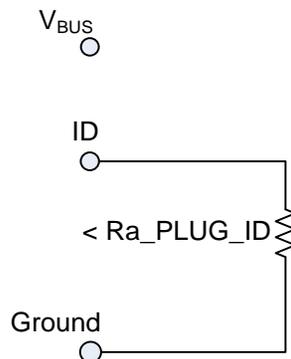


Figure 3-21 shows the schematic diagram of the Micro-A plug termination in a legacy cable assembly. Note the ID pin is connected to ground through an impedance of Ra_PLUG_ID (as defined in the Micro-USB Cables and Connectors specification v1.01 in [USB 2.0]) to indicate that the plug is an A plug. For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

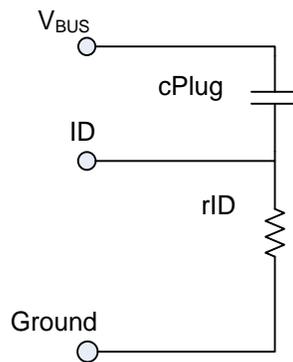
impedance to ground **shall** be maintained. However, any value less than *rID* is assumed to be interpreted by a legacy Port as a Micro-A plug.

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 Section 4.5.4 for additional information regarding Low Power Device characteristics.

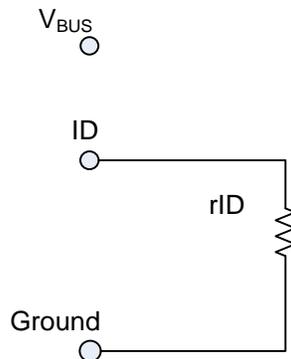
Figure 3-22 Schematic of a Micro-A Plug Marker Indicating Low Power Capability



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

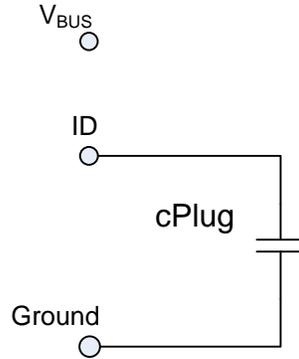
Electronic markers for PD cable assemblies with PD Standard-B plug connectors and Micro-B plug connectors are specified in this section. A legacy Micro-B connector's ID pin exhibits a very high resistance to ground. To maintain backward compatibility, capacitors are used as markers.

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.

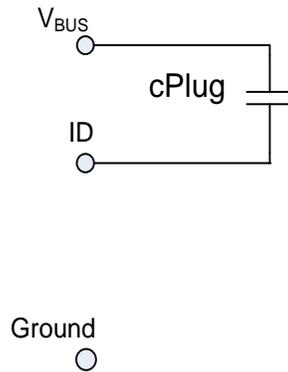
Figure 3-24 Schematic of a B Plug Connector Marker Indicating 3A Capability



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 [\[USB BC 1.2\]](#) 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 could be designed for use as power only (i.e., no USB data communication). Requirements that relate to the transfer of USB data might 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 Ω (Max) initial for V_{BUS} and GND contacts.
- Maximum change (delta) of +10m Ω 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 Ω (Max) initial for V_{BUS} and GND contacts.
- Maximum change (delta) of +10m Ω 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 Ω (Max) initial.
- Maximum change (delta) of 300m Ω 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 Ω 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.

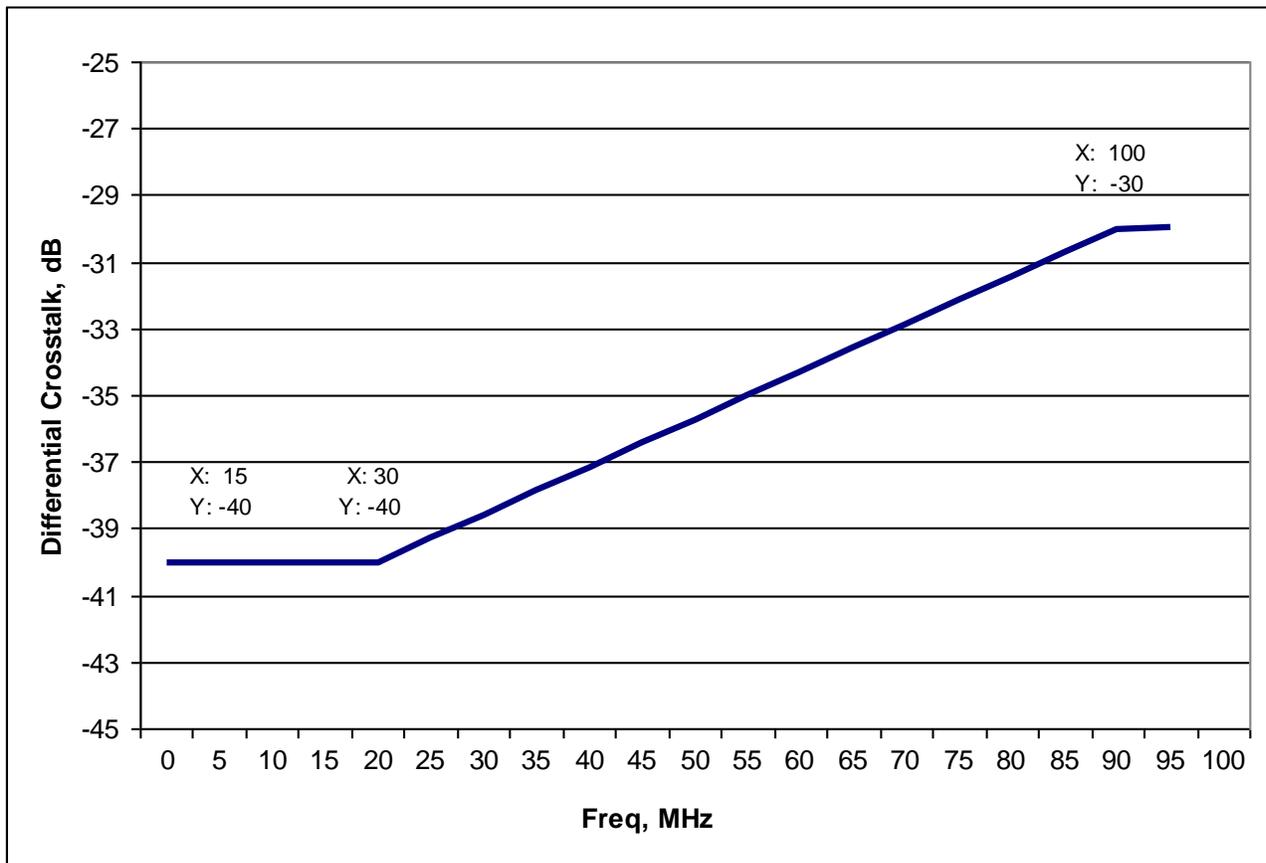
3.6.5.2 5A PD Connector Mated Pair (EIA 364-70, Method 2)

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

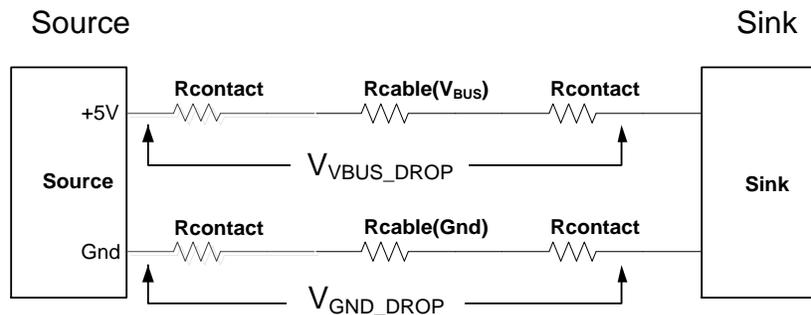
The insertion loss of the cable **shall not** exceed *insertionLoss*.

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.
- 3) The series resistance of the cable's GND wire.
- 4) 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 $E = IR \rightarrow R_{cable} (GND) =$ the maximum value for V_{GND_DROP} 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 $R = IR \rightarrow R_{cable} (V_{BUS}) =$ the maximum value for V_{VBUS_DROP} 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 **v_{IRDrop_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>VVBUS_DROP</i> and <i>VGND_DROP</i> .
<i>VGND_DROP</i> ¹			375	mV	Note: this value is the same as <i>VGND_OFFSET</i> as defined in [USB BC 1.2].
<i>VVBUS_DROP</i> ¹			625	mV	
<p>Note 1: For charging-only cables (e.g. cables without data lines), the maximum value of either <i>VVBUS_DROP</i> or <i>VGND_DROP</i> May be exceeded; however <i>VVBUS_DROP</i> plus <i>VGND_DROP</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.

4.1 Interoperability with other USB Specifications

USB Power Delivery **May** be implemented alongside the [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USB BC 1.2\]](#) and [\[USB Type-C 1.2\]](#) 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 Detached 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.

4.2 Dead Battery Detection / Unpowered Port Detection

Dead Battery/Unpowered operation is when a USB Device needs 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 USB 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.

4.2.1 Type-A to USB Type-B Dead Battery Operation

4.2.1.1 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.

4.2.1.2 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.

4.2.1.3 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.

4.2.1.4 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.

Figure 4-1 USB Type-A to USB Type-B Dead Battery / Unpowered Port Detection Flow



Table 4-1 describes the Message sequence shown in Figure 4-1.

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 $vSafe0V$.	The Consumer/Provider Shall start <i>DBDetectTimer</i> , used to determine when next to apply $vSafeDB$.
2	While $vSafeDB$ is not present, the Provider/Consumer Shall : <ul style="list-style-type: none"> • If willing to act as a Source, go and do this. (The first action will be to output $vSafe5V$). • If it does not wish to be powered, remain in this step. 	Until the <i>DBDetectTimer</i> expires, the Consumer/Provider Shall check whether V_{BUS} is above $vSafe0V$. 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 $vSafeDB$, in order to power the Provider/Consumer Bit Stream generation circuitry. The process of preparing this could 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. Apply $vSafe0V$ Procedure 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 $vSafe0V$ within <i>tDBDischargeVbus</i> max bearing in mind that V_{BUS} could be being driven with $vSafe5V$ 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 $vSafeDB$ is present, it Shall start to output Bit Stream within <i>tSendBitStream</i> of $vSafeDB$ 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"> • The time required for $vSafeDB$ to charge the capacitance presented by itself to a level at which it can begin operation. • The time required for enough of its own circuitry to power up to transmit a Bit Stream. • The fact that <i>tTurnOnImpliedSink</i> is included in this budget <p>Note that the effect of V_{BUS} capacitance on these timings is significant. The designer Should verify their own design in this respect.</p>	

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 DeviceReadyTimer 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.
Note: 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.		

4.2.2 Type-C to USB Type-C Dead Battery Operation

Dead Battery charging operation for connections between USB Type-C connectors is defined in [\[USB Type-C 1.2\]](#).

4.3 Cable IR Ground Drop (IR Drop)

Every PD Sink Port capable of USB communications can 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 [\[USB Type-C 1.2\]](#)).

4.4 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 *r_{ID}* max.

The Micro-A plug's ID pin is used for Insertion Detect. It is essentially connected to ground through at most a 1K Ω 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.

4.5 Cable Type Detection

4.5.1 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 USB 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.

Sources **Shall** detect the type of Attached cable and limit the Capabilities they offer based on the current carrying capability of the cable determined by the Cable capabilities determined using the **Discover Identity** Command (see Section 6.4.4.2) sent using SOP' Communication (see Section 2.4) to the Cable Plug. The Cable VDO returned as part of the **Discover Identity** Command details the maximum current and voltage values that **Shall** be negotiated for a given cable as part of an Explicit Contract.

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 20V or currents in excess of:
 - 1.5A for USB Type-A/Type-B
 - 3A for USB Type-C.
- Sinks with USB Type-A and USB Type-B connectors **Shall** run the cable detection process prior to sending any **Request** Messages.
- Sinks with USB 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 USB Type-C) and its known current carrying capability.
- The current carrying capability of the plug determined by:
 - The plug type (i.e. USB 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).

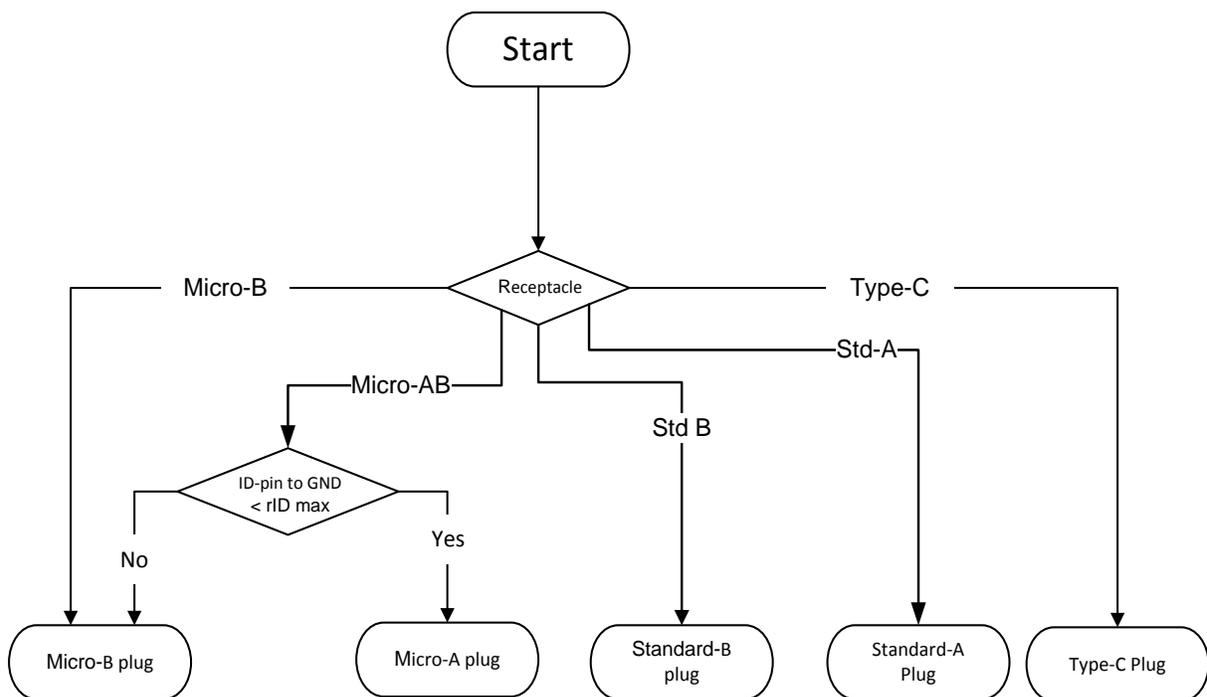
Sinks, except those with USB Type-C receptacles, **Shall** detect the type of Attached cable and limit their requests based on:

- 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).

4.5.2 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 USB Type-C, is determined. Detection of USB Type-C plugs is defined in **[USB Type-C 1.2]**.

Figure 4-2 Plug Type Determination



4.5.3 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:

- **Optionally** 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

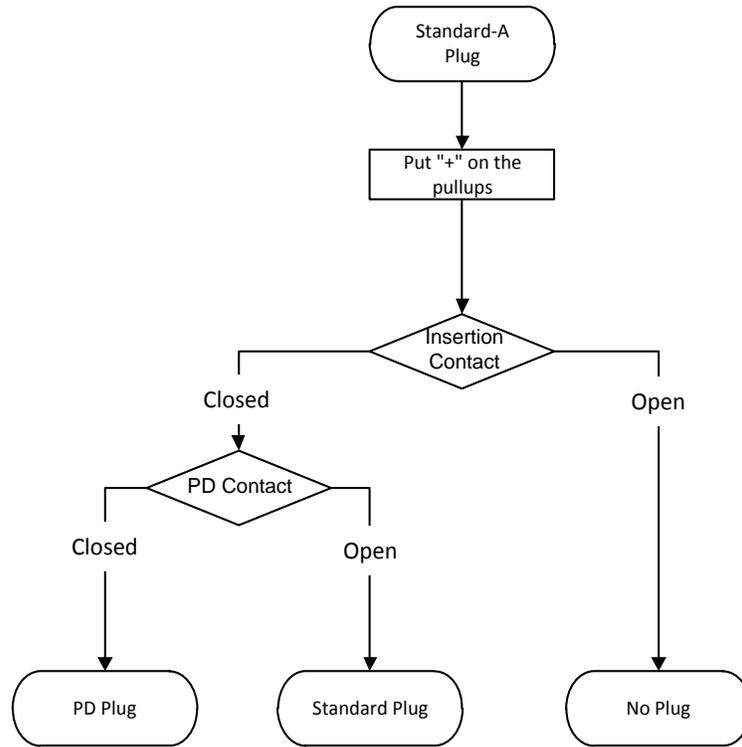
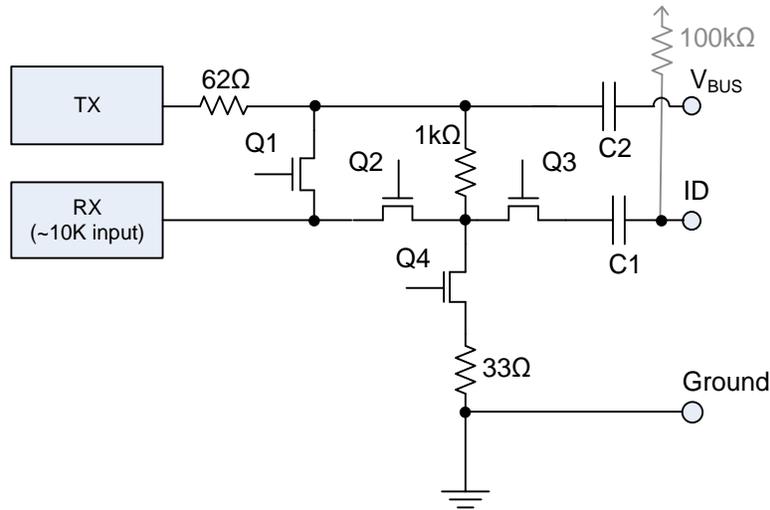


Figure 4-3 illustrates how the detection contacts in the A receptacle *shall* be used. Insertion Detect is an *Optional* feature (see Section 3.1.5). When not present, the path through the Figure 4-3 follows the Insertion Detect 'Closed' arc.

4.5.4 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 to 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 can 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 ¹	~ 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.

4.6 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).

4.7 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	4.6
<i>tBitStreamDetect</i>	100		300	ms	4.2
<i>tBitStreamOff</i>			100	ms	4.2
<i>tDBDetect</i>		10	15	s	4.2
<i>tDBDischargeVbus</i>			90	ms	4.2
<i>tDBSourceOff</i>		200	220	ms	4.2
<i>tDeviceReady</i>	60		90	s	4.2
<i>tSendBitStream</i>			95	ms	4.2
<i>tWaitForPower</i>	20			ms	4.2.1
<i>vLowPower</i>	2.5		<i>vSafe5V</i> max	V	4.6

Table 4-4 Electrical Timers

Timer	Parameter	Used By	Section
<i>BitStreamDetectTimer</i>	<i>tBitStreamDetect</i>	Policy Engine	4.2, 8.3.3.6.1.5
<i>DBDetectTimer</i>	<i>tDBDetect</i>	Policy Engine	4.2, 8.3.3.6.1.5
<i>DBSourceOffTimer</i>	<i>tDBSourceOff</i>	Policy Engine	4.2, 8.3.3.6.1.5
<i>DeviceReadyTimer</i>	<i>tDeviceReady</i>	Policy Engine	4.2, 8.3.3.6.1.5
<i>WaitForPowerTimer</i>	<i>tWaitForPower</i>	Policy Engine	4.2.1, 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 **Invalid**, 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

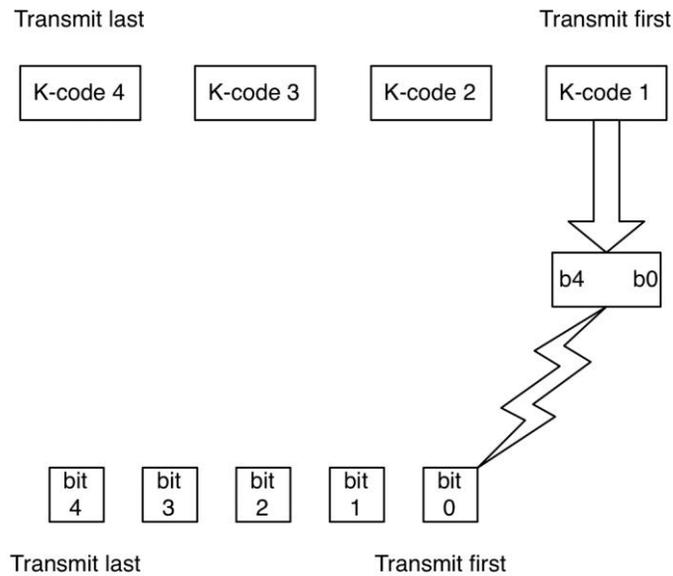
Name	4b	5b Symbol	Description
0	0000	11110	hex data 0
1	0001	01001	hex data 1
2	0010	10100	hex data 2
3	0011	10101	hex data 3
4	0100	01010	hex data 4
5	0101	01011	hex data 5
6	0110	01110	hex data 6
7	0111	01111	hex data 7
8	1000	10010	hex data 8
9	1001	10011	hex data 9
A	1010	10110	hex data A
B	1011	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
Sync-1	K-code	11000	Startsynch #1
Sync-2	K-code	10001	Startsynch #2
RST-1	K-code	00111	Hard Reset #1
RST-2	K-code	11001	Hard Reset #2
EOP	K-code	01101	EOP End Of Packet
Reserved	Error	00000	Shall Not be used
Reserved	Error	00001	Shall Not be used
Reserved	Error	00010	Shall Not be used
Reserved	Error	00011	Shall Not be used
Reserved	Error	00100	Shall Not be used
Reserved	Error	00101	Shall Not be used
Sync-3	K-code	00110	Startsynch #3
Reserved	Error	01000	Shall Not be used
Reserved	Error	01100	Shall Not be used
Reserved	Error	10000	Shall Not be used
Reserved	Error	11111	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	Reference
SOP	Section 5.6.1.2.1
SOP'	Section 5.6.1.2.2
SOP''	Section 5.6.1.2.3
Hard Reset	Section 5.6.4
Cable Reset	Section 5.6.5
SOP'_Debug	Section 5.6.1.2.4
SOP''_Debug	Section 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
Invalid (example)	K-code	Corrupt	K-code	Corrupt

5.5 Transmitted Bit Ordering

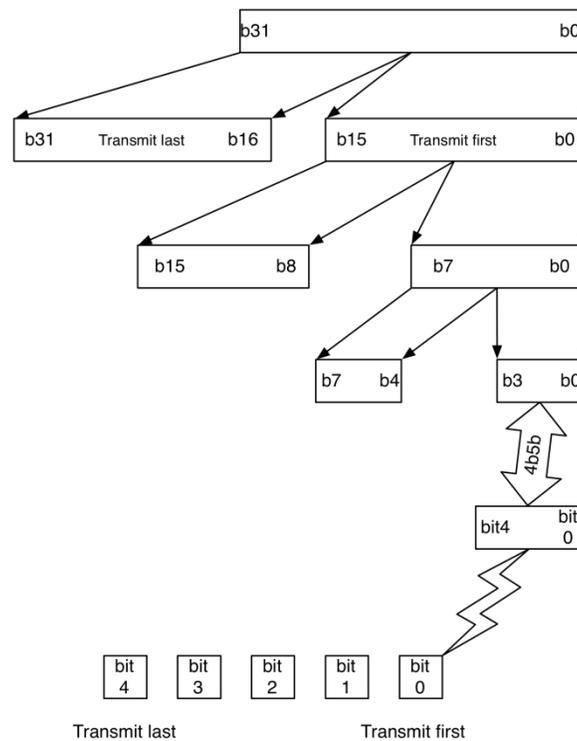
This section describes the order of bits on the wire that **Shall** be used when transmitting data of varying sizes. Table 5-4 shows the different data sizes that are possible.

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

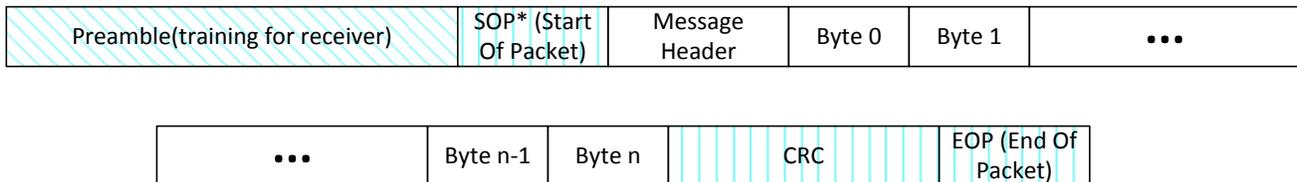
Figure 5-2 Transmit Order for Various Sizes of Data



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. See Section 6.2.1 for more details of the Packet construction for Control and Data Messages.

Figure 5-3 USB Power Delivery Packet Format



LEGEND:

Training sequence provided by the Physical layer, not 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	Sync-1
2	Sync-1
3	Sync-1
4	Sync-2

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 **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 required to respond to SOP Packets.

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	Sync-1
2	Sync-1
3	Sync-3
4	Sync-3

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 Discarded**. 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 Discarded**. When there is an Explicit Contract in place a UFP **Shall Not** send SOP' Packets and **Shall Discard** 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 Discard** 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	Sync-1
2	Sync-3
3	Sync-1
4	Sync-3

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 Discard** 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 Discarded**. For the DFP if a **Valid SOP*** is not detected (see Table 5-3) then the whole transmission **Shall be Discarded**. A UFP **Shall Not** send SOP'' Packets and **Shall Discard** 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 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

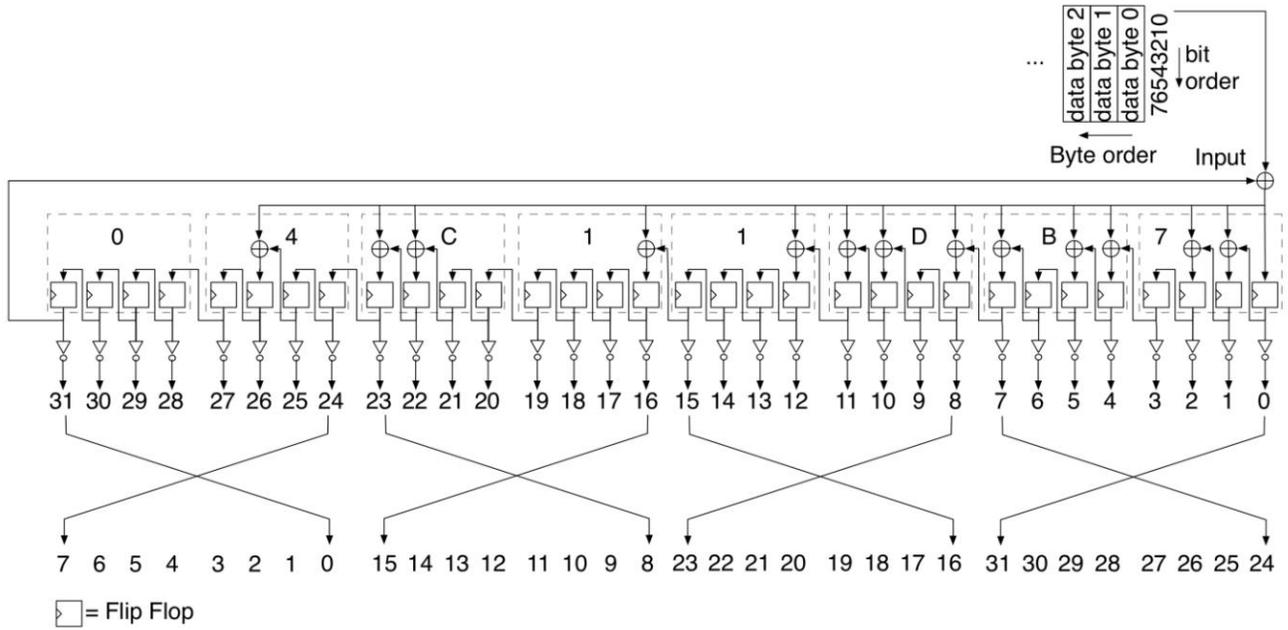


Table 5-10 CRC-32 Mapping

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
24	7

CRC-32	Result bit Position in CRC-32 Field
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	RST-1
2	RST-1
3	RST-1
4	RST-2

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 **Discarded**.

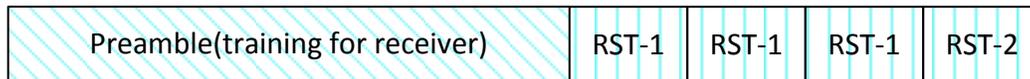
A Cable Plug **Shall** perform a Hard Reset when it detects **Hard Reset** Signaling being sent between the Port Partners. After receiving the **Hard Reset** Signaling, the device **Shall** reset as described in Section 6.7.2.

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) 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 $t_{InterFrameGap}$.
- 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:

Preamble provided by the Physical layer, **not** encoded with 4b5b

Provided by the Physical layer, encoded with 4b5b

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	RST-1
2	Sync-1
3	RST-1
4	Sync-3

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:

Preamble provided by the Physical layer, **not** encoded with 4b5b

Provided by the Physical layer, encoded with 4b5b

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

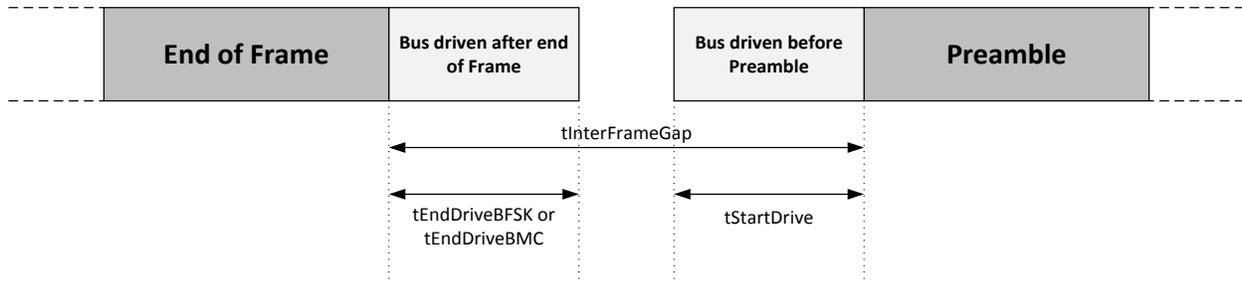
Table 5-15 Common Normative Signaling Scheme Requirements for Receiver

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

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 *tEndDriveBFSK* or *tEndDriveBMC* (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 *tInterFrameGap* after either:

- 1) Transmitting the previous frame, for example sending the next Message in an AMS immediately after having sent a *GoodCRC* Message, or
- 2) Receiving the previous frame, for example when responding to a received Message with a *GoodCRC* Message, or
- 3) Observing an idle condition on CC (see Section 5.7). In this case the Port is waiting to initiate an AMS observes idle (see Section 5.8.3.6.1) and then waits *tInterFrameGap* before transmitting the Frame. See also Section 5.7 for details on when an AMS can be initiated.

Note: the transmitter is also required to verify a bus idle condition immediately prior to starting transmission of the next Frame (see Section 5.8.3.6.1).

The transmitter of the next Frame **May** vary the start of the Preamble by *tStartDrive* (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 can 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

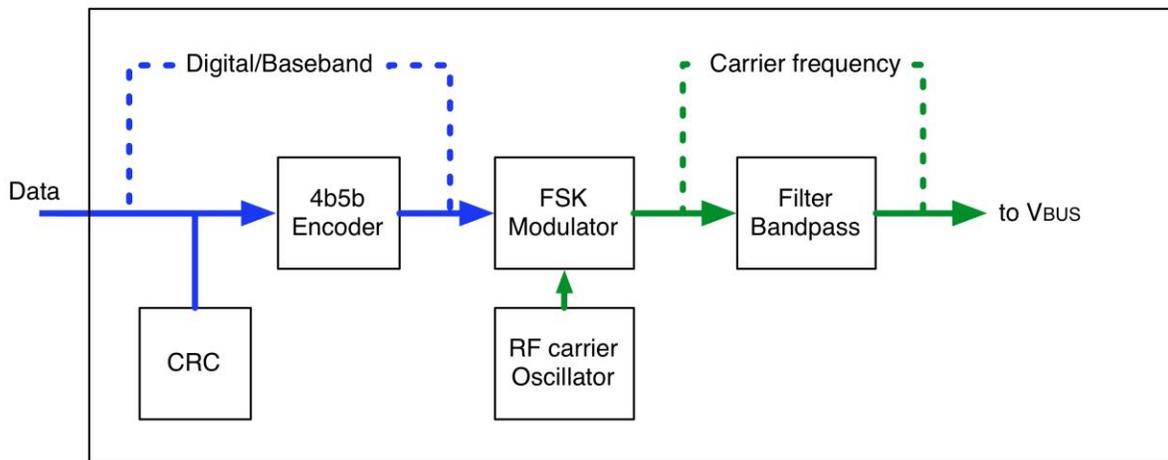


Figure 5-9 Receiver Block Diagram

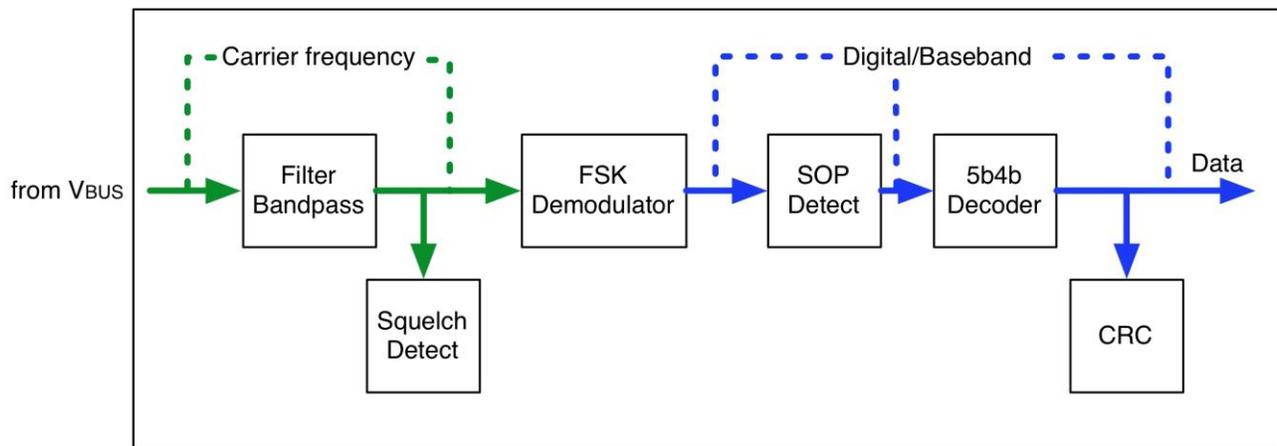
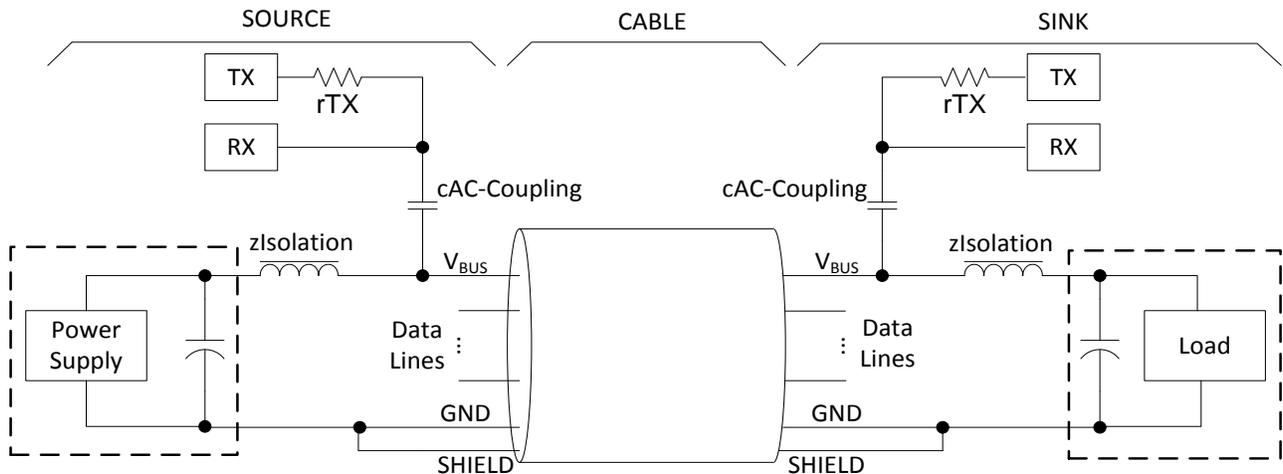


Figure 5-10 Channel Diagram (Cable Type Detection not shown)



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 could be present on V_{BUS} . For example, the PHY Layer could 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 Section 7.1.13.1 and Section 7.2.9.1 for additional detail.

Note: Isolation impedance surges can 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 could be present on V_{BUS} . For example, the Source and Sink could 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			Ω	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 μ 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.
<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 0 Ω 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	μ s	

Name	Description	Min	Nom	Max	Units	Comment
<i>vTX</i>	TX voltage injected on VBUS	100	150	200	mVRMS	This is the voltage on VBUS 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.

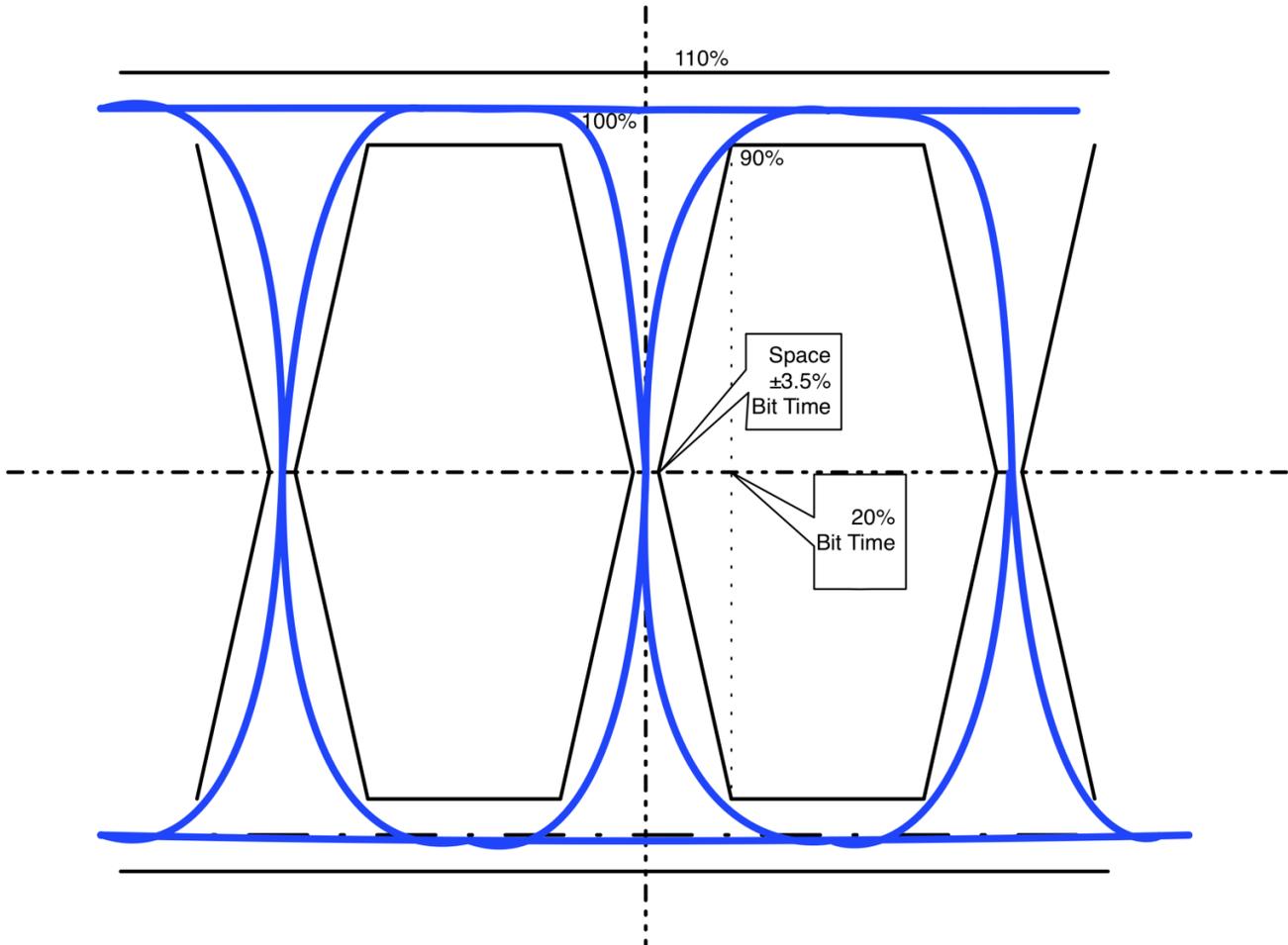
5.8.2.5.2 Transmitter Characteristics

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.

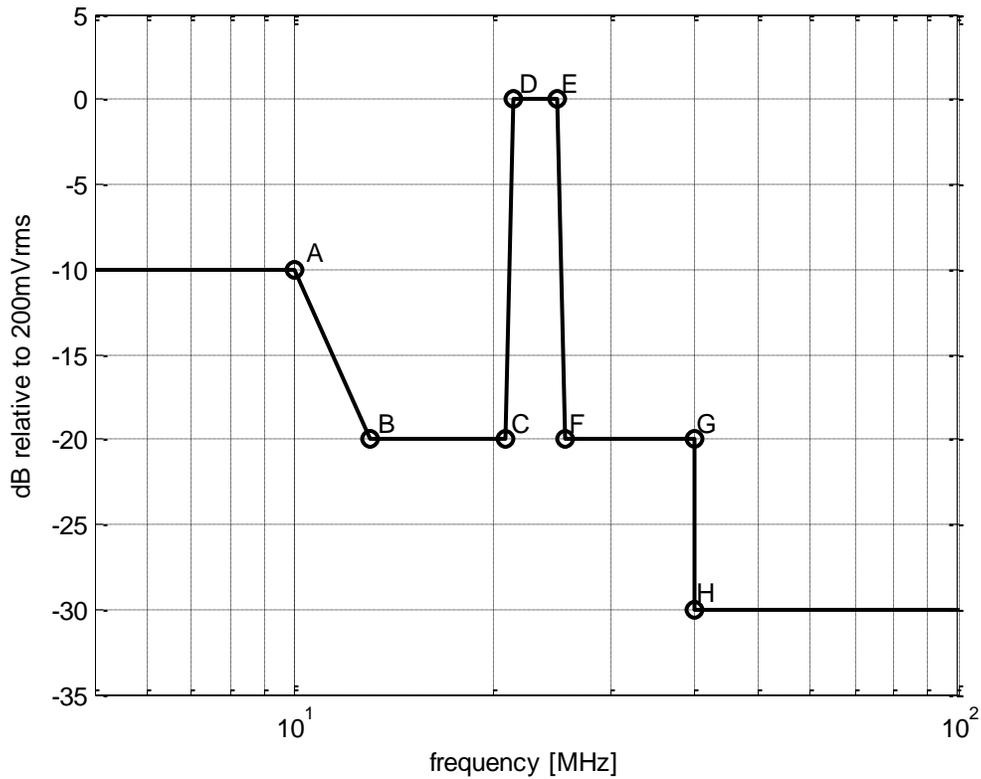
Figure 5-11 Eye diagram of BFSK Modulation



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 v_{TX}



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 4.5. 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 4.5.4).

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** could 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
tBitStreamComplete	The Bit Stream has stopped if the squelch has closed for tBitStreamComplete .	1		3	ms	
vRX	RX voltage received on V_{BUS}	55	150	300	mVRMS	This is the voltage on V_{BUS} (the link terminated by rTX). Legacy cables close to a quarter wavelength with characteristic impedance higher or lower than rTX could attenuate or amplify the signal level by up to 6dB. In practice, the minimum value will be the actual vSquelchOperating value.
vSquelchDetecting	Squelch detection sensitivity in Squelch Detection Mode	15	20	25	mVRMS	Informative only.
vSquelchOperating	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 fRange .
zRX	Receiver input impedance	10			k Ω	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 rTX (see Section 5.10). The high impedance zRX is required for cable-type detection (see Section 4.5.4).

5.8.2.6.2 Receiver Filter Specification

The design of the receiver filter represented by the “Filter Bandpass” in is implementation specific, but **Shall** take into account the out-of-band power-supply noise (see Section 7.1.13.2 and Section 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 7-7 and Figure 7-10 **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 USB Type-B connector to detect a Provider/Consumer with a Dead Battery on a USB Type-A connector (see Section 4.2).

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 can be viewed as concatenating multiple Preambles as shown in Figure 5-13 (note that the last Preamble might 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).

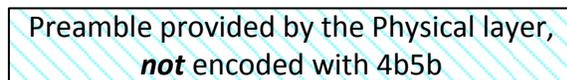
The Consumer/Provider’s receiver **shall** declare a Bit Stream is detected after detecting 128 consecutive bits that match the Preamble pattern of alternating “0s” and “1s”.

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:



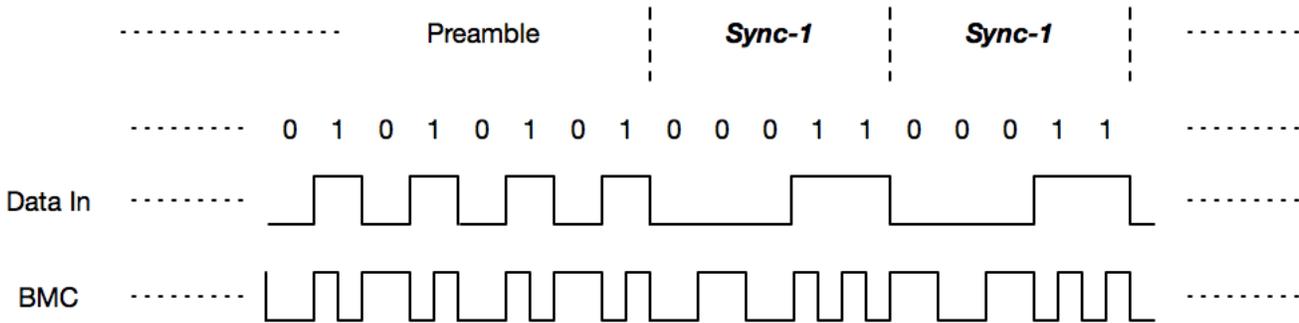
5.8.3 Biphase Mark Coding (BMC) Signaling Scheme

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 Biphase 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

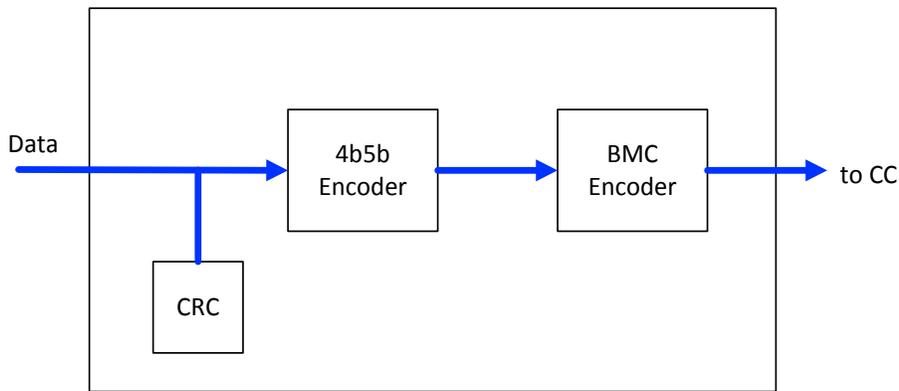
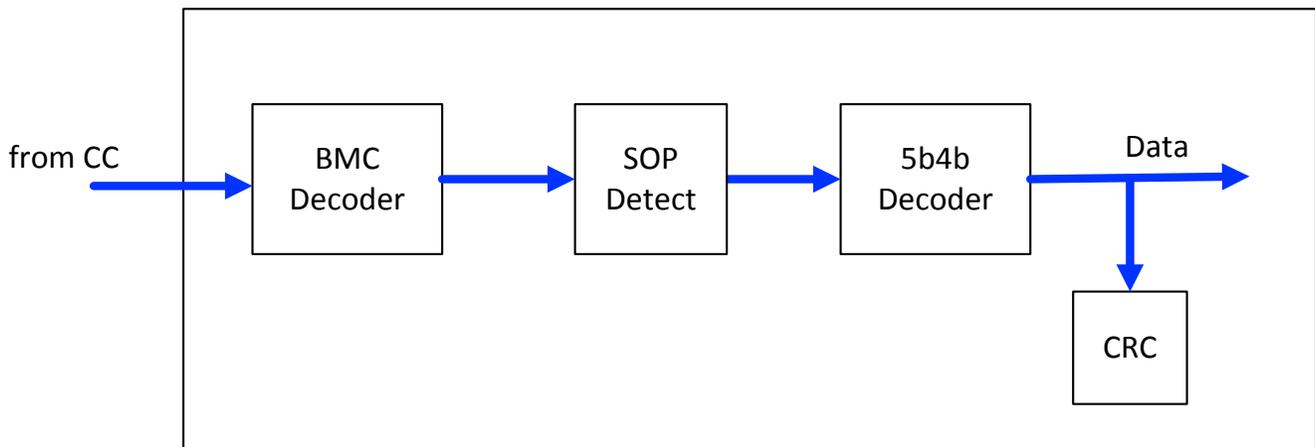


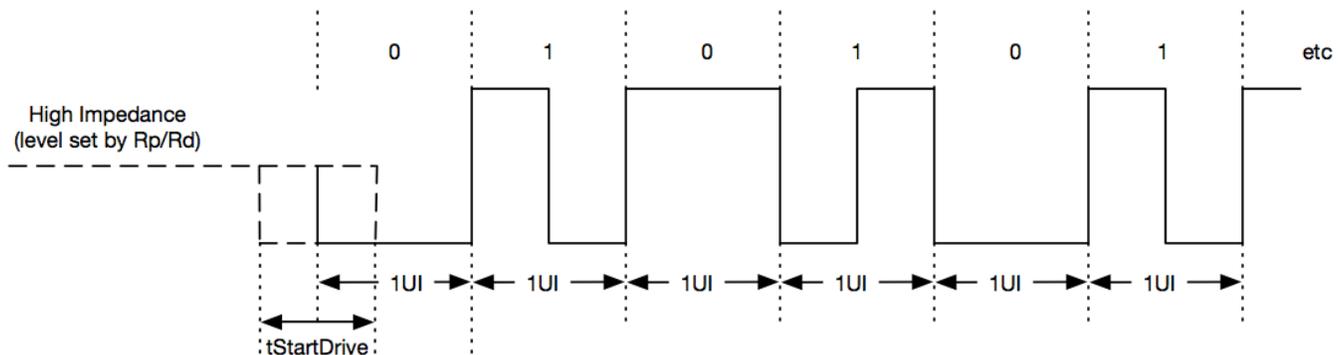
Figure 5-16 BMC Receiver Block Diagram



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 slew 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 USB Type-C fully featured cables (see [USB Type-C 1.2]). This slew 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 *tInterFrameGap* 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

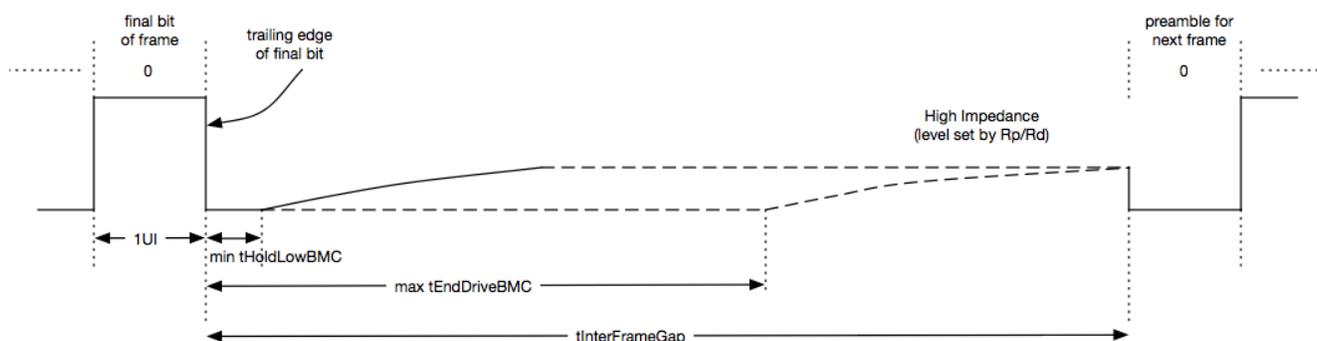
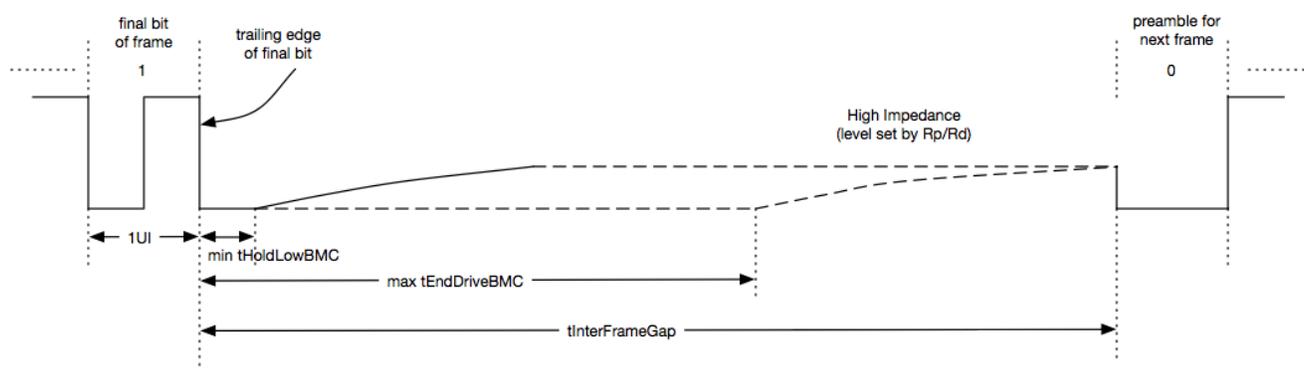


Figure 5-19 Transmitting or Receiving BMC Encoded Frame Terminated by One with High-to-Low Last Transition



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 one 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

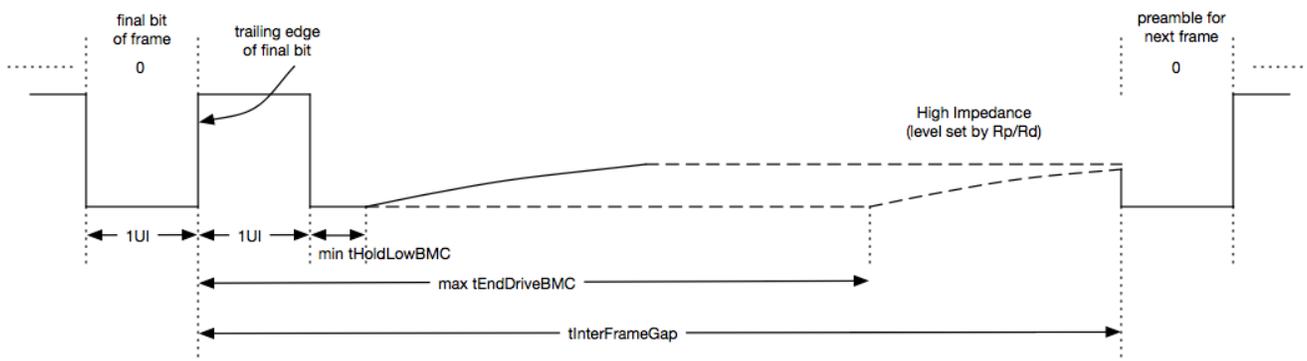
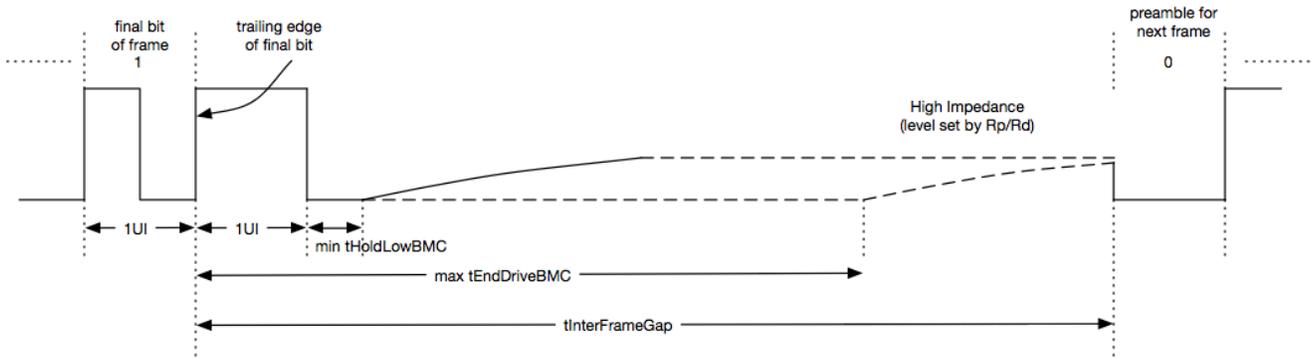


Figure 5-21 Transmitting or Receiving BMC Encoded Frame Terminated by One with Low to High Last Transition



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 5-22, Figure 5-23, 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 masks apply to the full range of R_p , R_d values as defined in [USB Type-C 1.2]. 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 transmitted signal **Shall** have a rise time no faster than t_{Rise} . The transmitted signal **Shall** have a fall time no faster than t_{Fall} . The maximum limits on the rise and fall times are enforced by the Tx inner masks.

Figure 5-22 BMC Tx 'ONE' Mask

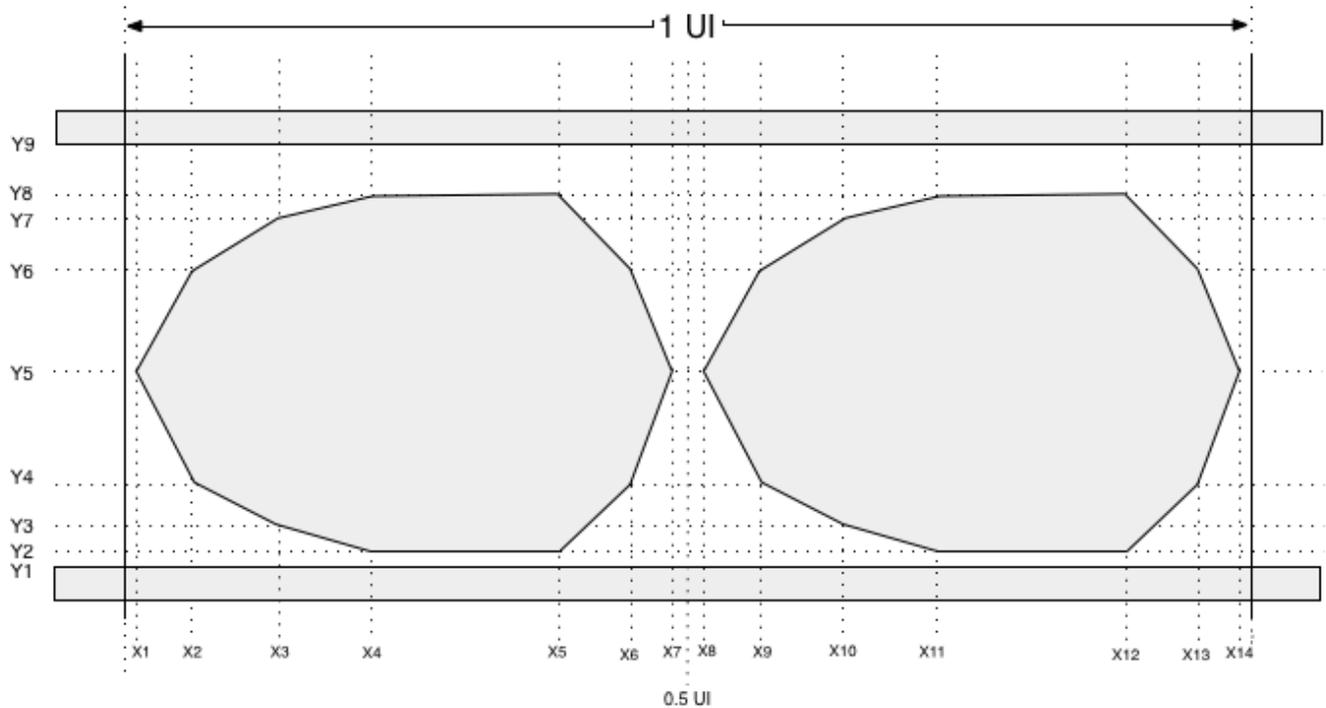


Figure 5-23 BMC Tx 'ZERO' Mask

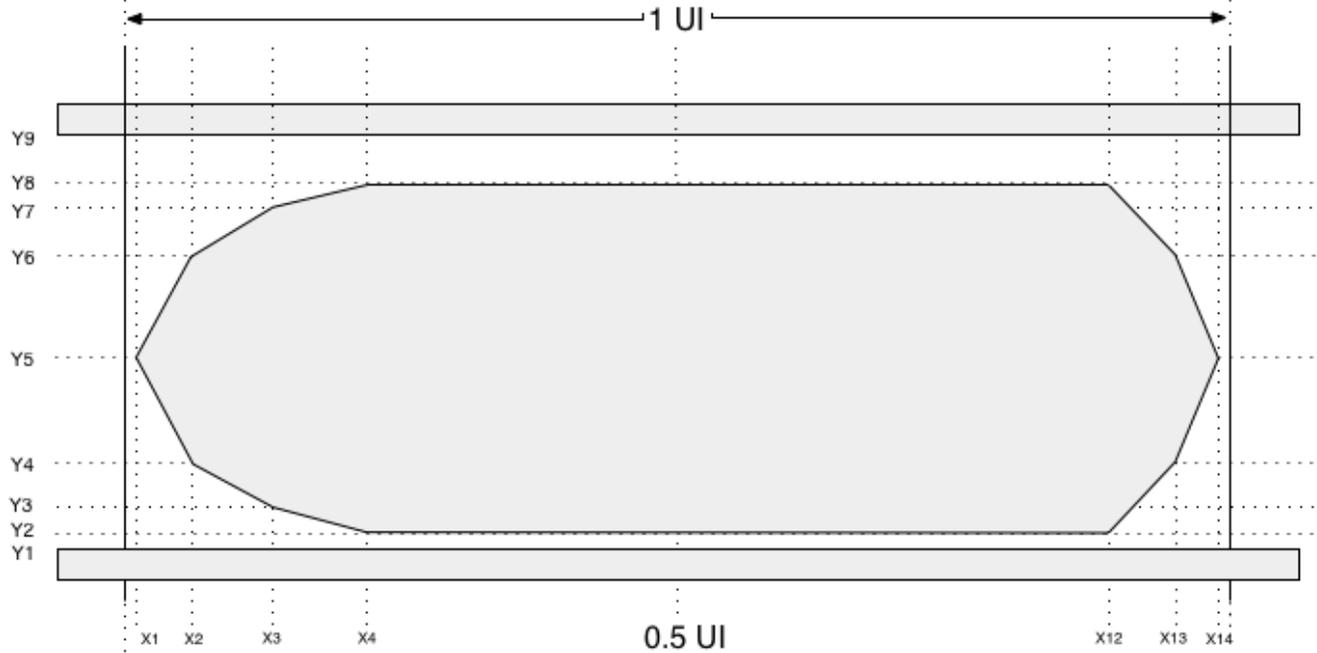


Table 5-21 BMC Tx Mask Definition, X Values

Name	Description	Value	Units
<i>X1Tx</i>	Left Edge of Mask	0.015	UI
<i>X2Tx</i>	see figure	0.07	UI
<i>X3Tx</i>	see figure	0.15	UI
<i>X4Tx</i>	see figure	0.25	UI
<i>X5Tx</i>	see figure	0.35	UI
<i>X6Tx</i>	see figure	0.43	UI
<i>X7Tx</i>	see figure	0.485	UI
<i>X8Tx</i>	see figure	0.515	UI
<i>X9Tx</i>	see figure	0.57	UI
<i>X10Tx</i>	see figure	0.65	UI
<i>X11Tx</i>	see figure	0.75	UI
<i>X12Tx</i>	see figure	0.85	UI
<i>X13Tx</i>	see figure	0.93	UI
<i>X14Tx</i>	Right Edge of Mask	0.985	UI

Table 5-22 BMC Tx Mask Definition, Y Values

Name	Description	Value	Units
<i>Y1Tx</i>	Lower bound of Outer mask	-0.075	V
<i>Y2Tx</i>	Lower bound of inner mask	0.075	V
<i>Y3Tx</i>	see figure	0.15	V

Name	Description	Value	Units
<i>Y4Tx</i>	see figure	0.325	V
<i>Y5Tx</i>	Inner mask vertical midpoint	0.5625	V
<i>Y6Tx</i>	see figure	0.8	V
<i>Y7Tx</i>	see figure	0.975	V
<i>Y8Tx</i>	see figure	1.04	V
<i>Y9Tx</i>	Upper Bound of Outer mask	1.2	V

5.8.3.2.2 Receive Masks

A Source using the BMC Signaling Scheme **Shall** be capable of receiving a signal that complies with the mask when sourcing power as defined in Figure 5-24, Figure 5-25 and Table 5-23. The Source Rx mask is bounded by sweeping a Tx mask compliant signal, with added *vNoiseActive* between power neutral and Source offsets.

A Sink using the BMC Signaling Scheme **Shall** be capable of receiving a signal that complies with the mask when sinking power as defined in Figure 5-28, Figure 5-29 and Table 5-23. The Sink Rx mask is bounded by sweeping a Tx mask compliant signal, with added *vNoiseActive* between power neutral and Sink 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 5-26, Figure 5-27 and Table 5-23.

Dual-Role Power Devices **Shall** meet the receiver requirements for a Source when providing power during any transmission using the BMC Signaling Scheme or a Sink when consuming power during any transmission using the BMC Signaling Scheme.

Cable Plugs **Shall** meet the receiver requirements for both a Source and a Sink 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 *tRxFilter* has been applied.

The boundaries of Rx outer mask, *Y1Rx* and *Y5Rx*, are specified according to *vSwing* max and accommodate half of *vNoiseActive* from cable noise coupling and the signal offset *vIRDropGNDC* 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 *vNoiseActive* to account for cable noise coupling. The received signal is composed of a waveform compliant to the Tx mask plus *vNoiseActive*.

The vertical dimension of the Rx inner mask for sourcing power is derived by reducing the vertical dimension of the Tx inner mask by *vNoiseActive* and *vIRDropGNDC* 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 *vNoiseActive* plus *vIRDropGNDC* where the *vIRDropGNDC* 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 *vNoiseActive* max and *vIRDropGNDC* 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 *vNoiseActive* plus *vIRDropGNDC* where the *vIRDropGNDC* 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 $vSwing$ for power neutral, and is shifted up by half of $vIRDropGND$ max for sourcing power and is shifted down by half of $vIRDropGND$ max for sinking power.

The receiver sensitivity *shall* be set such that the receiver does not treat noise on an undriven signal path as an incoming signal. Signal amplitudes below $vNoiseIdle$ max *shall* be treated as noise when BMC is idle.

Figure 5-24 BMC Rx 'ONE' Mask when Sourcing Power

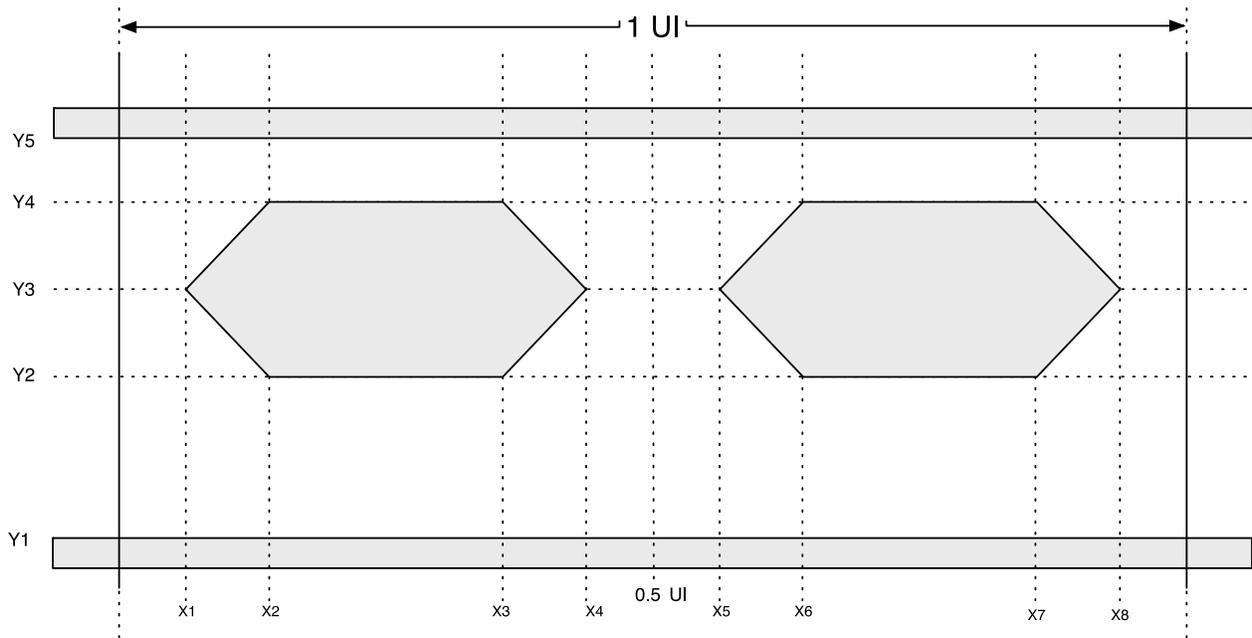


Figure 5-25 BMC Rx 'ZERO' Mask when Sourcing Power

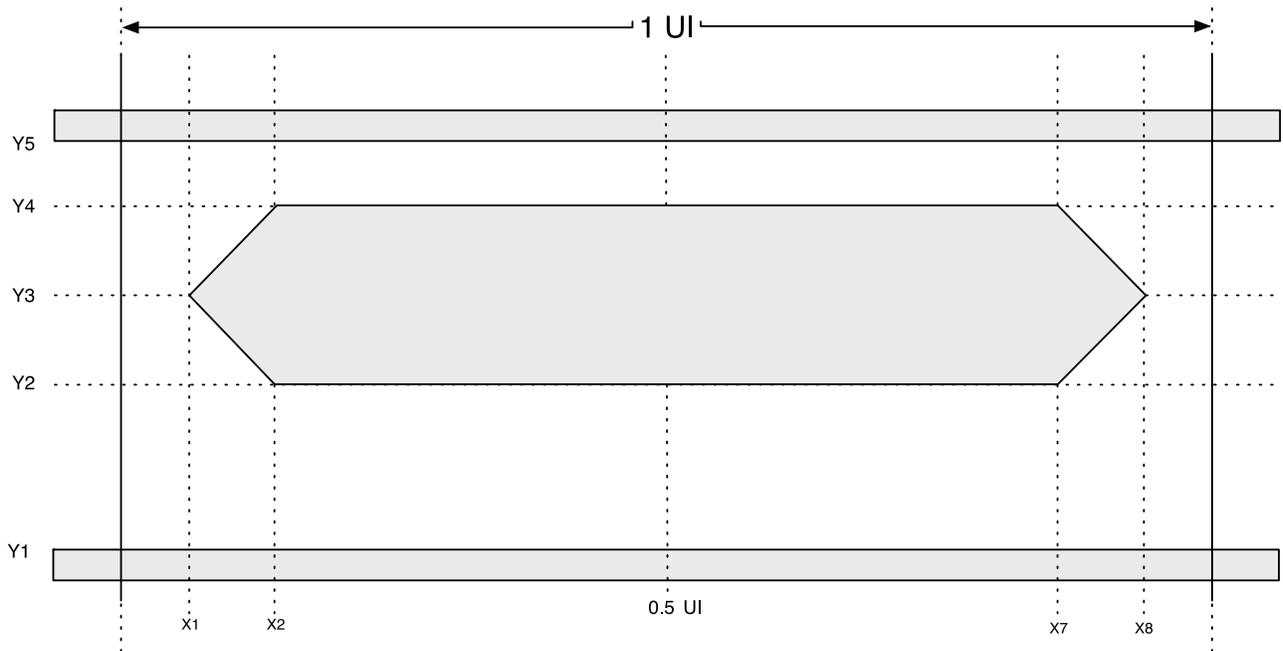


Figure 5-26 BMC Rx 'ONE' Mask when Power neutral

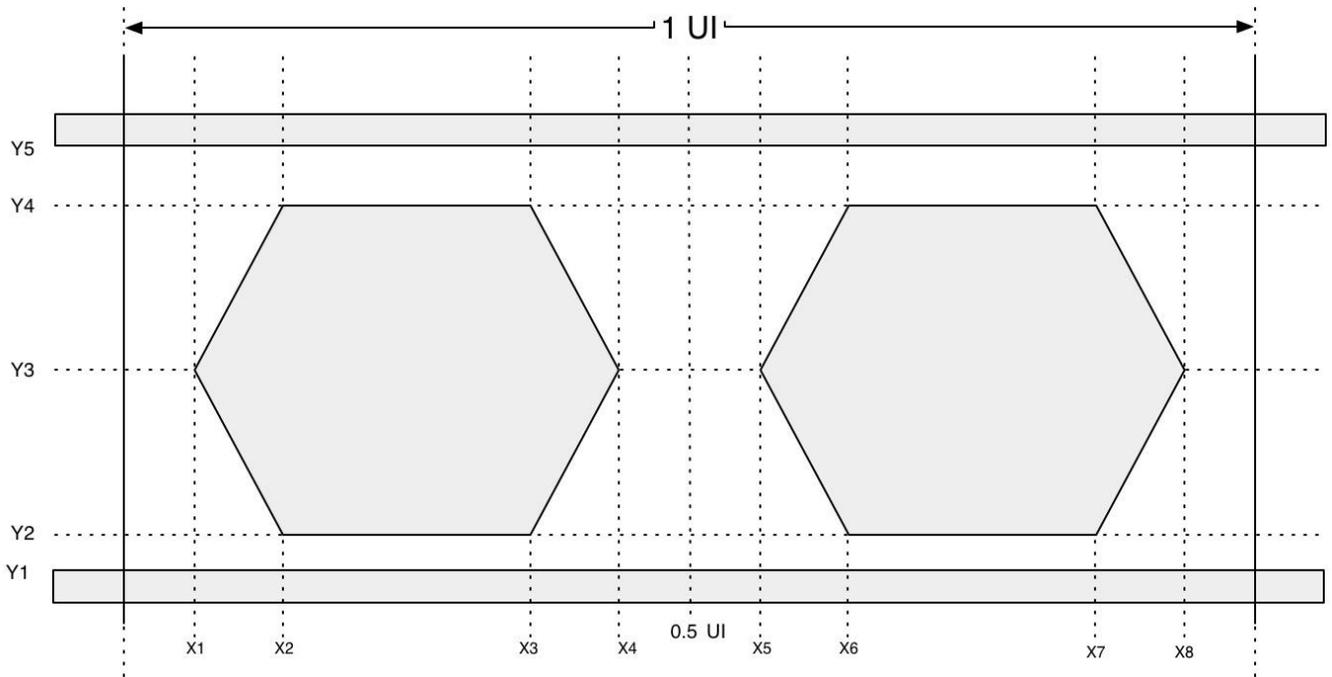


Figure 5-27 BMC Rx 'ZERO' Mask when Power neutral

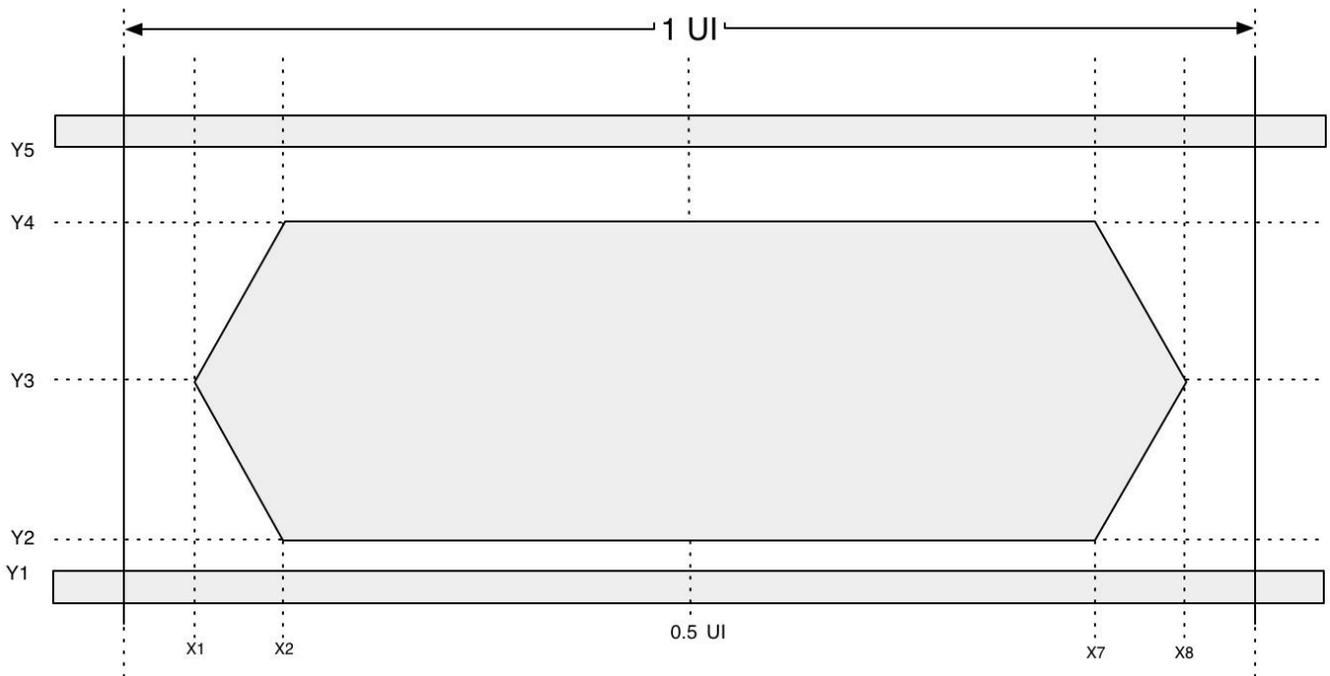


Figure 5-28 BMC Rx 'ONE' Mask when Sinking Power

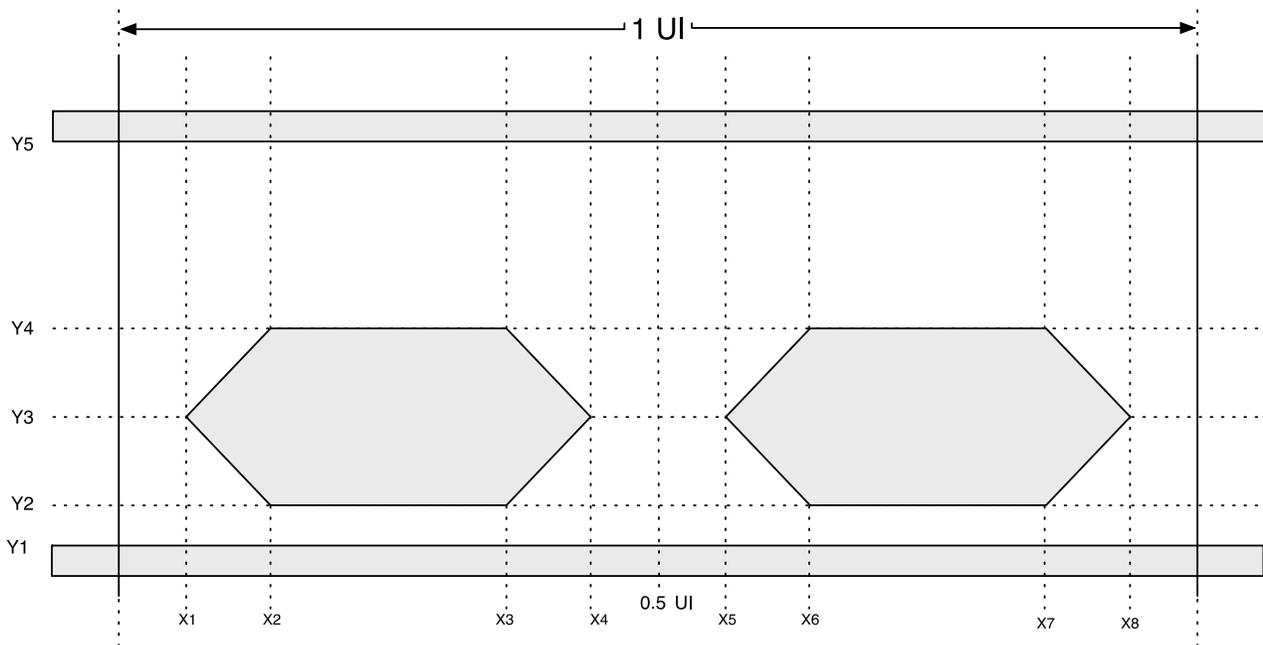


Figure 5-29 BMC Rx 'ZERO' Mask when Sinking Power

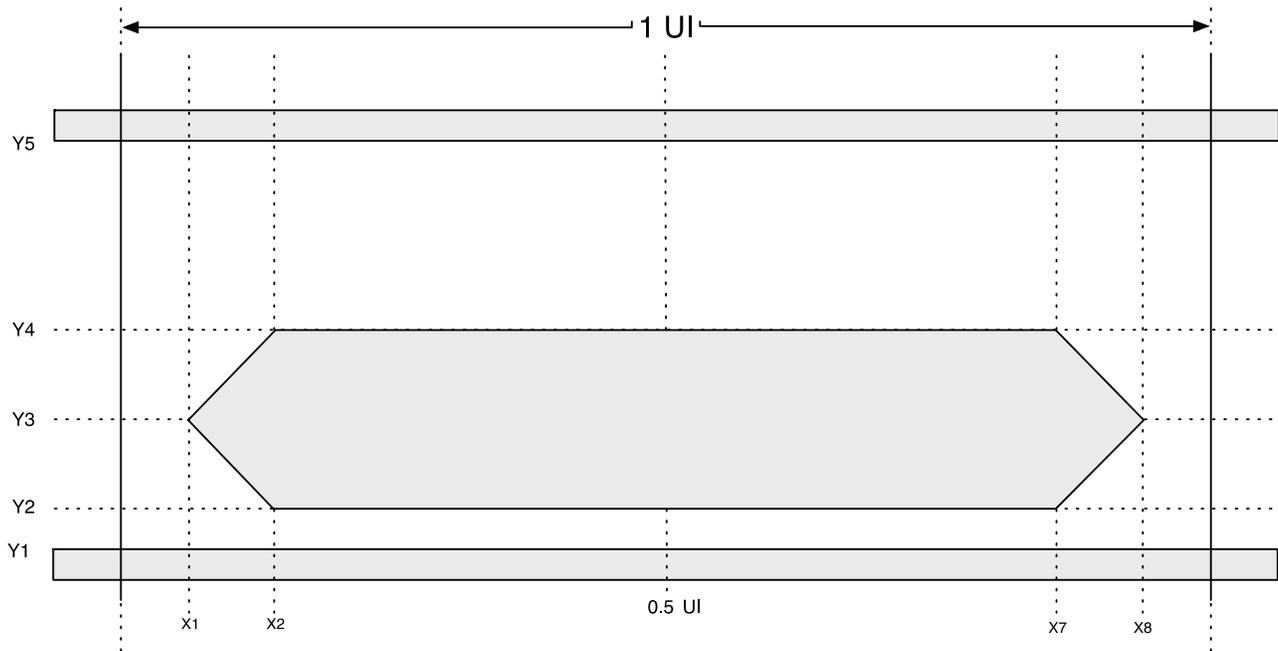


Table 5-23 BMC Rx Mask Definition

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	-0.3325	V
<i>Y2Rx</i>	Lower Bound of Inner Mask	<i>Y3Rx</i> - 0.205 when sourcing power ¹ or sinking power ¹ <i>Y3Rx</i> - 0.33 when power neutral ¹	V
<i>Y3Rx</i>	Center line of Inner Mask	0.6875 Sourcing Power ¹ 0.5625 Power Neutral ¹ 0.4375 Sinking Power ¹	V
<i>Y4Rx</i>	Upper bound of Inner mask	<i>Y3Rx</i> + 0.205 when sourcing power ¹ or sinking power ¹ <i>Y3Rx</i> + 0.33 when power neutral ¹	V
<i>Y5Rx</i>	Upper bound of the Outer mask	1.5325	V

Name	Description	Value	Units
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 **shall** be equivalent to the circuit outlined in Figure 5-30 for a Source and Figure 5-31 for a Sink. It is formed by the concatenation of a cable load model and a receiver load model. See [USB Type-C 1.2] for details of the Rp and Rd resistors. Note the parameters z_{Cable_CC} , $t_{CableDelay_CC}$ and $c_{CablePlug_CC}$ are defined in [USB Type-C 1.2].

Figure 5-30 Transmitter Load Model for BMC Tx from a Source

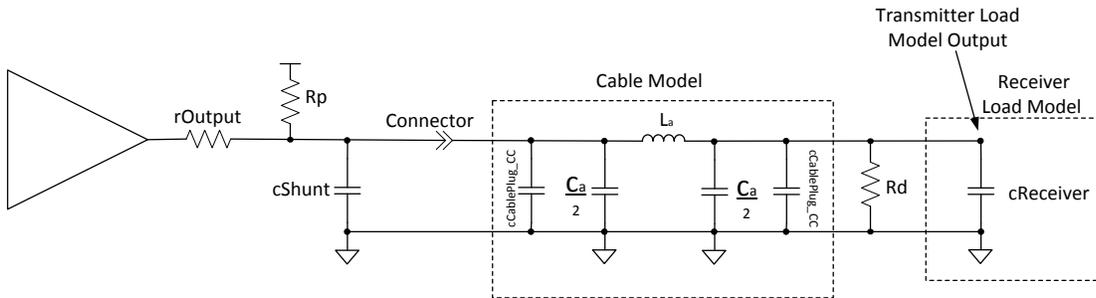
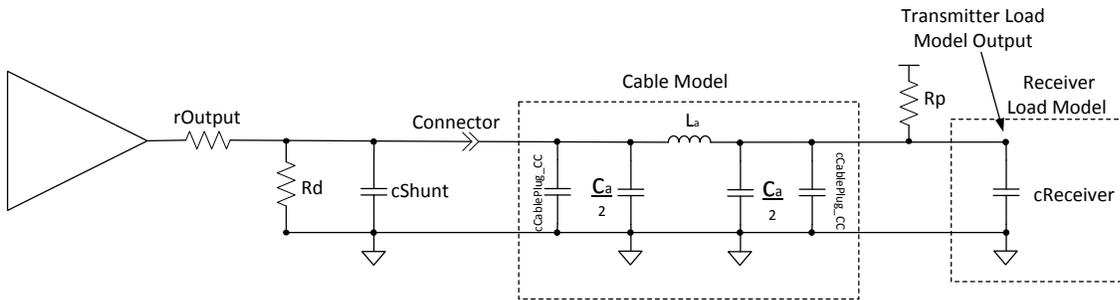


Figure 5-31 Transmitter Load Model for BMC Tx from a Sink



The transmitter system components r_{Output} and c_{Shunt} 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, L_a , (in nH) **shall** be calculated from the following formula:

$$L_a = t_{CableDelay_CC_{max}} * z_{Cable_CC_{min}}$$

$t_{CableDelay_CC}$ is the modeled signal propagation delay through the cable, and z_{Cable_CC} is the modeled cable impedance.

The modeled cable inductance is 640 nH for a cable with $z_{Cable_CC_{min}} = 32 \Omega$ and $t_{CableDelay_CC_{max}} = 20$ nS.

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

$$C_a = \frac{t_{CableDelay_CC_{max}}}{z_{Cable_CC_{min}}}$$

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

cCablePlug_CC 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> ¹	Unit Interval	3.03		3.70	μs	$1/f_{BitRate}$
Note 1: <i>tUnitInterval</i> 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.						

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.
<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 [USB 2.0] low speed (750 kHz) while the source is driving the CC line.

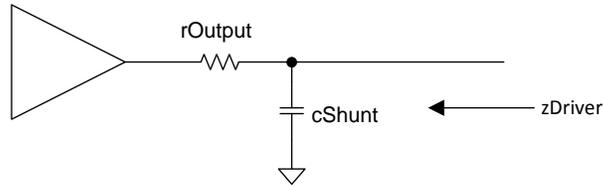
cReceiver is the capacitance that a DFP or UFP **Shall** present on the CC line when the DFP or UFP's receiver is not transmitting on the line. The transmitter **May** have more capacitance than *cReceiver* while driving the CC line, but **Shall** meet the waveform mask requirements. Once transmission 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 5-32 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 ¹ .
<i>tTransitionWindow</i>	Time window for detecting non-idle	12		20	μs	
<i>zBmcRx</i>	Receiver Input Impedance	1			MΩ	
<i>vNoiseActive</i>	Noise amplitude when BMC is active.			165	mV	Peak-to-peak noise from <i>V_{BUS}</i> , 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 <i>V_{BUS}</i> , 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 [USB Type-C 1.2]

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.

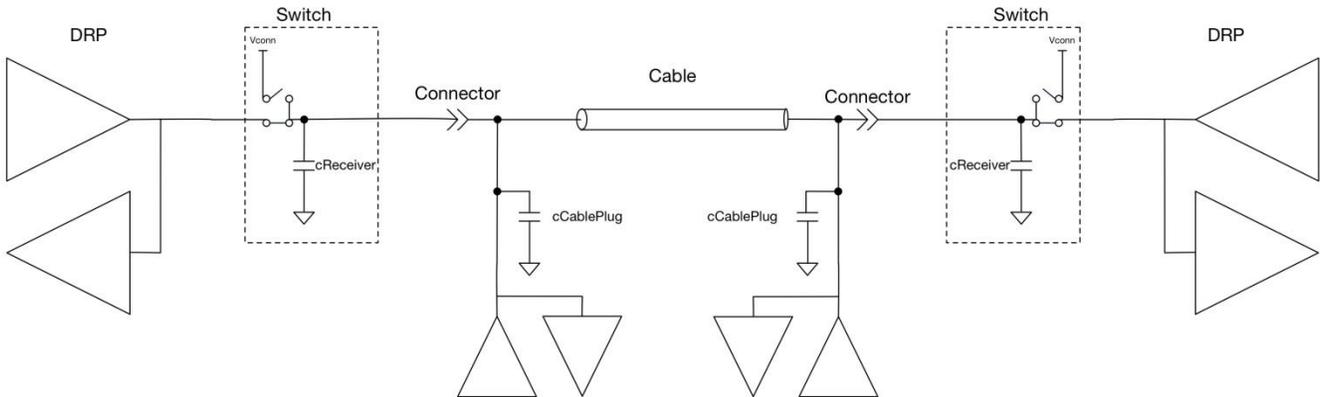
Refer to Section 5.8.1.4 for details of when transmissions **May** start.

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 5-33 below illustrates a typical Multi Drop configuration with two DRPs.

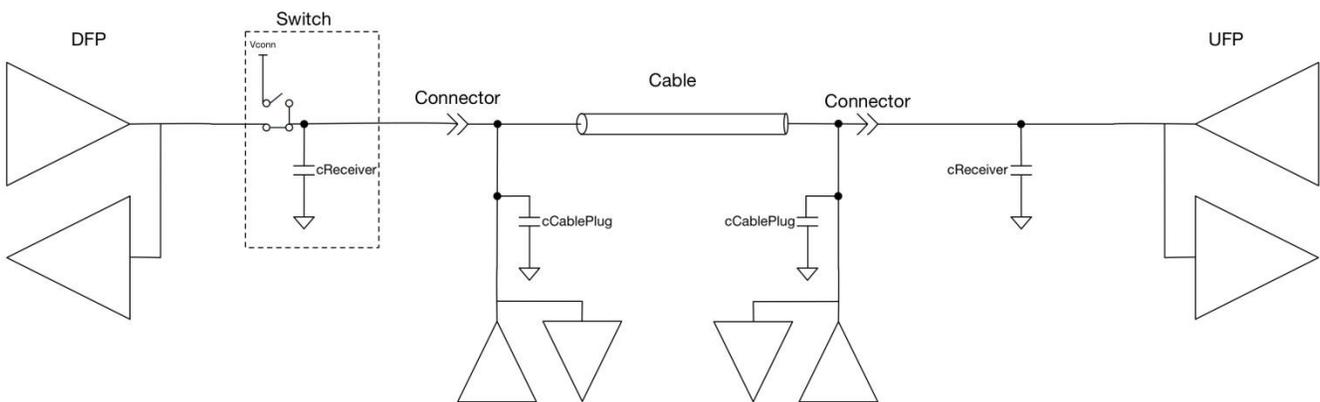
Figure 5-33 Example Multi-Drop Configuration showing two DRPs



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_CC*. 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.

It is possible to have a configuration at Attach where one Port is able to be a Vconn Source and the other Port is not able to be a Vconn Source, such that there is no switch in the second Port. An example of a DFP with a switch Attached to a UFP without a switch is outlined in Figure 5-34. The capacitance on the CC line for a Port not able to be a VCONN Source **Shall** still be within *cReceiver* except when transmitting.

Figure 5-34 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 Schemes, manufacturers **May** choose to support both BMC over the CC wire and BFSK over V_{BUS} on a USB Type-C connector. Products with USB Type-C connectors **Shall Not** support BFSK without supporting BMC. Note that any system utilizing the USB Type-C connector can see BFSK signaling on V_{BUS} when a suitable USB Type-A/B to USB Type-C adapter is used.

When both BMC and BFSK Signaling Schemes are supported by a **[USB Type-C 1.2]** 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)

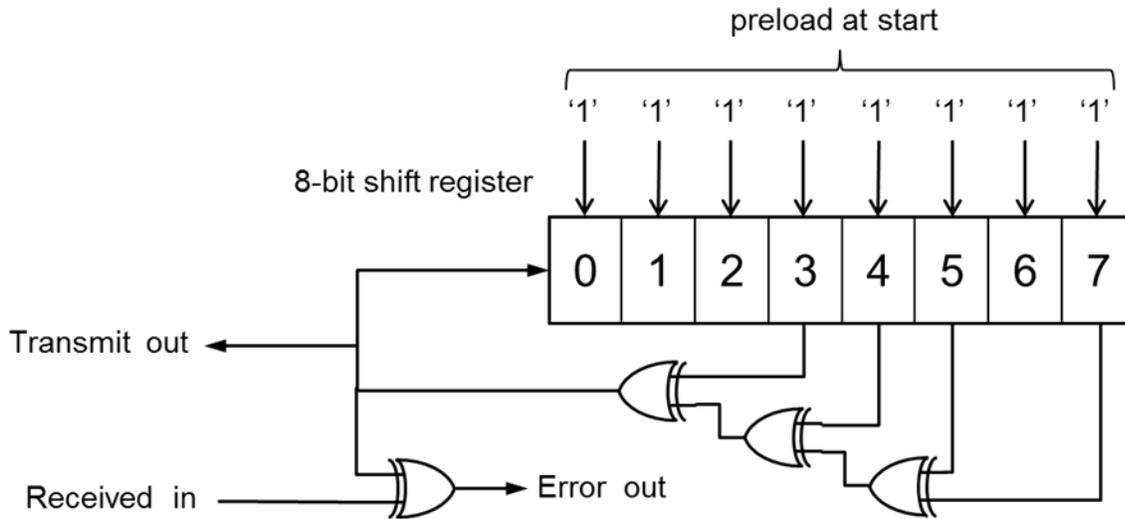
5.9.1 BIST PRBS Pattern

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

Figure 5-35 shows an example implementation of the PRBS-generator and checker.

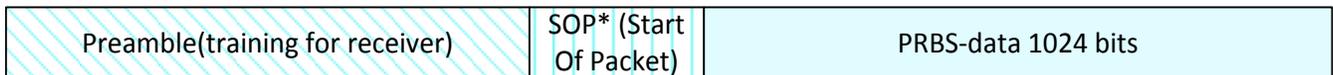
The preloaded pattern **Shall** be "all ones" i.e. all 8 bits in the shift register **Shall** be set to "1". The pattern **Shall** be preloaded when the request to enter test mode is given or received.

Figure 5-35 Example implementation of the BIST generator and checker

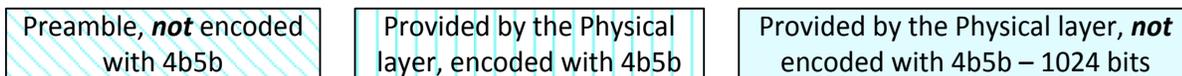


In BIST Transmit or Receiver Test Frames are constructed as shown in Figure 5-36 with a test pattern as defined in Section 5.9.1. Note that the Test Frame does not include an **EOP**. At least **nBISTConfidence** of these Test Frames **Shall** be sent/received without error (see Section 5.9.1.1).

Figure 5-36 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 4.61×10^6 (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

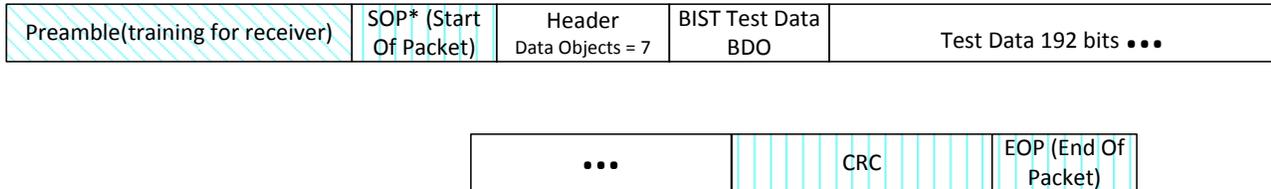
In *BIST Eye Pattern*, the Physical Layer **Shall** send out a continuous string of bits in accordance with Section 5.9.1. This produces a signal that will allow measurement of the eye pattern and of the spectrum mask.

5.9.7 BIST Test Data

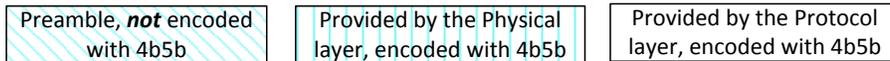
A **BIST Test Data** Message is used by the Tester to send various Tester generated test patterns to the UUT in order to test the UUT's receiver. See also Section 6.4.3.

Figure 5-37 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 a **BIST Test Data** BIST Data Object, followed by 6 Data Objects containing Test data, followed by the CRC and then an **EOP**.

Figure 5-37 Test Data Frame



LEGEND:



5.9.8 BIST Parameters

Table 5-28 BIST Parameters

Parameter	Description	Min	Nom	Max	Units	Comment
nBISTConfidence	Number of Test Frames to transmit in order to reach 99% confidence level	4502			Frames	
nBISTPayload	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
BIST Receiver Mode	√		
BIST Transmit Mode	√		
BIST Eye Pattern	√		
BIST Carrier Mode 0	√		
BIST Carrier Mode 1	√		
BIST Carrier Mode 2	√	√	
BIST Carrier Mode 3	√		
BIST Test Data	√	√	

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 illustrates a Control Message as part of a Packet showing the parts are provided by the Protocol and PHY Layers.

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



Legend:



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

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

Preamble	SOP* (Start Of Packet)	Message Header (16 bit)	0..7 Data Object(s)	CRC	EOP (End Of Packet)
----------	------------------------	-------------------------	---------------------	-----	---------------------

Legend:



6.2.1.1 Message Header

Every Message **Shall** start with a 16-bit Message Header, as shown in Figure 6-1 The Message Header contains basic information about the Message and the PD Port Capabilities. The Message Header **May** be used standalone as a **Control Message** when the **Number of Data Objects** field is zero or as the first part of a **Data Message** when the **Number of Data Objects** field is non-zero.

Table 6-1 Message Header

Bit(s)	Start of Packet	Field Name	Notes
15	N/A	Reserved	Shall be set to 0
14...12	SOP*	Number of Data Objects	See Section 6.2.1.2
11...9	SOP*	MessageID	See Section 6.2.1.3
8	SOP only	Port Power Role	See Section 6.2.1.4
	SOP'/SOP''	Cable Plug	See Section 6.2.1.7
7...6	SOP*	Specification Revision	See Section 6.2.1.5
5	SOP only	Port Data Role	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*	Message Type	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 the *Request* Message that are only ever sent by a Sink **Shall** always have the *Port Power Role* field set to Sink.

During the Power Role Swap Sequence, for the initial Source Port, the *Port Power Role* field **Shall** be set to Sink in the *PS_RDY* Message indicating that the initial Source's power supply is turned off (see 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 **Shall Not** lead to Soft Reset, Hard Reset or Error Recovery if it is incorrect.

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

6.2.1.5 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.

If a USB 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, USB Type-C error recovery actions as defined in [USB Type-C 1.2] **Shall** be performed.

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

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

6.2.1.7 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 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 6-2 Control Message Types

Bits 3...0	Message Type	Sent by	Description	Valid Start of Packet
0000	<i>Reserved</i>	<i>N/A</i>	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>	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	<i>Reserved</i>	<i>N/A</i>	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 while in a Continuous BIST Mode (see Section 6.4.3).

6.3.2 GotoMin Message

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

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

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

The Source sends this Message 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 6.5.2).

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.
- It **Shall** be sent by the recipient of a **DR_Swap** Message to indicate it is unable to do a Data Role Swap.
- It **Shall** be sent by the recipient of a **VCONN_Swap** Message to indicate it is unable to do a VCONN Swap.
- 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.
- It **Shall** be sent by a Sink which receives a **GotoMin** Message but does not support a GotoMin (GiveBack flag = 0 in its most recently sent **Request** Message)
- It **Shall** be sent by the recipient of an Unsupported Message or an Unrecognized Message (see Section 6.7.1)

The **Reject** Message **Shall** be sent within **tReceiverResponse** of the receipt of the last bit of Message (see Section 6.5.2).

6.3.5 Ping Message

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

The **Ping** Message is used on USB Type-A and USB Type-B connectors to determine the continued presence of the Sink when no other messaging is taking place (see Figure 8-43 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]**, **[USB Type-C 1.2]** or **[USB 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 USB Type-C Connectors

Type-C connectors have an alternative mechanism to determine Sink presence so when the Port Partners are both connected using USB Type-C connectors the **Ping** Message is not necessary. A Sink using a USB 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 8.3.2.2.1 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 USB Type-C connectors while maintaining the direction of power flow over V_{BUS} . The DR_Swap process can be used by **[USB Type-C 1.2]** 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 **[USB Type-C 1.2]** Ports. A UFP that supports USB Communication Capability starts as the USB Device on attachment of **[USB Type-C 1.2]** 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.

[USB Type-C 1.2] DRPs **Shall** have the capability to perform a Data Role Swap from the **PE_SRC_Ready** or **PE_SNK_Ready** states. **[USB Type-C 1.2]** DFPs and **[USB Type-C 1.2]** 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 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 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, USB Type-C error recovery actions, such as disconnect, as defined in **[USB Type-C 1.2]** will be necessary.

See Section 8.3.2.8.2, Section 8.3.3.6.2.1 and Section 8.3.3.6.2.2 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 USB 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 as defined in **[USB Type-C 1.2]**. When performing a Power Role Swap from Source to Sink, a USB Type-C Port **Shall** change its CC Wire resistor from R_p to R_d . When performing a Power Role Swap from Sink to Source, a USB Type-C Port **Shall** change its CC Wire resistor from R_d to R_p . The DFP (Host), UFP (Device) roles and V_{CONN} Source **Shall** remain unchanged during the Power Role Swap process.

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

For more information regarding the Power Role Swap, refer to Section 7.3.9 and Section 7.3.10 in the Power Supply chapter, Section 8.3.2.7.1, Section 8.3.2.8.1, Section 8.3.3.6.3.1 and Section 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 USB Type-C Port that utilizes the SSTX and SSRX pins and supports the **PR_Swap** Message.

The **VCONN_Swap** Message **May** be sent by either Port Partner to request an exchange of VCONN Source. The recipient of the Message **Shall** respond by sending an **Accept** Message, **Reject** Message or **Wait** Message.

- If an **Accept** Message is sent, the Port Partners **Shall** perform a VCONN Swap. The new VCONN Source **Shall** send a **PS_RDY** Message within **tVCONNSourceOn** to indicate that it is now sourcing VCONN. The initial VCONN Source **Shall** cease sourcing VCONN within **tVCONNSourceOff** of receipt of the last bit of the **EOP** of the **PS_RDY** Message.
- If a **Reject** Message is sent, the requester is informed that the recipient is unable, or unwilling, to do a VCONN Swap and no action **Shall** be taken.
- 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.

Before communicating with a Cable Plug a Port **Should** ensure that it is the Vconn Source and that the Cable Plugs are powered by performing a Vconn swap if necessary. Since it cannot be guaranteed that the present Vconn Source is supplying Vconn, the only means to ensure that the Cable Plugs are powered is for a Port wishing to communicate with a Cable Plug to become the Vconn Source.

6.3.12 Wait Message

The **Wait** Message is a **Valid** response to a **Request**, a **PR_Swap**, **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 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 6.5.2).

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** and **Shall** ensure that there is **tSinkRequest** after receiving the **Wait** Message before sending another **Request** Message.

6.3.12.2 Wait in response to a PR_Swap Message

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

this evaluation one of the Port Partners **Should** initiate the Power Role Swap process again by sending a **PR_Swap** Message.

The **Wait** Message is also used where a Hub is operating in hybrid mode when a request cannot be satisfied (see [USBCBridge 1.0]).

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

6.3.12.3 Wait in response to a DR_Swap Message

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

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

6.3.12.4 Wait in response to a VCONN_Swap Message

The **Wait** Message is used when responding to a **VCONN_Swap** Message to indicate that a VCONN Swap might be possible in the future. This can occur in any case where the device receiving the **VCONN_Swap** Message needs to evaluate the request further. The sender of the **VCONN_Swap** Message **Should** initiate the VCONN Swap process again at a future time by resending a **VCONN_Swap** Message.

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

6.3.13 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.

After a VCONN Swap the VCONN Source needs to reset the Cable Plug's Protocol Layer 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 several types of Data Objects:

- 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 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 Start of Packet
0000	<i>Reserved</i>		All values not explicitly defined are Reserved and Shall Not be used.	
0001	<i>Source_Capabilities</i>	Source or Dual-Role	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	<i>Reserved</i>		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 can 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 **Shall** 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 [\[USB Type-C 1.2\]](#)). The Source **Shall** limit its offered capabilities to the maximum current supported by its receptacle and Attached plug. A Sink with a USB Type-A or a USB Type-B receptacle **Shall** limit its requested capabilities to the maximum current supported by its receptacle and Attached plug. A Sink with a USB Type-C receptacle **May** make a request from any of the capabilities offered by the Source. For further details see Section 4.5.

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 (Vmin = Vmax)
	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 Power devices

Dual-Role Power devices send a *Source_Capabilities* Message (see Section 6.4.1) as part of advertising Port capabilities when operating in Source role. Dual-Role Power devices send a *Source_Capabilities* Message (see Section 6.4.1) in response to a *Get_Source_Cap* Message regardless of their present operating role. Similarly Dual-Role Power devices send a *Sink_Capabilities* Message (see Section 6.4.1.3) 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 Sinks with Accessory Support do not source V_{BUS} (see *[USB Type-C 1.2J]*) however when sourcing V_{CONN} 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 Non-Standard attachment of USB Type-A Ports

When a Provider/Consumer Port, with a USB 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 USB 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 USB Type-A Ports when Attached in a non-standard configuration.

Table 6-5 USB Type-A to USB 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).
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 Section 7.1.3 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 Power 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	Unconstrained Power
B26	USB Communications Capable
B25	Dual-Role Data
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 regardless of the device's present power role. If the Dual-Role Power bit is set to one in the *Source_Capabilities* Message the Dual-Role Power bit in the *Sink_Capabilities* Message **Shall** also be set to one. If the Dual-Role Power bit is set to zero in the *Source_Capabilities* Message the Dual-Role Power bit in the *Sink_Capabilities* Message **Shall** also be set to zero.

6.4.1.2.3.2 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], [USB Type-C 1.2] or [USB 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 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 Unconstrained Power

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

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

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

6.4.1.2.3.4 USB Communications Capable

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

6.4.1.2.3.5 Dual-Role Data

The Dual-Role Data bit **Shall** be set when the Port is USB Type-C (see [USB Type-C 1.2]) and supports the *DR_Swap* Message. This is a static capability which **Shall** remain fixed for a given device regardless of the device's present power role or data role. If the Dual-Role Data bit is set to one in the *Source_Capabilities* Message the Dual-Role Data bit in the *Sink_Capabilities* Message **Shall** also be set to one. If the Dual-Role Data bit is set to zero in the *Source_Capabilities* Message the Dual-Role Data bit in the *Sink_Capabilities* Message **Shall** also be set to zero.

6.4.1.2.3.6 Peak Current

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

For example when a computer system tries to maintain average power consumption, the higher the peak current, the longer the low current (see Section 7.2.8) 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 _{OC} (default)
01	Overload Capabilities: 1. Peak current equals 150% I _{OC} for 1ms @ 5% duty cycle (low current equals 97% I _{OC} for 19ms) 2. Peak current equals 125% I _{OC} for 2ms @ 10% duty cycle (low current equals 97% I _{OC} for 18ms) 3. Peak current equals 110% I _{OC} for 10ms @ 50% duty cycle (low current equals 90% I _{OC} for 10ms)
10	Overload Capabilities: 1. Peak current equals 200% I _{OC} for 1ms @ 5% duty cycle (low current equals 95% I _{OC} for 19ms) 2. Peak current equals 150% I _{OC} for 2ms @ 10% duty cycle (low current equals 94% I _{OC} for 18ms) 3. Peak current equals 125% I _{OC} for 10ms @ 50% duty cycle (low current equals 75% I _{OC} for 10ms)
11	Overload Capabilities: 1. Peak current equals 200% I _{OC} for 1ms @ 5% duty cycle (low current equals 95% I _{OC} for 19ms) 2. Peak current equals 175% I _{OC} for 2ms @ 10% duty cycle (low current equals 92% I _{OC} for 18ms) 3. Peak current equals 150% I _{OC} for 10ms @ 50% duty cycle (low current equals 50% I _{OC} for 10ms)

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 7.1.3 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 Section 7.1.3 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 Power ports presently operating as a Source.

Each Power Data Object **Shall** describe a specific Sink operational power level, such as a Battery (e.g. 2.8-4.1V) or a fixed power supply (e.g. 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 Section 7.1.3 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	Unconstrained Power
B26	USB Communications Capable
B25	Dual-Role Data
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 Power bit **Shall** be set when the Port is Dual-Role Power capable i.e. supports the **PR_Swap** Message. This is a static capability which **Shall** remain fixed for a given device regardless of the device's present power role. If the Dual-Role Power bit is set to one in the **Source_Capabilities** Message the Dual-Role Power bit in the **Sink_Capabilities** Message **Shall** also be set to one. If the Dual-Role Power bit is set to zero in the **Source_Capabilities** Message the Dual-Role Power bit in the **Sink_Capabilities** Message **Shall** also be set to zero.

6.4.1.3.1.2 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 Unconstrained Power

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

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

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

6.4.1.3.1.4 USB Communications Capable

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

6.4.1.3.1.5 Dual-Role Data

The Dual-Role Data bit **Shall** be set when the Port is a USB Type-C DRP (see [\[USB Type-C 1.2\]](#)) and supports the *DR_Swap* Message. This is a static capability which **Shall** remain fixed for a given device regardless of the device's present power role or data role. If the Dual-Role Data bit is set to one in the *Source_Capabilities* Message the Dual-Role Data bit in the *Sink_Capabilities* Message **Shall** also be set to one. If the Dual-Role Data bit is set to zero in the *Source_Capabilities* Message the Dual-Role Data bit in the *Sink_Capabilities* Message **Shall** also be set to zero.

6.4.1.3.2 Variable Supply (non-Battery) Power Data Object

Table 6-11 describes a Variable Supply (non-Battery) (10b) PDO used by a Sink. See Section 7.1.3 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 Section 7.1.3 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 8.3.2.2.1). 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 and Shall Not be used)
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 and Shall Not be used)
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 and Shall Not be used)
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 Power in 250mW units
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 and Shall Not be used)
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 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 can 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 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 Section 6.3.12.1 and Section 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 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 Section 6.3.12.1 and Section 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 data 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.

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 might 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 8.3.2.10 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 6.4.3.1
	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 Returned BIST Counters , this field Shall contain the contents of BISTErrorCounter otherwise it Shall be set to zero.	See Section 6.4.3.3

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 *tBISTMode* 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 *tBISTResponse* after the first bit of the Preamble of the previous Test Frame has been transmitted 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 *fCarrier - fDeviation*.

The UUT **Shall** exit the Continuous BIST Mode within *tBISTContMode* of this Continuous BIST Mode being enabled (see Section 6.5.8.4).

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 *fCarrier + fDeviation*.

The UUT **Shall** exit the Continuous BIST Mode within *tBISTContMode* of this Continuous BIST Mode being enabled (see Section 6.5.8.4).

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** exit the Continuous BIST Mode within *tBISTContMode* of this Continuous BIST Mode being enabled (see Section 6.5.8.4).

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** exit the Continuous BIST Mode within *tBISTContMode* of this Continuous BIST Mode being enabled (see Section 6.5.8.4).

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** exit the Continuous BIST Mode within *tBISTContMode* of this Continuous BIST Mode being enabled (see Section 6.5.8.4).

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 test **Shall** be ended by sending **Hard Reset** Signaling to 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 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 power related AMS.
- 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)	Parameter	Description
B31...16	Vendor ID (VID)	Unique 16-bit unsigned integer. Assigned by the USB-IF to the Vendor.
B15	VDM Type	0 = Unstructured VDM
B14...0	Available for Vendor Use	Content of this field is defined by the vendor.

6.4.4.1.1 USB Vendor ID

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

6.4.4.1.2 VDM Type

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

6.4.4.2 Structured VDM

Setting the VDM Type field to 1 (Structured VDM) defines the use of bits B14...0 in the Structured VDM Header. The fields in the Structured VDM Header are defined in Table 6-19.

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.
- When a Cable Plug does not support Structured VDMs any Structured VDMs received **Shall** be **Ignored**.
- A DFP, UFP or Cable Plug which supports Structured VDMs and receiving a Structured VDM for a SVID that it does not recognize **Shall** reply with a NAK Command.
- A Structured VDM Command sequence **Shall** be Interruptible e.g. due to the need for a power related AMS.

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
B14...13	Structured VDM Version	Version Number of the Structured VDM (not this specification Version): <ul style="list-style-type: none"> • Version 1.0 = 0 • Values 1-3 are Reserved and Shall Not be used
B12...11	Reserved	For Commands 0...15 Shall be set to 0 and Shall be Ignored SVID Specific Commands (16...31) defined by the SVID.

Bit(s)	Field	Description
B10...8	Object Position	For the <i>Enter Mode</i> , <i>Exit Mode</i> and <i>Attention</i> Commands (Requests/Responses): <ul style="list-style-type: none"> 000b = Reserved and Shall Not be used. 001b...110b = Index into the list of VDOs to identify the desired Mode VDO 111b = Exit all Active Modes (equivalent of a power on reset). Shall only be used with the <i>Exit Mode</i> Command. Commands 0...3, 7...15: <ul style="list-style-type: none"> 000b 001b...111b = Reserved and Shall Not be used. SVID Specific Commands (16...31) defined by the SVID.
B7...6	Command Type	00b = REQ (Request from Initiator Port) 01b = ACK (Acknowledge Response from Responder Port) 10b = NAK (Negative Acknowledge Response from Responder Port) 11b = BUSY (Busy Response from Responder Port)
B5	Reserved	Shall be set to 0 and Shall be Ignored
B4...0	Command ¹	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 the *SOP** 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> .	Shall only use <i>SOP/SOP'</i> .
<i>Discover SVIDs</i>	Shall only use the <i>PD SID</i> .	Shall only use <i>SOP/SOP'</i> .
<i>Discover Modes</i>	Valid with any SVID.	Shall only use <i>SOP/SOP'</i> .
<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> .
SVID Specific Commands	Valid with any SVID.	Valid with <i>SOP*</i> (defined by SVID).

6.4.4.2.1 SVID

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

Table 6-21 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.

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 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 Responder is responsible for deciphering the contents to know whether or not it supports the Mode at the Object Position.

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

6.4.4.2.5 Command Type

6.4.4.2.5.1 Commands other than Attention

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

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

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

- “Responder ACK” is the normal return and **Shall** be sent to indicate that the Command request was received and handled normally.
- “Responder NAK” **Shall** be returned when the Command request:
 - has an **Invalid** parameter (e.g. **Invalid** SVID or Mode)
 - cannot not be acted upon because the configuration is not correct (e.g. a Mode which has a dependency on another Mode or a request to exit a Mode which is not Active).
 - Is not recognized

The handling of “Responder NAK” is left up to the Initiator.

“Responder BUSY” **Shall** be sent in the response to a VDM when the Responder is unable to respond to the Command request immediately, but the Command request **May** be retried. The Initiator **Shall** wait **tVDMBusy** after a “Responder BUSY” response is received before retrying the Command request.

6.4.4.2.5.2 Attention Command

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

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

No response **Shall** be made to an **Attention** Command.

6.4.4.2.6 Command

6.4.4.2.6.1 Commands other than Attention

This field contains the value for the VDM Command being sent. The Commands explicitly listed in this field are used to identify devices and manage their operational Modes. 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 consists of a Command request and a Command response (ACK, NAK or BUSY). A Structured VDM Command is deemed to be completed (and if applicable, the transition to the requested functionality is made) when the **GoodCRC** Message has been successfully received by the Responder in reply to its Command response.

If Structured VDMs are supported but the Structured VDM Command request is not recognized, it **Shall** be NAKed (see Table 6-22).

6.4.4.2.6.2 Attention Command

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

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

If Structured VDMs are supported, but the Structured VDM **Attention** Command request is not recognized it **Shall** be **Ignored** (see Table 6-22).

6.4.4.3 Use of Commands

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

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

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

The Command sequences use the terms Initiator and Responder to identify messaging roles the ports are taking on relative to each other. This role is independent of the Port's power capability (Provider, Consumer etc.) or its present power role (Source or Sink). The Initiator is the Port sending the initial Command request and the Responder is the Port replying with the Command response. See Section 6.4.4.3.6.

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
Discover Identity	ACK, NAK, BUSY	Section 6.4.4.3.1

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

Examples of Command usage can be found in Appendix G.

6.4.4.3.1 Discover Identity

The **Discover Identity** Command is provided to enable an Initiator to identify its Port Partner and for an Initiator (V_{CONN} Source) to identify the Attached Responder (Cable Plug). The Discovery Identity Command is also used 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 Section 8.3.3.9.2 and Section 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 tV_{CONN}Stable after V_{CONN} has been applied (see **[USB Type-C 1.2]**). During Cable Plug discovery, when there is an Explicit Contract, **Discover Identity** Commands are sent at a rate defined by the **DiscoverIdentityTimer** (see Section 6.5.15) 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 request **shall** be set to the **PD SID** (see Table 6-21).

The Number of Data Objects field in the Message Header in the **Discover Identity** Command request **shall** be set to 1 since the **Discover Identity** Command request **shall not** contain and VDOs.

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

- Cable VDO (see Section 6.4.4.3.1.4).
- Alternate Mode Adapter VDO (see Section 6.4.4.3.1.5)

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

Figure 6-6 Discover Identity Command response

Header No. of Data Objects = 4-7 ¹	VDM Header	ID Header VDO	Cert Stat VDO	Product VDO	0..3 ² Product Type VDO(s)
--	------------	---------------	---------------	-------------	---------------------------------------

¹ Only Data objects defined in this specification can be sent as part of the **Discover Identity** Command.

² The following sections define the number and content of the VDOs for each Product Type.

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

6.4.4.3.1.1 ID Header VDO

The ID Header VDO contains information corresponding to the Power Delivery Product. The fields in the ID Header VDO **Shall** be as defined in Table 6-23.

Table 6-23 ID Header VDO

Bit(s)	Description	Reference
B31	USB Communications Capable as USB Host: <ul style="list-style-type: none"> Shall be set to one if the product is capable of enumerating USB Devices. Shall be set to zero otherwise 	Section 6.4.4.3.1.1.1
B30	USB Communications Capable as a USB Device: <ul style="list-style-type: none"> Shall be set to one if the product is capable of enumerating as a USB Device. Shall be set to zero otherwise 	Section 6.4.4.3.1.1.2
B29...27	Product Type (UFP): <ul style="list-style-type: none"> 000b – Undefined 001b – PDUSB Hub 010b – PDUSB Peripheral 011b...100b – Reserved, Shall Not be used. 101b – Alternate Mode Adapter (AMA) 110b...111b – Reserved, Shall Not be used. Product Type (Cable Plug): <ul style="list-style-type: none"> 000b – Undefined 001b...010b – Reserved, Shall Not be used. 011b – Passive Cable 100b – Active Cable 101b...111b – Reserved, Shall Not be used. 	Section 6.4.4.3.1.1.3
B26	Modal Operation Supported: <ul style="list-style-type: none"> Shall be set to one if the product supports Modal Operation. Shall be set to zero otherwise 	Section 6.4.4.3.1.1.5
B25...16	Reserved. Shall be set to zero.	
B15...0	16-bit unsigned integer. USB Vendor ID	[USB 2.0] / [USB 3.1]

6.4.4.3.1.1.1 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.

6.4.4.3.1.1.2 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.

6.4.4.3.1.1.3 Product Type (UFP)

The Product Type (UFP) field indicates the type of Product when in UFP Data Role, whether a VDO will be returned and if so the type of VDO to be returned. Table 6-24 defines the Product Type VDOs which **Shall** be returned.

Table 6-24 Product Types (UFP)

Product Type	Description	Product Type VDO	Reference
Undefined	Shall be used where no other Product Type value is appropriate.	None	
PDUSB Hub	Shall be used when the Product is a PDUSB Hub.	None	

Product Type	Description	Product Type VDO	Reference
PDUSB Peripheral	Shall be used when the Product is a PDUSB Device other than a PDUSB Hub.	None	
Alternate Mode Adapter	Shall be used when the Product is a PDUSB Device that supports one or more Alternate Modes.	AMA VDO	Section 6.4.4.3.1.5

6.4.4.3.1.1.4 Product Type (Cable Plug)

The Product Type (Cable Plug) field indicates the type of Product when the Product is a Cable Plug, whether a VDO will be returned and if so the type of VDO to be returned. Table 6-25 defines the Product Type VDOs which **Shall** be returned.

Table 6-25 Product Types (Cable Plug)

Product Type	Description	Product Type VDO	Reference
Undefined	Shall be used where no other Product Type value is appropriate.	None	
Active Cable	Shall be used when the Product is a cable that incorporates signal conditioning circuits.	Active Cable VDO	Section 6.4.4.3.1.4.2
Passive Cable	Shall be used when the Product is a cable that does not incorporate signal conditioning circuits.	Passive Cable VDO	Section 6.4.4.3.1.4.1

6.4.4.3.1.1.5 Modal Operation Supported

The Modal Operation Supported bit is used to indicate whether or the not the Product supports Modes.

6.4.4.3.1.1.6 Vendor ID

Manufacturers **Shall** set the Vendor ID field to the value of the Vendor ID assigned to them by USB-IF. For USB Devices or Hubs which support USB communications the Vendor ID field **Shall** be identical to the Vendor ID field defined in the product's USB Device Descriptor (see [\[USB 2.0\]](#) and [\[USB 3.1\]](#)).

6.4.4.3.1.2 Cert Stat VDO

The Cert Stat VDO **Shall** contain the XID assigned by USB-IF to the product before certification in binary format. The fields in the Cert Stat VDO **Shall** be as defined in Table 6-26.

Table 6-26 Cert Stat VDO

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

6.4.4.3.1.3 Product VDO

The Product VDO contains identity information relating to the product. The fields in the Product VDO **Shall** be as defined in Table 6-27.

Table 6-27 Product VDO

Bit(s)	Description	Reference
B31...16	16-bit unsigned integer. USB Product ID	[USB 2.0] / [USB 3.1]
B15...0	16-bit unsigned integer. bcdDevice	[USB 2.0] / [USB 3.1]

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

6.4.4.3.1.4 Cable VDO

The Cable VDO defined in this section **Shall** be sent when the Product Type is given as Passive or Active Cable. Table 6-28 and Table 6-29 define the Cable VDO which **Shall** be sent in each case.

6.4.4.3.1.4.1 Passive Cable VDO

A Passive Cable has a USB Plug on each end at least one of which is a Cable Plug supporting SOP' Communication. A Passive Cable **Shall Not** incorporate data bus signal conditioning circuits and hence has no concept of Super Speed Directionality. A Passive Cable **Shall** include a V_{BUS} wire and **Shall** only respond to SOP' Communication. Passive Cables **Shall** support the Structured VDM **Discover Identity** Command and **Shall** return the Passive Cable VDO in a **Discover Identity** Command ACK as shown in Table 6-28.

Table 6-28 Passive Cable VDO

Bit(s)	Field	Description
B31...28	HW Version	0000b...1111b assigned by the VID owner
B27...24	Firmware Version	0000b...1111b assigned by the VID owner
B23...20	Reserved	Shall be set to zero.
B19...18	USB Type-C plug to USB Type-A/B/C/Captive	00b = USB Type-A 01b = USB Type-B 10b = USB Type-C 11b = Captive
B17	Reserved	Shall be set to zero.
B16...13	Cable Latency	0000b - Reserved, Shall Not be used 0001b - <10ns (~1m) 0010b - 10ns to 20ns (~2m) 0011b - 20ns to 30ns (~3m) 0100b - 30ns to 40ns (~4m) 0101b - 40ns to 50ns (~5m) 0110b - 50ns to 60ns (~6m) 0111b - 60ns to 70ns (~7m) 1000b - > 70ns (>~7m) 1001b1111b Reserved, Shall Not be used Includes latency of electronics in Active Cable
B12...11	Cable Termination Type	00b = V_{CONN} not required. Cable Plugs that only support Discover Identity Commands Shall set these bits to 00b. 01b = V_{CONN} required 10b...11b = Reserved, Shall Not be used
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_{BUS} Current Handling Capability	00b = Reserved, Shall Not be used. 01b = 3A 10b = 5A 11b = Reserved, Shall Not be used.
B4	V_{BUS} through cable	0 = No 1 = Yes
B3	Reserved	Shall be set to 0.

The directionality bits **Shall** not be interpreted as having any significance.

Bit(s)	Field	Description
B2...0	USB SuperSpeed Signaling Support	000b = USB 2.0 only, no SuperSpeed support 001b = [USB 3.1] Gen1 010b = [USB 3.1] Gen1 and Gen2 011b... 111b = Reserved, Shall Not be used See [USB Type-C 1.2] for definitions.

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

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

The Connector Type field (B19...18) **Shall** contain a value corresponding to the connector type on the opposite end from the USB Type-C connector.

The Plug/Receptacle field (B17) **Shall** contain a value corresponding to whether the connector type as defined in the Connector Type field is a plug or receptacle. When the Connector Type field is Type-C or captive then receptacle **Shall Not** be indicated.

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

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

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

The USB SuperSpeed Signaling Support field (B2...0) **Shall** indicate whether the cable supports only [\[USB 2.0\]](#), or in addition Supports [\[USB 3.1\]](#) Gen1, or Gen1 and Gen2.

6.4.4.3.1.4.2 Active Cable VDO

An Active Cable has a USB Plug on each end at least one of which is a Cable Plug supporting SOP' Communication. An Active Cable **Shall** incorporate data bus signal conditioning circuits and **May** have a concept of Super Speed Directionality on its Super Speed wires. An Active Cable **May** include a V_{BUS} wire. An Active Cable **Shall** respond to SOP' Communication and **May** respond to SOP'' Communication. Active Cables **Shall** support the Structured VDM [Discover Identity](#) Command and **Shall** return the Active Cable VDO in a [Discover Identity](#) Command ACK as shown in Table 6-29.

Table 6-29 Active Cable VDO

Bit(s)	Field	Description
B31...28	HW Version	0000b...1111b assigned by the VID owner
B27...24	Firmware Version	0000b...1111b assigned by the VID owner
B23...20	Reserved	Shall be set to zero.
B19...18	USB Type-C plug to USB Type-A/B/C/Captive	00b = USB Type-A 01b = USB Type-B 10b = USB Type-C 11b = Captive
B17	Reserved	Shall be set to zero.
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) 1011b1111b Reserved, Shall not be used Includes latency of electronics in Active Cable
B12...11	Cable Termination Type	00b...01b = Reserved, Shall Not be used 10b = One end Active, one end passive, VCONN required 11b = Both ends Active, VCONN required
B10	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
The directionality bits Shall Not be interpreted as having any significance.		
B6...5	V _{BUS} Current Handling Capability	When V _{BUS} Through Cable is “No”, Reserved, Shall Not be used. When V _{BUS} Through Cable is “Yes”: 00b = Reserved, Shall Not be used. 01b = 3A 10b = 5A 11b = Reserved, Shall Not be used.
B4	V _{BUS} 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

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

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

The Connector Type field (B19...18) **Shall** contain a value corresponding to the connector type on the opposite end from the USB Type-C connector.

The Plug/Receptacle field (B17) **Shall** contain a value corresponding to whether the connector type as defined in the Connector Type field is a plug or receptacle. When the Connector Type field is Type-C or captive then receptacle **Shall Not** be indicated.

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

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

The Directionality Support field (B10...7) **Shall** indicate whether the direction of data flow through the signal conditioning circuits is fixed or configurable. An Active Cable **Shall** default the direction of data flow as SSTX to SSRX at power on. An Active Cable that supports data flow from SSTX to SSRX only **Shall** set B10...7 to zero, Fixed directionality support. An Active Cable which supports data flow from SSTX to SSRX and SSRX to SSTX **Shall** set B10...7 to one, Configurable directionality support. Each cable lane can be defined as either Fixed or Configurable directionality supported. The mechanism for configuration of the data bus signaling is to be defined.

The V_{BUS} Current Handling Capability field (B6...5) **Shall** indicate whether the cable is capable of carrying 3A or 5A. The V_{BUS} Current Handling Capability **Shall** only be **Valid** when the V_{BUS} Through Cable field indicates an end to end V_{BUS} wire.

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

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

The USB SuperSpeed Signaling Support field (B2...0) **Shall** indicate whether the cable supports only **[USB 2.0]**, or in addition Supports **[USB 3.1]** Gen1, or Gen1 and Gen2.

6.4.4.3.1.5 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 6-30 defines the AMA VDO which **Shall** be sent.

Table 6-30 AMA VDO

Bit(s)	Field	Description
B31...28	HW Version	0000b...1111b assigned by the VID owner
B27...24	Firmware Version	0000b...1111b assigned by the VID owner
B23...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

Bit(s)	Field	Description
B7...5	V _{CONN} power	When the V _{CONN} required field is set to "Yes" V _{CONN} power needed by adapter for full functionality 000b = 1W 001b = 1.5W 010b = 2W 011b = 3W 100b = 4W 101b = 5W 110b = 6W 111b = Reserved, Shall Not be used When the V _{CONN} required field is set to "No", Reserved, Shall be set to zero.
B4	V _{CONN} required	0 = No 1 = Yes
B3	V _{BUS} required	0 = No 1 = Yes
B2...0	USB SuperSpeed Signaling Support	000b = [USB 2.0] only 001b = [USB 3.1] Gen1 and USB 2.0 010b = [USB 3.1] Gen1, Gen2 and USB 2.0 011b = [USB 2.0] billboard only 100b...111b = Reserved, Shall Not be used

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

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

When the V_{CONN} required field indicates that V_{CONN} is required the V_{CONN} power field **Shall** indicate how much power the AMA needs in order to fully operate.

The V_{CONN} required field **Shall** indicate whether V_{CONN} is needed for the AMA to operate.

The V_{BUS} required field **Shall** indicate whether V_{BUS} is needed for the AMA to operate.

The USB SuperSpeed Signaling Support field (B2...0) **Shall** indicate whether the cable supports only **[USB 2.0]**, or in addition Supports **[USB 3.1]** Gen1, or Gen1 and Gen2 or **[USB 2.0]** billboard only.

6.4.4.3.2 Discover SVIDs

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

The SVID in the **Discover SVIDs** Command **Shall** be set to the **PD SID** (see Table 6-21) by both the Initiator and the Responder for this Command.

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

The **Discover SVIDs** Command ACK sent back by the Responder **Shall** contain one or more SVIDs. The SVIDs are returned 2 per VDO (see Table 6-31). If there are an odd number of supported SVIDs, the **Discover SVIDs** Command is returned ending with a SVID value of 0x0000 in the last part of the last VDO. If there are an even number of supported SVIDs, the **Discover SVIDs** Command is returned ending with an additional VDO containing two SVIDs with values of 0x0000. A Responder **Shall** only return SVIDs for which a **Discover Modes** Command request for that SVID will return at least one Mode.

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

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

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

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

Figure 6-7 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 6-31 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

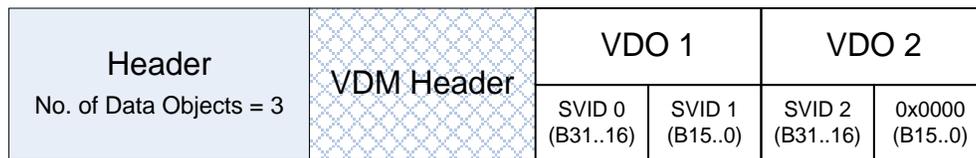


Figure 6-8 Example Discover SVIDs response with 4 SVIDs

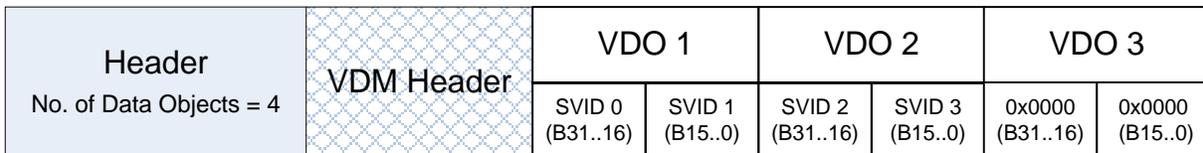
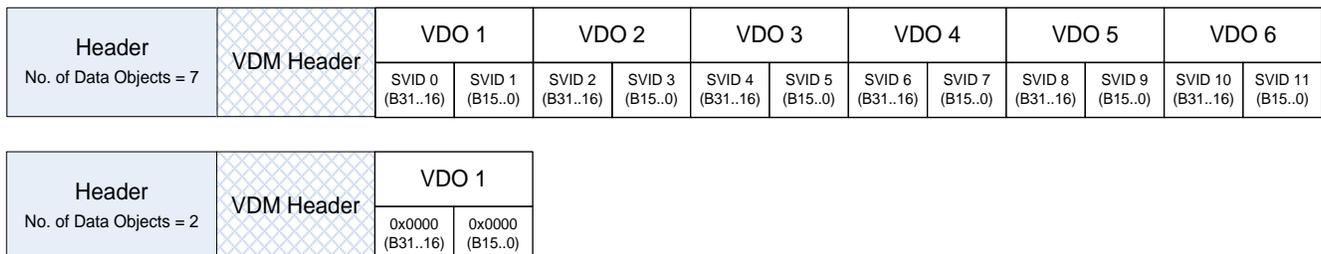


Figure 6-9 Example Discover SVIDs response with 12 SVIDs followed by an empty response



6.4.4.3.3 Discover Modes

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

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

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

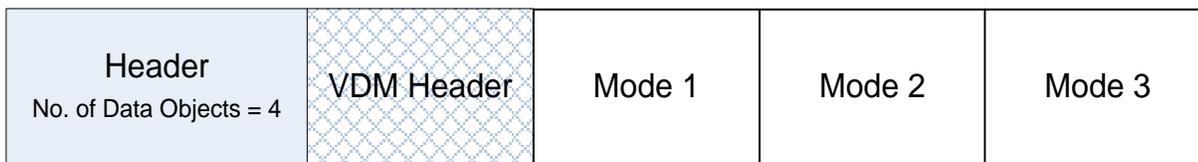
The *Discover Modes* Command ACK sent back by the Responder **Shall** contain one or more Modes. The *Discover Modes* Command ACK **Shall** contain a Message Header with the *Number of Data Objects* field set to a value of 1 to 7 (the actual value is the number of Mode objects plus one). If the ID is 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.

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

Figure 6-10 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 since the *Enter Mode* Command request **Shall Not** contain more than 1 VDO. When a VDO is included in an *Enter Mode* Command request the contents of the 32 bit VDO is defined by the Mode.

The *Number of Data Objects* field in the Command response **Shall** be set to 1 since an *Enter Mode* Command response (ACK, NAK, 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 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 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 shows a sequence that is Interrupted by a *Source_Capabilities* Message, that completes a Contract Negotiation, and then the sequence is Re-run. Figure 6-13 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

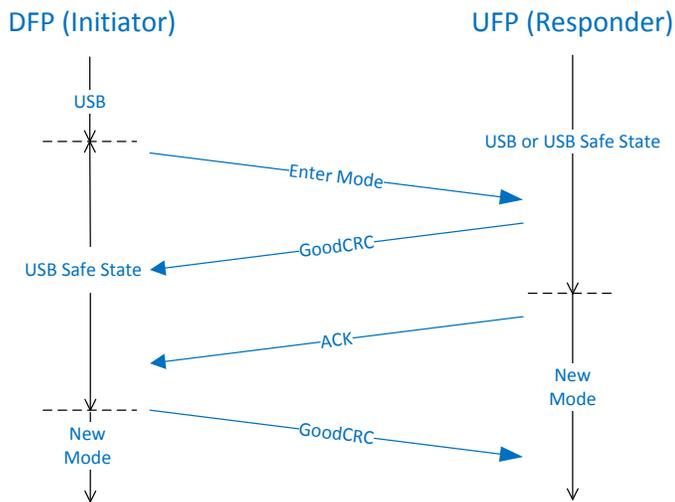


Figure 6-12 Enter Mode sequence Interrupted by Source Capabilities and then Re-run

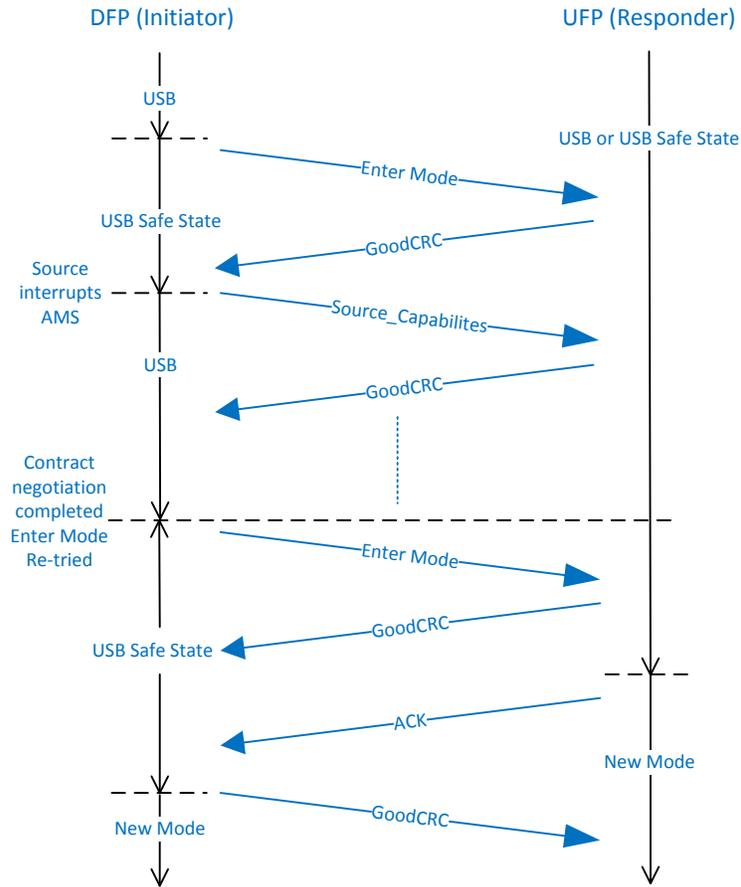
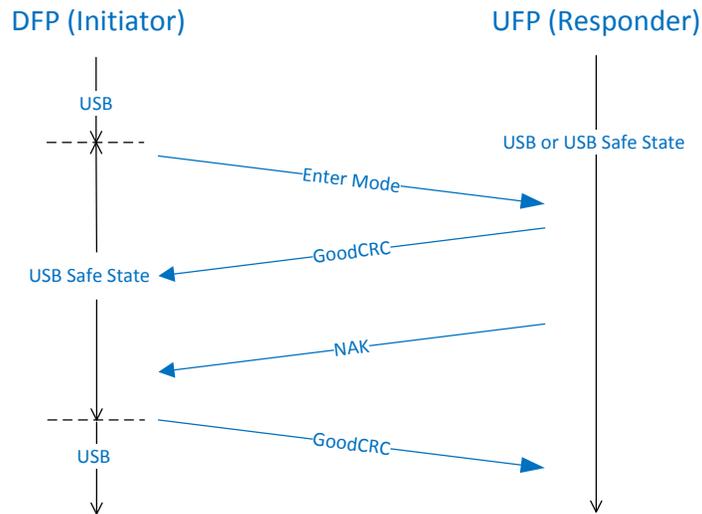


Figure 6-13 Unsuccessful Enter Mode sequence due to NAK



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 all Active Modes:

- A PD Hard Reset
- The Port Partners or Cable Plug(s) are Detached
- A Cable Reset (only exits the Cable Plug’s Active Modes)

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

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

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

6.4.4.3.5 Exit Mode Command

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

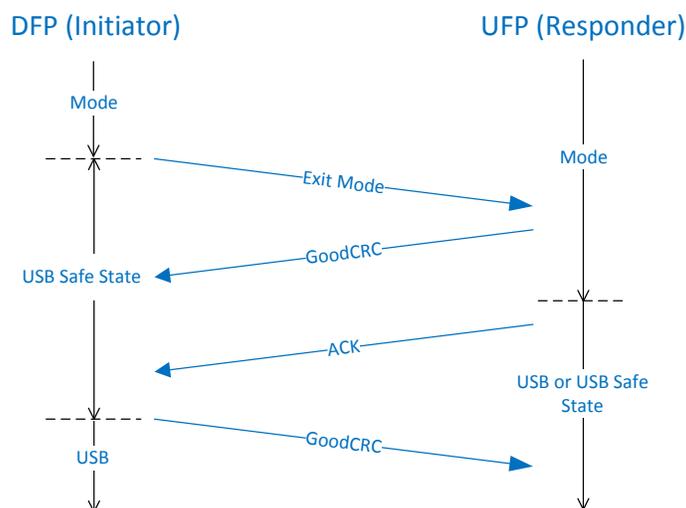
The value in the Object Position field **Shall** indicate to which Mode in the **Discover Modes** Command the VDO refers (see Figure 6-10) and **Shall** have been used previously in an **Enter Mode** Command request for an Active Mode. The value 1 always indicates the first Mode as it is the first object following the VDM Header. The value 2 refers to the next Mode and so forth. A value of 111b in the Object Position field **Shall** indicate that all Active Modes **Shall** be exited.

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

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

Figure 6-14 shows the sequence of events during the transition between exiting an Active Mode and USB operation. It illustrates when the Responder’s Mode changes and when the Initiator’s Mode changes.

Figure 6-14 Exit Mode sequence



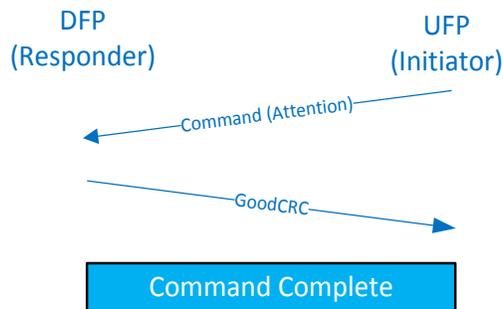
6.4.4.3.6 Attention

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. 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.

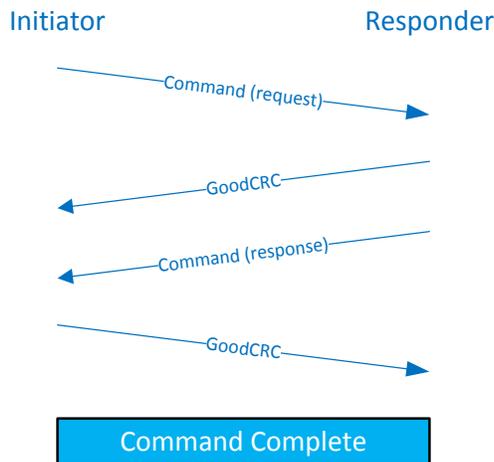
Figure 6-15 Attention Command request/response sequence



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 6-16 shows the request/response sequence including the **GoodCRC** Messages.

Figure 6-16 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.

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

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.

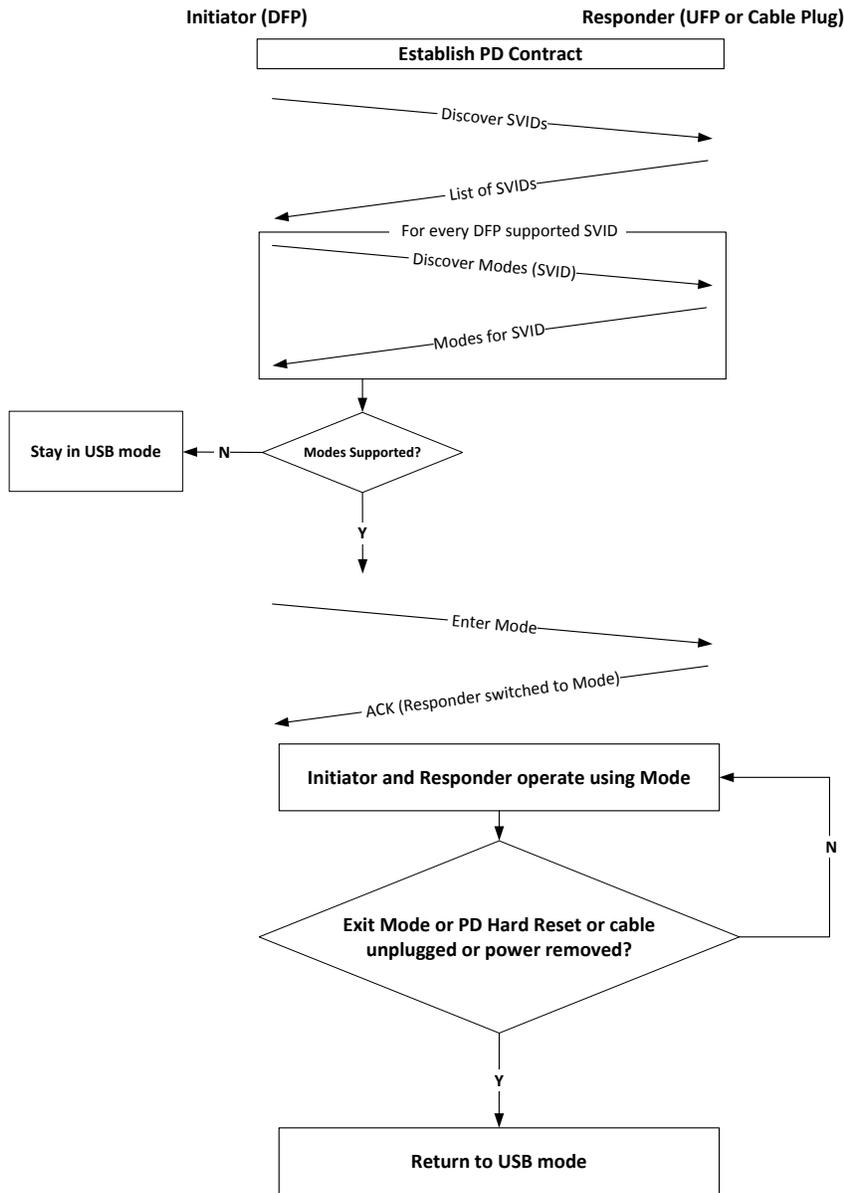
6.4.4.4.2 Enter Vendor Mode / Exit Vendor Mode Processes

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

The Responder (UFP or Cable Plug) and Initiator continue using the Active Mode until the Active Mode is exited. In a managed termination, using the *Exit Mode* Command, the Active Mode **Shall** be exited in a controlled manner as described in Section 6.4.4.3.5. In an unmanaged termination, triggered by a Power Delivery Hard Reset (i.e. **Hard Reset** Signaling sent by either Port Partner) or by cable Detach (device unplugged), the Active Mode **Shall** still be exited but there **Shall Not** be a transition through the USB Safe State. In both the managed and unmanaged terminations the Initiator and Responder return to USB operation as defined in [\[USB Type-C 1.2\]](#) following an exit from a Mode.

The overall Message flow is illustrated in Figure 6-17.

Figure 6-17 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.

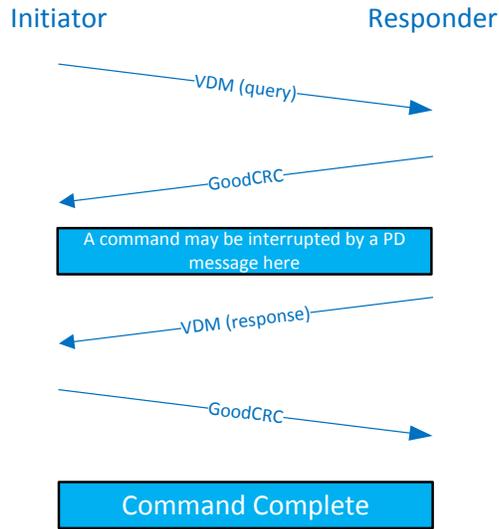
The timing and interspersing of VDMs between regular PD Messages **Shall** be done without perturbing the PD Message sequences. This requirement **Shall** apply to both Unstructured VDMs and Structured VDMs.

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 sequence will need to be Re-run after the USB PD Message sequence has completed.

The overall Message flow is illustrated in Figure 6-18.

Figure 6-18 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 to retry **nRetryCount** times. Note: that Cable Plugs do not retry Messages (see Section 6.6.2). Sending of the Preamble corresponding to the retried Message *Shall* start within **tRetry** of the **CRCReceiveTimer** expiring.

The **CRCReceiveTimer** *Shall* be started when the last bit of the Message **EOP** has been transmitted by the Physical Layer. The **CRCReceiveTimer** *Shall* be stopped when the last bit of the **EOP** corresponding to the **GoodCRC** Message has been received by the Physical Layer.

The Protocol Layer receiving a Message *Shall* respond with a **GoodCRC** Message within **tTransmit** in order to ensure that the sender's **CRCReceiveTimer** does not expire. The **tTransmit** *Shall* be measured from when the last bit of the Message **EOP** has been received by the Physical Layer until the first bit of the Preamble of the **GoodCRC** Message has been transmitted by the Physical Layer.

6.5.2 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 **[USB Type-C 1.2]** connector (see Section 6.3.5.2)

6.5.3.1 SourceActivityTimer

A PD Source that is not using a USB 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 might 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 8.3.3.3.7. 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 **[USB Type-C 1.2]** connector or
- Using a USB Type-A operating in its default Source role or using a USB 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 **tTypeC_SendSourceCap** for the USB 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 USB 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 Wait Timers and Times

6.5.5.1 SinkRequestTimer

The *SinkRequestTimer* is used to ensure that the time before the next Sink *Request* Message, after a *Wait* Message has been received from the Source in response to a Sink *Request* Message, is a minimum of *tSinkRequest* (see Section 6.3.12).

The *SinkRequestTimer* **Shall** be started when the *EOP* of a *Wait* Message has been received and **Shall** be stopped if any other Message is received or during a Hard Reset.

The Sink **Shall** wait at least *tSinkRequest*, after receiving the *EOP* of a *Wait* Message sent in response to a Sink *Request* Message, before sending a new *Request* Message. Whenever there is a *SinkRequestTimer* timeout the Sink **May** send a *Request* Message. It **Shall** then re-initialize and restart the *SinkRequestTimer*.

6.5.5.2 tPRSwapWait

The time before the next *PR_Swap* Message, after a *Wait* Message has been received in response to a *PR_Swap* Message is a minimum of *tPRSwapWait* min (see Section 6.3.12). The Port **Shall** wait at least *tPRSwapWait* after receiving the *EOP* of a *Wait* Message sent in response to a *PR_Swap* Message, before sending a new *PR_Swap* Message.

6.5.5.3 tDRSwapWait

The time before the next *DR_Swap* Message, after a *Wait* Message has been received in response to a *DR_Swap* Message is a minimum of *tDRSwapWait* min (see Section 6.3.12). The Port **Shall** wait at least *tDRSwapWait* after receiving the *EOP* of a *Wait* Message sent in response to a *DR_Swap* Message, before sending a new *DR_Swap* Message.

6.5.5.4 tVconnSwapWait

The time before the next *VCONN_Swap* Message, after a *Wait* Message has been received in response to a *VCONN_Swap* Message is a minimum of *tVCONNSwapWait* min (see Section 6.3.12). The Port **Shall** wait at least *tVCONNSwapWait* after receiving the *EOP* of a *Wait* Message sent in response to a *VCONN_Swap* Message, before sending a new *VCONN_Swap* Message.

6.5.6 Power Supply Timers

6.5.6.1 PSTransitionTimer

The *PSTransitionTimer* is used by the Policy Engine to timeout on a *PS_RDY* Message. It is started when a request for a new Capability has been accepted and will timeout after *tpSTransition* if a *PS_RDY* Message has not been received. This condition leads to a Hard Reset and a return to USB Default Operation. The *PSTransitionTimer* relates to the time taken for the Source to transition from one voltage, or current level, to another (see Section 7.1).

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 Power device that is currently acting as a Sink to timeout on a ***PS_RDY*** Message during a Power Role Swap sequence. This condition leads to a Hard Reset for USB Type-A and Type-B Plugs and Error Recovery for Type-C plugs and return to USB Default Operation.

If a ***PR_Swap*** Message request has been sent by the Dual-Role Power device currently acting as a Source the Sink can respond with an ***Accept*** Message. When the last bit of the ***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 Power 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 Power device to stop supplying power (see also Section 7.3.9 and Section 7.3.10). The timer **Shall** time out if a ***PS_RDY*** Message has not been received from the remote Dual-Role Power device within ***tPSSourceOff*** indicating this has occurred.

6.5.6.3 ***PSSourceOnTimer***

The ***PSSourceOnTimer*** is used by the Policy Engine in Dual-Role Power device that has just stopped sourcing power and is waiting to start sinking power to timeout on a ***PS_RDY*** Message during a Power Role Swap. This condition leads to a Hard Reset for USB Type-A and Type-B Plugs and Error Recovery for Type-C plugs 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 Power device to start sourcing power (see also Section 7.3.9 and Section 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 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 Section 6.5.8.5 and Section 6.4.3.1). 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 Soft Reset Timers

6.5.10.1 tSoftReset

A failure to see a *GoodCRC* Message in response to any Message within *tReceive* (after *nRetryCount* retries), when a Port Pair is Connected, is indicative of a communications failure. This **shall** cause the Source or Sink to send a *Soft_Reset* Message, transmission of which **shall** be completed within *tSoftReset* of the *CRCReceiveTimer* expiring.

6.5.10.2 tProtErrSoftReset

If the Protocol Error occurs that causes the Source or Sink to send a *Soft_Reset* Message, the transmission of the *Soft_Reset* Message **shall** be completed within *tProtErrSoftReset* of the *EOP* of the *GoodCRC* sent in response to the Message that caused the Protocol Error.

6.5.11 Hard Reset Timers

6.5.11.1 HardResetCompleteTimer

The *HardResetCompleteTimer* is used by the Protocol Layer in the case where it has asked the PHY Layer to send *Hard Reset* Signaling and the PHY Layer is unable to send the Signaling within a reasonable time due to a non-idle channel. If the PHY Layer does not indicate that the *Hard Reset* Signaling has been sent within *tHardResetComplete* of the Protocol Layer requesting transmission, then the Protocol Layer **shall** inform the Policy Engine that the *Hard Reset* Signaling has been sent in order to ensure the power supply is reset in a timely fashion.

6.5.11.2 PSHardResetTimer

The *PSHardResetTimer* is used by the Policy Engine in a Source to ensure that the Sink has had sufficient time to process *Hard Reset* Signaling before turning off its power supply to V_{BUS} .

When a Hard Reset occurs the Source stops driving V_{CONN} , removes R_p from the V_{CONN} pin and starts to transition the V_{BUS} voltage to *vSafe0V* either:

- *tPSHardReset* after the last bit of the *Hard Reset* Signaling has been received from the Sink or
- *tPSHardReset* after the last bit of the *Hard Reset* Signaling has been sent by the Source

See Section 7.1.6

6.5.11.3 **tDRSwapHardReset**

If a *DR_Swap* Message is received during Modal Operation then a Hard Reset **Shall** be initiated by the recipient of the unexpected *DR_Swap* Message; *Hard Reset* Signaling **Shall** be generated within *tDRSwapHardReset* of the EOP of the *GoodCRC* sent in response to the *DR_Swap* Message.

6.5.11.4 **tProtErrHardReset**

If a Protocol Error occurs that directly leads to a HardReset, the transmission of the *Hard Reset* Signaling **Shall** be completed within *tProtErrHardReset* of the *EOP* of the *GoodCRC* sent in response to the Message that caused the Protocol Error.

6.5.12 **Structured VDM Timers**

6.5.12.1 **VDMResponseTimer**

The *VDMResponseTimer* **Shall** be used by the Initiator's Policy Engine to ensure that a Structured VDM Command request needing a response (e.g. *Discover Identity* Command request) is responded to within a bounded time of *tVDMSenderResponse*. The *VDMResponseTimer* **Shall** be applied to all Structured VDM Commands except the *Enter Mode* and *Exit Mode* Commands which have their own timers (*VDMModeEntryTimer* and *VDMModeExitTimer* respectively). Failure to receive the expected response is detected when the *VDMResponseTimer* expires.

The Policy Engine's response when the *VDMResponseTimer* expires **Shall** be dependent on the Message sent (see Section 8.3).

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.

6.5.12.2 **VDMModeEntryTimer**

The *VDMModeEntryTimer* **Shall** be used by the Initiator's Policy Engine to ensure that the response to a Structured VDM *Enter Mode* Command request (ACK or NAK with ACK indicating that the requested Mode has been entered) arrives within a bounded time of *tVDMWaitModeEntry*. Failure to receive the expected response is detected when the *VDMModeEntryTimer* expires.

The Policy Engine's response when the *VDMModeEntryTimer* expires is to inform the Device Policy Manager (see Section 8.3.3.9.5).

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.

6.5.12.3 VDMModeExitTimer

The **VDMModeExitTimer** *Shall* be used by the Initiator's Policy Engine to ensure that the ACK response to a Structured VDM **Exit Mode** Command, indicating that the requested Mode has been exited, arrives within a bounded time of **tVDMWaitModeExit**. Failure to receive the expected response is detected when the **VDMModeExitTimer** expires.

The Policy Engine's response when the **VDMModeExitTimer** expires is to inform the Device Policy Manager (see Section 8.3.3.9.6).

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.

6.5.12.4 tVDMBusy

The Initiator *Shall* wait at least **tVDMBusy**, after receiving a BUSY Command response, before repeating the Structured VDM request again.

6.5.13 VCONN Timers

6.5.13.1 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.

6.5.13.2 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.

6.5.14 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.

6.5.15 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.

6.5.16 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.

6.5.17 Time Values and Timers

Table 6-32 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 6-33 lists the timers.

Table 6-32 Time Values

Parameter	Value (min)	Value (max)	Units	Reference
<i>tAttentionAverage</i>		10	s	Section 6.5.16
<i>tAttentionBurstSpacing</i>	100		ms	Section 6.5.16
<i>tAttentionSpacing</i>	250		ms	Section 6.5.16
<i>tBISTMode</i>		300	ms	Section 6.5.8.1
<i>tBISTContMode</i>	30	60	ms	Section 6.5.8.4
<i>tBISTReceive</i>	1.0	1.2	ms	Section 6.5.8.5
<i>tBISTResponse</i>		15	ms	Section 6.5.8.2
<i>tCableMessage</i>	750		µs	Section 6.5.14
<i>tDiscoverIdentity</i>	40	50	ms	Section 6.5.14
<i>tDRSwapHardReset</i>		15	ms	Section 6.5.11.3
<i>tDRSwapWait</i>	100		ms	Section 6.5.5.3
<i>tFirstSourceCap</i>		250	ms	Section 6.5.4.3
<i>tHardReset</i>		5	ms	Section 6.3.13
<i>tHardResetComplete</i>	4	5	ms	Section 6.5.10
<i>tNoResponse</i>	4.5	5.5	s	Section 6.5.7
<i>tProtErrHardReset</i>		15	ms	Section 6.5.11.4
<i>tProtErrSoftReset</i>		15	ms	Section 6.5.10.2
<i>tPRSwapWait</i>	100		ms	Section 6.5.5.2
<i>tPSHardReset</i>	25	35	ms	Section 6.5.11.2
<i>tPSSourceOff</i>	750	920	ms	Section 6.5.6.2
<i>tPSSourceOn</i>	390	480	ms	Section 6.5.6.3
<i>tPSTransition</i>	450	550	ms	Section 6.5.6.1
<i>tReceive</i>	0.9	1.1	ms	Section 6.5.1
<i>tReceiverResponse</i>		15	ms	Section 6.5.2
<i>tRetry</i>		75	µs	Section 6.5.1

Parameter	Value (min)	Value (max)	Units	Reference
<i>tSenderResponse</i>	24	30	ms	Section 6.5.2
<i>tSendSourceCap</i>	1	2	s	Section 6.5.4.1
<i>tSinkActivity</i>	120	150	ms	Section 6.5.3.2
<i>tSinkRequest</i>	100		ms	Section 6.5.5
<i>tSinkWaitCap</i>	2.1	2.5	s	Section 6.5.4.2
<i>tSoftReset</i>		15	ms	Section 6.5.3.1, 6.7.1
<i>tSourceActivity</i>	40	50	ms	Section 6.5.3.1
<i>tSwapSinkReady</i>		15	ms	Section 6.5.9.2
<i>tSwapSourceStart</i>	20		ms	Section 6.5.9.2
<i>tTransmit</i>		195	μs	Section 6.5.1
<i>tTypeCSendSourceCap</i>	100	200	ms	Section 6.5.4.1
<i>tTypeCSinkWaitCap</i>	310	620	ms	Section 6.5.4.2
<i>tVCONNSourceOff</i>		25	ms	Section 6.5.13
<i>tVCONNSourceOn</i>		100	ms	Section 6.5.13
<i>tVCONNSwapWait</i>	100		ms	Section 6.5.5.4
<i>tVDMBusy</i>	50		ms	Section 6.5.12.4
<i>tVDMEnterMode</i>		25	ms	Section 6.5.12.2
<i>tVDMExitMode</i>		25	ms	Section 6.5.12.3
<i>tVDMReceiverResponse</i>		15	ms	Section 6.5.12.1
<i>tVDMSenderResponse</i>	24	30	ms	Section 6.5.12.1
<i>tVDMWaitModeEntry</i>	40	50	ms	Section 6.5.12.2
<i>tVDMWaitModeExit</i>	40	50	ms	Section 6.5.12.3

Table 6-33 Timers

Timer	Parameter	Used By	Reference
<i>BISTContModeTimer</i>	<i>tBISTContMode</i>	Policy Engine	Section 6.5.8.4
<i>BISTReceiveErrorTimer</i>	<i>tBISTReceive</i>	Protocol	Section 6.5.8.5
<i>BISTStartTimer</i>	Defined by Tester	Policy Engine	Section 6.5.8.3
<i>CRCReceiveTimer</i>	<i>tReceive</i>	Protocol	Section 6.5.1
<i>DiscoverIdentityTimer</i>	<i>tDiscoverIdentity</i>	Policy Engine	Section 6.5.15
<i>HardResetCompleteTimer</i>	<i>tHardResetComplete</i>	Protocol	Section 6.5.10
<i>NoResponseTimer</i>	<i>tNoResponse</i>	Policy Engine	Section 6.5.7
<i>PShardResetTimer</i>	<i>tPShardReset</i>	Policy Engine	Section 6.5.11.2
<i>PSSourceOffTimer</i>	<i>tPSSourceOff</i>	Policy Engine	Section 6.5.6.2
<i>PSSourceOnTimer</i>	<i>tPSSourceOn</i>	Policy Engine	Section 6.5.6.3
<i>PSTransitionTimer</i>	<i>tPSTransition</i>	Policy Engine	Section 6.5.6.1
<i>SenderResponseTimer</i>	<i>tSenderResponse</i>	Policy Engine	Section 6.5.2
<i>SinkActivityTimer</i>	<i>tSinkActivity</i>	Policy Engine	Section 6.5.3.2
<i>SinkRequestTimer</i>	<i>tSinkRequest</i>	Policy Engine	Section 6.5.5

Timer	Parameter	Used By	Reference
<i>SinkWaitCapTimer</i>	<i>tSinkWaitCap,</i> <i>tTypeCSinkWaitCap</i>	Policy Engine	Section 6.5.4.2
<i>SourceActivityTimer</i>	<i>tSourceActivity</i>	Policy Engine	Section 6.5.3.1
<i>SourceCapabilityTimer</i>	<i>tSendSourceCap,</i> <i>tTypeCSendSourceCap</i>	Policy Engine	Section 6.5.4.1
<i>SwapRecoveryTimer</i>	<i>tSwapRecover</i>	Policy Engine	Section 6.5.9
<i>SwapSourceStartTimer</i>	<i>tSwapSourceStart</i>	Policy Engine	Section 6.5.9.2
<i>VCONNOnTimer</i>	<i>tVCONNSourceOn</i>	Policy Engine	Section 6.5.13.1
<i>VDMModeEntryTimer</i>	<i>tVDMWaitModeEntry</i>	Policy Engine	Section 6.5.12.2
<i>VDMModeExitTimer</i>	<i>tVDMWaitModeExit</i>	Policy Engine	Section 6.5.12.3
<i>VDMResponseTimer</i>	<i>tVDMSenderResponse</i>	Policy Engine	Section 6.5.12.1

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 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* decide that the Sink is non-responsive, stop sending *Source_Capabilities* Messages and disable PD.

A Sink *Shall* use the *SinkWaitCapTimer* to trigger the resending of *Source_Capabilities* Messages by a USB Power Delivery capable Source which has previously stopped sending *Source_Capabilities* Messages. Any Sink which is Attached and does not detect a *Source_Capabilities* Message, *Shall* issue *Hard Reset* Signaling when the *SinkWaitCapTimer* times out in order to reset the Source. Resetting the Source *Shall* also reset the *CapsCounter* and restart the sending of *Source_Capabilities* Messages.

6.6.5 BIST Error Counter

The Tester and UUT *Shall* maintain a count of errors detected *BISTErrorCounter* (see Section 6.4.3.1 and Section 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 [USB Type-C 1.2] 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 *AttentionAverage* period (see Section 6.4.4.3.6).

6.6.9 Counter Values and Counters

Table 6-35 lists the counters used in this section and Table 6-34 shows the corresponding parameters.

Table 6-34 Counter parameters

Parameter	Value	Reference
<i>nAttentionCount</i>	10	Section 6.6.8
<i>nBusyCount</i>	5	Section 6.6.7
<i>nCapsCount</i>	50	Section 6.6.4
<i>nDiscoverIdentityCount</i>	20	Section 6.6.6
<i>nHardResetCount</i>	2	Section 6.6.3
<i>nMessageIDCount</i>	7	Section 6.6.1
<i>nRetryCount</i>	3	Section 6.6.2

Table 6-35 Counters

Counter	Max	Reference
<i>BISTErrorCounter</i>	FFFFh	Section 6.6.5
<i>CapsCounter</i>	<i>nCapsCount</i>	Section 6.6.4
<i>DiscoverIdentityCounter</i>	<i>nDiscoverIdentityCount</i>	Section 6.6.6
<i>HardResetCounter</i>	<i>nHardResetCount</i>	Section 6.6.3
<i>MessageIDCounter</i>	<i>nMessageIDCount</i>	Section 6.6.1
<i>RetryCounter</i>	<i>nRetryCount</i>	Section 6.6.2
<i>VDMBusyCounter</i>	<i>nBusyCount</i>	Section 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 and Protocol Error

A **Soft_Reset** Message is used to cause a Soft Reset of protocol communication when this has broken down in some way. It **Shall Not** have any impact on power supply operation, but is used to correct a Protocol Error occurring during an Atomic Message Sequence (AMS). The Soft Reset **May** be triggered by either Port Partner in response to the Protocol Error.

Protocol Errors are any unexpected Message during an AMS. If the first Message in an AMS has been passed to the Protocol Layer by the Policy Engine but has not yet been sent (**GoodCRC** Message not received) when the Protocol Error occurs, the Policy Engine **Shall Not** issue a Soft Reset but **Shall** return to the **PE_SNK_Ready** or **PE_SRC_Ready** state and then process the incoming Message. If the Protocol Error occurs during an Interruptible AMS then the Policy Engine **Shall Not** issue a Soft Reset but **Shall** return to the **PE_SNK_Ready** or **PE_SRC_Ready** state and then process the incoming Message. If the incoming Message is an Unexpected Message received in the **PE_SNK_Ready** or **PE_SRC_Ready** state the Policy Engine **Shall** issue a Soft Reset. If the Protocol Error occurs during a Non-interruptible AMS this **Shall** lead to a Soft Reset in order to re-synchronize the Policy Engine state machines except when the voltage is transition when a Protocol Error **Shall** lead to a Hard Reset (see Section 8.3.3.2). Details of Interruptible and Non-interruptible AMS's can be found in Section 8.3.2.2.1.

An unrecognized or unsupported Message (except for a Structured VDM), received in the **PE_SNK_Ready** or **PE_SRC_Ready** states, **Shall Not** cause a **Soft_Reset** Message to be generated but instead a **Reject** Message **Shall** be generated.

Table 6 36 summarizes the responses that **Shall** be made to an incoming Message.

Table 6-36 Response to an incoming Message

Recipient's Power Role	Recipient's state	Incoming Message			
		Recognized			Unrecognized
		Supported		Unsupported	
		Expected	Unexpected		
Source	<i>PE_SRC_Ready</i>	Process Message	<i>Soft_Reset</i> Message	Reject (except for VDM) See Section 6.4.4.1 for UVDM, Section 6.10.3 for SVDM	Reject (except for VDM) See Section 6.4.4.1 for UVDM, Section 6.10.3 for SVDM
	During Interruptible AMS	Process Message	return to <i>PE_SRC_Ready</i> state and process Message		
	During Non-interruptible AMS (not power transitioned)	Process Message	<i>Soft_Reset</i> Message		
	During Non-interruptible AMS (power transitioned)	Process Message	<i>Hard_Reset</i> Signaling		
Sink	<i>PE_SNK_Ready</i>	Process Message	<i>Soft_Reset</i> Message	Reject (except for VDM) See Section 6.4.4.1 for UVDM, Section 6.10.3 for SVDM	Reject (except for VDM) See Section 6.4.4.1 for UVDM, Section 6.10.3 for SVDM
	During Interruptible AMS	Process Message	return to <i>PE_SNK_Ready</i> state and process Message		
	During Non-interruptible AMS (not power transitioned)	Process Message	<i>Soft_Reset</i> Message		
	During Non-interruptible AMS (power transitioned)	Process Message	<i>Hard_Reset</i> Signaling		

A failure to see a *GoodCRC* Message in response to any Message within *tReceive* (after *nRetryCount* retries), when a Port Pair is Connected, is indicative of a communications failure resulting in a Soft Reset (see Section 6.5.10.1).

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 8-7). Message numbers **Shall** be set to zero prior to sending the *Soft_Reset/Accept* Message since the issue might be with the counters. The sender of a *Soft_Reset* Message **Shall** reset its *MessageIDCounter* and *RetryCounter*, the receiver of the Message **Shall** reset its *MessageIDCounter* and *RetryCounter* before sending the *Accept* Message response. Any failure in the Soft Reset process will trigger a Hard Reset when SOP Packets are being used or Cable Reset for any other SOP* Packets; for example a *GoodCRC* Message is not received during the Soft Reset process (see Section 6.7.2 and Section 6.7.3).

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 Section 7.3.12 to Section 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 Section 6.5.7 and Section 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 USB Type-A/B Power Role Swap, a Hard Reset **Shall** cause the Port Partners to return to their default Source/Sink roles. A USB Type-A Port **Shall** return to Source operation. A USB Type-B 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 [USB Type-C 1.2]) 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 [USB Type-C 1.2]).

If the Hard Reset is insufficient to clear the error condition then the Port **Should** use error recovery mechanisms as defined in [USB Type-C 1.2].

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 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. A DFP **Shall** only generate **Cable Reset** Signaling within an Explicit Contract.

A Cable Reset **Shall** cause all Active Modes in the Cable Plugs to be exited (see Section 6.4.4.3.4).

6.8 Message Discarding

On receiving a received Message on **SOP**, the Protocol Layer **Shall Discard** any pending SOP* Messages. A received Message on **SOP'/SOP''** **Shall Not** cause any pending **SOP*** Messages to be **Discarded**.

It is assumed that Messages using **SOP'/SOP''** constitute a simple request/response AMS, with the Cable Plug providing the response so there is no reason for a pending **SOP*** Message to be **Discarded**. There can only be one AMS between the Port Partners and these also take priority over Cable Plug communications so a Message received on **SOP** will always cause a Message pending on **SOP*** to be **Discarded**.

See Table 6-37 for details of the Messages that **Shall/Shall Not** be **Discarded**.

Table 6-37 Message discarding

Message pending transmission	Message received	Discard pending transmission?
<i>SOP</i>	<i>SOP</i>	Yes
<i>SOP</i>	<i>SOP'/SOP''</i>	No
<i>SOP'</i>	<i>SOP</i>	Yes
<i>SOP'</i>	<i>SOP'</i>	No
<i>SOP'</i>	<i>SOP''</i>	No
<i>SOP''</i>	<i>SOP</i>	Yes
<i>SOP''</i>	<i>SOP'</i>	No
<i>SOP''</i>	<i>SOP''</i>	No

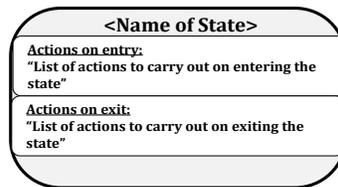
6.9 State behavior

6.9.1 Introduction to state diagrams used in Chapter 6

The state diagrams defined in Section 6.9 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 6-19 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 6-19 Outline of States

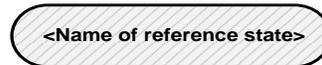


Transitions from one state to another are indicated by arrows with the conditions listed on the arrow. Where there are multiple conditions these are connected using either a logical OR “|” or a logical AND “&”. The inverse of a condition is shown with a “NOT” in front of the condition.

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 6-20 indicates how such references are made. The reference is indicated with a hatched box. The box contains the name of the referenced state.

Figure 6-20 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.)

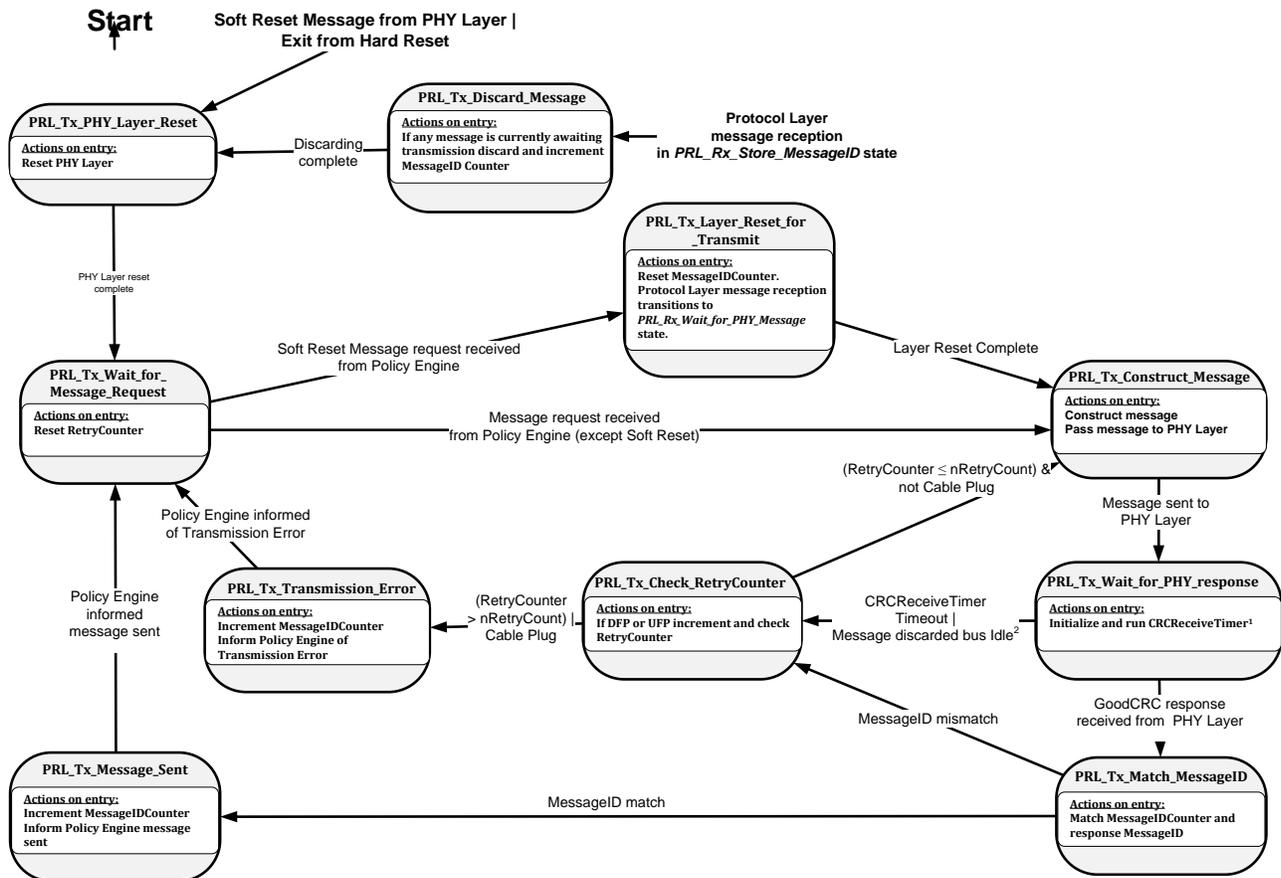
6.9.2 State Operation

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).

6.9.2.1 Protocol Layer Message Transmission

Figure 6-21 shows the state behavior for the Protocol Layer when transmitting a Message.

Figure 6-21 Protocol Layer Message transmission



¹ 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).

² 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.

³ See Section 6.8 for details of when Messages are **Discarded**.

6.9.2.1.1 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.

6.9.2.1.2 PRL_Tx_Wait_for_Message_Request state

In the *PRL_Tx_Wait_for_Message_Request* state the Protocol Layer waits until the Policy Engine directs it to send a Message.

On entry to the *PRL_Tx_Wait_for_Message_Request* state the Protocol Layer **Shall** reset the *RetryCounter*.

The Protocol Layer **Shall** transition to the *PRL_Tx_Construct_Message* state when:

- A Message request is received from the Policy Engine which is not a *Soft_Reset* Message.

The Protocol Layer **Shall** transition to the *PRL_Tx_Layer_Reset_for_Transmit* state when:

- A Message request is received from the Policy Engine which is a *Soft_Reset* Message.

6.9.2.1.3 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 6.9.2.2.1) 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.

6.9.2.1.4 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.

6.9.2.1.5 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 received from the PHY Layer.

The Protocol Layer **Shall** transition to the *PRL_Tx_Check_RetryCounter* state when:

- The *CRCReceiveTimer* times out
- Or the PHY Layer indicates that a Message has been *Discarded* due to the channel being busy but the channel is now idle (see Section 5.7).

6.9.2.1.6 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.

6.9.2.1.7 PRL_Tx_Message_Sent state

On entry to the **PRL_Tx_Message_Sent** state the Protocol Layer **Shall** increment the **MessageIDCounter** and inform the Policy Engine that the Message has been sent.

The Protocol Layer **Shall** transition to the **PRL_Tx_Wait_for_Message_Request** state when:

- The Policy Engine has been informed that the Message has been sent.

6.9.2.1.8 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** \leq **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.

6.9.2.1.9 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.

6.9.2.1.10 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 according to the rules in Section 6.8 and increment the **MessageIDCounter**.

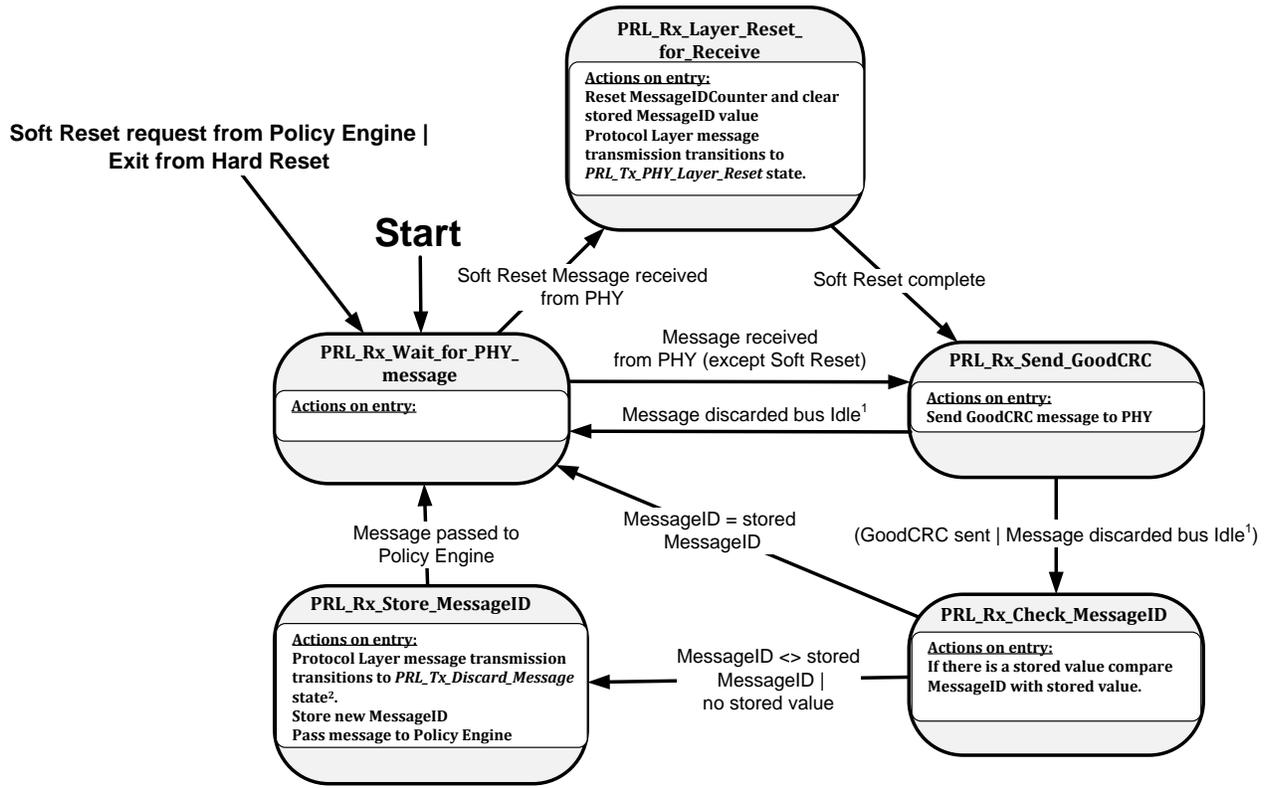
The Protocol Layer **Shall** transition to the **PRL_Tx_PHY_Layer_Reset** state when:

- Discarding is complete i.e. the Message queue is empty.

6.9.2.2 Protocol Layer Message Reception

Figure 6-22 shows the state behavior for the Protocol Layer when receiving a Message.

Figure 6-22 Protocol layer Message reception



¹ 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.

² In the case of a Ping message being received, in order to maintain robust communications in the presence of collisions, the outgoing message **Shall Not** be **Discarded**.

³ See Section 6.8 for details of when Messages are **Discarded**.

6.9.2.2.1 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.

6.9.2.2.2 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 6.9.2.1.1).

The Protocol Layer **Shall** transition to the **PRL_Rx_Send_GoodCRC** State when:

- The Soft Reset actions in this state have been completed.

6.9.2.2.3 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

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

6.9.2.2.4 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

6.9.2.2.5 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 **Shall 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.

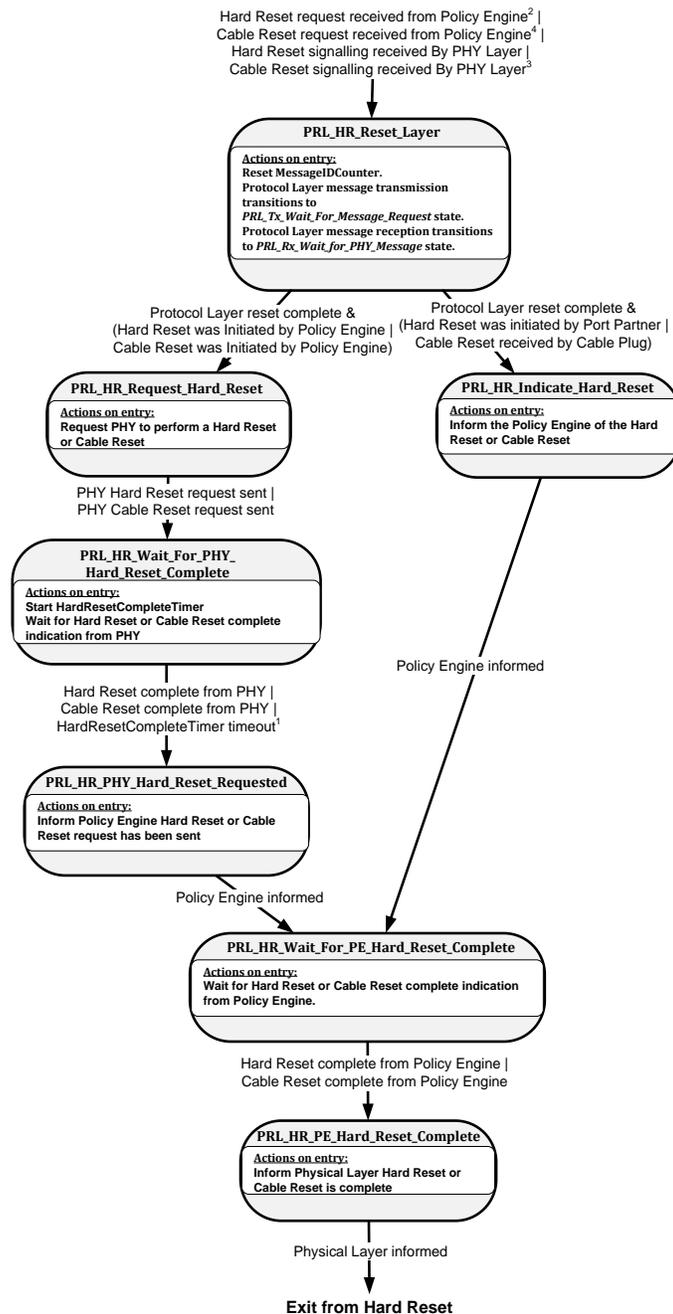
The Protocol Layer **Shall** transition to the **PRL_Rx_Wait_for_PHY_Message** state when:

- The Message has been passed up to the Policy Engine.

6.9.2.3 Hard Reset operation

Figure 6-23 shows the state behavior for the Protocol Layer when receiving a Hard Reset or Cable Reset request from the Policy Engine or **Hard Reset** Signaling or **Cable Reset** Signaling from the Physical Layer (see also Section 6.7.2 and Section 6.7.3).

Figure 6-23 Hard/Cable Reset



¹ If the *HardResetCompleteTimer* timeout occurs this means that the PHY is still waiting to send the Hard Reset due to a non-idle channel. This condition will be cleared once the PE Hard Reset is completed.

² Cable Plugs do not generate *Hard Reset* signaling but are required to monitor for *Hard Reset* signaling between the Port Partners and respond by resetting.

³ Cable Reset signaling is only recognized by a Cable Plug.

⁴ Cable Reset signaling cannot be generated by Cable Plugs.

6.9.2.3.1 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 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 can 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).

6.9.2.3.2 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.

6.9.2.3.3 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.

6.9.2.3.4 PRL_HR_Wait_for_PHY_Hard_Reset_Complete state

In the **PRL_HR_PHY_Hard_Reset_Requested** 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.

6.9.2.3.5 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.

6.9.2.3.6 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.

6.9.2.3.7 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:

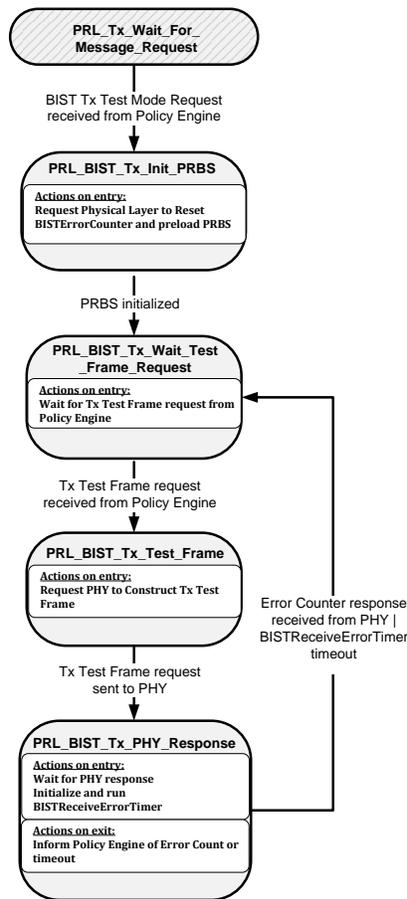
- The Physical Layer has been informed that the Hard Reset is complete so that it will re-enable the communications channel. If **Hard Reset** Signaling is still pending due to a non-idle channel this **Shall** be cleared and not sent or
- The Physical Layer has been informed that the Cable Reset is complete.

6.9.3 BIST Operation

6.9.3.1 BIST Transmitter Test

Figure 6-24 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 6.9.2.1) to BIST Transmitter Test Mode when directed by the Policy Engine.

Figure 6-24 BIST Transmitter Test



6.9.3.1.1 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 6.9.2.1.1) 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.

6.9.3.1.2 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.

6.9.3.1.3 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.

6.9.3.1.4 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.

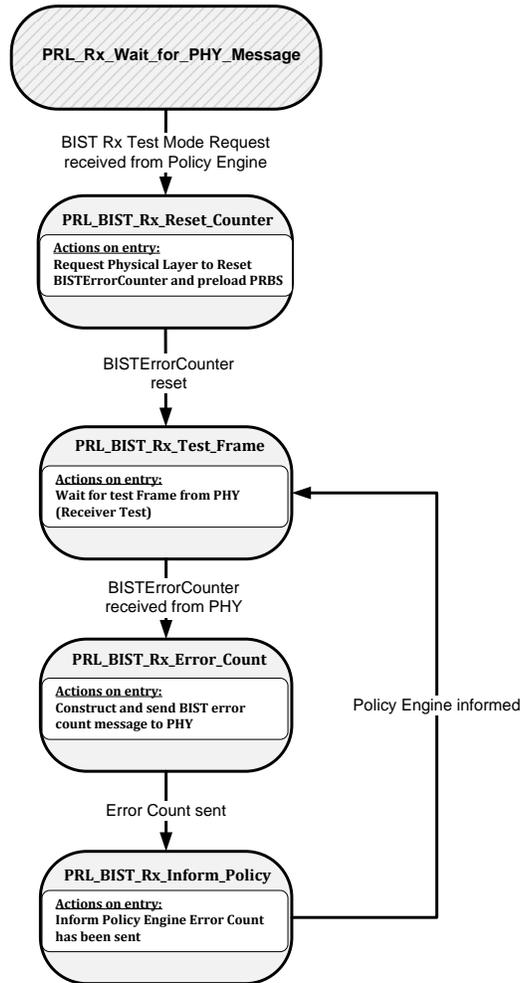
The Protocol Layer **Shall** transition to the *PRL_BIST_Tx_Wait_Test_Frame_Request* state (see Section 6.9.3.1.2) when:

- An error counter response is received from the Physical Layer.
- Or the *BISTReceiveErrorTimer* times out.

6.9.3.2 BIST Receiver Test

Figure 6-25 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 6.9.2.2) to BIST Receiver Test Mode when directed by the Policy Engine. Exit from this mode of operation is via Hard Reset (see Section 6.9.2.3).

Figure 6-25 BIST Receiver Test



6.9.3.2.1 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.

6.9.3.2.2 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.

6.9.3.2.3 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.

6.9.3.2.4 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.

6.9.4 List of Protocol Layer States

Table 6-38 lists the states used by the various state machines.

Table 6-38 Protocol Layer States

State name	Reference
Protocol Layer Message Transmission	
<i>PRL_Tx_PHY_Layer_Reset</i>	Section 6.9.2.1.1
<i>PRL_Tx_Wait_for_Message_Request</i>	Section 6.9.2.1.2
<i>PRL_Tx_Layer_Reset_for_Transmit</i>	Section 6.9.2.1.3
<i>PRL_Tx_Construct_Message</i>	Section 6.9.2.1.4
<i>PRL_Tx_Wait_for_PHY_Response</i>	Section 6.9.2.1.5
<i>PRL_Tx_Match_MessageID</i>	Section 6.9.2.1.6
<i>PRL_Tx_Message_Sent</i>	Section 6.9.2.1.7
<i>PRL_Tx_Check_RetryCounter</i>	Section 6.9.2.1.8
<i>PRL_Tx_Transmission_Error</i>	Section 6.9.2.1.9
<i>PRL_Tx_Discard_Message</i>	Section 6.9.2.1.10
Protocol Layer Message Reception	
<i>PRL_Rx_Wait_for_PHY_Message</i>	Section 6.9.2.2.1
<i>PRL_Rx_Layer_Reset_for_Receive</i>	Section 6.9.2.2.2
<i>PRL_Rx_Send_GoodCRC</i>	Section 6.9.2.2.3
<i>PRL_Rx_Check_MessageID</i>	Section 6.9.2.2.4
<i>PRL_Rx_Store_MessageID</i>	Section 6.9.2.2.5
Hard Reset Operation	
<i>PRL_HR_Reset_Layer</i>	Section 6.9.2.3.1
<i>PRL_HR_Indicate_Hard_Reset</i>	Section 6.9.2.3.2
<i>PRL_HR_Request_Hard_Reset</i>	Section 6.9.2.3.3
<i>PRL_HR_Wait_for_PHY_Hard_Reset_Complete</i>	Section 6.9.2.3.4
<i>PRL_HR_PHY_Hard_Reset_Requested</i>	Section 6.9.2.3.5
<i>PRL_HR_Wait_for_PE_Hard_Reset_Complete</i>	Section 6.9.2.3.6
<i>PRL_HR_PE_Hard_Reset_Complete</i>	Section 6.9.2.3.7
BIST Transmitter Test	
<i>PRL_BIST_Tx_Init_PRBS</i>	Section 6.9.3.1.1
<i>PRL_BIST_Tx_Wait_Test_Frame_Request</i>	Section 6.9.3.1.2
<i>PRL_BIST_Tx_Test_Frame</i>	Section 6.9.3.1.3
<i>PRL_BIST_Tx_PHY_Response</i>	Section 6.9.3.1.4
BIST Receiver Test	
<i>PRL_BIST_Rx_Reset_Counter</i>	Section 6.9.3.2.1
<i>PRL_BIST_Rx_Test_Frame</i>	Section 6.9.3.2.2
<i>PRL_BIST_Rx_Error_Count</i>	Section 6.9.3.2.3
<i>PRL_BIST_Rx_Inform_Policy</i>	Section 6.9.3.2.4

6.10 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:

- N – **Normative; Shall** be supported by this Port/Cable Plug
- CN – **Conditional Normative ; 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
- RJ - Reject; this Port/Cable Plug **Shall** return a *Reject* Message when received.
- I – **Ignore; Shall** be **Ignored** by this Port/Cable Plug when received.
- NK – NAK; this Port/Cable Plug **Shall** return Responder NAK to this Command when NA.
- NA – Not allowed; **Shall Not** be transmitted by this Port/Cable Plug.
- DR- Don't Recognize; there **Shall** no response at all (i.e. not even a *GoodCRC* Message) from this Port/Cable Plug when received.

For the case of **Conditional Normative** a note has been added to indicate the condition. “CN/” notation is used to indicate the level of support when the condition is not present.

“R/” and “O/” notation is used to indicate the response when the Recommended or **Optional** Message is not supported.

Note: that where NS/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.

6.10.1 Applicability of Control Messages

Table 6-39 details Control Messages that **Shall/Should/Shall Not** be transmitted and received by a Source, Sink or Cable Plug. Requirements for Dual-Role Power Ports and Dual-Role Data Ports **Shall** override any requirements for Source-only or Sink-Only Ports.

Table 6-39 Applicability of Control Messages

Message Type	Source	Sink	Dual-Role Power	Dual-Role Data	Cable Plug
Transmitted Message					
<i>Accept</i>	N	N			N
<i>DR_Swap</i>	O	O		N	NA
<i>Get_Sink_Cap</i>	R	NA	N		NA
<i>Get_Source_Cap</i>	NA	R	N		NA
<i>GoodCRC</i>	N	N			N
<i>GotoMin</i>	CN ¹ /O	NA			NA
<i>Ping</i>	O	NA			NA
<i>PR_Swap</i>	NA	NA	N		NA
<i>PS_RDY</i>	N	NA	N		NA

Message Type	Source	Sink	Dual-Role Power	Dual-Role Data	Cable Plug
<i>Reject</i>	N	NA	O	O	NA
<i>Soft_Reset</i>	N	N			NA
<i>VCONN_Swap</i>	R	R			NA
<i>Wait</i>	CN ² /O	NA	O	O	NA
Received Message					
<i>Accept</i>	N	N	N	N	I
<i>DR_Swap</i>	O/RJ	O/ RJ		N	I
<i>Get_Sink_Cap</i>	RJ	N	N		I
<i>Get_Source_Cap</i>	N	RJ	N		I
<i>GoodCRC</i>	N	N			N
<i>GotoMin</i>	RJ	R ³			I
<i>Ping</i>	RJ	I			I
<i>PR_Swap</i>	RJ	RJ	N		I
<i>PS_RDY</i>	RJ	N	N		I
<i>Reject</i>	CN ⁵ /RJ	N	N	N	I
<i>Soft_Reset</i>	N	N			N
<i>VCONN_Swap</i>	CN ⁴ / RJ	CN ⁴ / RJ			I
<i>Wait</i>	CN ⁵ /RJ	N	N	N	I
<p>Note 1: Shall be supported by a Hub with multiple Downstream Ports. Should be supported by a Host with multiple Downstream Ports.</p> <p>Note 2: Shall be supported when transmission of <i>GotoMin</i> Messages is supported.</p> <p>Note 3: Should be supported by Sinks which use PD power for charging.</p> <p>Note 4: Shall be supported by any Port that can operate as a VCONN Source.</p> <p>Note 5: Shall be supported when <i>VCONN_Swap</i> is supported.</p>					

6.10.2 Applicability of Data Messages

Table 6-40 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 6-40 Applicability of Data Messages

Message Type	Source	Sink	Dual-Role Power	Cable Plug
Transmitted Message				
<i>Source_Capabilities</i>	N	NA	N	NA
<i>Request</i>	NA	N		NA
<i>BIST</i>	N ¹	N ¹		NA
<i>Sink_Capabilities</i>	NA	N	N	NA
Received Message				
<i>Source_Capabilities</i>	RJ	N	N	I
<i>Request</i>	N	RJ		I
<i>BIST</i>	N ¹	N ¹		N ¹

Message Type	Source	Sink	Dual-Role Power	Cable Plug
<i>Sink_Capabilities</i>	N	RJ	N	I
Note 1: For details of which BIST Modes and Messages Shall be supported see Section 5.9 and Section 6.4.3.				

6.10.3 Applicability of Structured VDM Commands

Table 6-41 details Structured VDM Commands that **Shall/Should/Shall Not** be transmitted and received by a DFP, UFP or Cable Plug. If Structured VDMs are not supported, a Structured VDM Command received by a DFP or UFP **Shall** be **Ignored**.

Table 6-41 Applicability of Structured VDM Commands

Command Type	DFP	UFP	Cable Plug
Transmitted Command Request			
<i>Discover Identity</i>	CN ¹ /R	R ²	NA
<i>Discover SVIDs</i>	CN ¹ /O	NA	NA
<i>Discover Modes</i>	CN ¹ /O	NA	NA
<i>Enter Mode</i>	CN ¹ /NA	NA	NA
<i>Exit Mode</i>	CN ¹ /NA	NA	NA
<i>Attention</i>	NA	O	NA
Received Command Request/Transmitted Command Response			
<i>Discover Identity</i>	NK ³	CN ¹ /R/NK ³	N
<i>Discover SVIDs</i>	NK ³	CN ¹ /NK ³	CN ¹ /NK
<i>Discover Modes</i>	NK ³	CN ¹ /NK ³	CN ¹ /NK
<i>Enter Mode</i>	NK ³	CN ¹ /NK ³	CN ¹ /NK
<i>Exit Mode</i>	NK ³	CN ¹ /NK ³	CN ¹ /NK
<i>Attention</i>	O/I	O/I	I
<p>Note 1: Shall be supported when Modal Operation is supported.</p> <p>Note 2: May be transmitted by a UFP/Source during cable discovery (see Section 6.4.4.3.1 and Section 8.3.3.10.11).</p> <p>Note 3: If Structured VDMs are not supported, a Structured VDM Command received by a DFP or UFP Shall be Ignored.</p>			

6.10.4 Applicability of Reset Signaling

Table 6-42 details Reset Signaling that *Shall/Should/Shall Not* be transmitted and received by a DFP/UFP or Cable Plug.

Table 6-42 Applicability of Reset Signaling

Signaling Type	DFP	UFP	Cable Plug
Transmitted Message/Signaling			
<i>Soft_Reset</i>	N	N	NA
<i>Hard_Reset</i>	N	N	NA
<i>Cable_Reset</i>	CN ¹	NA	NA
Received Message/Signaling			
<i>Soft_Reset</i>	N	N	N
<i>Hard_Reset</i>	N	N	N
<i>Cable_Reset</i>	DR	DR	N
Note 1: <i>Shall</i> be supported when transmission of SOP' Packets are supported.			

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], [USB Type-C 1.2] or [USB 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], [USB Type-C 1.2] or [USB 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.

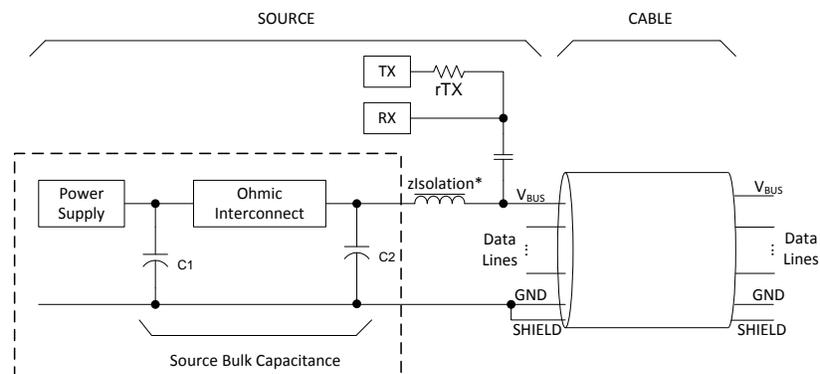
- **Shall** support Dead Battery operation as defined in Section 4.2.1 for USB Type-A to USB Type-B connections and as defined in [USB Type-C 1.2] for USB Type-C to USB Type-C connections.
- **Shall** return to Sink operation when *Hard Reset* Signaling is received.
- **Shall** control V_{BUS} voltage 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 Source bulk capacitance consists of C1 and C2 as shown in Figure 7-1. The Ohmic Interconnect might consist of PCB traces for power distribution or power switching devices. The capacitance might be a single capacitor, a capacitor bank or distributed capacitance. If the power supply is shared across multiple ports, the 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



* $z_{isolation}$ is only required where BFSK over V_{BUS} signaling is implemented otherwise this can be omitted

7.1.3 Types of Sources

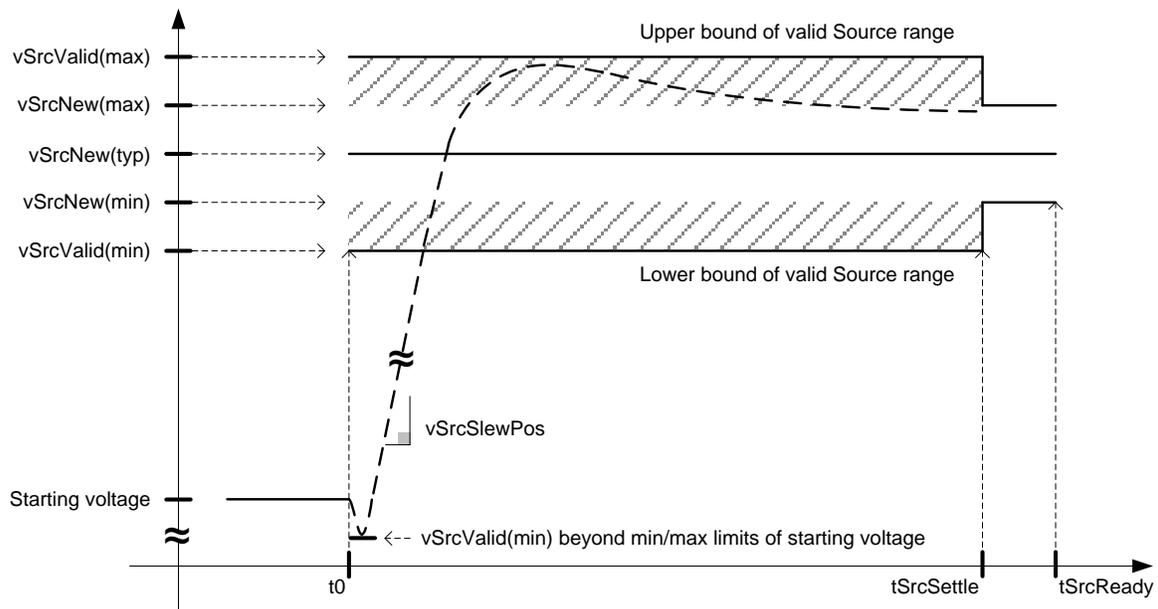
Consistent with the Power Data Objects discussed in Section 6.4.1, the three possible power supply types that are available as Sources in a USB Power Delivery System are:

- The Fixed Supply PDO exposes well-regulated fixed voltage power supplies. Sources **Shall** support at least one Fixed Supply capable of supplying **vSafe5V**. The output voltage of a Fixed Supply **Shall** remain within the range defined by the relative tolerance **vSrcNew** and the absolute band **vSrcValid** as listed in Table 7-22 and described in Section 7.1.9.
- The Variable Supply (non-Battery) PDO exposes very poorly regulated Sources. The output voltage of a Variable Supply (non-Battery) **Shall** remain within the absolute maximum output voltage and the absolute minimum output voltage exposed in the Variable Supply PDO.
- The Battery Supply PDO exposes Batteries that can be connected directly as a Source to V_{BUS} . The output voltage of a Battery Supply **Shall** remain within the absolute maximum output voltage and the absolute minimum output voltage exposed in the Battery Supply PDO.

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 nominal value for **vSrcNew**. 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 transitioning Source output voltage **Shall** settle within **vSrcNew** by **tSrcSettle**. 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. The starting time, t_0 , in Figure 7-2 starts **tSrcTransition** after the last bit of the **EOP** of the **GoodCRC** Message has been received by the Source.

Figure 7-2 Transition Envelope for Positive Voltage Transitions

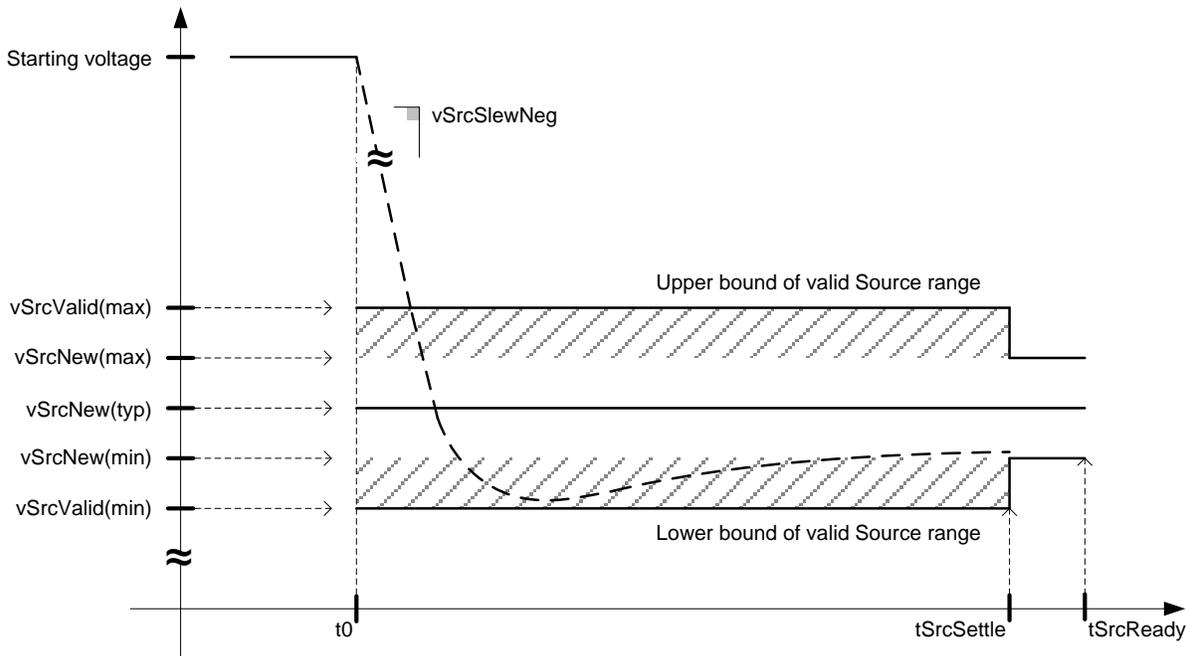


At the start of the positive voltage transition the V_{BUS} voltage level **Shall Not** droop **vSrcValid** min below either **vSrcNew** (i.e., if the starting V_{BUS} voltage level is not **vSafe5V**) or **vSafe5V** as applicable.

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 *vSafe0V* transitions. The slew rate of the negative transition **Shall Not** exceed *vSrcSlewNeg*. The negative voltage transition **Shall** remain monotonic while the transitioning voltage is above *vSrcValid* max and **Shall** remain within the *vSrcValid* range upon crossing *vSrcValid* max as shown in Figure 7-3. The starting time, t_0 , in Figure 7-3 starts *tSrcTransition* after the last bit of the *EOP* of the *GoodCRC* Message has been received by the Source.

Figure 7-3 Transition Envelope for Negative Voltage Transitions



If the newly negotiated voltage is *vSafe5V*, then the *vSrcValid* limits **Shall** determine the transition window and the transitioning Source **Shall** settle within the *vSafe5V* limits by *tSrcSettle*.

7.1.6 Response to Hard Resets

Hard Reset Signaling indicates a communication failure has occurred and the Source **Shall** stop driving V_{CONN} , **Shall** remove R_p from the V_{CONN} pin and **Shall** drive V_{BUS} to *vSafe0V* as shown in Figure 7-4. The USB connection **May** reset during a Hard Reset since the V_{BUS} voltage will be less than *vSafe5V* for an extended period of time. After establishing the *vSafe0V* voltage condition on V_{BUS} , the Source **Shall** wait *tSrcRecover* before re-applying V_{CONN} and restoring V_{BUS} to *vSafe5V*. A Source **Shall** conform to the V_{CONN} timing as specified in [USB Type-C 1.2].

Device operation during and after a Hard Reset is defined as follows:

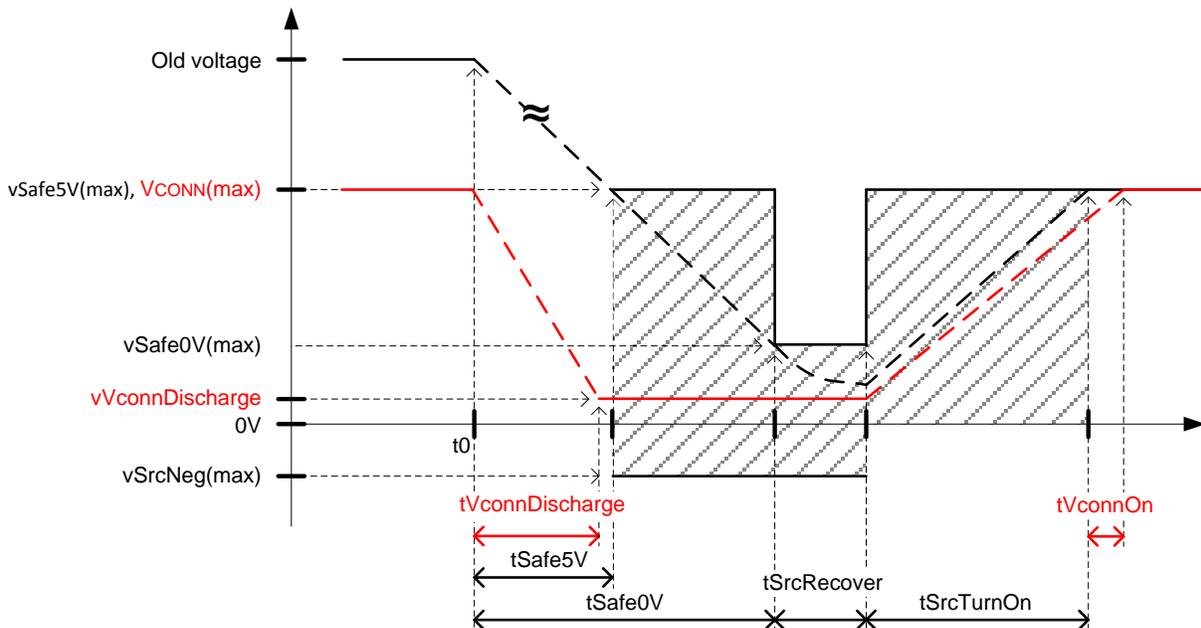
- Self-powered devices **Should Not** disconnect from USB during a Hard Reset (see Section 9.1.2).
- Self-powered devices operating at more than *vSafe5V* **May Not** maintain full functionality after a **Hard Reset**.
- Bus powered devices will disconnect from USB during a Hard Reset due to the loss of their power source.

When a Hard Reset occurs the Source **Shall** stop driving V_{CONN} , **Shall** remove R_p from the V_{CONN} pin and **Shall** start to transition the V_{BUS} voltage to *vSafe0V* either:

- *tPSHardReset* after the last bit of the **Hard Reset** Signaling has been received from the Sink or
- *tPSHardReset* after the last bit of the **Hard Reset** Signaling has been sent by the Source.

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



V_{CONN} will meet $t_{VConnDischarge}$ relative to the start of the voltage transition as shown in Figure 7-4 due to the discharge circuitry in the Cable Plug. V_{CONN} **Shall** meet $t_{VconnOn}$ relative to V_{BUS} reaching v_{Safe5V} . Note $t_{VconnOn}$ and $t_{VConnDischarge}$ are defined in [USB Type-C 1.2].

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 $t_{SrcReady}$.

7.1.8 Robust Source Operation

7.1.8.1 Output Over Current Protection

Sources **Shall** implement output over current protection to prevent damage from output current that exceeds the current handling capability of the Source. The definition of current handling capability is left to the discretion of the Source implementation and **Shall** take into consideration the current handling capability of the connector contacts. The response to over current **Shall Not** interfere with the negotiated V_{BUS} current level.

Sources **Should** attempt to send a **Hard Reset** message when over current protection. The over current protection response **May** engage at either the port or system level. Systems or ports that have engaged over current protection **Should** attempt to resume default operation after determining that the cause of over current is no longer present and **May** latch off to protect the port or system. The definition of how to detect if the cause of over current is still present is left to the discretion of the Source implementation.

The Source **Shall** renegotiate with the Sink (or Sinks) after choosing to resume default operation. The decision of how to renegotiate after an over current event is left to the discretion of the Source implementation.

The Source **Shall** prevent continual system or port cycling if over current protection continues to engage after initially resuming either default operation or renegotiation. Latching off the port or system is an acceptable response to recurring over current.

During the over current response and subsequent system or port shutdown, all affected Source ports operating with V_{BUS} greater than **vSafe5V** **Shall** discharge V_{BUS} to **vSafe5V** by the time **tSafe5V** and **vSafe0V** by the time **tSafe0V**.

7.1.8.2 Over Temperature Protection

Sources **Shall** implement over temperature protection to prevent damage from temperature that exceeds the thermal capability of the Source. The definition of thermal capability and the monitoring locations used to trigger the over temperature protection are left to the discretion of the Source implementation.

Sources **Should** attempt to send a **Hard Reset** message when over temperature protection. The over temperature protection response **May** engage at either the port or system level. Systems or ports that have engaged over temperature protection **Should** attempt to resume default operation and **May** latch off to protect the port or system.

The Source **Shall** renegotiate with the Sink (or Sinks) after choosing to resume default operation. The decision of how to renegotiate after an over temperature event is left to the discretion of the Source implementation.

The Source **Shall** prevent continual system or port cycling if over temperature protection continues to engage after initially resuming either default operation or renegotiation. Latching off the port or system is an acceptable response to recurring over temperature.

During the over temperature response and subsequent system or port shutdown, all affected Source ports operating with V_{BUS} greater than **vSafe5V** **Shall** discharge V_{BUS} to **vSafe5V** by the time **tSafe5V** and **vSafe0V** by the time **tSafe0V**.

7.1.8.3 vSafe5V Externally Applied to Ports Supplying vSafe5V

Safe operation mandates that Power Delivery Sources **Shall** be tolerant of **vSafe5V** being present on V_{BUS} when simultaneously applying power to V_{BUS} . Normal USB PD communication **Shall** be supported when this **vSafe5V** to **vSafe5V** connection exists.

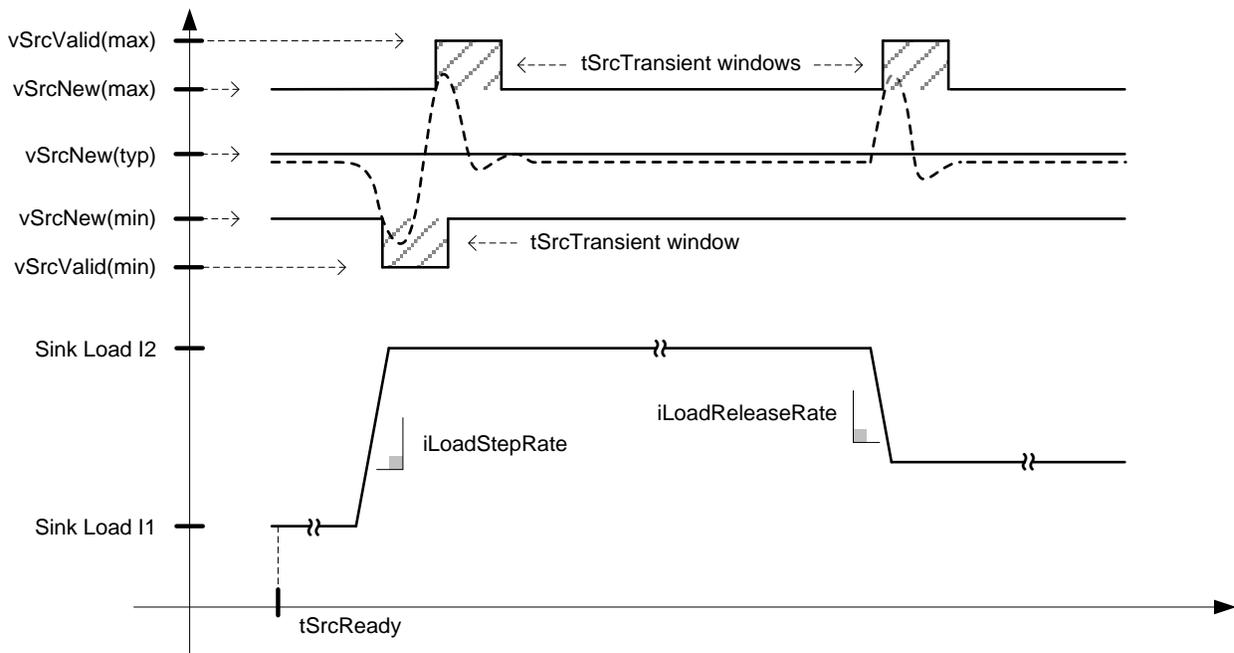
7.1.8.4 Detach

A USB Detach is detected electrically using CC detection on the USB Type-C connector. When the Source is Detached the Source **Shall** transition to **vSafe0V** by **tSafe0V** relative to when the Detach event occurred. During the transition to **vSafe0V** the V_{BUS} voltage **Shall** be below **vSafe5V** max by **tSafe0V** relative to when the Detach event occurred and **Shall Not** exceed **vSafe5V** max after this time.

7.1.9 Output Voltage Tolerance and Range

After a voltage transition is complete (i.e. after **tSrcReady**) and during static load conditions the Source output voltage **Shall** remain within the **vSrcNew** or **vSafe5V** limits as applicable. The ranges defined by **vSrcNew** and **vSafe5V** account for DC regulation accuracy, line regulation, load regulation and output ripple. After a voltage transition is complete (i.e. after **tSrcReady**) and during transient load conditions 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 either **vSrcNew** or **vSafe5V** and **vSrcValid** **Shall Not** exceed **tSrcTransient**. Refer to Table 7-22 for the output voltage tolerance specifications. Figure 7-5 illustrates the application of **vSrcNew** and **vSrcValid** after the voltage transition is complete

Figure 7-5 Application of v_{SrcNew} and $v_{SrcValid}$ limits after $t_{SrcReady}$



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 **Shall** be sufficient to allow for the output voltage to settle between load steps. In some systems it might 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 whether compensation is necessary is left to the discretion of the Source implementation.

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 Power 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:

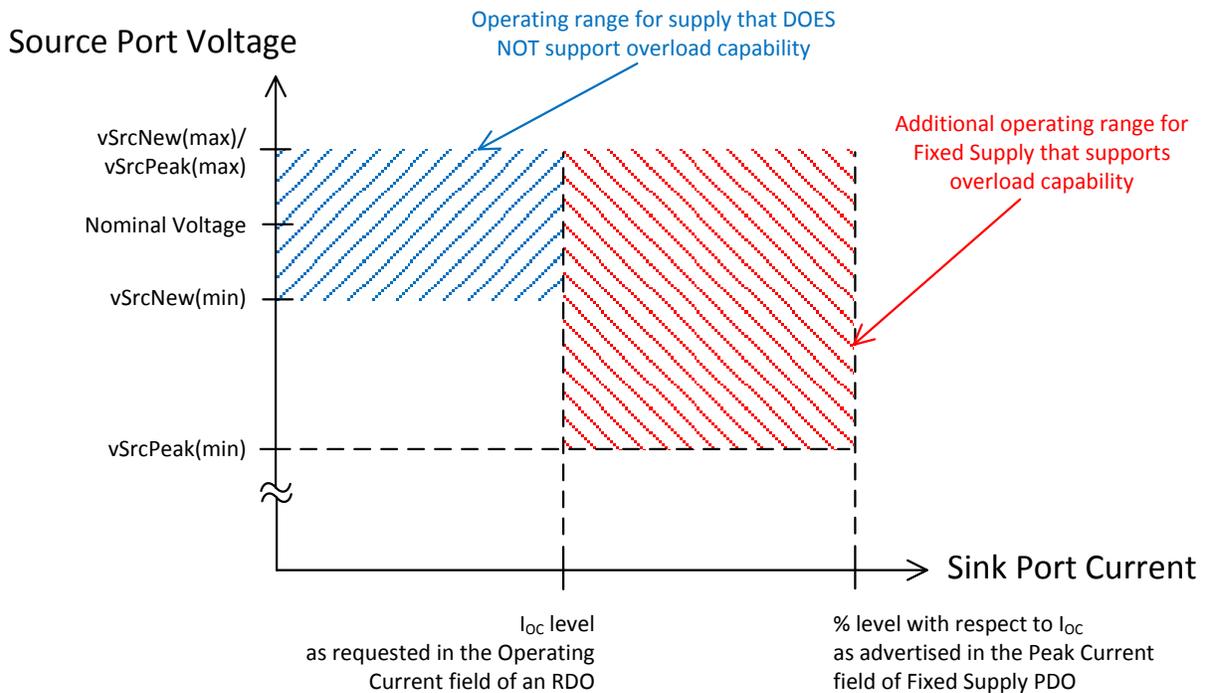
- The Source **Shall Not** drive V_{BUS} that is therefore expected to remain at v_{Safe0V} .
- Any discharge circuitry that was used to achieve v_{Safe0V} **Shall** be removed from V_{BUS} .
- The Dual-Role Power Port **Shall** be configured as a Sink
- The USB connection **Shall Not** reset even though 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_{NewSnk} . 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 $vSrcPeak$ (instead of $vSrcNew$) relative to the nominal value (see Figure 7-6). 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 7-6 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 might 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 might be required to meet the in-band signal-to-noise ratio requirement of $snrSrc$. 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 7-7 and the levels indicated in Table 7-1. Figure 7-7 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 7-7 and Table 7-1.

Figure 7-7s 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 7-7 Noise Spectral Mask, given in absolute terms relative to the maximum value of v_{TX}

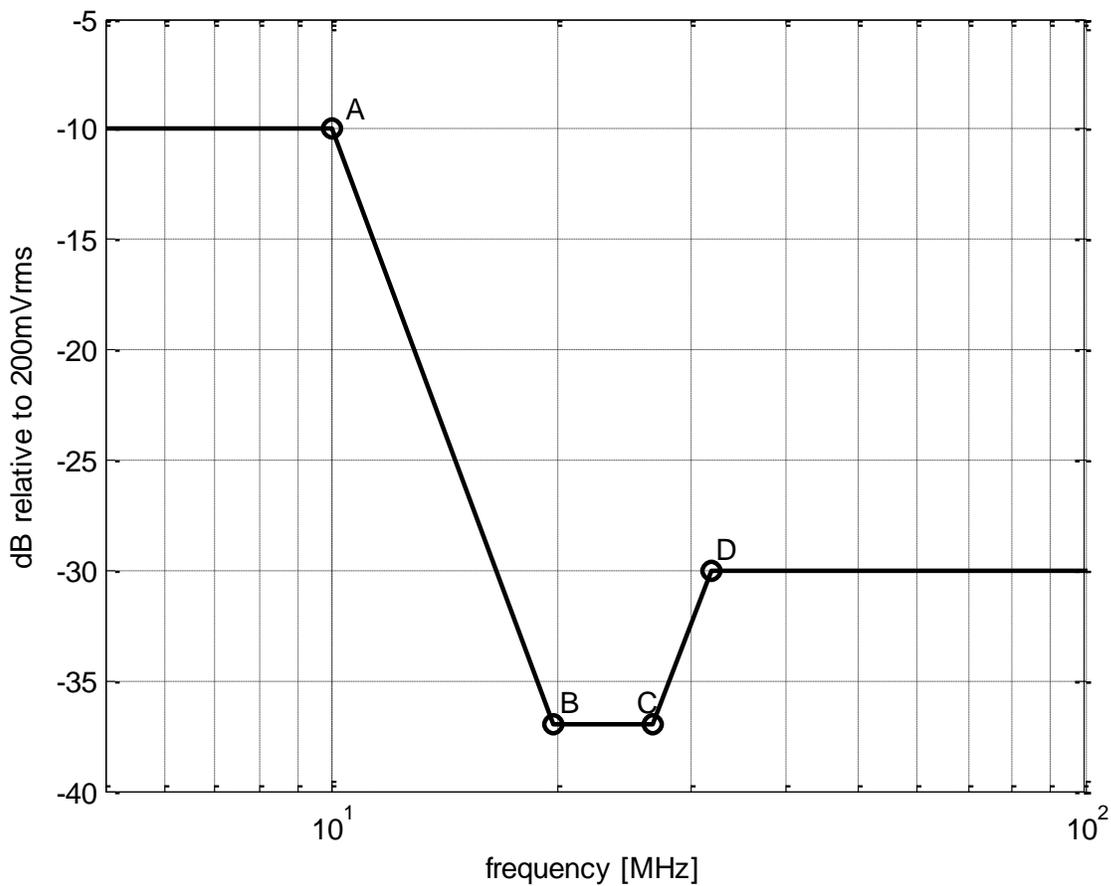


Table 7-1 Noise Spectral Mask Corners

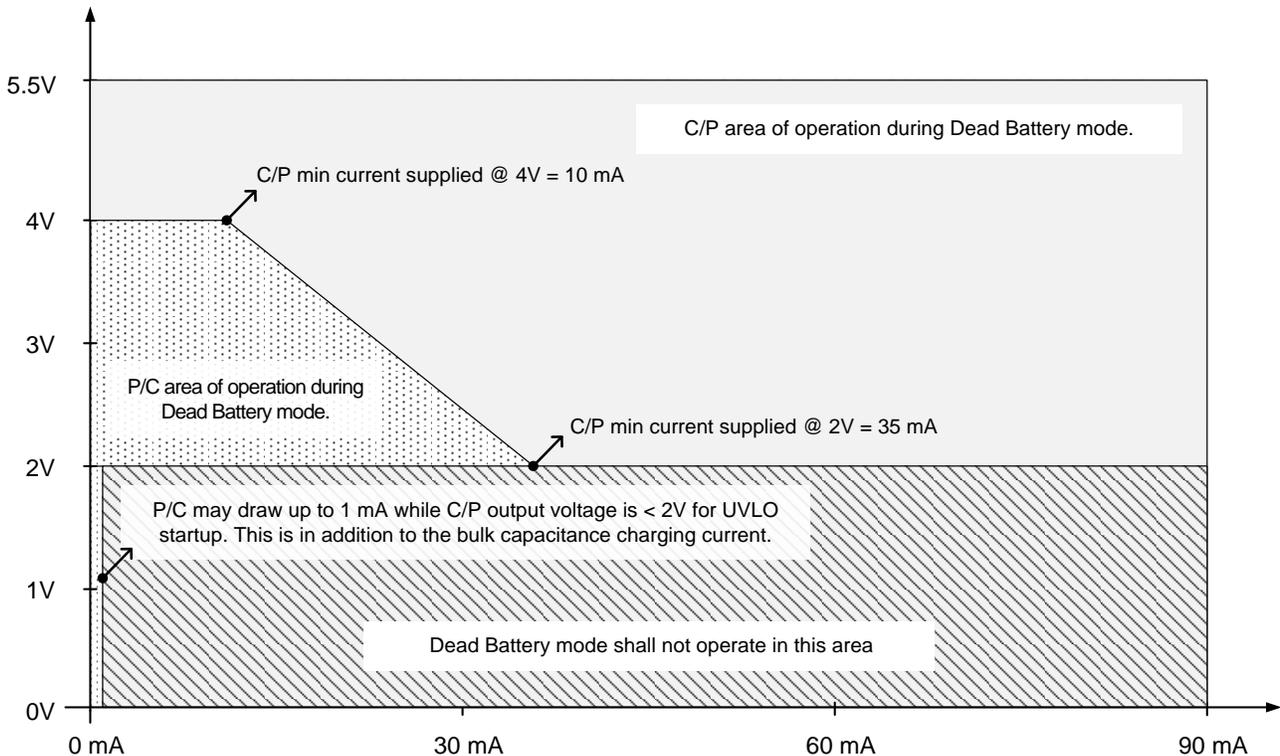
Frequency (MHz)	Maximum Allowed Signal Level	
	(dB)	(mVpp)
<10 (A)	-10	178.7
19.7(B) to 26.7 (C)	-37	7.92
>32 (D)	-30	17.8

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 4.2). The Operation Region of **vSafeDB** source **Shall** be as shown in Figure 7-8. 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 7-8 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 7-8 vSafeDB Operating Region



7.1.14 Application of V_{BUS} Slew Rate Limits

Scenarios where V_{BUS} slew rate limits **Shall Not** apply and V_{BUS} **May** transition faster than specified are as follows:

- When first applying V_{BUS} to a port operating as DFP
- When discharging V_{BUS} to *vSafe0V* during a Hard Reset
- When increasing V_{BUS} from *vSafe0V* to *vSafe5V* during a Hard Reset

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], [USB Type-C 1.2]* or *[USB 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], [USB Type-C 1.2]* or *[USB 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], [USB Type-C 1.2]* or *[USB 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.

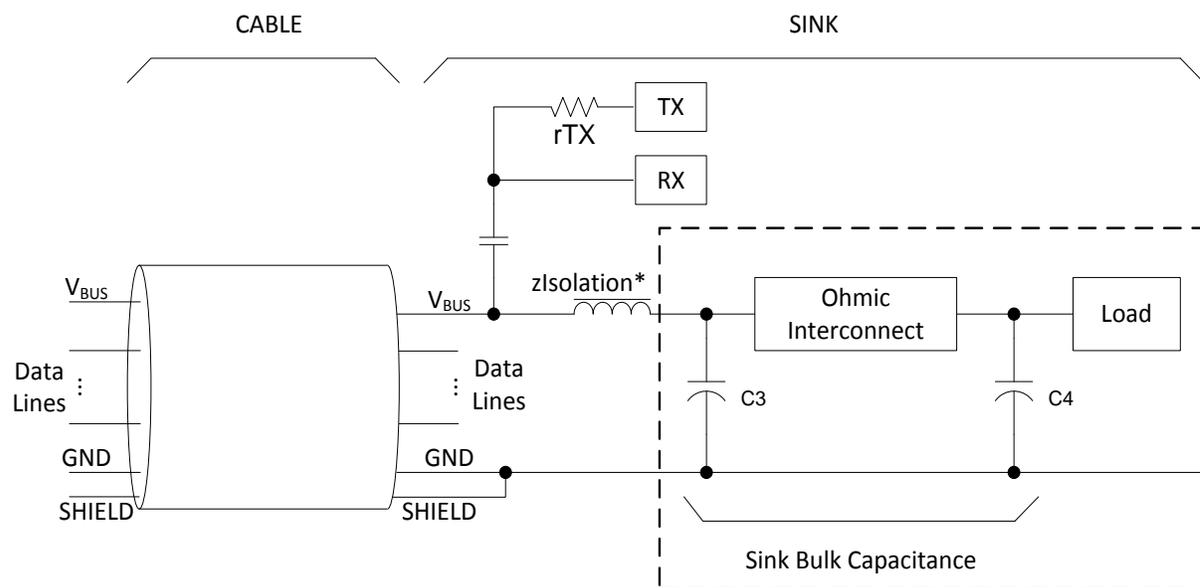
- **May** support Dead Battery operation as defined in Section 4.2.1 for USB Type-A to USB Type-B connections and as defined in *[USB Type-C 1.2]* for USB Type-C to USB Type-C connections.
- **Shall** return to Source operation when responding to a Hard Reset.
- **Shall** control V_{BUS} in-rush current when increasing and decreasing current consumption.

7.2.2 Sink Bulk Capacitance

The Sink **Shall** have a bulk capacitance, *cSnkBulk*, located between the input of the power supply and the transceiver isolation impedance as shown in Figure 7-9. 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 7-9. The Ohmic Interconnect might consist of PCB traces for power distribution or power switching devices. The capacitance might be a single capacitor, a capacitor bank or distributed capacitance. An upper bound of *cSnkBulkPd* **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.

The Sink bulk capacitance that is within the *cSnkBulk* max or *cSnkBulkPd* max 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. Changing the Sink bulk capacitance **Shall Not** cause a transient current on V_{BUS} that violates the present Contract. 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 7-9 Placement of Sink Bulk Capacitance



* zIsolation is only required where BFSK over V_{BUS} 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_{BUS}. During Sink Standby the Sink **Shall** reduce its power draw to *pSnkStdby*. 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.

See Section 7.3 for details of when *pSnkStdby* **Shall** be applied for any given transition.

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_{BUS} and the Dual-Role Port **Shall** be configured as a Source after V_{BUS} 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_{BUS} conductor (see Section 9.1.2). The time for the Sink to transition to Swap Standby **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 *tNewSrc*.

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 might 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 might be required to meet the in-band signal-to-noise ratio requirement of *snrSnk*. 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 7-10 and as detailed in Table 7-2. Figure 7-10 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 *vTX*) – *snrSnk*]. The limits shown in Figure 7-10 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 7-10 Noise Spectral Mask, given in absolute terms relative to the maximum value of vTX

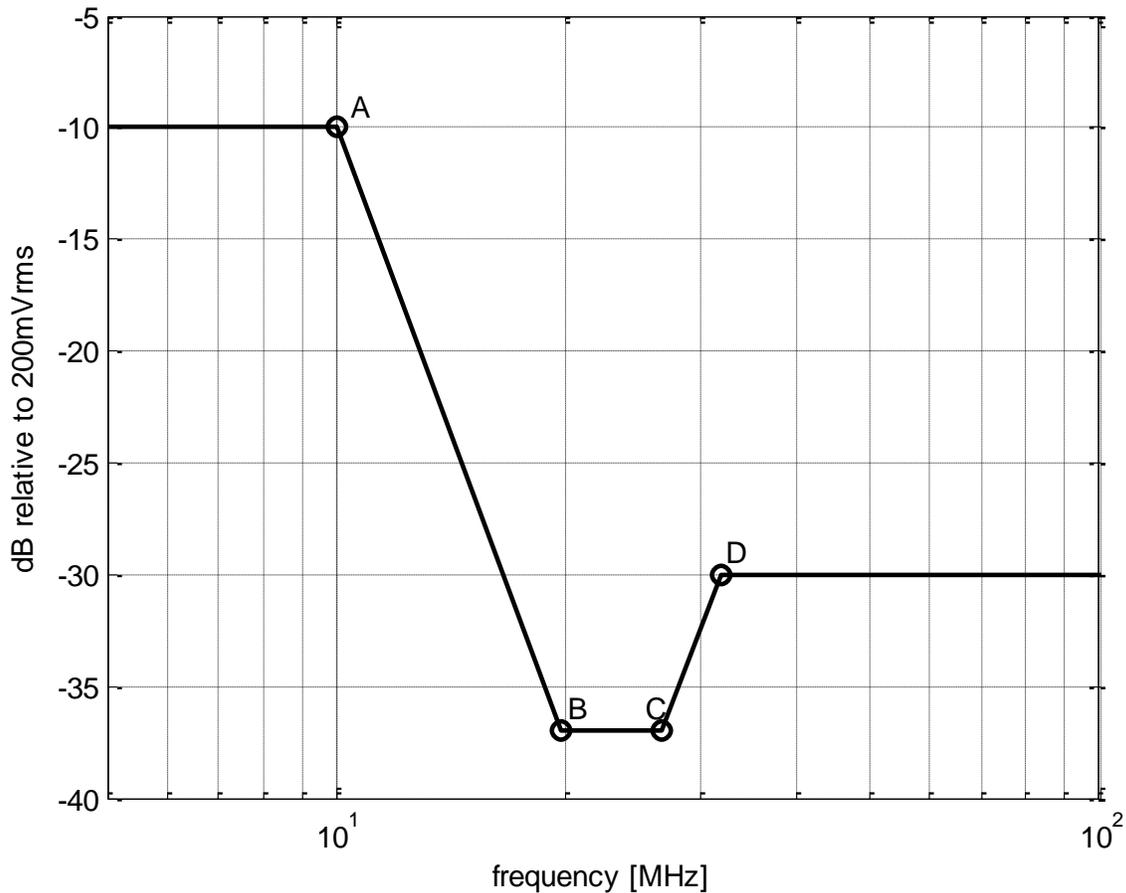


Table 7-2 Noise Spectral Mask Corners

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 7-8. The Provider/Consumer might 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 Section 4.2 and Section 7.1.13.3.

7.2.10 Robust Sink Operation

7.2.10.1 Sink Bulk Capacitance Discharge at Detach

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 **vSafe5V** or lower by no longer than **tSafe5V** from the Detach event. This safe Sink requirement **Shall** apply to all Sinks operating with a negotiated V_{BUS} level greater than **vSafe5V** 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 **Shall** be high enough so that V_{BUS} will decay below **vSrcValid**(min) well within **tSafe5V** 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 μ 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 205ms. At this point, with V_{BUS} well below **vSrcValid** (min) an approximate 100mW 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 **tSafe5V**, 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 **cSnkBulkPd** limit. A Detach occurring during Sink high power operating modes must quickly discharge the Sink bulk capacitance to **vSafe5V** or lower as long as the Sink continues to draw adequate power until V_{BUS} has decayed to **vSafe5V** or lower.

7.2.10.2 Input Over Voltage Protection

Sinks **Shall** implement input over voltage protection to prevent damage from input voltage that exceeds the voltage handling capability of the Sink. The definition of voltage handling capability is left to the discretion of the Sink implementation. The response to over voltage **Shall Not** interfere with the negotiated V_{BUS} voltage level.

Sinks **Should** attempt to send a **Hard Reset** message when over voltage protection engages. The over voltage protection response **May** engage at either the port or system level. Systems or ports that have engaged over voltage protection **Shall** resume default operation when the Source has re-established **vSafe5V** on V_{BUS} .

The Sink **Shall** be able to renegotiate with the Source after resuming default operation. The decision of how to respond to renegotiation after an over voltage event is left to the discretion of the Sink implementation.

The Sink **Shall** prevent continual system or port cycling if over voltage protection continues to engage after initially resuming either default operation or renegotiation. Latching off the port or system is an acceptable response to recurring over voltage.

7.2.10.3 Over Temperature Protection

Sinks **Shall** implement over temperature protection to prevent damage from temperature that exceeds the thermal capability of the Sink. The definition of thermal capability and the monitoring locations used to trigger the over temperature protection are left to the discretion of the Sink implementation.

Sinks **Shall** attempt to send a **Hard Reset** message when over temperature protection engages. The over temperature protection response **May** engage at either the port or system level. Systems or ports that have engaged over temperature protection **Should** attempt to resume default operation after sufficient cooling is achieved and **May** latch off to protect the port or system. The definition of sufficient cooling is left to the discretion of the Sink implementation.

The Sink **Shall** be able to renegotiate with the Source after resuming default operation. The decision of how to respond to renegotiation after an over temperature event is left to the discretion of the Sink implementation.

The Sink **Shall** prevent continual system or port cycling if over temperature protection continues to engage after initially resuming either default operation or renegotiation. Latching off the port or system is an acceptable response to recurring over temperature.

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], [USB Type-C 1.2]** or **[USB 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], [USB Type-C 1.2]** or **[USB 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 7-11. 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 7-11 Transition Diagram for Increasing the Current

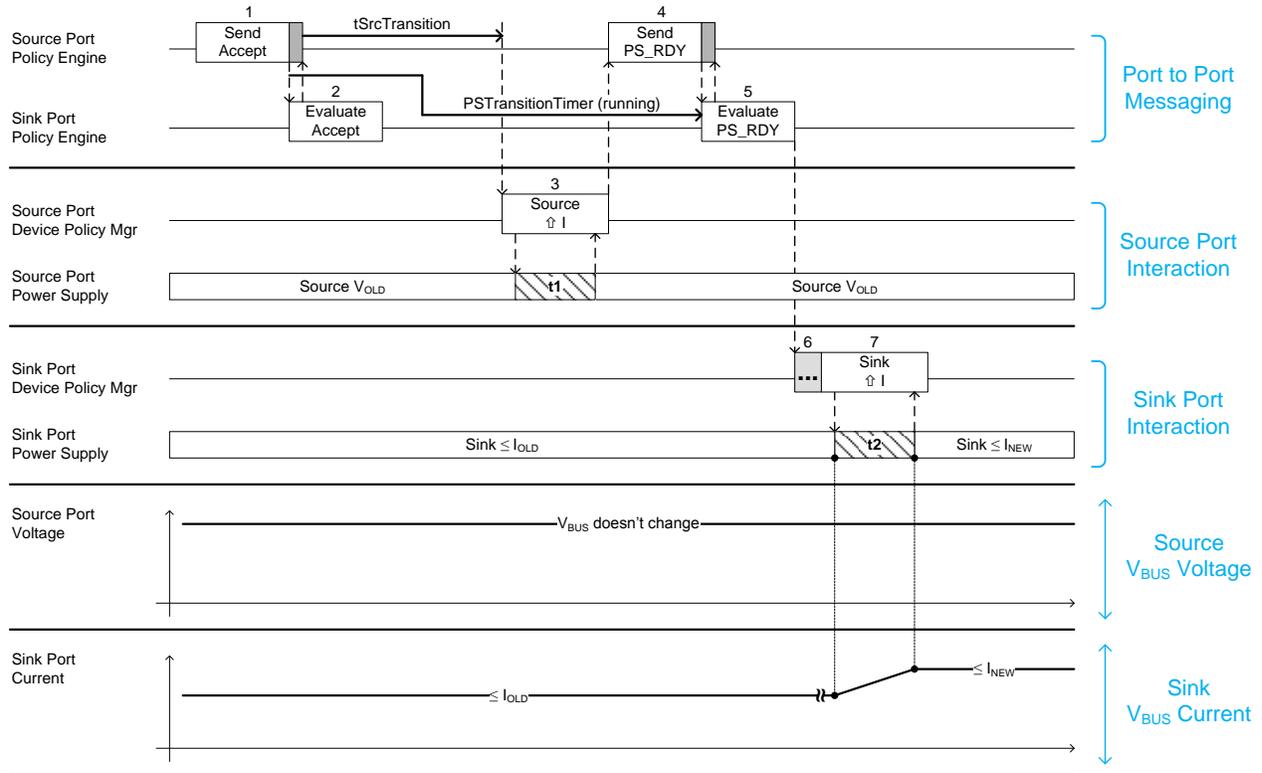


Table 7-3 Sequence Description for Increasing the Current

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 tells 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	<i>tSrcTransition</i> after the <i>GoodCRC</i> Message has been received the power supply starts 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 7-12. 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 7-12 Transition Diagram for Increasing the Voltage

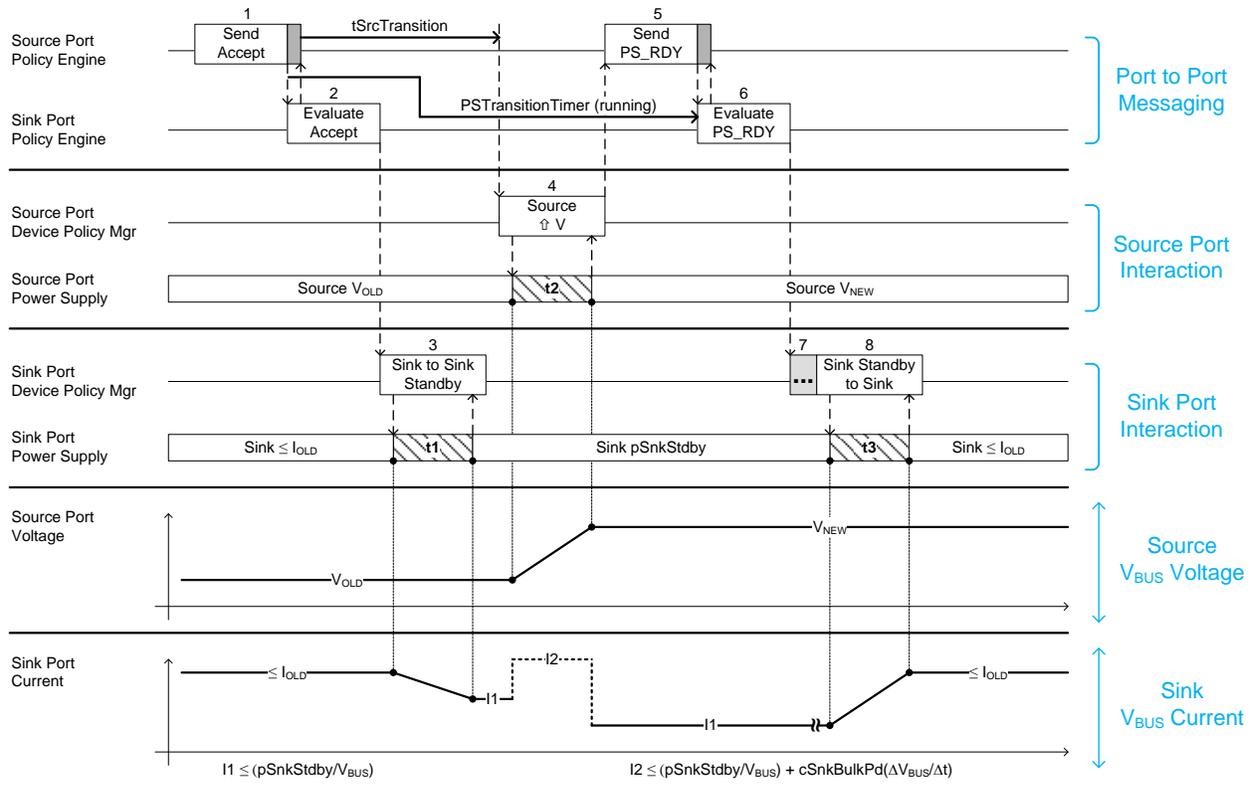


Table 7-4 Sequence Description for Increasing the Voltage

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 tells 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> (t1); t1 <i>Shall</i> complete before <i>tSrcTransition</i> . The Sink <i>Shall Not</i> violate transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.
4	<i>tSrcTransition</i> after the <i>GoodCRC</i> Message has been received the power supply starts to change its output power capability. The power supply <i>Shall</i> 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 <i>May</i> 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 <i>Shall Not</i> 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 7-13. 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 7-13 Transition Diagram for Increasing the Voltage and Current

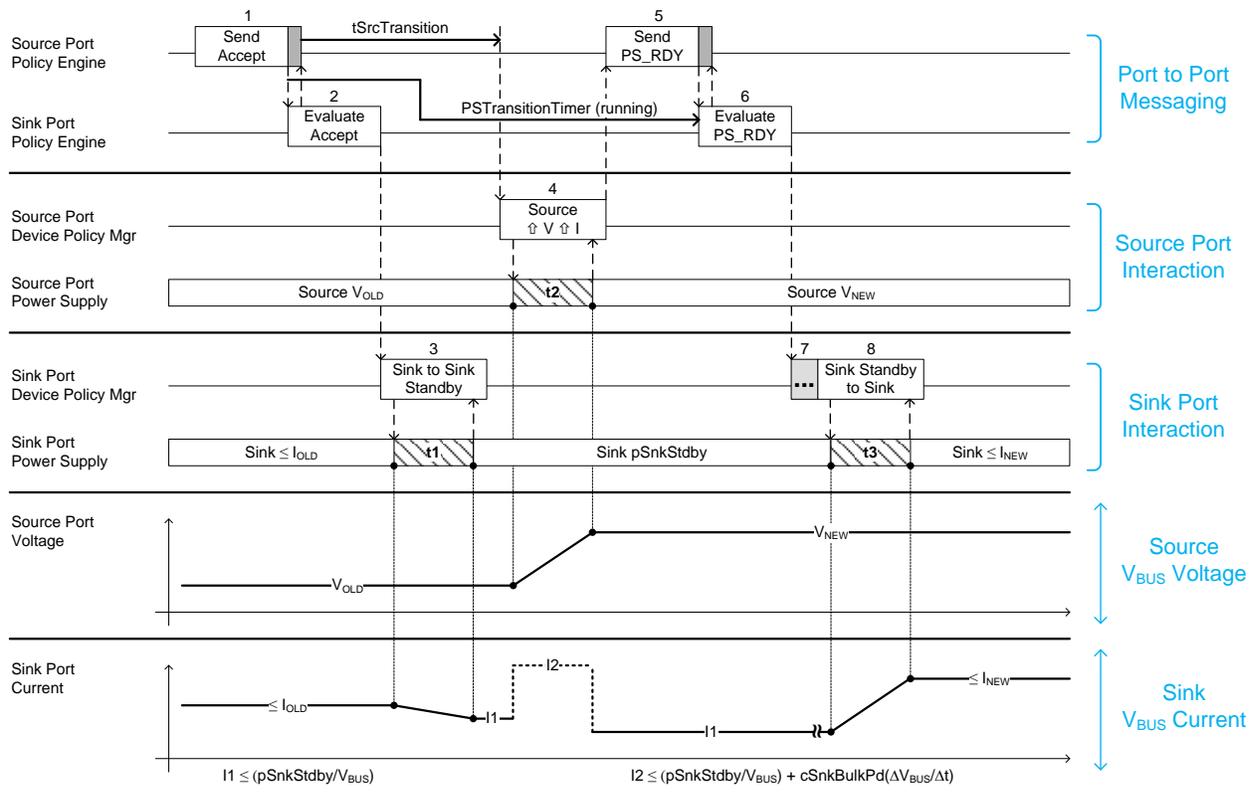


Table 7-5 Sequence Diagram for Increasing the Voltage and Current

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 tells 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> (t1) ; t1 <i>Shall</i> complete before <i>tSrcTransition</i> . The Sink <i>Shall Not</i> violate transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.
4	<i>tSrcTransition</i> after the <i>GoodCRC</i> Message has been received the power supply starts to change its output power capability. The power supply <i>Shall</i> 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 <i>May</i> 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 <i>Shall Not</i> 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.

7.3.4 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 7-14. 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 7-14 Transition Diagram for Increasing the Voltage and Decreasing the Current

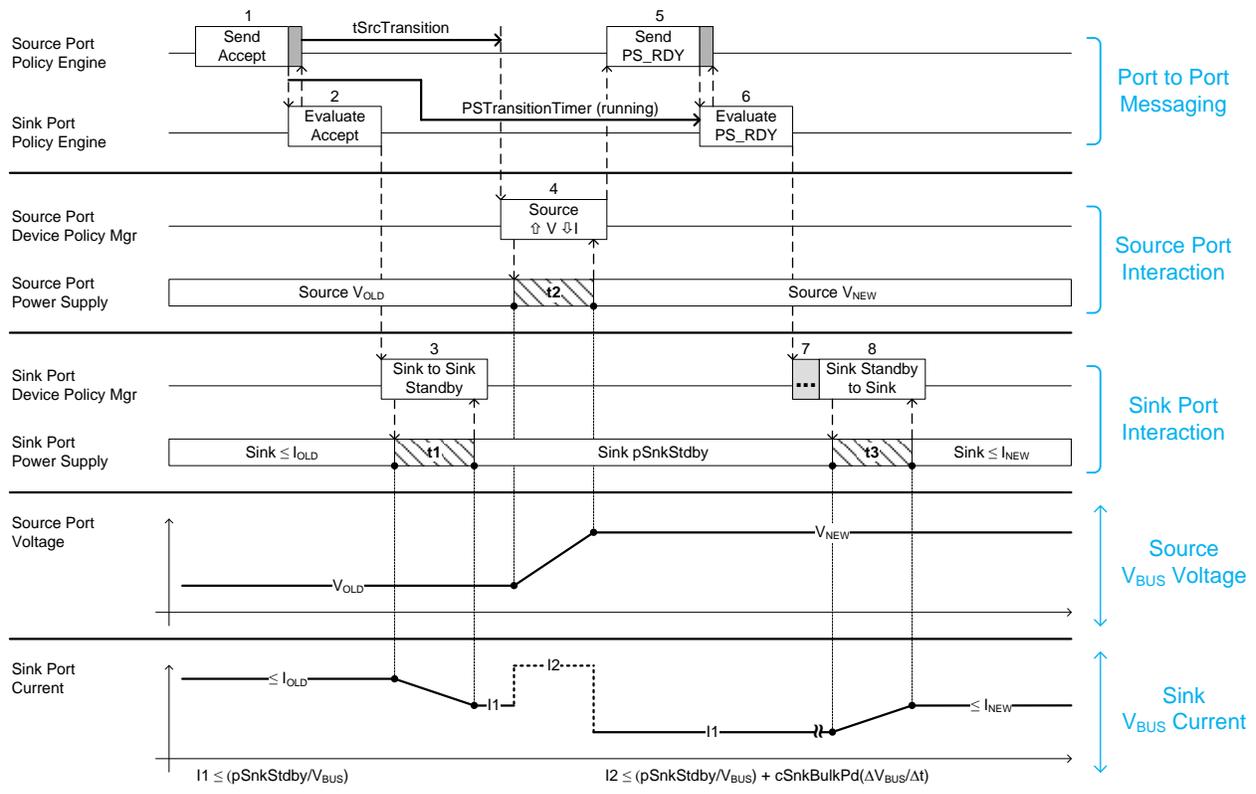


Table 7-6 Sequence Description for Increasing the Voltage and Decreasing the Current

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 tells 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> (t1) ; t1 <i>shall</i> complete before <i>tSrcTransition</i> . The Sink <i>shall Not</i> violate transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.
4	<i>tSrcTransition</i> after the <i>GoodCRC</i> Message has been received the power supply starts to change its output power capability. The power supply <i>shall</i> 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 <i>may</i> 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 <i>shall Not</i> 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.

7.3.5 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 7-15. 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 7-15 Transition Diagram for Decreasing the Voltage and Increasing the Current

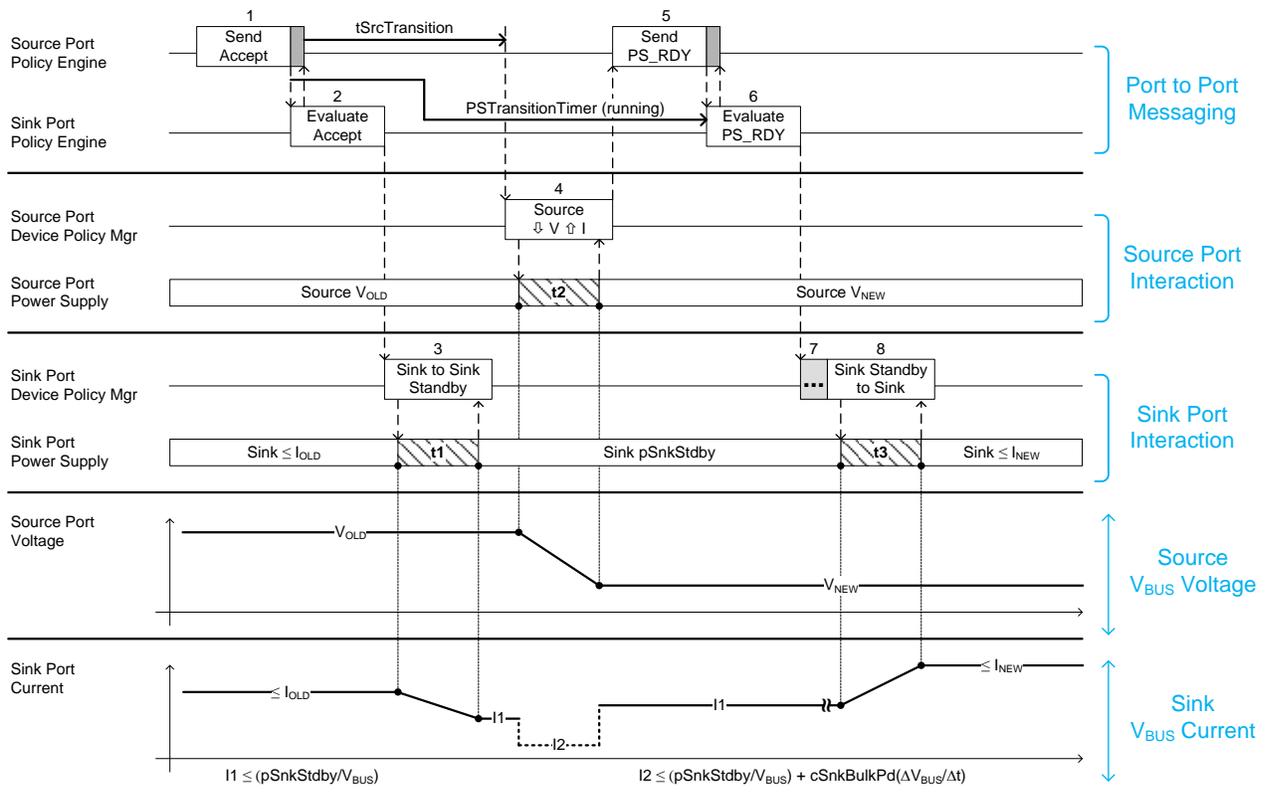


Table 7-7 Sequence Description for Decreasing the Voltage and Increasing the Current

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 tells 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> (t1) ; t1 <i>shall</i> complete before <i>tSrcTransition</i> . The Sink <i>shall Not</i> violate transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.
4	<i>tSrcTransition</i> after the <i>GoodCRC</i> Message has been received the power supply starts to change its output power capability. The power supply <i>shall</i> 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 <i>may</i> 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 <i>shall Not</i> 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.

7.3.6 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 7-16. 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 7-16 Transition Diagram for Decreasing the Current

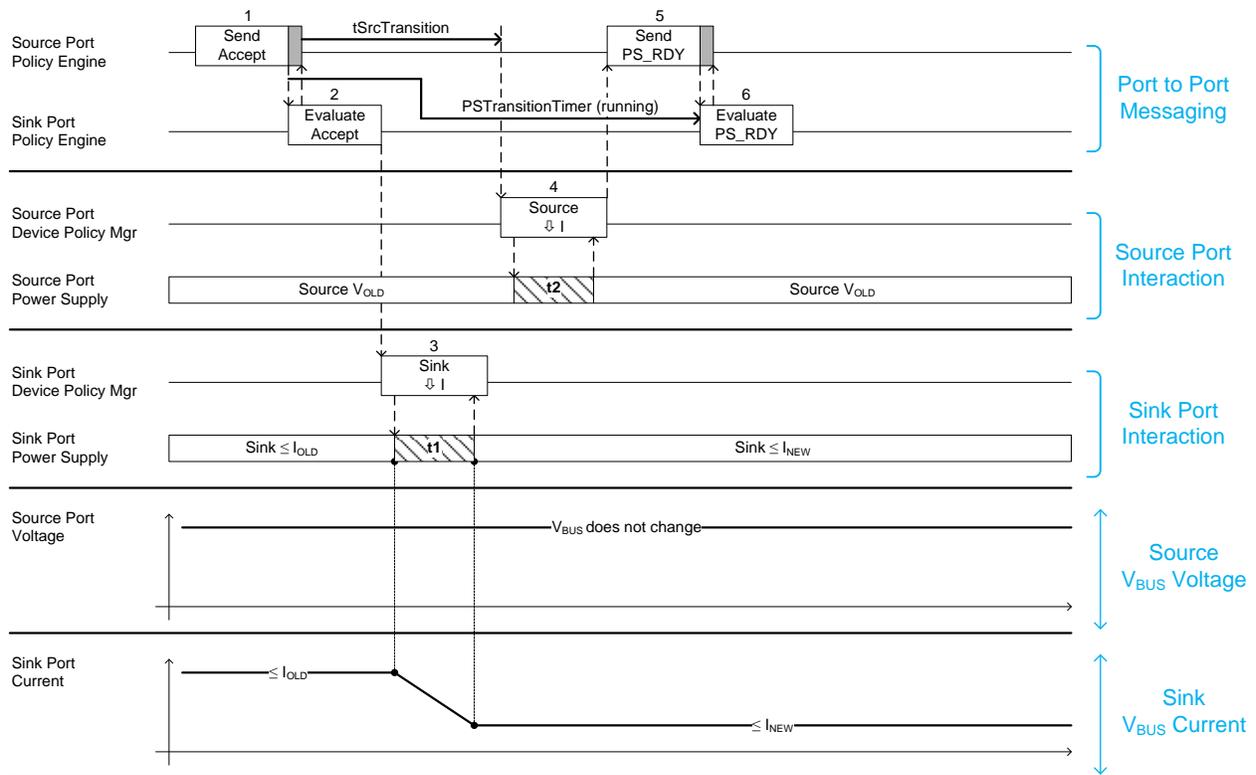


Table 7-8 Sequence Description for Decreasing the Current

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 tells 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	<i>tSrcTransition</i> after the <i>GoodCRC</i> Message has been received the power supply starts 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 7-17. 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 7-17 Transition Diagram for Decreasing the Voltage

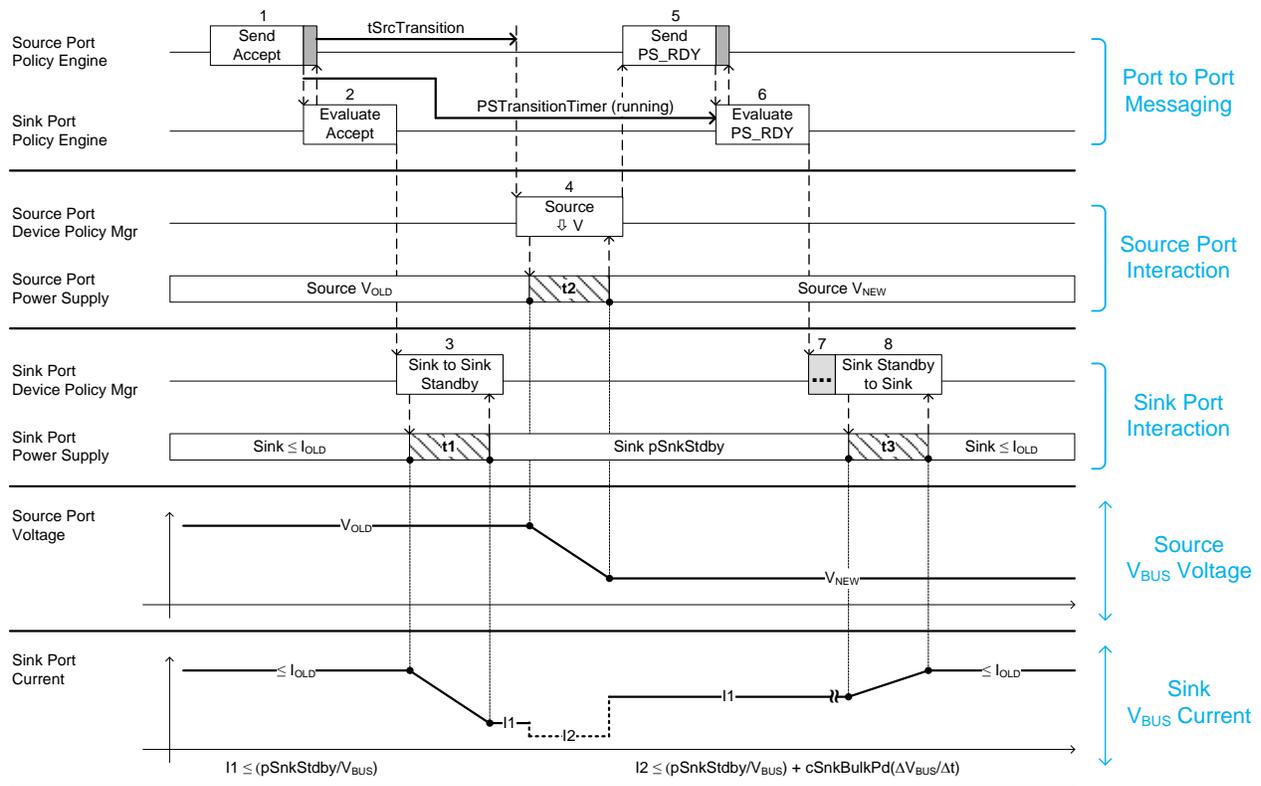


Table 7-9 Sequence Description for Decreasing the Voltage

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 tells 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> (t1) ; t1 <i>shall</i> complete before <i>tSrcTransition</i> . The Sink <i>shall Not</i> violate transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.
4	<i>tSrcTransition</i> after the <i>GoodCRC</i> Message has been received the power supply starts to change its output power capability. The power supply <i>shall</i> 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 <i>may</i> 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 <i>shall Not</i> 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.

7.3.8 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 7-18. 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 7-18 Transition Diagram for Decreasing the Voltage and the Current

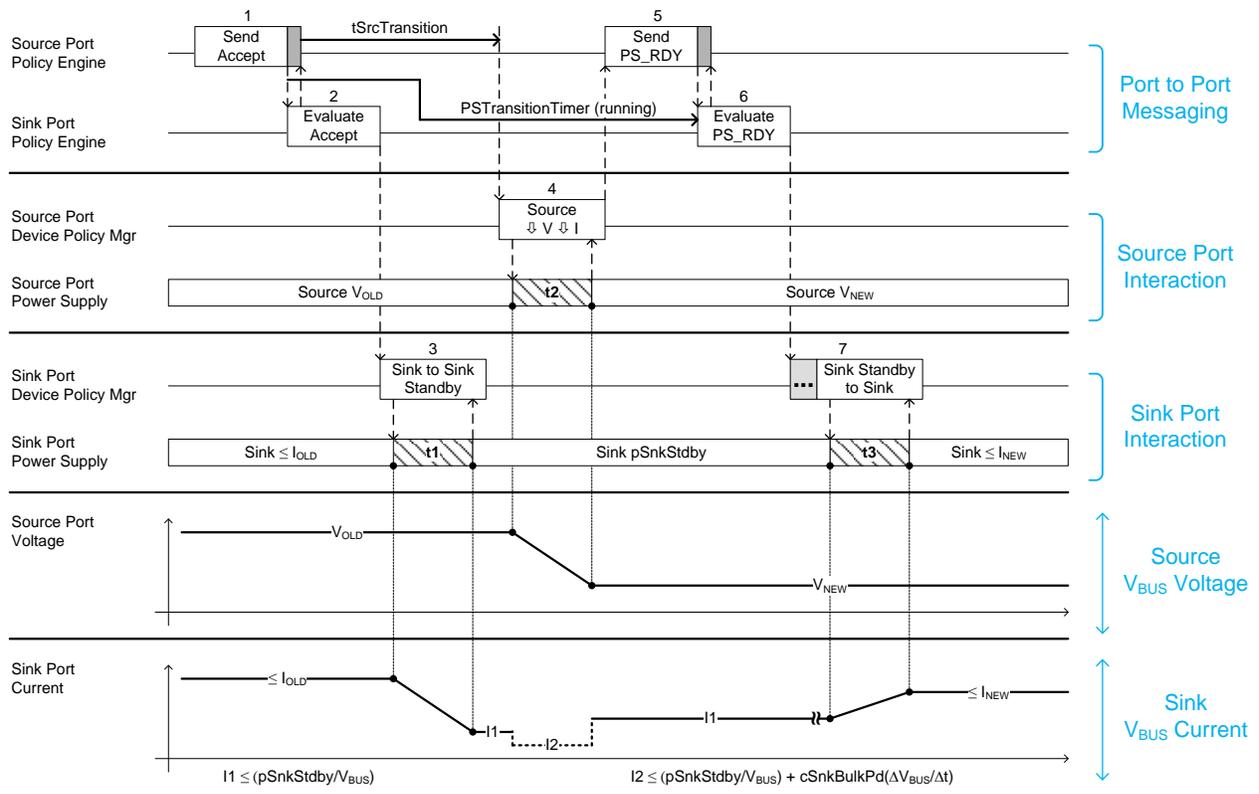


Table 7-10 Sequence Description for Decreasing the Voltage and the Current

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 tells 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> (t1) ; t1 <i>shall</i> complete before <i>tSrcTransition</i> . The Sink <i>shall Not</i> violate transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level.
4	<i>tSrcTransition</i> after the <i>GoodCRC</i> Message has been received the power supply starts to change its output power capability. The power supply <i>shall</i> 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 <i>may</i> 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 <i>shall Not</i> 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.

7.3.9 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 7-19. 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 7-19 Transition Diagram for a Sink Requested Power Role Swap

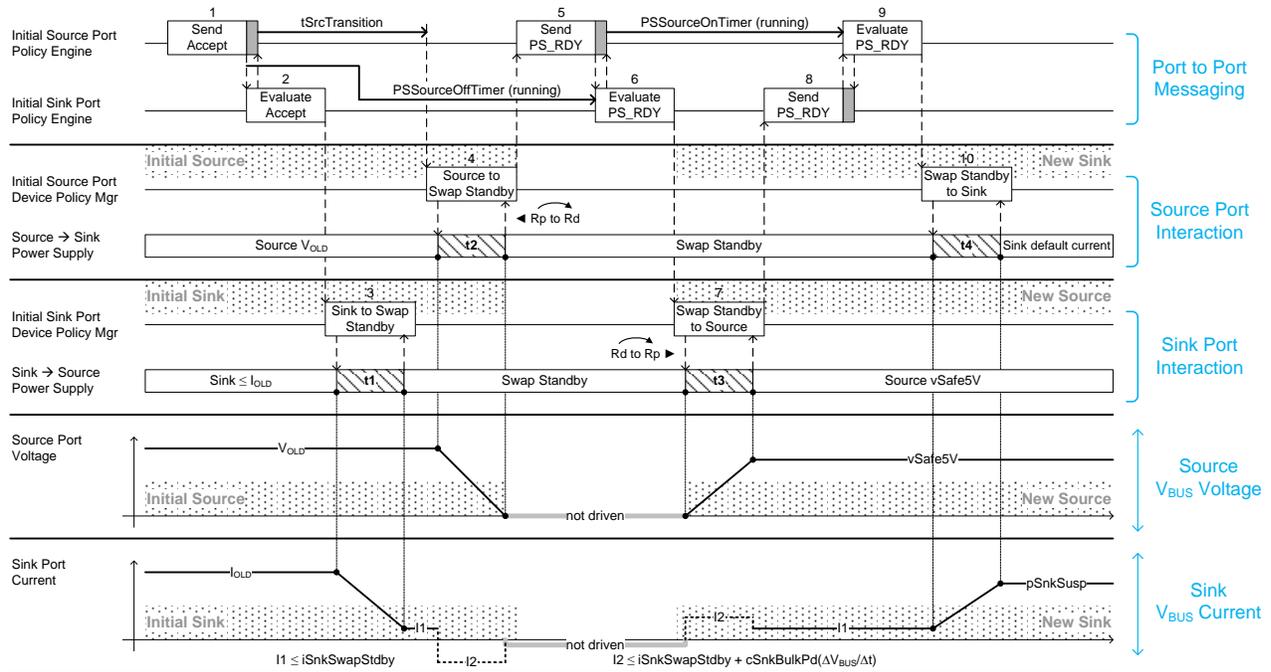


Table 7-11 Sequence Description for a Sink Requested Power Role Swap

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 tells 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>tSnkStdby</i> (t1) ; t1 Shall complete before <i>tSrcTransition</i> . When in Sink Standby the Initial Sink Shall Not draw more than <i>iSnkSwapStdby</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	<i>tSrcTransition</i> after the <i>GoodCRC</i> Message has been received the power supply starts to change its output power capability to Swap Standby (see Section 7.1.11). The power supply Shall complete the transition to Swap Standby within <i>tSrcSwapStdby</i> (t2). The power supply informs the Device Policy Manager that it is ready to operate as the new Sink. The CC termination is changed from Rp to Rd (see [USB Type-C 1.2]). 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 CC termination is changed from Rd to Rp (see [USB Type-C 1.2]). 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> (t4). 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 (t4) 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 7-20. 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 7-20 Transition Diagram for a Source Requested Power Role Swap

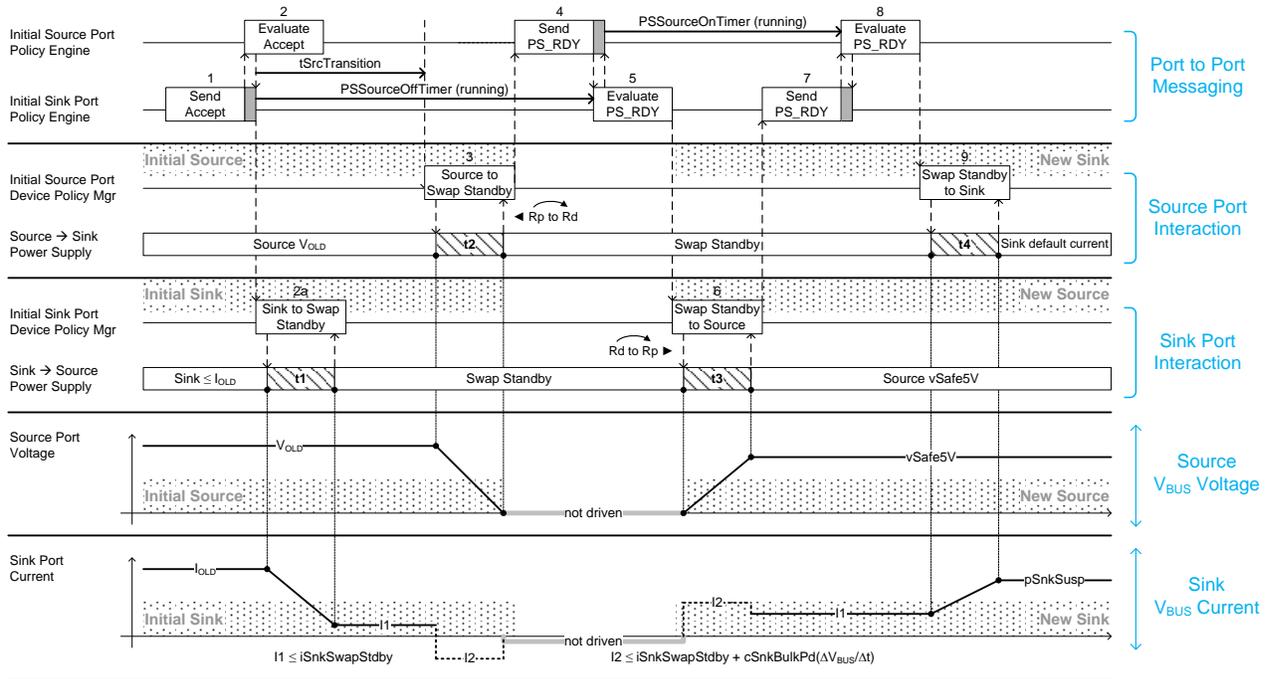


Table 7-12 Sequence Description for a Source Requested Power Role Swap

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 receives the <i>GoodCRC</i> Message from the Sink. The Policy Engine tells the Device Policy Manager to instruct the power supply to modify its output power.	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 <i>Shall</i> complete the transition to Swap Standby within <i>tSnkStdby</i> (t1) ; t1 <i>Shall</i> complete before <i>tSrcTransition</i> . The Sink <i>Shall Not</i> 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 <i>Shall Not</i> draw more than <i>iSnkSwapStdby</i> (I1).
3	<i>tSrcTransition</i> after the <i>GoodCRC</i> Message has been received the power supply starts to change its output power capability to Swap Standby (see Section 7.1.11). The power supply <i>Shall</i> complete the transition to Swap Standby within <i>tSrcSwapStdby</i> (t2). The power supply informs the Device Policy Manager that it is ready to operate as the new Sink. The CC termination is changed from Rp to Rd (see <i>[USB Type-C 1.2]</i>). 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 CC termination is changed from Rd to Rp (see <i>[USB Type-C 1.2]</i>). 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> (t4). 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 (t4) 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 7-21. 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 7-21 Transition Diagram for a GotoMin Current Decrease

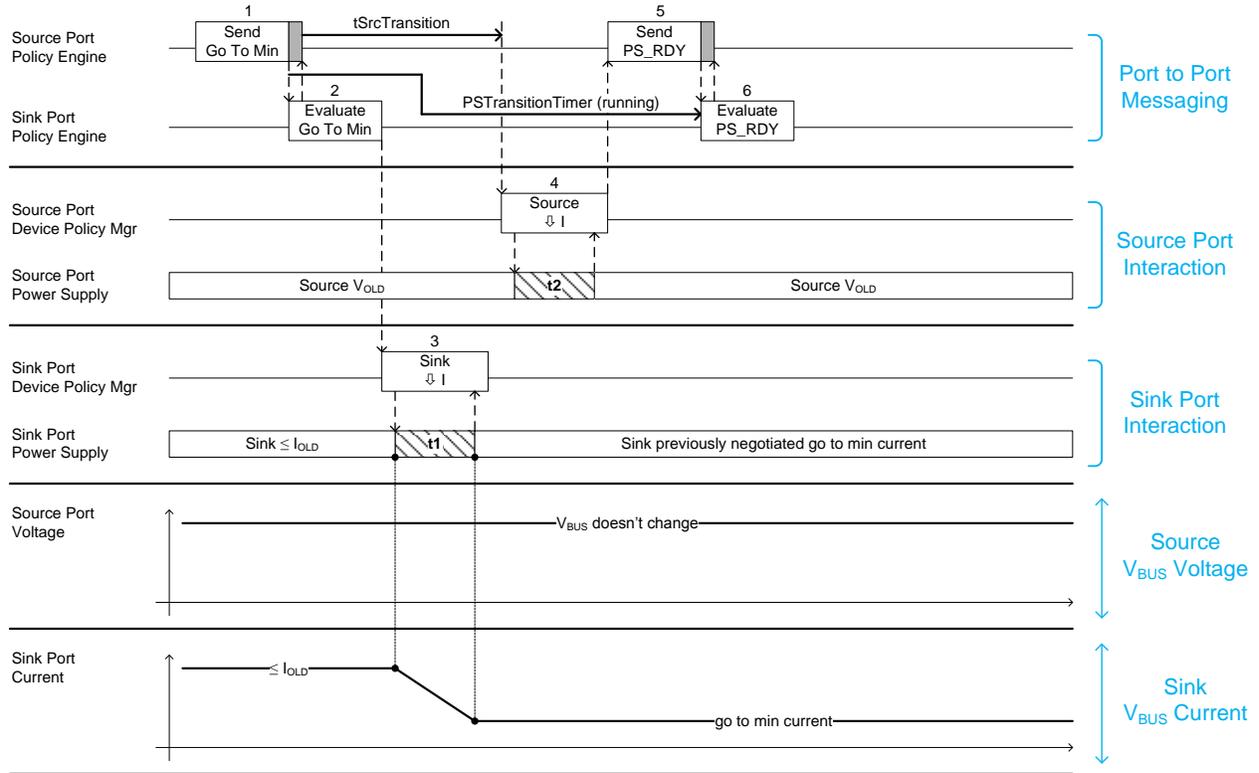


Table 7-13 Sequence Description for a GotoMin Current Decrease

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 tells 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	<i>tSrcTransition</i> after <i>GoodCRC Message has been received</i> the power supply starts 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 7-22. 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 7-22 Transition Diagram for a Source Initiated Hard Reset

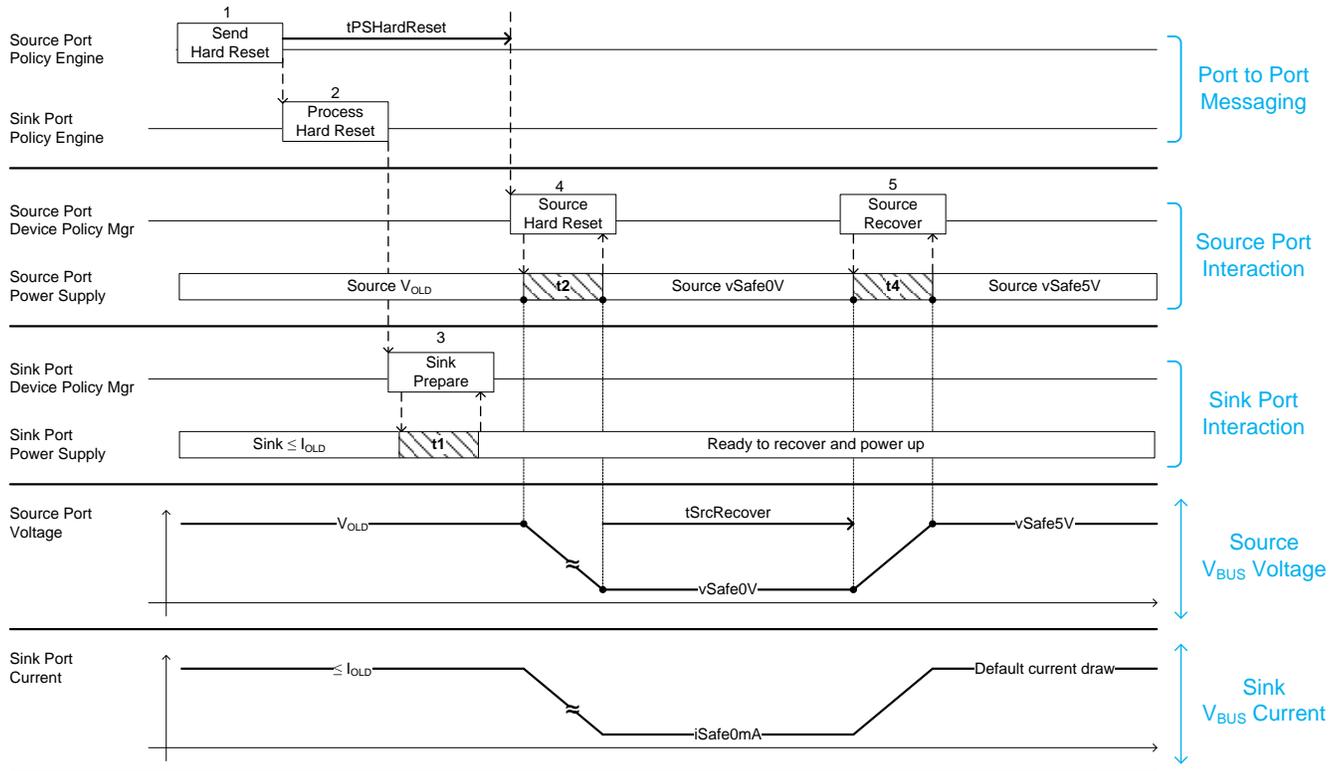


Table 7-14 Sequence Description for a Source Initiated Hard Reset

Step	Source Port	Sink Port
1	Policy Engine sends <i>Hard Reset</i> Signaling to the Sink.	Sink receives <i>Hard Reset</i> Signaling.
2		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.
3		The Sink prepares for the Hard Reset within <i>tSnkHardResetPrepare</i> (t1)) and passes an indication to the Device Policy Manger The Sink Shall Not draw more than <i>iSafe0mA</i> when V_{BUS} is driven to <i>vSafe0V</i> .
4	Policy Engine waits <i>tPSHardReset</i> 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 <i>vSafe0V</i> Shall occur within <i>tSafe0V</i> (t2).	
5	After <i>tSrcRecover</i> 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 <i>vSafe5V</i> Shall occur within <i>tSrcTurnOn</i> (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.

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 7-23. 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 7-23 Transition Diagram for a Sink Initiated Hard Reset

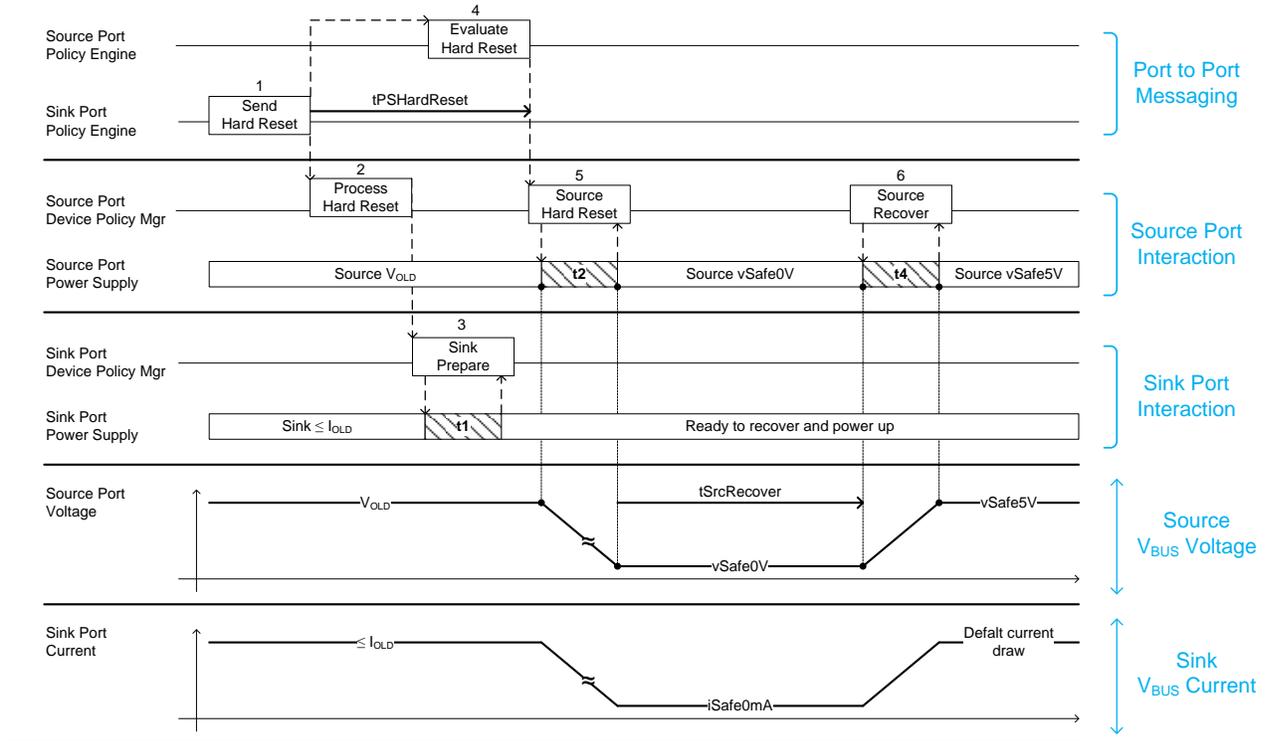


Table 7-15 Sequence Description for a Sink Initiated Hard Reset

Step	Source Port	Sink Port
1		Policy Engine sends Hard Reset Signaling to the Source.
2		Policy Engine tells the Device Policy Manager to instruct the power supply to prepare for a Hard Reset.
3		The Sink prepares for the Hard Reset within <i>tSnkHardResetPrepare</i> (t1) and passes an indication to the Device Policy Manger The Sink Shall Not draw more than <i>iSafe0mA</i> when V_{BUS} is driven to <i>vSafe0V</i> .
4	Policy Engine is informed of the Hard Reset.	
5	Policy Engine waits <i>tPSHardReset</i> after receiving Hard Reset Signaling and then tells the Device Policy Manager to instruct the power supply to perform a Hard Reset. The transition to <i>vSafe0V</i> Shall occur within <i>tSafe0V</i> (t2).	
6	After <i>tSrcRecover</i> 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 <i>vSafe5V</i> Shall occur within <i>tSrcTurnOn</i> (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.

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 USB Type-A Port acting as Sink and a USB Type-B Port acting as Source.

7.3.14.1 Type-B New Source Initiates Hard Reset and USB 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 USB Type-B new Source when the USB Type-A new Sink receives **Hard Reset** Signaling is shown in Figure 7-24. 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.

Figure 7-24 Transition Diagram for USB Type-B New Source Initiated Hard Reset and USB Type-A New Sink Receives Hard Reset Signaling

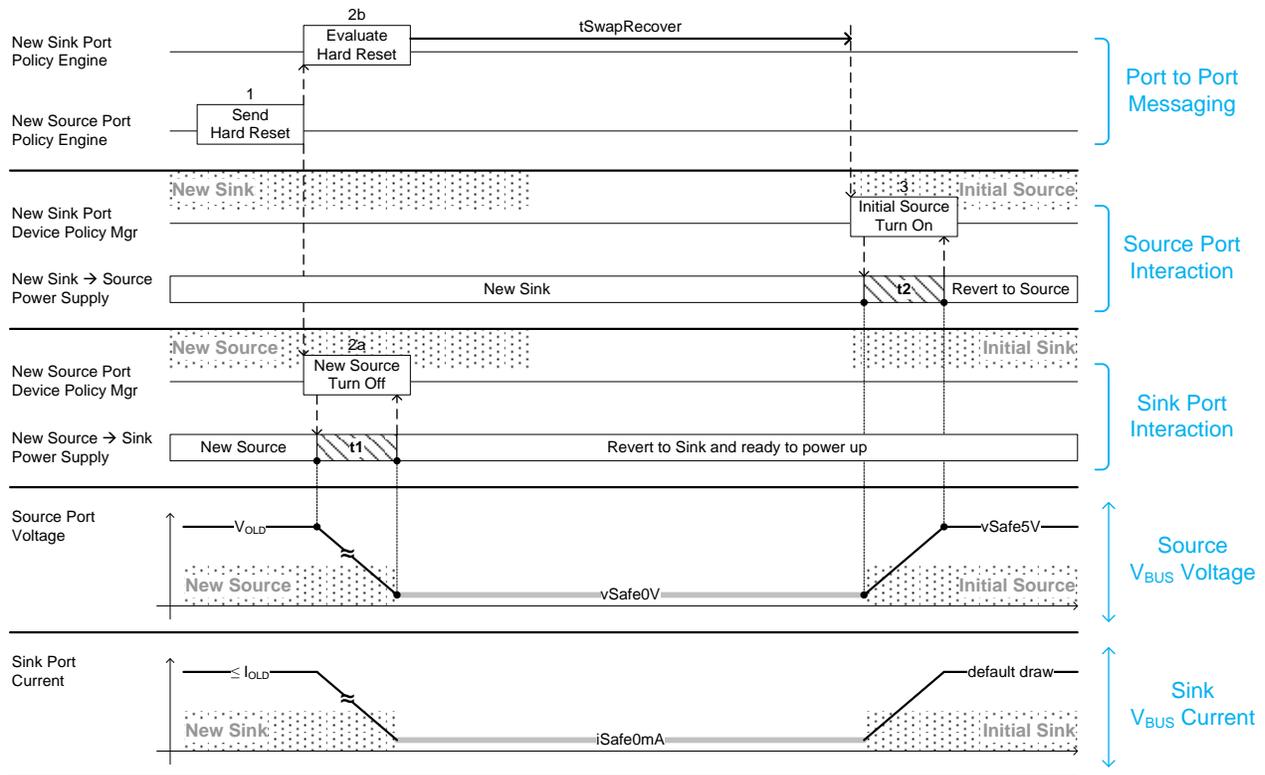


Table 7-16 Sequence Description for USB Type-B New Source Initiated Hard Reset and USB Type-A New Sink Receives Hard Reset Signaling

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 <i>[USB 2.0]</i> , <i>[USB 3.1]</i> , <i>[USB Type-C 1.2]</i> or <i>[USB BC 1.2]</i> .	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.

7.3.14.2 Type-B New Source Initiates Hard Reset and USB 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 USB Type-B new Source when the USB Type-A new Sink does not receive the **Hard Reset** Signaling is shown in Figure 7-25. 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.

Figure 7-25 Transition Diagram for USB Type-B New Source Initiated Hard Reset and USB Type-A New Sink Does Not Receive Hard Reset Signaling

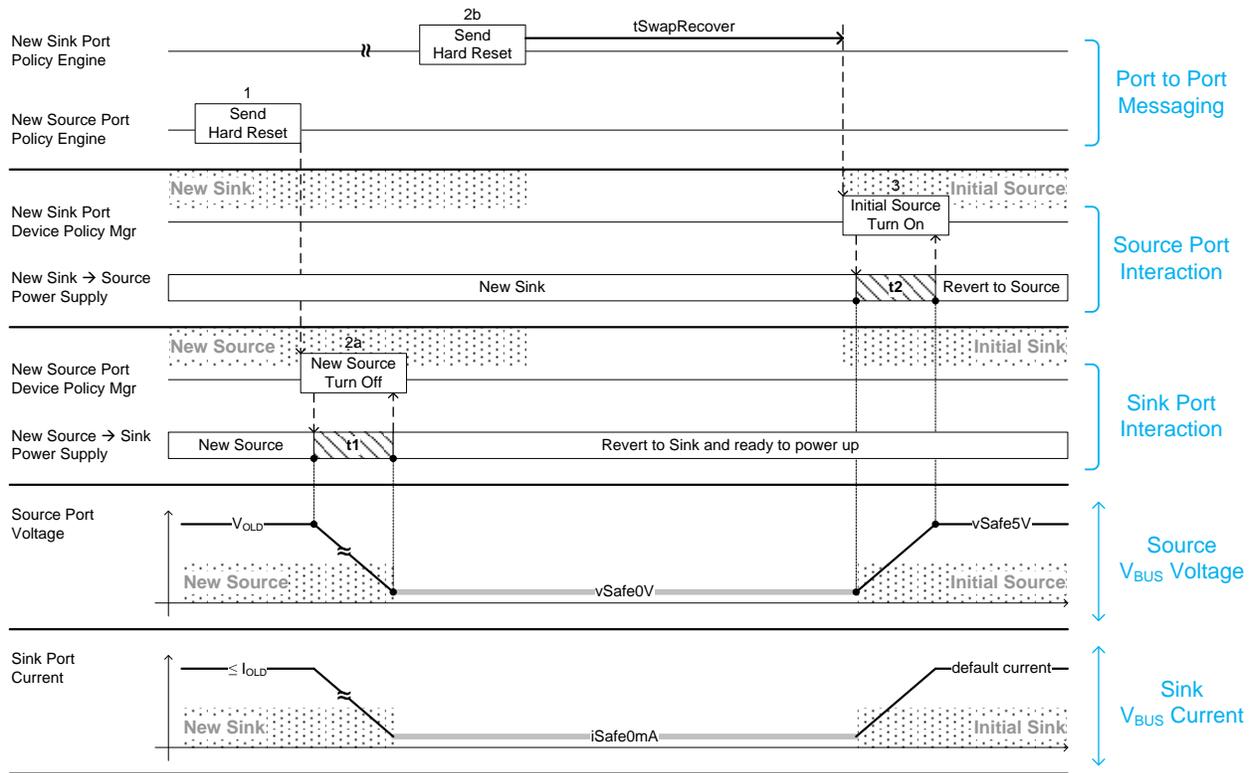


Table 7-17 Sequence Description for USB Type-B New Source Initiated Hard Reset and USB Type-A New Sink Does Not Receive Hard Reset Signaling

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>[USB Type-C 1.2]</i> or <i>[USB 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.

7.3.14.3 Type-A New Sink Initiates Hard Reset and USB 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 USB Type-A new Sink when the USB Type-B new Source receives the **Hard Reset** Signaling is shown in Figure 7-26. 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 6.7.2.

Figure 7-26 Transition Diagram for USB Type-A New Sink Initiated Hard Reset and USB Type-B New Source Receives Hard Reset Signaling

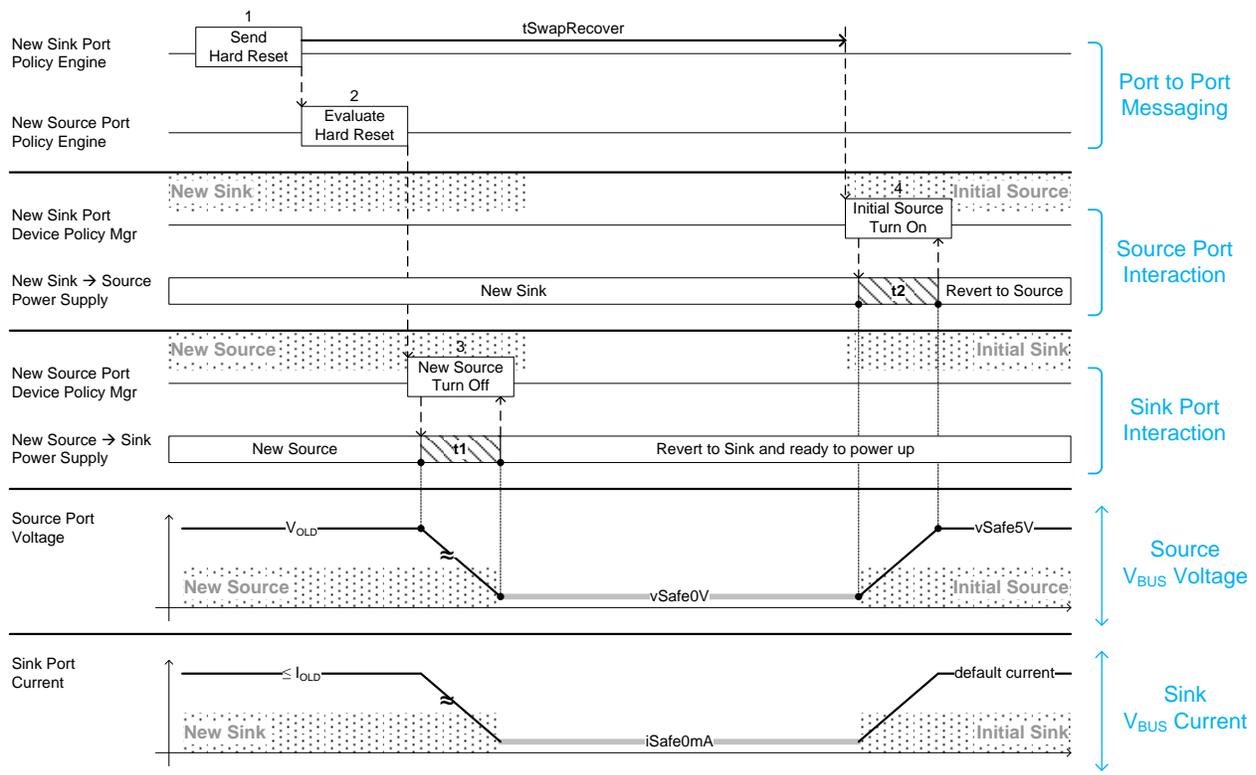


Table 7-18 Sequence Description for USB Type-A New Sink Initiated Hard Reset and USB Type-B New Source Receives Hard Reset Signaling

Step	New Source Port	New Sink Port
1		Policy Engine sends the Hard Reset Signaling to the new Source and waits tSwapRecover 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 tSafe0V (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 vSafe5V when V_{BUS} is within vSafe0V . The transition to vSafe5V Shall be completed with tSrcTurnOn (t2).

7.3.14.4 Type-A New Sink Initiates Hard Reset and USB 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 USB Type-A new Sink when the USB Type-B new Source does not receive the **Hard Reset** Signaling is shown in Figure 7-27. 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.

Figure 7-27 Transition Diagram for USB Type-A New Sink Initiated Hard Reset and USB Type-B New Source Does Not Receive Hard Reset Signaling

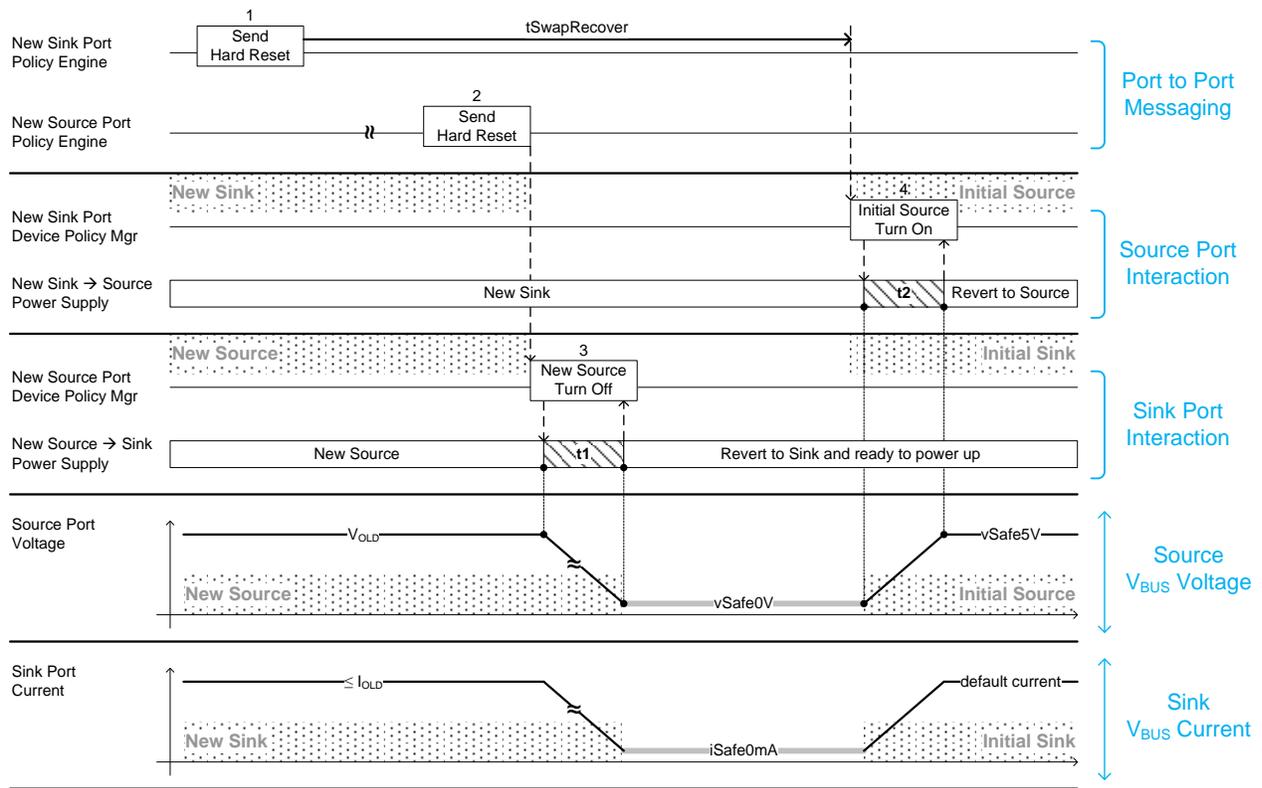


Table 7-19 Sequence Description for USB Type-A New Sink Initiated Hard Reset and USB Type-B New Source Does Not Receive Hard Reset Signaling

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.

7.3.15 Type-A to USB Type-B Dead Battery Operation

The interaction of the System Policy, Device Policy, and power supply that **shall** be followed during USB Type-A to USB Type-B Dead Battery operation is shown in Figure 7-28. 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.

Figure 7-28 USB Type-A to USB Type-B Transition Diagram for Dead Battery Operation

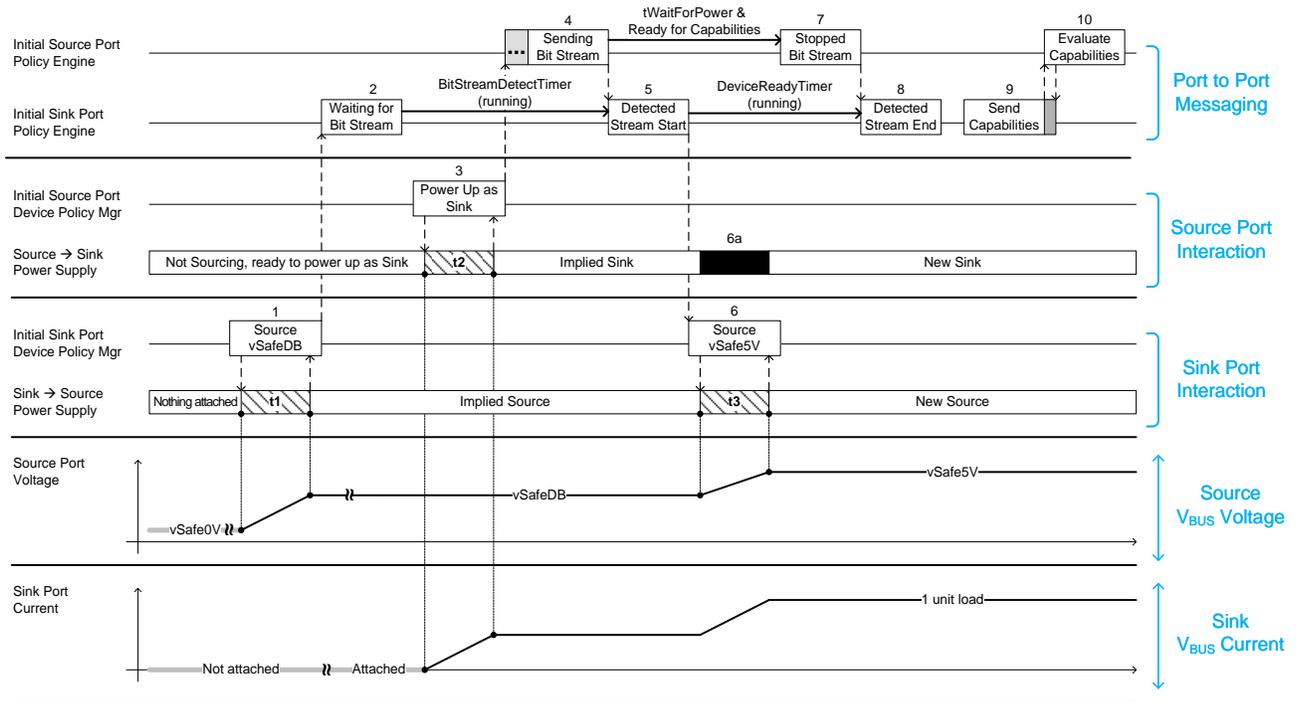


Table 7-20 Sequence Description for USB Type-A to USB Type-B Dead Battery Operation

Step	Initial Source Port	Initial Sink Port
1		The Initial Sink detects V_{BUS} is within <i>vSafe0V</i> and performs an implied Power Role Swap to apply <i>vSafeDB</i> to V_{BUS} . The turn on time for <i>vSafeDB</i> is <i>tTurnOnSafeDB</i> (t1).
2		The implied Source begins to wait for the Bit Stream after having applied <i>vSafeDB</i> to V_{BUS} . If an attach does not occur before <i>BitStreamDetectTimer</i> times out, then the implied Source removes <i>vSafeDB</i> and discharges V_{BUS} to <i>vSafe0V</i> . The Dead Battery start up routine for the Source will loop this behavior until an attach occurs (see Section 4.2.1). 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 <i>tTurnOnImpliedSink</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.	
4	If the implied Sink is unpowered it sends the Bit Stream after having been powered up for some time, by <i>vSafeDB</i> from the implied Source.	
5		When the Bit Stream is detected the <i>DeviceReadyTimer</i> is started and this diagram assumes the Bit Stream will end before the <i>DeviceReadyTimer</i> has timed out.
6		The implied Source will apply <i>vSafe5V</i> to V_{BUS} after the start of the Bit Stream has been detected. The transition time to change from <i>vSafeDB</i> to <i>vSafe5V</i> is <i>tSafeDBtoSafe5V</i> (t3).
6a	The implied Sink will transition to new Sink operation as the V_{BUS} voltage changes from <i>vSafeDB</i> to <i>vSafe5V</i> and May draw up to 1 unit load(as specified in [USB 2.0] or [USB 3.1]) once <i>vSafe5V</i> 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 <i>Source_Capabilities</i> Message.	
8		The new Source detects the end of the Bit Stream.
9		The Policy Engine of the new Source sends a <i>Source_Capabilities</i> Message to the new Sink.
10	The Policy Engine of the new Sink evaluates the <i>Source_Capabilities</i> 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 7-29. 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 7-29 Transition Diagram for no change in Current or Voltage

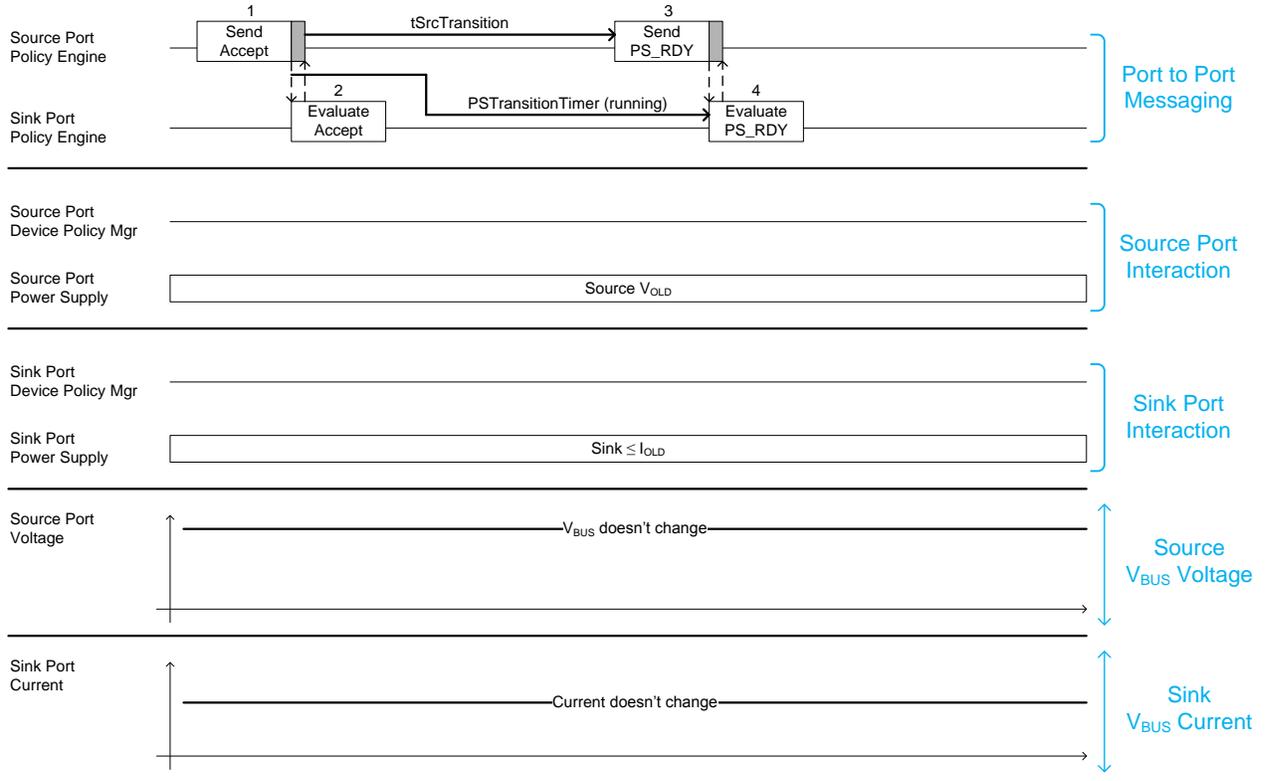


Table 7-21 Sequence Description for no change in Current or Voltage

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>cSrcBulk¹</i>	Source bulk capacitance when a Port is powered from a dedicated supply.	10			μF	Section 7.1.2
<i>cSrcBulkDB</i>	Source bulk capacitance on V _{BUS} when applying <i>vSafeDB</i> .	0		10	μF	Section 7.1.13.3
<i>cSrcBulkShared¹</i>	Source bulk capacitance when a Port is powered from a shared supply.	120			μF	Section 7.1.2
<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 7-19, Figure 7-20
<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 ₀) to when the Source is ready to provide the newly negotiated power level.			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
<i>tSrcSettle</i>	Time from positive/negative transition start (t ₀) to when the transitioning voltage is within the range <i>vSrcNew</i> .			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>	The maximum time for the Source output voltage to be between <i>vSrcNew</i> and <i>vSrcValid</i> in response to a load transient.			5	ms	Section 7.1.9
<i>tSrcTransition</i>	The time the Source Shall wait before transitioning the power supply to ensure that the Sink has sufficient time to prepare.	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

Parameter	Description	MIN	TYP	MAX	UNITS	Reference
<i>vSrcNeg</i>	Most negative voltage allowed during transition.			-0.3	V	Figure 7-4
<i>vSrcNew</i>	Fixed Supply output measured at the Source receptacle.	PDO Voltage *0.95	PDO Voltage	PDO Voltage *1.05	V	Figure 7-2 Figure 7-3
	Variable Supply output measured at the Source receptacle.	PDO Minimum Voltage		PDO Maximum Voltage	V	
	Battery Supply output measured at the Source receptacle.	PDO Minimum Voltage		PDO Maximum Voltage	V	
<i>vSrcPeak</i>	The range that a Fixed Supply in Peak Current operation is allowed when overload conditions occur.	PDO Voltage *0.90		PDO Voltage *1.05	V	Table 6-7 Figure 7-6
<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 µF.			-30	mV/ µs	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 µF.			30	mV/ µs	Section 7.1.4 Figure 7-2
<i>vSrcValid</i>	The range in addition to <i>vSrcNew</i> which a newly negotiated voltage is considered Valid during and after a transition. This range also applies to <i>vSafe5V</i> .	-0.5		0.5	V	Figure 7-2 Figure 7-3
Note 1: The Source Shall charge and discharge the total bulk capacitance to meet the transition time requirements.						

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>cSnkBulk¹</i>	Sink bulk capacitance on V _{BUS} at attach.	1		10	μF	Section 7.2.2
<i>cSnkBulkDB¹</i>	Bulk capacitance on V _{BUS} when acting as a Sink during Dead Battery.	1		10	μF	Section 7.2.9.3
<i>cSnkBulkPd</i>	Bulk capacitance on V _{BUS} a Sink is allowed after a successful negotiation.	1		100	μF	Section 7.2.2
<i>iLoadReleaseRate</i>	Load release di/dt, refer to [USB Type-C 1.2] Section 3.7.3.3.2 for cable details.	-150			mA/μs	Section 7.2.6
<i>iLoadStepRate</i>	Load step di/dt, refer to [USB Type-C 1.2] Section 3.7.3.3.2 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 [USB Type-C 1.2] Section 3.7.3.3.2 for cable details.	-230		230	mA	Section 7.2.6
<i>iSafe0mA</i>	Maximum current a Sink is allowed to draw when V _{BUS} is driven to <i>vSafe0V</i> .			1.0	mA	Figure 7-22 Figure 7-23
<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>	Maximum power consumption while in Sink Standby.			2.5	W	Section 7.2.3
<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 _{BUS} .	31			dB	Section 7.2.9.2
<i>tNewSnkRevertToSrc</i>	Hard Reset during Power Role Swap timing parameter.			90	ms	Figure 7-24, Figure 7-25, Figure 7-26, Figure 7-27
<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

Parameter	Description	MIN	TYP	MAX	UNITS	Reference
<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	Table 7-14
<i>tSnkStdby</i>	Time to transition to Sink Standby from Sink.			15	ms	Section 7.2.3
<i>tSnkSwapStdby</i>	Maximum time for the Sink to transition to Swap Standby.			15	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 **Shall** incorporate some form of V_{BUS} 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>	Time to reach <i>vSafe0V</i> max.			650	ms	Section 7.1.6 Figure 7-4 Table 7-14 Table 7-15 Table 7-16 Table 7-17 Table 7-18 Table 7-19
<i>tSafe5V</i>	Time to reach <i>vSafe5V</i> max.			275	ms	Section 7.1.5 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 [USB 2.0] and [USB 3.1] for allowable V _{BUS} voltage range.	4.75		5.5	V	Section 7.1.6
<i>vSafeDB</i>	Safe operating voltage for Dual-Role ports operating as DB Source. See [USB 2.0] and [USB 3.1] for allowable V _{BUS} voltage range.	4.75	Refer to Operating Region in Figure 7-8	5.5	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.

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 might be offered. After this the Device Policy Manager will consider the available power supply resources since this will bound which voltages and currents might 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 USB 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 might 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 Power 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 might 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 might 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.
- Managing Modes in its Port Partner and Cable Plug(s).

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 Dual-Role Power Device **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 might have no USB capability, or PD might 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 [USBCBridge 1.0]. 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 might request that a Battery powered Device temporarily stops charging so that there is sufficient power for a HDD to spin up.

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 Dual-Role Power Device **Shall** manage the transition between Source/Sink roles for each PD Dual-Role Power 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 4.2.

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 USB 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 might be necessary for the Device Policy Manager to also initiate additional discovery using the **Discover Identity** Command 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 might 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 might 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, might 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 might use the Minimum Operating Current and GiveBack flag.

For example in the first case, a mobile device might 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) might 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 might 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 might have the extra power available immediately and can immediately honor the request. If the power is not available, the Provider might 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 8.3.2.10.4. 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 "Unconstrained Power" 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 "Unconstrained Power" 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 [\[USBCBridge 1.0\]](#)) and the Battery state (see Section 9.4.1) **Shall** also be reported to the System Policy Manager using the USB interface.

8.2.6.1 AC Supplies

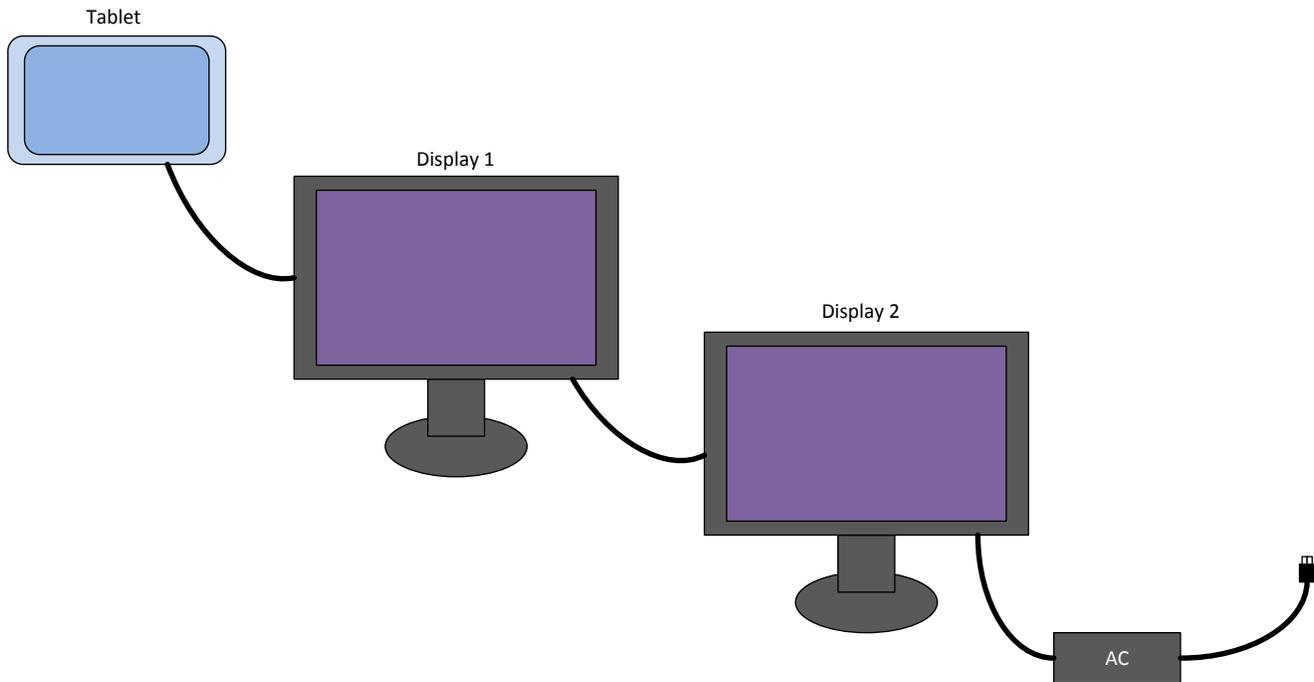
The Unconstrained Power bit provided by Sources and Sinks (see Section 6.4.1.3.1.3 and Section 6.4.1.2.3.3) notifies a connected device that it is acceptable to use the advertised power for charging as well as for what is needed for normal operation. A device that sets the Unconstrained Power bit has either an external source of power that is sufficient to adequately power the system while charging external devices, or expects to charge external devices as a primary state of function (such as a battery pack).

In the case of the external power source, the 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 unconstrained power from its power supply. The Unconstrained Power 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, from a battery bank, or similar:

- If the "Unconstrained Power" 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 have "Unconstrained Power" for the power advertised as a provider Port if there is unconstrained power beyond that needed for normal operation coming 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 has "Unconstrained Power" 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 could 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 a sufficiently-sized wall wart to the 1st display, by setting its "Unconstrained Power" bit. The 1st display will then in turn assess and indicate the presence of the extra power to the tablet by setting its "Unconstrained Power" 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



Another example use case is a Laptop computer that is attached to both an external supply and a Tablet computer. In this situation, if the external supply is large enough to power the laptop in its normal state as well as charge an external device, the laptop would set its “Unconstrained Power” bit and the tablet will allow itself to charge at its peak rate. If the external supply is small, however, and would not prevent the laptop from discharging if maximal power is drawn by the external device, the laptop would not set its “Unconstrained Power” bit, and the tablet can choose to draw less than what is offered. This amount could be just enough to prevent the tablet from discharging, or none at all. Alternatively, if the tablet determines that the laptop has significantly larger battery with more charge than the tablet has, the tablet can still choose to charge itself, although possibly not at the maximal rate.

In this way, Sinks that do not receive the “Unconstrained Power” bit from the connected Source can still choose to charge their batteries, or charge at a reduced rate, if their policy determines that the impact to the Source is minimal -- such as in the case of a phone with a small battery charging from a laptop with a large battery. These policies can be decided via further USB PD communication.

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.

If the device is battery-powered but is in a state that is primarily for charging external devices, the device is considered to be an unconstrained source of power and thus **Should** set the “Unconstrained Power” bit.

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 do not have Unconstrained Power, they **Should** define their own policies so as to prevent constant Power Role Swapping.

This algorithm uses the “Unconstrained Power” bit (see Section 6.4.1.2.3.3 and Section 6.4.1.3.1.3), thus the decisions are based on the availability and sufficiency of an external supply, not the full capabilities of a system or device or product.

Recommendations:

1. Provider/Consumers using large external sources (“Unconstrained Power” bit set) **Should** always deny Power Role Swap requests from Consumer/Providers not using external sources (“Unconstrained Power” bit cleared).
2. Provider/Consumers not using large external sources (“Unconstrained Powered” bit cleared) **Should** always accept a Power Role Swap request from a Consumer/Provider using large external power sources (“Unconstrained Power” 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 USB Type-A to USB Type-B Dead Battery (see Section 4.2.1).

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 4.2).

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 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
- State diagrams covering operation of Sources, Sinks and Cable Plugs

8.3.2 Atomic 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.

An AMS **shall** be defined as a Message sequence that starts and/or ends in either the *PE_SRC_Ready*, *PE_SNK_Ready* or *PE_CBL_Ready* states (see Section 8.3.3.2, Section 8.3.3.3 and Section 8.3.3.10).

In addition the Cable Plug discovery sequence specified in Section 8.3.3.10.11 **shall** be defined as an AMS.

The Source and Sink indicate to the Protocol Layer when an AMS starts and ends on entry to/exit from *PE_SRC_Ready* or *PE_SNK_Ready* (see Section 8.3.3.2 and Section 8.3.3.3).

Section 8.3.2.2.1 gives details of which of these AMS's are Interruptible or Non-interruptible.

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 might be either a Source or Sink while the receiver might 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)

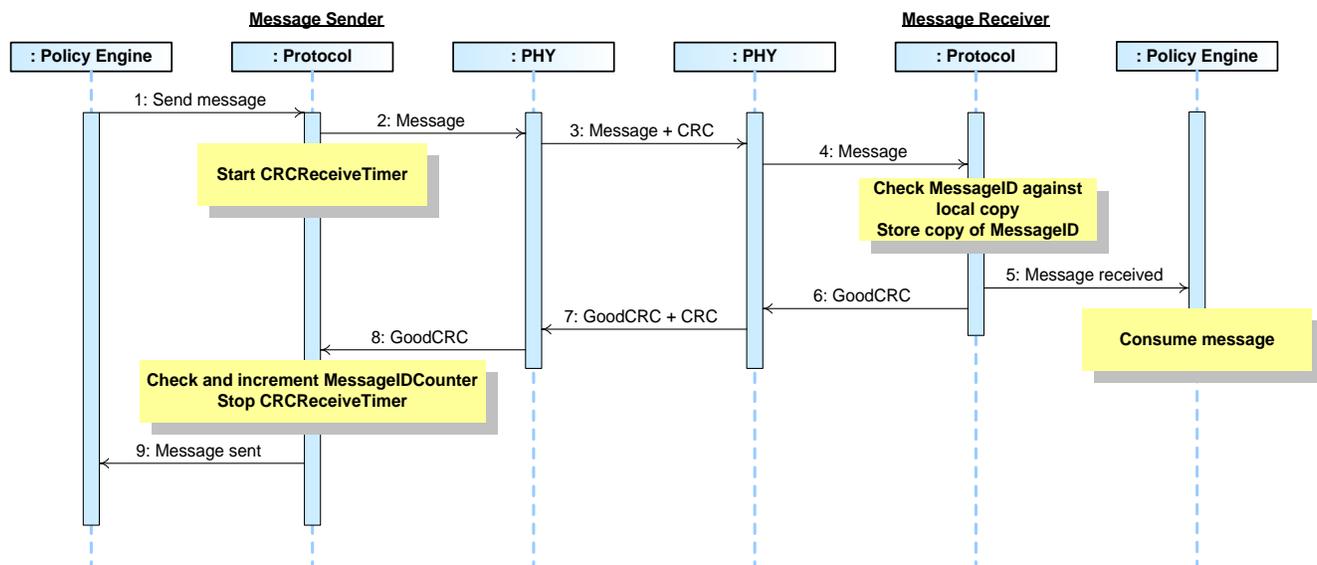


Table 8-1 Basic Message Flow

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 point issues can occur.

Figure 8-3 Basic Message flow indicating possible errors

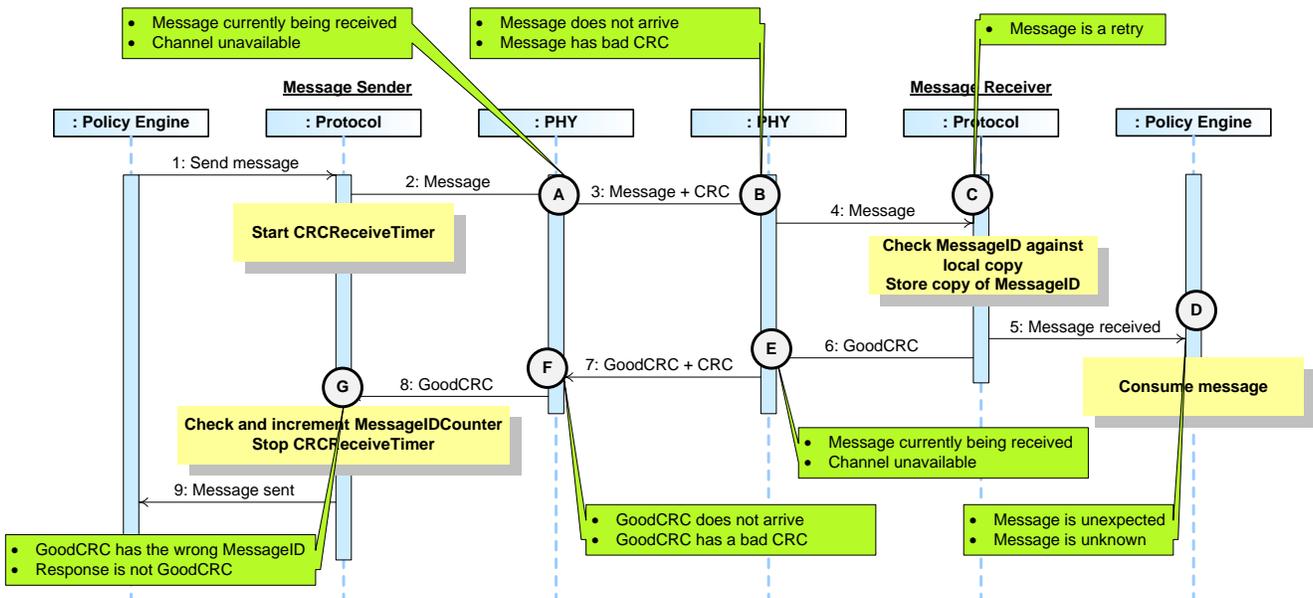


Table 8-2 Potential issues in Basic Message Flow

Point	Possible issues
A	<ol style="list-style-type: none"> 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. 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.
B	<ol style="list-style-type: none"> 1. Message does not arrive at the Physical Layer due to noise on the channel. 2. Message arrives but has been corrupted and has a bad CRC. <p>There is no Message to pass up to the Protocol Layer on the receiver which means a GoodCRC Message is not sent. This leads to a CRCReceiveTimer timeout in the Message Sender.</p>
C	<ol style="list-style-type: none"> 1. MessageID of received Message matches stored MessageID so this is a retry. Message is not passed up to the Policy Engine.
D	<ol style="list-style-type: none"> 1. Policy Engine receives a known Message that it was not expecting. 2. Policy Engine receives an unknown (unrecognized) Message. <p>These cases are errors in the protocol which leads to the generation of a Soft_Reset Message.</p>
E	Same as point A but at the Message Receiver side.
F	<ol style="list-style-type: none"> 1. GoodCRC Message response does not arrive at the Message Sender side due to the noise on the channel. 2. GoodCRC Message response arrives but has a bad CRC. <p>A GoodCRC Message is not received by the Message Sender's Protocol Layer. This leads to a CRCReceiveTimer timeout in the Message Sender.</p>
G	<ol style="list-style-type: none"> 1. GoodCRC Message is received but does contain the same MessageID as the transmitted Message. 2. A Message is received but it is not a GoodCRC Message (similar case to that of an unexpected or unknown Message but this time detected in the Protocol Layer). <p>Both of these issues indicate errors in receiving an expected GoodCRC Message which will lead to a CRCReceiveTimer 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 "n' strikes and you are out" basis (**nRetryCount**).

Figure 8-4 Basic Message Flow with Bad CRC followed by a Retry

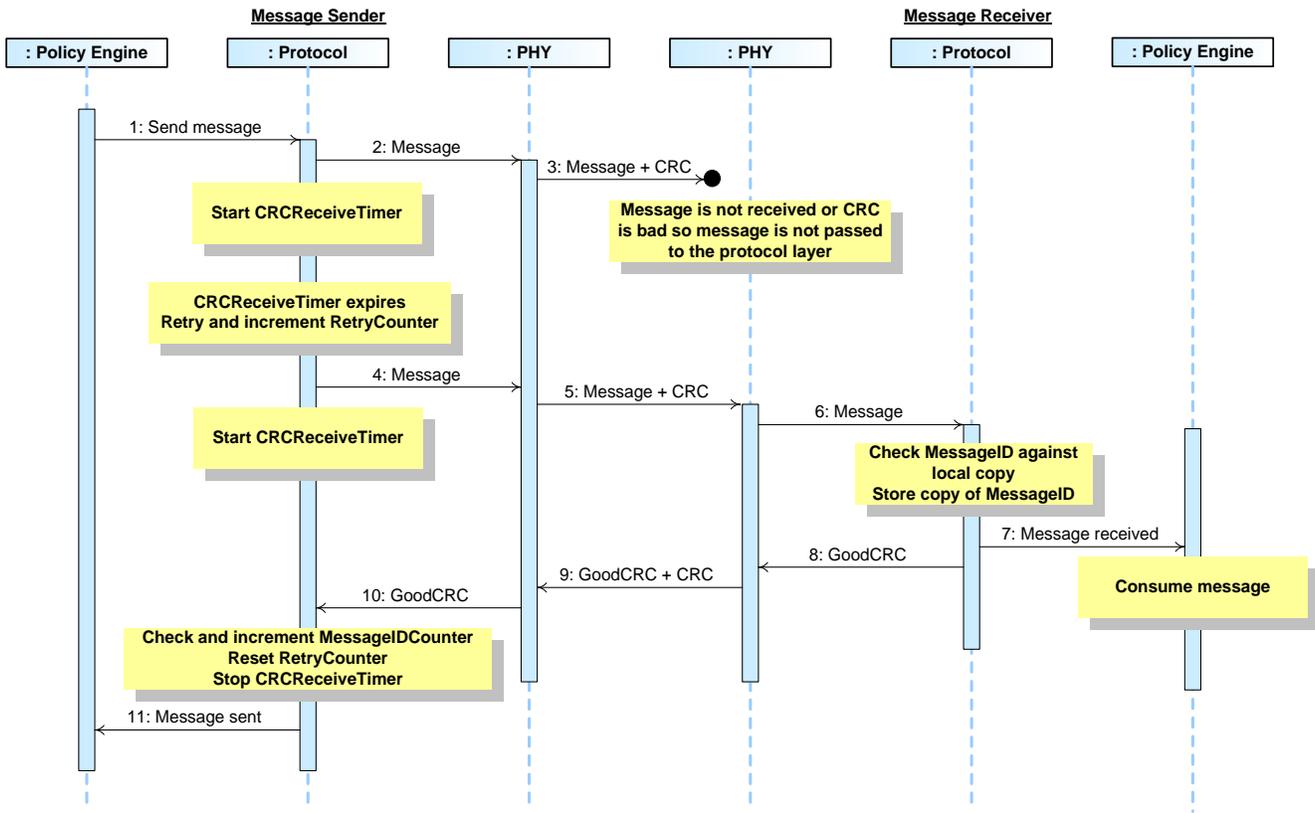


Table 8-3 Basic Message Flow with CRC failure

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.	

8.3.2.2.1 Interruptible and Non-interruptible Atomic Message Sequences

Table 8-4 details which AMS (as defined in Section) **Shall** be treated as Interruptible or Non-interruptible during the sequence. Every AMS which starts with the same Message **Shall** obey the Interruptible/Non-interruptible requirement. Note that every AMS is Interruptible until the first Message in the sequence has been successfully sent (*GoodCRC* Message received). Any Sequence of VDMs **Shall** be Interruptible. After the AMS that caused the interruption has completed, if the original AMS is still needed the interrupted AMS **Shall** be Re-run.

Table 8-4 Interruptible and Non-interruptible AMS

AMS	Interruptible	Reference
Power Negotiation	No	Section 8.3.3.2, Section 8.3.3.3
GotoMin	No	Section 8.3.3.2, Section 8.3.3.3
Soft Reset	No	Section 8.3.3.4
Hard Reset	No	Section 8.3.3.2, Section 8.3.3.3
Cable Reset	No	Section 8.3.3.10.9
Get Source Capabilities	No	Section 8.3.3.2, Section 8.3.3.3
Get Sink Capabilities	No	Section 8.3.3.2, Section 8.3.3.3
Power Role Swap	No	Section 8.3.3.6.3.1, Section 8.3.3.6.3.2
Data Role Swap	No	Section 8.3.3.6.2.1, Section 8.3.3.6.2.2
VCONN Swap	No	Section 8.3.3.7
Discover Identity	Yes	Section 8.3.3.9.1, Section 8.3.3.8.1
Source startup Cable Plug Discover Identity	Yes	Section 8.3.3.9.1, Section 8.3.3.10.11
Discover SVIDs	Yes	Section 8.3.3.8.2, Section 8.3.3.9.3
Discover Modes	Yes	Section 8.3.3.8.3, Section 8.3.3.9.4
DFP to UFP Enter Mode	Yes	Section 8.3.3.9.5, Section 8.3.3.8.4
DFP to UFP Exit Mode	Yes	Section 8.3.3.9.6, Section 8.3.3.8.5
DFP to Cable Plug Enter Mode	Yes	Section 8.3.3.9.5, Section 8.3.3.8.4
DFP to Cable Plug Exit Mode	Yes	Section 8.3.3.9.6, Section 8.3.3.8.5
Attention	<i>N/A</i>	Section 8.3.3.8.6, Section 8.3.3.9.7
Built in Self-Test (BIST)	No	Section 8.3.2.10
Sequence of Unstructured VDMs	Yes	Section 6.4.4.1
Sequence of Structured VDMs using Vendor Commands	Yes	Section 6.4.4.2

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

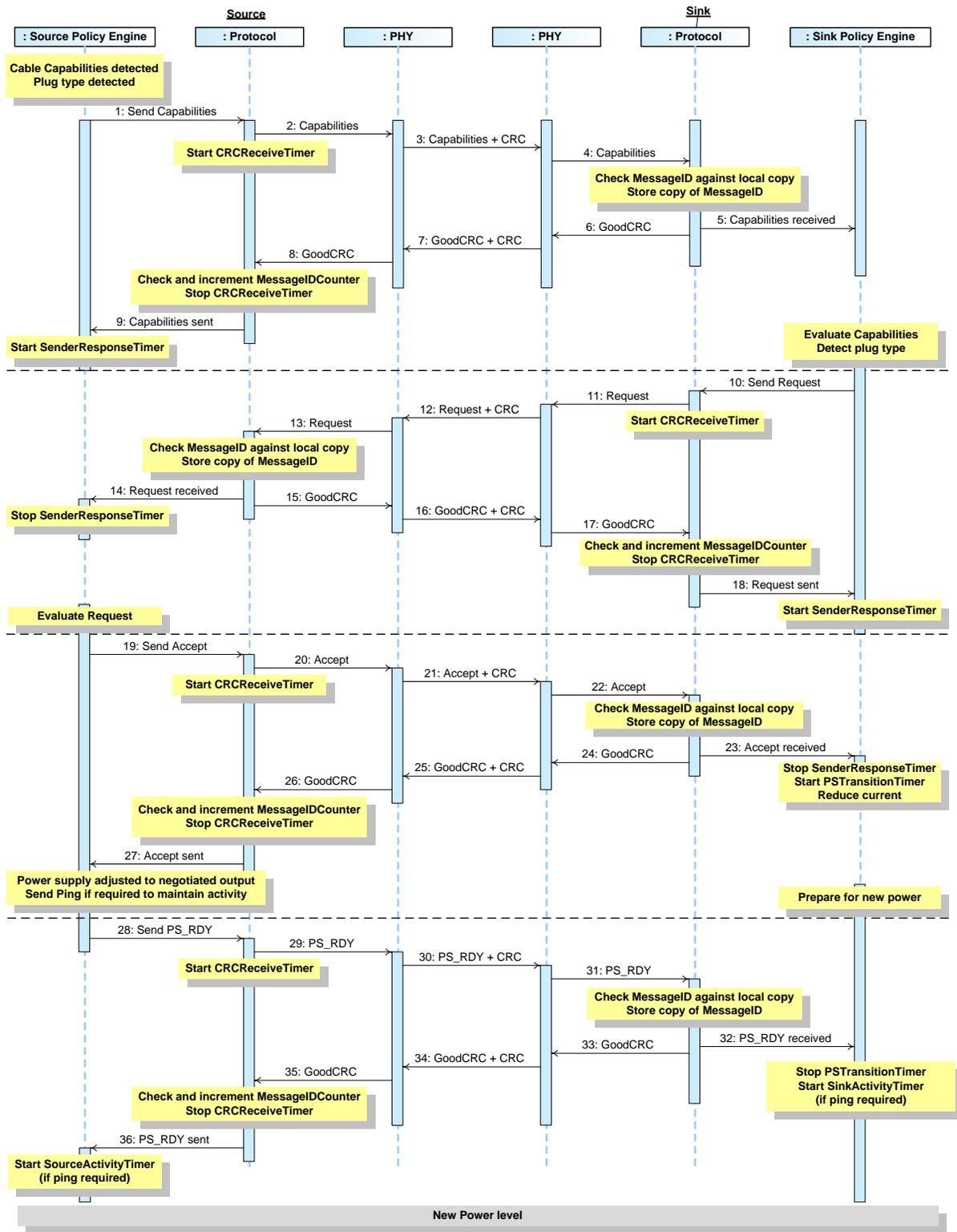


Table 8-5 below provides a detailed explanation of what happens at each labeled step in Figure 8-5 above.

Table 8-5 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 4.5). 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 4.5) 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.	
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.

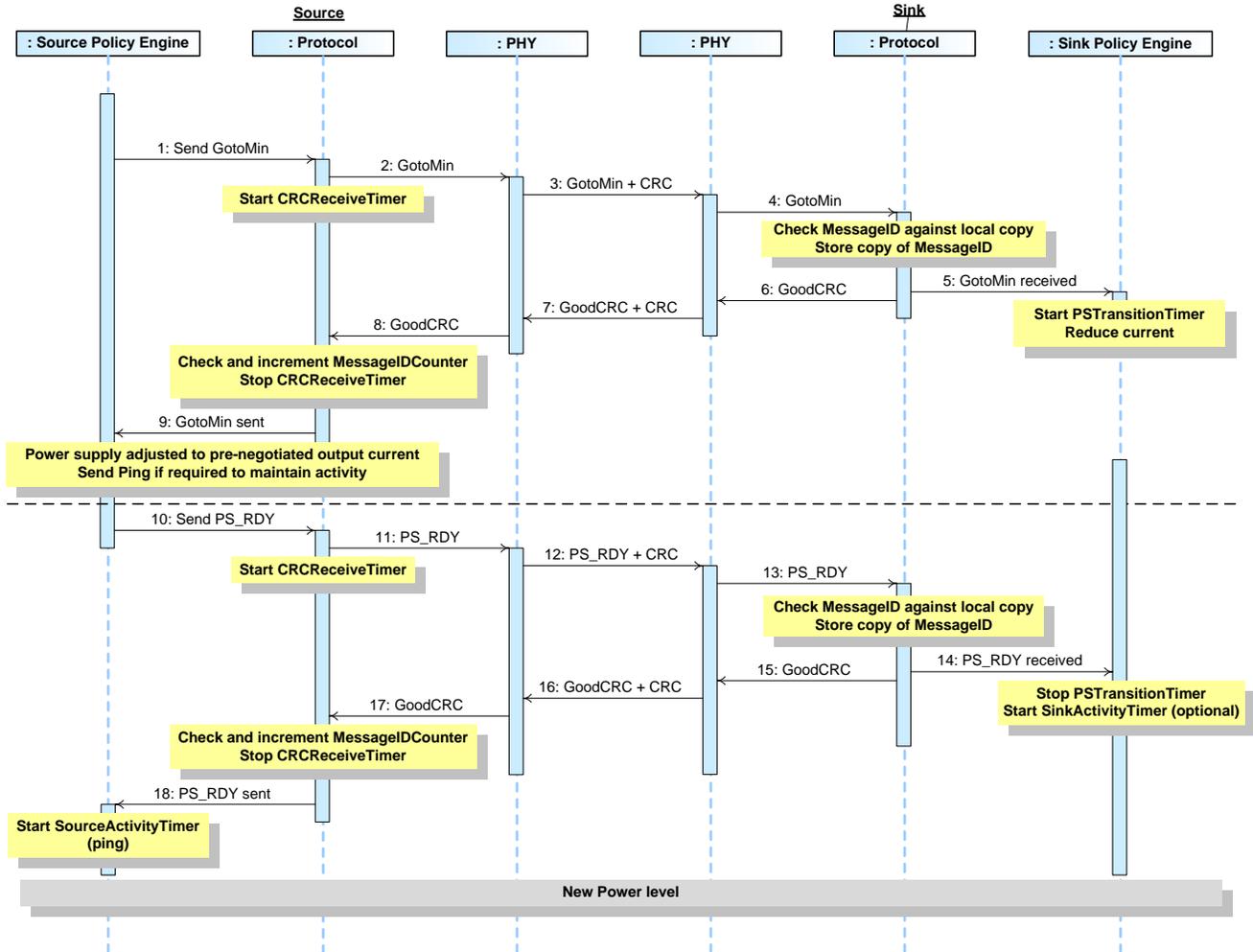
Step	Source	Sink
17		Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		The protocol Layer verifies and increments the <i>MessageIDCounter</i> . It informs the Policy Engine that the <i>Request</i> Message was successfully sent. The Protocol Layer stops the <i>CRCReceiveTimer</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.	
power supply Adjusts its Output to the Negotiated Value		
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.
31		Physical Layer forwards the <i>PS_RDY</i> Message to the Protocol Layer.

Step	Source	Sink
32		<p>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.</p> <p>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).</p>
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 8-6 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 <i>Optionally</i> 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

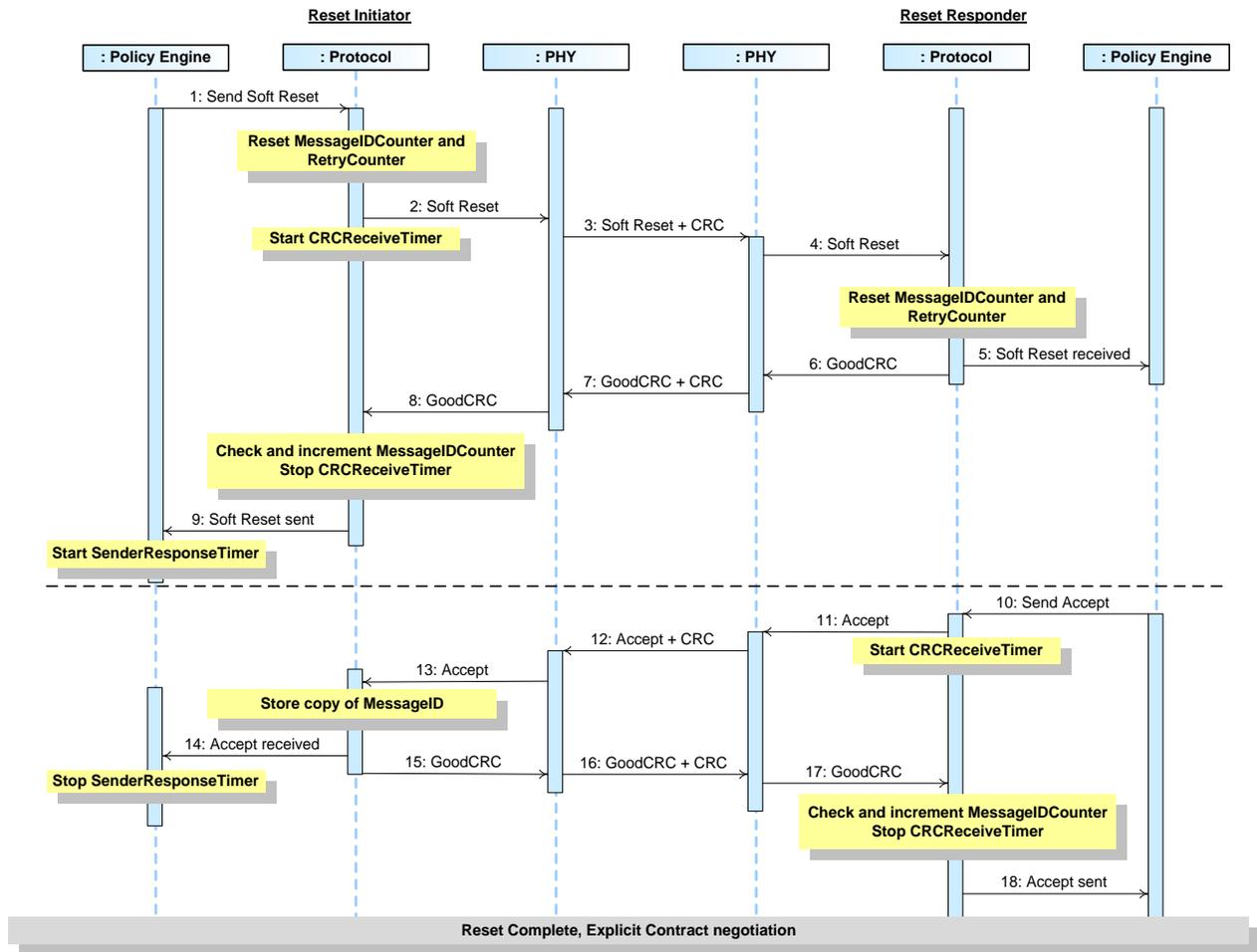


Table 8-7 below provides a detailed explanation of what happens at each labeled step in Figure 8-7 above.

Table 8-7 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.
4		Physical Layer removes the CRC and forwards the <i>Soft_Reset</i> Message to the Protocol Layer.

Step	Reset Initiator	Reset Responder
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. Port Partners perform an Explicit Contract negotiation to re-synchronize their state machines.	

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 Source Initiated Hard Reset

This is an example of a Hard Reset operation when initiated by a Source. Figure 8-8 shows the Messages as they flow across the bus and within the devices to accomplish the Hard Reset.

Figure 8-8 Source initiated Hard Reset

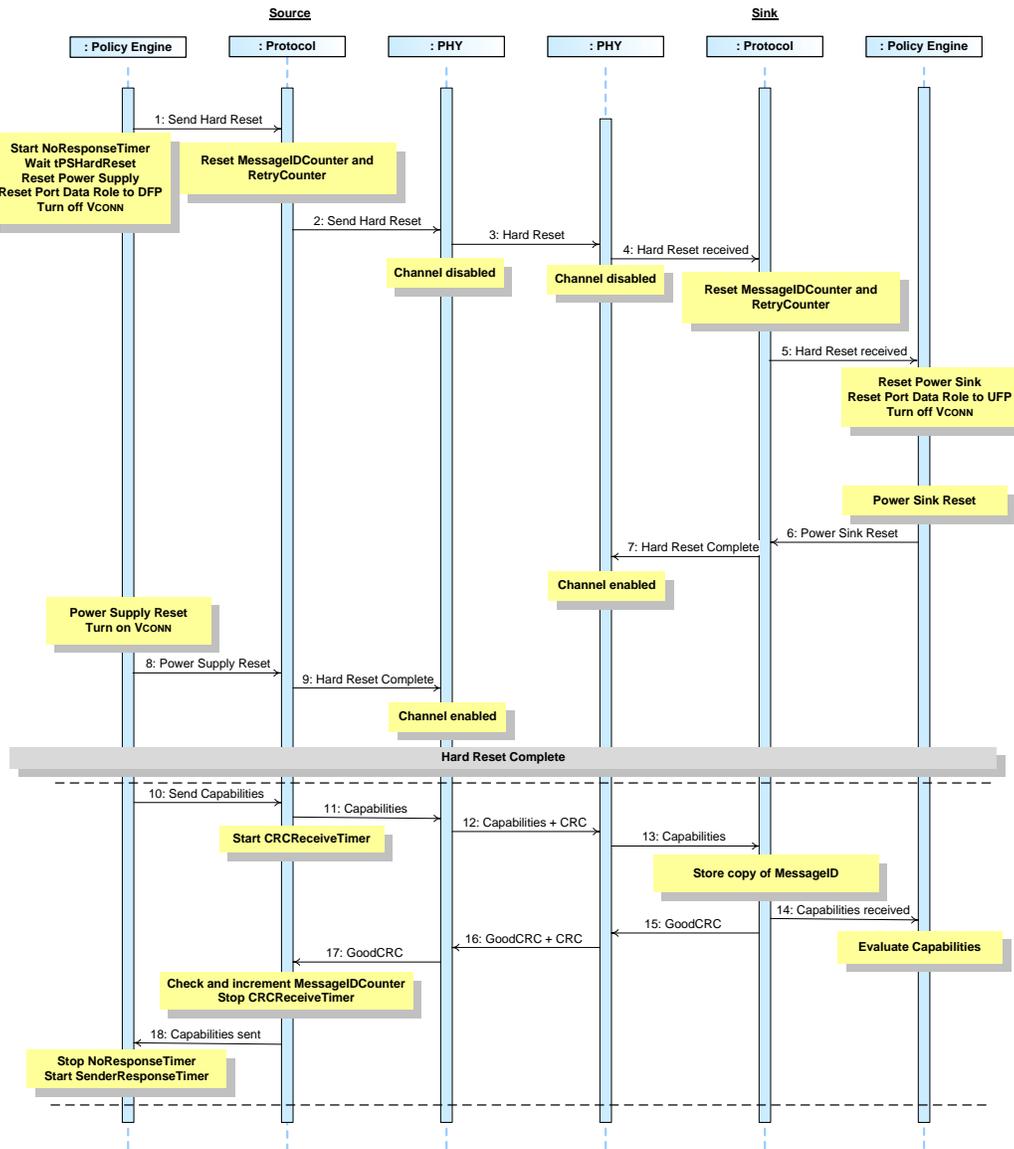


Table 8-8 Steps for Source initiated Hard Reset

Step	Source	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 requests the Device Policy Manager to reset the power supply to USB Default Operation. If this is a USB 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.	
2	Protocol Layer resets <i>MessageIDCounter</i> and <i>RetryCounter</i> . Protocol Layer requests the Physical Layer send <i>Hard Reset</i> Signaling.	
3	Physical Layer sends <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> and <i>RetryCounter</i> .
5		The Protocol Layer informs the Policy Engine of the Hard Reset. The Policy Engine requests the Device Policy Manager to reset the Power Sink to default operation. If this is a USB 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.
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. If this is a USB Type-C connector VCONN is turned on. 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.
13		Physical Layer removes the CRC and forwards the <i>Source_Capabilities</i> Message to the Protocol Layer.

Step	Source	Sink
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.
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 Sink Initiated Hard Reset

This is an example of a Hard Reset operation when initiated by a Sink. Figure 8-9 shows the Messages as they flow across the bus and within the devices to accomplish the Hard Reset.

Figure 8-9 Sink Initiated Hard Reset

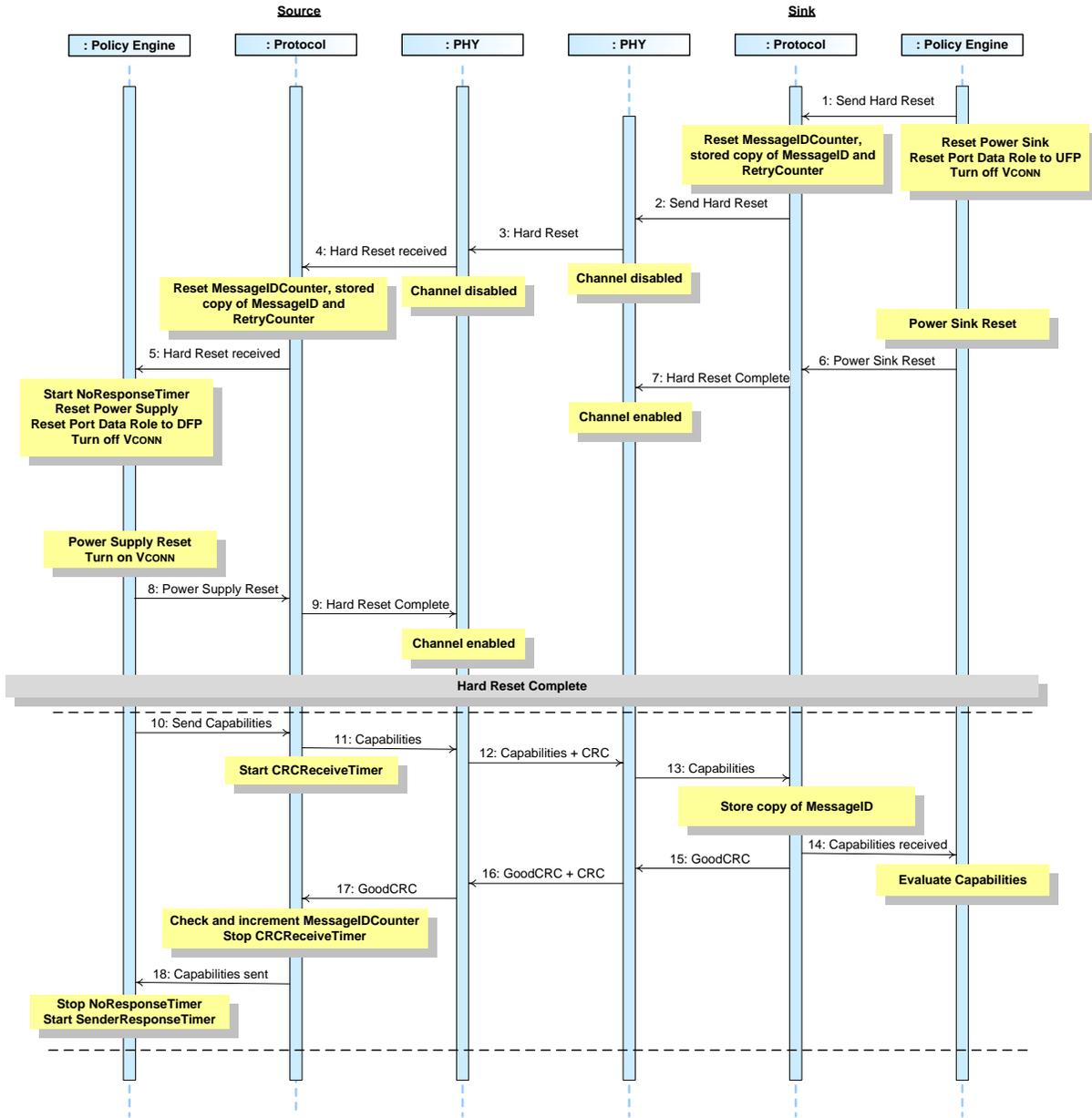


Table 8-9 Steps for Sink initiated Hard Reset

Step	Source	Sink
1		The Policy Engine directs the Protocol Layer to generate Hard Reset Signaling. The Policy Engine requests the Device Policy Manager to reset the power supply to USB Default Operation. If this is a USB 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.
2		Protocol Layer resets MessageIDCounter , stored copy of MessageID and RetryCounter . Protocol Layer requests the Physical Layer send Hard Reset Signaling.
3	Physical Layer receives the Hard Reset Signaling and disables the PHY Layer communications channel for transmission and reception.	Physical Layer sends the Hard Reset 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 MessageIDCounter , stored copy of MessageID and RetryCounter .	
5	The Protocol Layer Informs the Policy Engine of the Hard Reset. The Policy Engine starts the NoResponseTimer and requests the Device Policy Manager to reset the Power Sink to default operation. If this is a USB 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.	
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. If this is a USB Type-C connector VCONN is turned on. 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 Source_Capabilities Message that represents the power supply's present capabilities.	
11	Protocol Layer creates the Message and passes to Physical Layer. Starts CRCReceiveTimer .	
12	Physical Layer appends CRC and sends the Source_Capabilities Message.	Physical Layer receives the Source_Capabilities Message and checks the CRC to verify the Message.
13		Physical Layer removes the CRC and forwards the Source_Capabilities Message to the Protocol Layer.

Step	Source	Sink
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.
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 Source Initiated Hard Reset – Sink Long Reset

This is an example of a Hard Reset operation when initiated by a Source. In this example the Sink is slow responding to the reset causing the Source to send multiple *Source_Capabilities* Messages before it receives a *GoodCRC* Message response. Figure 8-10 shows the Messages as they flow across the bus and within the devices to accomplish the Hard Reset.

Figure 8-10 Source initiated reset - Sink long reset

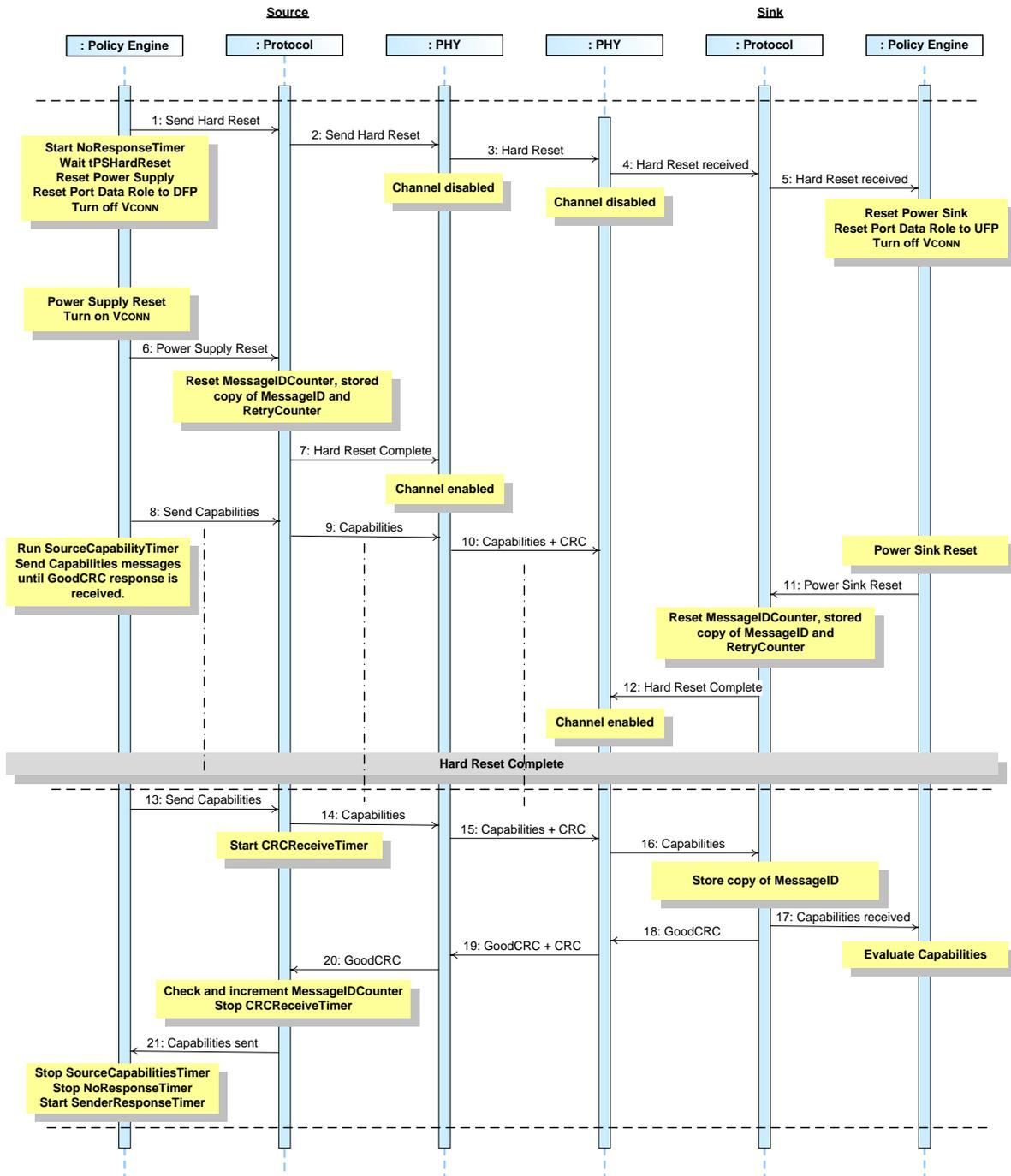


Table 8-10 Steps for Source initiated Hard Reset – Sink long reset

Step	Source	Sink
1	The Policy Engine directs the Protocol Layer to generate Hard Reset Signaling. The Policy Engine starts the NoResponseTimer and requests the Device Policy Manager to reset the power supply to USB Default Operation. If this is a USB 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.	
2	Protocol Layer resets MessageIDCounter , stored copy of MessageID and RetryCounter . Protocol Layer requests the Physical Layer send Hard Reset Signaling.	
3	Physical Layer sends the Hard Reset Signaling and then disables the PHY Layer communications channel for transmission and reception.	Physical Layer receives the Hard Reset 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 MessageIDCounter , stored copy of MessageID and RetryCounter .
5		The Protocol Layer Informs the Policy Engine of the Hard Reset. The Policy Engine requests the Device Policy Manager to reset the Power Sink to default operation. If this is a USB 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.
6	The power supply is reset to USB Default Operation. If this is a USB Type-C connector VCONN is turned on. 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 Source_Capabilities Message that represents the power supply's present capabilities. Policy Engine starts the SourceCapabilityTimer . The SourceCapabilityTimer times out one or more times until a GoodCRC Message response is received.	
9	Protocol Layer creates the Message and passes to Physical Layer. Starts CRCReceiveTimer .	
10	Physical Layer appends CRC and sends the Source_Capabilities Message.	Note: Source_Capabilities 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	Source	Sink
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.
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 USB Type-A or USB 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 USB Type-A or USB Type-B Successful Power Role Swap Sequence Initiated by the Source

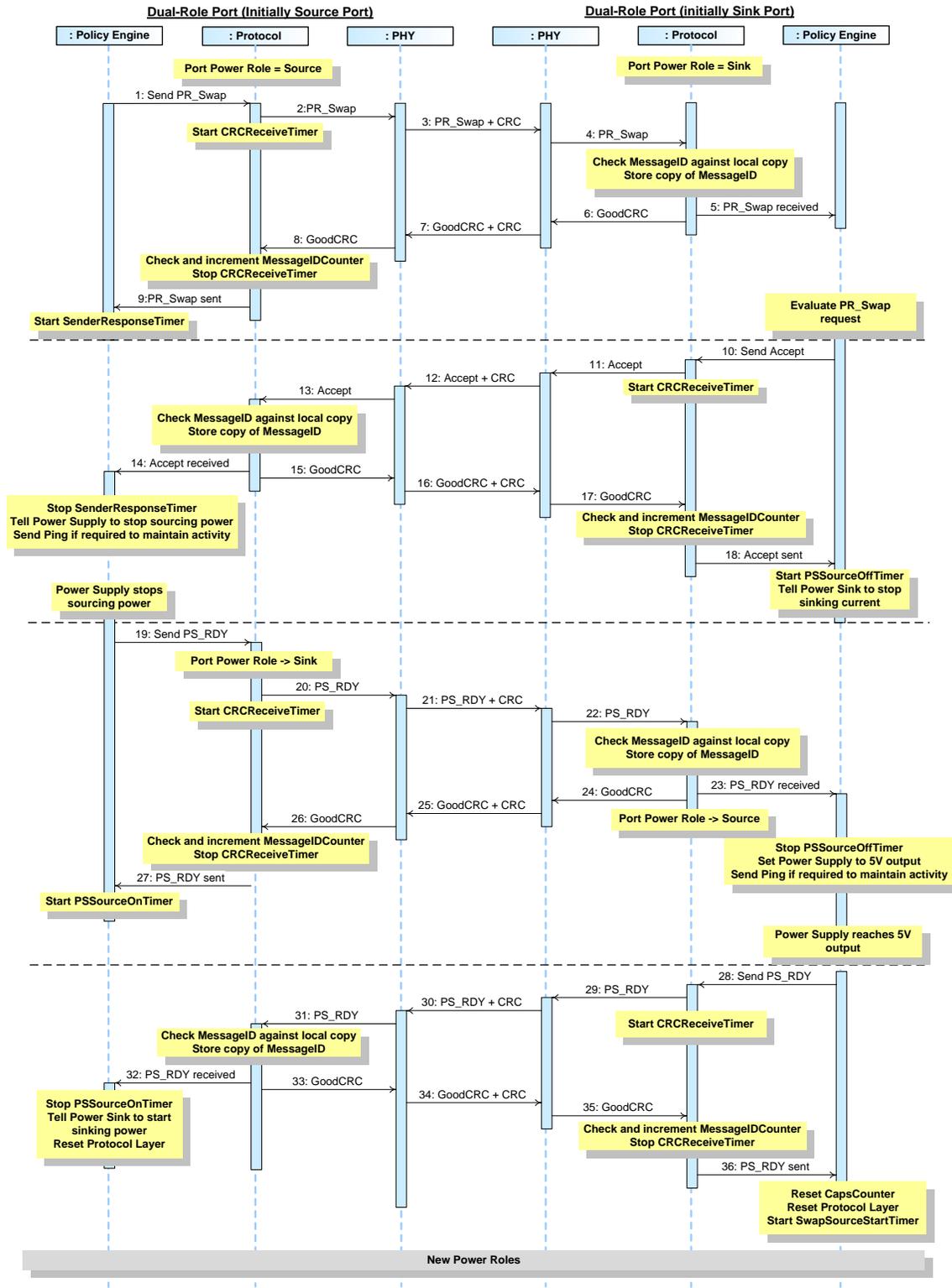


Table 8-11 below provides a detailed explanation of what happens at each labeled step in Figure 8-11 above.

Table 8-11 Steps for a Successful USB Type-A or USB Type-B Source Initiated Power Role Swap Sequence

Step	Dual-Role Port (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-Role Port (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 Message 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 Message 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-Role Port (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 USB Type-A or USB 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 8.3.2.2.1).

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 USB Type-A or USB Type-B Successful Power Role Swap Sequence Initiated by the Sink

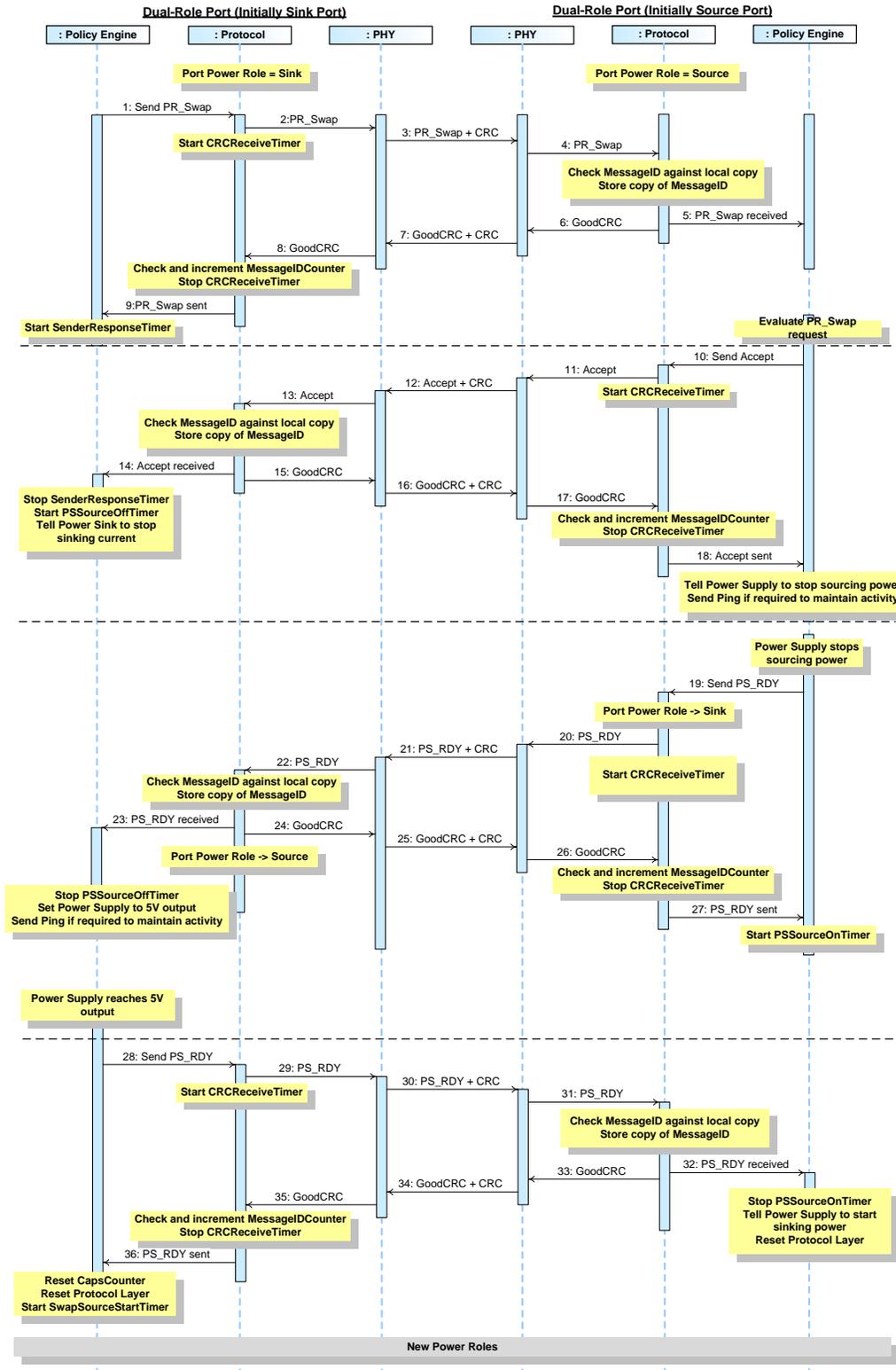


Table 8-12 Steps for a Successful USB Type-A or USB Type-B Sink Initiated Power Role Swap Sequence

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 Message 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 Message 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 <i>Optionally</i> 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 USB Type-A or USB Type-B Dual-Role Power 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 Power 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 USB Type-A or USB Type-B Source initiated Hard Reset (Power Role Swapped)

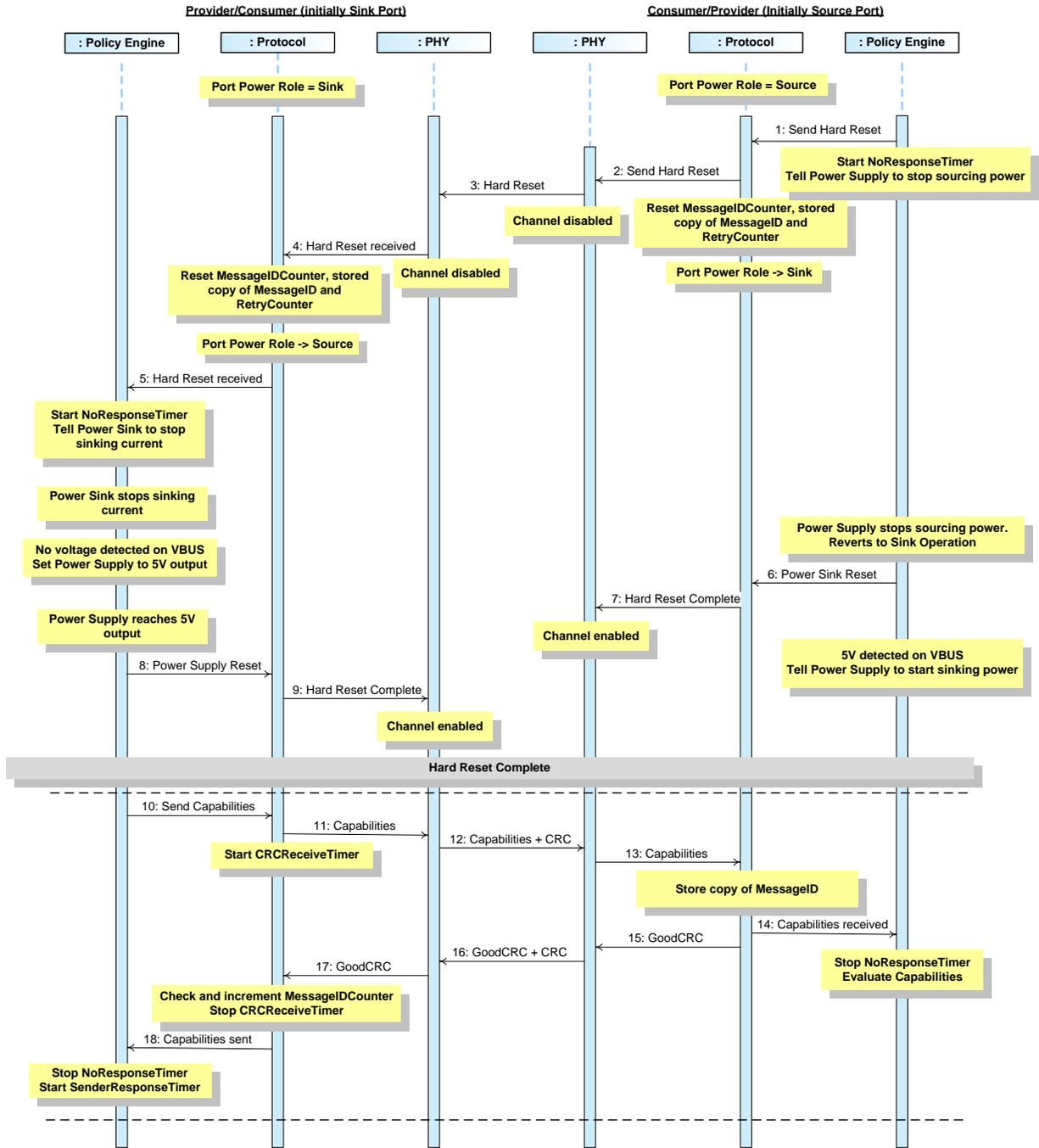


Table 8-13 Steps for USB Type-A or USB Type-B Source initiated Hard Reset (Power Role Swapped)

Step	Provider/Consumer (Initially Sink Port)	Consumer/Provider (Initially Source 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 (Initially Sink Port)	Consumer/Provider (Initially Source 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 USB Type-A or USB Type-B Dual-Role Power 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 Power devices return 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 USB Type-A or USB Type-B Sink Initiated Hard Reset (Power Role Swapped)

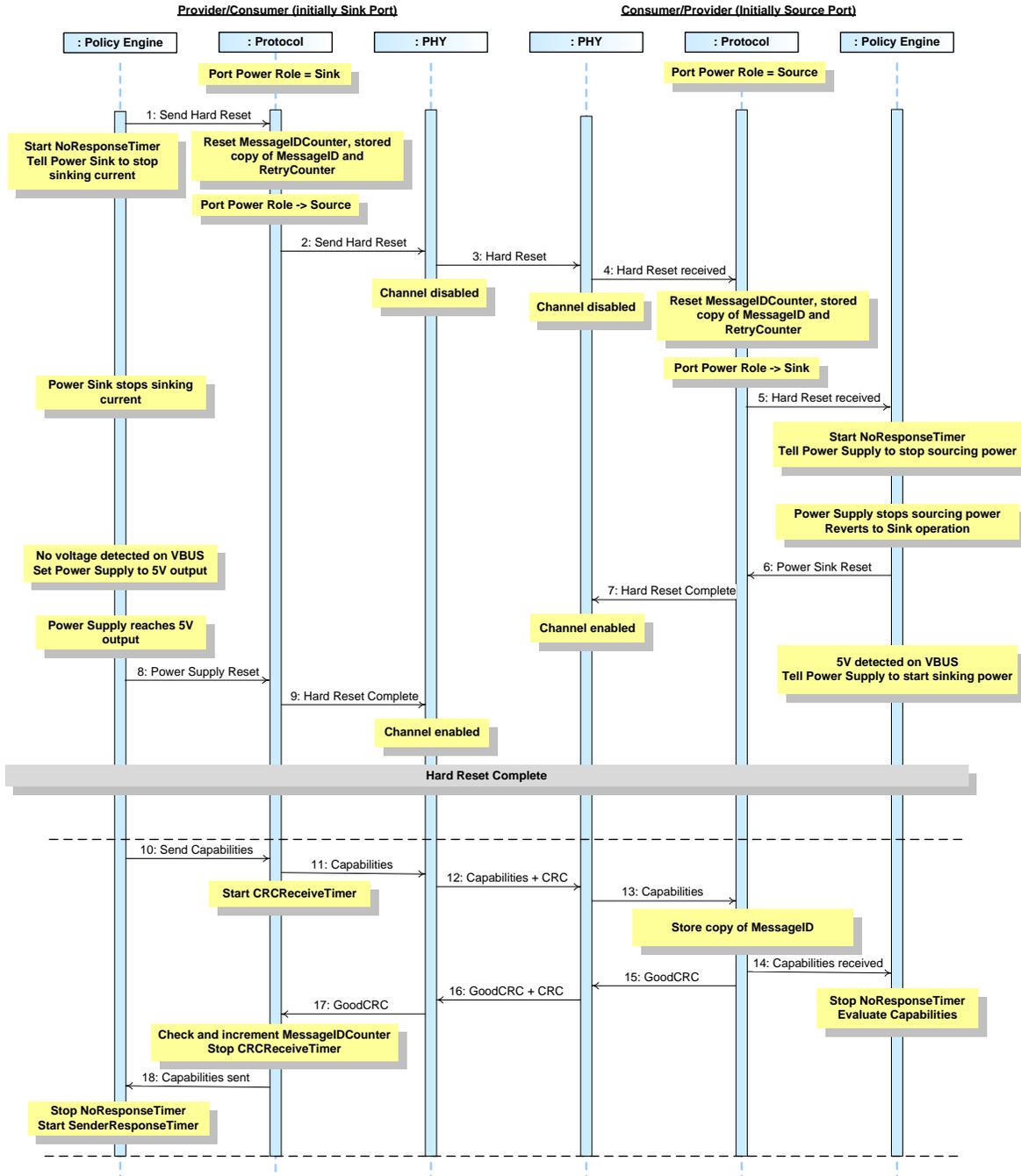


Table 8-14 Steps for USB Type-A or USB Type-B Sink initiated Hard Reset (Power Role Swapped)

Step	Provider/Consumer (Initially Sink Port)	Consumer/Provider (Initially Source 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 (Initially Sink Port)	Consumer/Provider (Initially Source 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 *[USB Type-C 1.2]* 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 USB Type-C Successful Power Role Swap Sequence Initiated by the Source

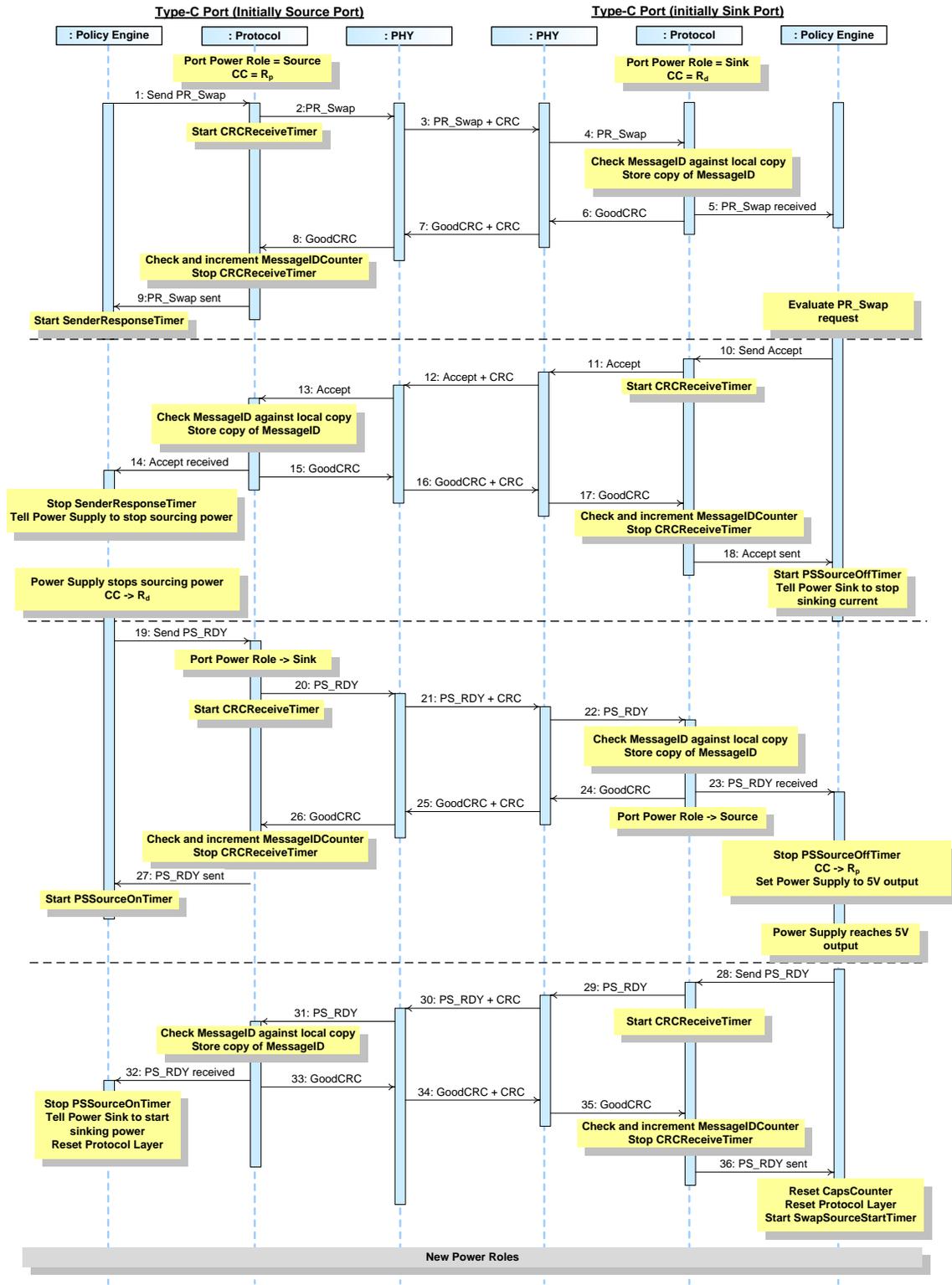


Table 8-11 below provides a detailed explanation of what happens at each labeled step in Figure 8-15 above.

Table 8-15 Steps for a Successful USB Type-C Source Initiated Power Role Swap Sequence

Step	Type-C Port (initially Source Port)	Type-C Port (initially Sink Port)
1	The [USB Type-C 1.2] Port has Port Power Role set to Source and the R _p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a PR_Swap Message.	The [USB Type-C 1.2] Port has Port Power Role set to Sink with the R _a pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer. Starts CRCReceiveTimer .	
3	Physical Layer appends CRC and sends the PR_Swap Message.	Physical Layer receives the PR_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the PR_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID 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 PR_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the PR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the PR_Swap 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 Accept Message.
11		Protocol Layer creates the Message and passes to Physical Layer. Starts CRCReceiveTimer .
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Accept Message.	Physical Layer appends a CRC and sends the Accept Message.
13	Protocol Layer checks the MessageID 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 PR_Swap Message information to the Policy Engine that consumes it.	
14	The Policy Engine requests its power supply to stop supplying power and stops the SenderResponseTimer .	
15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the GoodCRC 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 Message 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 Message 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 *[USB Type-C 1.2]* 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 8.3.2.2.1).

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 8-16 shows the Messages as they flow across the bus and within the devices to accomplish the Power Role Swap.

Figure 8-16 USB Type-C Successful Power Role Swap Sequence Initiated by the USB Type-C Sink

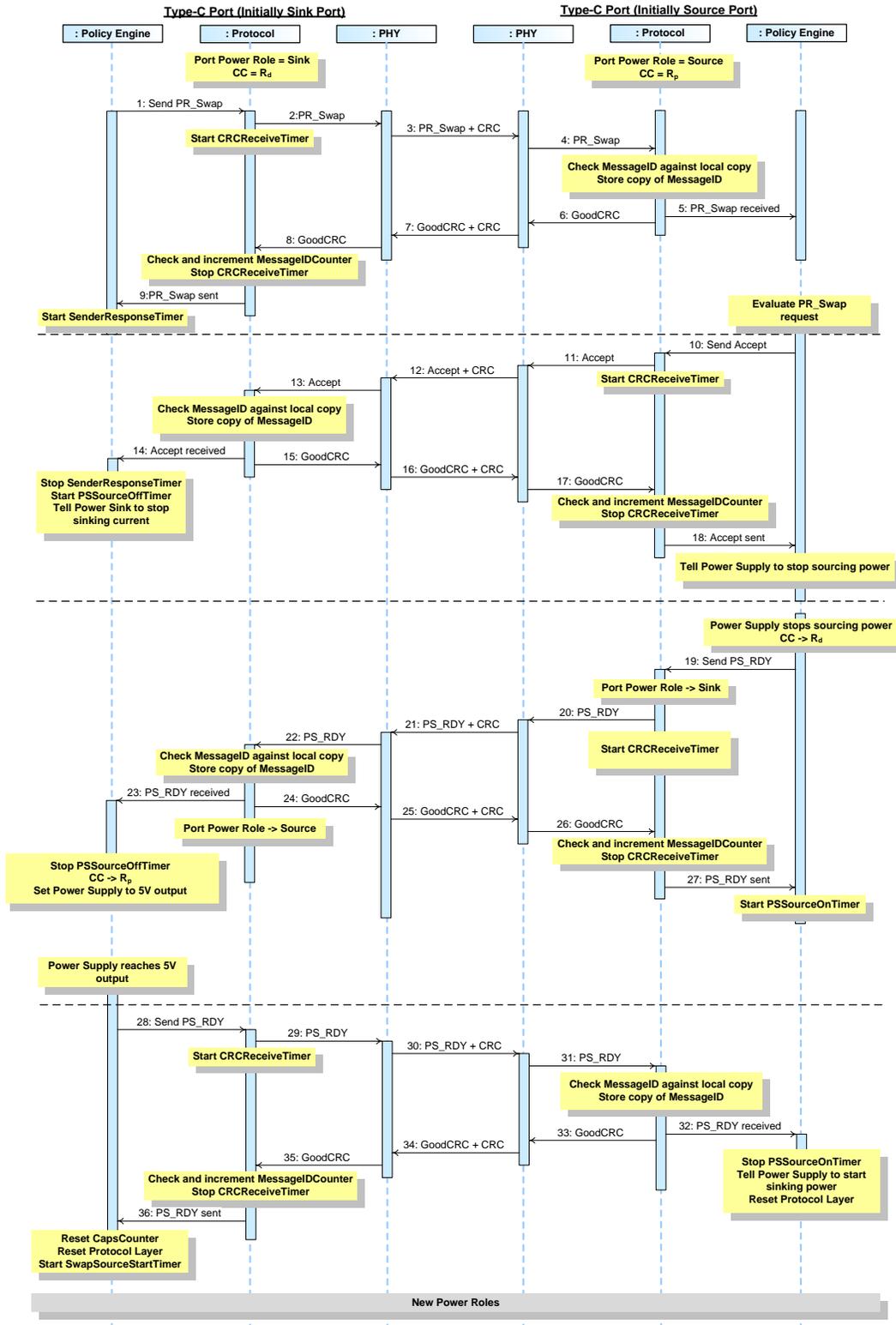


Table 8-16 Steps for a Successful USB Type-C Sink Initiated Power Role Swap Sequence

Step	Type-C Port (initially Sink Port)	Type-C Port (initially Source Port)
1	The [USB Type-C 1.2] Port has Port Power Role set to Sink with the R _d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a PR_Swap Message.	The [USB Type-C 1.2] Port has Port Power Role set to Source and the R _p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer. Starts CRCReceiveTimer .	
3	Physical Layer appends CRC and sends the PR_Swap Message.	Physical Layer receives the PR_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the PR_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID 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 PR_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the PR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the PR_Swap 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 Accept Message.
11		Protocol Layer creates the Message and passes to Physical Layer. Starts CRCReceiveTimer .
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Accept Message.	Physical Layer appends a CRC and sends the Accept Message.
13	Protocol Layer checks the MessageID 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 PR_Swap Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer , starts the PSSourceOffTimer and tells the power supply to stop sinking current.	
15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the GoodCRC 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 Message 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 Message 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.
31		Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.

Step	Type-C Port (initially Sink Port)	Type-C Port (initially Source 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. 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 USB Type-C connector.

Figure 8-17 shows an example sequence between a *[USB Type-C 1.2]* DRP, which is initially a UFP (Device) and a Sink (R_d asserted), and a *[USB Type-C 1.2]* 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 USB Type-C Data Role Swap, UFP operating as Sink initiates

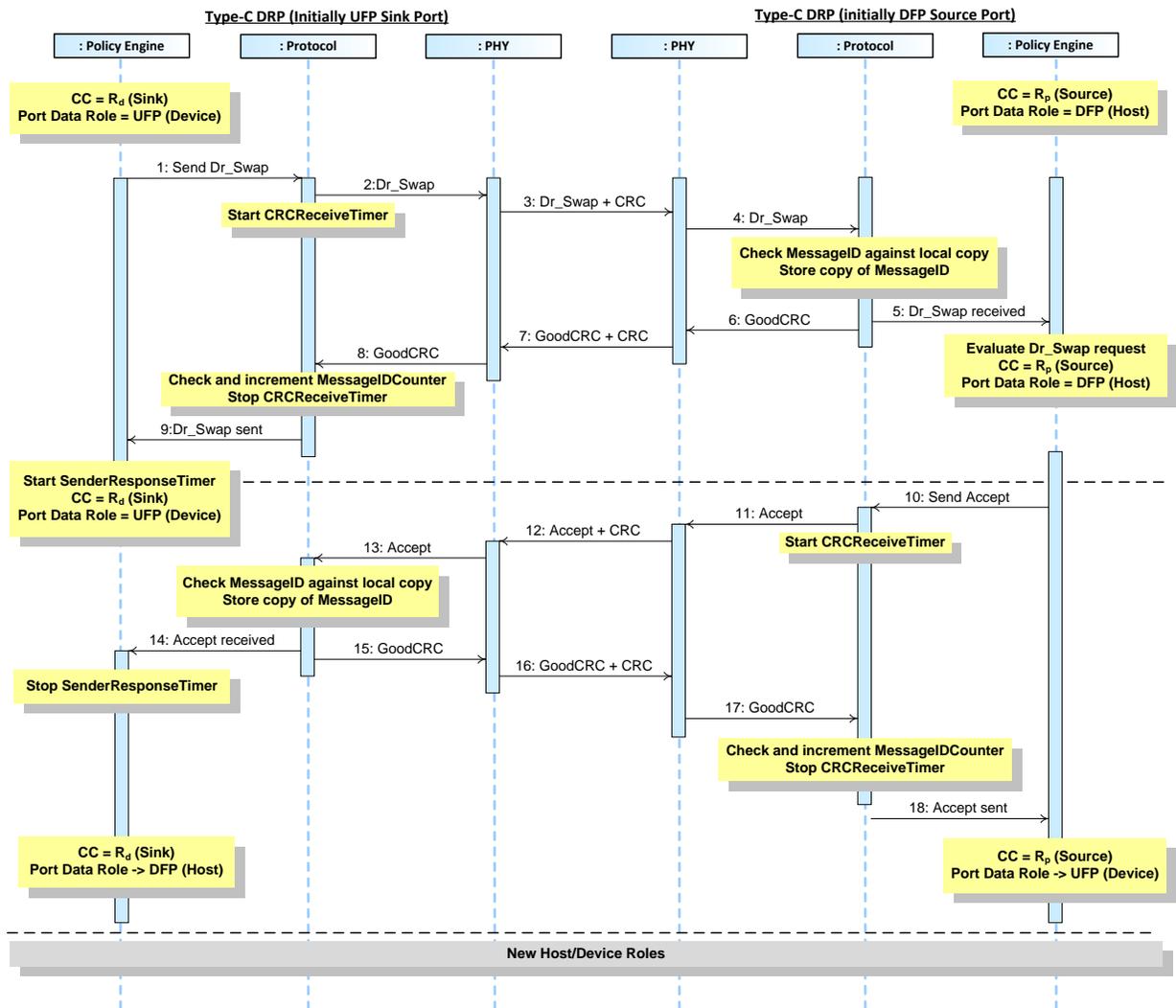


Table 8-11 below provides a detailed explanation of what happens at each labeled step in Figure 8-17 above.

Table 8-17 Steps for USB 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> .	
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 DRP (initially UFP Sink Port)	Type-C DRP (initially DFP Source Port)
18	<p>The Policy Engine requests that Data Role is changed from UFP (Device) to DFP (Host). The Power Delivery role is now a DFP (Host), with Port Data Role set to DFP, still operating as a Sink (R_d asserted).</p>	<p>Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer. Protocol Layer informs the Policy Engine that the Accept Message was successfully sent. The Policy Engine requests that the Data Role is changed to UFP (Device), with Port Data Role set to UFP and continues supplying power as a Source (R_p asserted).</p>
	<p>The Data Role Swap is complete, the data roles have been reversed while maintaining the direction of power flow.</p>	

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 USB Type-C connector.

Figure 8-18 shows an example sequence between a [USB Type-C 1.2] DRP, which is initially a UFP (Device) and a Source (R_p asserted), and a [USB Type-C 1.2] 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 USB Type-C Data Role Swap, UFP operating as Source initiates

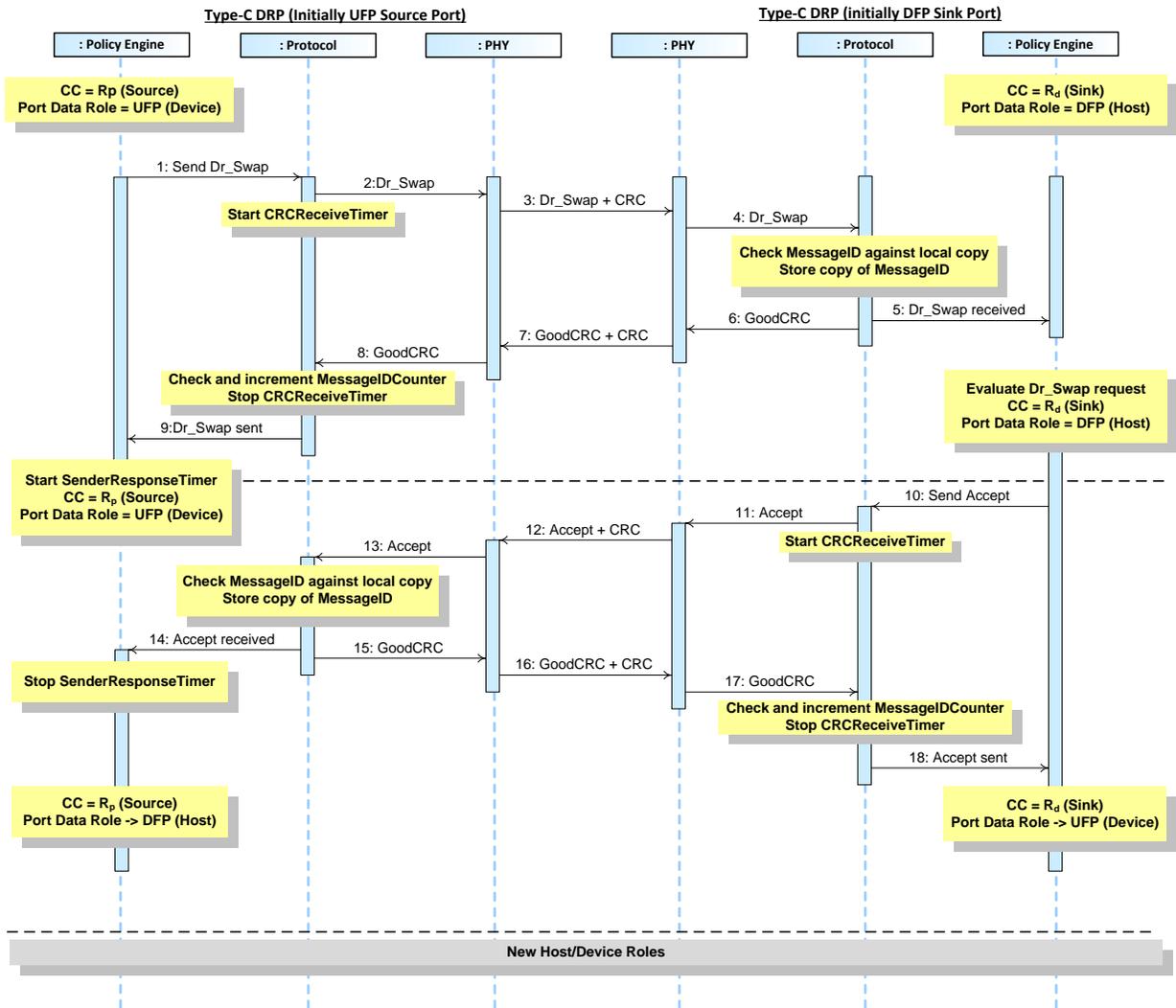


Table 8-18 below provides a detailed explanation of what happens at each labeled step in Figure 8-18 above.

Table 8-18 Steps for USB 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_a 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> .	
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 DRP (initially UFP Source Port)	Type-C DRP (initially DFP Sink Port)
18	<p>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_p asserted).</p>	<p>Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer. Protocol Layer informs the Policy Engine that the Accept Message was successfully sent. The Policy Engine requests that the Data Role is changed to UFP (Device), with Port Data Role set to UFP and still operating as a Sink (R_p asserted).</p>
<p>The Data Role Swap is complete, the data roles have been reversed while maintaining the direction of power flow.</p>		

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 USB Type-C connector.

Figure 8-19 shows an example sequence between a *[USB Type-C 1.2]* DRP, which is initially a UFP (Device) and a Sink (R_d asserted), and a *[USB Type-C 1.2]* 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 USB Type-C Data Role Swap, DFP operating as Source initiates

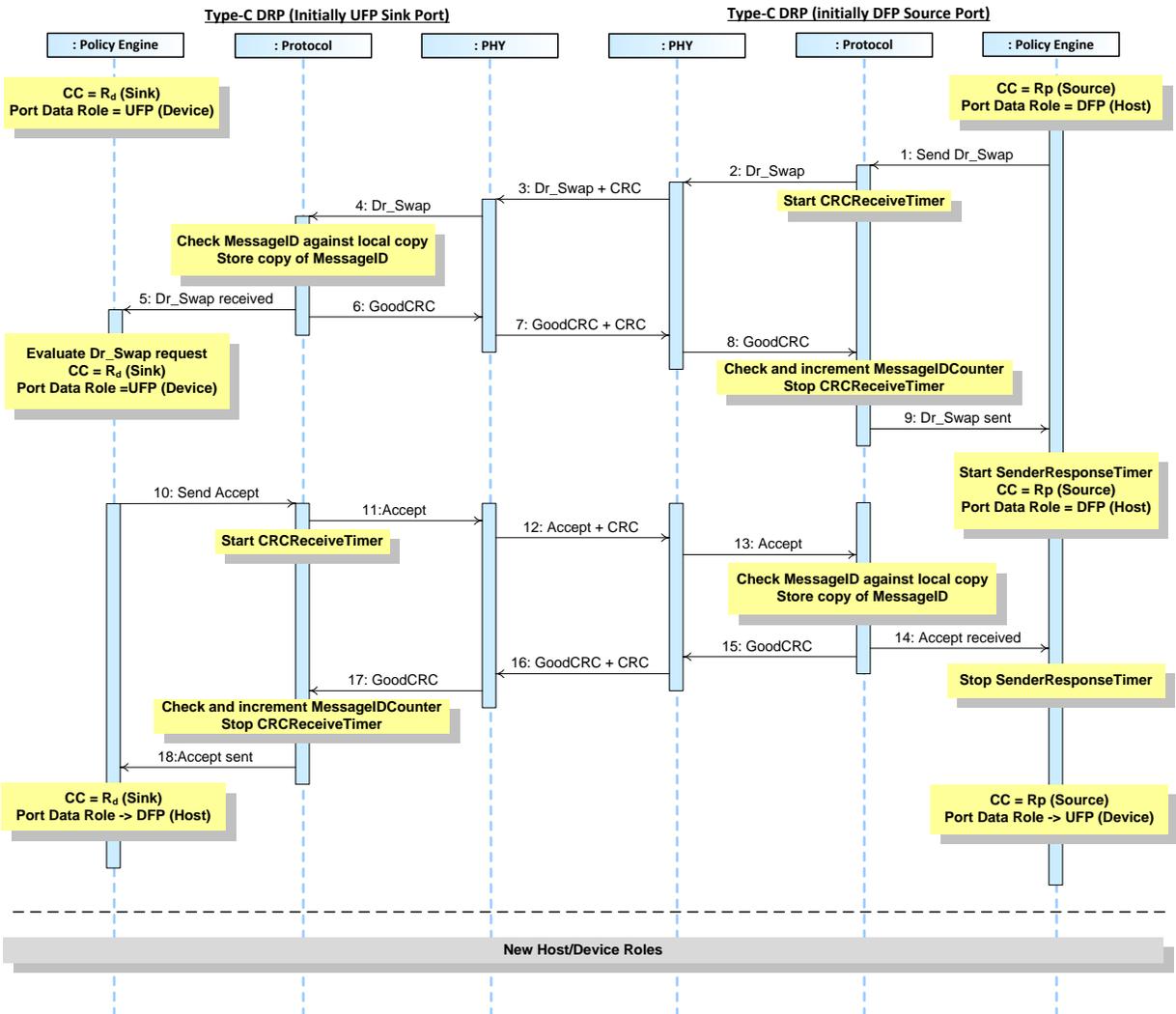


Table 8-19 below provides a detailed explanation of what happens at each labeled step in Figure 8-19 above.

Table 8-19 Steps for USB 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> .
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.
17	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	

Step	Type-C DRP (initially UFP Sink Port)	Type-C DRP (initially DFP 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 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).	The Policy Engine 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).
	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 USB Type-C connector.

Figure 8-20 shows an example sequence between a [\[USB Type-C 1.2\]](#) DRP, which is initially a UFP (Device) and a Source (R_p asserted), and a [\[USB Type-C 1.2\]](#) 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 USB Type-C Data Role Swap, DFP operating as Sink initiates

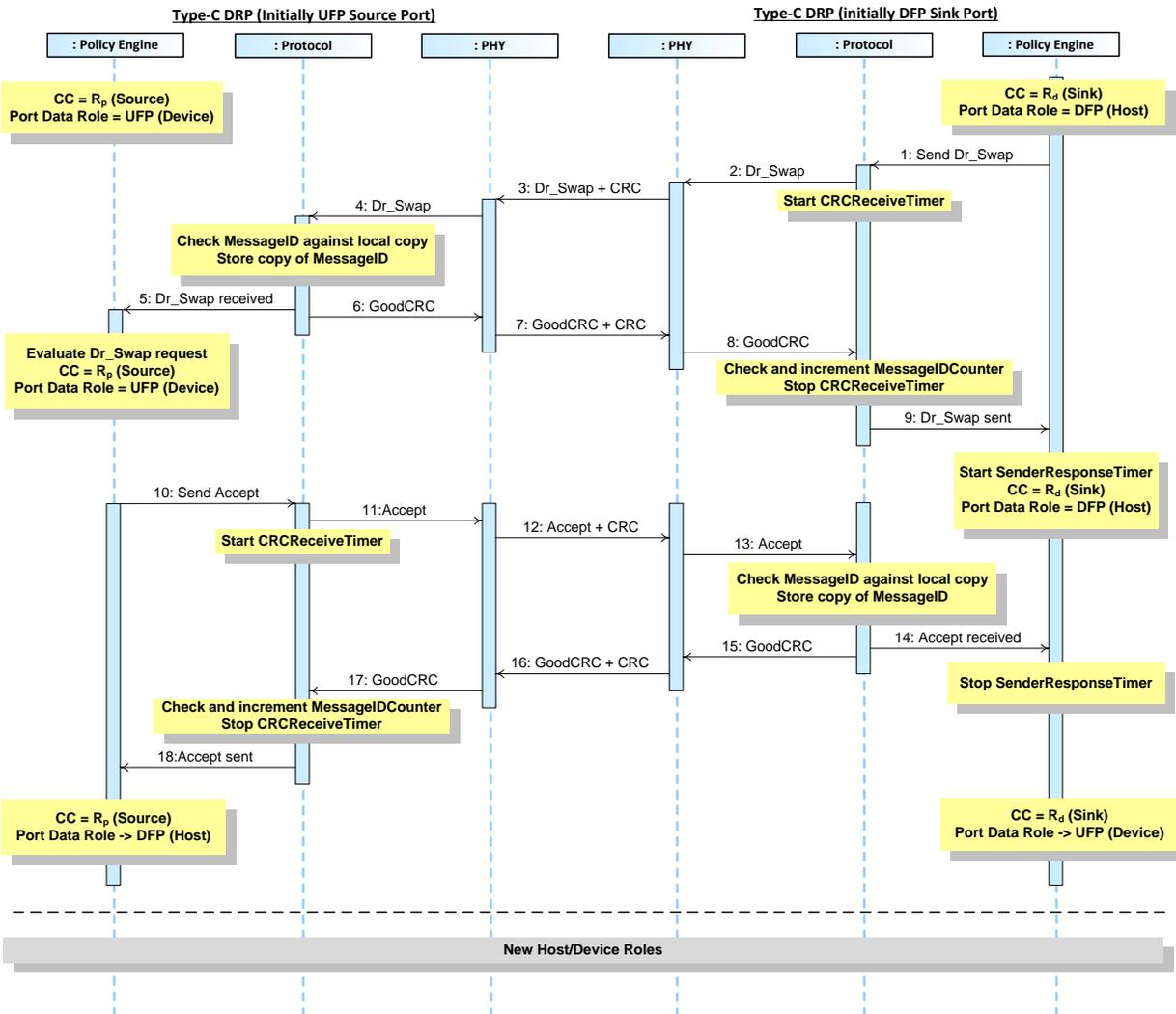


Table 8-20 below provides a detailed explanation of what happens at each labeled step in Figure 8-20 above.

Table 8-20 Steps for USB 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_a 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> .
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.
17	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	

Step	Type-C DRP (initially UFP Source Port)	Type-C DRP (initially DFP 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 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).	The Policy Engine 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).
The Data Role Swap is complete, the data roles have been reversed while maintaining the direction of power flow.		

8.3.2.8.3 VCONN SWAP

8.3.2.8.3.1 Type-C Source to Sink VCONN Source Swap

This is an example of Message sequence between a Port Pair both utilizing the USB Type-C connector.

Figure 8-21 shows an example sequence between a USB Type-C Source and Sink, where the Source is initially supplying VCONN and then tells the Sink to supply VCONN. During the process the Port Partners, keep their role as Source or Sink, maintain their operation as either a Source or a Sink (power remains constant) but exchange the VCONN Source from the Source to the Sink.

Figure 8-21 USB Type-C Source to Sink VCONN Source Swap

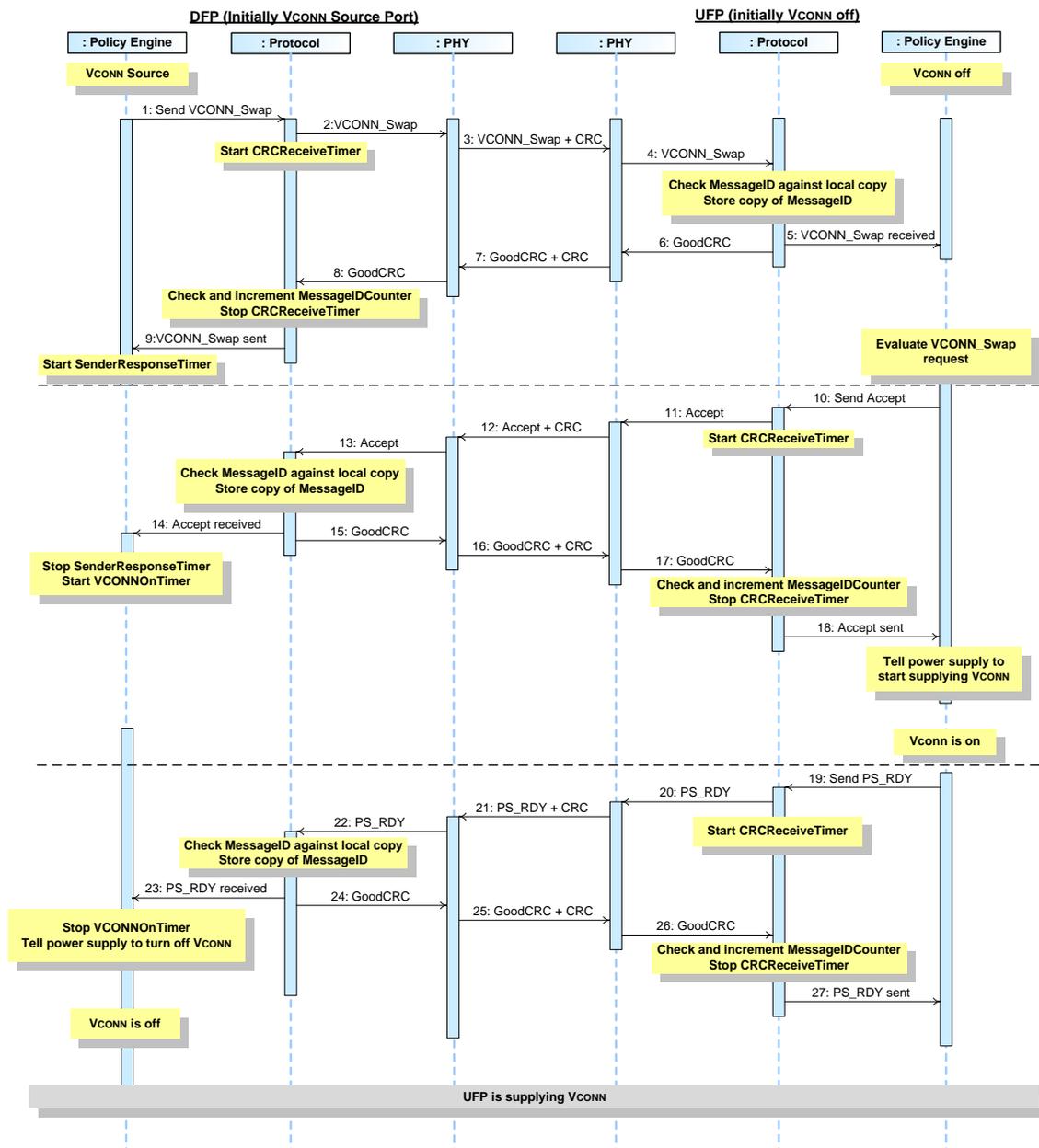


Table 8-21 below provides a detailed explanation of what happens at each labeled step in Figure 8-21 above.

Table 8-21 Steps for USB Type-C Source to Sink VCONN Source Swap

Step	Source (initially VCONN Source)	Sink (Initially VCONN off)
1	The Source starts as the VCONN Source. The Policy Engine directs the Protocol Layer to send a <i>VCONN_Swap</i> Message.	The Sink 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 Source 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	Source (Initially VCONN Source)	Sink (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 Source 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 Sink is now the VCONN Source and the Source has VCONN turned off.	

8.3.2.8.3.2 Type-C Sink to Source VCONN Source Swap

This is an example of Message sequence between a Port Pair both utilizing the USB Type-C connector.

Figure 8-22 shows an example sequence between a USB Type-C Source and Sink, where the Sink is initially supplying VCONN and then the Source tells the Sink that it will become the VCONN Source. During the process the Port Partners, keep their role as Source or Sink, maintain their operation as either a Source or a Sink (power remains constant) but exchange the VCONN Source from the Sink to the Source.

Figure 8-22 USB Type-C Sink to Source VCONN Source Swap

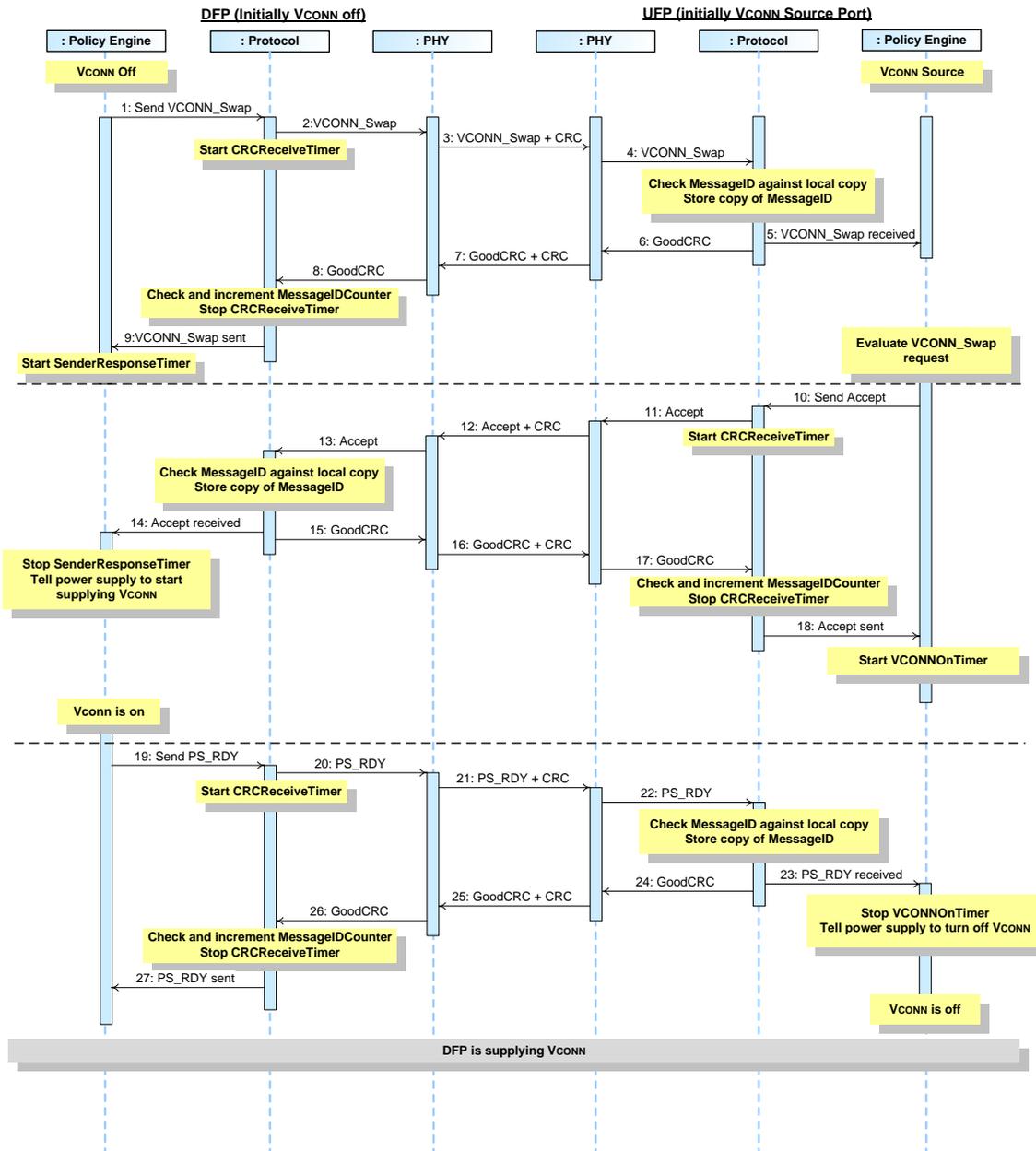


Table 8-22 below provides a detailed explanation of what happens at each labeled step in Figure 8-22 above.

Table 8-22 Steps for USB Type-C Sink to Source VCONN Source Swap

Step	Source	Sink
1	The Source starts with VCONN off. The Policy Engine directs the Protocol Layer to send a <i>VCONN_Swap</i> Message.	The Sink 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 Source 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	Source	Sink
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>VCONNOnTimer</i> .
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 Sink 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 Source is now the VCONN Source and the Sink has VCONN turned off.	

8.3.2.8.4 Getting Capabilities

8.3.2.8.4.1 Sink Gets Source Capabilities

Figure 8-23 shows an example sequence between a Source and a Sink when the Sink gets the Source's capabilities.

Figure 8-23 Sink Gets Source's Capabilities

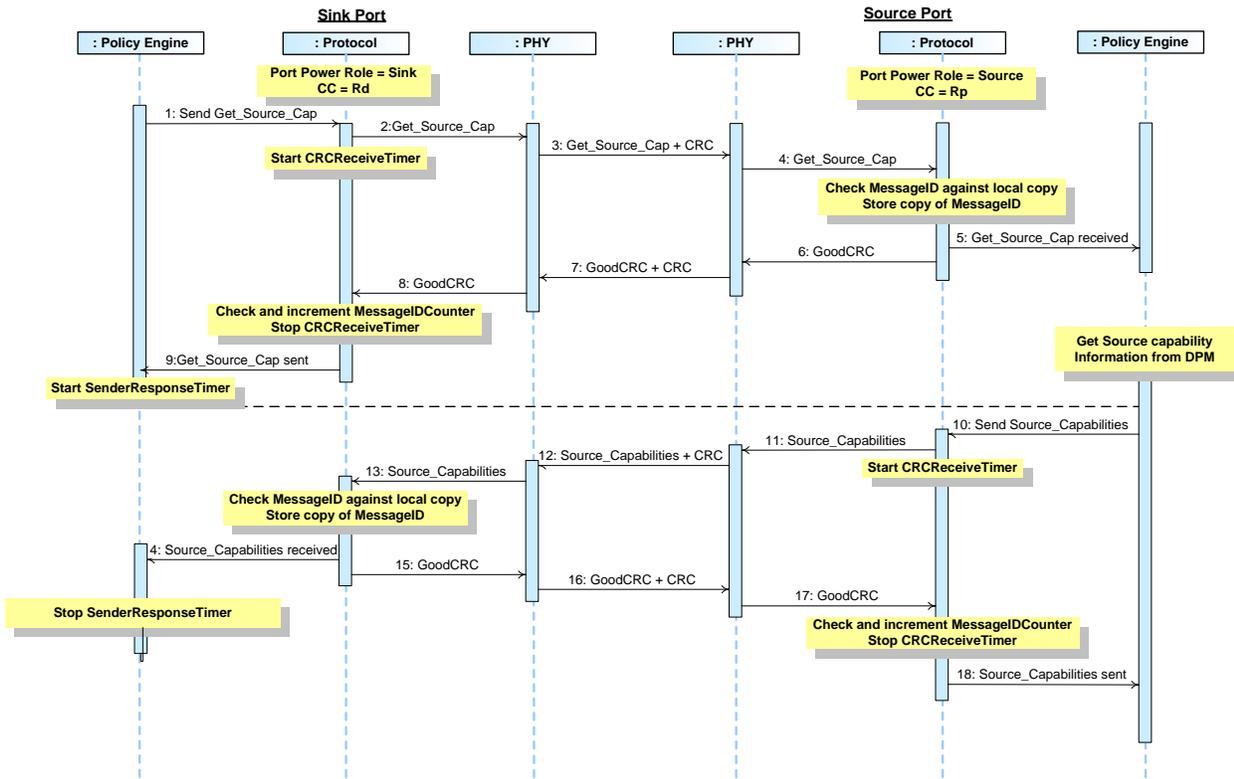


Table 8-23 below provides a detailed explanation of what happens at each labeled step in Figure 8-23 above.

Table 8-23 Steps for a Sink getting Source capabilities Sequence

Step	Sink Port	Source Port
1	The Port has <i>Port Power Role</i> set to Sink with the Rd pull down on its CC wire. Policy Engine directs the Protocol Layer to send a <i>Get_Source_Cap</i> Message.	The Port has <i>Port Power Role</i> set to Source and the Rp 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>Get_Source_Cap</i> Message.	Physical Layer receives the <i>Get_Source_Cap</i> Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>Get_Source_Cap</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>Get_Source_Cap</i> Message information to the Policy Engine that consumes it.

Step	Sink Port	Source Port
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>Get_Source_Cap</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .	
10		Policy Engine requests the DPM for the present Source capabilities which are provided. The Policy Engine tells the Protocol Layer to form a <i>Source_Capabilities</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>Source_Capabilities</i> Message.	Physical Layer appends a CRC and sends the <i>Source_Capabilities</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>Source_Capabilities</i> Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
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.
	The Source has informed the Sink of its capabilities.	

8.3.2.8.4.2 Dual-Role Source Gets Source Capabilities from a Dual-Role Sink

Figure 8-24 shows an example sequence between a Dual-Role Source and a Dual-Role Sink when the Source gets the Sink’s capabilities as a Source.

Figure 8-24 Dual-Role Source Gets Dual-Role Sink’s Capabilities as a Source

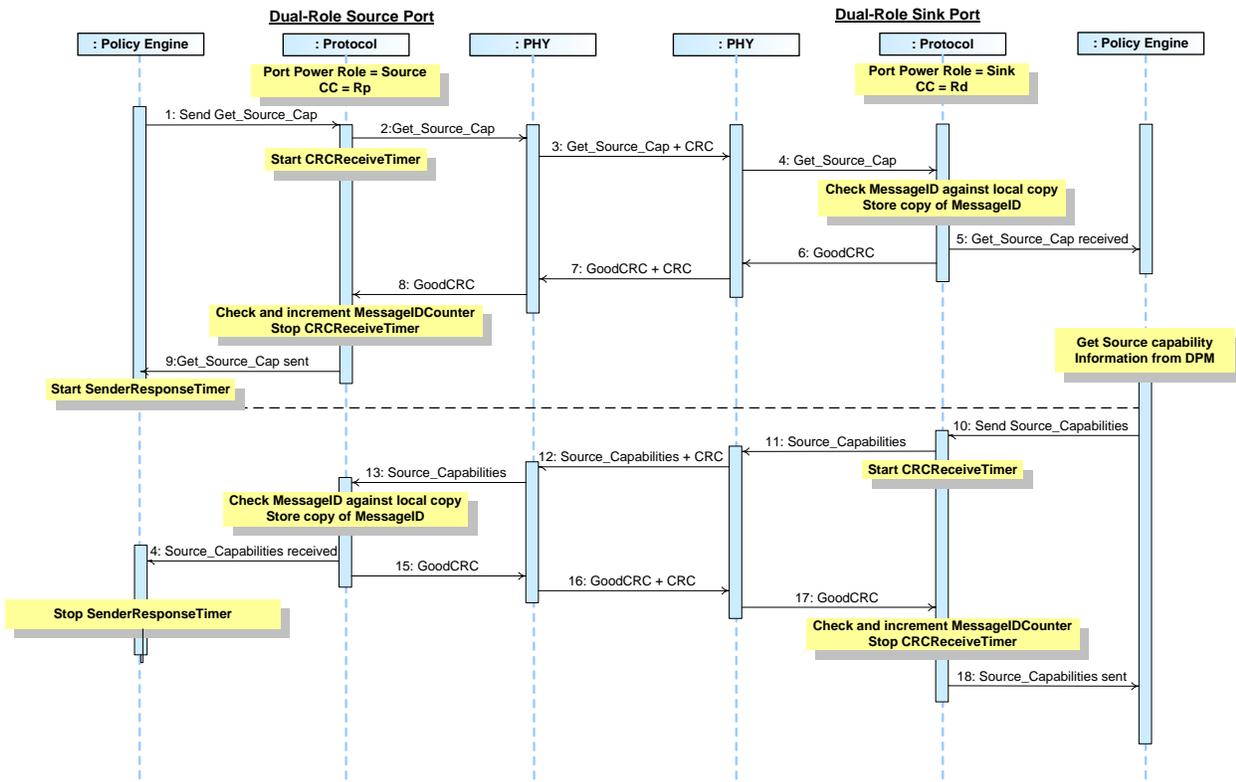


Table 8-24 below provides a detailed explanation of what happens at each labeled step in Figure 8-24 above.

Table 8-24 Steps for a Dual-Role Source getting Dual-Role Sink’s capabilities as a Source Sequence

Step	Dual-Role Source Port	Dual-Role Sink Port
1	The Port has <i>Port Power Role</i> set to Source and the Rp pull up on its CC wire. Policy Engine directs the Protocol Layer to send a <i>Get_Source_Cap</i> Message.	The Port has <i>Port Power Role</i> set to Sink with the Rd 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>Get_Source_Cap</i> Message.	Physical Layer receives the <i>Get_Source_Cap</i> Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>Get_Source_Cap</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>Get_Source_Cap</i> Message information to the Policy Engine that consumes it.

Step	Dual-Role Source Port	Dual-Role Sink Port
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>Get_Source_Cap</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .	
10		Policy Engine requests the DPM for the present Source capabilities which are provided. The Policy Engine tells the Protocol Layer to form a <i>Source_Capabilities</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>Source_Capabilities</i> Message.	Physical Layer appends a CRC and sends the <i>Source_Capabilities</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>Source_Capabilities</i> Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
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.
	The Dual-Role Sink has informed the Dual-Role Source of its capabilities.	

8.3.2.8.4.3 Source Gets Sink Capabilities

Figure 8-25 shows an example sequence between a Source and a Sink when the Source gets the Sink's capabilities.

Figure 8-25 Source Gets Sink's Capabilities

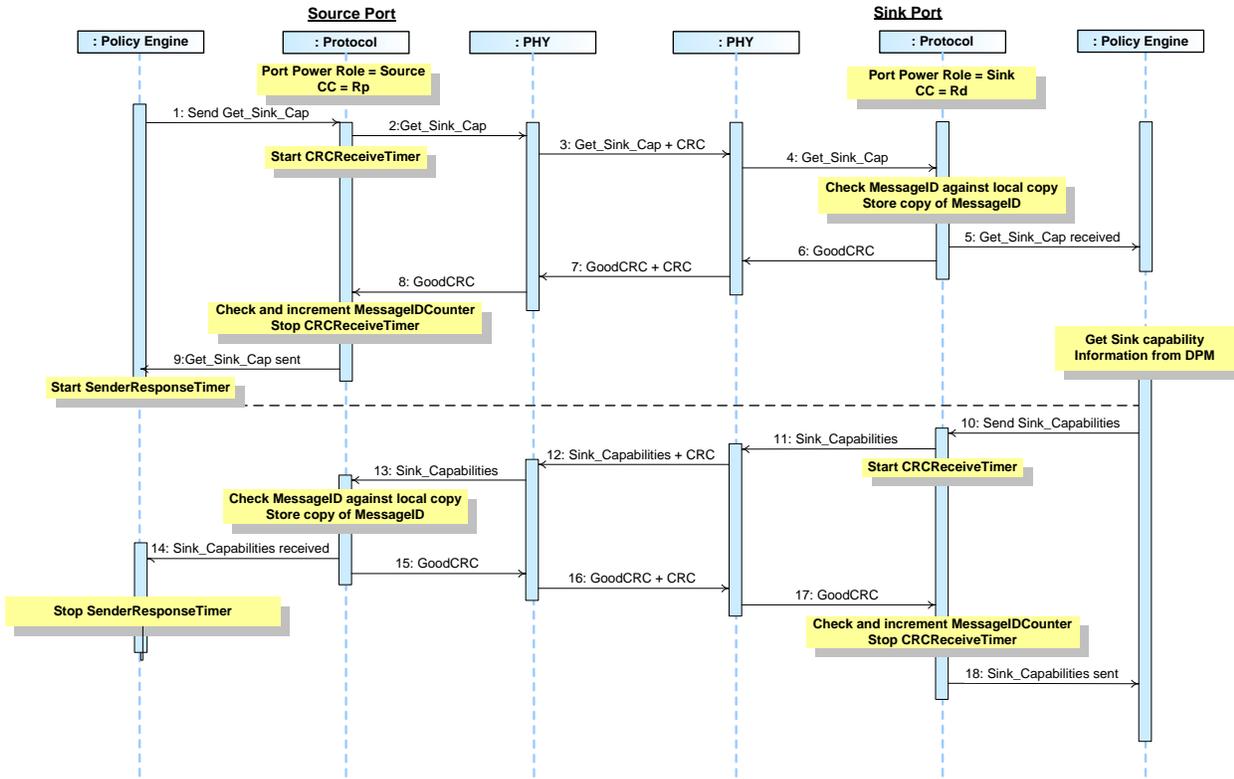


Table 8-25 below provides a detailed explanation of what happens at each labeled step in Figure 8-25 above.

Table 8-25 Steps for a Source getting Sink capabilities Sequence

Step	Source Port	Sink Port
1	The Port has <i>Port Power Role</i> set to Source and the Rp pull up on its CC wire. Policy Engine directs the Protocol Layer to send a <i>Get_Sink_Cap</i> Message.	The Port has <i>Port Power Role</i> set to Sink with the Rd 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>Get_Sink_Cap</i> Message.	Physical Layer receives the <i>Get_Sink_Cap</i> Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>Get_Sink_Cap</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>Get_Sink_Cap</i> Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.

Step	Source Port	Sink Port
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>Get_Sink_Cap</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .	
10		Policy Engine requests the DPM for the present Sink capabilities which are provided. The Policy Engine tells the Protocol Layer to form a <i>Sink_Capabilities</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>Sink_Capabilities</i> Message.	Physical Layer appends a CRC and sends the <i>Sink_Capabilities</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>Sink_Capabilities</i> Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Sink_Capabilities</i> Message was successfully sent.
	The Sink has informed the Source of its capabilities.	

8.3.2.8.4.4 Dual-Role Sink Get Sink Capabilities from a Dual-Role Source

Figure 8-26 shows an example sequence between a Dual-Role Source and a Dual-Role Sink when the Dual-Role Sink gets the Dual-Role Source’s capabilities as a Sink.

Figure 8-26 Dual-Role Sink Gets Dual-Role Source’s Capabilities as a Sink

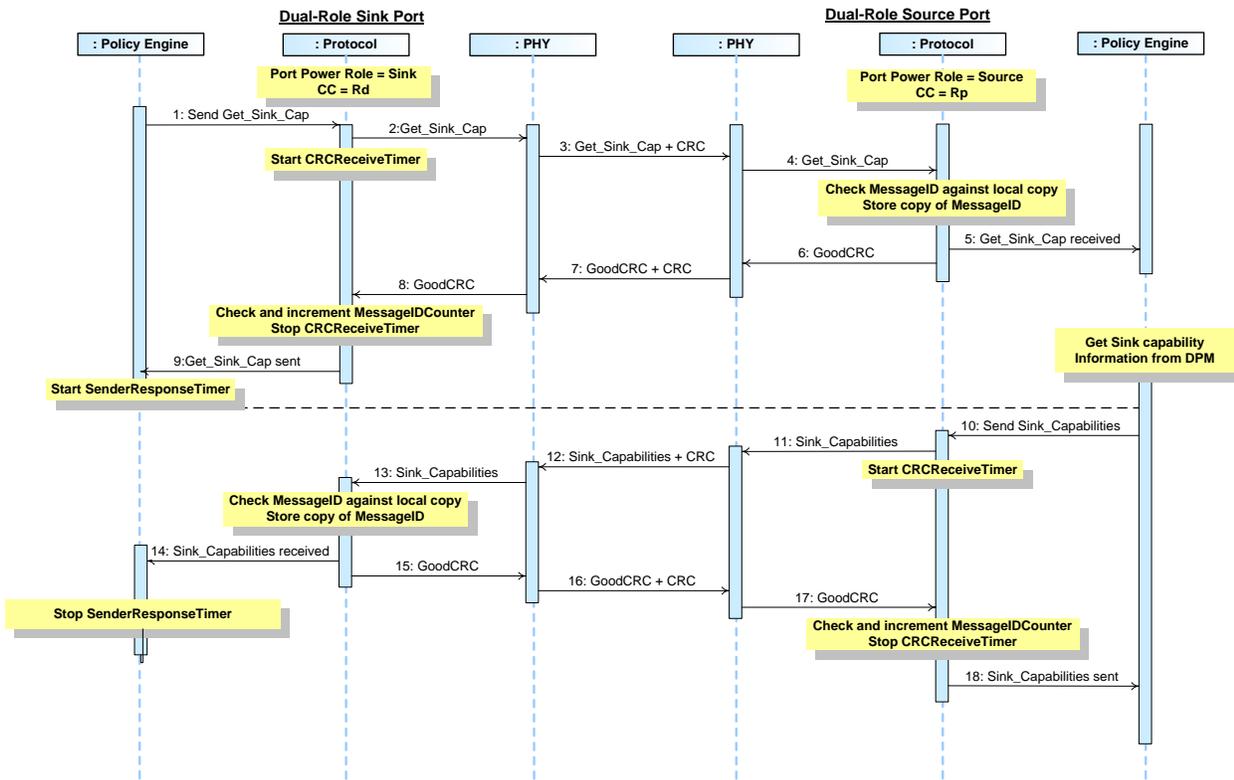


Table 8-26 below provides a detailed explanation of what happens at each labeled step in Figure 8-26 above.

Table 8-26 Steps for a Dual-Role Sink getting Dual-Role Source capabilities as a Sink Sequence

Step	Dual-Role Sink Port	Dual-Role Source Port
1	The Port has <i>Port Power Role</i> set to Dual-Role Sink with the Rd pull down on its CC wire. Policy Engine directs the Protocol Layer to send a <i>Get_Sink_Cap</i> Message.	The Port has <i>Port Power Role</i> set to Dual-Role Source and the Rp 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>Get_Sink_Cap</i> Message.	Physical Layer receives the <i>Get_Sink_Cap</i> Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>Get_Sink_Cap</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>Get_Sink_Cap</i> Message information to the Policy Engine that consumes it.

Step	Dual-Role Sink Port	Dual-Role Source Port
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>Get_Sink_Cap</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .	
10		Policy Engine requests the DPM for the present Dual-Role Source capabilities which are provided. The Policy Engine tells the Protocol Layer to form a <i>Sink_Capabilities</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>Sink_Capabilities</i> Message.	Physical Layer appends a CRC and sends the <i>Sink_Capabilities</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>Sink_Capabilities</i> Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Sink_Capabilities</i> Message was successfully sent.
	The Dual-Role Source has informed the Dual-Role Sink of its capabilities as a Sink.	

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 USB Type-C connector.

Figure 8-27 shows an example sequence between a USB 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 8-27 DFP to UFP Discover Identity

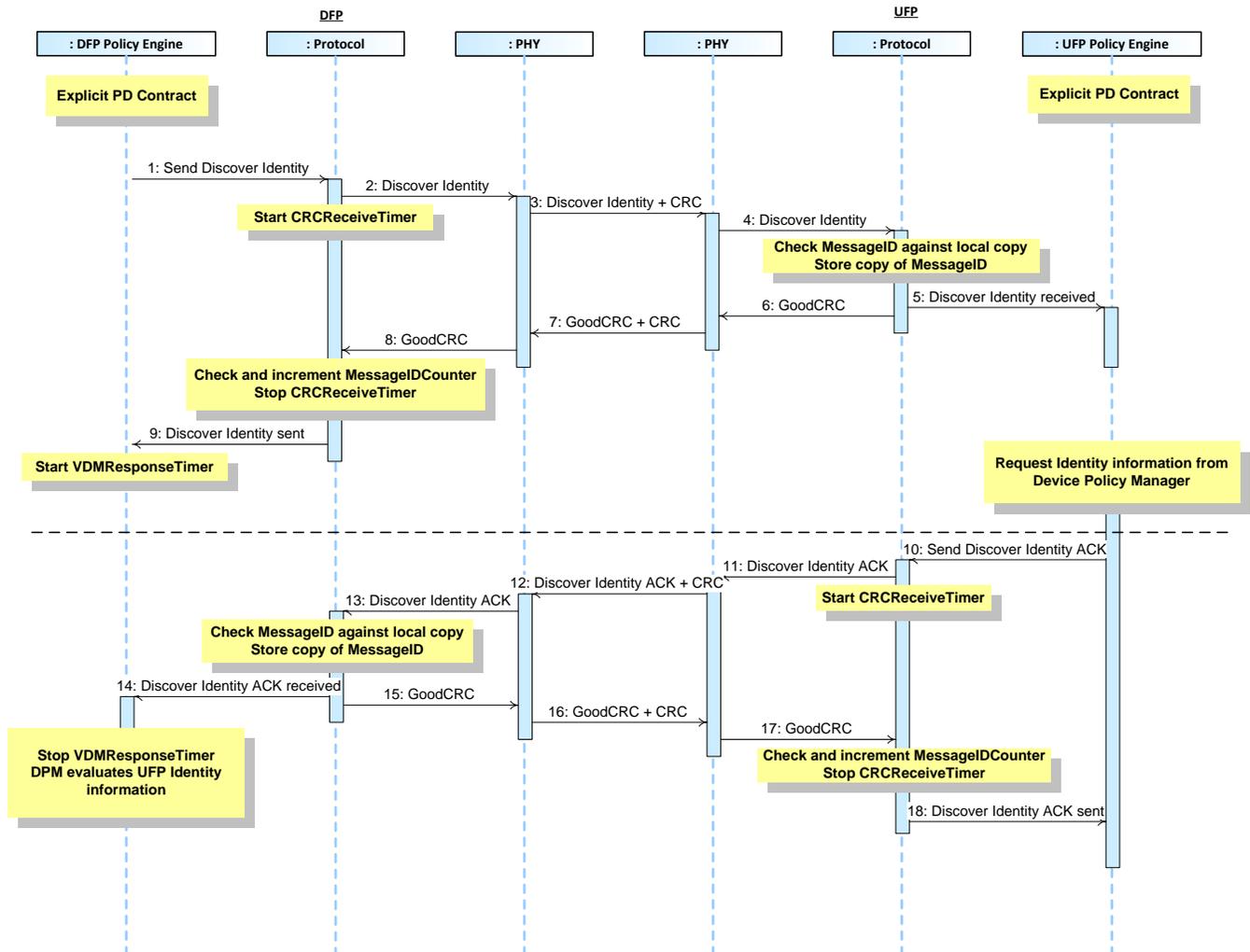


Table 8-27 below provides a detailed explanation of what happens at each labeled step in Figure 8-27 above.

Table 8-27 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 USB Type-C connector. Figure 8-28 shows an example sequence between a USB 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 8-28 Source Port to Cable Plug Discover Identity

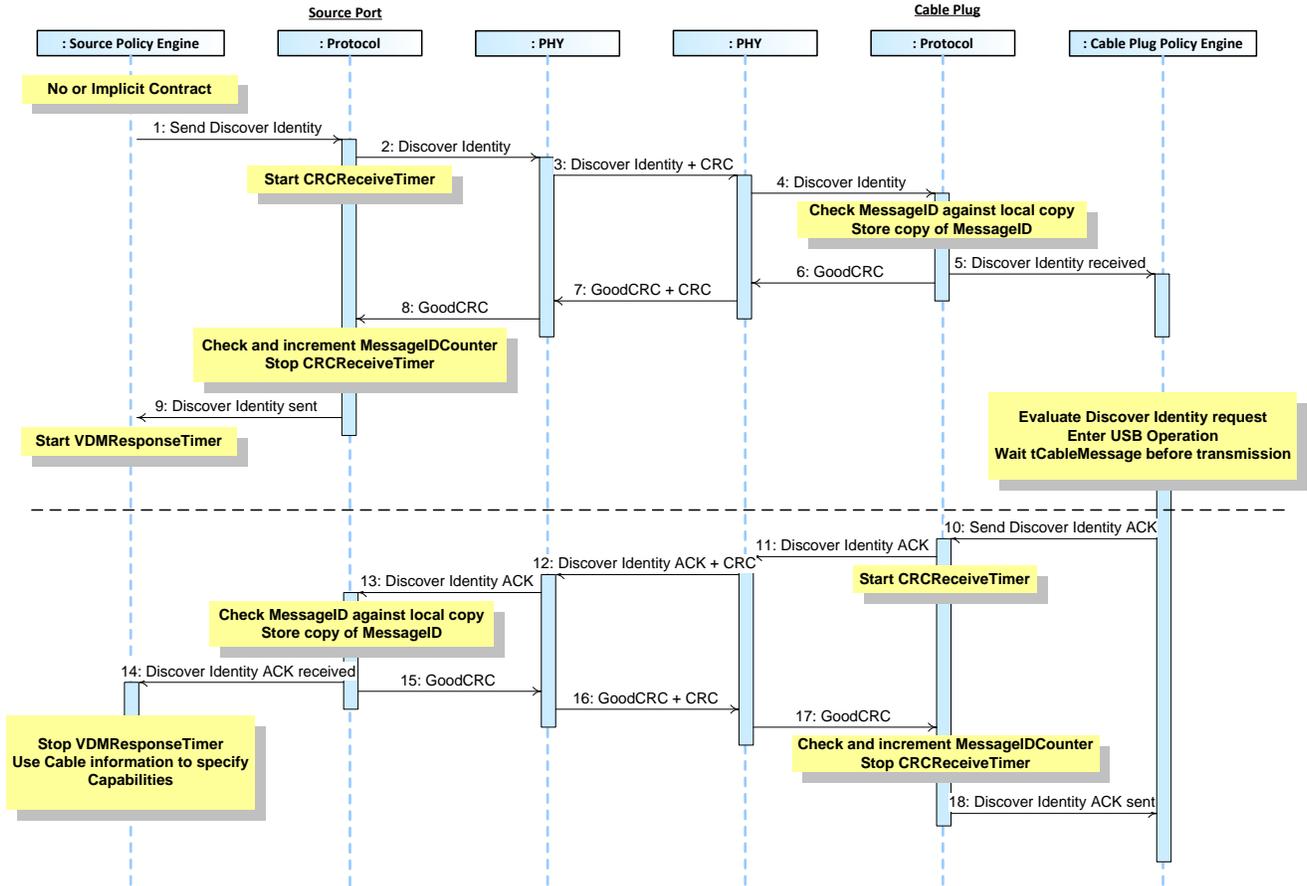


Table 8-28 below provides a detailed explanation of what happens at each labeled step in Figure 8-28 above.

Table 8-28 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</i> after the <i>GoodCRC</i> Message was sent 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 USB Type-C connector. Figure 8-29 shows an example sequence between a USB Type-C DFP and a Cable Plug, where the DFP attempts to discover identity information from the Cable Plug.

Figure 8-29 DFP to Cable Plug Discover Identity

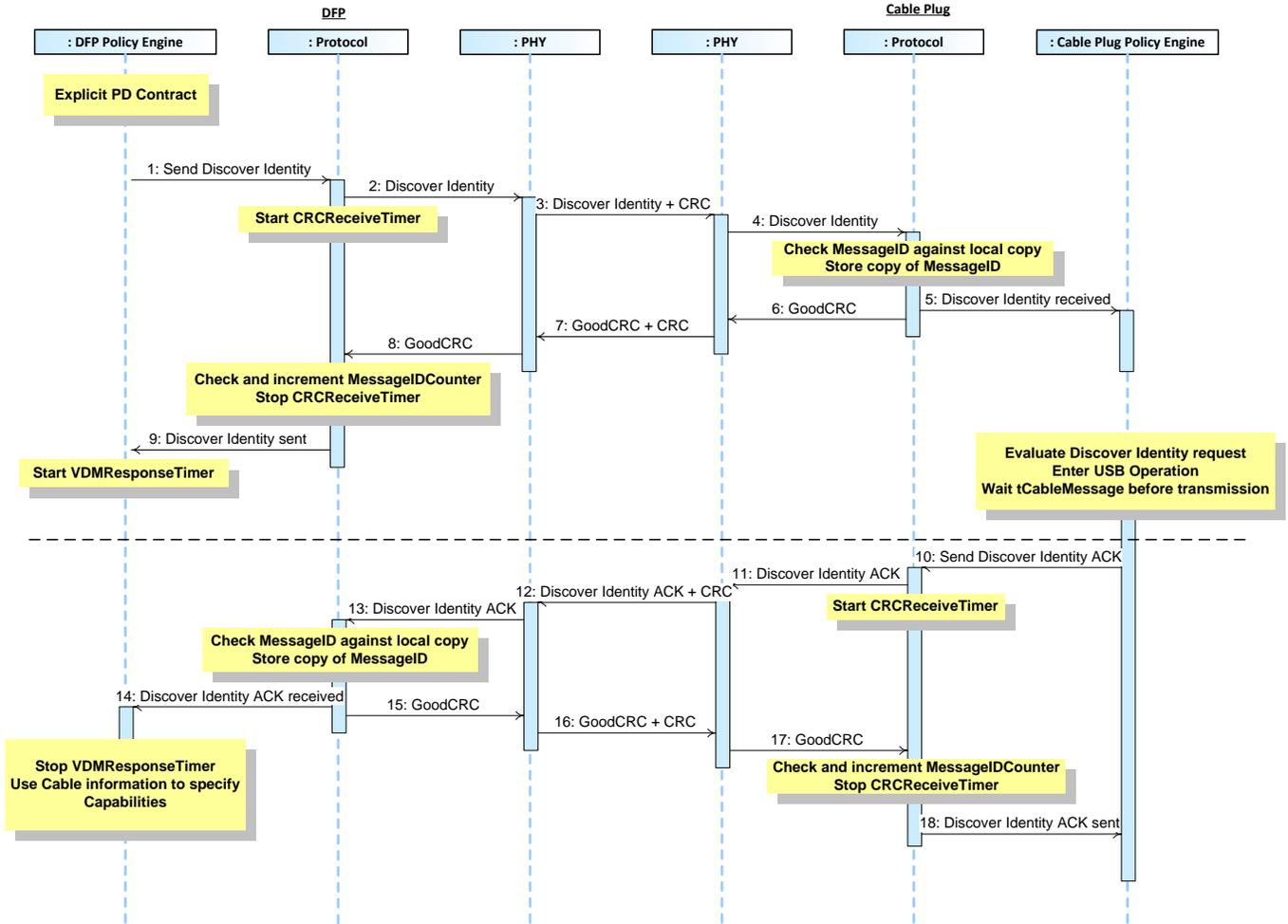


Table 8-29 below provides a detailed explanation of what happens at each labeled step in Figure 8-29 above.

Table 8-29 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> .	
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	DFP	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</i> after the <i>GoodCRC</i> Message was sent 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.
18	The DFP when acting as a DFP 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

Figure 8-30 shows an example sequence between a USB Type-C DFP and a UFP that occurs after the DFP has discovered supported SVIDs and Modes at which point it selects and enters a Mode.

Figure 8-30 DFP to UFP Enter Mode

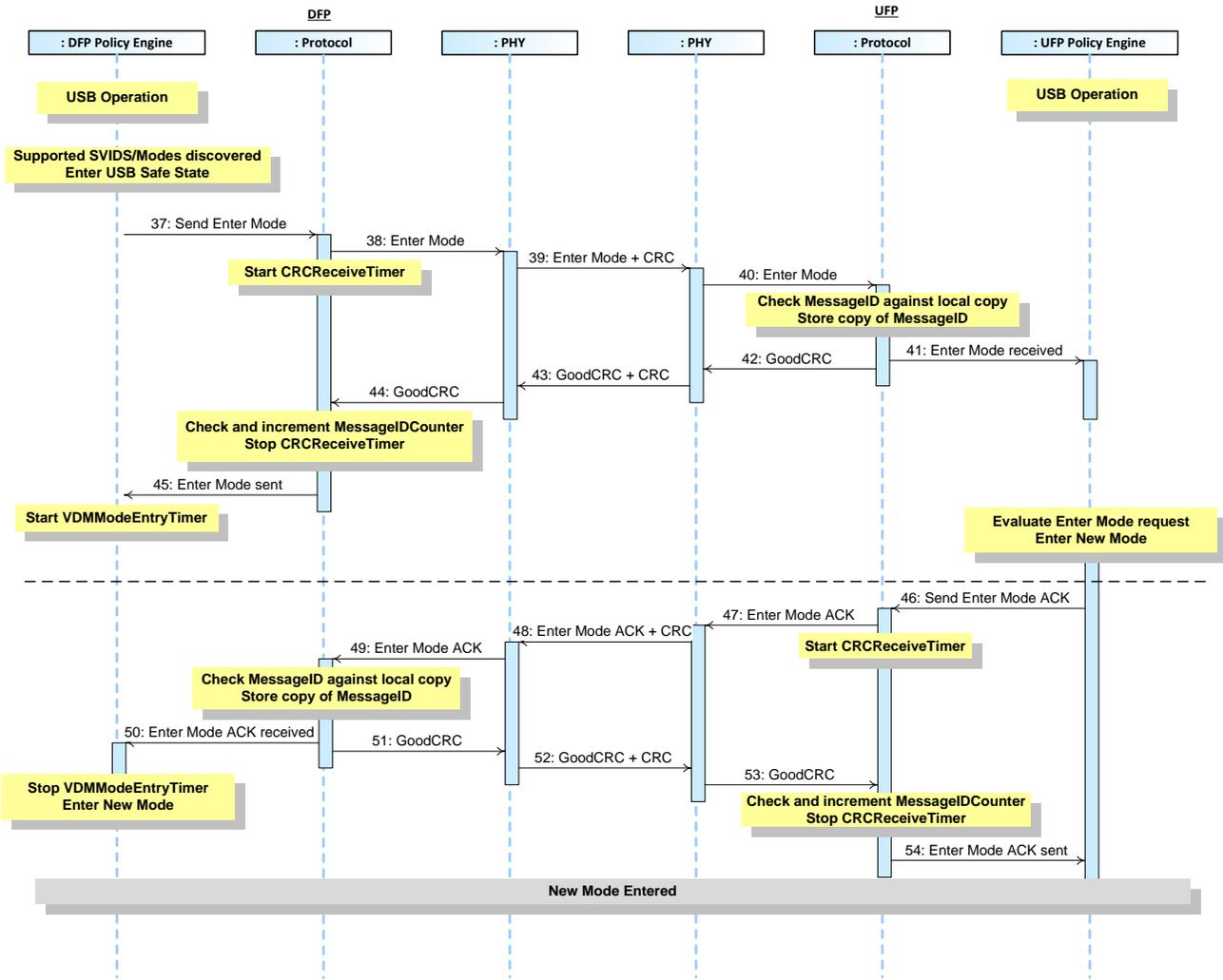


Table 8-30 below provides a detailed explanation of what happens at each labeled step in Table 8-30 above.

Table 8-30 Steps for DFP to UFP Enter Mode

Step	DFP	UFP
1	The DFP has an Explicit Contract The DFP has discovered the supported SVIDs using the <i>Discover SVIDs</i> Command request and the supported Modes using the <i>Discover Modes</i> Command request 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.	The UFP has an Explicit Contract.
2	Protocol Layer creates the <i>Enter Mode</i> Command request and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	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.
4		Physical Layer removes the CRC and forwards the <i>Enter 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>Enter 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>Enter Mode</i> Command request was successfully sent. Policy Engine starts the <i>VDMModeEntryTimer</i> .	
10		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.
11		Protocol Layer creates the <i>Enter Mode</i> Command ACK response and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	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.
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>Enter Mode</i> Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>VDMModeEntryTimer</i> and requests the Device Policy Manager to enter the new Mode.	

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>Enter Mode</i> Command ACK response was successfully sent.
	DFP and UFP are operating in the new Mode	

8.3.2.9.5 DFP to UFP Exit Mode

Figure 8-31 shows an example sequence between a USB Type-C DFP and a UFP, where the DFP commands the UFP to exit the only Active Mode.

Figure 8-31 DFP to UFP Exit Mode

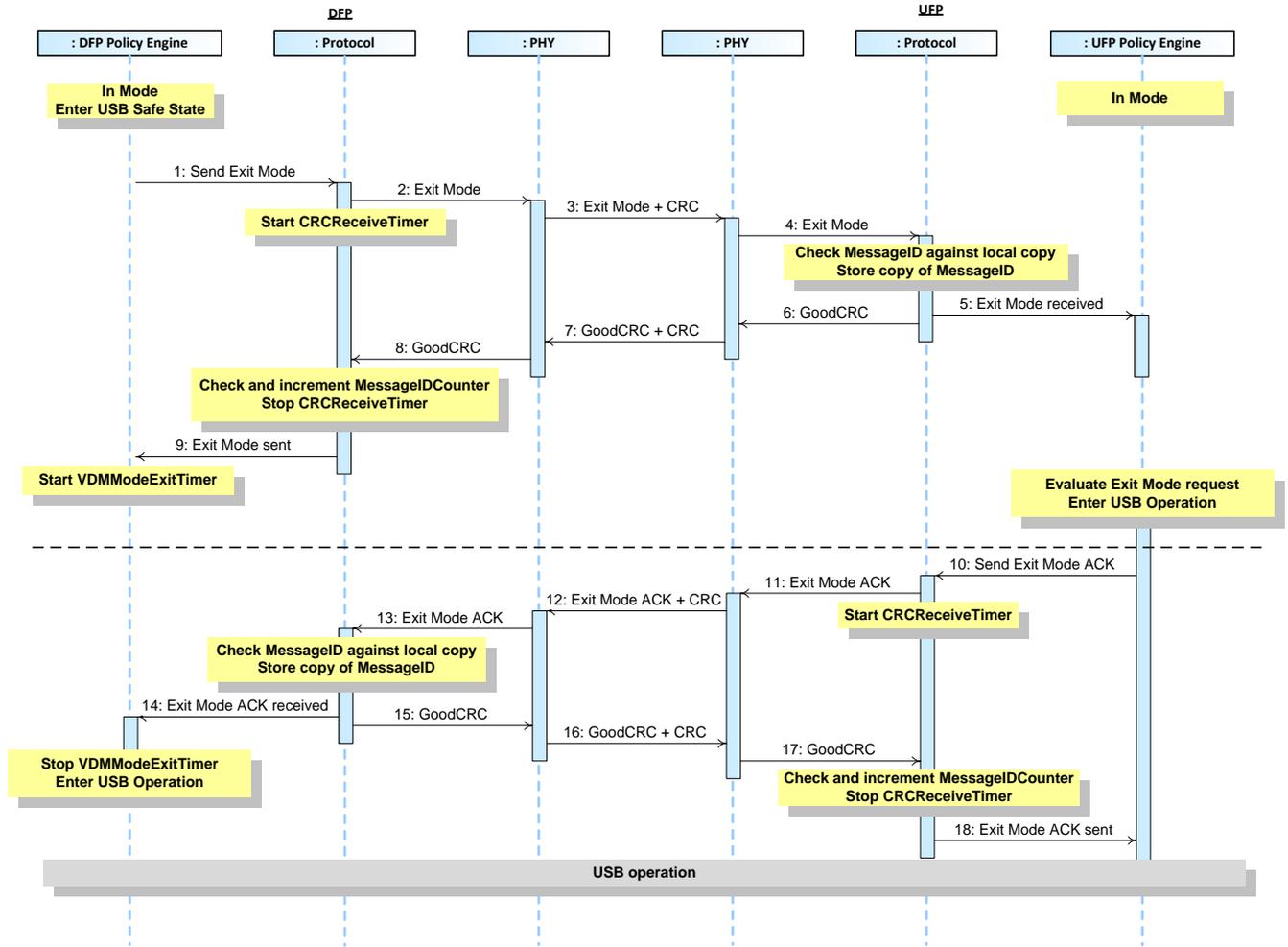


Table 8-31 below provides a detailed explanation of what happens at each labeled step in Figure 8-31 above.

Table 8-31 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 an <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.
4		Physical Layer removes the CRC and forwards the <i>Exit Mode</i> Command request to the Protocol Layer.

Step	DFP	UFP
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

Figure 8-32 shows an example sequence between a DFP and a Cable Plug that occurs after the DFP has discovered supported SVIDs and Modes at which point it selects and enters a Mode.

Figure 8-32 DFP to Cable Plug Enter Mode

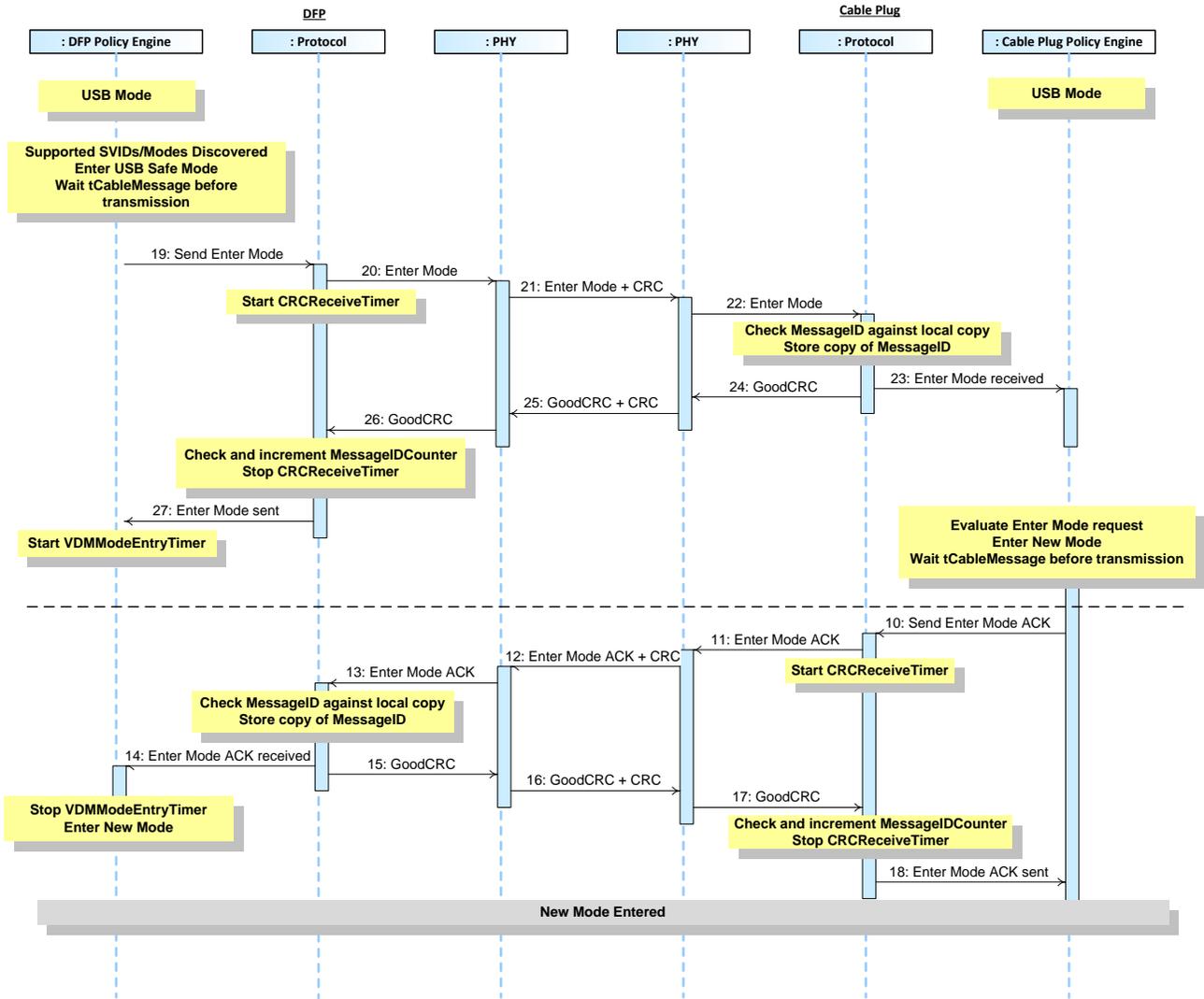


Table 8-32 below provides a detailed explanation of what happens at each labeled step in Figure 8-32 above.

Table 8-32 Steps for DFP to Cable Plug Enter Mode

Step	DFP	Cable Plug
1	The DFP has an Explicit Contract The DFP has discovered the supported SVIDs using the <i>Discover SVIDs</i> Command request and the supported Modes using the <i>Discover Modes</i> Command request The DFP goes to USB Safe State. The Device Policy Manager requests the Policy Engine to enter a Mode. <i>tCableMessage</i> after the last <i>GoodCRC</i> Message was sent the Policy Engine directs the Protocol Layer to send an <i>Enter Mode</i> Command request.	
2	Protocol Layer creates the <i>Enter Mode</i> Command request and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
3	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.
4		Physical Layer removes the CRC and forwards the <i>Enter 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>Enter 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>Enter Mode</i> Command request was successfully sent. Policy Engine starts the <i>VDMModeEntryTimer</i> .	
10		Policy Engine requests the Device Policy Manager to enter the new Mode. <i>tCableMessage</i> after the <i>GoodCRC</i> Message was sent the Policy Engine tells the Protocol Layer to form an <i>Enter Mode</i> Command ACK response.
11		Protocol Layer creates the <i>Enter Mode</i> Command ACK response and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
12	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.
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>Enter Mode</i> Command ACK response information to the Policy Engine that consumes it.	

Step	DFP	Cable Plug
14	The Policy Engine stops the <i>VDMModeEntryTimer</i> and requests the Device Policy Manager to enter the new Mode.	
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>Enter Mode</i> Command ACK response was successfully sent.
	DFP and Cable Plug 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 USB Type-C connector.

Figure 8-33 shows an example sequence between a USB Type-C DFP and a Cable Plug, where the DFP commands the Cable Plug to exit an Active Mode.

Figure 8-33 DFP to Cable Plug Exit Mode

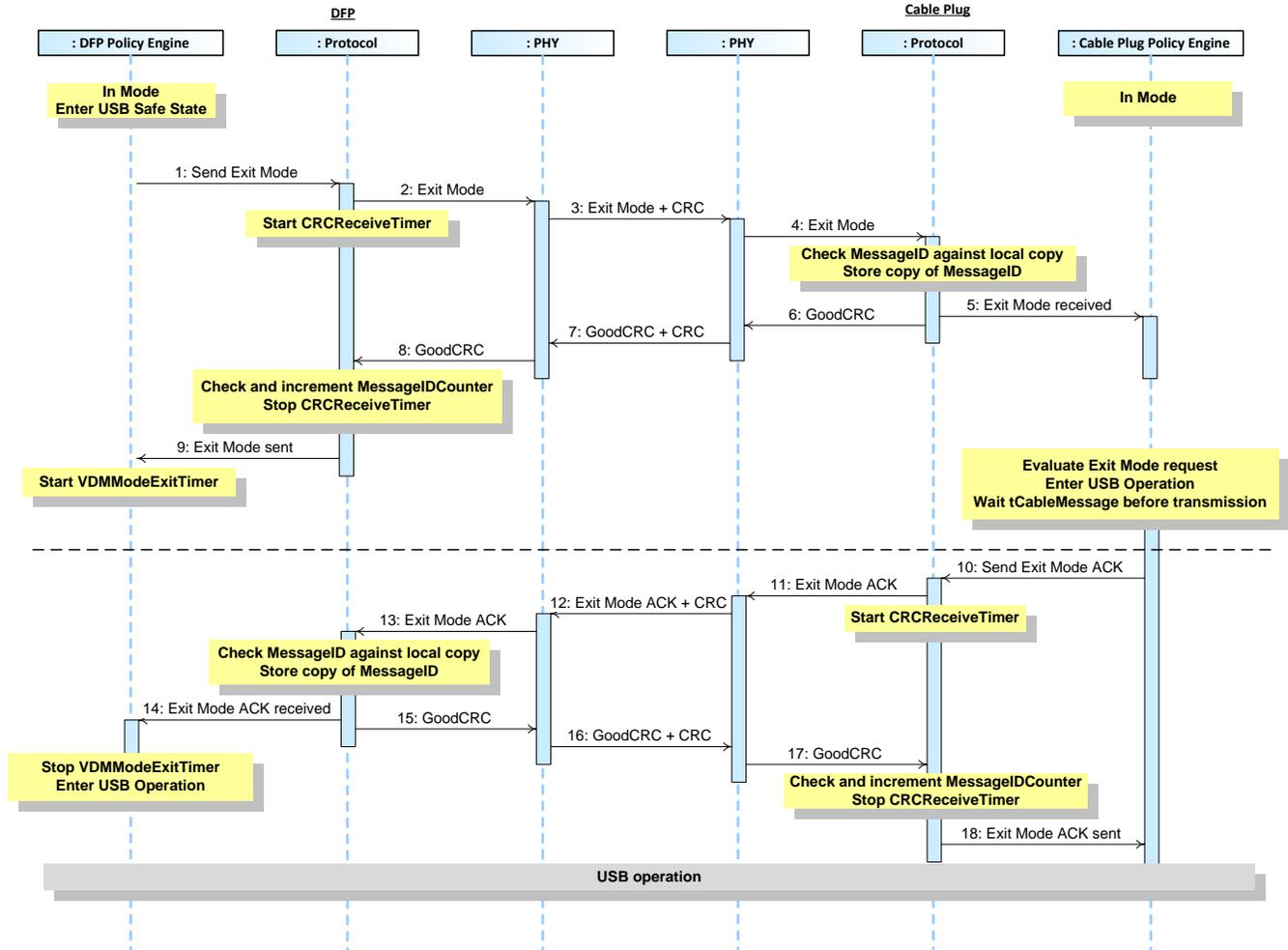


Table 8-33 below provides a detailed explanation of what happens at each labeled step in Figure 8-33 above.

Table 8-33 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 an <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</i> after the <i>GoodCRC</i> Message was sent 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 USB Type-C connector.

Figure 8-34 shows an example sequence between a USB Type-C DFP and a UFP, where the UFP requests attention from the DFP.

Figure 8-34 UFP to DFP Attention

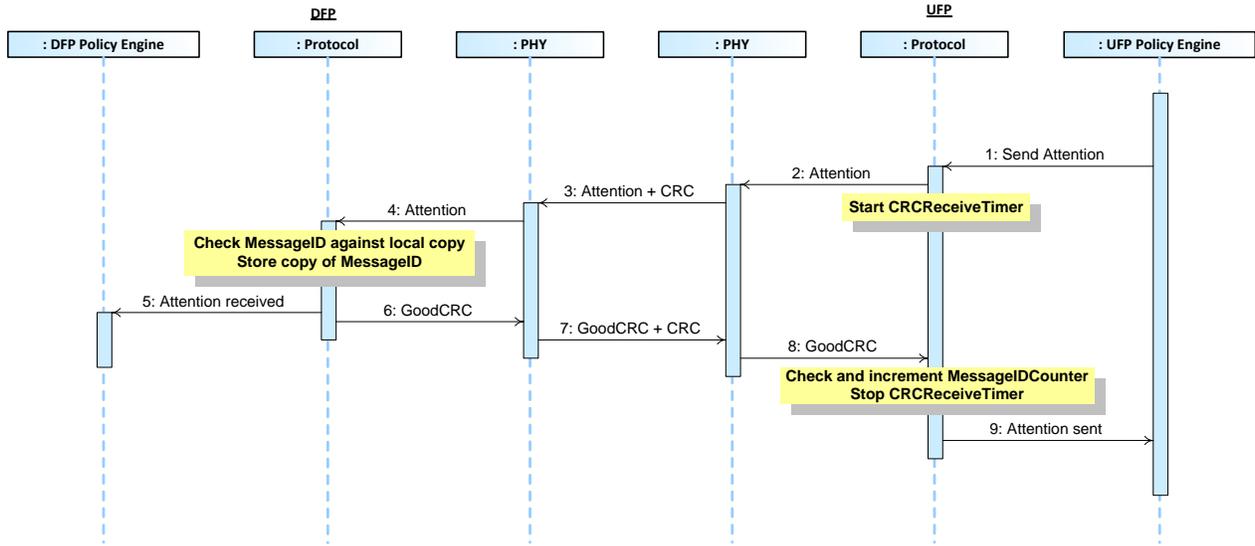


Table 8-34 below provides a detailed explanation of what happens at each labeled step in Figure 8-34 above.

Table 8-34 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 <i>Attention</i> Command request.
2		Protocol Layer creates the <i>Attention</i> Command request and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .
3	Physical Layer receives the <i>Attention</i> Command request and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>Attention</i> Command request.
4	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>Attention</i> Command request information to the Policy Engine that consumes it.	
5	The Policy Engine informs the Device Policy Manager	
6	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
7	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.
8		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> 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.

8.3.2.10 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 8-35 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 8-35 BIST Receiver Mode test

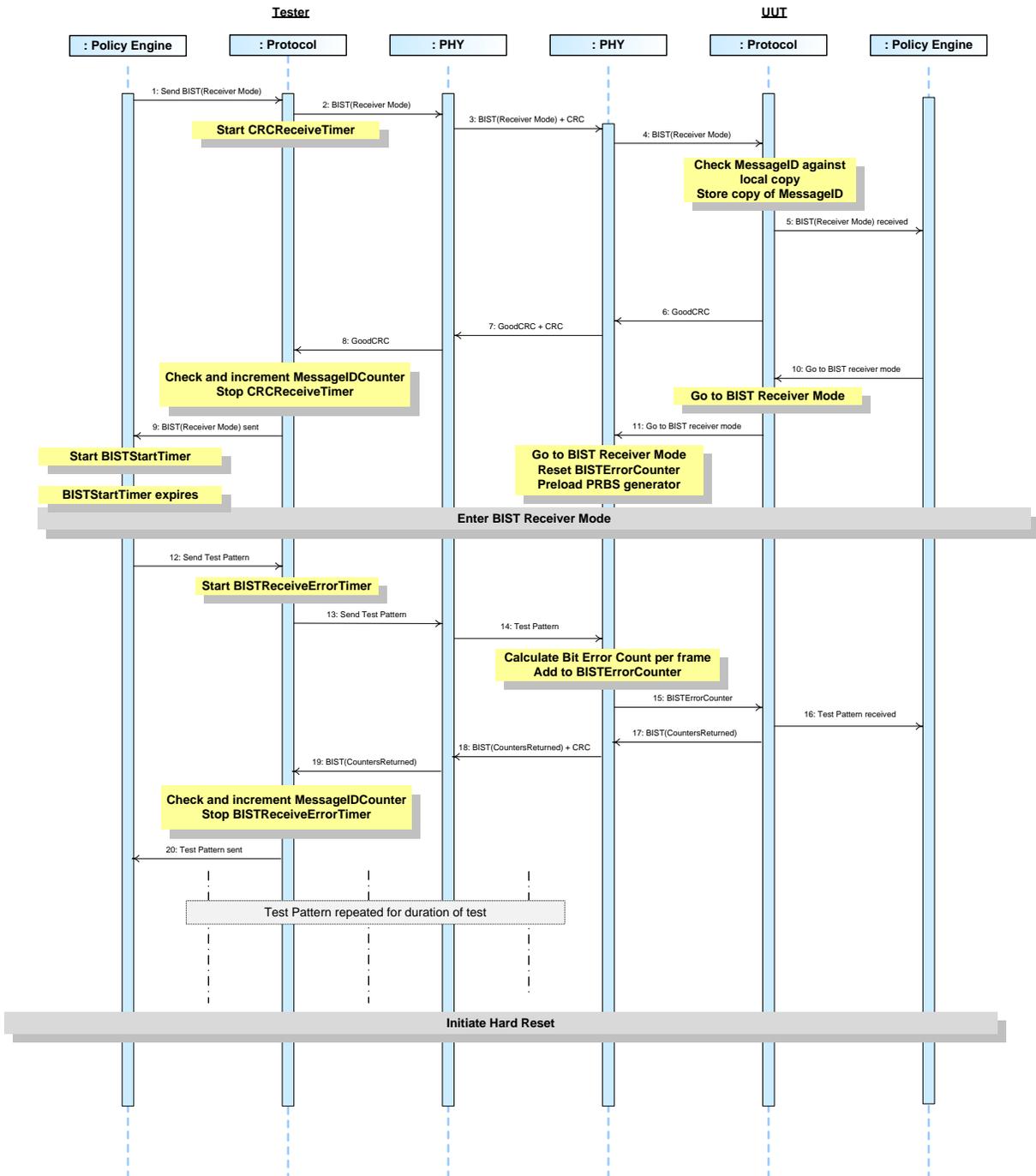


Table 8-35 Steps for BIST Receiver Mode test

Step	Tester	UUT
1	The Policy Engine directs the Protocol Layer to generate a BIST Message with BIST Data Object of BIST Receiver Mode to put the UUT into BIST receiver mode.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts CRCReceiveTimer .	
3	Physical Layer appends CRC and sends the BIST Message.	Physical Layer receives the BIST Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the BIST Message to the Protocol Layer.
5		Protocol Layer checks the MessageID 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 BIST Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the BIST Message was successfully sent. The Policy Engine initializes and runs the BISTStartTimer .	
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 BISTErrorCounter and preloads the PRBS generator.
	UUT enters BIST Receiver Mode	
12	After the BISTStartTimer 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 BISTReceiveErrorTimer .	
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 BISTErrorCounter .
15		Physical Layer forwards the BISTErrorCounter 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 BIST Message with BIST Data Object of Returned BIST Counters and passes it Physical Layer.
18	Physical Layer receives BIST Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the BIST 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 8-36 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 8-36 BIST Transmit Mode test

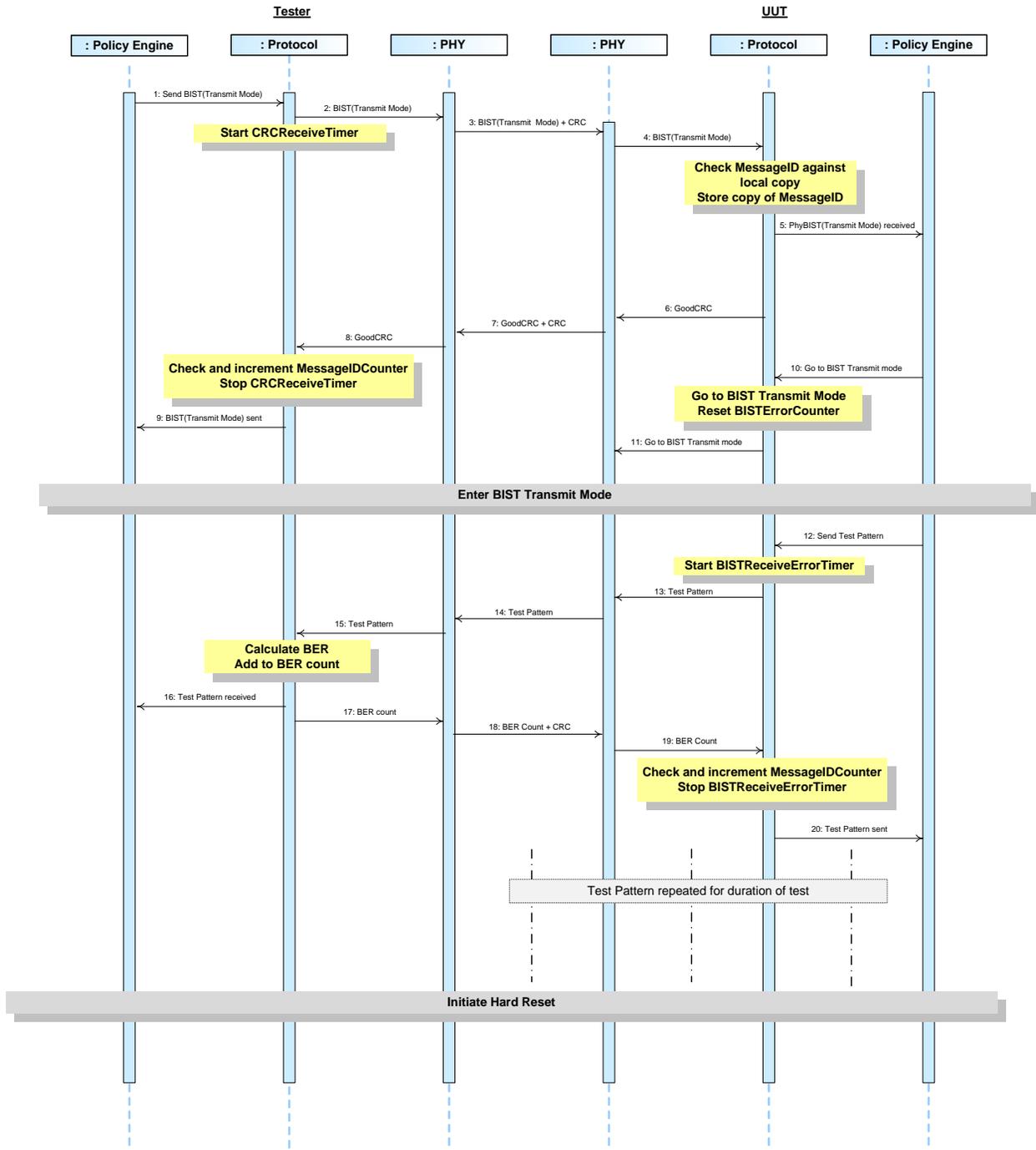


Table 8-36 Steps for BIST Transmit Mode test

Step	Tester	UUT
1	The Policy Engine directs the Protocol Layer to generate a BIST Message, with BIST Data Object of BIST Transmit Mode , to put the UUT into BIST Transmit Mode.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts CRCReceiveTimer .	
3	Physical Layer appends CRC and sends the BIST Message.	Physical Layer receives the BIST Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the BIST Message to the Protocol Layer.
5		Protocol Layer checks the MessageID 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 BIST Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the BIST 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 BISTErrorCounter .
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 BISTReceiveErrorTimer .
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 BIST Message with a BIST Data Object of Returned BIST Counters and passes it Physical Layer.	
18	Physical Layer appends CRC and sends the BIST Message.	Physical Layer receives the BIST Message and checks the CRC to verify the BIST Message.
19		Physical Layer removes the CRC and forwards the BIST Message to the Protocol Layer.

Step	Tester	UUT
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.	

8.3.2.10.3 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 8-37 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 Section 6.4.3.4 through Section 6.4.3.8.

Figure 8-37 BIST Eye Pattern Test

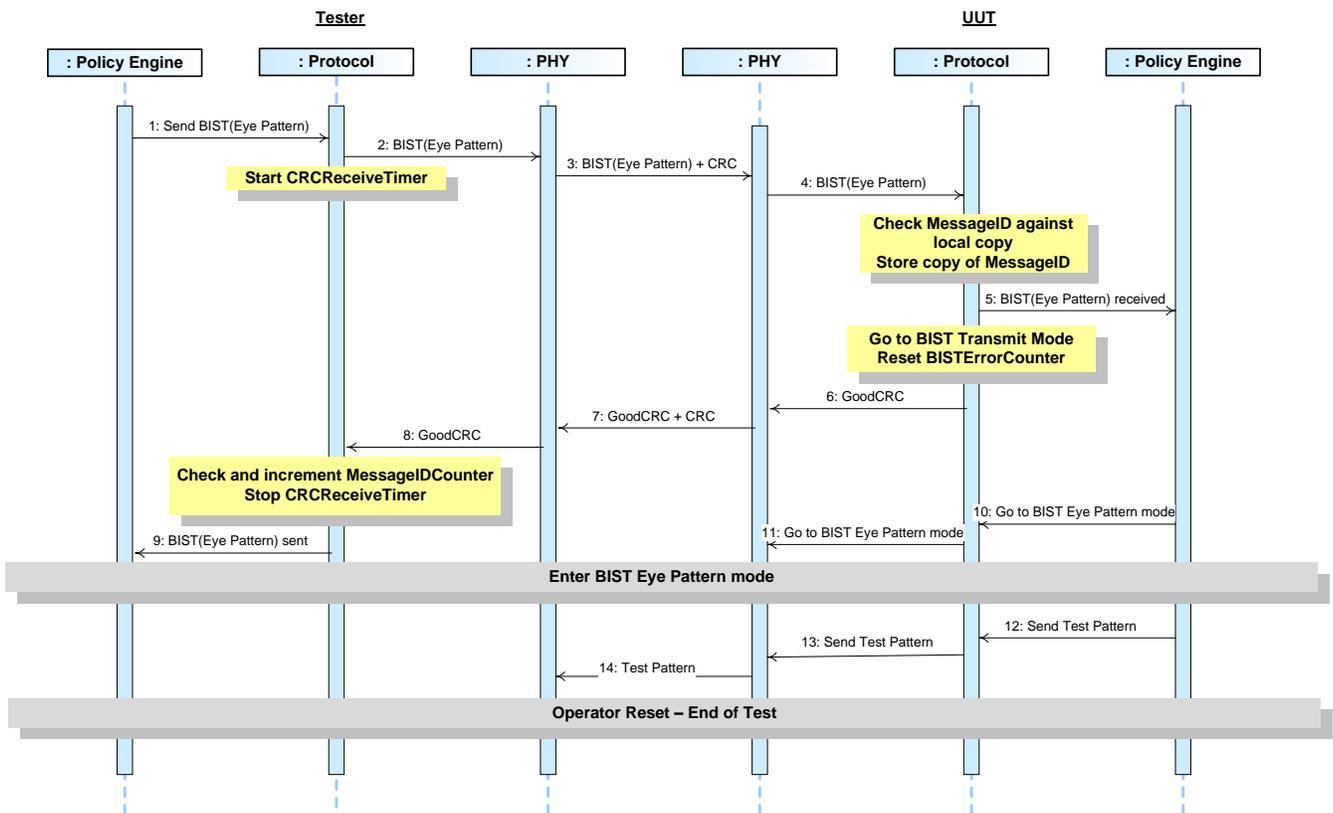


Table 8-37 Steps for BIST Eye Pattern Test

Step	Tester	UUT
1	The Policy Engine directs the Protocol Layer to generate a BIST Message, with a BIST Data Object of BIST Eye Pattern , to put the UUT into BIST Eye Pattern Test.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts CRCReceiveTimer .	
3	Physical Layer appends CRC and sends the BIST Message.	Physical Layer receives the BIST Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the BIST Message to the Protocol Layer.
5		Protocol Layer checks the MessageID 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 BIST Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the BIST 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	

8.3.2.10.4 BIST Test Data

The following is an example of a **BIST Test Data** test between a Tester and a UUT. When the UUT is connected to the Tester the sequence below is executed.

Figure 8-38 shows the Messages as they flow across the bus and within the devices.

- 1) Connection is established and stable.
- 2) Tester sends a **BIST** Message with a **BIST Test Data** BIST Data Object.
- 3) UUT answers with a **GoodCRC** Message.
- 4) Steps 2) and 3) are repeated any number of times.
- 5) The test ends after **Hard Reset** Signaling is issued.

See also Section 5.9.7 and Section 6.4.3.9.

Figure 8-38 BIST Test Data Test

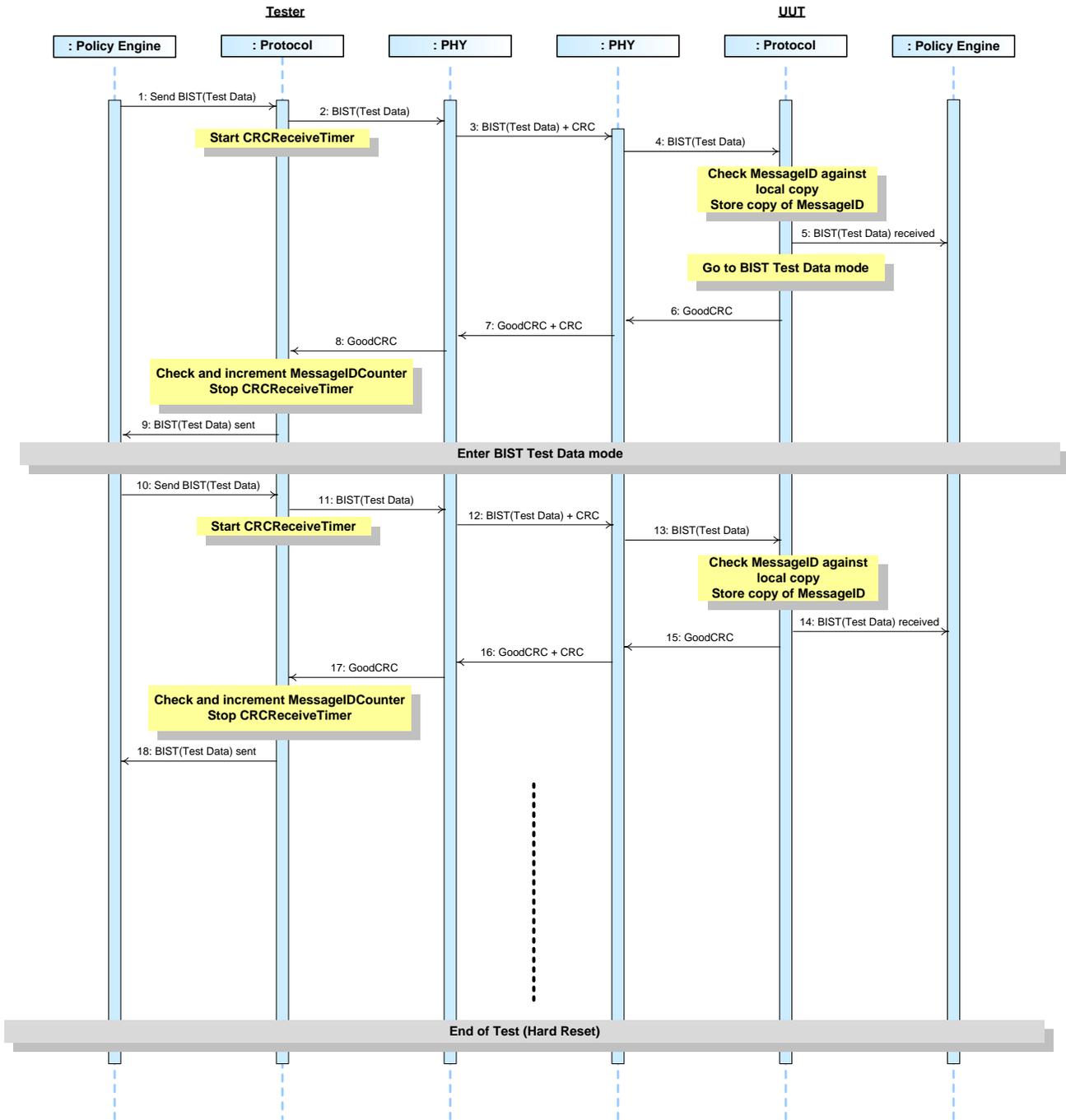


Table 8-38 Steps for BIST Test Data Test

Step	Tester	UUT
1	The Policy Engine directs the Protocol Layer to generate a <i>BIST</i> Message, with a BIST Data Object of <i>BIST Test Data</i> , to put the UUT into BIST Test Data test 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. The Policy Engine goes into BIST Test Data Mode where it sends no further Messages except for <i>GoodCRC</i> Messages in response to received Messages (see Section 6.4.3.9).
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.	
	UUT enters BIST Test Data test mode	
10	The Policy Engine directs the Protocol Layer to generate a <i>BIST</i> Message, with a BIST Data Object of <i>BIST Test Data</i> , to put the UUT into BIST Test Data test mode.	
11	Protocol Layer creates the Message and passes to Physical Layer. Starts <i>CRCReceiveTimer</i> .	
12	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.
13		Physical Layer removes the CRC and forwards the <i>BIST</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 forwards the received <i>BIST</i> Message information to the Policy Engine that consumes it. The Policy Engine goes into BIST Test Data Mode where it sends no further Messages except for <i>GoodCRC</i> Messages in response to received Messages (see Section 6.4.3.9).
15		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.

Step	Tester	UUT
16	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.
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>BIST</i> Message was successfully sent.	
	Repeat steps 10-18 any number of times	
	The UUT exits BIST Test Data test mode after a Hard Reset	

8.3.3 State Diagrams

8.3.3.1 Introduction to state diagrams used in Chapter 8

The state diagrams defined in Section 8.3.2.10.4 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 8-39 Outline of States

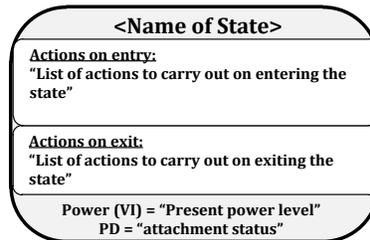


Figure 8-39 shows an outline of the states defined in the following sections. At the top there is the name of the state. This is followed by "Actions on entry" a list of actions carried out on entering the state. If there are also "Actions on exit" a list of actions carried out on exiting the state then 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 8-40 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 Sink. 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 8-41). It is also possible that the entry and return states are the same. Figure 8-42 indicates a state reference where each referenced state corresponds to either the entry state or the exit state.

Figure 8-40 References to states

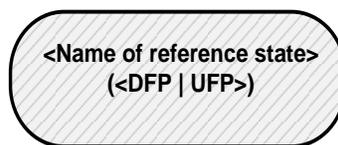


Figure 8-41 Example of state reference with conditions

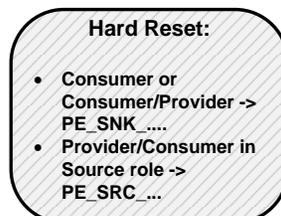
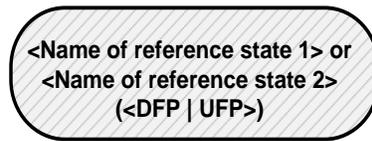


Figure 8-42 Example of state reference with the same entry and exit



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:

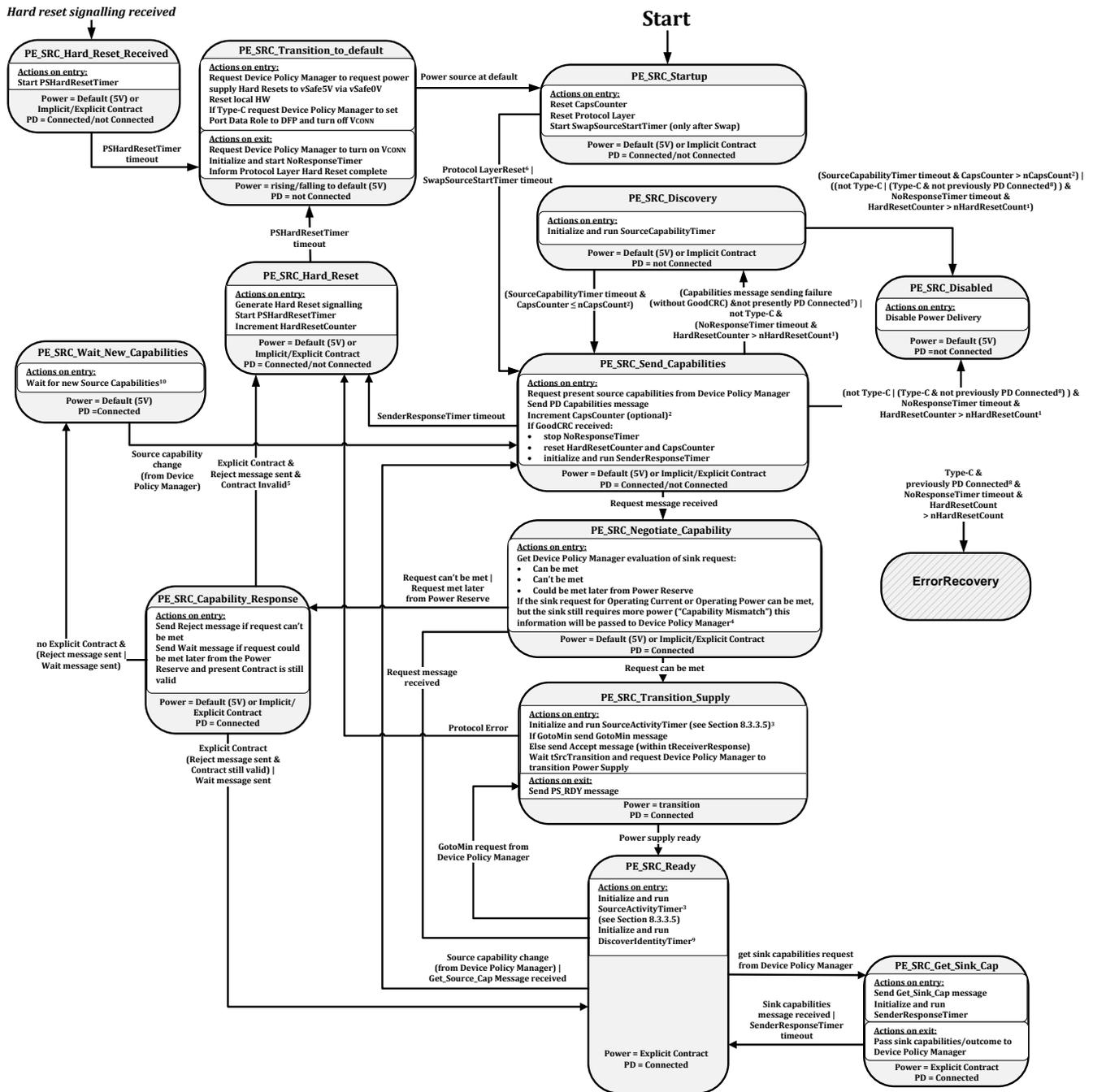
- 1) Message and related indications passed up to the Policy Engine from the Protocol Layer (Message sent, Message received etc.)
- 2) Events triggered within the Policy Engine e.g. timer timeouts.
- 3) Information and requests coming from the Device Policy manager relating either to Local Policy, or 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 could be encountered. For example where an outgoing Message is **Discarded**, due to an incoming Message by the Protocol Layer (see Section 6.9.2.2) 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 8-43 below shows the state diagram for the Policy Engine in a Source Port. The following sections describe operation in each of the states.

Figure 8-43 Source Port Policy Engine state diagram



¹ For a USB 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.

² Implementation of the **CapsCounter** is **Optional**. In the case where this is not implemented the Source **Shall** continue to send **Source_Capabilities** Messages each time the **SourceCapabilityTimer** times out.

³ 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.

⁴ 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.

⁵ “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.

⁶ After a Power Swap the new Source is required to wait an additional *tSwapSourceStart* before sending a *Source_Capabilities* Message. This delay is not required when first starting up a system.

⁷ 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 (USB Type-C, USB Type-A with insert detect, USB Micro-AB).

⁸ Port Partners are no longer PD Connected after a Hard Reset but for USB 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 USB Type-C Error Recovery.

⁹ 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* Message has yet been received in response to a *Discover Identity* Command.

¹⁰ In the *PE_SRC_Wait_New_Capabilities* State the Device Policy Manager *Should* either decide to send no further Source Capabilities or *Should* send a different set of Source Capabilities. Continuing to send the same set of Source Capabilities could result in a live lock situation.

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 6.9.2.3).

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-A 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:

- 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 USB Type-C Port
- Or the Port is a USB Type-C Port and the Port Partners have not been PD Connected (the USB 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:

- The Port is a USB Type-C Port
- And the Port Partners have previously been PD Connected (the USB 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

Note: this state can 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:

- The Port is not a USB Type-C Port
- Or the Port is a USB Type-C Port and the Port Partners have not been PD Connected (the USB 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:

- The Port is a USB Type-C Port
- And the Port Partners have previously been PD Connected (the USB 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 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.

The Policy Engine **Shall** transition to the **PE_SRC_Hard_Reset** state when:

- A Protocol Error occurs.

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.
- A **Get_Source_Cap** Message is received.

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_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

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.

8.3.3.2.10 PE_SRC_Hard_Reset_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)
- request a reset of the local hardware
- for a USB 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 USB Type-C connector **Shall** request the Device Policy Manager to turn on VCONN
- initialize and run the **NoResponseTimer**. 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.

8.3.3.2.12 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.

8.3.3.2.13 PE_SRC_Wait_New_Capabilities state

In this state the Policy Engine has been unable to negotiate an Explicit Contract and is waiting for new Capabilities from the Device Policy Manager.

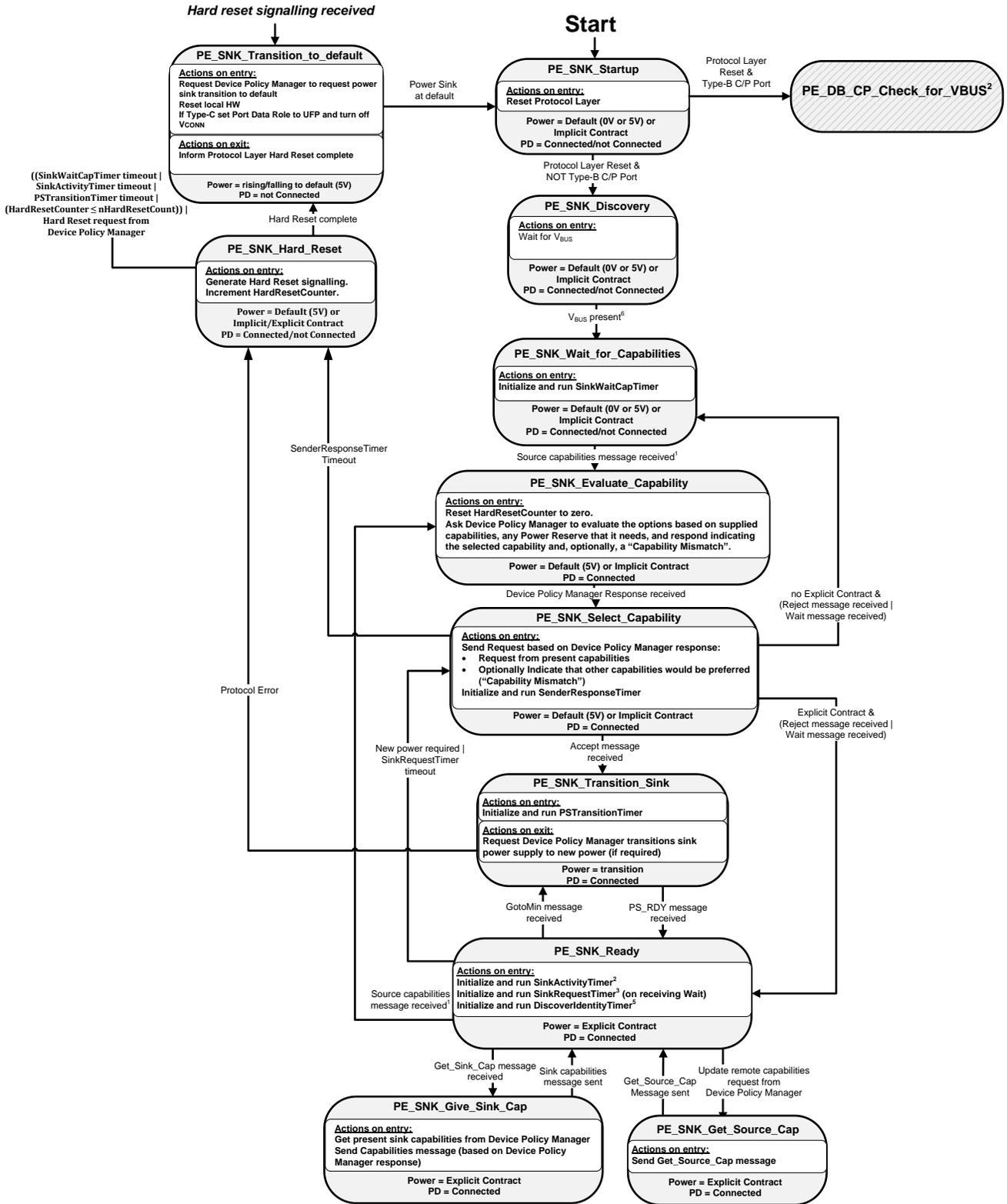
The Policy Engine **Shall** transition to the **PE_SRC_Send_Capabilities** state when:

- The Device Policy Manager indicates that Source Capabilities have changed.

8.3.3.3 Policy Engine Sink Port State Diagram

Figure 8-44 below shows the state diagram for the Policy Engine in a Sink Port. The following sections describe operation in each of the states.

Figure 8-44 Sink Port state diagram



¹ Source capabilities messages received in states other than *PE_SNK_Wait_for_Capabilities* and *PE_SNK_Ready* constitute a Protocol Error.

² The *SinkActivityTimer* **Shall Not** be run when operating at *vSafe5V* or when two systems using the USB Type-C connector are communicating, since *Ping* messages are *Optional*.

³ 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* Message which is not reset by a *Ping* Message.

⁴ During a Hard Reset the Source voltage will transition to *vSafe0V* and then transition to *vSafe5V*. Sinks need to ensure that V_{BUS} present is not indicated until after the Source has completed the Hard Reset process by detecting both of these transitions.

⁵ The *DiscoverIdentityTimer* is run when this is a VCONN Source and a PD Connection with a Cable Plug needs to be established i.e. no *GoodCRC* Message has yet been received in response to a *Discover Identity* Command.

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 6.9.2.3).

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 USB 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 *PE_SNK_Hard_Reset* state when:

- The Port is not a USB Type-C connector and
- There has been a *NoResponseTimer* timeout and
- The *HardResetCounter* \leq *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* Message is received.

When the *SinkWaitCapTimer* times out, the Policy Engine will perform a Hard Reset.

8.3.3.3.4 PE_SNK_Evaluate_Capability state

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 the new power level required, selected from the present offered capabilities. The Device Policy Manager **Shall** also indicate to the Policy engine a Capability Mismatch if the 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). 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 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.
- Request a reset of the local hardware.
- For a USB Type-C connector **Shall** request that the Port Data Role is set to UFP.

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:

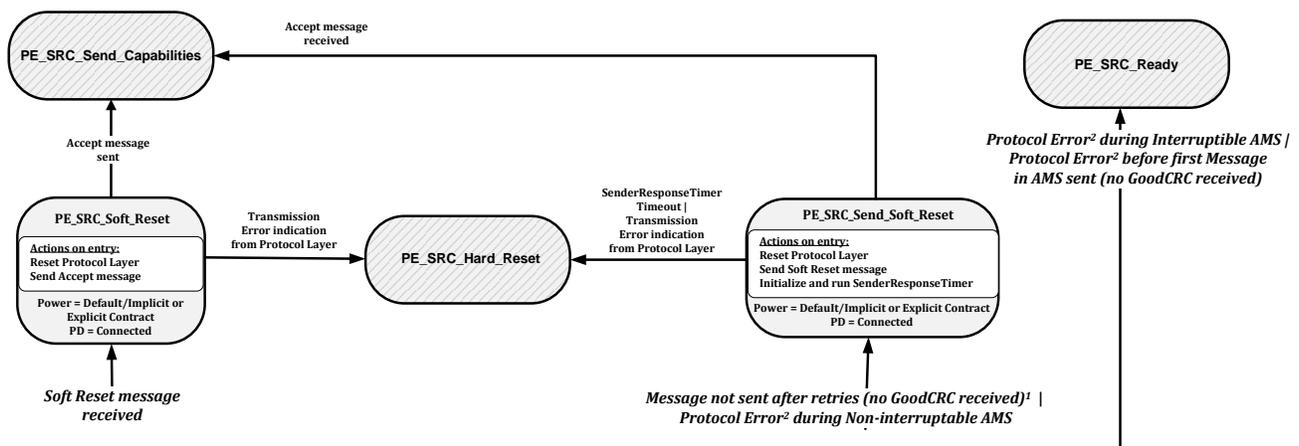
- The *Get_Source_Cap* Message is sent.

8.3.3.4 Soft Reset and Protocol Error State Diagrams

8.3.3.4.1 Source Port Soft Reset and Protocol Error State Diagram

Figure 8-45 below shows the state diagram for the Policy Engine in a Source Port when performing a Soft Reset of its Port Partner. The following sections describe operation in each of the states.

Figure 8-45 Source Port Soft Reset and Protocol Error State Diagram



¹ Excludes the *Soft_Reset* Message itself.

² An unrecognized or unsupported Message will result in a *Reject* Message response being generated (see Section 6.7.1).

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 during a Non-interruptible AMS (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 8-43). In this case, the *PE_SRC_Discovery* state is entered (see Section 8.3.3.2.3).
- When the voltage is in transition due to a new Explicit Contract being negotiated (see Section 8.3.3.2). In this case a Hard Reset will be generated.
- During a Power Role Swap when the power supply is in transition (see Section 8.3.3.6.3.1 and Section 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 USB Type-C Error Recovery will be triggered directly.

Note that Protocol Errors occurring in the following situations **Shall Not** lead to a Soft Reset, but **Shall** result in a transition to the *PE_SRC_Ready* state where the Message received will be handled as if it had been received in the *PE_SRC_Ready* state:

- Protocol Errors occurring during an Interruptible AMS.
- Protocol Errors occurring during any AMS where the first Message in the sequence has not yet been sent i.e. an unexpected Message is received instead of the expected *GoodCRC* Message response.

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* state when:

- The Protocol Layer indicates that a transmission error has occurred.

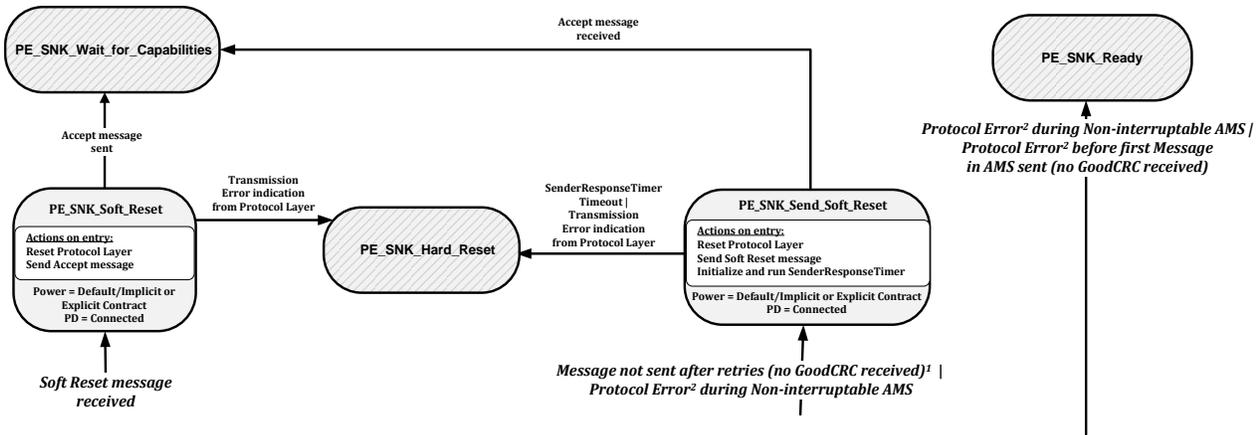
The decision as to whether to go to **PE_SRC_Hard_Reset** or **PE_SNK_Hard_Reset** **Shall** depend on the type of device:

- 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 and Protocol Error State Diagram

Figure 8-46 below shows the state diagram for the Policy Engine in a Sink Port when performing a Soft Reset of its Port Partner. The following sections describe operation in each of the states.

Figure 8-46 Sink Port Soft Reset and Protocol Error Diagram



1

Excludes the **Soft_Reset** Message itself.

² An unrecognized or unsupported Message will result in a **Reject** Message response being generated (see Section 6.3.4).

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 during a Non-interruptible AMS (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:

- When the voltage is in transition due to a new Explicit Contract being negotiated (see Section 8.3.3.3). In this case a Hard Reset will be generated.
- During a Power Role Swap when the power supply is in transition (see Section 8.3.3.6.3.1 and Section 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 USB Type-C Error Recovery will be triggered directly.

Note that Protocol Errors occurring in the following situations **Shall Not** lead to a Soft Reset, but **Shall** result in a transition to the **PE_SNK_Ready** state where the Message received will be handled as if it had been received in the **PE_SNK_Ready** state:

- Protocol Errors occurring during an Interruptible AMS.
- Protocol Errors occurring during any AMS where the first Message in the sequence has not yet been sent i.e. an unexpected Message is received instead of the expected **GoodCRC** Message response.

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_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 Sink Port in a Provider/Consumer **Shall** go to *PE_SRC_Hard_Reset*.
- The Sink 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.

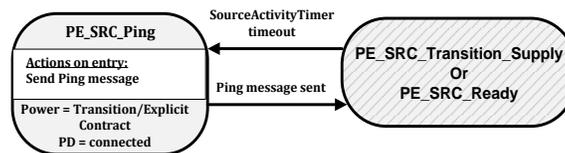
The decision as to whether to go to *PE_SRC_Hard_Reset* or *PE_SNK_Hard_Reset* **Shall** depend on the type of device:

- The Sink Port in a USB Type-A Provider/Consumer **Shall** go to *PE_SRC_Hard_Reset*.
- The Sink 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 8-47 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 8-47 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 8-43) 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.

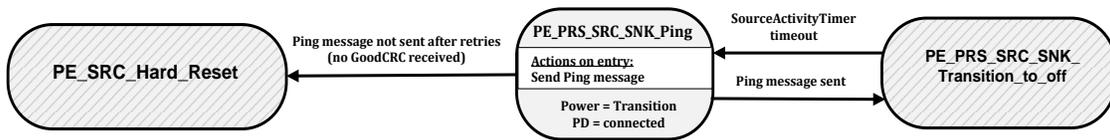
8.3.3.6.1 Type-A/B Dual-Role State Diagrams

The State Diagrams in this section **shall** apply to all USB Type-A/B Dual-Role Ports.

8.3.3.6.1.1 Type-A/B Dual-Role (initially Source Port) Ping State Diagram

Figure 8-48 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 8-48 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 8-56) when:

- The **Ping** Message has been successfully sent.

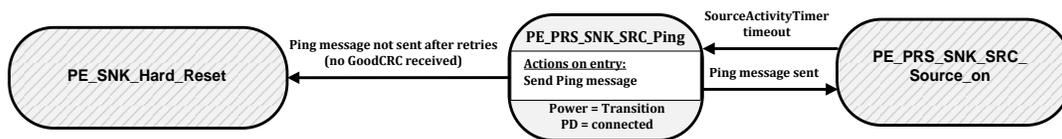
The Policy Engine **shall** transition to the **PE_SRC_Hard_Reset** when:

- 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 8-49 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 8-49 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 8-57) when:

- The *Ping* Message has been successfully sent.

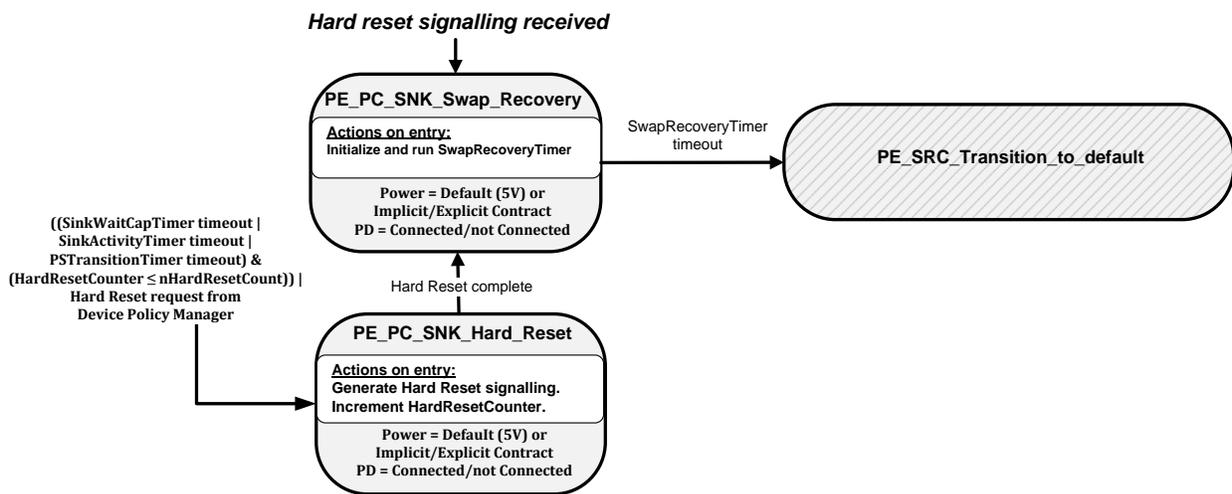
The Policy Engine **Shall** transition to the *PE_SNK_Hard_Reset* state when:

- 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 8-50 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 8-50 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 |
- (*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*.

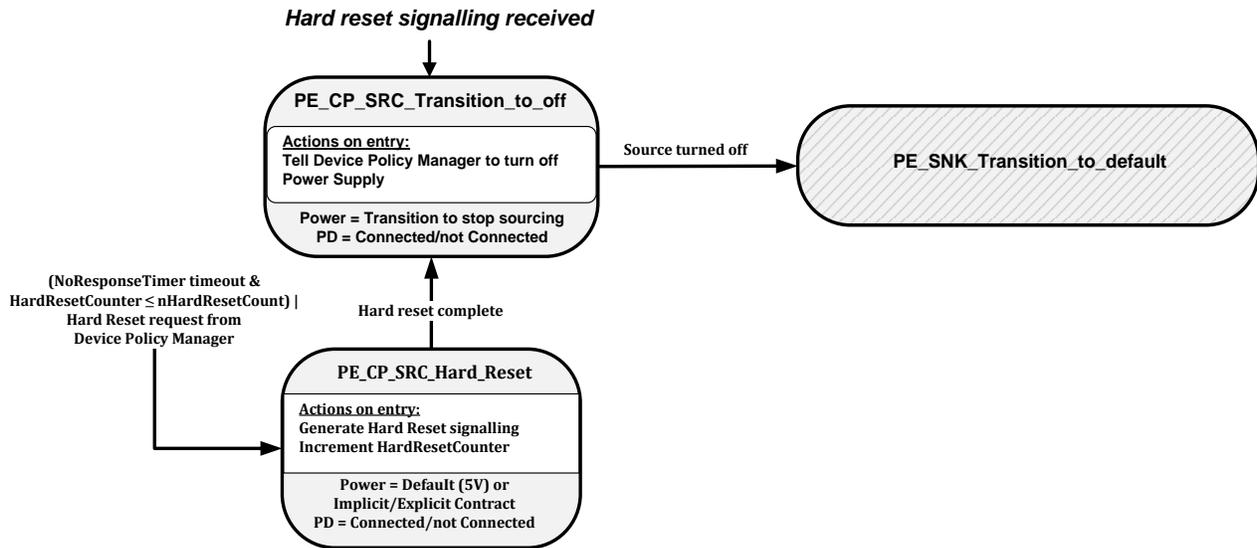
The Policy Engine **Shall** transition to the *PE_SRC_Transition_to_default* state for a Source Port when:

- 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 8-51 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 8-51 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** \leq **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:

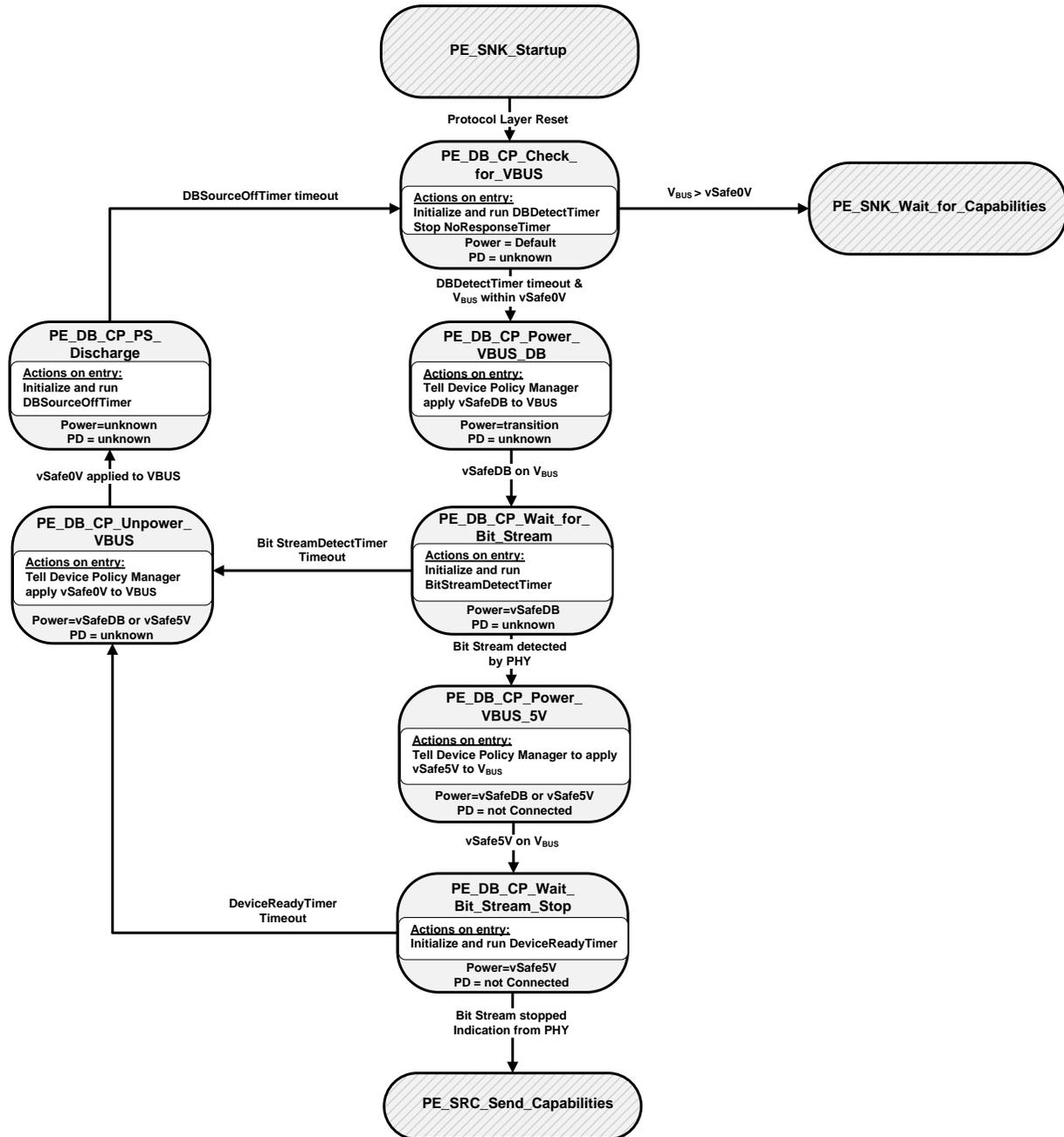
- The power supply has been turned off.

8.3.3.6.1.5 Type-A/B Consumer/Provider Dead Battery/Power Loss State Diagram

Figure 8-52 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 8-52 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:

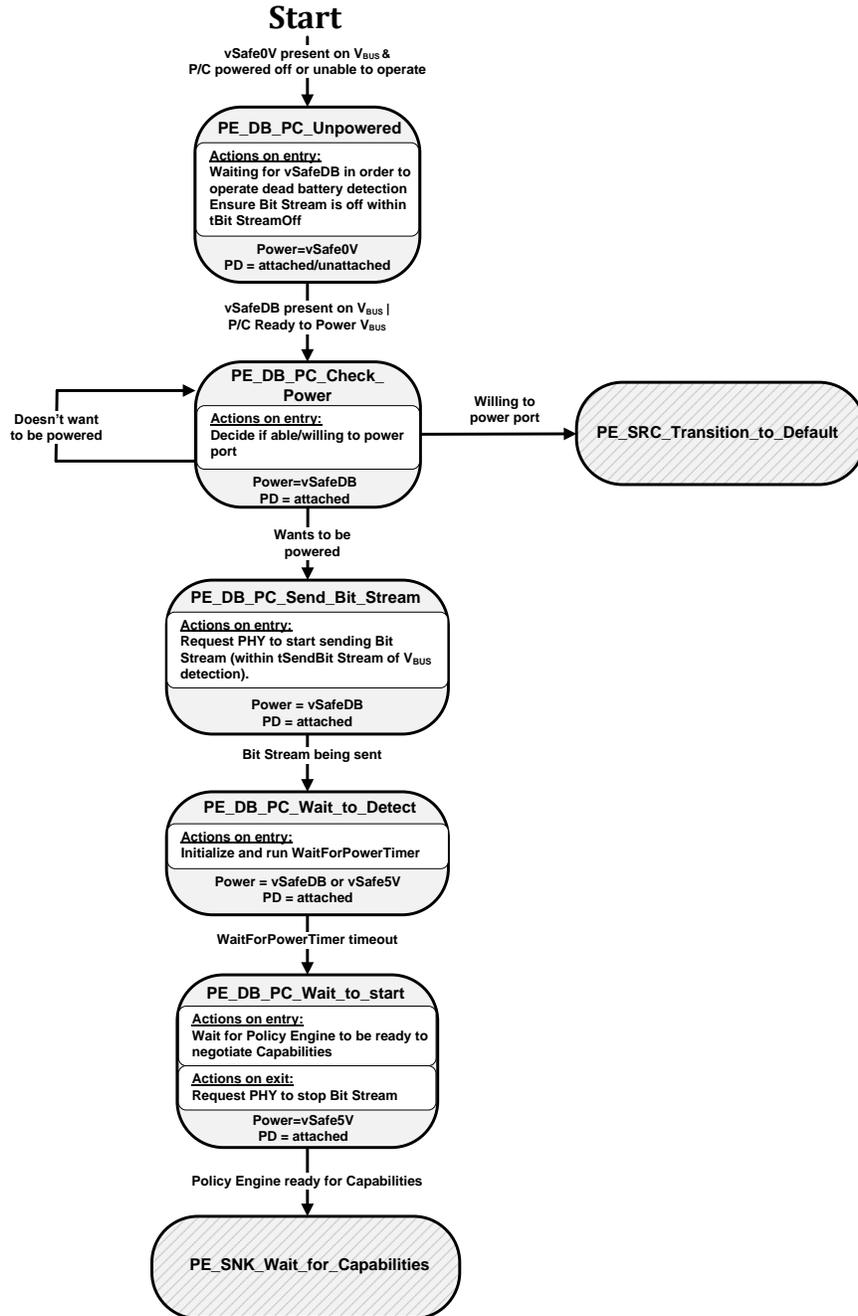
- The *DBSourceOffTimer* times out.

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 8-53 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** start up as a Source Port.

Figure 8-53 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 might 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 4.2.1).

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 **WaitForPowerTimer** 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.

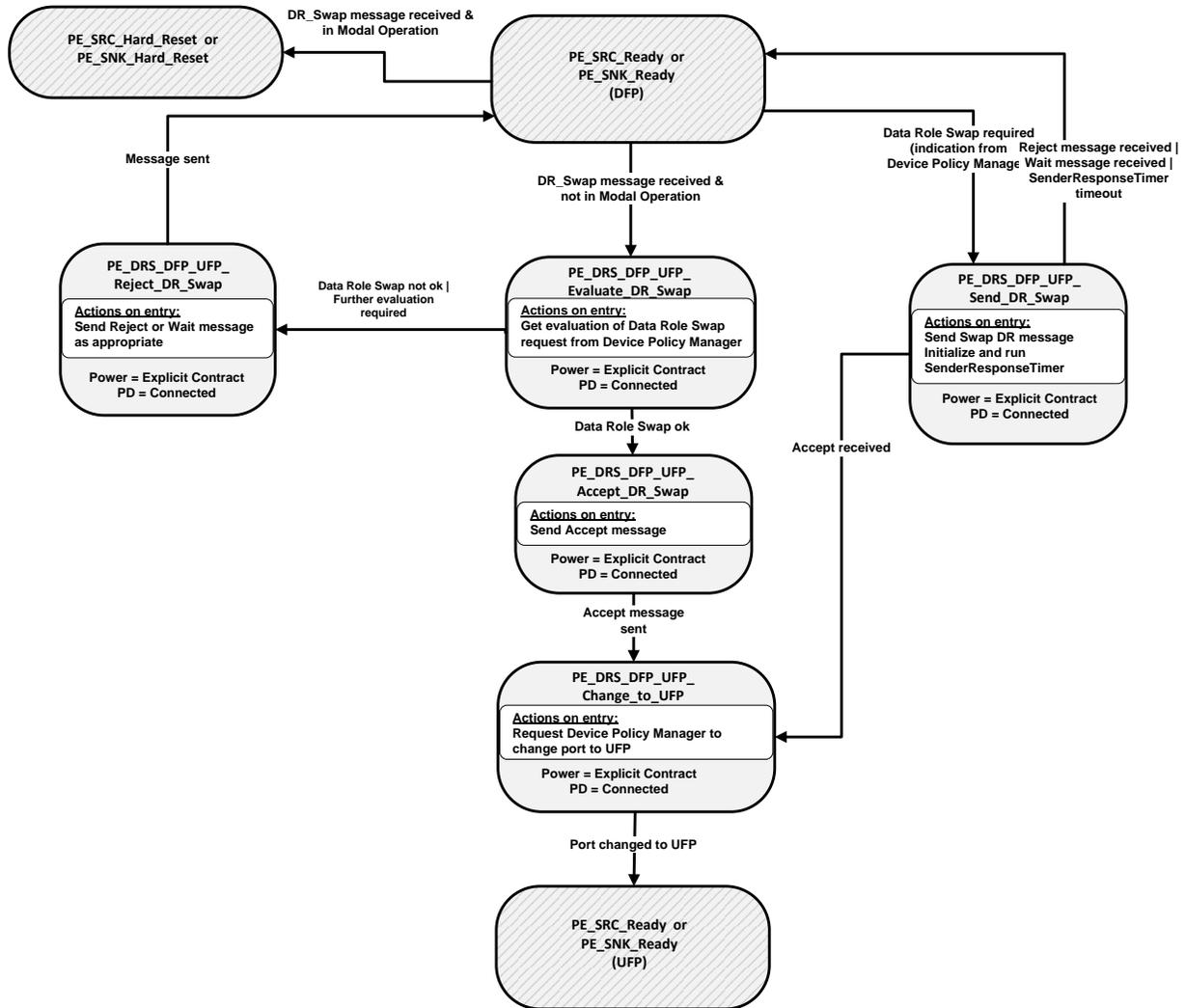
8.3.3.6.2 Type-C DR_Swap State Diagrams

The State Diagrams in this section **Shall** apply to all Dual-Role Ports that are **[USB Type-C 1.2]** DRPs.

8.3.3.6.2.1 Type-C Policy Engine in DFP to UFP Data Role Swap State Diagram

Figure 8-54 shows the additional state diagram required to perform a Data Role Swap from USB Type-C DFP to UFP operation and the changes that **Shall** be followed for error and Hard Reset handling.

Figure 8-54: USB 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_DRD_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 USB 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).

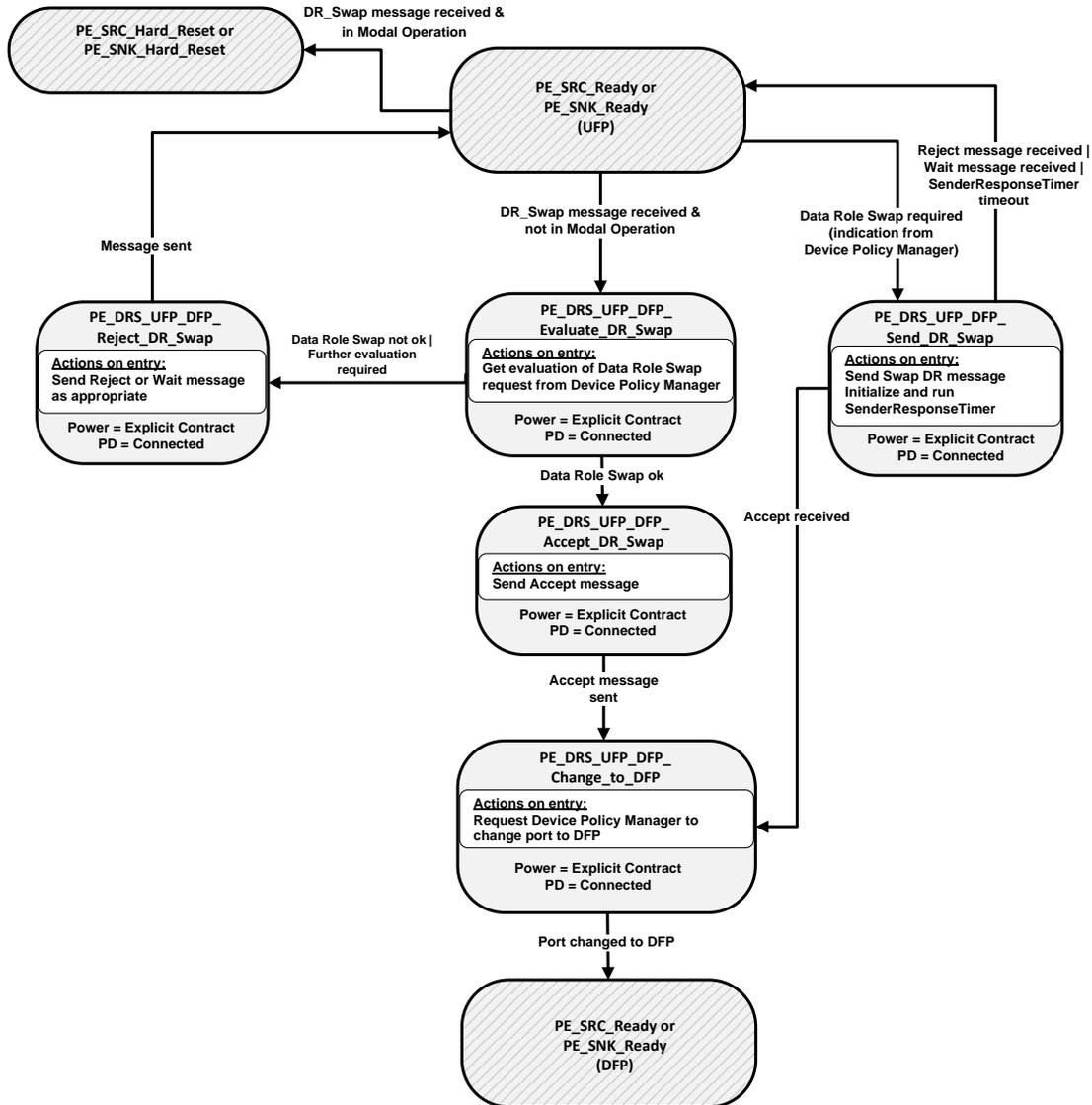
The Policy Engine **Shall** continue as a DFP and **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state when:

- The **Reject** or **Wait** Message has been sent.

8.3.3.6.2.2 Type-C Policy Engine in UFP to DFP Data Role Swap State Diagram

Figure 8-55 shows the additional state diagram required to perform a Data Role Swap from USB Type-C DRP UFP to DFP operation and the changes that **Shall** be followed for error and Hard Reset handling.

Figure 8-55: USB 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 USB 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.

8.3.3.6.3 Common Dual-Role Port State Diagrams

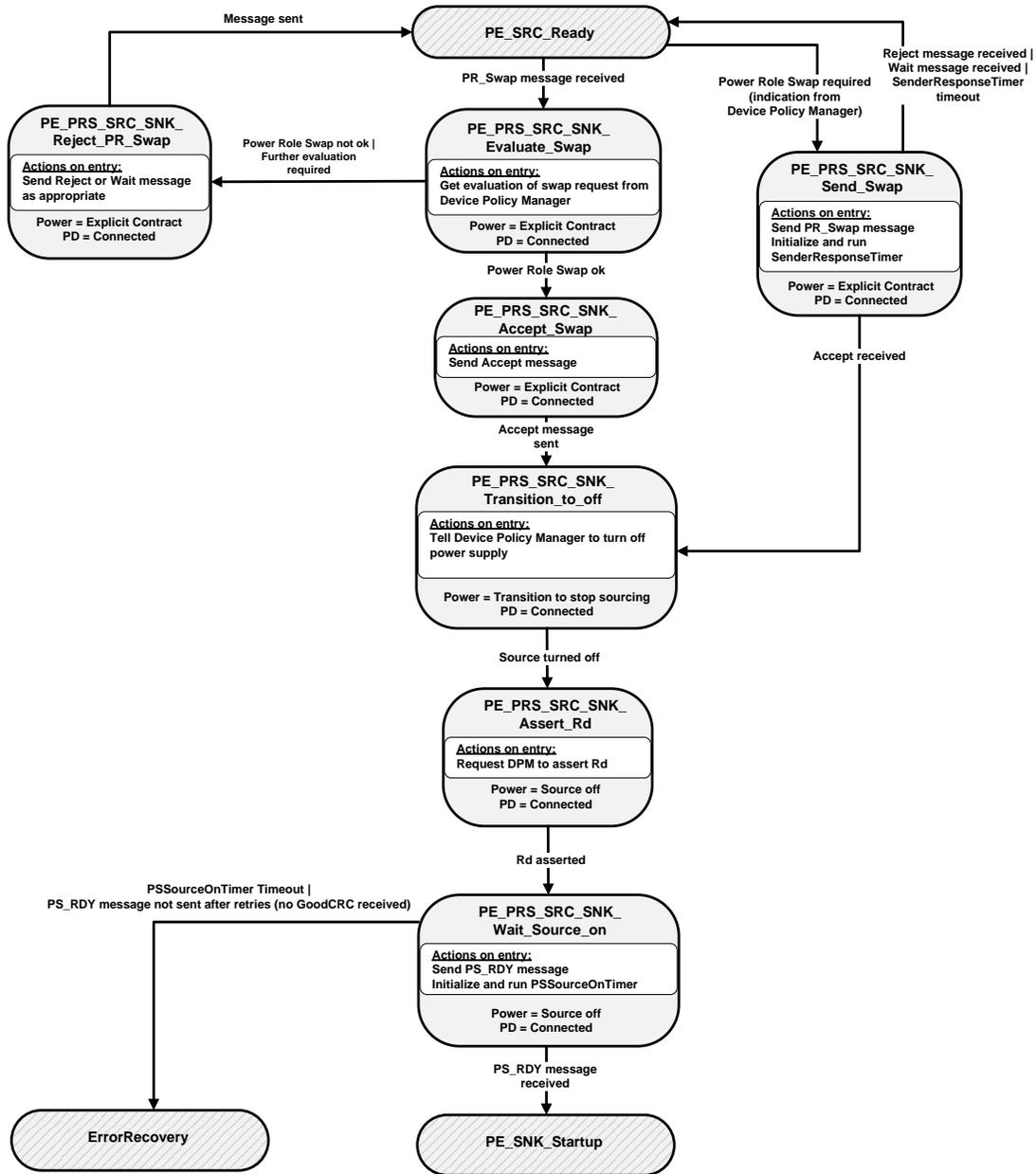
The State Diagrams in this section **Shall** apply to all Dual-Role Ports: both USB Type-A/B and **[USB Type-C 1.2]** DRP.

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 8-56 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 8-56: Dual-Role Port in Source to Sink Power Role Swap State Diagram



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** 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 *[USB Type-C 1.2]* 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 *[USB Type-C 1.2]* DRP

8.3.3.6.3.1.5 PE_PRS_SRC_SNK_Assert_Rd state

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_on 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 [USB Type-C 1.2] 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 [USB Type-C 1.2] 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.

8.3.3.6.3.1.7 PE_PRS_SRC_SNK_Send_PR_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_PR_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:

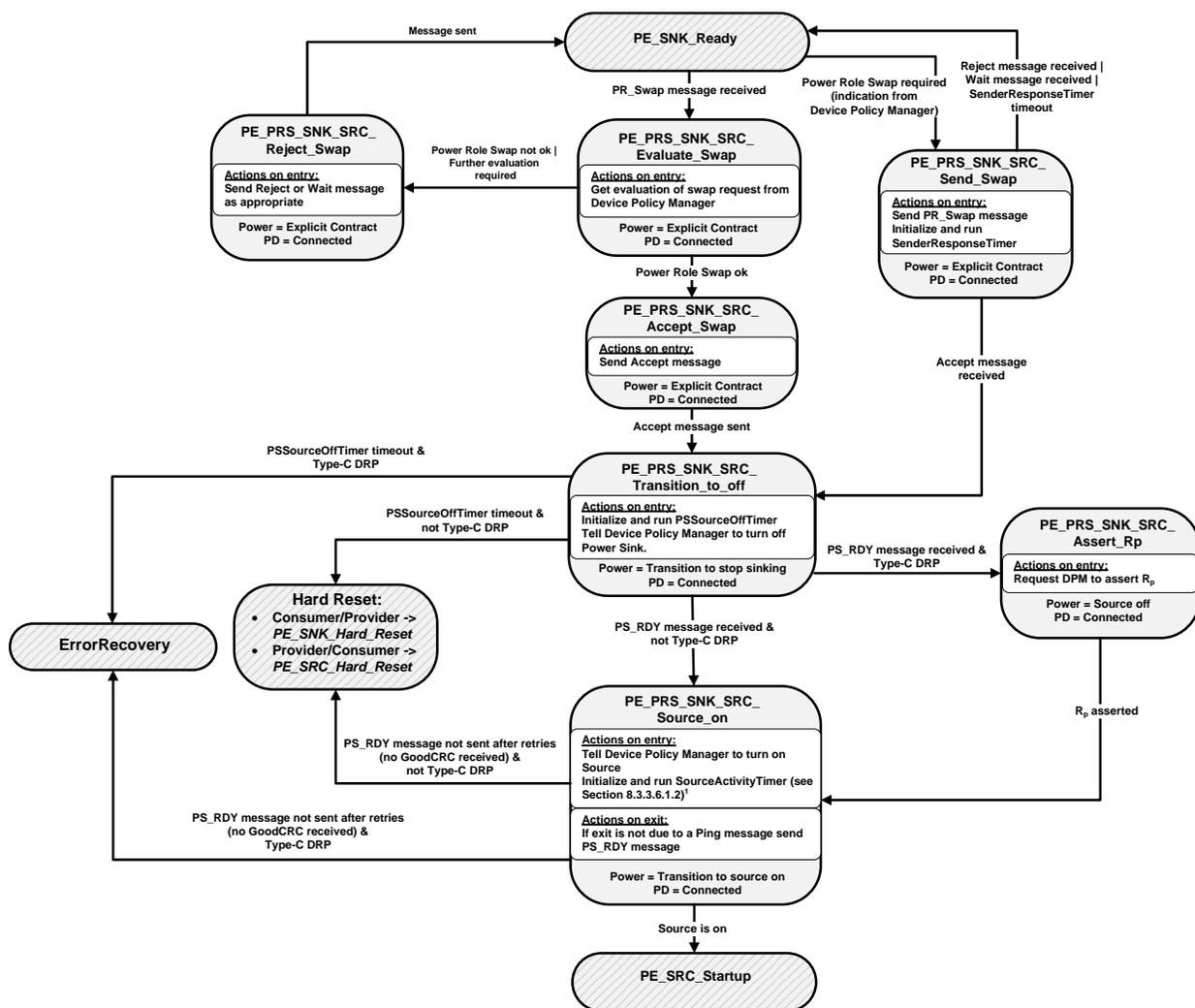
- The **Reject** or **Wait** Message has been sent.

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 8-57 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 8-57: Dual-role Port in Sink to Source Power Role Swap State Diagram



¹ 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_PR_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_PR_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 Port is not a **[USB Type-C 1.2]** 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 Policy Engine **Shall** transition to the **ErrorRecovery** state when:

- The **PSSourceOffTimer** times out and
- The Port is a **[USB Type-C 1.2]** 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 **[USB Type-C 1.2]** DRP

The Policy Engine **Shall** transition to the **PE_PRS_SNK_SRC_Assert_Rp** state when:

- A **PS_RDY** Message is received and
- The Port is a **[USB Type-C 1.2]** 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_d to R_p .

The Policy Engine **Shall** transition to the **PE_PRS_SNK_SRC_Source_on** state when:

- The Device Policy Manager indicates that R_d 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 [USB Type-C 1.2] 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 [USB Type-C 1.2] 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_PR_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_PR_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).

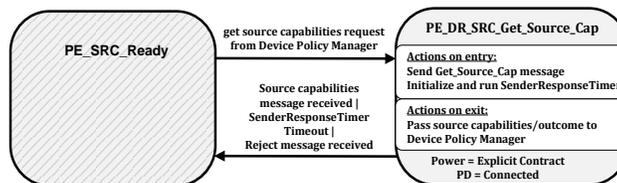
The Policy Engine **Shall** transition to the **PE_SNK_Ready** state when:

- The **Reject** or **Wait** Message has been sent.

8.3.3.6.3.3 Dual-Role (Source Port) Get Source Capabilities State Diagram

Figure 8-58 shows the state diagram for a Dual-Role Power 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 8-58 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).

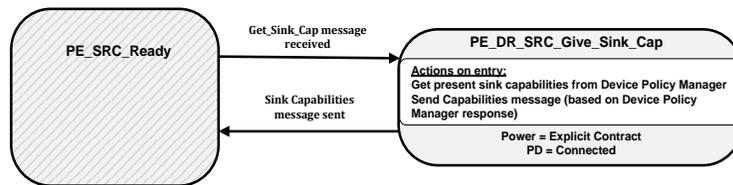
The Policy Engine **Shall** transition back to the **PE_SRC_Ready** state (see Figure 8-43) when:

- 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 8-59 shows the state diagram for a Dual-Role Power device, presently operating as a Source, on receiving a **Get_Sink_Cap** Message. See also Section 6.4.1.1.3.

Figure 8-59 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.

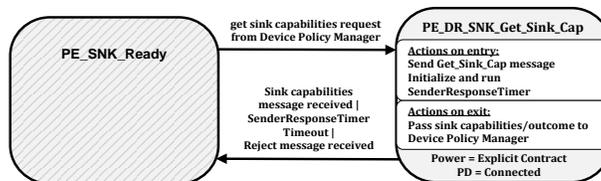
The Policy Engine **Shall** transition back to the **PE_SRC_Ready** state (see Figure 8-43) when:

- The **Sink_Capabilities** Message has been successfully sent.

8.3.3.6.3.5 Dual-Role (Sink Port) Get Sink Capabilities State Diagram

Figure 8-60 shows the state diagram for a Dual-Role Power 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 8-60 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).

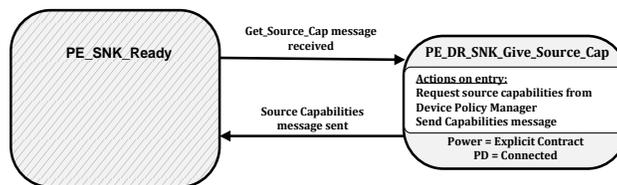
The Policy Engine **Shall** transition back to the **PE_SNK_Ready** state (see Figure 8-44 and Figure 8-49) when:

- 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 8-61 shows the state diagram for a Dual-Role Power device, presently operating as a Sink, on receiving a **Get_Source_Cap** Message. See also Section 6.4.1.1.3.

Figure 8-61 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.

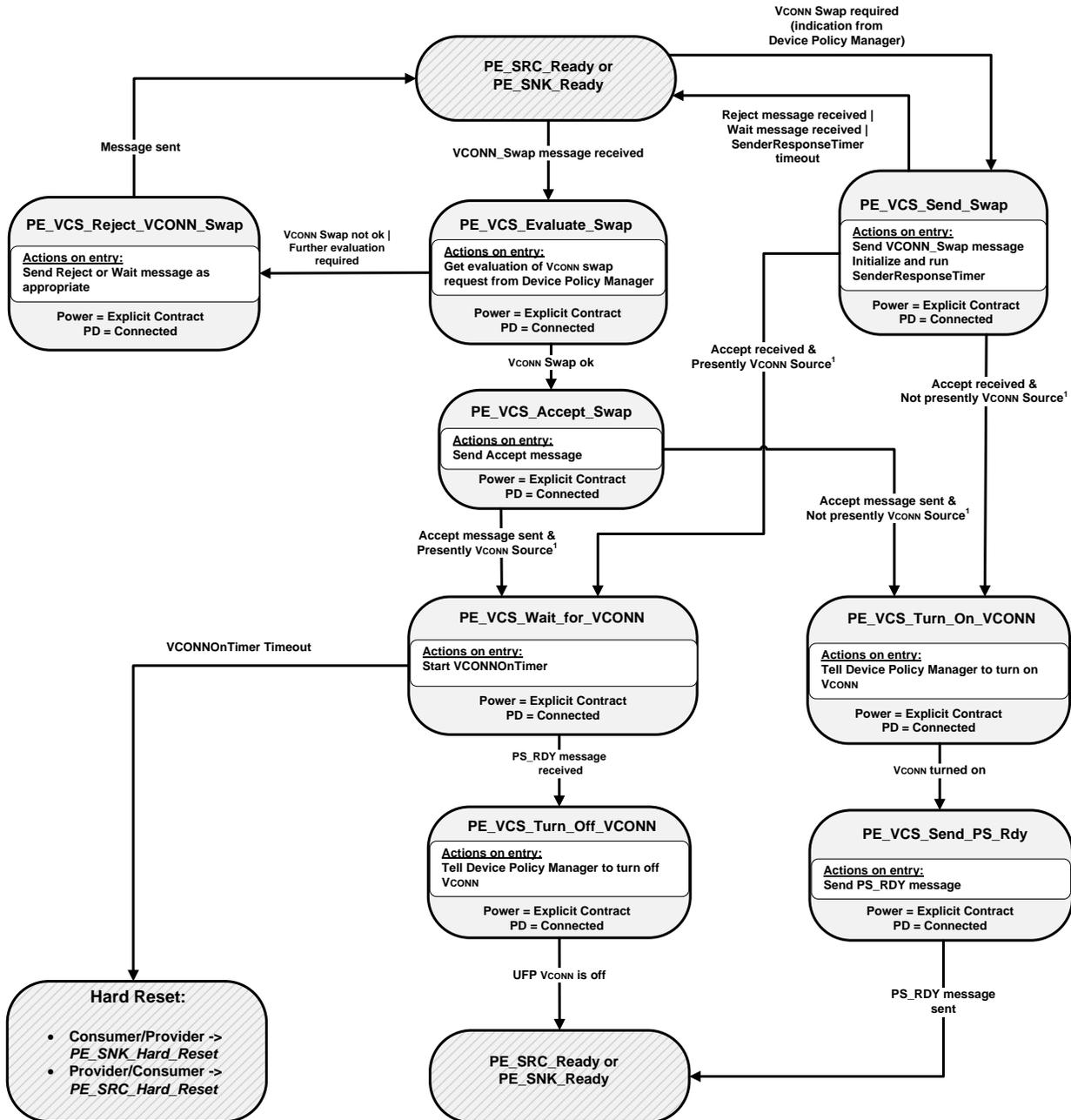
The Policy Engine **Shall** transition back to the **PE_SNK_Ready** state (see Figure 8-44 and Figure 8-49) when:

- The **Source_Capabilities** Message has been successfully sent.

8.3.3.7 Type-C VCONN Swap State Diagram

The State Diagram in this section **Shall** apply to **[USB Type-C 1.2]** Ports that supply VCONN. Figure 8-62 shows the state operation for a USB Type-C Port on sending or receiving a VCONN Swap request.

Figure 8-62 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_Send_Swap state

The *PE_VCS_Send_Swap* state is entered from either the *PE_SRC_Ready* or *PE_SNK_Ready* state 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**.

The Policy Engine **Shall** transition to the **PE_VCS_Wait_For_VCONN** state when:

- 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_Evaluate_Swap state

The **PE_VCS_Evaluate_Swap** state is entered from either the **PE_SRC_Ready** or **PE_SNK_Ready** state when the Policy Engine receives a **VCONN_Swap** Message.

On entry to the **PE_VCS_Evaluate_Swap** 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.

8.3.3.7.1.3 PE_VCS_Accept_Swap state

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.

8.3.3.7.1.4 PE_VCS_Reject_Swap state

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.

8.3.3.7.1.5 PE_VCS_UFP_Wait_for_VCONN state

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.

8.3.3.7.1.6 PE_VCS_Turn_Off_VCONN state

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.

8.3.3.7.1.7 PE_VCS_Turn_On_VCONN state

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.

8.3.3.7.1.8 PE_VCS_Send_Ps_Rdy state

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.

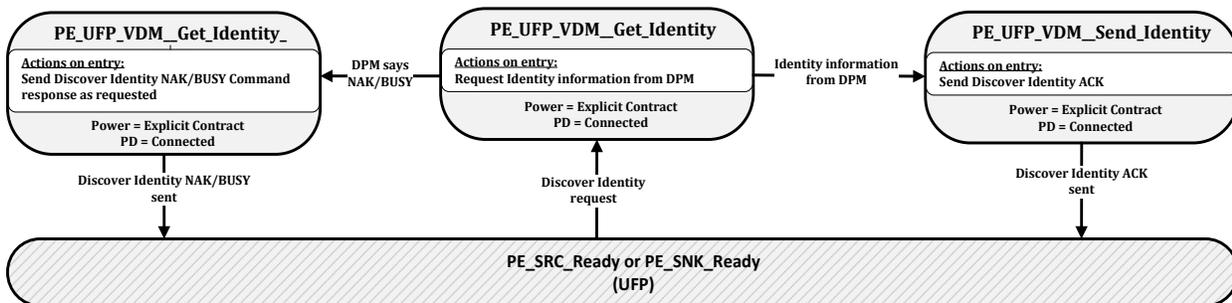
8.3.3.8 UFP Structured VDM State Diagrams

The State Diagrams in this section **Shall** apply to all UFPs that support structured VDMs.

8.3.3.8.1 UFP Structured VDM Discover Identity State Diagram

Figure 8-63 shows the state diagram for a UFP in response to a **Discover Identity** Command.

Figure 8-63 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 state

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.

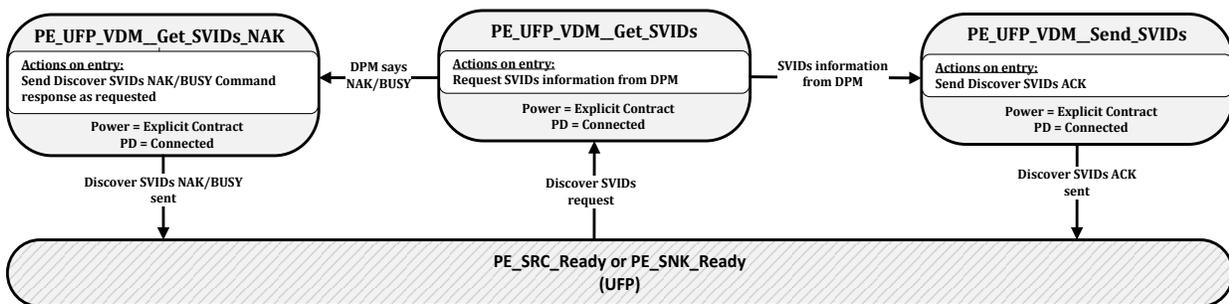
The Policy Engine *Shall* transition to either the *PE_SRC_Ready* or *PE_SNK_Ready* state for a UFP when:

- 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 8-64 shows the state diagram for a UFP in response to a *Discover SVIDs* Command.

Figure 8-64 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 response has been sent.

8.3.3.8.2.3 PE_UFP_VDM_Get_SVIDs_NAK state

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.

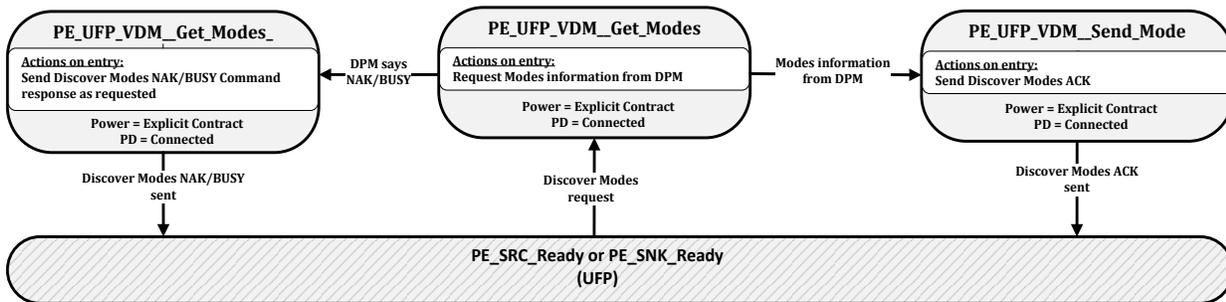
The Policy Engine **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state for a UFP when:

- 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 8-65 shows the state diagram for a UFP in response to a **Discover Modes** Command.

Figure 8-65 UFP Structured VDM Discover Modes State Diagram



8.3.3.8.3.1 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.

8.3.3.8.3.2 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 state

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.

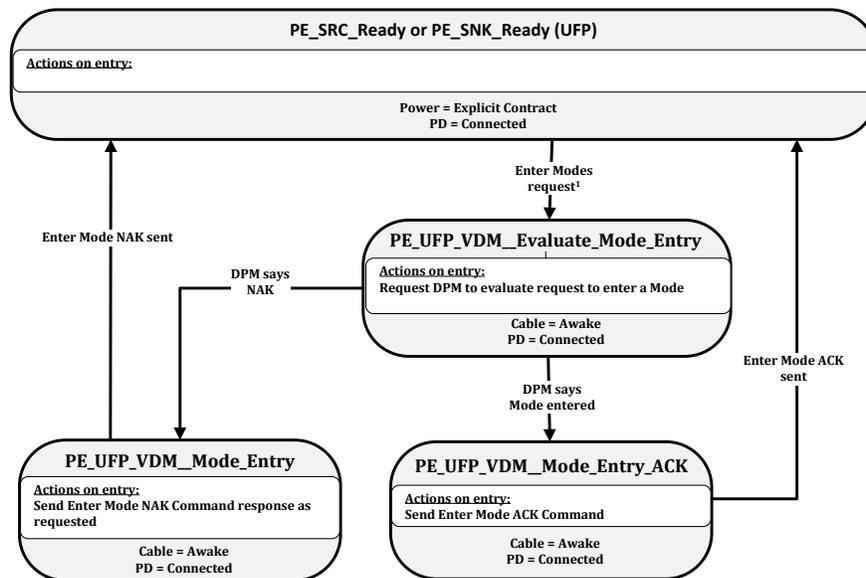
The Policy Engine **Shall** transition to either the *PE_SRC_Ready* or *PE_SNK_Ready* state for a UFP when:

- 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 8-66 shows the state diagram for a UFP in response to an *Enter Mode* Command.

Figure 8-66 UFP Structured VDM Enter Mode State Diagram



¹ The UFP is required to be in USB operation or USB Safe State at this point.

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 response to the Mode request is NAK.

8.3.3.8.4.2 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.

8.3.3.8.4.3 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.

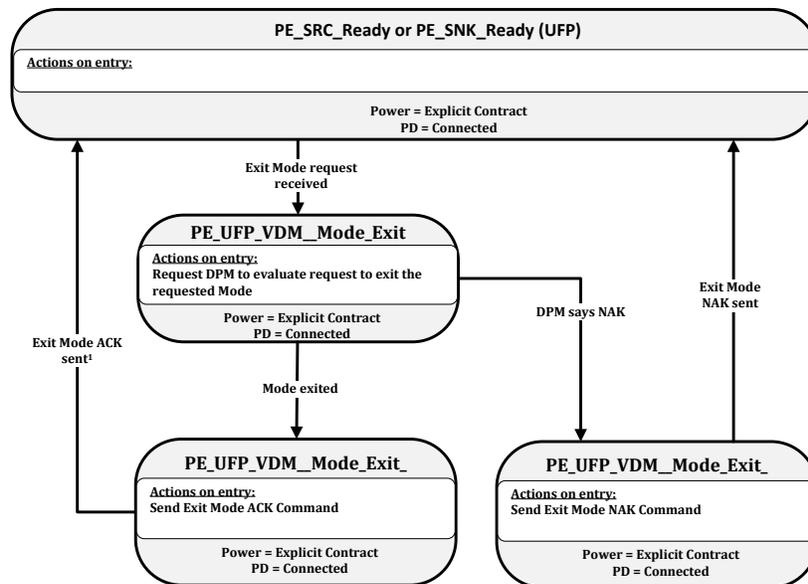
The Policy Engine **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state for a UFP when:

- The Structured VDM **Enter Mode** NAK Command response has been sent.

8.3.3.8.5 UFP Structured VDM Exit Mode State Diagram

Figure 8-67 shows the state diagram for a UFP in response to an **Exit Mode** Command.

Figure 8-67 UFP Structured VDM Exit Mode State Diagram



¹ The UFP is required to be in USB operation or USB Safe State at this point.

8.3.3.8.5.1 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 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** Command request is NAK.

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.

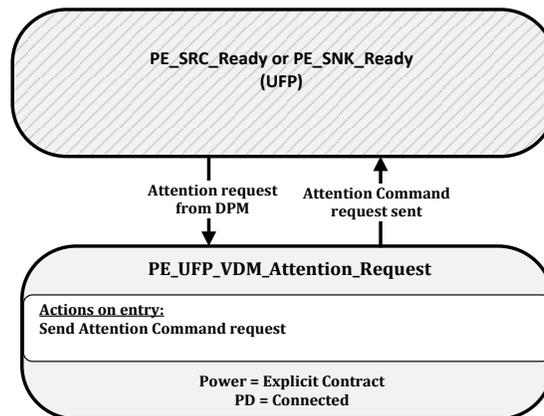
The Policy Engine **Shall** transition to either the either the **PE_SRC_Ready** or **PE_SNK_Ready** state for a UFP when:

- The Structured VDM **Exit Mode** NAK Command response has been sent.

8.3.3.8.6 UFP Structured VDM Attention State Diagram

Figure 8-75 shows the state diagram for a UFP when sending an **Attention** Command request.

Figure 8-68 UFP VDM Attention State Diagram



8.3.3.8.6.1 PE_UFP_VDM_Attention_Request state

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 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 UFP when:

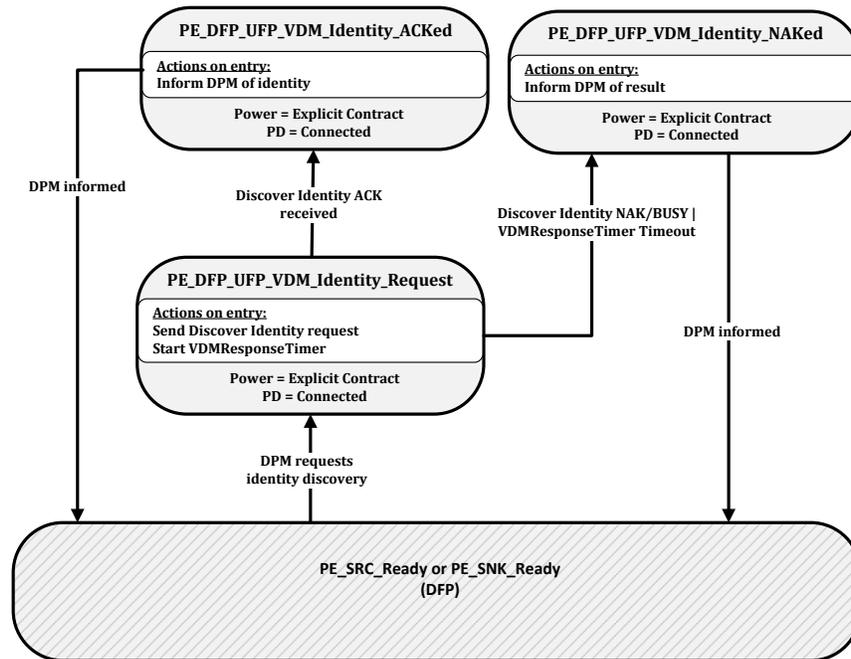
- The **Attention** Command request has been sent.

8.3.3.9 DFP Structured VDM State Diagrams

8.3.3.9.1 DFP to UFP Structured VDM Discover Identity State Diagram

Figure 8-69 shows the state diagram for a DFP when discovering the identity of a UFP.

Figure 8-69 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

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.

8.3.3.9.1.3 PE_DFP_VDM_Identity_NAKed state

On entry to the *PE_DFP_UFP_VDM_Identity_NAKed* state the Policy Engine **Shall** inform the Device Policy Manager of the result (NAK, BUSY or timeout).

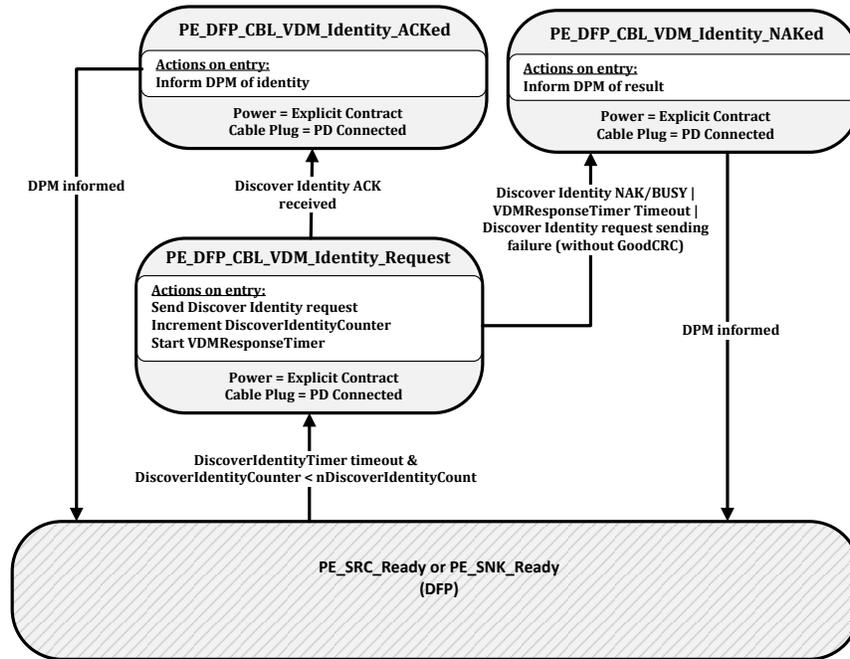
The Policy Engine **Shall** transition to either the *PE_SRC_Ready* or *PE_SNK_Ready* state for a DFP when:

- The Device Policy Manager has been informed.

8.3.3.9.2 DFP to Cable Plug Structured VDM Discover Identity State Diagram

Figure 8-69 shows the state diagram for a DFP when discovering the identity of a Cable Plug.

Figure 8-70 DFP VDM Discover Identity State Diagram



8.3.3.9.2.1 PE_DFP_VDM_Identity_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 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.

8.3.3.9.2.3 PE_DFP_VDM_Identity_NAKed state

On entry to the **PE_DFP_CBL_VDM_Identity_NAKed** state the Policy Engine **Shall** inform the Device Policy Manager of the result (NAK, BUSY or timeout).

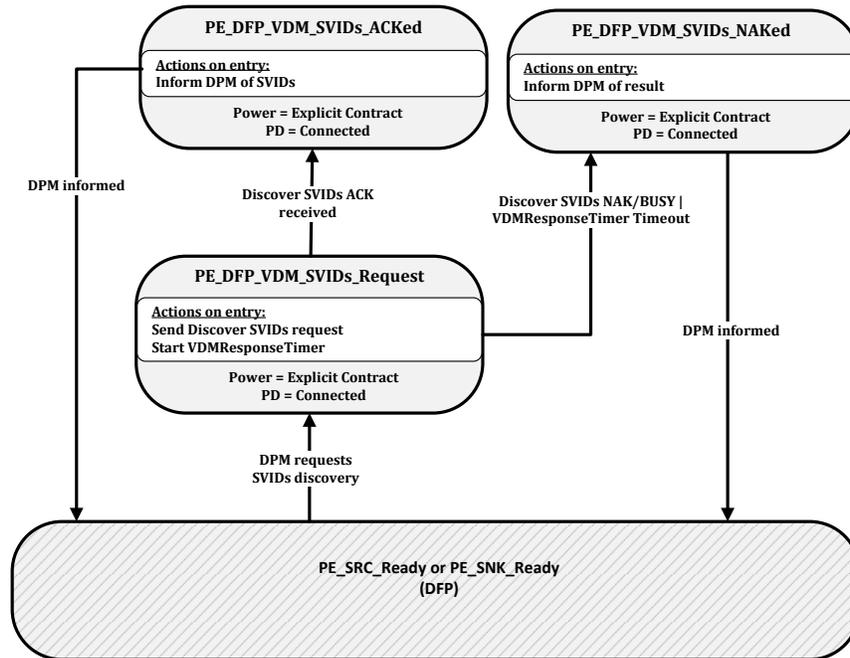
The Policy Engine **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state for a DFP when:

- The Device Policy Manager has been informed.

8.3.3.9.3 DFP Structured VDM Discover SVIDs State Diagram

Figure 8-71 shows the state diagram for a DFP when discovering SVIDs.

Figure 8-71 DFP VDM Discover SVIDs State Diagram



8.3.3.9.3.1 PE_DFP_VDM_SVIDs_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 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).

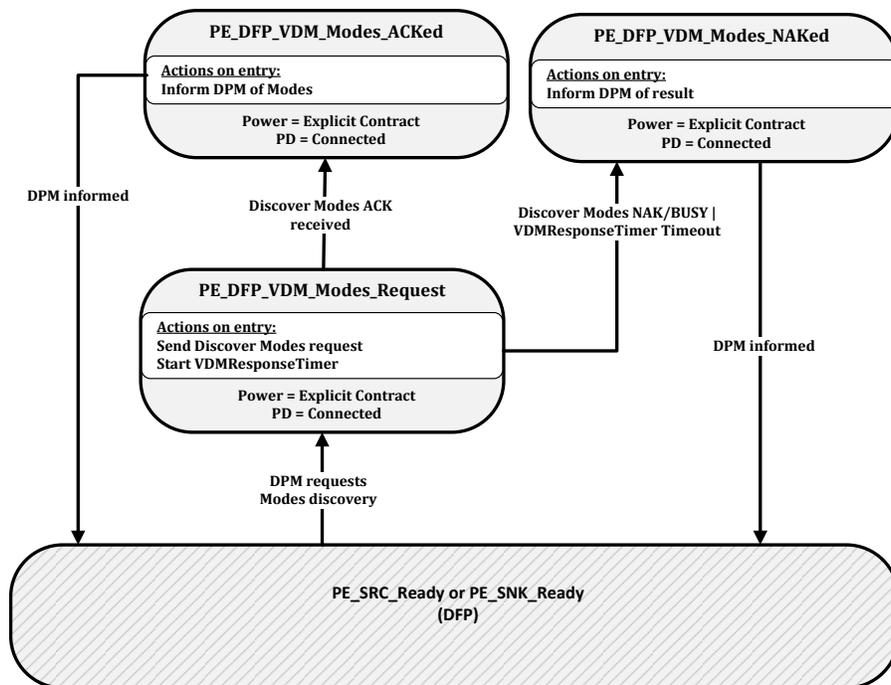
The Policy Engine **Shall** transition to either the *PE_SRC_Ready* or *PE_SNK_Ready* state for a DFP when:

- The Device Policy Manager has been informed.

8.3.3.9.4 DFP Structured VDM Discover Modes State Diagram

Figure 8-72 shows the state diagram for a DFP when discovering Modes.

Figure 8-72 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).

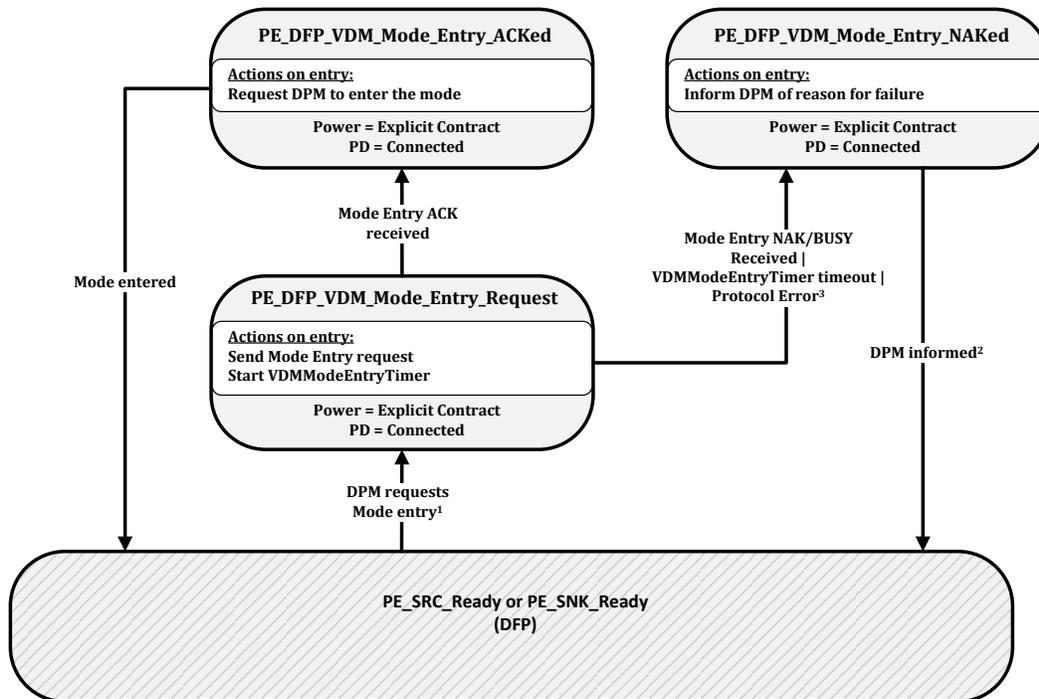
The Policy Engine **Shall** transition to either the *PE_SRC_Ready* or *PE_SNK_Ready* state for a DFP when:

- The Device Policy Manager has been informed.

8.3.3.9.5 DFP Structured VDM Mode Entry State Diagram

Figure 8-73 shows the state operation for a DFP when entering a Mode.

Figure 8-73 DFP VDM Mode Entry State Diagram



¹ The Device Policy Manager **Shall** have placed the system into USB Safe State before issuing this request when entering Modal operation.

² The Device Policy Manager **Shall** have returned the system to USB operation if not in Modal operation at this point.

³ Protocol Errors are handled by informing the DPM, returning to USB Safe State and then processing the Message once the **PE_SRC_Ready** or **PE_SNK_Ready** state has been entered.

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.

8.3.3.9.5.2 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.

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).

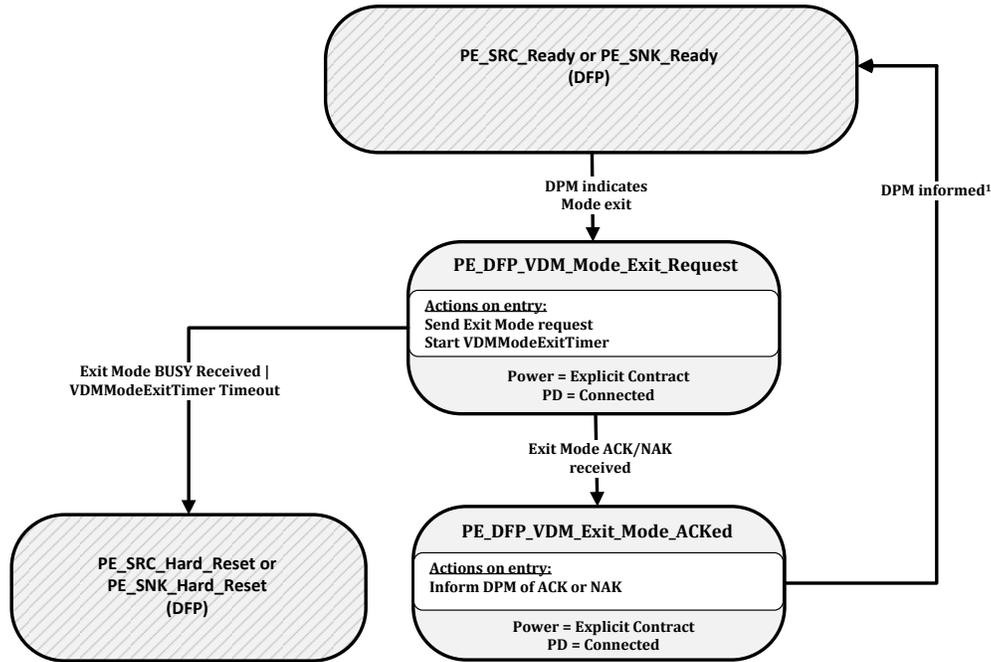
The Policy Engine **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state for a DFP when:

- The Device Policy Manager has been informed.

8.3.3.9.6 DFP Structured VDM Mode Exit State Diagram

Figure 8-74 shows the state diagram for a DFP when exiting a Mode.

Figure 8-74 DFP VDM Mode Exit State Diagram



¹ The Device Policy Manager is required to return the system to USB operation at this point when exiting Modal Operation.

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* BUSY Command response is received or
- The *VDMModeExitTimer* times out.

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.

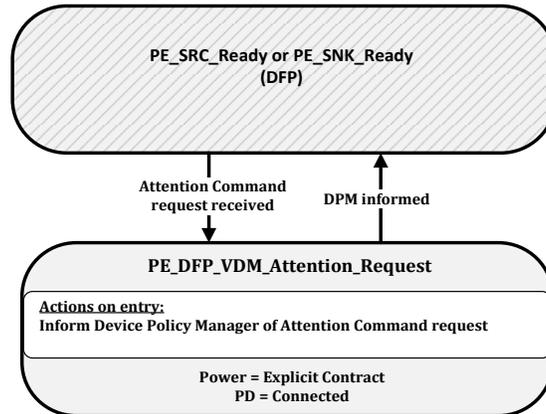
The Policy Engine **Shall** transition to either the *PE_SRC_Ready* or *PE_SNK_Ready* state for a DFP when:

- The Device Policy Manager has been informed.

8.3.3.9.7 DFP Structured VDM Attention State Diagram

Figure 8-75 shows the state diagram for a DFP when receiving an *Attention* Command request.

Figure 8-75 DFP VDM Attention State Diagram



8.3.3.9.7.1 PE_DFP_VDM_Attention_Request state

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.

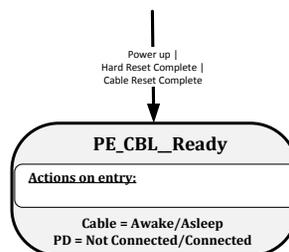
8.3.3.10 Cable Plug Related State Diagrams

The State Diagrams in this section *Shall* apply to all Cable Plugs that support Structured VDMs.

8.3.3.10.1 Cable Plug Cable Ready State Diagram

Figure 8-76 shows the Cable Ready state diagram for a Cable Plug.

Figure 8-76 Cable Ready VDM State Diagram



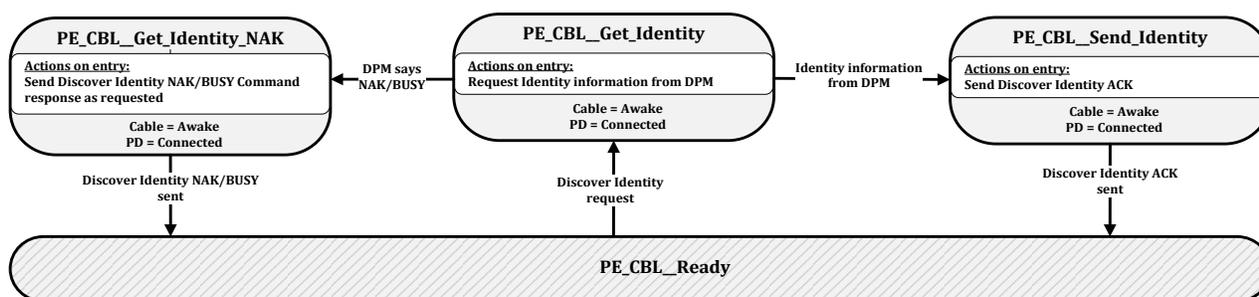
8.3.3.10.1.1 PE_CBL_Ready state

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 8-77 shows the state diagram for a Cable Plug in response to a *Discover Identity* Command.

Figure 8-77 Cable Plug Structured VDM Discover Identity State Diagram



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.

8.3.3.10.2.2 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 state

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.

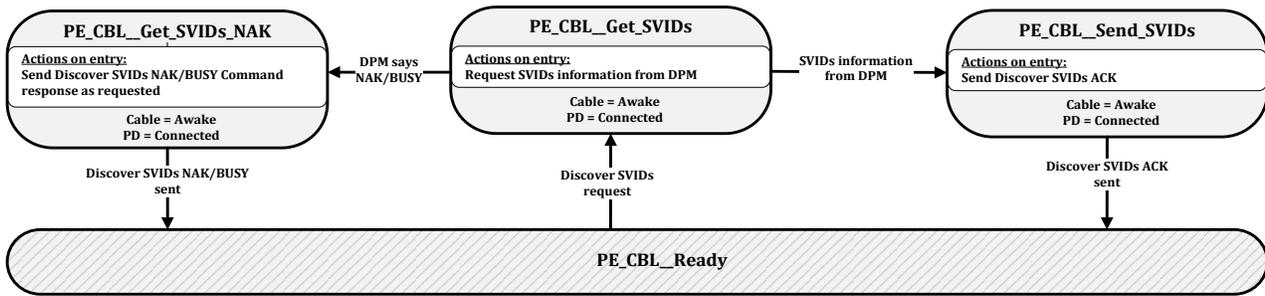
The Policy Engine *Shall* transition to the *PE_CBL_Ready* state when:

- 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 8-78 shows the state diagram for a Cable Plug in response to a *Discover SVIDs* Command.

Figure 8-78 Cable Plug Structured VDM Discover SVIDs State Diagram



8.3.3.10.3.1 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.

8.3.3.10.3.2 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 state

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.

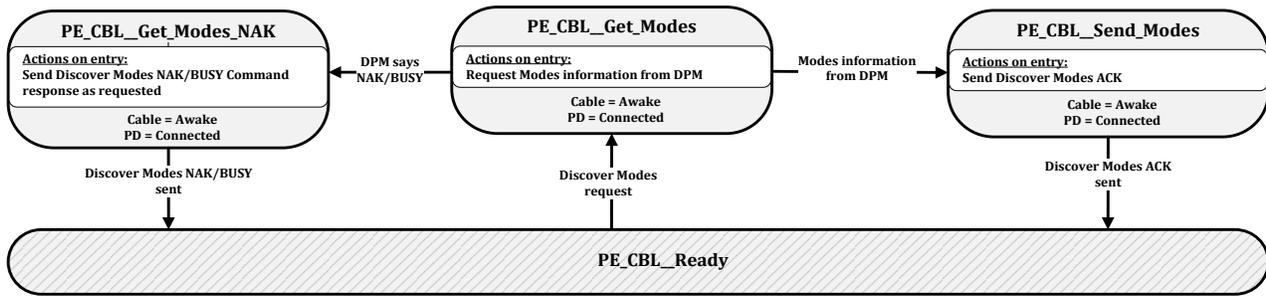
The Policy Engine *Shall* transition to the *PE_CBL_Ready* state when:

- 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 8-79 shows the state diagram for a Cable Plug in response to a *Discover Modes* Command.

Figure 8-79 Cable Plug Structured VDM Discover Modes State Diagram



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.

8.3.3.10.4.2 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 state

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.

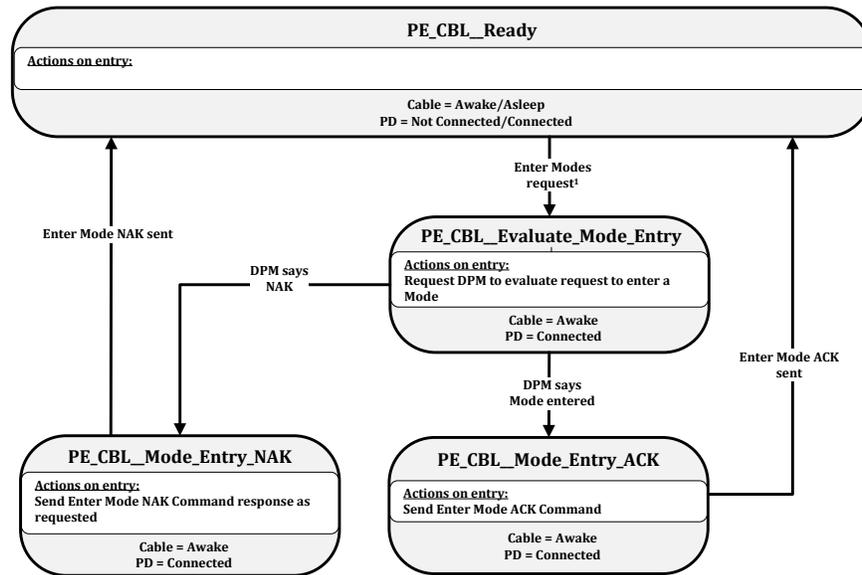
The Policy Engine *Shall* transition to the *PE_CBL_Ready* state when:

- 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 8-80 shows the state diagram for a Cable Plug in response to an *Enter Mode* Command.

Figure 8-80 Cable Plug Structured VDM Enter Mode State Diagram



¹ The Cable is required to be in USB operation or USB Safe State at this point.

8.3.3.10.5.1 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.

8.3.3.10.5.2 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.

8.3.3.10.5.3 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.

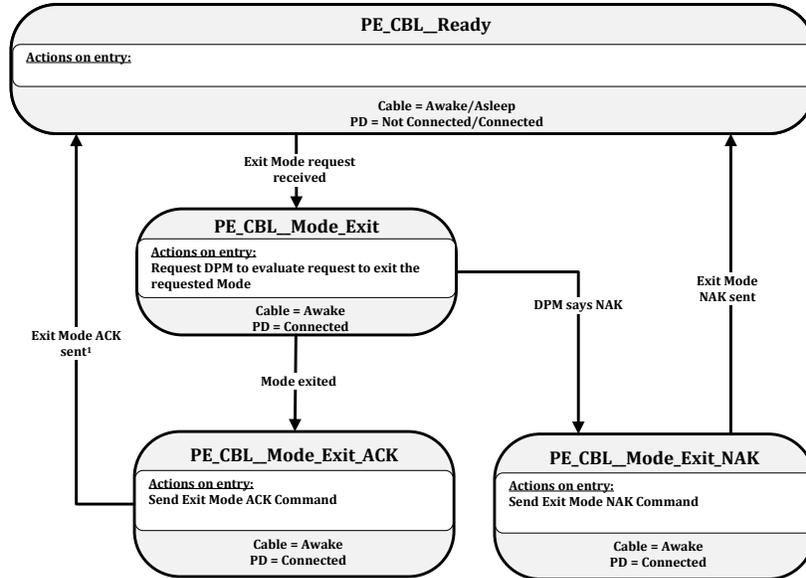
The Policy Engine **Shall** transition to the **PE_CBL_Ready** state when:

- 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 8-81 shows the state diagram for a Cable Plug in response to an *Exit Mode* Command.

Figure 8-81 Cable Plug Structured VDM Exit Mode State Diagram



¹ The Cable is required to be in USB operation or USB Safe State at this point.

8.3.3.10.6.1 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 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.

8.3.3.10.6.2 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_CBL_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.

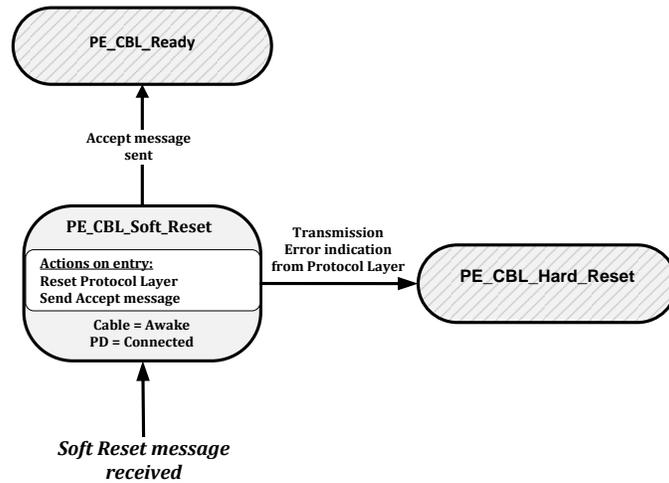
The Policy Engine **Shall** transition to the *PE_CBL_Ready* state when:

- The Structured VDM *Exit Mode* NAK Command response has been sent.

8.3.3.10.7 Cable Plug Soft Reset State Diagram

Figure 8-82 shows the Cable Plug state diagram on reception of a *Soft_Reset* Message.

Figure 8-82 Cable Plug Soft Reset State Diagram



8.3.3.10.7.1 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.

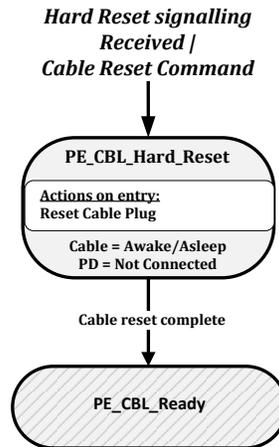
The Policy Engine **Shall** transition to the *PE_CBL_Hard_Reset* state when:

- The Protocol Layer indicates that a transmission error has occurred.

8.3.3.10.8 Cable Plug Hard Reset State Diagram

Figure 8-83 shows the Cable Plug state diagram for a Hard Reset or Cable Reset.

Figure 8-83 Cable Plug Hard Reset State Diagram



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).

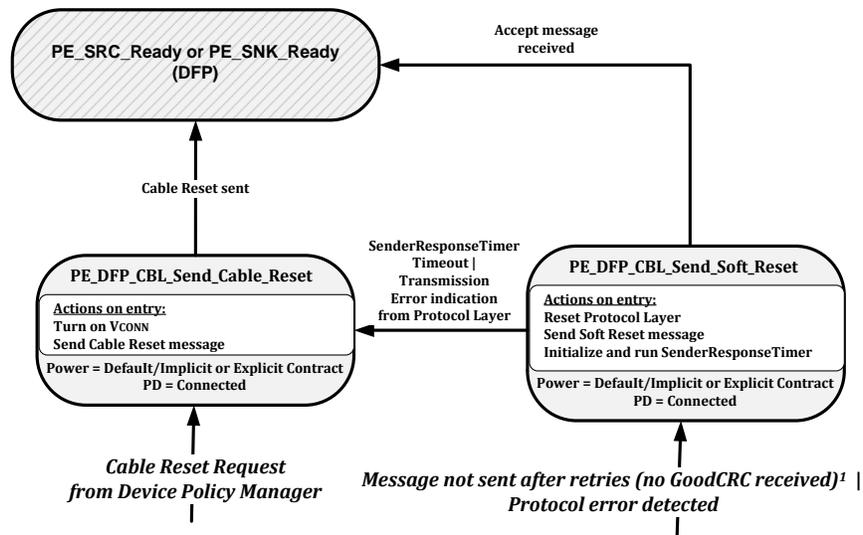
The Policy Engine *shall* transition to the **PE_CBL_Ready** state when:

- The Cable Plug reset is complete.

8.3.3.10.9 DFP Soft Reset or Cable Reset of a Cable Plug State Diagram

Figure 8-87 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 8-84 DFP Soft Reset or Cable Reset of a Cable Plug State Diagram



¹ 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.

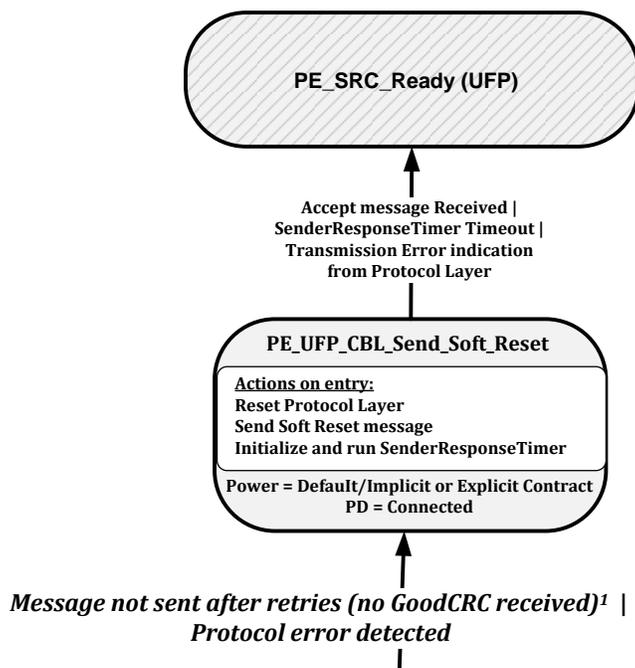
The Policy Engine **Shall** transition to either the *PE_SRC_Ready* or *PE_SNK_Ready* state, depending on the DFP's Power Role, when:

- *Cable Reset* Signaling has been sent.

8.3.3.10.10 UFP Source Soft Reset of a Cable Plug State Diagram

Figure 8-87 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 8-85 UFP Source Soft Reset of a Cable Plug State Diagram



¹ 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**.

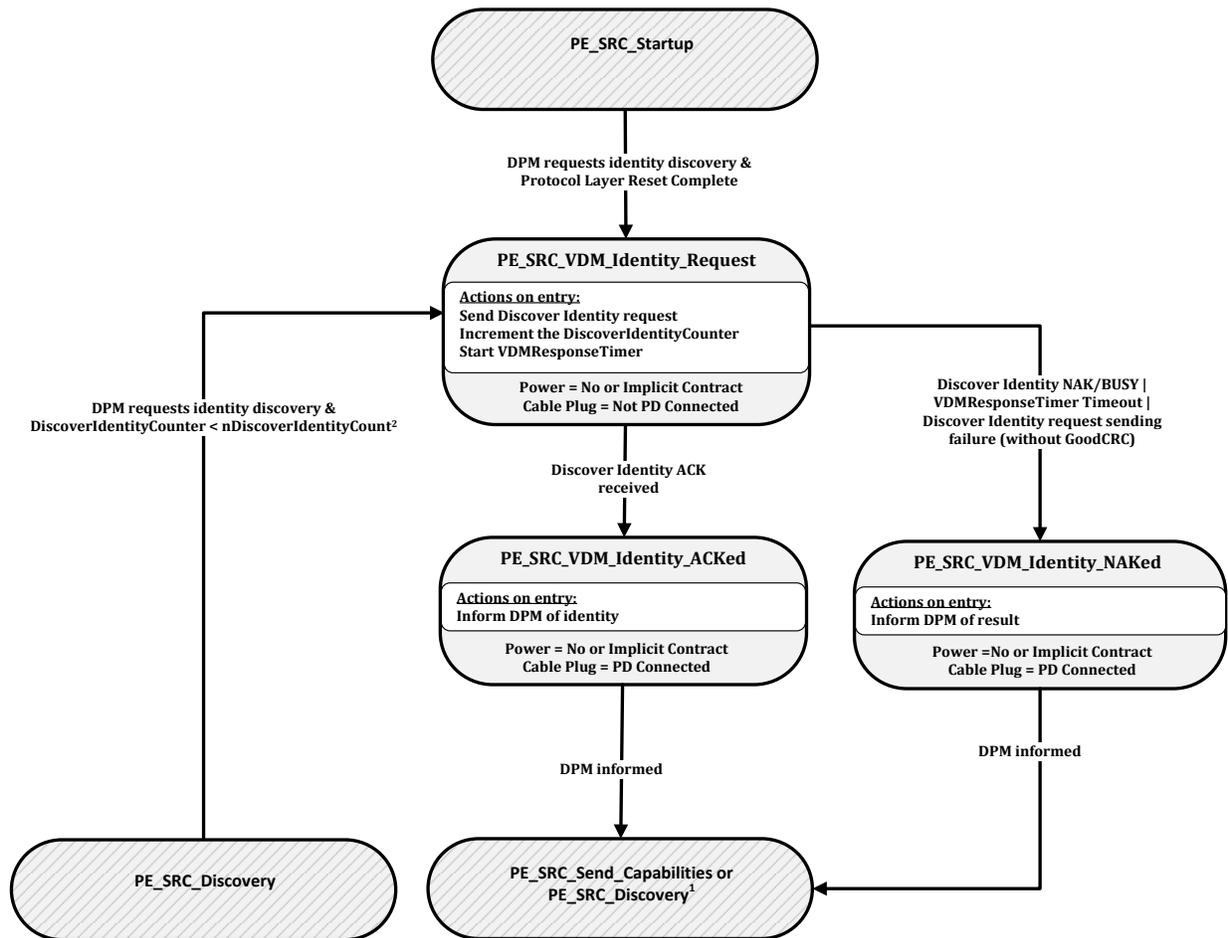
The Policy Engine **Shall** transition to the **PE_SRC_Ready** state when:

- 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 8-86 shows the state diagram for Source discovery of identity information from a Cable Plug during the startup sequence.

Figure 8-86 Source Startup Structured VDM Discover Identity State Diagram



¹ If the *Discover Identity* REQ Command is being sent at startup then the Policy Engine will subsequently transition to the *PE_SRC_Send_Capabilities* state as normal. Otherwise the Policy Engine will transition to the *PE_SRC_Discovery* state.

² The *SourceCapabilityTimer* 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.

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*.

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 8-86 and **Shall Not** be initialized on re-entry to *PE_SRC_Discovery*.

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:

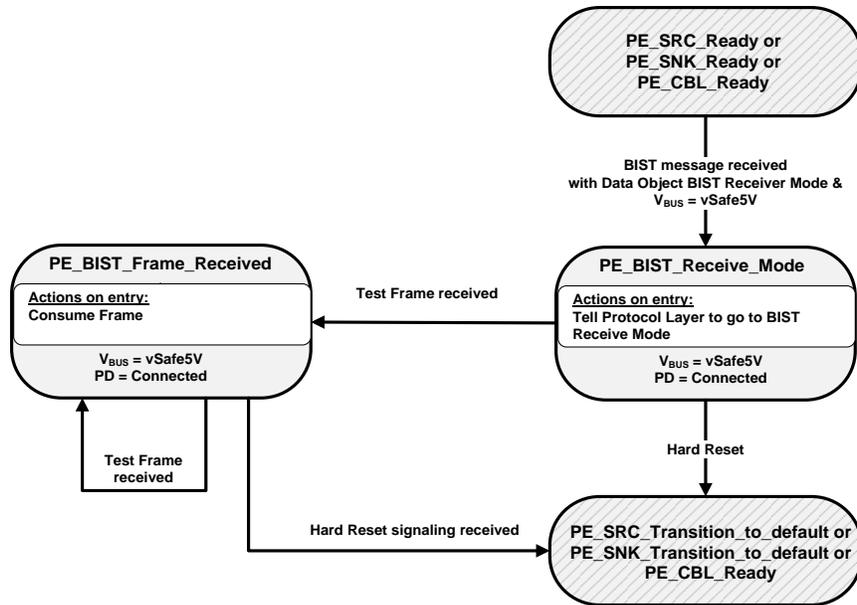
The Device Policy Manager has been informed.

8.3.3.11 BIST State diagrams

8.3.3.11.1 BIST Receive Mode State Diagram

Figure 8-87 shows the state diagram required by a UUT, which can be either a Source, 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*, *PE_SNK_Ready* or *PE_CBL_Ready* state. See Section 5.9.9 for which tests are applicable to BFSK and BMC.

Figure 8-87 BIST Receive Mode State Diagram



8.3.3.11.1.1 PE_BIST_Receive_Mode state

The Source, Sink or Cable Plug **Shall** enter the **PE_BIST_Receive_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 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, **PE_SNK_Transition_to_default** state or **PE_CBL_Ready** state (as appropriate) when:

- **Hard Reset** Signaling is received.

8.3.3.11.1.2 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.

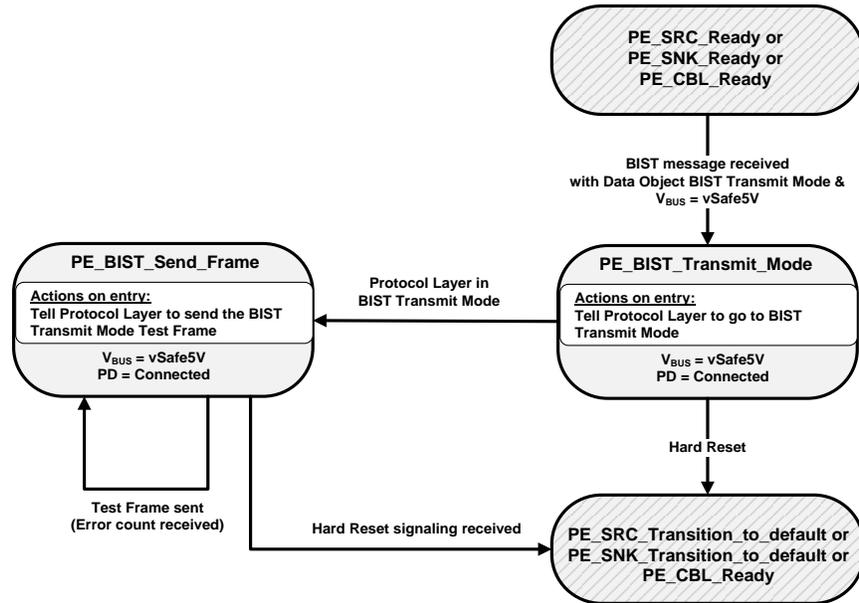
The Policy Engine **Shall** transition to either the **PE_SRC_Transition_to_default** state, **PE_SNK_Transition_to_default** state or **PE_CBL_Ready** state (as appropriate) when:

- **Hard Reset** Signaling is received.

8.3.3.11.2 BIST Transmit Mode State Diagram

Figure 8-88 shows the state diagram required by a UUT that can be either a Source, 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*, *PE_SNK_Ready* or *PE_CBL_Ready* state. See Section 5.9.9 for which tests are applicable to BFSK and BMC.

Figure 8-88 BIST Transmit Mode State Diagram



8.3.3.11.2.1 PE_BIST_Transmit_Mode state

The Source, Sink or Cable Plug **shall** enter the *PE_BIST_Transmit_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 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, *PE_SNK_Transition_to_default* state or *PE_CBL_Ready* state (as appropriate) when:

- *Hard Reset* Signaling is received.

8.3.3.11.2.2 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).

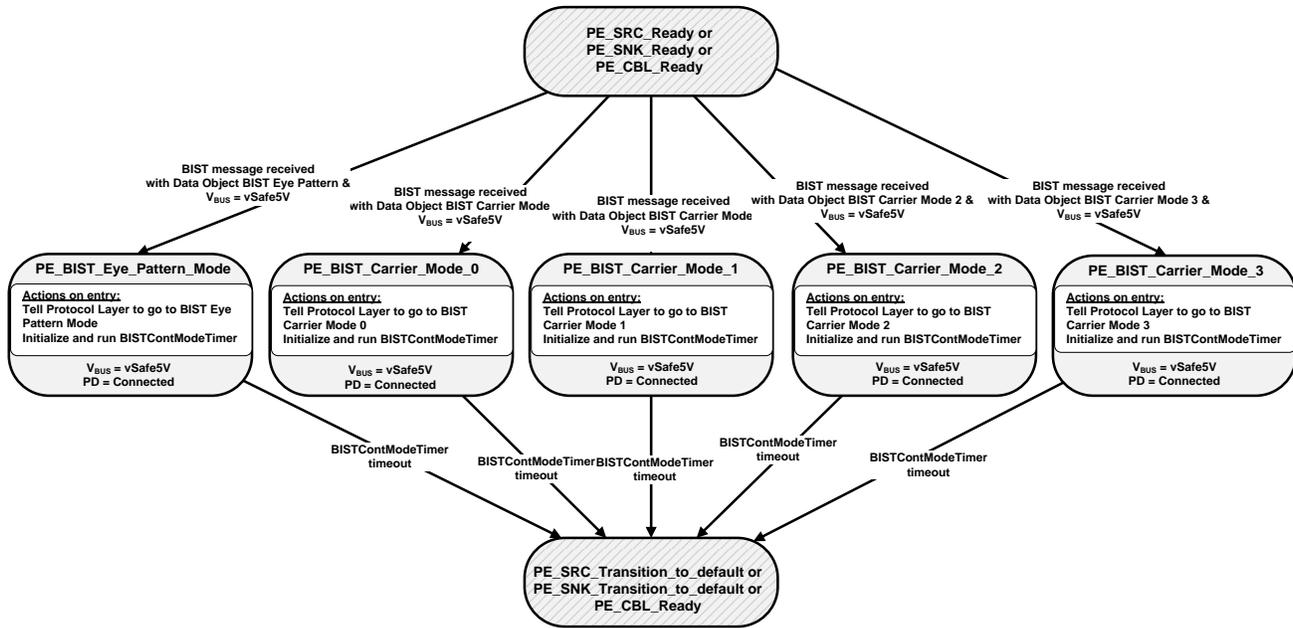
The Policy Engine **shall** transition to either the *PE_SRC_Transition_to_default* state, *PE_SNK_Transition_to_default* state or *PE_CBL_Ready* state (as appropriate) when:

- **Hard Reset** Signaling is received.

8.3.3.11.3 BIST Carrier Mode and Eye Pattern State Diagram

Figure 8-89 shows the state diagram required by a UUT, which can be either a Source, 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**, **PE_SNK_Ready** or **PE_CBL_Ready** states. See Section 5.9.9 for which tests are applicable to BFSK and BMC.

Figure 8-89 BIST Carrier Mode and Eye Pattern State Diagram



8.3.3.11.3.1 PE_BIST_Eye_Pattern_Mode state

The Source, Sink or 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, **PE_SNK_Transition_to_default** state or **PE_CBL_Ready** state (as appropriate) when:

- The **BISTContModeTimer** times out.

8.3.3.11.3.2 PE_BIST_Carrier_Mode_0 state

The Source, Sink or 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:

- 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 Mode 0 and **Shall** initialize and run the **BISTContModeTimer**.

The Policy Engine **Shall** transition to either the **PE_SRC_Transition_to_default** state, **PE_SNK_Transition_to_default** state or **PE_CBL_Ready** state (as appropriate) when:

- The **BISTContModeTimer** times out.

8.3.3.11.3.3 PE_BIST_Carrier_Mode_1 state

The Source, Sink or Cable 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:

- 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 Mode 1 and **Shall** initialize and run the **BISTContModeTimer**.

The Policy Engine **Shall** transition to either the **PE_SRC_Transition_to_default** state, **PE_SNK_Transition_to_default** state or **PE_CBL_Ready** state (as appropriate) when:

- The **BISTContModeTimer** times out.

8.3.3.11.3.4 PE_BIST_Carrier_Mode_2 state

The Source, Sink or Cable 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:

- 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 Mode 2 and **Shall** initialize and run the **BISTContModeTimer**.

The Policy Engine **Shall** transition to either the **PE_SRC_Transition_to_default** state, **PE_SNK_Transition_to_default** state or **PE_CBL_Ready** state (as appropriate) when:

- The **BISTContModeTimer** times out.

8.3.3.11.3.5 PE_BIST_Carrier_Mode_3 state

The Source, Sink or Cable 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:

- 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, **PE_SNK_Transition_to_default** state or **PE_CBL_Ready** state (as appropriate) when:

- The **BISTContModeTimer** times out.

8.3.3.12 Type-C Referenced States

This section contains states cross-referenced from the [\[USB Type-C 1.2\]](#) specification.

8.3.3.12.1 ErrorRecovery state

The **ErrorRecovery** state is used to electronically disconnect Port Partners using the USB Type-C connector. The **ErrorRecovery** state **Shall** be entered when there are errors on USB Type-C Ports which cannot be recovered by Hard Reset. The **ErrorRecovery** state **Shall** map to USB Type-C ErrorRecovery state operation as defined in the [\[USB Type-C 1.2\]](#) specification, including any other state transitions mandated in cases where USB 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.

8.3.3.13 Policy Engine States

Table 8-39 lists the states used by the various state machines.

Table 8-39 Policy Engine States

State name	Reference
Source Port	
<i>PE_SRC_Startup</i>	Section 8.3.3.2.1
<i>PE_SRC_Discovery</i>	Section 8.3.3.2.2
<i>PE_SRC_Send_Capabilities</i>	Section 8.3.3.2.3
<i>PE_SRC_Negotiate_Capability</i>	Section 8.3.3.2.4
<i>PE_SRC_Transition_Supply</i>	Section 8.3.3.2.5
<i>PE_SRC_Ready</i>	Section 8.3.3.2.6
<i>PE_SRC_Disabled</i>	Section 8.3.3.2.7
<i>PE_SRC_Capability_Response</i>	Section 8.3.3.2.8
<i>PE_SRC_Hard_Reset</i>	Section 8.3.3.2.9
<i>PE_SRC_Hard_Reset_Received</i>	Section 8.3.3.2.10
<i>PE_SRC_Transition_to_default</i>	Section 8.3.3.2.11
<i>PE_SRC_Get_Sink_Cap</i>	Section 8.3.3.2.12
<i>PE_SRC_Wait_New_Capabilities</i>	Section 8.3.3.2.13
Sink Port	
<i>PE_SNK_Startup</i>	Section 8.3.3.3.1
<i>PE_SNK_Discovery</i>	Section 8.3.3.3.2
<i>PE_SNK_Wait_for_Capabilities</i>	Section 8.3.3.3.3
<i>PE_SNK_Evaluate_Capability</i>	Section 8.3.3.3.4
<i>PE_SNK_Select_Capability</i>	Section 8.3.3.3.5
<i>PE_SNK_Transition_Sink</i>	Section 8.3.3.3.6
<i>PE_SNK_Ready</i>	Section 8.3.3.3.7
<i>PE_SNK_Hard_Reset</i>	Section 8.3.3.3.8
<i>PE_SNK_Transition_to_default</i>	Section 8.3.3.3.9
<i>PE_SNK_Give_Sink_Cap</i>	Section 8.3.3.3.10
<i>PE_SNK_Get_Source_Cap</i>	Section 8.3.3.3.11
Source Port Soft Reset	
<i>PE_SRC_Send_Soft_Reset</i>	Section 8.3.3.4.1.1
<i>PE_SRC_Soft_Reset</i>	Section 8.3.3.4.1.2
Sink Port Soft Reset	
<i>PE_SNK_Send_Soft_Reset</i>	Section 8.3.3.4.2.1
<i>PE_SNK_Soft_Reset</i>	Section 8.3.3.4.2.2
Source Port Ping	
<i>PE_SRC_Ping</i>	Section 8.3.3.5.1
Type-A/B Dual-Role (initially Source Port) Ping	
<i>PE_PRS_SRC_SNK_Ping</i>	Section 8.3.3.6.1.1.1

State name	Reference
Type-A/B Dual-Role (initially Sink Port) Ping	
<i>PE_PRS_SNK_SRC_Ping</i>	Section 8.3.3.6.1.2.1
Type-A/B Hard Reset of P/C in Sink Role	
<i>PE_PC_SNK_Hard_Reset</i>	Section 8.3.3.6.1.3.1
<i>PE_PC_SNK_Swap_Recovery</i>	Section 8.3.3.6.1.3.2
Type-A/B Hard Reset of C/P in Source Role	
<i>PE_CP_SRC_Hard_Reset</i>	Section 8.3.3.6.1.4.1
<i>PE_CP_SRC_Transition_to_off</i>	Section 8.3.3.6.1.4.2
Type-A/B C/P Dead Battery/Power Loss	
<i>PE_DB_CP_Check_for_VBUS</i>	Section 8.3.3.6.1.5.1
<i>PE_DB_CP_Power_VBUS_DB</i>	Section 8.3.3.6.1.5.2
<i>PE_DB_CP_Wait_For_Bit_Stream</i>	Section 8.3.3.6.1.5.3
<i>PE_DB_CP_Power_VBUS_5V</i>	Section 8.3.3.6.1.5.4
<i>PE_DB_CP_Wait_Bit_Stream_Stop</i>	Section 8.3.3.6.1.5.5
<i>PE_DB_CP_Unpower_VBUS</i>	Section 8.3.3.6.1.5.6
<i>PE_DB_CP_PS_Discharge</i>	Section 8.3.3.6.1.5.7
Type-A/B P/C Dead Battery/Power Loss	
<i>PE_DB_PC_Unpowered</i>	Section 8.3.3.6.1.6.1
<i>PE_DB_PC_Check_Power</i>	Section 8.3.3.6.1.6.2
<i>PE_DB_PC_Send_Bit_Stream</i>	Section 8.3.3.6.1.6.3
<i>PE_DB_PC_Wait_to_Detect</i>	Section 8.3.3.6.1.6.4
<i>PE_DB_PC_Wait_to_Start</i>	Section 8.3.3.6.1.6.5
Type-C DFP to UFP Data Role Swap	
<i>PE_DRS_DFP_UFP_Evaluate_DR_Swap</i>	Section 8.3.3.6.2.1.2
<i>PE_DRS_DFP_UFP_Accept_DR_Swap</i>	Section 8.3.3.6.2.1.3
<i>PE_DRS_DFP_UFP_Change_to_UFP</i>	Section 8.3.3.6.2.1.4
<i>PE_DRS_DFP_UFP_Send_DR_Swap</i>	Section 8.3.3.6.2.1.5
<i>PE_DRS_DFP_UFP_Reject_DR_Swap</i>	Section 8.3.3.6.2.1.6
Type-C UFP to DFP Data Role Swap	
<i>PE_DRS_UFP_DFP_Evaluate_DR_Swap</i>	Section 8.3.3.6.2.2.2
<i>PE_DRS_UFP_DFP_Accept_DR_Swap</i>	Section 8.3.3.6.2.2.3
<i>PE_DRS_UFP_DFP_Change_to_DFP</i>	Section 8.3.3.6.2.2.4
<i>PE_DRS_UFP_DFP_Send_DR_Swap</i>	Section 8.3.3.6.2.2.5
<i>PE_DRS_UFP_DFP_Reject_DR_Swap</i>	Section 8.3.3.6.2.2.6
Source to Sink Power Role Swap	
<i>PE_PRS_SRC_SNK_Evaluate_PR_Swap</i>	Section 8.3.3.6.3.1.2
<i>PE_PRS_SRC_SNK_Accept_PR_Swap</i>	Section 8.3.3.6.3.1.3
<i>PE_PRS_SRC_SNK_Transition_to_off</i>	Section 8.3.3.6.3.1.4
<i>PE_PRS_SRC_SNK_Assert_Rd</i>	Section 8.3.3.6.3.1.5
<i>PE_PRS_SRC_SNK_Wait_Source_on</i>	Section 8.3.3.6.3.1.6

State name	Reference
<i>PE_PRS_SRC_SNK_Send_PR_Swap</i>	Section 8.3.3.6.3.1.7
<i>PE_PRS_SRC_SNK_Reject_PR_Swap</i>	Section 8.3.3.6.3.1.8
Sink to Source Power Role Swap	
<i>PE_PRS_SNK_SRC_Evaluate_PR_Swap</i>	Section 8.3.3.6.3.2.2
<i>PE_PRS_SNK_SRC_Accept_PR_Swap</i>	Section 8.3.3.6.3.2.3
<i>PE_PRS_SNK_SRC_Transition_to_off</i>	Section 8.3.3.6.3.2.4
<i>PE_PRS_SNK_SRC_Assert_Rp</i>	
<i>PE_PRS_SNK_SRC_Source_on</i>	Section 8.3.3.6.3.2.5
<i>PE_PRS_SNK_SRC_Send_PR_Swap</i>	Section 8.3.3.6.3.2.7
<i>PE_PRS_SNK_SRC_Reject_PR_Swap</i>	Section 8.3.3.6.3.2.8
Dual-Role Source Port Get Source Capabilities	
<i>PE_DR_SRC_Get_Source_Cap</i>	Section 8.3.3.6.3.3.1
Dual-Role Source Port Give Sink Capabilities	
<i>PE_DR_SRC_Give_Sink_Cap</i>	Section 8.3.3.6.3.4.1
Dual-Role Sink Port Get Sink Capabilities	
<i>PE_DR_SNK_Get_Sink_Cap</i>	Section 8.3.3.6.3.5.1
Dual-Role Sink Port Give Source Capabilities	
<i>PE_DR_SNK_Give_Source_Cap</i>	Section 8.3.3.6.3.6.1
Type-C VCONN Swap	
<i>PE_VCS_Send_Swap</i>	Section 8.3.3.7.1.1
<i>PE_VCS_Evaluate_Swap</i>	Section 8.3.3.7.1.2
<i>PE_VCS_Accept_Swap</i>	Section 8.3.3.7.1.3
<i>PE_VCS_Reject_Swap</i>	Section 8.3.3.7.1.4
<i>PE_VCS_Wait_For_VCONN</i>	Section 8.3.3.7.1.5
<i>PE_VCS_Turn_Off_VCONN</i>	Section 8.3.3.7.1.6
<i>PE_VCS_Turn_On_VCONN</i>	Section 8.3.3.7.1.7
<i>PE_VCS_Send_Ps_Rdy</i>	Section 8.3.3.7.1.8
UFP Structured VDM	
UFP Structured VDM Discovery Identity	
<i>PE_UFP_VDM_Get_Identity</i>	Section 8.3.3.8.1.1
<i>PE_UFP_VDM_Send_Identity</i>	Section 8.3.3.8.1.2
<i>PE_UFP_VDM_Get_Identity_NAK</i>	Section 8.3.3.8.1.3
UFP Structured VDM Discovery SVIDs	
<i>PE_UFP_VDM_Get_SVIDs</i>	Section 8.3.3.8.2.1
<i>PE_UFP_VDM_Send_SVIDs</i>	Section 8.3.3.8.2.2
<i>PE_UFP_VDM_Get_SVIDs_NAK</i>	Section 8.3.3.8.2.3
UFP Structured VDM Discovery Modes	
<i>PE_UFP_VDM_Get_Modes</i>	Section 8.3.3.8.3.1
<i>PE_UFP_VDM_Send_Modes</i>	Section 8.3.3.8.3.2
<i>PE_UFP_VDM_Get_Modes_NAK</i>	Section 8.3.3.8.3.3

State name	Reference
UFP Structured VDM Enter Mode	
<i>PE_UFP_VDM_Evaluate_Mode_Entry</i>	Section 8.3.3.8.4.1
<i>PE_UFP_VDM_Mode_Entry_ACK</i>	Section 8.3.3.8.4.2
<i>PE_UFP_VDM_Mode_Entry_NAK</i>	Section 8.3.3.8.4.3
UFP Structured VDM Exit Mode	
<i>PE_UFP_VDM_Mode_Exit</i>	Section 8.3.3.8.5.1
<i>PE_UFP_VDM_Mode_Exit_ACK</i>	Section 8.3.3.8.5.2
<i>PE_UFP_VDM_Mode_Exit_NAK</i>	Section 8.3.3.8.5.3
UFP Structured VDM Attention	
<i>PE_UFP_VDM_Attention_Request</i>	Section 8.3.3.8.6.1
DFP Structured VDM	
DFP to UFP Structured VDM Discover Identity	
<i>PE_DFP_UFP_VDM_Identity_Request</i>	Section 8.3.3.9.1.1
<i>PE_DFP_UFP_VDM_Identity_ACKed</i>	Section 8.3.3.9.1.2
<i>PE_DFP_UFP_VDM_Identity_NAKed</i>	Section 8.3.3.9.1.3
DFP to Cable Plug Structured VDM Discover Identity	
<i>PE_DFP_CBL_VDM_Identity_Request</i>	Section 8.3.3.9.2.1
<i>PE_DFP_CBL_VDM_Identity_ACKed</i>	Section 8.3.3.9.2.2
<i>PE_DFP_CBL_VDM_Identity_NAKed</i>	Section 8.3.3.9.2.3
DFP Structured VDM Discover SVIDs	
<i>PE_DFP_VDM_SVIDs_Request</i>	Section 8.3.3.9.3.1
<i>PE_DFP_VDM_SVIDs_ACKed</i>	Section 8.3.3.9.3.2
<i>PE_DFP_VDM_SVIDs_NAKed</i>	Section 8.3.3.9.3.3
DFP Structured VDM Discover Modes	
<i>PE_DFP_VDM_Modes_Request</i>	Section 8.3.3.9.4.1
<i>PE_DFP_VDM_Modes_ACKed</i>	Section 8.3.3.9.4.2
<i>PE_DFP_VDM_Modes_NAKed</i>	Section 8.3.3.9.4.3
DFP Structured VDM Mode Entry	
<i>PE_DFP_VDM_Mode_Entry_Request</i>	Section 8.3.3.9.5.1
<i>PE_DFP_VDM_Mode_Entry_ACKed</i>	Section 8.3.3.9.5.2
<i>PE_DFP_VDM_Mode_Entry_NAKed</i>	Section 8.3.3.9.5.3
DFP Structured VDM Mode Exit	
<i>PE_DFP_VDM_Mode_Exit_Request</i>	Section 8.3.3.9.6.1
<i>PE_DFP_VDM_Mode_Exit_ACKed</i>	Section 8.3.3.9.6.2
DFP Structured VDM Attention	
<i>PE_DFP_VDM_Attention_Request</i>	Section 8.3.3.9.7.1
Cable Plug Related	
Cable Ready	
<i>PE_CBL_Ready</i>	Section 8.3.3.10.1.1
Discover Identity	
<i>PE_CBL_Get_Identity</i>	Section 8.3.3.10.2.1

State name	Reference
<i>PE_CBL_Send_Identity</i>	Section 8.3.3.10.2.2
<i>PE_CBL_Get_Identity_NAK</i>	Section 8.3.3.10.2.3
Discover SVIDs	
<i>PE_CBL_Get_SVIDs</i>	Section 8.3.3.10.3.1
<i>PE_CBL_Send_SVIDs</i>	Section 8.3.3.10.3.2
<i>PE_CBL_Get_SVIDs_NAK</i>	Section 8.3.3.10.3.3
Discover Modes	
<i>PE_CBL_Get_Modes</i>	Section 8.3.3.10.4.1
<i>PE_CBL_Send_Modes</i>	Section 8.3.3.10.4.2
<i>PE_CBL_Get_Modes_NAK</i>	Section 8.3.3.10.4.3
Mode Entry	
<i>PE_CBL_Evaluate_Mode_Entry</i>	Section 8.3.3.10.5.1
<i>PE_CBL_Mode_Entry_ACK</i>	Section 8.3.3.10.5.2
<i>PE_CBL_Mode_Entry_NAK</i>	Section 8.3.3.10.5.3
Mode Exit	
<i>PE_CBL_Mode_Exit</i>	Section 8.3.3.10.6.1
<i>PE_CBL_Mode_Exit_ACK</i>	Section 8.3.3.10.6.2
<i>PE_CBL_Mode_Exit_NAK</i>	
Cable Soft Reset	
<i>PE_CBL_Soft_Reset</i>	Section 8.3.3.10.7.1
Cable Hard Reset	
<i>PE_CBL_Hard_Reset</i>	Section 8.3.3.10.8.1
DFP Soft Reset or Cable Reset	
<i>PE_DFP_CBL_Send_Soft_Reset</i>	Section 8.3.3.10.9.1
<i>PE_DFP_CBL_Send_Cable_Reset</i>	Section 8.3.3.10.9.2
UFP Source Soft Reset	
<i>PE_UFP_CBL_Send_Soft_Reset</i>	Section 8.3.3.10.10
Source Startup Structured VDM Discover Identity	
<i>PE_SRC_VDM_Identity_Request</i>	Section 8.3.3.10.11.1
<i>PE_SRC_VDM_Identity_ACKed</i>	Section 8.3.3.10.11.2
<i>PE_SRC_VDM_Identity_NAKed</i>	Section 8.3.3.10.11.3
BIST Receive Mode	
<i>PE_BIST_Receive_Mode</i>	Section 8.3.3.11.1.1
<i>PE_BIST_Frame_Received</i>	Section 8.3.3.11.1.2
BIST Transmit Mode	
<i>PE_BIST_Transmit_Mode</i>	Section 8.3.3.11.2.1
<i>PE_BIST_Send_Frame</i>	Section 8.3.3.11.2.2
BIST Carrier Mode and Eye Pattern	
<i>PE_BIST_Eye_Pattern_Mode</i>	Section 8.3.3.11.3.1
<i>PE_BIST_Carrier_Mode_0</i>	Section 8.3.3.11.3.2
<i>PE_BIST_Carrier_Mode_1</i>	Section 8.3.3.11.3.3

State name	Reference
<i>PE_BIST_Carrier_Mode_2</i>	Section 8.3.3.11.3.4
<i>PE_BIST_Carrier_Mode_3</i>	Section 8.3.3.11.3.5
Type-C referenced states	
<i>ErrorRecovery</i>	Section 8.3.3.12.1

9. States and Status Reporting

9.1 Overview

This chapter describes the Status reporting mechanisms for devices with data connections (e.g. D+/D- and or SSTx+/- and SSRx+/-). It also describes the corresponding USB state a device that supports USB PD **Shall** transition to as a result of changes to the USB PD state that the device is in.

This chapter does not define the System Policy or the System Policy Manager. That is defined in [\[USBCBridge 1.0\]](#). In addition the Policies themselves are not described here; these are left to the implementers of the relevant products and systems to define.

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 [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USB Type-C 1.2\]](#) or [\[USB BC 1.2\]](#) When operating in [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USB Type-C 1.2\]](#) or [\[USB BC 1.2\]](#) 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

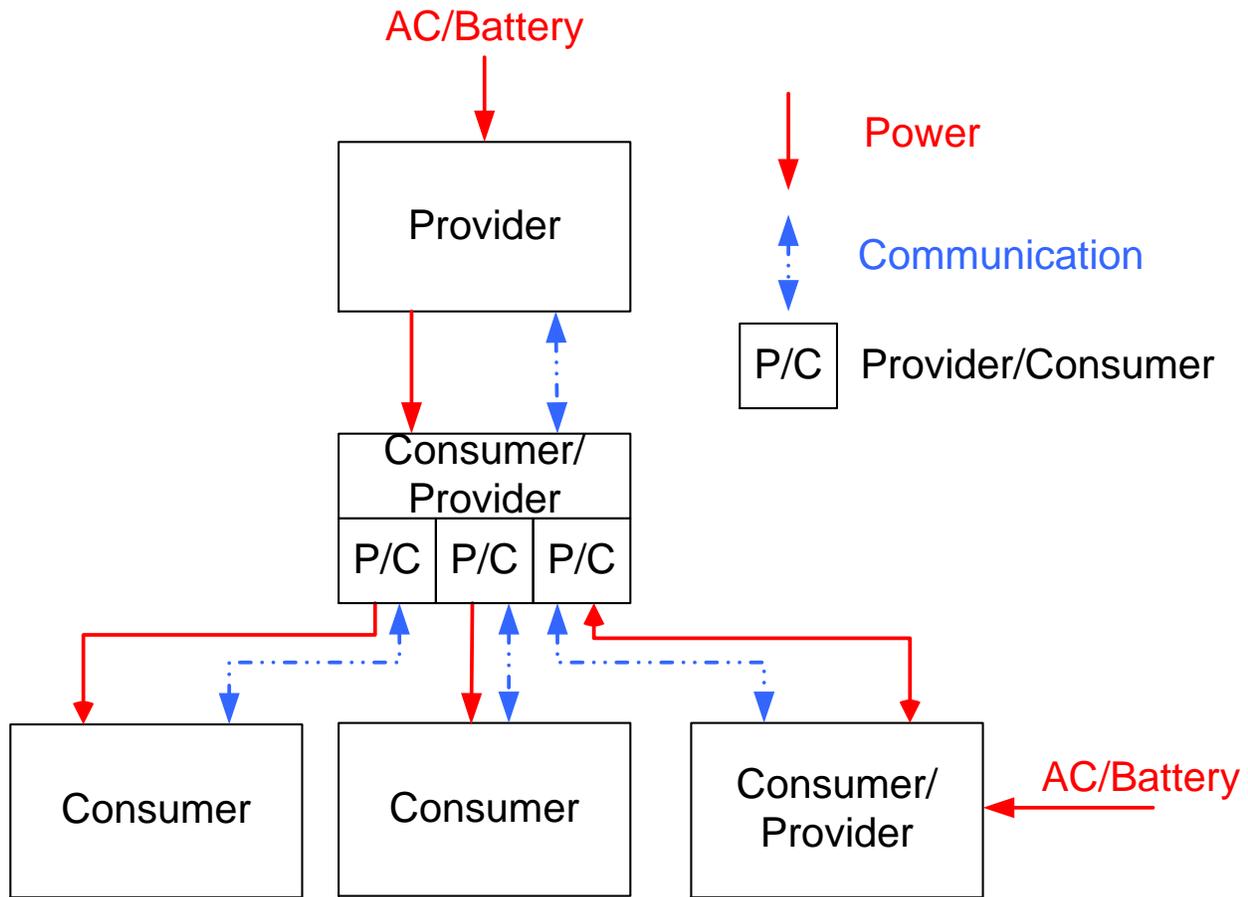
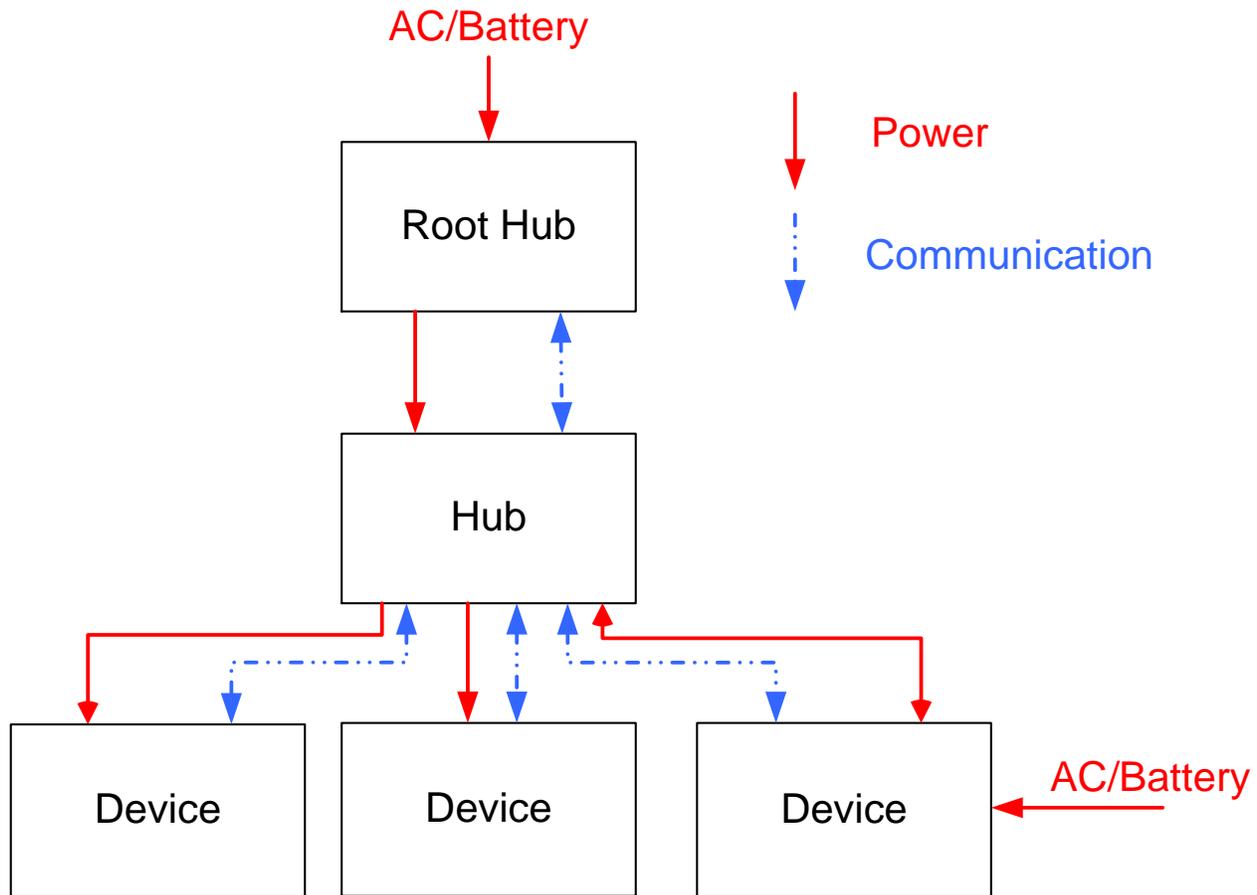


Figure 9-2 shows how this same topology can be mapped to USB. In a USB based system, policy is managed by the host and communication of system level policy information is via standard USB data line communication. This is a separate mechanism to the USB Power Delivery V_{BUS} protocol which is used to 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.

Real systems will be a mixture of devices which in terms of power management support might have implemented PD, [\[USB 2.0\]](#), [\[USB 3.1\]](#), [\[USB Type-C 1.2\]](#) or [\[USB BC 1.2\]](#) or they might 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 status mechanisms described here is to provide a mechanism whereby each connected entity in the system provides as much information as possible on the status of itself.

Figure 9-2 Mapping of PD Topology to USB



Information described in this section that is communicated to the SPM is as follows:

- Versions of USB Type-C Current, PD and BC supported
- Capabilities as a Provider/Consumer
- Current operational state of each Port e.g. Standard, USB 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 bridge as defined in [\[USBCBridge 1.0\]](#).

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. USB 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

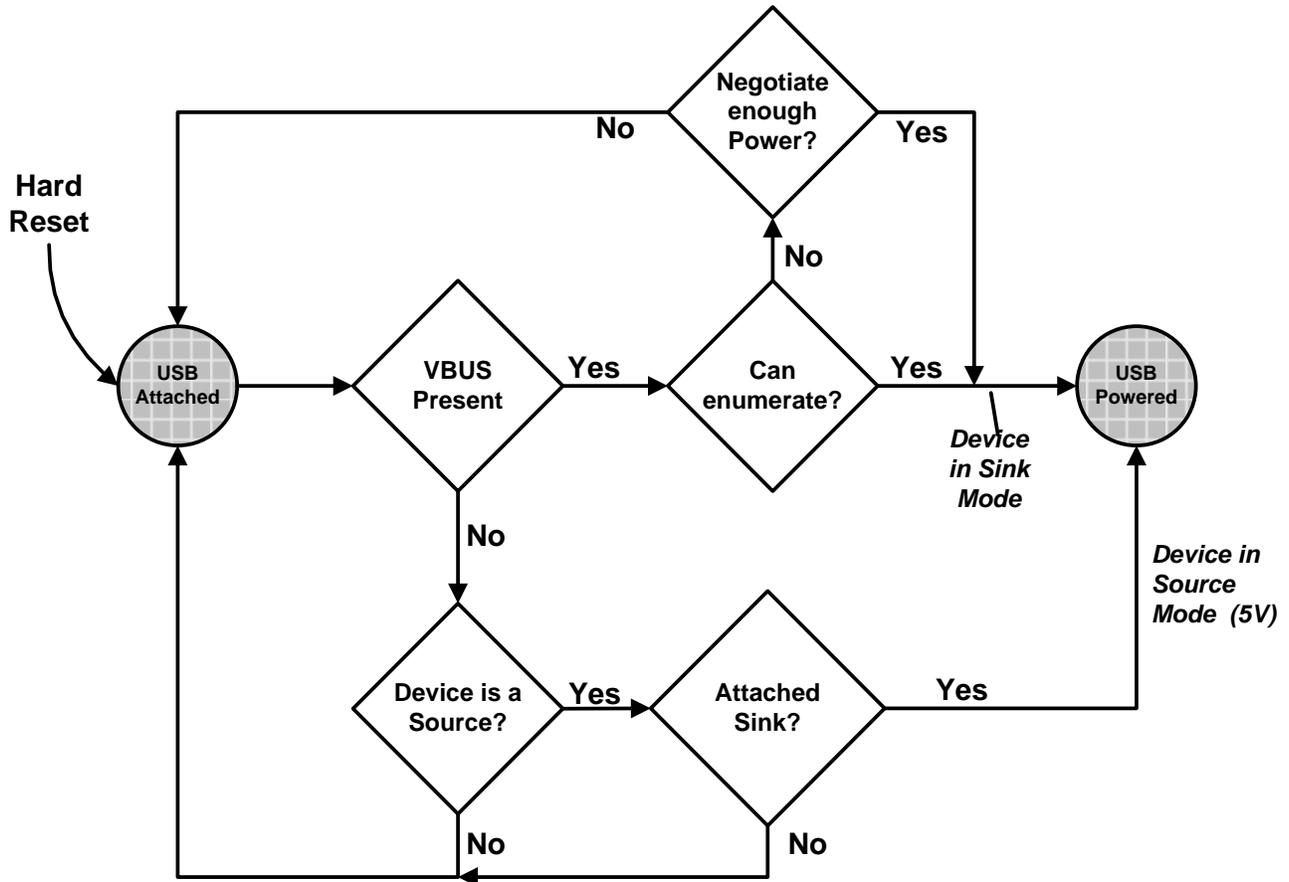


Figure 9-4 shows how a PDUSB Device determines when to transition from the USB Powered state to the USB Attached state when the device is a Consumer. A PDUSB Device determines that it is performing a Power Role Swap as described in Section 8.3.3.6.3.1 and Section 8.3.3.6.3.2. See Section 7.1.6 for additional information on device behavior during Hard Resets.

Figure 9-4 Any USB State to USB Attached State Transition (When operating as a Consumer)

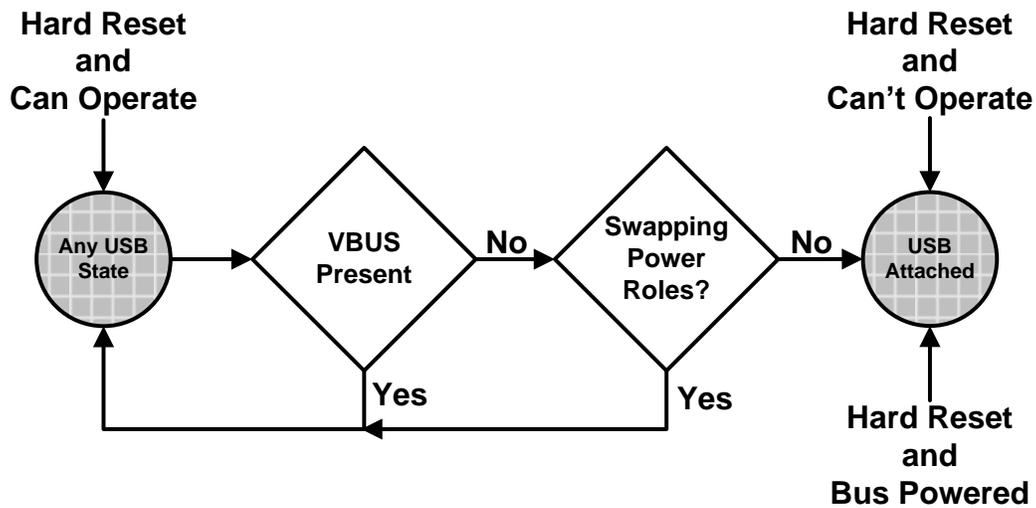


Figure 9-5 shows how a PDUSB Device determines when to transition from the USB Powered state to the USB Attached state when the device is a Provider.

Figure 9-5 Any USB State to USB Attached State Transition (When operating as a Provider)

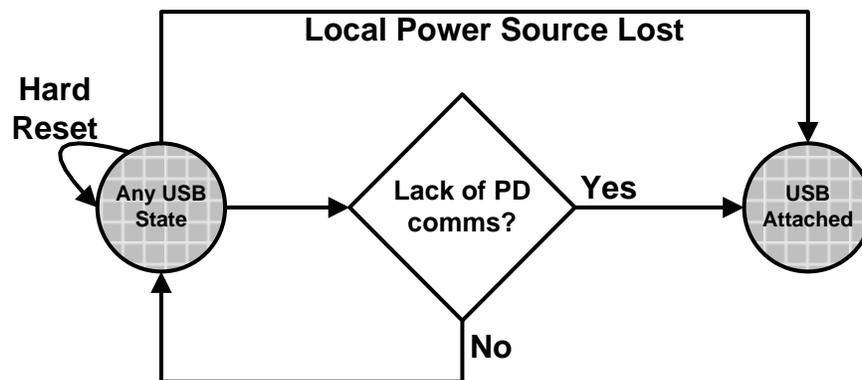
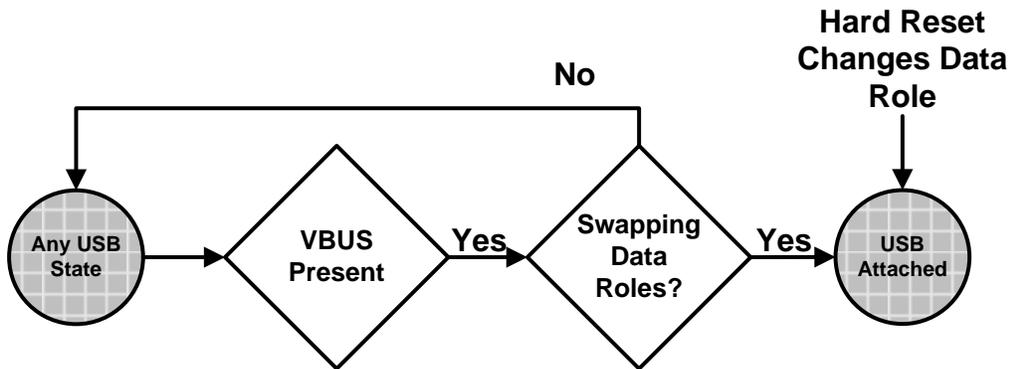


Figure 9-6 shows how a PDUSB Device using the USB Type-C connector determines when to transition from the USB Powered state to the USB Attached state after a Data Role Swap has been performed i.e. it has just changed from operation as a PDUSB Host to operation as a PDUSB Device. The Data Role Swap is described in Section 6.3.9. A Hard Reset will also return a Sink acting as a PDUSB Host to PDUSB Device operation as described in Section 6.7.2. See Section 7.1.6 for additional information on device behavior during Hard Resets.

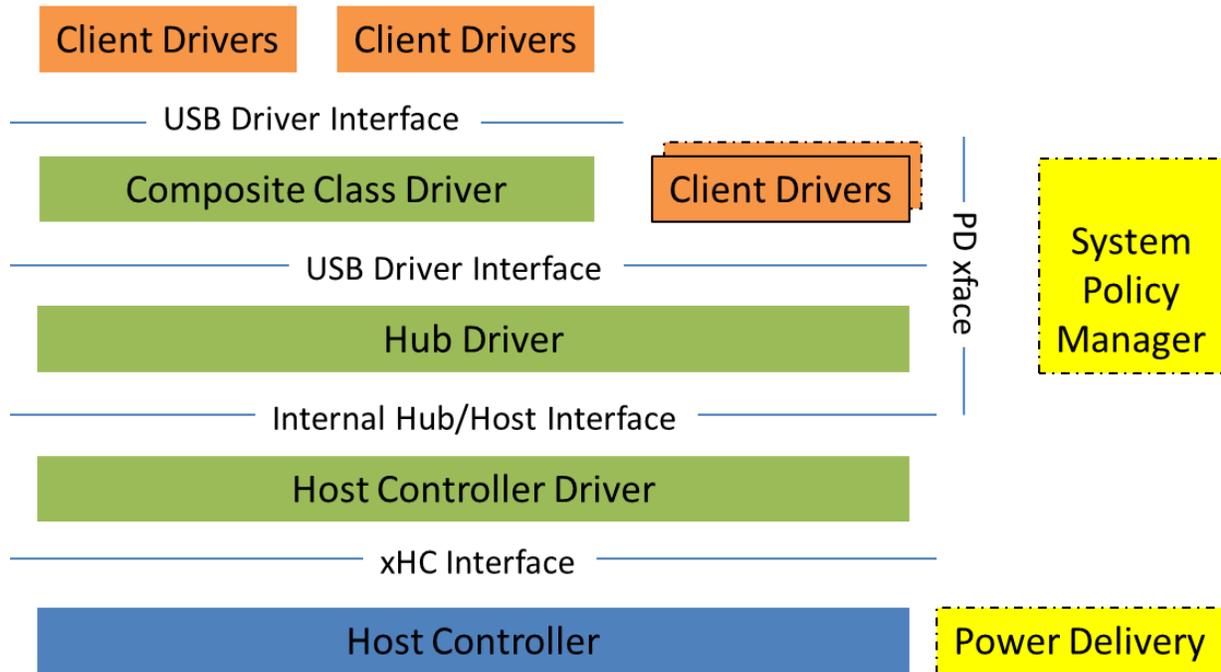
Figure 9-6 Any USB State to USB Attached State Transition (After a USB Type-C Data Role Swap)



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.

Figure 9-7 Software stack on a PD aware OS

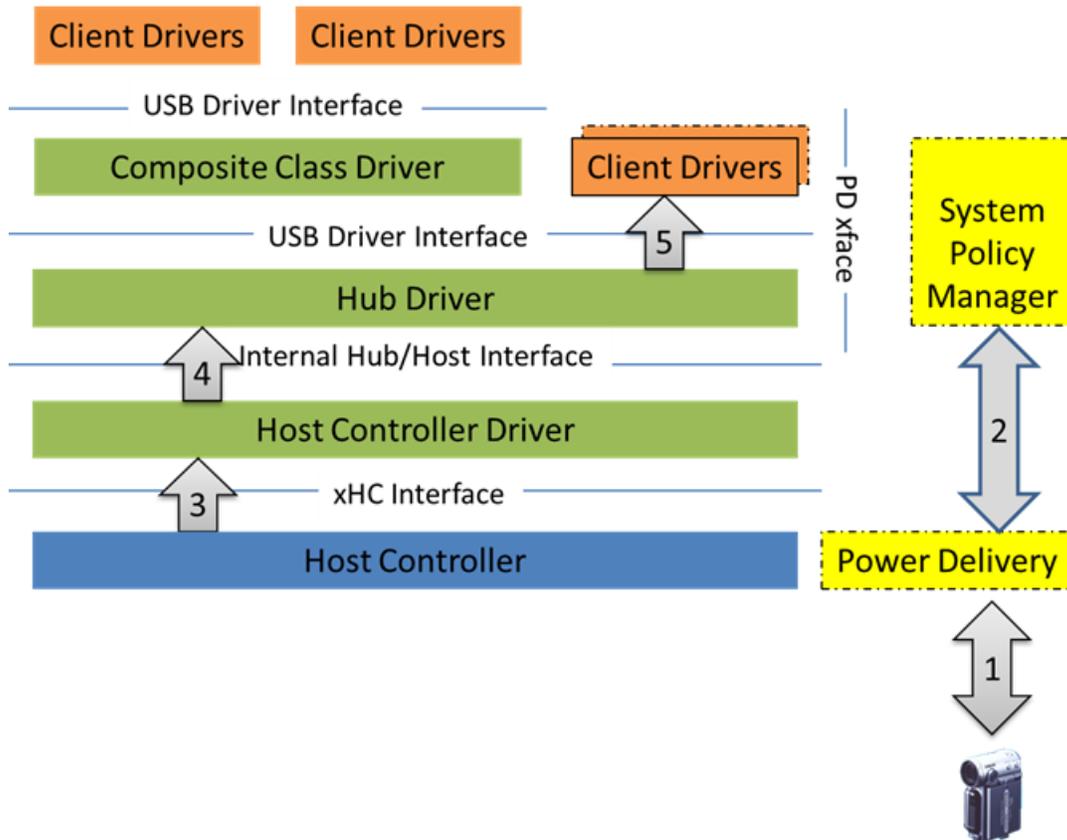


9.1.4 PDUSB Device Enumeration

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. 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.

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 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 to indicate that this device is a consumer of power. This field is only Valid if Bit 2 is set to one.</td> </tr> <tr> <td>5</td> <td>This bit Shall be set to 1 to indicate that this device supports the feature CHARGING_POLICY. Note that supporting the CHARGING_POLICY feature does not require a BC or PD mechanism to be implemented.</td> </tr> <tr> <td>6</td> <td>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</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 to indicate that this device is a consumer of power. This field is only Valid if Bit 2 is set to one.	5	This bit Shall be set to 1 to indicate that this device supports the feature CHARGING_POLICY. Note that supporting the CHARGING_POLICY feature does not require a BC or PD mechanism to be implemented.	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
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 to indicate that this device is a consumer of power. This field is only Valid if Bit 2 is set to one.																			
5	This bit Shall be set to 1 to indicate that this device supports the feature CHARGING_POLICY. Note that supporting the CHARGING_POLICY feature does not require a BC or PD mechanism to be implemented.																			
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																			

Offset	Field	Size	Value	Description														
				bcdUSBTypeCVersion field 7 Reserved . Shall be set to zero. 15:8 bmPowerSource. At least one of the following bits 8, 9 and 14 Shall be set to indicate which power sources are supported. <table border="1" style="margin-left: 20px;"> <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_{BUS}</td> </tr> <tr> <td>15</td> <td>Reserved and Shall be set to zero.</td> </tr> </tbody> </table> 31:16 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 _{BUS}	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 _{BUS}																	
15	Reserved and Shall be set to zero.																	
8	<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.														
10	<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.														
12	<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: BATTERY_INFO_CAPABILITY
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."

Offset	Field	Size	Value	Description
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).
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 is a 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: PD_CONSUMER_PORT_CAPABILITY										
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="792 890 1248 1094"> <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 is a Provider.

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: PD_PROVIDER_PORT_CAPABILITY

Offset	Field	Size	Value	Description										
3	<i>bReserved</i>	1	Number	Reserved and Shall be set to zero.										
4	<i>bmCapabilities</i>	2	Bitmap	This field Shall indicate the specification the Provider Port will operation under. <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 nd 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 Specific Requests and Events

A PDUSB Device that is compliant to this specification **Shall** support the Battery related requests if it has a battery.

A PDUSB Hub that is compliant to this specification **Shall** support a USB PD Bridge as described in [\[USBCBridge 1.0\]](#) irrespective of whether the PDUSB Hub is a Provider, a Consumer, or both.

9.3.1 PD 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 Requests

Request	bmRequestType	bRequest	wValue	wIndex	wLength	Data
GetBatteryStatus	10000000B	<i>GET_BATTERY_STATUS</i>	Zero	Battery ID	Eight	Battery Status
SetPDFeature	00000000B	SET_FEATURE	Feature Selector	Feature Specific	Zero	None

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 Request Codes

bRequest	Value
<i>GET_BATTERY_STATUS</i>	21

Table 9-8 gives the **Valid** feature selectors for the PD class. Refer to Section 9.4.2.1, and Section 9.4.2.2 for a description of the features.

Table 9-8 PD Feature Selectors

Feature Selector	Recipient	Value
<i>BATTERY_WAKE_MASK</i>	Device	40
<i>CHARGING_POLICY</i>	Device	54

9.4 PDUSB Hub and PDUSB Peripheral Device Requests

9.4.1 GetBatteryStatus

This request returns the current status of the Battery in a PDUSB Hub/Peripheral.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
1000000B	<i>GET_BATTERY_STATUS</i>	Zero	Battery ID	Eight	Battery Status

The PDUSB Hub/Peripheral **Shall** return the Battery Status of the Battery identified by the value of *wIndex* field.

Every PDUSB 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 9-9 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	<p>Shall indicate the Battery State of Charge given as percentage value from Battery Remaining Capacity.</p>																				
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																							
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 9.4.2 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																							

Offset	Field	Size	Value	Description
4	<i>wRemainingOperatingTime</i>	2	Number	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. A Battery that is not capable of returning this information Shall return a value of 0xFFFF.
6	<i>wRemainingChargeTime</i>	2	Number	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". A Battery that is not capable of returning this information Shall return a value of 0xFFFF.

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.

If the PDUSB Hub is not configured, the PDUSB Hub's response to this request is undefined.

9.4.2 SetPDFeature

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**
- **CHARGING_POLICY**

9.4.2.1 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 9-10 Battery Wake Mask

Bit	Description
0	Battery Present: When this bit is set then the PDUSB Device Shall generate a wake event if it detects that a Battery has been Attached.
1	Charging Flow: 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	Battery Error: 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.

9.4.2.2 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 9-11. 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 [USB Type-C 1.2] or USB PD specifications **Shall** apply.

Table 9-11 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.
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: ICCHPF as defined in the USB 2.0 or USB 3.1 specification, regardless of its USB state. Current limit as negotiated through other USB mechanisms such as BC.
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: ICCLPF as defined in the USB 2.0 or USB 3.1 specification, regardless of its USB state. Current limit as negotiated through other USB mechanisms such as BC.
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 5 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.

10. Power Rules

10.1 Introduction

The flexibility of power provision on USB Type-C is expected to lead to adapter re-use and the increasingly widespread provision of USB power outlets in domestic and public places and in transport of all kinds. Environmental considerations could result in unbundled adapters. Rules are needed to avoid incompatibility between the Sources and the Sinks they are used to power, in order to avoid user confusion and to meet user expectations. This section specifies a set of rules that Sources and Sinks **Shall** follow. These rules provide a simple and consistent user experience.

10.2 Source Power Rules

10.2.1 Source Power Rule Considerations

The Source power rules are designed to:

- Ensure the PD Power (PDP) of an adapter specified in watts explicitly defines the voltages and currents at each voltage the adapter supports
- Ensure that adapters with a large PDP are always capable of providing the power to devices designed for use with adapters with a smaller PDP
- Enable an ecosystem of adapters that are interoperable with the devices in the ecosystem.

The considerations that lead to the Source power rules are based are summarized in Table 10-1.

Table 10-1 Considerations for Sources

Considerations	Rationale	Consequence
Simple to identify capability	A user going into an electronics retailer knows what they need	Cannot have a complex identification scheme
Higher power Sources are a superset of smaller ones	Bigger is always better in user's eyes – don't want a degradation in performance	Higher power Sources do everything smaller ones do
Unambiguous Source definitions	Sources with the same power rating but different VI combinations might not interoperate	To avoid user confusion, any given power rating has a single definition
A range of power ratings	Users and companies will want freedom to pick appropriate Source ratings	Fixed profiles at specific power levels don't provide adequate flexibility, e.g. profiles as defined in previous versions of PD.
5V@3A USB Type-C Source is defined by [USB Type-C 1.2]	5V@3A USB Type-C Source is considered	All > 15W adapters must support 5V@3A or superset consideration is violated
Maximize 3A cable utilization	3A cables will be ubiquitous	Increase to maximum voltage (20V) before increasing current beyond 3A
Optimize voltage rail count	More rails are a higher burden for Sources, particularly in terms of testing	5V is a basic USB requirement. 20V provides the maximum capability.
Some Sources are not able to provide significant power	Some small Battery operated Sources e.g. mobile devices, are able to provide more power directly from their Battery than from a regulated 5V supply	In addition to the minimal 5V advertisement are able to advertise more power from their Battery
Some Sources share power between multiple Ports (Hubs)	Hubs have to be supported	See Section 10.2.4

10.2.2 Normative Voltages and Currents

The voltages and currents a Source with a PDP of x Watts **Shall** support are as defined in Table 10-2.

Table 10-2 Normative Voltages and Currents

PDP (W)	Current at 5V (A)	Current at 9V (A)	Current at 15V (A)	Current at 20V (A)
$0.5 \leq x \leq 15$	$x \div 5$			
$15 < x \leq 27$	3	$x \div 9$		
$27 < x \leq 45$	3	3	$x \div 15$	
$45 < x \leq 60$	3	3	3	$x \div 20$
$60 < x \leq 100$	3	3	3	$x \div 20^1$

¹ Requires a 5A cable.

Figure 10-1 illustrates the maximum current and power rails that a Source **shall** support at each voltage for a given PDP.

Figure 10-1 Source Power Rule Illustration

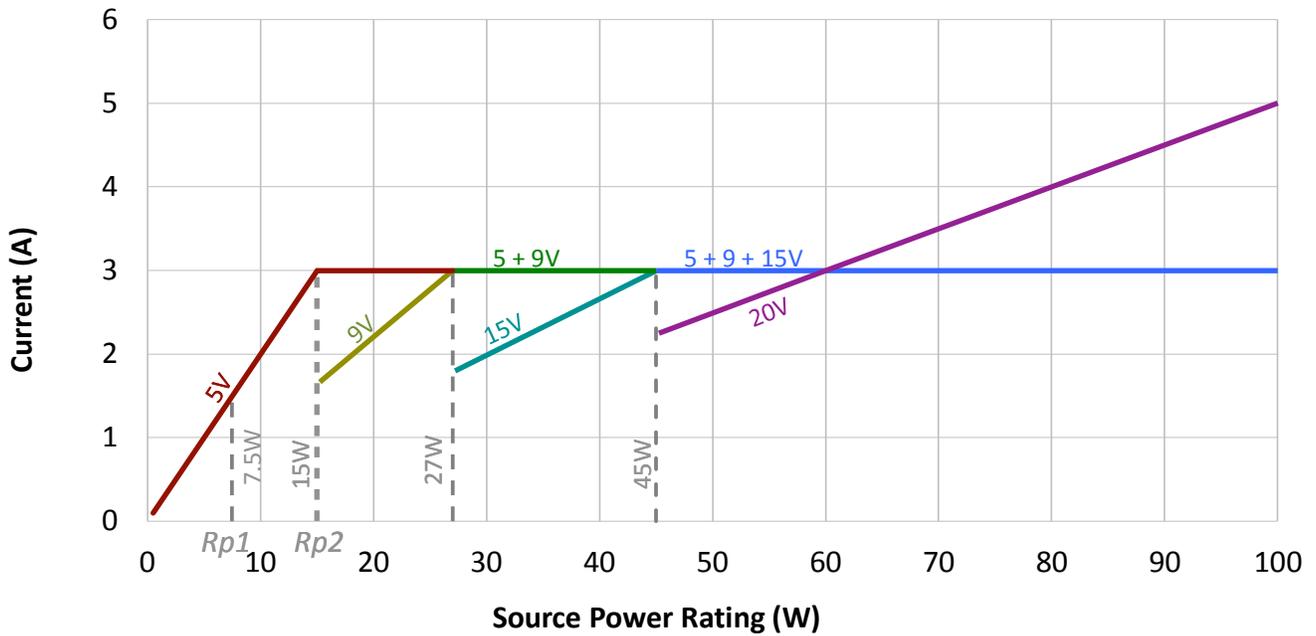


Figure 10-2 shows an example of an adapter with a rating at 50W. The adapter is required to support 20V at 2.5A, 15V at 3A, 9V at 3A and 5V at 3A.

Figure 10-2 Source Power Rule Example

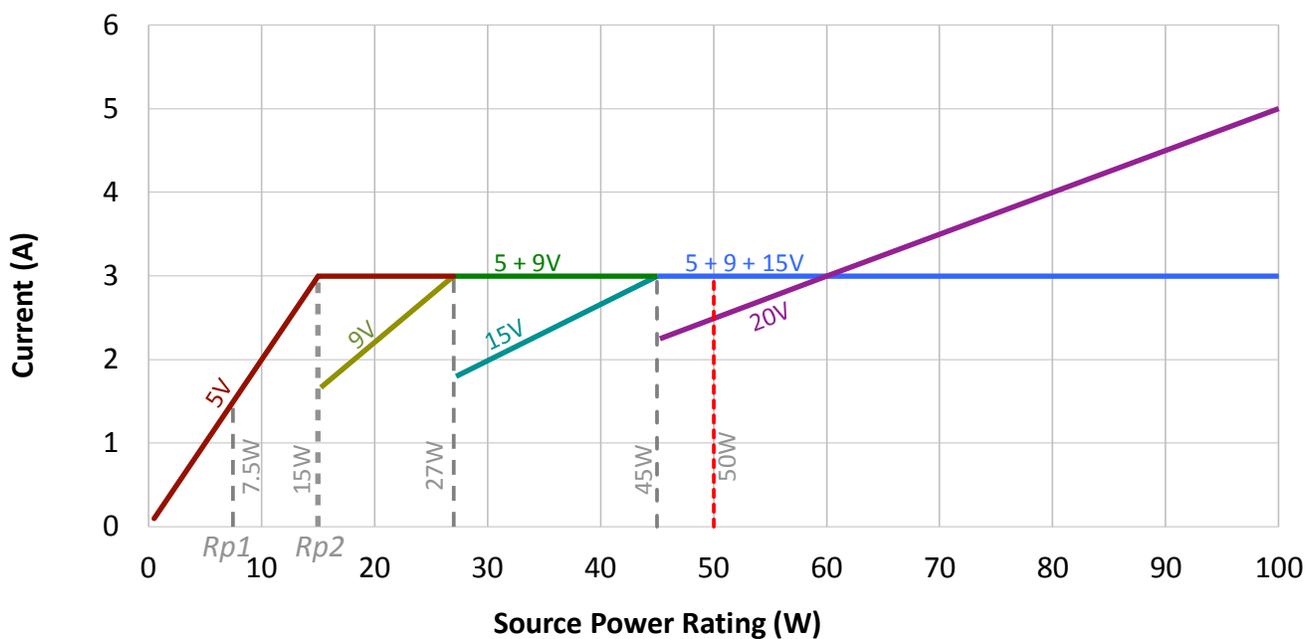


Table 10-3, Table 10-4, Table 10-5 and Table 10-6 show the Fixed Supply PDOs that **Shall** be supported for each of the **Normative** voltages defined in Table 10-2.

Table 10-3 Fixed Supply PDO – Source 5V

Bit(s)	Description						
B31...30	Fixed supply						
B29	Dual-Role Power						
B28	USB Suspend Supported						
B27	Unconstrained Power						
B26	USB Communications Capable						
B25	Dual-Role Data						
B24...22	Reserved – Shall be set to zero.						
B21...20	Peak Current						
B19...10	5V						
B9...0	<table border="1"> <thead> <tr> <th>PDP (x)</th> <th>Current (A)</th> </tr> </thead> <tbody> <tr> <td>$0.5 \leq x \leq 15$</td> <td>$x \div 5$</td> </tr> <tr> <td>$15 < x \leq 100$</td> <td>3</td> </tr> </tbody> </table>	PDP (x)	Current (A)	$0.5 \leq x \leq 15$	$x \div 5$	$15 < x \leq 100$	3
PDP (x)	Current (A)						
$0.5 \leq x \leq 15$	$x \div 5$						
$15 < x \leq 100$	3						

Table 10-4 Fixed Supply PDO – Source 9V

Bit(s)	Description								
B31...30	Fixed Supply								
B29...22	Reserved – Shall be set to zero.								
B21...20	Peak Current								
B19...10	9V								
B9...0	<table border="1"> <thead> <tr> <th>PDP (x)</th> <th>Current (A)</th> </tr> </thead> <tbody> <tr> <td>$0.5 \leq x \leq 15$</td> <td>PDO not required</td> </tr> <tr> <td>$15 < x \leq 27$</td> <td>$x \div 9$</td> </tr> <tr> <td>$27 < x \leq 100$</td> <td>3</td> </tr> </tbody> </table>	PDP (x)	Current (A)	$0.5 \leq x \leq 15$	PDO not required	$15 < x \leq 27$	$x \div 9$	$27 < x \leq 100$	3
PDP (x)	Current (A)								
$0.5 \leq x \leq 15$	PDO not required								
$15 < x \leq 27$	$x \div 9$								
$27 < x \leq 100$	3								

Table 10-5 Fixed Supply PDO – Source 15V

Bit(s)	Description								
B31...30	Fixed Supply								
B29...22	Reserved – Shall be set to zero.								
B21...20	Peak Current								
B19...10	15V								
B9...0	<table border="1"> <thead> <tr> <th>PDP (x)</th> <th>Current (A)</th> </tr> </thead> <tbody> <tr> <td>$0.5 \leq x \leq 27$</td> <td>PDO not required</td> </tr> <tr> <td>$27 < x \leq 45$</td> <td>$x \div 15$</td> </tr> <tr> <td>$45 < x \leq 100$</td> <td>3</td> </tr> </tbody> </table>	PDP (x)	Current (A)	$0.5 \leq x \leq 27$	PDO not required	$27 < x \leq 45$	$x \div 15$	$45 < x \leq 100$	3
PDP (x)	Current (A)								
$0.5 \leq x \leq 27$	PDO not required								
$27 < x \leq 45$	$x \div 15$								
$45 < x \leq 100$	3								

Table 10-6 Fixed Supply PDO – Source 20V

Bit(s)	Description						
B31...30	Fixed Supply						
B29...22	Reserved – Shall be set to zero.						
B21...20	Peak Current						
B19...10	20V						
B9...0	<table border="1"> <thead> <tr> <th>PDP (x)</th> <th>Current (A)</th> </tr> </thead> <tbody> <tr> <td>$0.5 \leq x \leq 45$</td> <td>PDO not required</td> </tr> <tr> <td>$45 < x \leq 100$</td> <td>$x \div 20$</td> </tr> </tbody> </table>	PDP (x)	Current (A)	$0.5 \leq x \leq 45$	PDO not required	$45 < x \leq 100$	$x \div 20$
PDP (x)	Current (A)						
$0.5 \leq x \leq 45$	PDO not required						
$45 < x \leq 100$	$x \div 20$						

More current **May** be offered in the PDOs when **Optional** voltages/currents are supported and a 5A cable is being used (see Section 10.2.3).

10.2.3 Optional Voltages/Currents

In addition to the voltages and currents specified in Section 10.2.2 a Source that is optimized for use with a specific Sink or a specific class of Sinks **May Optionally** supply additional voltages and increased currents. When **Optional** voltages and increased currents are provided, the following requirements **Shall** apply:

- The Source **Shall** be able to meet its PDP at the **Normative** voltages and currents as specified in Section 10.2.2.
- A Source **Shall Not** advertise Fixed Supply PDO **Optional** voltages and currents that exceed the PDP.
- A Source **Shall Not** advertise Variable Supply PDO **Optional** maximum voltages and currents that exceed the PDP.
- A Source **Shall Not** advertise a Battery Supply PDO **Optional** maximum allowable power that exceeds the PDP.

10.2.4 Power sharing between ports

The Source power rules defined in Section 10.2.2 and Section 10.2.3 **Shall** apply independently to each port on a system with multiple ports.

10.3 Sink Power Rules

10.3.1 Sink Power Rule Considerations

The Sink power rules are designed to ensure the best possible user experience when a given Sink used with a compliant Source of arbitrary Output Power Rating that only supplies the **Normative** voltages and currents.

The Sink Power Rules are based on the following considerations:

- Low power Sources (e.g., 5V) are expected to be very common and will be used with Sinks designed for a higher PDP.
- Optimizing the user experience when Sources with a high PDP are used with low power Sinks.
- Preventing Sinks that only function well (or at all) when using **Optional** voltages and currents.

10.3.2 Normative Sink Rules

Sinks designed to use for Sources with a PDP of x W **Shall**:

- either operate or charge from Sources that have a PDP $\geq x$ W
- either operate, charge or indicate a capability mismatch (see Section 6.4.2.3) from Sources that have a PDP $< x$ W and ≥ 0.5 W

A Sink optimized for a Source with **Optional** voltages and currents or power as described in Section 10.2.3 with a PDP of x W **Shall** provide a similar user experience when powered from a Source with a PDP of $\geq x$ W that supplies only the **Normative** voltages and currents as specified in Section 10.2.2.

The following requirements **Shall** apply to the advertised Sink Capabilities:

- A Sink **Shall Not** advertise Fixed Supply PDO maximum voltages and currents that exceed the PDP they were designed to use.
- A Sink **Shall Not** advertise Variable Supply PDO maximum voltages and currents that exceed the PDP they were designed to use.
- A Sink **Shall Not** advertise a Battery Supply PDO maximum allowable power that exceeds the PDP they were designed to use.

A. Power Profiles – This section is Deprecated

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,rl_crc;

    for(int i=0;i<len;i++) {

        newbit = ((crc>>31) ^ ((x>>i)&1)) & 1;
        if(newbit) newword=poly; else newword=0;
        rl_crc = (crc<<1) | newbit;
        crc = rl_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");

}

```

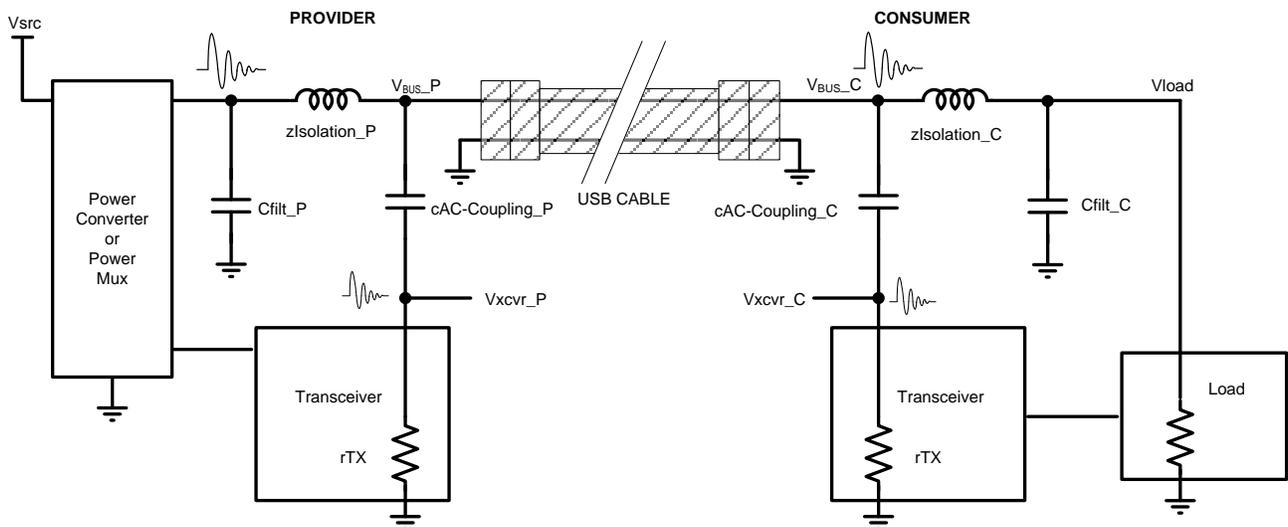

C. Power Implementation Considerations

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 might 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 might arise due to this impedance. Measurement techniques to validate conducted noise will be discussed.

Figure C-1 Typical System Electrical Model



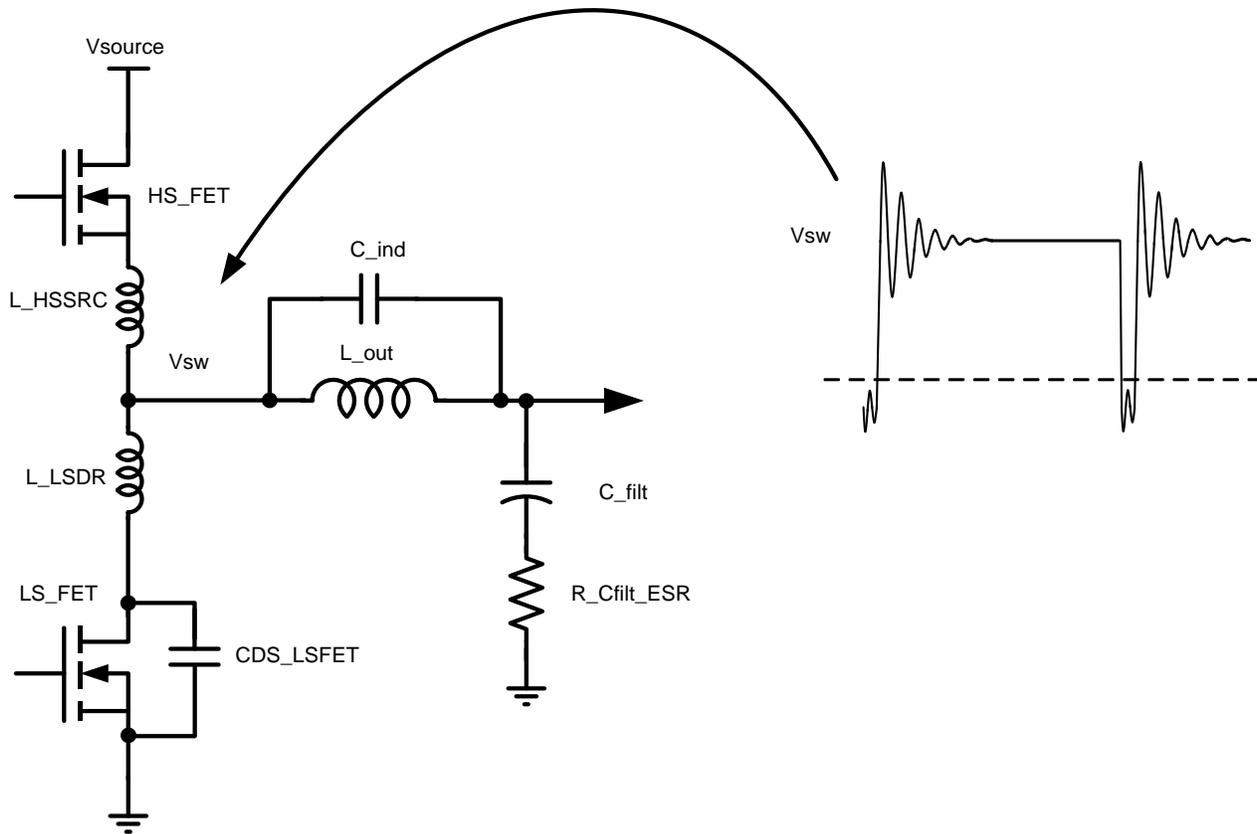
C.1.1 In-band $f_{Carrier}$ Spurious Noise

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 might 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 might be in contention with $f_{Carrier}$. L_{HSSRC} , L_{LSDRV} , and CDS_{LSFET} are typically responsible for creating resonant noise on the V_{SW} node. Other topologies might have different parasitic elements to consider. This waveform can couple to the output through C_{ind} 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



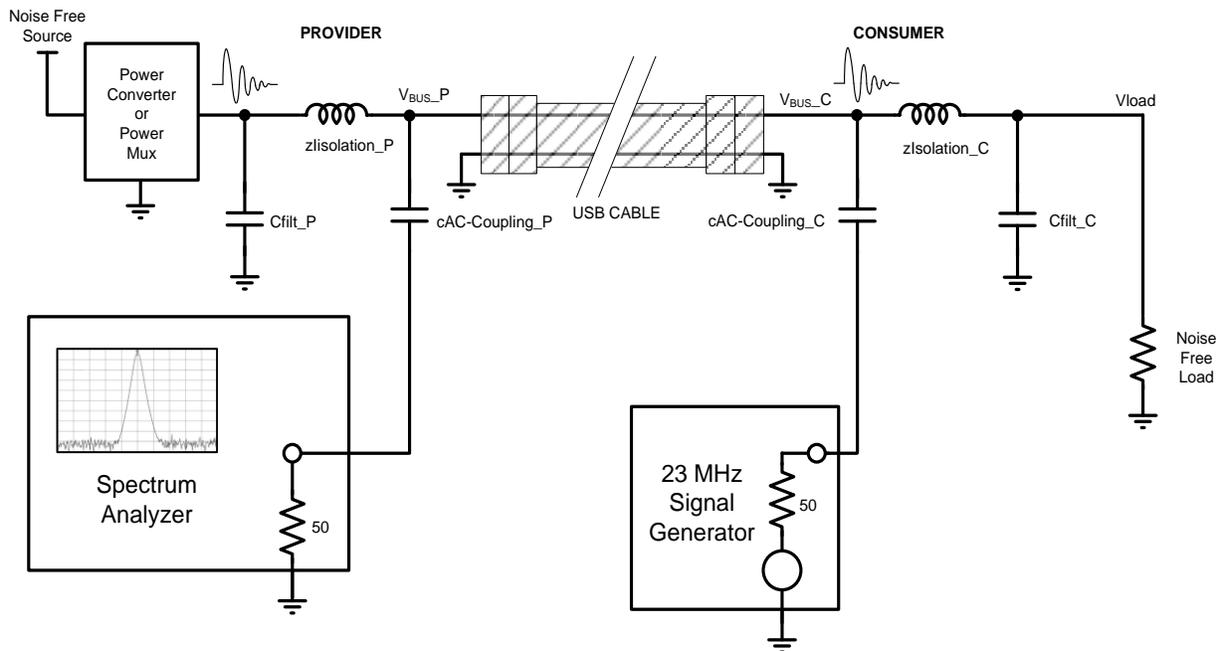
Measurement of in-band spurious noise from the power converter involves setting up measurement equipment with the correct scales and impedances. The process involves calibrating a known carrier signal in a test setup containing the correct source and load impedances with the power converter connected, 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 might be necessary to test in an environment with EMI shielding from radio noise sources.
- 6) Turn on the power converter and turn off the $f_{Carrier}$ nominal signal from the signal generator.
- 7) Confirm that the noise floor has not increased beyond $(\text{dBm}_{Baseline} - snrSrc)$. Reference Figure 7-7 for in-band noise. The $(\text{dBm}_{Baseline} - snrSrc)$ represents the 0dB level in Figure 7-7. Confirm that all measured noise falls below the curve.

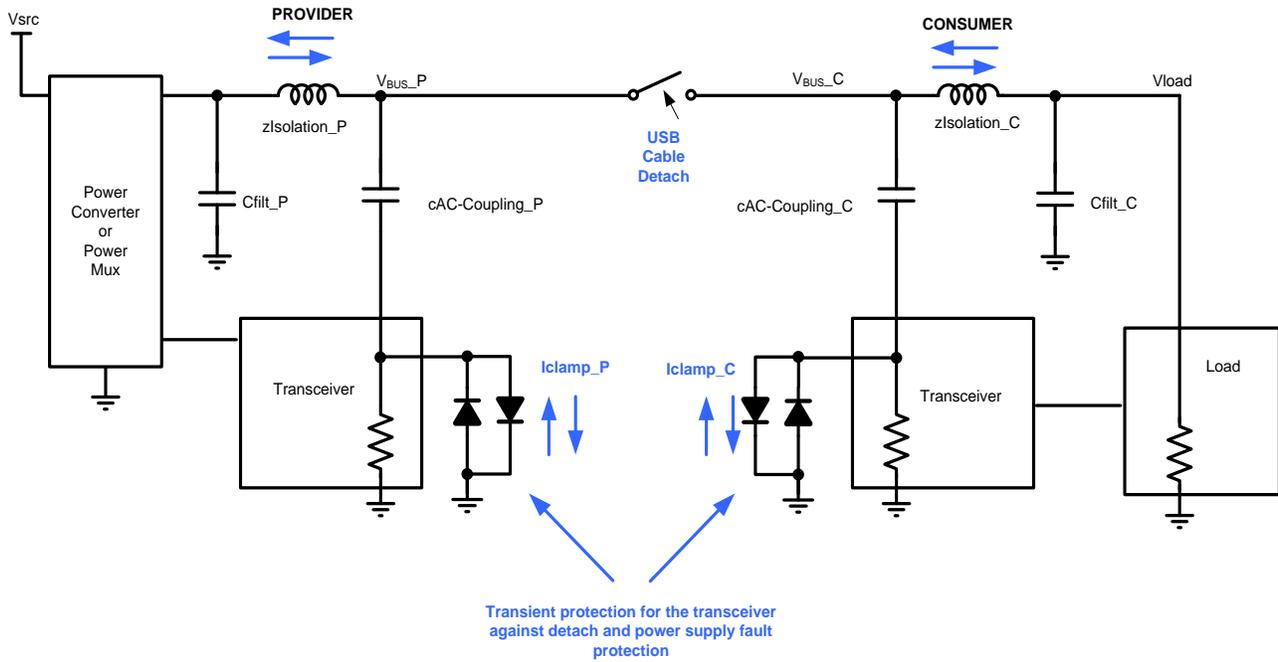
The same measurement can be made from the perspective of the Sink using a quiet source. In this case the frequency generator is placed on the Provider side and the signal generator is placed on the Consumer side. The levels would be compared to Figure 7-10 in this case.

* 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 *zIsolation_P* and *zIsolation_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 (*zIsolation_P* and *zIsolation_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

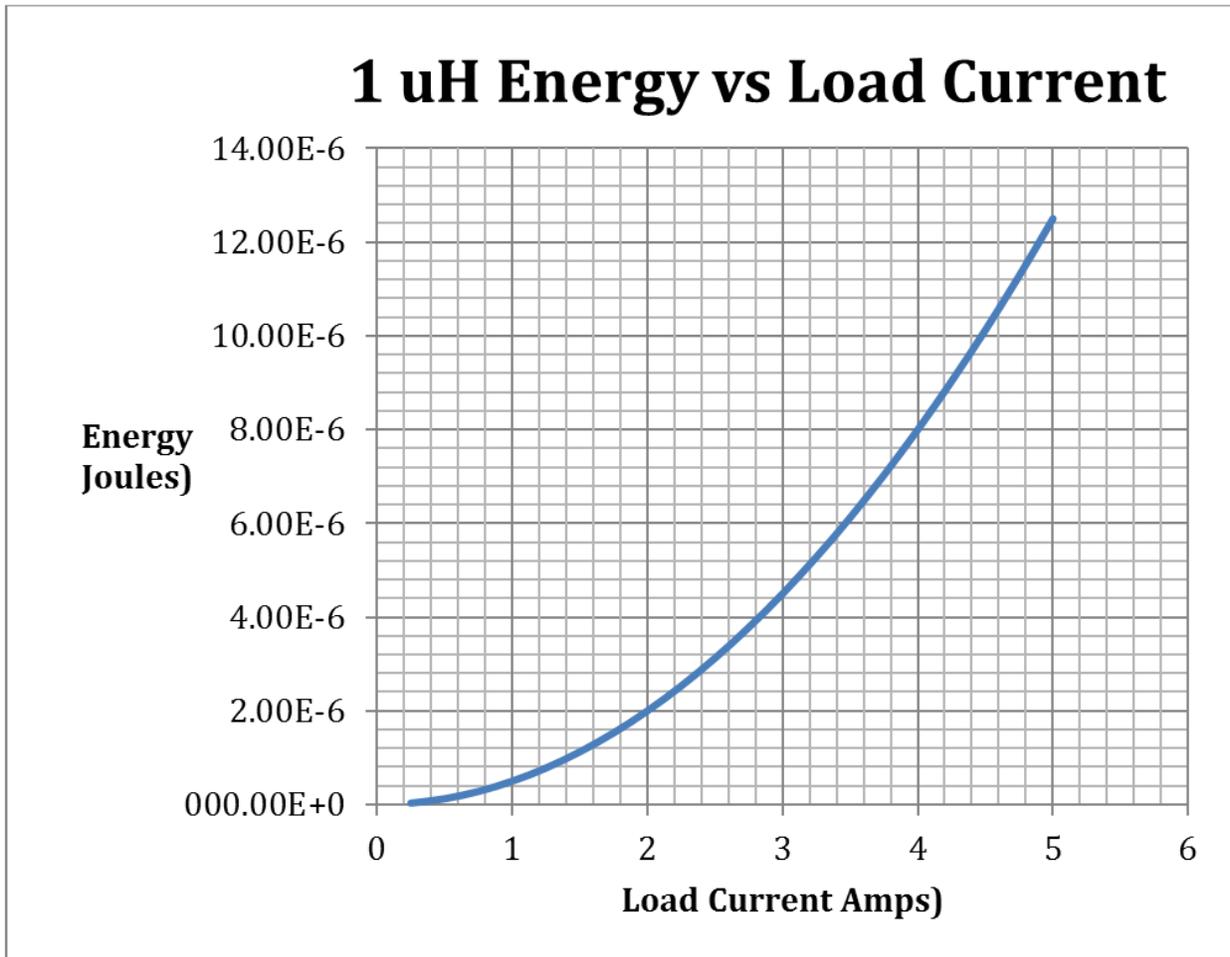


Figure C-5 shows the total energy that could be delivered during a Detach event with the example 1μH inductor where:

$$Energy = \frac{LI^2}{2}$$

An external or integrated clamp **Should** be implemented to absorb this energy and limit the applied voltage at the transceiver input.

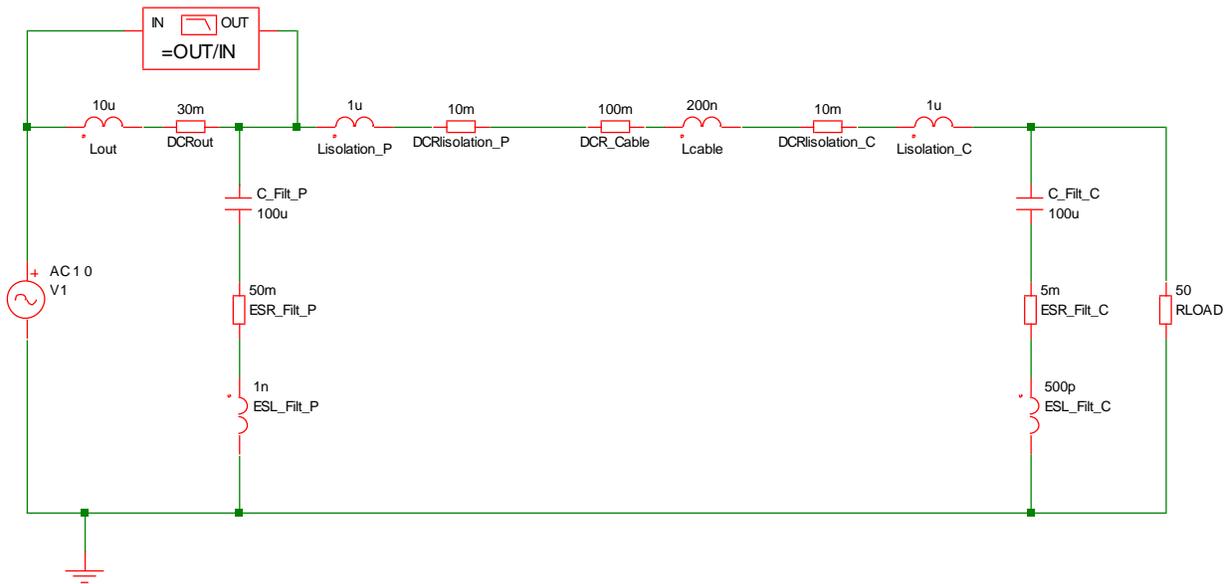
C.3 Closed Loop Stability Effects

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.

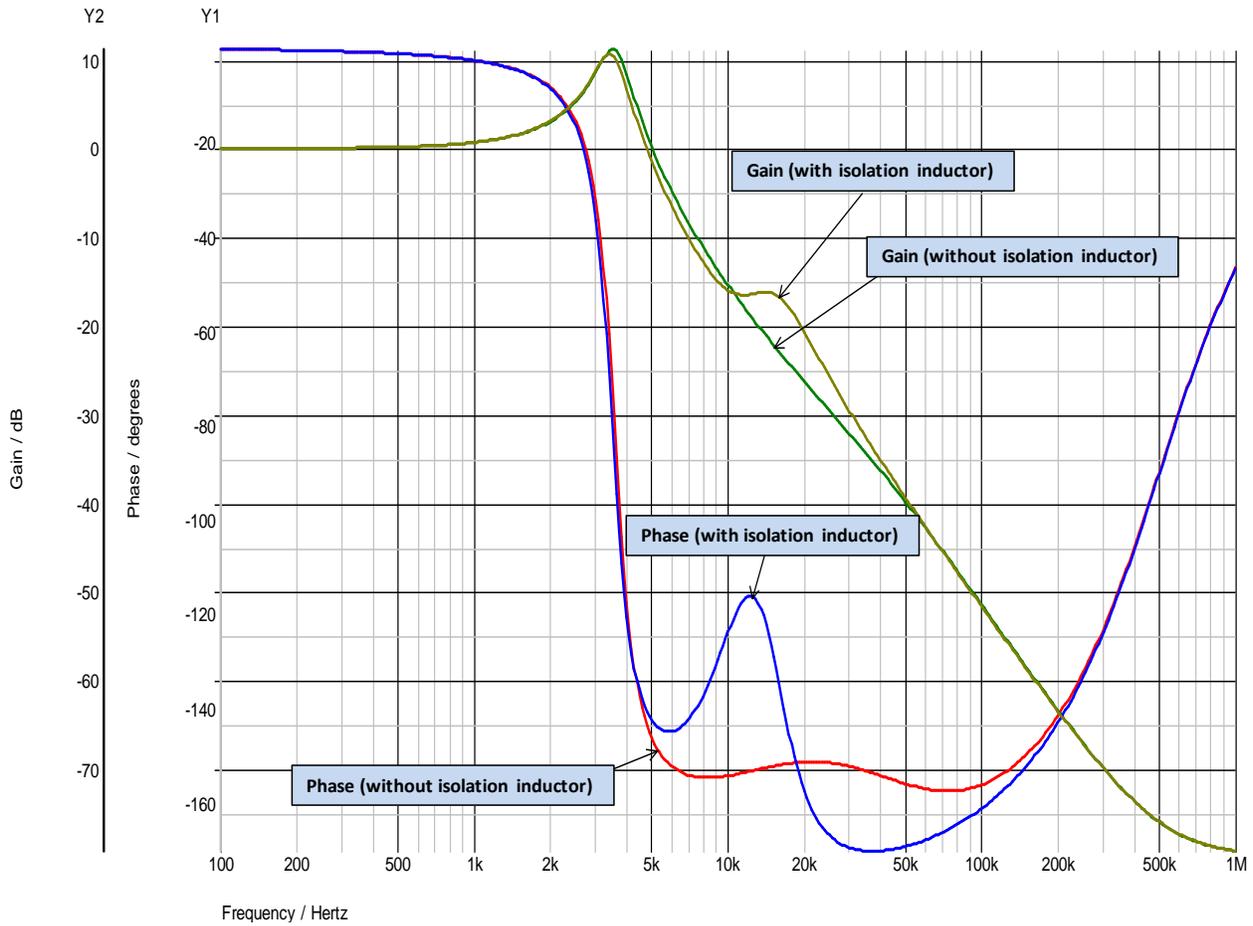
Figure C-6 Simplified Small Signal AC Model



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 $(z_{isolation_P} + L_{cable} + z_{isolation_C})$. 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 $(z_{isolation_P} + L_{cable} + z_{isolation_C})$ 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 de-tune the dominant pole will help reduce the negative phase effects.

C.3.2 Feedback Past Isolation Inductor

It could 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)

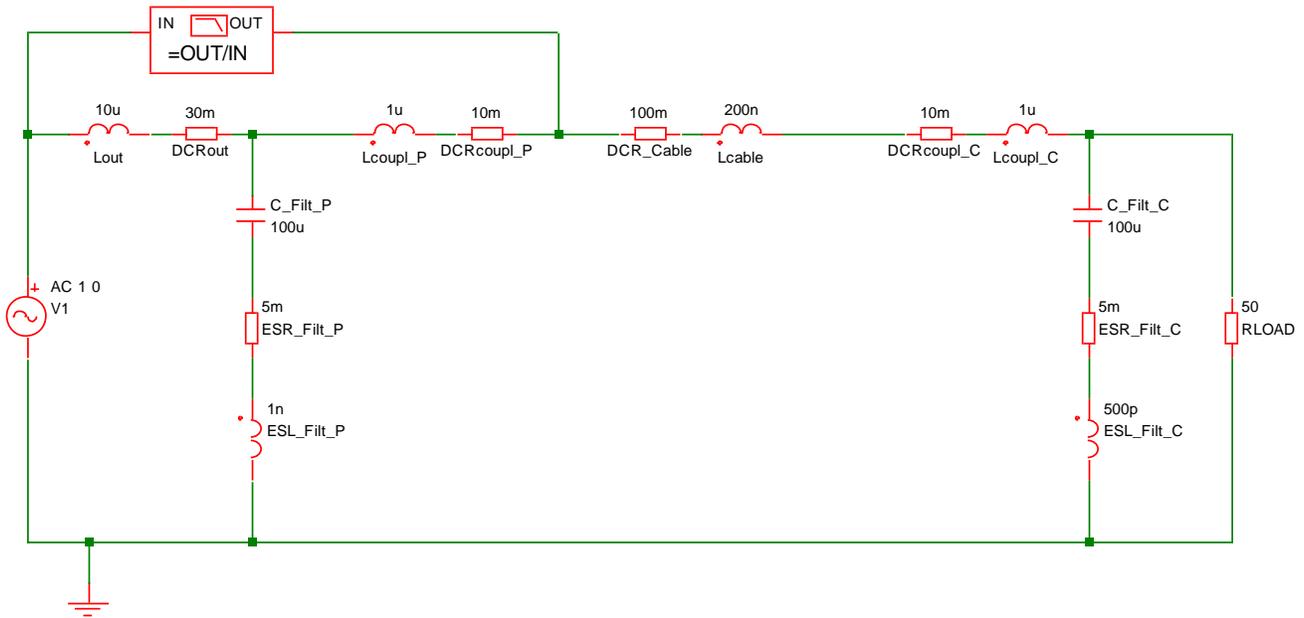
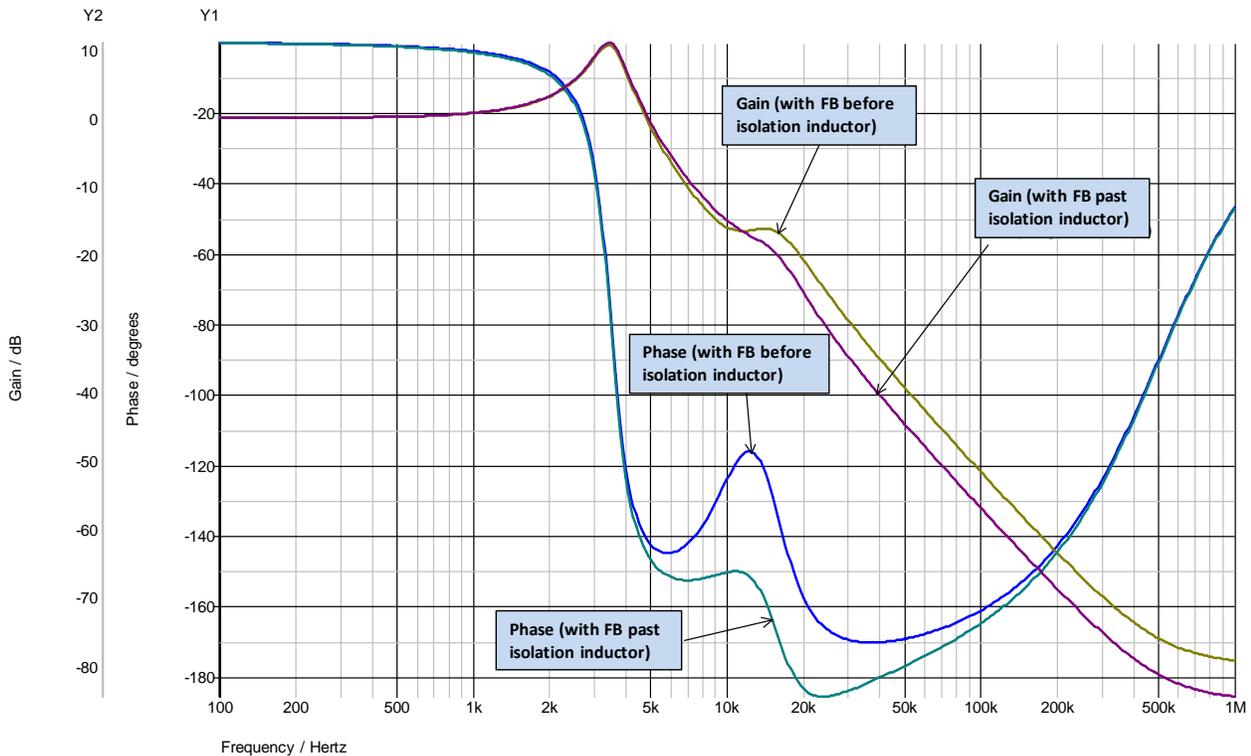


Figure C-8 compares power stage response with compensation taken before and after the *zIsolation_P* Provider isolation inductor. As can be seen in Figure C-9, the phase and gain impacts are larger. A pole is created by the *zIsolation_P* inductor that forces a phase reduction beyond 180 degrees which will have to be accounted for in the compensation network.

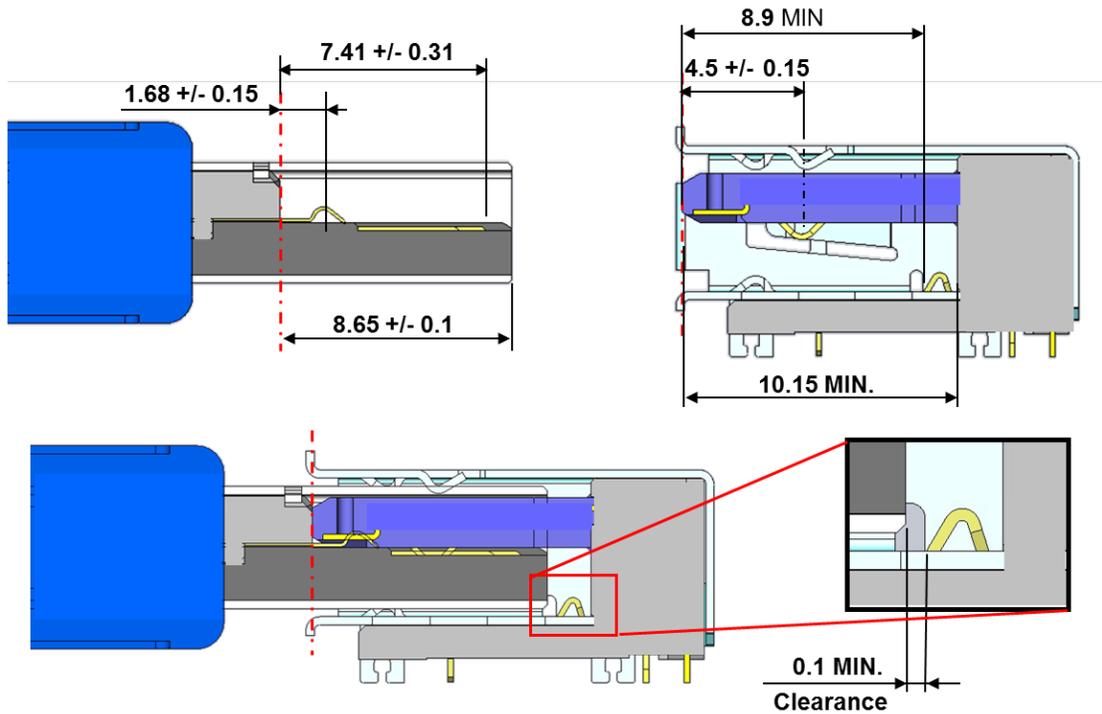
Figure C-9 Simplified Small Signal AC Model (Feedback before and after Inductor zIsolation_P)



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.

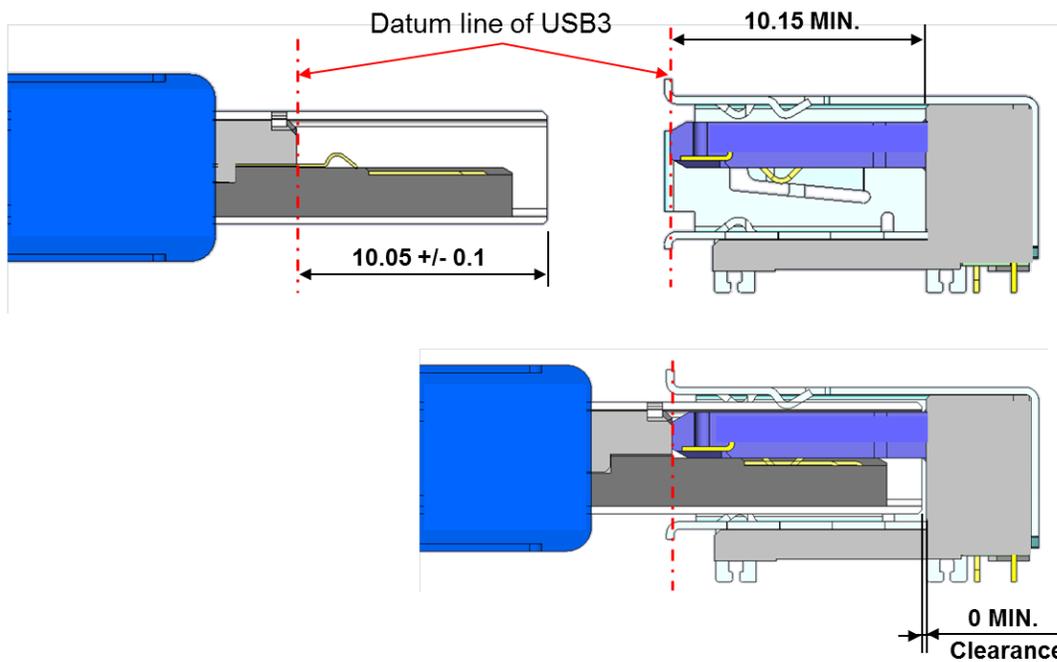
Figure D-1 USB 3.1 Standard-A Plug with USB 2.0 PD or 3.1 PD Standard-A Receptacle



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.

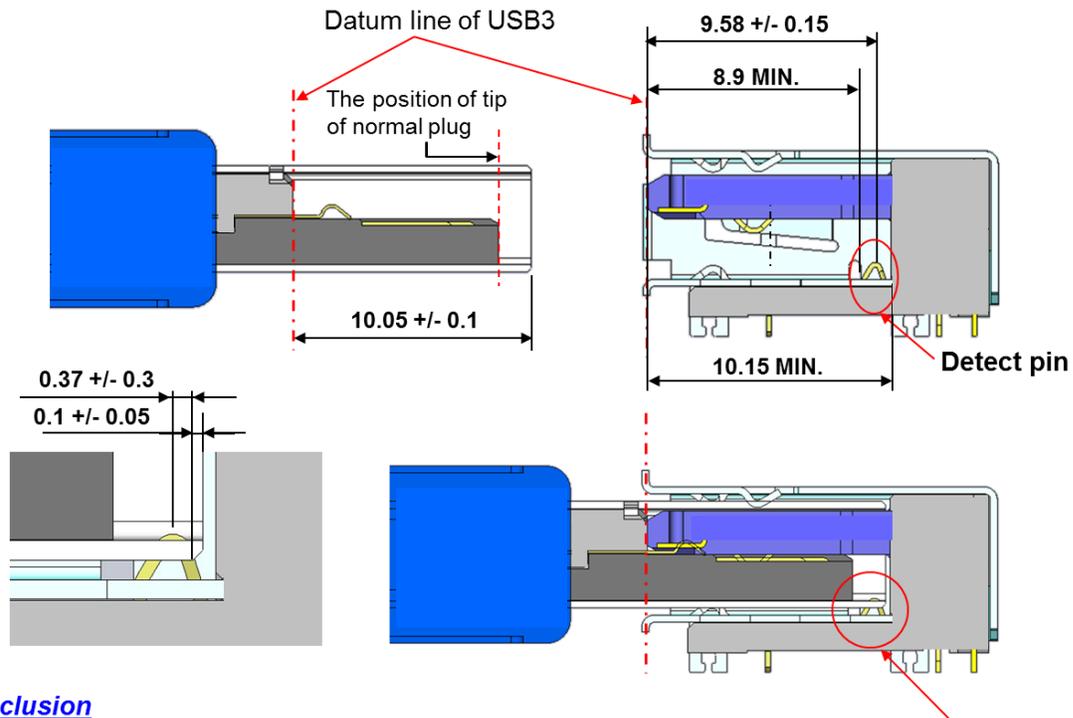
Figure D-2 USB 3.1 PD Standard-A Plug with USB 2.0 or 3.1 Standard-A Receptacle



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

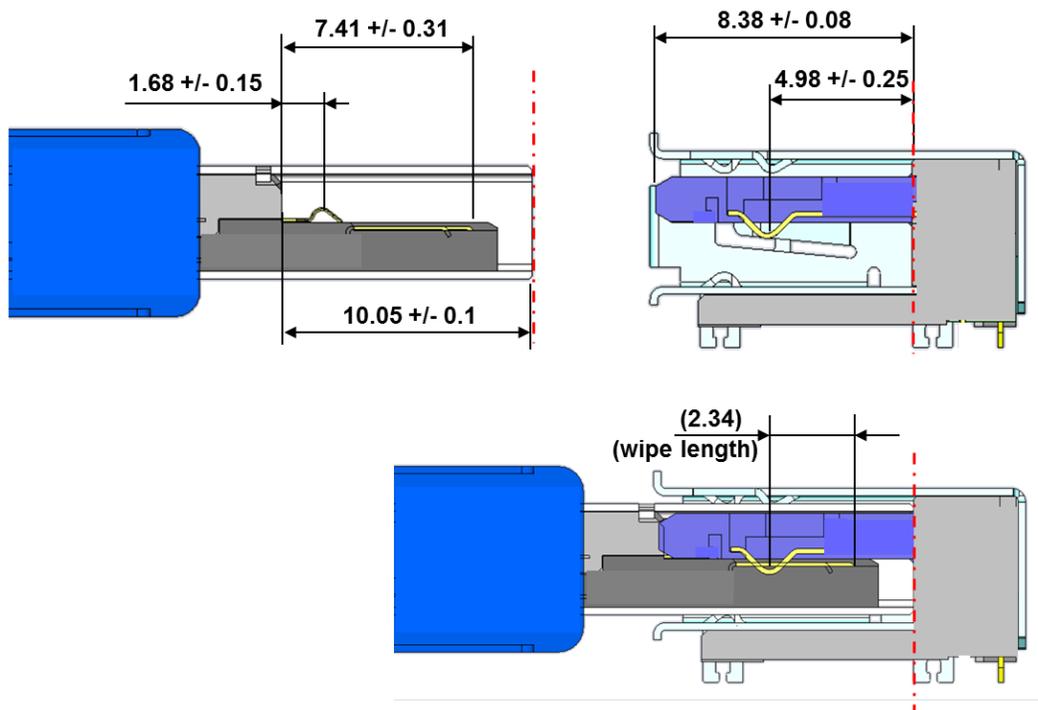


Conclusion

- 1) Connectors are identified as PD connectors due to connecting with detect pin.
- 2) Wipe length of detect pin is 0.37 +/- 0.3mm in normal mating condition.

Tip of plug shell will contact detect pin.

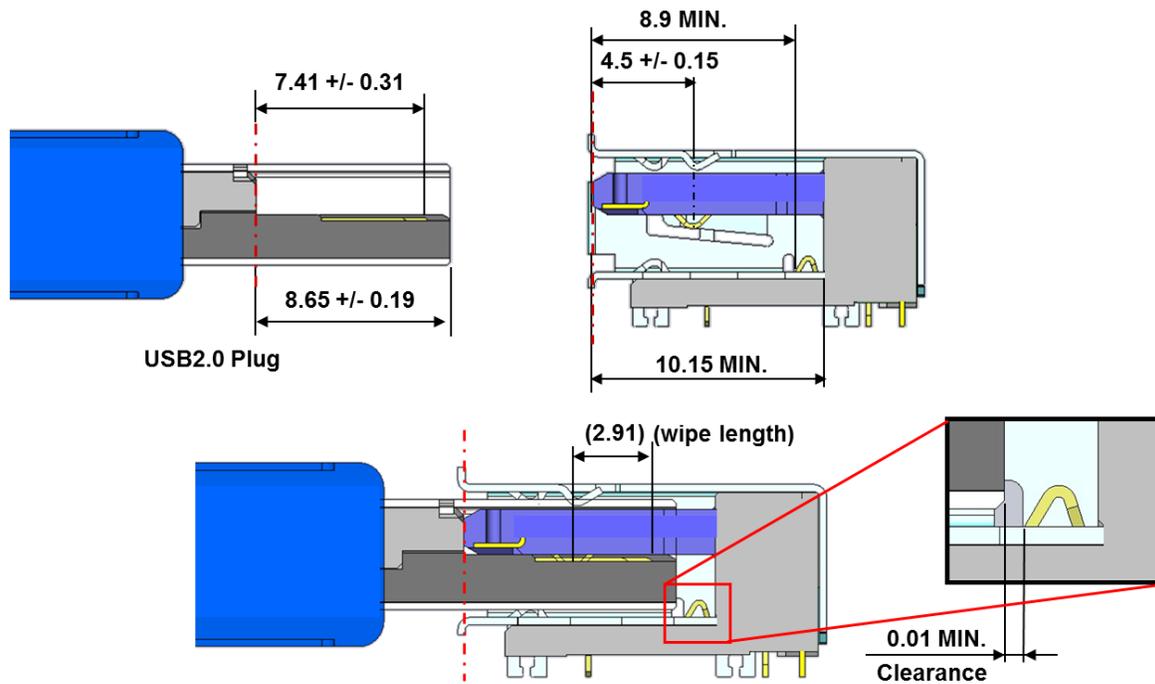
Figure D-4 USB 2.0 PD or 3.1 PD Standard-A Plug with USB 2.0 Standard-A Receptacle



Conclusion

USB2 legacy receptacle does not have detect pins, so PD plug will work as normal USB2 plug.

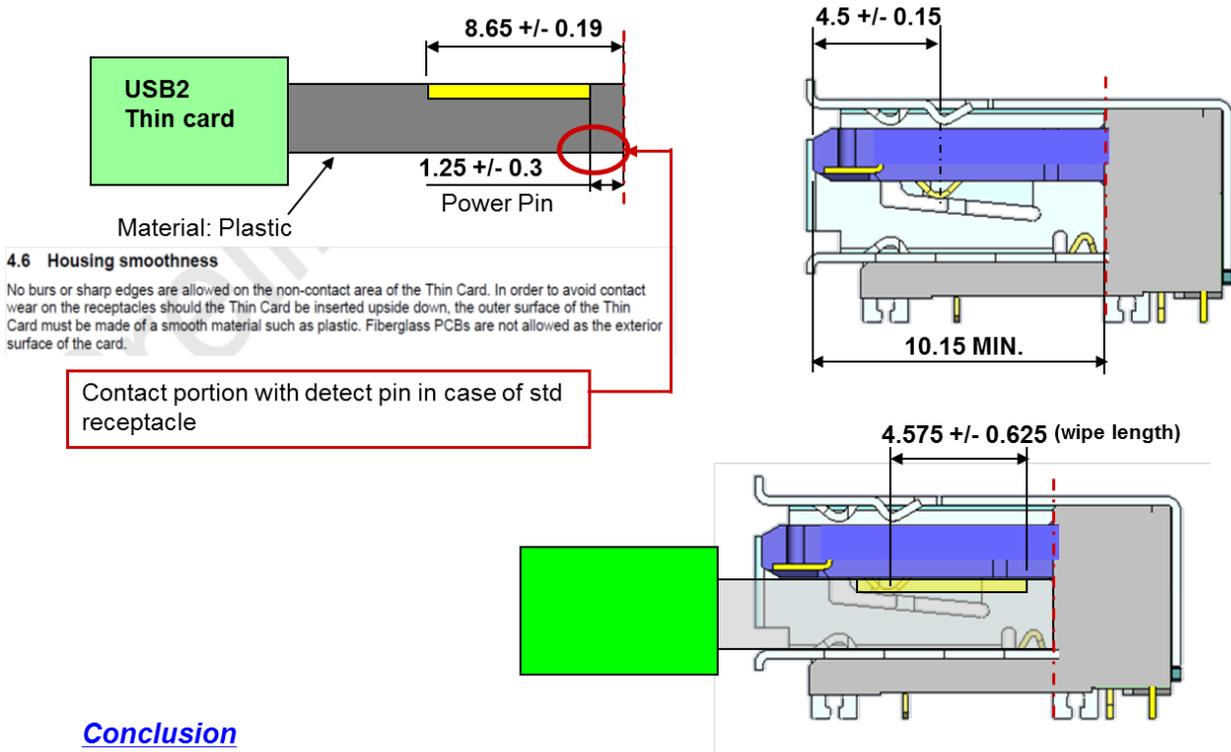
Figure D-5 USB 2.0 Standard-A Plug with USB 2.0 or USB3.1 PD Standard-A Receptacle



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.

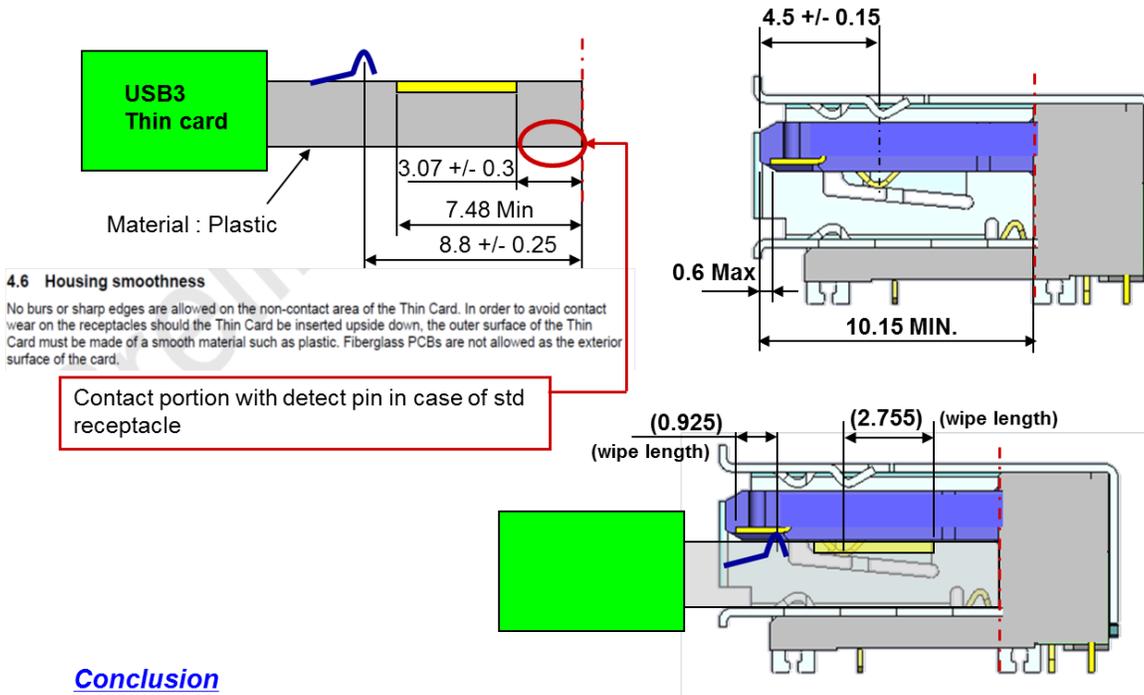
Figure D-6 USB 2.0 Thin Card with USB 2.0 PD or 3.1 PD Standard-A Receptacle



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

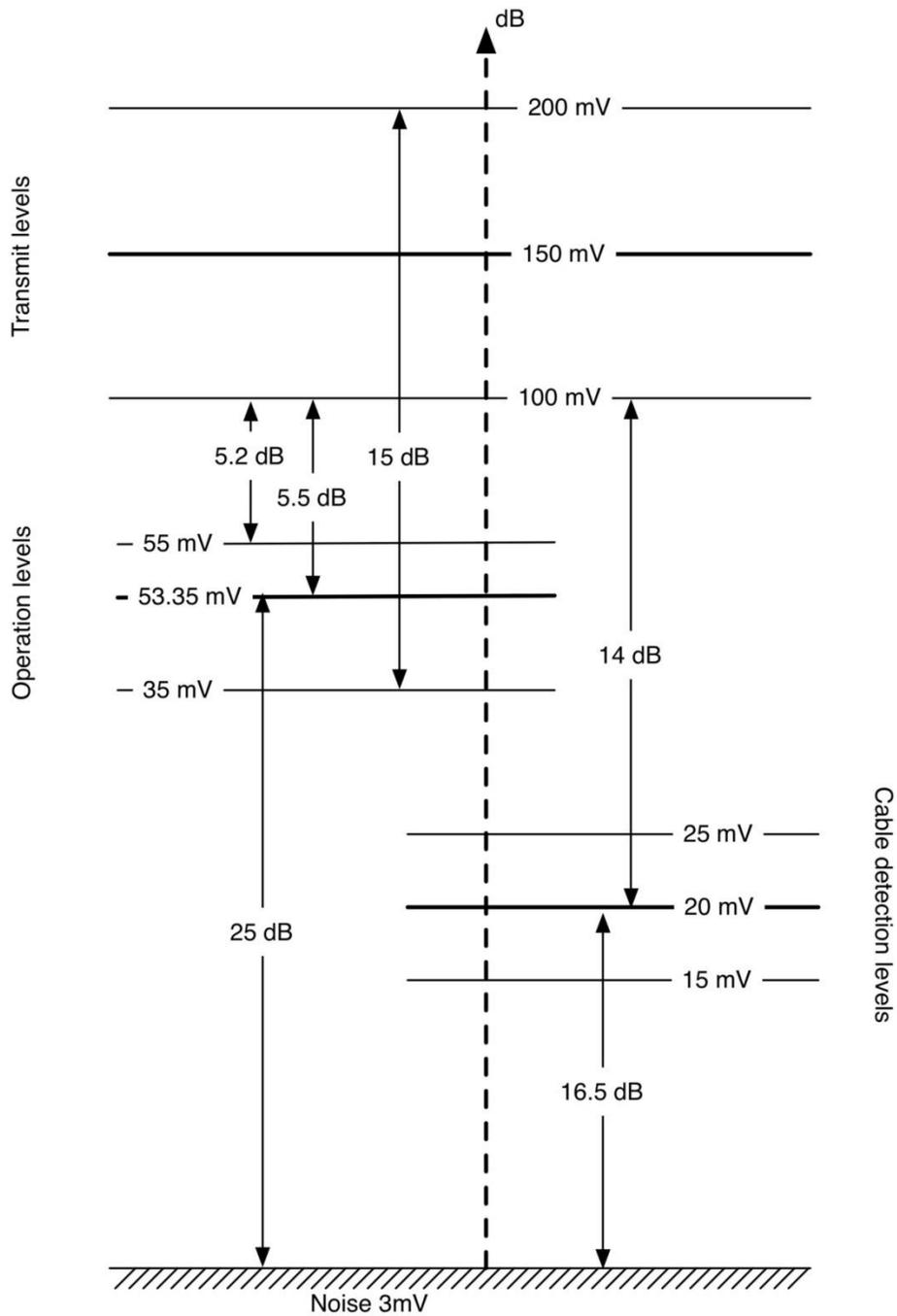


Figure E-1 shows the signal levels in mV (rms) needed to illustrate the squelch budget for normal operation (*vSquelchOperating*) and cable-type detection operation (*vSquelchDetecting*). The transmission level of the signal is between 100mV and 200mV, or at least 26.5dB above a 3mV noise floor. During 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 might 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 might be applied; it does not contain any **Normative** requirements.

In these illustrations all Ports are assumed to implement cold-socket functionality. V_{BUS} is only applied when 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 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.

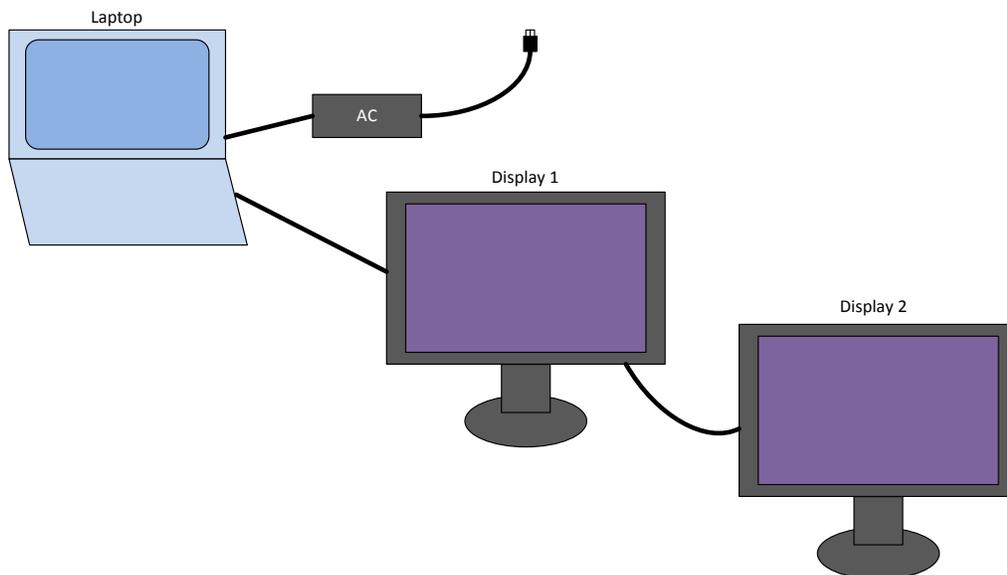
Although the Hubs in these illustrations support Power Delivery on both their UFPs and DFPs this is only one possible Hub implementation.

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 Enhanced SuperSpeed Port. According to the Source Power Rules described in Section 10.2 this means that the Port has a PD Power of 60W and so can supply: 5V@3A, 9V@3A, 15V@3A and 20V@3A.

- 2) Display 1 requires 30W to display and therefore a PD Power of 60W to operate itself plus Display 2 connected downstream. Display 1 initially uses 15V@2A to operate itself, since this also allows operation with a Source of 30W PD Power. On connection of Display 2, Display 1 will move to operation at 20V@3A to allow operation of the additional 30W ganged display. According to the Sink Power Rules described in Section 10.3 this means that Display 1 requires a Source with a PD Power of 60W to fully operate. Display 1 contains a Hub allowing Display 2 to be connected to Display 1.
- 3) Display 2 requires 30W operate itself and does not support an additional display connected downstream. Display 1 uses 15V@2A to operate itself from a Source of 30W PD Power.
- 4) In USB suspend Display 1 and Display 2 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	Power (W)
Display 1					
1	Connected to wall supply	Unattached	Unattached		0
2	Display 1 Attached, V _{BUS} powered.	Attached, drawing 5V@150mA.	Unattached		0.75
3	Set of Source Capabilities sent including: 5V@3A (15W), 9V@3A (27W), 15V@3A(45W) and 20V@3A (60W). The Unconstrained Power 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 15V@2A (30W) from laptop	Unattached	Display 1 knows it needs 15V@2A (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 15V@2A. Display 1 turns on and starts operating.	Unattached	Laptop evaluates the request, finds that it can meet this and so sends an accept.	30
Display 2					

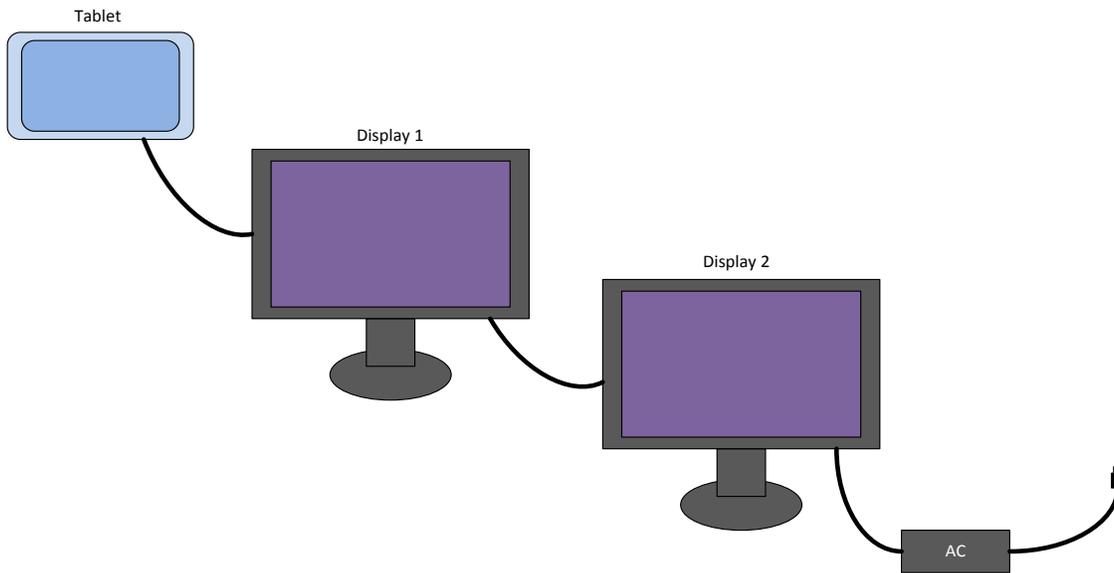
Step	Laptop	Display 1	Display 2	Device Policy Manager	Power (W)
7	Powering Display 1	Detects attach	Attached, no V_{BUS}		30
8	Request received	Display 1 requests 20V@1.73A (34.6W) from Laptop.	Attached, no V_{BUS}	Display 1 detects attach and requests additional 4.5W of power for USB 3.1 Port.	30
9	Sends Accept	Accept received.	Attached, no V_{BUS}		34.6
10	Sends PS_RDY	PS_RDY received	Attached, no V_{BUS}		
11		Powers V_{BUS}	Attached, drawing 5V@150mA.		34.6
12		Sends out Source Capabilities including: 5V@0.9A to Display 2. The Unconstrained Power 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 15V@2A.		34.6

Step	Laptop	Display 1	Display 2	Device Policy Manager	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 Unconstrained Power 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 15V@2A.		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 15V@2A, turns on and starts operating.	Display 2 now has the power it needs and can start working.	60
USB Suspend					
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 PDUSB 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	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 15V@2A.	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 5V@0.2A (1W) during normal operation and up to 5V@2.4A (12W) in order to charge.
- 2) Display 1 requires 30W to operate and therefore a PD Power of 42W to operate itself and charge the tablet. Display 1 uses 15V@2A to operate itself, since this allows operation with a Source of 30W PD Power and then moves to operation at 20V@2.1A to allow charging of the laptop. According to the Sink Power Rules described in Section 10.3 this means that the Display 1 requires a Source with a PD Power of 42W to fully operate.
- 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 PD Power upstream.

Table F-2 External power is supplied upstream

Step	Tablet	Display 1	Display 2	Device Policy Manager	Power (W)
Display 1 - Dead Battery					
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	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@3A (15W), 9V@3A (27W), 15V@3A (45W) and 20V@3A (60W). The Unconstrained Power 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 15V@2A (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
Tablet – Power Role Swap					
9	Tablet is Attached to Display 1.	Attached, V_{BUS} powered.			30
10	Tablet sends out a set of capabilities including: 5V@0.5A (2.5W). The Unconstrained Power bit cleared and USB suspend bit set.	Capabilities received			30

Step	Tablet	Display 1	Display 2	Device Policy Manager	Power (W)
11	Request received	Display 1 requests 5V@0A from the Tablet. The Unconstrained Power 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 5V@0.2A (1W) as a Sink.	Sink Capabilities received			30
16		Display 1 requests 15V@2.1A (31.5W) 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	Power (W)
18		PS_RDY received	Sends PS_RDY	Display 2 indicates that the power supply is ready to supply the power.	31.5
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.	31.5
20	Accept sent. Tablet turns off its V_{BUS} supply.	Accept received.		Tablet is happy to accept a Power Role Swap from any device offering it power.	31.5
21	Send PS_RDY	PS_RDY received. Display 1 turns on its V_{BUS} supply		Tablet indicates that its supply has been turned off.	31.5
22	PS_RDY received.	PS_RDY sent.		Display 1 indicates that its power supply is ready so the Tablet starts drawing power.	31.5
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 Unconstrained Power and USB suspend bits are set.			31.5
24	The Tablet requests 12V@0.2A.	Request received.		Tablet can now request the power it needs.	31.5
25	Accept received	Accept sent		Power is within the capabilities of Display 1 so it accepts the request.	31.5
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.	31.5

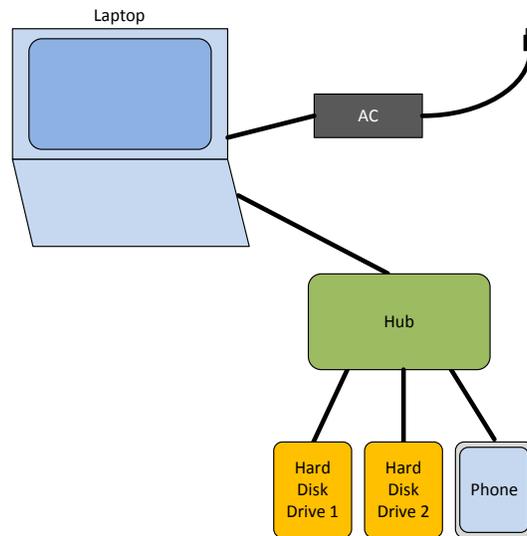
Step	Tablet	Display 1	Display 2	Device Policy Manager	Power (W)
Tablet – Charge					
27	Tablet requests 12V@0.2A (2.4W) from Display 1. The Tablet needs to charge and so sets the Capability Mismatch bit and the No USB Suspend bit.	Request received.		Tablet needs to charge but the power offered is not sufficient. Since Display 1 claims to have an external supply the Tablet will try to get more power using the Capability Mismatch Flag.	31.5
28	Accept received	Accept sent		A <i>Valid</i> request has been made so Display 1 accepts the request.	31.5
29	PS_RDY received	PS_RDY received		Tablet indicates a capability mismatch to the user.	31.5
30	Get Sink Capabilities received.	Get Sink Capabilities sent		Due to the Capability Mismatch Flag Display 1 requests Sink Capabilities from the Tablet.	31.5
31	The Tablet returns Sink capabilities containing: 5V@2.4A (12W). The Unconstrained Power bit is cleared.	Sink Capabilities received			31.5

Step	Tablet	Display 1	Display 2	Device Policy Manager	Power (W)
32		Display 1 requests 15V@2.8A (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.	31.5
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@3A (15W), 9V@3A (27W), 15V@3A (45W) and 20V@3A (60W). The Unconstrained Power 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	Power (W)
36		Display 1 requests 15V@2.8A (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@2.4A (12W). The Unconstrained Power 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 5V@2.4A (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 5V@2.4A (12W). 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 PD Power downstream.
- 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 5V@2A (10W) to spin up and 5V@1A (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)
Connect Hub					
1	Connected to wall supply	Detached	Detached		Default
2	Hub is Attached	Attached, V_{BUS} powered			Default
3	Laptop sends out a set of capabilities including: 5V@3A (15W), 12V@3A (36W), and 20V@3A (60W). The Unconstrained Power and USB suspend bits are set.	Source Capabilities received		Laptop sends out details of all available power via external supply	Default

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
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
Connect Hard Disk Drive 1					
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
11		Hub turns on V_{BUS} and sends out a set of capabilities to Hard Disk Drive 1 including: 5V@0.15A. The Unconstrained Power 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

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
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
Hard Disk Drive 1 spin up					
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
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 5V@2A.		1.5
20	Request received	The Hub requests 5V@2.2A (11W) from the Laptop.		The Hub evaluates that it now needs 0.75W for the Hub and 10W for Hard Disk Drive 1.	1.5

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
21	Accept sent	Accept received		Power request from the Hub is within the Laptop's capabilities so the Laptop accepts the request.	11
22	PS_RDY sent	PS_RDY received		Laptop completes the Explicit Contract.	11
23		Hub sends out a set of capabilities to Hard Disk Drive 1 including: 5V@2A. The Unconstrained Power and USB suspend bits are set.	Source Capabilities received	Hub now offers Hard Disk Drive 1 what it needs.	11
24		Request received	Hard Disk Drive 1 requests 5V@2A operating current and indicates 5V@2A maximum current.	Hard Disk Drive 1 is operating at its maximum current to spin up so sets operating current = maximum current.	11
25		Accept sent	Accept received	Request is within the Hubs capabilities so it accepts.	11
26		PS_RDY sent	PS_RDY received. Hard Disk Drive 1 starts to draw 5V@2A and spins up.	Hub indicates its power supply is ready so Hard Disk Drive 1 starts to draw power.	11
27		Request received	Once spun up Hard Disk Drive 1 requests 5V@1A operating current and 5V@2A maximum current.	Hard Disk Drive 1 is operating at a lower current so sets operating current < maximum current.	11
28		Accept sent	Accept received	The Hub will maintain a Power Reserve of 5V@1A (5W) for Hard Disk Drive 1 in addition to the 5V@1A (5W) it is currently using.	11

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
29		PS_RDY sent	PS_RDY received	Hub completes the Explicit Contract.	11
Hard Disk Drive 2 spin up					
30		Attach detected	Hard Disk Drive 2 is Attached to one of the downstream ports of the Hub.		11
31	Request received	The Hub requests 5V@2.3A (11.5W) 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.	11
32	Accept sent	Accept received		Power request from the Hub is within the Laptop's capabilities so it accepts the request.	11
33	PS_RDY sent	PS_RDY received		Laptop indicates its power supply is ready.	11.5
34		Hub sends out a set of capabilities to Hard Disk Drive 2 including: 5V@0.15A. The Unconstrained Power and USB suspend bits are set.	Source Capabilities received by Hard Disk Drive 2	Hub offers Hard Disk Drive 2 enough power to enumerate.	11.5
35		Request received	Hard Disk Drive 2 requests 5V@0.15A from the Hub.		11.5

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
36		Accept sent to Hard Disk Drive 2	Accept received by Hard Disk Drive 2	Request is within available capabilities so the Hub accepts	11.5
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	11.5
Phone charge					
38		Attach detected	The phone is Attached to one of the downstream ports of the Hub.		11.5
39	Request received	The Hub Requests 5V@2.5A (12.5W) 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.	11.5
40	Accept sent	Accept received		Request is within available capabilities so the Laptop accepts	12.5
41	PS_RDY sent	PS_RDY received		Laptop indicates that its power supply is ready.	12.5
42		The Hub powers V_{BUS} and sends out a set of capabilities to the Phone including: 5V@0.15A. The Unconstrained Power and USB suspend bits are set.	Source Capabilities received by the Phone	The Hub offers the Phone 1 unit load to enumerate.	12.5

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
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.	12.5
44		Accept sent	Accept received	Request is within available capabilities so the Hub accepts	12.5
45		PS_RDY sent	PS_RDY received	Hub indicates that its power supply is ready	12.5
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	12.5
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	12.5
48	Request received	The Hub Requests 9V@2.4A (21.6W) 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.	12.5
49	Accept sent	Accept received		Request is within available capabilities so the Laptop accepts	12.5
50	PS_RDY sent	PS_RDY received		Laptop indicates that its power supply is ready.	21.6

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
51		The Hub sends out a set of capabilities to the Phone including: 5V@2A. The Unconstrained Power 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.	21.6
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.	21.6
53		Accept sent to the Phone	Accept received by the Phone		21.6
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		21.6
55	Request received	The Hub Requests 9V@1.9A (17.1W) 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.	21.6

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
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.	21.6
57	PS_RDY sent	PS_RDY received		Laptop indicates that its power supply is ready.	17.1
Hard Disk Drive 2 spin up					
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	17.1
59		Accept sent	Accept received	The request is within its capabilities so the Hub accepts.	17.1
60		PS_RDY sent	PS_RDY received	The Hub indicates that its power supply is ready.	17.1
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	17.1
62		Sink Capabilities received	Hard Disk Drive 2 returns capabilities indicating that it requires 20V@0.5A maximum current.		17.1

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
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.	17.1
64			The Phone drops to zero current draw.		17.1
65		PD_RDY sent	PS_RDY received.	Hub indicates that its power supply has changed to the new level.	17.1
66	Request received	The Hub Requests 9V@2.4A (21.6W) 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.	17.1
67	Accept sent	Accept received		Request is within available capabilities so the Laptop accepts.	17.1
68	PS_RDY sent	PS_RDY received		Laptop indicates that its power supply is ready.	21.6

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
69		Hub sends out a set of capabilities to Hard Disk Drive 2 including: 5V@0.5A and 20V@0.5A. The Unconstrained Power 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.	21.6
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.	21.6
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.	21.6
72		PS_RDY sent.	PS_RDY sent. Hard Disk Drive 2 starts to draw 20V@0.5A and spins up.		21.6
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.	21.6
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.	21.6
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.	21.6
76		The Hub sends out a set of capabilities to the Phone including: 5V@2A. The Unconstrained Power 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	21.6

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
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.	21.6
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.	21.6
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.	21.6

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
Message Header		
15	<i>Reserved</i>	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	<i>Reserved</i>	0
3...0	<i>Message Type</i>	1111b (Vendor Defined Message)
VDM Header		
B31...16	Standard or Vendor ID (SVID)	0xFF00 (<i>PD SID</i>)
B15	VDM Type	1 (Structured VDM)
B14...13	Structured VDM Version	01b (Version 2.0)
B12...11	<i>Reserved</i>	00b
B10...8	Object Position	000b
B7...6	Command Type	00b (Initiator)
B5	<i>Reserved</i>	0
B4...0	Command ¹	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
Message Header		
15	<i>Reserved</i>	0
14...12	<i>Number of Data Objects</i>	5 (VDM Header + ID Header VDO + Cert Stat VDO + Product VDO + Cable VDO)
11...9	<i>MessageID</i>	0...7
8	<i>Cable Plug</i>	1
7...6	<i>Specification Revision</i>	01b
5...4	<i>Reserved</i>	0
3...0	<i>Message Type</i>	1111b (Vendor Defined Message)

Bit(s)	Field	Value
VDM Header		
B31...16	Standard or Vendor ID (SVID)	0xFF00 (<i>PD SID</i>)
B15	VDM Type	1 (Structured VDM)
B14...13	Structured VDM Version	01b (Version 2.0)
B12...11	Reserved	00b
B10...8	Object Position	000b
B7...6	Command Type	01b (Responder ACK)
B5	Reserved	0
B4...0	Command	1 (<i>Discover Identity</i>)
ID Header VDO		
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
Cert Stat VDO		
B31...20	Reserved, Shall be set to zero.	0
B19...0	20-bit unsigned integer	USB-IF assigned TID for this cable
Product VDO		
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
Cable VDO (returned for Product Type "Active Cable")		
B31...28	HW Version	Cable HW version number (vendor defined)
B27...24	Firmware Version	Cable FW version number (vendor defined)
B23...20	Reserved	0
B19...18	Type-C to USB 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 _{BUS} Current Handling Capability	01b (3A)
B4	V _{BUS} 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 Hub Responder Example

Bit(s)	Field	Value
Message Header		
15	Reserved	0

Bit(s)	Field	Value
14...12	<i>Number of Data Objects</i>	4 (VDM Header + ID Header VDO + Cert 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)
VDM Header		
B31...16	Standard or Vendor ID (SVID)	0xFF00 (<i>PD SID</i>)
B15	VDM Type	1 (Structured VDM)
B14...13	Structured VDM Version	01b (Version 2.0)
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>)
ID Header VDO		
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
Cert Stat VDO		
B31...20	Reserved Shall be set to zero.	0
B19...0	20-bit unsigned integer	USB-IF assigned TID for this hub
Product VDO		
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
Message Header		
15	<i>Reserved</i>	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	<i>Reserved</i>	0
3...0	<i>Message Type</i>	1111b (Vendor Defined Message)
VDM Header		
B31...16	Standard or Vendor ID (SVID)	0xFF00 (<i>PD SID</i>)
B15	VDM Type	1 (Structured VDM)
B14...13	Structured VDM Version	01b (Version 2.0)
B12...11	<i>Reserved</i>	00b
B10...8	Object Position	000b
B7...6	Command Type	00b (Initiator)
B5	<i>Reserved</i>	0
B4...0	Command ¹	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 not used.

Table G-5 Discover SVIDs Command response from Responder Example

Bit(s)	Field	Value
Message Header		
15	<i>Reserved</i>	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	<i>Reserved</i>	0
3...0	<i>Message Type</i>	1111b (Vendor Defined Message)
VDM Header		
B31...16	Standard or Vendor ID (SVID)	0xFF00 (<i>PD SID</i>)
B15	VDM Type	1 (Structured VDM)

Bit(s)	Field	Value
B14...13	Structured VDM Version	01b (Version 2.0)
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>)
VDO 1		
B31...16	SVID 0	SVID value
B15...0	SVID 1	SVID value
VDO 2		
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
Message Header		
15	<i>Reserved</i>	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	<i>Reserved</i>	0
3...0	<i>Message Type</i>	1111b (Vendor Defined Message)
VDM Header		
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	01b (Version 2.0)
B12...11	<i>Reserved</i>	00b
B10...8	Object Position	000b
B7...6	Command Type	00b (Initiator)
B5	<i>Reserved</i>	0
B4...0	Command ¹	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
Message Header		
15	<i>Reserved</i>	0
14...12	<i>Number of Data Objects</i>	2 (VDM Header + 1 Mode VDO)
11...9	<i>MessageID</i>	0...7
8	<i>Port Power Role</i>	0 or 1

Bit(s)	Field	Value
7...6	<i>Specification Revision</i>	01b
5...4	<i>Reserved</i>	0
3...0	<i>Message Type</i>	1111b (Vendor Defined Message)
VDM Header		
B31...16	Standard or Vendor ID (SVID)	SVID for which Modes were requested
B15	VDM Type	1 (Structured VDM)
B14...13	Structured VDM Version	01b (Version 2.0)
B12...11	<i>Reserved</i>	00b
B10...8	Object Position	000b
B7...6	Command Type	01b (Responder ACK)
B5	<i>Reserved</i>	0
B4...0	Command	3 (<i>Discover Modes</i>)
Mode VDO		
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
Message Header		
15	<i>Reserved</i>	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	<i>Reserved</i>	0
3...0	<i>Message Type</i>	1111b (Vendor Defined Message)
VDM Header		
B31...16	Standard or Vendor ID (SVID)	SVID for the Mode being entered
B15	VDM Type	1 (Structured VDM)
B14...13	Structured VDM Version	01b (Version 2.0)
B12...11	<i>Reserved</i>	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	<i>Reserved</i>	0
B4...0	Command	4 (<i>Enter Mode</i>)

G.4.2 Enter Mode Command response

The Responder that is the target of the *Enter 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 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
Message Header		
15	<i>Reserved</i>	0
14...12	<i>Number of Data Objects</i>	1 (VDM Header)
11...9	<i>MessageID</i>	0...7

Bit(s)	Field	Value
8	<i>Port Power Role</i>	0 or 1
7...6	<i>Specification Revision</i>	01b
5...4	<i>Reserved</i>	0
3...0	<i>Message Type</i>	1111b (Vendor Defined Message)
VDM Header		
B31...16	Standard or Vendor ID (SVID)	SVID for the Mode entered
B15	VDM Type	1 (Structured VDM)
B14...13	Structured VDM Version	01b (Version 2.0)
B12...11	<i>Reserved</i>	00b
B10...8	Object Position	001b (offset of the Mode entered)
B7...6	Command Type	01b (Responder ACK)
B5	<i>Reserved</i>	0
B4...0	Command	4 (<i>Enter Mode</i>)

G.4.1 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* 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 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 with an additional VDO.

Table G-10 Enter Mode Command request from Initiator Example

Bit(s)	Field	Value
Message Header		
15	<i>Reserved</i>	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	<i>Reserved</i>	0
3...0	<i>Message Type</i>	1111b (Vendor Defined Message)
VDM Header		
B31...16	Standard or Vendor ID (SVID)	SVID for the Mode being entered
B15	VDM Type	1 (Structured VDM)
B14...13	Structured VDM Version	01b (Version 2.0)
B12...11	<i>Reserved</i>	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	<i>Reserved</i>	0
B4...0	Command	4 (<i>Enter Mode</i>)
Including optional Mode specific VDO		
B31...0	Mode specific	

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
Message Header		
15	<i>Reserved</i>	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	<i>Reserved</i>	0
3...0	<i>Message Type</i>	1111b (Vendor Defined Message)
VDM Header		
B31...16	Standard or Vendor ID (SVID)	SVID for the Mode being exited
B15	VDM Type	1 (Structured VDM)
B14...13	Structured VDM Version	01b (Version 2.0)
B12...11	<i>Reserved</i>	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	<i>Reserved</i>	0
B4...0	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
Message Header		
15	<i>Reserved</i>	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)
VDM Header		
B31...16	Standard or Vendor ID (SVID)	SVID for the Mode exited
B15	VDM Type	1 (Structured VDM)
B14...13	Structured VDM Version	01b (Version 2.0)
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
Message Header		
15	Reserved	0
14...12	Number of Data Objects	1 (VDM Header)
11...9	MessageID	0...7
8	Port Power Role	0 or 1
7...6	Specification Revision	01b
5...4	Reserved	0
3...0	Message Type	1111b (Vendor Defined Message)
VDM Header		
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	01b (Version 2.0)
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	Command	6 (Attention)

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
Message Header		
15	Reserved	0
14...12	Number of Data Objects	2 (VDM Header + VDO)
11...9	MessageID	0...7
8	Port Power Role	0 or 1
7...6	Specification Revision	01b
5...4	Reserved	0

Bit(s)	Field	Value
3...0	<i>Message Type</i>	1111b (Vendor Defined Message)
VDM Header		
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	01b (Version 2.0)
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	Command	6 (<i>Attention</i>)
Including optional Mode specific VDO		
B31...0	Mode specific	