IEC 62680-1-2

Edition 5.0 2021-03

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Universal serial bus interfaces for data and power
Part 1-2: Common components – USB Power Delivery specification

Interfaces de bus universel en série pour les données et l'alimentation électrique
Partie 1-2: Composants communs – Spécification de l'alimentation électrique par port USB

IEC 62680-1-2:2021-03(en-fr)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

About the IEC
The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications
The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigendum or an amendment might have been published.

IEC publications search - webstore.iec.ch/advsearchform
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee, …). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and once a month by email.

IEC Customer Service Centre - webstore.iec.ch/csc
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

IEC online collection - oc.iec.ch
Discover our powerful search engine and read freely all the publications previews. With a subscription you will always have access to up to date content tailored to your needs.

Electropedia - www.electropedia.org
The world's leading online dictionary on electrotechnology, containing more than 22 000 terminological entries in English and French, with equivalent terms in 18 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

A propos de l'IEC
La Commission Electrotechnique Internationale (IEC) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications IEC
Le contenu technique des publications IEC est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

Recherche de publications IEC - webstore.iec.ch/advsearchform
La recherche avancée permet de trouver des publications IEC en utilisant différents critères (numéro de référence, texte, comité d'études, …). Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

IEC Just Published - webstore.iec.ch/justpublished
Restez informé sur les nouvelles publications IEC. Just Published détaille les nouvelles publications parues. Disponible en ligne et une fois par mois par email.

Service Clients - webstore.iec.ch/csc
Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: sales@iec.ch.

IEC online collection - oc.iec.ch
Découvrez notre puissant moteur de recherche et consultez gratuitement tous les aperçus des publications. Avec un abonnement, vous aurez toujours accès à un contenu à jour adapté à vos besoins.

Electropedia - www.electropedia.org
Le premier dictionnaire d'électrotechnologie en ligne au monde, avec plus de 22 000 articles terminologiques en anglais et en français, ainsi que les termes équivalents dans 16 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

IEC 62680-1-2

Edition 5.0 2021-03

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Universal serial bus interfaces for data and power
Part 1-2: Common components – USB Power Delivery specification

Interfaces de bus universel en série pour les données et l'alimentation électrique
Partie 1-2: Composants communs – Spécification de l'alimentation électrique par port USB

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

ICS 29.220; 33.120; 35.200

ISBN 978-2-8322-9288-4

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

UNIVERSAL SERIAL BUS INTERFACES FOR DATA AND POWER

Part 1-2: Common components – USB Power Delivery specification

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 62680-1-2 has been prepared by technical area 18: Multimedia home systems and applications for end-user networks, of IEC technical committee 100: Audio, video and multimedia systems and equipment.

The text of this standard was prepared by the USB Implementers Forum (USB-IF). The structure and editorial rules used in this publication reflect the practice of the organization which submitted it.

The text of this International Standard is based on the following documents:

CDV	Report on voting
100/3440/CDV	100/3505/RVC

Full information on the voting for the approval of this International Standard can be found in the report on voting indicated in the above table.

This document has been drafted in accordance with the ISO/IEC Directives, Part 2.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under "http://webstore.iec.ch" in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IMPORTANT – The "colour inside" logo on the cover page of this document indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

The IEC 62680 series is based on a series of specifications that were originally developed by the USB Implementers Forum (USB-IF). These specifications were submitted to the IEC under the auspices of a special agreement between the IEC and the USB-IF.

This standard is the USB-IF publication Universal Serial Bus Power Delivery Specification Revision 3.0, Version 2.0.

The USB Implementers Forum, Inc.(USB-IF) is a non-profit corporation founded by the group of companies that developed the Universal Serial Bus specification. The USB-IF was formed to provide a support organization and forum for the advancement and adoption of Universal Serial Bus technology. The Forum facilitates the development of high-quality compatible USB peripherals (devices), and promotes the benefits of USB and the quality of products that have passed compliance testing.

ANY USB SPECIFICATIONS ARE PROVIDED TO YOU "AS IS, "WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE. THE USB IMPLEMENTERS FORUM AND THE AUTHORS OF ANY USB SPECIFICATIONS DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY PROPRIETARY RIGHTS, RELATING TO USE OR IMPLEMENTATION OR INFORMATION IN THIS SPECIFICAITON.

THE PROVISION OF ANY USB SPECIFICATIONS TO YOU DOES NOT PROVIDE YOU WITH ANY LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS.

Entering into USB Adopters Agreements may, however, allow a signing company to participate in a reciprocal, RAND-Z licensing arrangement for compliant products. For more information, please see:

https://www.usb.org/documents

IEC DOES NOT TAKE ANY POSITION AS TO WHETHER IT IS ADVISABLE FOR YOU TO ENTER INTO ANY USB ADOPTERS AGREEMENTS OR TO PARTICIPATE IN THE USB IMPLEMENTERS FORUM.

Universal Serial Bus
Power Delivery Specification

Revision: ***3.0***

Version: ***2.0***

Release date: ***29 August 2019***

LIMITED COPYRIGHT LICENSE

THE USB 3.0 PROMOTERS GRANT A CONDITIONAL COPYRIGHT LICENSE UNDER THE COPYRIGHTS EMBODIED IN THE USB POWER DELIVERY SPECIFICATION TO USE AND REPRODUCE THE SPECIFICATION FOR THE SOLE PURPOSE OF, AND SOLELY TO THE EXTENT NECESSARY FOR, EVALUATING WHETHER TO IMPLEMENT THE SPECIFICATION IN PRODUCTS THAT WOULD COMPLY WITH THE SPECIFICATION. WITHOUT LIMITING THE FOREGOING, USE THE OF SPECIFICATION FOR THE PURPOSE OF FILING OR MODIFYING ANY PATENT APPLICATION TO TARGET THE SPECIFICATION OR USB COMPLIANT PRODUCTS IS NOT AUTHORIZED. EXCEPT FOR THIS EXPRESS COPYRIGHT LICENSE, NO OTHER RIGHTS OR LICENSES ARE GRANTED, INCLUDING WITHOUT LIMITATION ANY PATENT LICENSES. IN ORDER TO OBTAIN ANY ADDITIONALY INTELLECTUAL PROPERTY LICENSES OR LICENSING COMMITMENTS ASSOCIATED WITH THE SPECIFICATION A PARTY MUST EXECUTE THE USB 3.0 ADOPTERS AGREEMENT. NOTE: BY USING THE SPECIFICATION, YOU ACCEPT THESE LICENSE TERMS ON YOUR OWN BEHALF AND, IN THE CASE WHERE YOU ARE DOING THIS AS AN EMPLOYEE, ON BEHALF OF YOUR EMPLOYER.

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

USB Type-C® and USB4™ are trademarks of the Universal Serial Bus Implementers Forum (USB-IF). Thunderbolt™ is a trademark of Intel Corporation.

You may only use the Thunderbolt™ trademark or logo in conjunction with products designed to this specification that complete proper certification and executing a Thunderbolt™ trademark license – see http://usb.org/compliance for further information.

Chairs

Alvin Cox	Cabling Sub-Chair
Bob Dunstan	Specification Chair/Protocol Subgroup Chair
Deric Waters	PHY Chair
Ed Berrios	Power Supply Chair
Rahman Ismail	System Policy Chair
Richard Petrie	Specification Chair/Device Policy Chair

Editors

Bob Dunstan

Richard Petrie

Contributors

Charles Wang	ACON, Advanced-Connectek, Inc.	Jie min	Cadence Design Systems, Inc.
Conrad Choy	ACON, Advanced-Connectek, Inc.	Mark Summers	Cadence Design Systems, Inc.
Dennis Chuang	ACON, Advanced-Connectek, Inc.	Michal Staworko	Cadence Design Systems, Inc.
Steve Sedio	ACON, Advanced-Connectek, Inc.	Sathish Kumar Ganesan	Cadence Design Systems, Inc.
Sunney Yang	ACON, Advanced-Connectek, Inc.	Alessandro Ingrassia	Canova Tech
Vicky Chuang	ACON, Advanced-Connectek, Inc.	Andrea Colognese	Canova Tech
Joseph Scanlon	Advanced Micro Devices	Antonio Orzelli	Canova Tech
Caspar Lin	Allion Labs, Inc.	Davide Ghedin	Canova Tech
Casper Lee	Allion Labs, Inc.	Matteo Casalin	Canova Tech
Danny Shih	Allion Labs, Inc.	Michael Marioli	Canova Tech
Howard Chang	Allion Labs, Inc.	Nicola Scantamburlo	Canova Tech
Greg Stewart	Analogix Semiconductor, Inc.	Paolo Pilla	Canova Tech
Mehran Badii	Analogix Semiconductor, Inc.	Yi-Feng Lin	Canyon Semiconductor
Alexei Kosut	Apple	YuHung Lin	Canyon Semiconductor
Bill Cornelius	Apple	David Tsai	Chrontel, Inc.
Carlos Colderon	Apple	Anshul Gulati	Cypress Semiconductor
Chris Uiterwijk	Apple	Anup Nayak	Cypress Semiconductor
Colin Whitby-Strevens	Apple	Benjamin Kropf	Cypress Semiconductor
Corey Axelowitz	Apple	Dhanraj Rajput	Cypress Semiconductor
Corey Lange	Apple	Ganesh Subramaniam	Cypress Semiconductor
Dave Conroy	Apple	Jagadeesan Raj	Cypress Semiconductor
David Sekowski	Apple	Junjie cui	Cypress Semiconductor
Girault Jones	Apple	Manu Kumar	Cypress Semiconductor
James Orr	Apple	Muthu M	Cypress Semiconductor
Jason Chung	Apple	Nicholas Bodnaruk	Cypress Semiconductor
Jay Kim	Apple	Pradeep Bajpai	Cypress Semiconductor
Jeff Wilcox	Apple	Rajaram R	Cypress Semiconductor
Jennifer Tsai	Apple	Rama Vakkantula	Cypress Semiconductor
Karl Bowers	Apple	Rushil Kadakia	Cypress Semiconductor
Keith Porthouse	Apple	Simon Nguyen	Cypress Semiconductor
Kevin Hsiue	Apple	Steven Wong	Cypress Semiconductor
Matt Mora	Apple	Subu Sankaran	Cypress Semiconductor
Paul Baker	Apple	Sumeet Gupta	Cypress Semiconductor
Reese Schreiber	Apple	Tejender Sheoran	Cypress Semiconductor
Ruchi Chaturvedi	Apple	Venkat Mandagulathar	Cypress Semiconductor
Sameer Kelkar	Apple	Xiaofeng Shen	Cypress Semiconductor
Sasha Tietz	Apple	Zeng Wei	Cypress Semiconductor
Scott Jackson	Apple	Adie Tan	Dell Inc.
Sree Raman	Apple	Adolfo Montero	Dell Inc.
William Ferry	Apple	Bruce Montag	Dell Inc.
Zaki Moussaoui	Apple	Gary Verdun	Dell Inc.
Jeff Liu	ASMedia Technology Inc.	Marcin Nowak	Dell Inc.
Kuo Lung Li	ASMedia Technology Inc.	Merle Wood	Dell Inc.
Ming-Wei Hsu	ASMedia Technology Inc.	Mohammed Hijazi	Dell Inc.
PS Tseng	ASMedia Technology Inc.	Siddhartha Reddy	Dell Inc.
Sam Tzeng	ASMedia Technology Inc.	Bindhu Vasu	Dialog Semiconductor (UK) Ltd
Thomas Hsu	ASMedia Technology Inc.	Chanchal Gupta	Dialog Semiconductor (UK) Ltd
Weikao Chang	ASMedia Technology Inc.	Dipti Baheti	Dialog Semiconductor (UK) Ltd
Yang Cheng	ASMedia Technology Inc.	Duc Doan	Dialog Semiconductor (UK) Ltd
Shawn Meng	Bizlink Technology Inc.	Holger Petersen	Dialog Semiconductor (UK) Ltd
Bernard Shyu	Bizlink Technology, Inc.	Jianming Yao	Dialog Semiconductor (UK) Ltd
Eric Wu	Bizlink Technology, Inc.	John Shi	Dialog Semiconductor (UK) Ltd
Morphy Hsieh	Bizlink Technology, Inc.	KE Hong	Dialog Semiconductor (UK) Ltd
Sean O'Neal	Bizlink Technology, Inc.	Kevin Mori	Dialog Semiconductor (UK) Ltd
Tiffany Hsiao	Bizlink Technology, Inc.	Larry Ping	Dialog Semiconductor (UK) Ltd
Weichung Ooi	Bizlink Technology, Inc.	Mengfei Liu	Dialog Semiconductor (UK) Ltd
Rahul Bhushan	Broadcom Corp.	Scott Brown	Dialog Semiconductor (UK) Ltd
Asila nahas	Cadence Design Systems, Inc.	Yimin Chen	Dialog Semiconductor (UK) Ltd
Claire Ying	Cadence Design Systems, Inc.	Yong Li	Dialog Semiconductor (UK) Ltd

Dan Ellis	DisplayLink (UK) Ltd.	Alan Berkema	Hewlett Packard
Jason Young	DisplayLink (UK) Ltd.	Lee Atkinson	Hewlett Packard
Kevin Jacobs	DisplayLink (UK) Ltd.	Rahul Lakdawala	Hewlett Packard
Paulo Alcobia	DisplayLink (UK) Ltd.	Robin Castell	Hewlett Packard
Peter Burgers	DisplayLink (UK) Ltd.	Roger Benson	Hewlett Packard
Richard Petrie	DisplayLink (UK) Ltd.	Ron Schooley	Hewlett Packard
Abel Astley	Ellisys	Hideyuki HAYAFUJI	Hosiden Corporation
Chuck Trefts	Ellisys	Keiji Mine	Hosiden Corporation
Emmanuel Durin	Ellisys	Masaki YAMAOKA	Hosiden Corporation
Mario Pasquali	Ellisys	Takashi MUTO	Hosiden Corporation
Tim Wei	Ellisys	Yasunori NISHIKAWA	Hosiden Corporation
Chien-Cheng Kuo	Etron Technology, Inc.	Kenneth Chan	HP Inc.
Jack Yang	Etron Technology, Inc.	Lee Atkinson	HP Inc.
Richard Crisp	Etron Technology, Inc.	Steve Chen	HP Inc.
Shyanjia Chen	Etron Technology, Inc.	Suketu Partiwala	HP Inc.
TsungTa Lu	Etron Technology, Inc.	Suketu Partiwala	HP Inc.
Christian Klein	Fairchild Semiconductor	Vaibhav Malik	HP Inc.
Oscar Freitas	Fairchild Semiconductor	Walter Fry	HP Inc.
Souhib Harb	Fairchild Semiconductor	Bai Sean	Huawei Technologies Co., Ltd.
Amanda Ying	Feature Integration Technology Inc.	Chunjiang Zhao	Huawei Technologies Co., Ltd.
Jacky Chan	Feature Integration Technology Inc.	JianQuan Wu	Huawei Technologies Co., Ltd.
Kenny Hsieh	Feature Integration Technology Inc.	Li Zongjian	Huawei Technologies Co., Ltd.
KungAn Lin	Feature Integration Technology Inc.	Lihua Duan	Huawei Technologies Co., Ltd.
Paul Yang	Feature Integration Technology Inc.	Min Chen	Huawei Technologies Co., Ltd.
su Jaden	Feature Integration Technology Inc.	Wang Feng	Huawei Technologies Co., Ltd.
Yu-Lin Chu	Feature Integration Technology Inc.	Wei Haihong	Huawei Technologies Co., Ltd.
Yulin Lan	Feature Integration Technology Inc.	Robert Heaton	Indie Semiconductor
AJ Yang	Foxconn / Hon Hai	Vincent Wang	Indie Semiconductor
Bob Hall	Foxconn / Hon Hai	Sie Boo Chiang	Infineon Technologies
Fred Fons	Foxconn / Hon Hai	Tue Fatt David Wee	Infineon Technologies
Jie zheng	Foxconn / Hon Hai	Wee Tar Richard Ng	Infineon Technologies
Patrick Casher	Foxconn / Hon Hai	Wolfgang Furtner	Infineon Technologies
Steve Sedio	Foxconn / Hon Hai	Bob Dunstan	Intel Corporation
Terry Little	Foxconn / Hon Hai	Brad Saunders	Intel Corporation
Bob McVay	Fresco Logic Inc.	Chee Lim Nge	Intel Corporation
Christopher Meyers	Fresco Logic Inc.	Christine Krause	Intel Corporation
Dian Kurniawan	Fresco Logic Inc.	Dan Froelich	Intel Corporation
Tom Burton	Fresco Logic Inc.	David Harriman	Intel Corporation
Adam Rodriguez	Google Inc.	David Hines	Intel Corporation
Alec Berg	Google Inc.	David Thompson	Intel Corporation
Dave Bernard	Google Inc.	Guobin Liu	Intel Corporation
David Schneider	Google Inc.	Harry Skinner	Intel Corporation
Jim Guerin	Google Inc.	Henrik Leegaard	Intel Corporation
Juan Fantin	Google Inc.	Jenn Chuan Cheng	Intel Corporation
Ken Wu	Google Inc.	Jervis Lin	Intel Corporation
Mark Hayter	Google Inc.	John Howard	Intel Corporation
Nithya Jagannathan	Google Inc.	Karthi Vadivelu	Intel Corporation
Srikanth Lakshmikanthan	Google Inc.	Leo Heiland	Intel Corporation
Todd Broch	Google Inc.	Maarit Harkonen	Intel Corporation
Toshak Singhal	Google Inc.	Nge Chee Lim	Intel Corporation
Vincent Palatin	Google Inc.	Paul Durley	Intel Corporation
Xuelin Wu	Google Inc.	Rahman Ismail	Intel Corporation
Alan Kinningham	Granite River Labs	Rajaram Regupathy	Intel Corporation
Balamurugan Manialagan	Granite River Labs	Ronald Swartz	Intel Corporation
Mike Engbretson	Granite River Labs	Sarah Sharp	Intel Corporation
Mike Wu	Granite River Labs	Scott Brenden	Intel Corporation
Mukesh Tatiya	Granite River Labs	Sridharan Ranganathan	Intel Corporation
Rajaraman V	Granite River Labs	Steve McGowan	Intel Corporation
Tim Lin	Granite River Labs	Tim McKee	Intel Corporation

Toby Opferman	Intel Corporation
Ziv Kabiry	Intel Corporation
Jia Wei	Intersil Corporation
Al Hsiao	ITE Tech. Inc.
Greg Song	ITE Tech. Inc.
Richard Guo	ITE Tech. Inc.
Victor Lin	ITE Tech. Inc.
Y.C. Chou	ITE Tech. Inc.
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.
Jit Lim	Keysight Technologies Inc.
Babu Mailachalam	Lattice Semiconductor Corp
Gianluca Mariani	Lattice Semiconductor Corp
Joel Coplen	Lattice Semiconductor Corp
Thomas Watza	Lattice Semiconductor Corp
Vesa Lauri	Lattice Semiconductor Corp
Keneth Kim	LG electronics
Bruce Chuang	Leadtrend
Eilian Liu	Leadtrend
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
Aaron Melgar	Lion Semiconductor
Chris Zhou	Lion Semiconductor
Sehyung Jeon	Lion Semiconductor
Wonyoung Kim	Lion Semiconductor
Yongho Kim	Lion Semiconductor
Dave Thompson	LSI Corporation
Alan Kinningham	Luxshare-ICT
Daniel Chen	Luxshare-ICT
Eric Wen	Luxshare-ICT
James Stevens	Luxshare-ICT
Josue Castillo	Luxshare-ICT
Pat Young	Luxshare-ICT
Scott Shuey	Luxshare-ICT
Chikara Kakizawa	Maxim Integrated Products
Jacob Scott	Maxim Integrated Products
Ken Helfrich	Maxim Integrated Products
Michael Miskho	Maxim Integrated Products
Chris Yokum	MCCI Corporation
Geert Knapen	MCCI Corporation
Terry Moore	MCCI Corporation
Velmurugan Selvaraj	MCCI Corporation
Satoru Kumashiro	MegaChips Corporation
Brian Marley	Microchip Technology Inc.
Dave Perchlik	Microchip Technology Inc.
Don Perkins	Microchip Technology Inc.
Fernando Gonzalez	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.
Prasanna Vengateshan	Microchip Technology Inc.
Rich Wahler	Microchip Technology Inc.
Richard Petrie	Microchip Technology Inc.
Ronald Kunin	Microchip Technology Inc.
Shannon Cash	Microchip Technology Inc.
Thomas Farkas	Microchip Technology Inc.
Andrew Yang	Microsoft Corporation
Anthony Chen	Microsoft Corporation
Arvind Murching	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
Adib Al Abaji	Molex LLC
Aaron Xu	Monolithic Power Systems Inc.
Bo Zhou	Monolithic Power Systems Inc.
Christian Sporck	Monolithic Power Systems Inc.
Di Han	Monolithic Power Systems Inc.
Zhihong Yu	Monolithic Power Systems Inc.
Dan Wagner	Motorola Mobility Inc.
Ben Crowe	MQP Electronics Ltd.
Pat Crowe	MQP Electronics Ltd.
Sten Carlsen	MQP Electronics Ltd.
Kenji Oguma	NEC Corporation
Frank Borngräber	Nokia Corporation
Kai Inha	Nokia Corporation
Pekka Leinonen	Nokia Corporation
Richard Petrie	Nokia Corporation
Sten Carlsen	Nokia Corporation
Abhijeet Kulkarni	NXP Semiconductors
Ahmad Yazdi	NXP Semiconductors
Bart Vertenten	NXP Semiconductors
Dennis Ha	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
Winston Langeslag	NXP Semiconductors
Robert Heaton	Obsidian Technology
Andrew Yoo	ON Semiconductor
Brady Maasen	ON Semiconductor
Bryan McCoy	ON Semiconductor

Christian Klein	ON Semiconductor	Matti Kulmala	Salcomp Plc
Cor Voorwinden	ON Semiconductor	Toni Lehimo	Salcomp Plc
Edward Berrios	ON Semiconductor	Tong Kim	Samsung Electronics Co. Ltd.
Michael Smith	ON Semiconductor	Alvin Cox	Seagate Technology LLC
Oscar Freitas	ON Semiconductor	Emmanuel Lemay	Seagate Technology LLC
Tom Duffy	ON Semiconductor	John Hein	Seagate Technology LLC
Craig Wiley	Parade Technologies Inc.	Marc Noblitt	Seagate Technology LLC
Aditya Kulkarni	Power Integrations	Michael Morgan	Seagate Technology LLC
Amruta Patra	Power Integrations	Ronald Rueckert	Seagate Technology LLC
Rahul Joshi	Power Integrations	Tony Priborsky	Seagate Technology LLC
Ricardo Pregiteer	Power Integrations	Chin Chang	Semtech Corporation
Shruti Anand	Power Integrations	Tom Farkas	Semtech Corporation
Amit gupta	Qualcomm, Inc	Ning Dai	Silergy Corp.
George Paparrizos	Qualcomm, Inc	Wanfeng Zhang	Silergy Corp.
Giovanni Garcea	Qualcomm, Inc	Kafai Leung	Silicon Laboratories, Inc.
Jack Pham	Qualcomm, Inc	Kok Hong Soh	Silicon Laboratories, Inc.
James Goel	Qualcomm, Inc	Sorin Badiu	Silicon Laboratories, Inc.
Joshua Warner	Qualcomm, Inc	Steven Ghang	Silicon Laboratories, Inc.
Karyn Vuong	Qualcomm, Inc	Abhishek Sardeshpande	SiliConch Systems Private Limited
Lalan Mishra	Qualcomm, Inc	Aniket Mathad	SiliConch Systems Private Limited
Vamsi Samavedam	Qualcomm, Inc	Chandana N	SiliConch Systems Private Limited
Vatsal Patel	Qualcomm, Inc	Jaswanth Ammineni	SiliConch Systems Private Limited
Chris Sporck	Qualcomm, Inc.	Jinisha Patel	SiliConch Systems Private Limited
Craig Aiken	Qualcomm, Inc.	Kaustubh Kumar	SiliConch Systems Private Limited
Narendra Mehta	Qualcomm, Inc.	Nitish Nitish	SiliConch Systems Private Limited
Terry Remple	Qualcomm, Inc.	Pavitra Balasubramanian	SiliConch Systems Private Limited
Will Kun	Qualcomm, Inc.	Rakesh Polasa	SiliConch Systems Private Limited
Yoram Rimoni	Qualcomm, Inc.	Satish Anand Verkila	SiliConch Systems Private Limited
Fan-Hau Hsu	Realtek Semiconductor Corp.	Shubham Paliwal	SiliConch Systems Private Limited
Tsung-Peng Chuang	Realtek Semiconductor Corp.	Vishnu Pusuluri	SiliConch Systems Private Limited
Atsushi Mitamura	Renesas Electronics Corp.	John Sisto	SMSC
Bob Dunstan	Renesas Electronics Corp.	Ken Gay	SMSC
Brian Allen	Renesas Electronics Corp.	Mark Bohm	SMSC
Dan Aoki	Renesas Electronics Corp.	Richard Wahler	SMSC
Hajime Nozaki	Renesas Electronics Corp.	Shannon Cash	SMSC
John Carpenter	Renesas Electronics Corp.	Tim Knowlton	SMSC
Kiichi Muto	Renesas Electronics Corp.	William Chiechi	SMSC
Masami Katagiri	Renesas Electronics Corp.	Shigenori Tagami	Sony Corporation
Nobuo Furuya	Renesas Electronics Corp.	Shinichi Hirata	Sony Corporation
Patrick Yu	Renesas Electronics Corp.	Amanda Hosler	Specwerkz
Peter Teng	Renesas Electronics Corp.	Bob Dunstan	Specwerkz
Philip Leung	Renesas Electronics Corp.	Diane Lenox	Specwerkz
Steve Roux	Renesas Electronics Corp.	Michael Munn	StarTech.com Ltd.
Tetsu Sato	Renesas Electronics Corp.	Fabien Friess	ST-Ericsson
Toshifumi Yamaoka	Renesas Electronics Corp.	Giuseppe Platania	ST-Ericsson
Chunan Kuo	Richtek Technology Corporation	Jean-Francois Gatto	ST-Ericsson
Heinz Wei	Richtek Technology Corporation	Milan Stamenkovic	ST-Ericsson
TZUHSIEN CHUANG	Richtek Technology Corporation	Nicolas Florenchie	ST-Ericsson
Tatsuya Irisawa	Ricoh Company Ltd.	Patrizia Milazzo	ST-Ericsson
Akihiro Ono	Rohm Co. Ltd.	Christophe Cochard	STMicroelectronics
Chris Lin	Rohm Co. Ltd.	Christophe Lorin	STMicroelectronics
Hidenori Nishimoto	Rohm Co. Ltd.	Filippo Bonaccorso	STMicroelectronics
Kris Bahar	Rohm Co. Ltd.	Jessy Guilbot	STMicroelectronics
Manabu Miyata	Rohm Co. Ltd.	Joel Huloux	STMicroelectronics
Ruben Balbuena	Rohm Co. Ltd.	John Bloomfield	STMicroelectronics
Takashi Sato	Rohm Co. Ltd.	Massimo Panzica	STMicroelectronics
Vijendra Kuroodi	Rohm Co. Ltd.	Meriem Mersel	STMicroelectronics
Yusuke Kondo	Rohm Co. Ltd.	Nathalie Ballot	STMicroelectronics
Kazuomi Nagai	ROHM Co., Ltd.	Pascal Legrand	STMicroelectronics

Patrizia Milazzo	STMicroelectronics
Richard O'Connor	STMicroelectronics
Morten Christiansen	Synopsys, Inc.
Nivin George	Synopsys, Inc.
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
Grant Ley	Texas Instruments
Gregory Watkins	Texas Instruments
Ingolf Frank	Texas Instruments
Ivo Huber	Texas Instruments
Javed Ahmad	Texas Instruments
Jean Picard	Texas Instruments
John Perry	Texas Instruments
Martin Patoka	Texas Instruments
Mike Campbell	Texas Instruments
Scott Jackson	Texas Instruments
Shafiuddin Mohammed	Texas Instruments
Srinath Hosur	Texas Instruments
Steven Tom	Texas Instruments
Yoon Lee	Texas Instruments
Tim Wilhelm	The Silanna Group Pty. Ltd.
Tod Wolf	The Silanna Group Pty. Ltd.
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.
Ho Wen Tsai	Weltrend Semiconductor
Hung Chiang	Weltrend Semiconductor
Jeng Cheng Liu	Weltrend Semiconductor
Priscilla Lee	Weltrend Semiconductor
Wayne Lo	Weltrend Semiconductor
Charles Neumann	Western Digital Technologies, Inc.
Curtis Stevens	Western Digital Technologies, Inc.
John Maroney	Western Digital Technologies, Inc.
Joe O'Brien	Wilder Technologies
Will Miller	Wilder Technologies
Juejia Zhou	Xiaomi Communications Co., Ltd.
Xiaoxing Yang	Xiaomi Communications Co., Ltd.

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 11-January-2017	11 January 2017
3.0	1.0	Initial release Revision 3.0	11 December 2015
3.0	1.0a	Including errata through 25-March-2016	25 March 2016
3.0	1.1	Including errata through 12-January-2016	12 January 2017
3.0	1.2	Including errata through 21-June-2018	21 June 2018
3.0	2.0	Including errata through 29-August-2019	29 August 2019

Table of Contents

INTELLECTUAL PROPERTY DISCLAIMER....6
Chairs....7
Editors....7
Contributors....7
Revision History....13
Table of Contents....14
List of Tables....22
List of Figures....30
1. Introduction....39
1.1 Overview....39
1.2 Purpose....40
1.3 Scope....40
1.4 Conventions....41
1.4.1 Precedence....41
1.4.2 Keywords....41
1.4.3 Numbering....42
1.5 Related Documents....42
1.6 Terms and Abbreviations....43
1.7 Parameter Values....51
1.8 Changes from Revision 2.0....51
1.9 Compatibility with Revision 2.0....51
2. Overview....52
2.1 Introduction....52
2.2 Section Overview....53
2.3 Compatibility with Revision 2.0....54
2.4 USB Power Delivery Capable Devices....54
2.5 SOP* Communication....55
2.5.1 Introduction....55
2.5.2 SOP* Collision Avoidance....55
2.5.3 SOP Communication....55

2.5.4 SOP'/SOP'' Communication with Cable Plugs 55
2.6 Operational Overview 57
2.6.1 Source Operation 57
2.6.2 Sink Operation 59
2.6.3 Cable Plugs 60
2.7 Architectural Overview 61
2.7.1 Policy 64
2.7.2 Message Formation and Transmission 65
2.7.3 Collision Avoidance 65
2.7.4 Power supply 66
2.7.5 DFP/UFP 66
2.7.6 Cable and Connectors 67
2.7.7 Interactions between Non-PD, BC and PD devices 67
2.7.8 Power Rules 67
3. USB Type-A and USB Type-B Cable Assemblies and Connectors 68
4. Electrical Requirements 69
4.1 Interoperability with other USB Specifications 69
4.2 Dead Battery Detection / Unpowered Port Detection 69
4.3 Cable IR Ground Drop (IR Drop) 69
4.4 Cable Type Detection 69
5. Physical Layer 70
5.1 Physical Layer Overview 70
5.2 Physical Layer Functions 70
5.3 Symbol Encoding 71
5.4 Ordered Sets 72
5.5 Transmitted Bit Ordering 73
5.6 Packet Format 74
5.6.1 Packet Framing 74
5.6.2 CRC 76
5.6.3 Packet Detection Errors 78
5.6.4 Hard Reset 78
5.6.5 Cable Reset 79
5.7 Collision Avoidance 79
5.8 Biphase Mark Coding (BMC) Signaling Scheme 80
5.8.1 Encoding and signaling 80

5.8.2 Transmit and Receive Masks 83
5.8.3 Transmitter Load Model 89
5.8.4 BMC Common specifications 90
5.8.5 BMC Transmitter Specifications 91
5.8.6 BMC Receiver Specifications 93
5.9 Built in Self-Test (BIST) 97
5.9.1 BIST Carrier Mode 97
5.9.2 BIST Test Data 97
6. Protocol Layer 98
6.1 Overview 98
6.2 Messages 98
6.2.1 Message Construction 98
6.3 Control Message 108
6.3.1 GoodCRC Message 109
6.3.2 GotoMin Message 109
6.3.3 Accept Message 109
6.3.4 Reject Message 110
6.3.5 Ping Message 110
6.3.6 PS_RDY Message 110
6.3.7 Get_Source_Cap Message 110
6.3.8 Get_Sink_Cap Message 111
6.3.9 DR_Swap Message 111
6.3.10 PR_Swap Message 111
6.3.11 VCONN_Swap Message 112
6.3.12 Wait Message 113
6.3.13 Soft Reset Message 114
6.3.14 Data_Reset Message 114
6.3.15 Data_Reset_Complete Message 115
6.3.16 Not_Supported Message 115
6.3.17 Get_Source_Cap_Extended Message 115
6.3.18 Get_Status Message 115
6.3.19 FR_Swap Message 115
6.3.20 Get_PPS_Status 116
6.3.21 Get_Country_Codes 116
6.3.22 Get_Sink_Cap_Extended Message 116

6.4 Data Message....116
6.4.1 Capabilities Message....117
6.4.2 Request Message....126
6.4.3 BIST Message....131
6.4.4 Vendor Defined Message....133
6.4.5 Battery_Status Message....162
6.4.6 Alert Message....163
6.4.7 Get_Country_Info Message....165
6.4.8 Enter_USB Message....165
6.5 Extended Message....167
6.5.1 Source_Capabilities_Extended Message....168
6.5.2 Status Message....172
6.5.3 Get_Battery_Cap Message....175
6.5.4 Get_Battery_Status Message....175
6.5.5 Battery_Capabilities Message....176
6.5.6 Get_Manufacturer_Info Message....177
6.5.7 Manufacturer_Info Message....177
6.5.8 Security Messages....178
6.5.9 Firmware Update Messages....179
6.5.10 PPS_Status Message....180
6.5.11 Country_Codes Message....181
6.5.12 Country_Info Message....182
6.5.13 Sink_Capabilities_Extended Message....182
6.6 Timers....186
6.6.1 CRCReceiveTimer....186
6.6.2 SenderResponseTimer....186
6.6.3 Capability Timers....186
6.6.4 Wait Timers and Times....187
6.6.5 Power Supply Timers....188
6.6.6 NoResponseTimer....189
6.6.7 BIST Timers....190
6.6.8 Power Role Swap Timers....190
6.6.9 Soft Reset Timers....190
6.6.10 Data Reset Timers....191
6.6.11 Hard Reset Timers....191
6.6.12 Structured VDM Timers....192

6.6.13 VCONN Timers 193
6.6.14 tCableMessage 193
6.6.15 DiscoverIdentityTimer 193
6.6.16 Collision Avoidance Timers 193
6.6.17 Fast Role Swap Timers 194
6.6.18 Chunking Timers 194
6.6.19 Programmable Power Supply Timers 195
6.6.20 tEnterUSB 195
6.6.21 Time Values and Timers 196
6.7 Counters 200
6.7.1 MessageID Counter 200
6.7.2 Retry Counter 200
6.7.3 Hard Reset Counter 201
6.7.4 Capabilities Counter 201
6.7.5 Discover Identity Counter 201
6.7.6 VDMBusyCounter 201
6.7.7 Counter Values and Counters 201
6.8 Reset 202
6.8.1 Soft Reset and Protocol Error 202
6.8.2 Data Reset 204
6.8.3 Hard Reset 204
6.8.4 Cable Reset 205
6.9 Collision Avoidance 205
6.10 Message Discarding 205
6.11 State behavior 207
6.11.1 Introduction to state diagrams used in Chapter 6 207
6.11.2 State Operation 207
6.11.3 List of Protocol Layer States 229
6.12 Message Applicability 231
6.12.1 Applicability of Control Messages 232
6.12.2 Applicability of Data Messages 233
6.12.3 Applicability of Extended Messages 234
6.12.4 Applicability of Structured VDM Commands 235
6.12.5 Applicability of Reset Signaling 236
6.12.6 Applicability of Fast Role Swap signal 236
6.13 Value Parameters 237

7. Power Supply 238
7.1 Source Requirements 238
7.1.1 Behavioral Aspects 238
7.1.2 Source Bulk Capacitance 238
7.1.3 Types of Sources 238
7.1.4 Source Transitions 239
7.1.5 Response to Hard Resets 246
7.1.6 Changing the Output Power Capability 247
7.1.7 Robust Source Operation 247
7.1.8 Output Voltage Tolerance and Range 248
7.1.9 Charging and Discharging the Bulk Capacitance on V_{BUS} 249
7.1.10 Swap Standby for Sources 249
7.1.11 Source Peak Current Operation 250
7.1.12 Source Capabilities Extended Parameters 251
7.1.13 Fast Role Swap 253
7.1.14 Non-application of V_{BUS} Slew Rate Limits 254
7.1.15 VCONN Power Cycle 255
7.2 Sink Requirements 256
7.2.1 Behavioral Aspects 256
7.2.2 Sink Bulk Capacitance 256
7.2.3 Sink Standby 257
7.2.4 Suspend Power Consumption 257
7.2.5 Zero Negotiated Current 257
7.2.6 Transient Load Behavior 257
7.2.7 Swap Standby for Sinks 258
7.2.8 Sink Peak Current Operation 258
7.2.9 Robust Sink Operation 258
7.2.10 Fast Role Swap 259
7.3 Transitions 261
7.3.1 Increasing the Current 262
7.3.2 Increasing the Voltage 264
7.3.3 Increasing the Voltage and Current 266
7.3.4 Increasing the Voltage and Decreasing the Current 268
7.3.5 Decreasing the Voltage and Increasing the Current 270
7.3.6 Decreasing the Current 272

7.3.7 Decreasing the Voltage 274
7.3.8 Decreasing the Voltage and the Current 276
7.3.9 Sink Requested Power Role Swap 278
7.3.10 Source Requested Power Role Swap 281
7.3.11 GotoMin Current Decrease 284
7.3.12 Source Initiated Hard Reset 286
7.3.13 Sink Initiated Hard Reset 288
7.3.14 No change in Current or Voltage 290
7.3.15 Fast Role Swap 292
7.3.16 Increasing the Programmable Power Supply Voltage 294
7.3.17 Decreasing the Programmable Power Supply Voltage 296
7.3.18 Changing the Source PDO or APDO 298
7.3.19 Increasing the Programmable Power Supply Current 300
7.3.20 Decreasing the Programmable Power Supply Current 302
7.3.21 Same Request Programmable Power Supply 304
7.4 Electrical Parameters 305
7.4.1 Source Electrical Parameters 305
7.4.2 Sink Electrical Parameters 309
7.4.3 Common Electrical Parameters 310
8. Device Policy 312
8.1 Overview 312
8.2 Device Policy Manager 312
8.2.1 Capabilities 313
8.2.2 System Policy 313
8.2.3 Control of Source/Sink 314
8.2.4 Cable Detection 314
8.2.5 Managing Power Requirements 314
8.2.6 Use of “Unconstrained Power” bit with Batteries and AC supplies 316
8.2.7 Interface to the Policy Engine 318
8.3 Policy Engine 319
8.3.1 Introduction 319
8.3.2 Atomic Message Sequence Diagrams 319
8.3.3 State Diagrams 479
9. States and Status Reporting 567
9.1 Overview 567

9.1.1 PDUSB Device and Hub Requirements 569
9.1.2 Mapping to USB Device States 569
9.1.3 PD Software Stack 572
9.1.4 PDUSB Device Enumeration 572
9.2 PD Specific Descriptors 574
9.2.1 USB Power Delivery Capability Descriptor 574
9.2.2 Battery Info Capability Descriptor 575
9.2.3 PD Consumer Port Capability Descriptor 576
9.2.4 PD Provider Port Capability Descriptor 576
9.3 PD Specific Requests and Events 578
9.3.1 PD Specific Requests 578
9.4 PDUSB Hub and PDUSB Peripheral Device Requests 579
9.4.1 GetBatteryStatus 579
9.4.2 SetPDFeature 580
10. Power Rules 582
10.1 Introduction 582
10.2 Source Power Rules 582
10.2.1 Source Power Rule Considerations 582
10.2.2 Normative Voltages and Currents 583
10.2.3 Optional Voltages/Currents 586
10.2.4 Power sharing between ports 588
10.3 Sink Power Rules 588
10.3.1 Sink Power Rule Considerations 588
10.3.2 Normative Sink Rules 588
A. CRC calculation 590
A.1 C code example 590
A.2 Table showing the full calculation over one Message 592
B. PD Message Sequence Examples 593
B.1 External power is supplied downstream 593
B.2 External power is supplied upstream 597
B.3 Giving back power 604
C. VDM Command Examples 616
C.1 Discover Identity Example 616
C.1.1 Discover Identity Command request 616
C.1.2 Discover Identity Command response – Active Cable 616

C.1.3 Discover Identity Command response – Hub 618
C.2 Discover SVIDs Example 619
C.2.1 Discover SVIDs Command request 619
C.2.1 Discover SVIDs Command response 619
C.3 Discover Modes Example 621
C.3.1 Discover Modes Command request 621
C.3.2 Discover Modes Command response 621
C.4 Enter Mode Example 623
C.4.1 Enter Mode Command request 623
C.4.2 Enter Mode Command response 623
C.4.1 Enter Mode Command request with additional VDO 624
C.5 Exit Mode Example 625
C.5.1 Exit Mode Command request 625
C.5.2 Exit Mode Command response 625
C.6 Attention Example 627
C.6.1 Attention Command request 627
C.6.2 Attention Command request with additional VDO 627
D. BMC Receiver Design Examples 629
D.1 Finite Difference Scheme 629
D.1.1 Sample Circuitry 629
D.1.2 Theory 629
D.1.3 Data Recovery 631
D.1.4 Noise Zone and Detection Zone 632
D.2 Subtraction Scheme 632
D.2.1 Sample Circuitry 632
D.2.2 Output of Each Circuit Block 633
D.2.3 Subtractor Output at Power Source and Power Sink 633
D.2.4 Noise Zone and Detection Zone 634
E. FRS System Level Example 635
E.1 Overview 635
E.2 FRS Initial Setup 637
E.3 FRS Process 639

List of Tables

Table 1-1 Terms and Abbreviations 43

Table 5-1 4b5b Symbol Encoding Table 71
Table 5-2 Ordered Sets 72
Table 5-3 Validation of Ordered Sets 72
Table 5-4 Data Size 73
Table 5-5 SOP ordered set 74
Table 5-6 SOP' ordered set 75
Table 5-7 SOP'' ordered set 75
Table 5-8 SOP'_Debug ordered set 76
Table 5-9 SOP''_Debug ordered set 76
Table 5-10 CRC-32 Mapping 77
Table 5-11 Hard Reset ordered set 78
Table 5-12 Cable Reset ordered set 79
Table 5-13 Rp values used for Collision Avoidance 80
Table 5-14 BMC Tx Mask Definition, X Values 84
Table 5-15 BMC Tx Mask Definition, Y Values 85
Table 5-16 BMC Rx Mask Definition 89
Table 5-17 BMC Common Normative Requirements 91
Table 5-18 BMC Transmitter Normative Requirements 91
Table 5-19 BMC Receiver Normative Requirements 94
Table 6-1 Message Header 99
Table 6-2 Revision Interoperability during an Explicit Contract 102
Table 6-3 Extended Message Header 103
Table 6-4 Use of Unchunked Message Supported bit 105
Table 6-5 Control Message Types 108
Table 6-6 Data Message Types 116
Table 6-7 Power Data Object 118
Table 6-8 Augmented Power Data Object 118
Table 6-9 Fixed Supply PDO - Source 120
Table 6-10 Fixed Power Source Peak Current Capability 122
Table 6-11 Variable Supply (non-Battery) PDO - Source 122
Table 6-12 Battery Supply PDO - Source 123
Table 6-13 Programmable Power Supply APDO - Source 123
Table 6-14 Fixed Supply PDO - Sink 124
Table 6-15 Variable Supply (non-Battery) PDO - Sink 126
Table 6-16 Battery Supply PDO - Sink 126
Table 6-17 Programmable Power Supply APDO - Sink 126
Table 6-18 Fixed and Variable Request Data Object 127
Table 6-19 Fixed and Variable Request Data Object with GiveBack Support 127

Table 6-20 Battery Request Data Object 127
Table 6-21 Battery Request Data Object with GiveBack Support 128
Table 6-22 Programmable Request Data Object 128
Table 6-23 BIST Data Object 132
Table 6-24 Unstructured VDM Header 134
Table 6-25 Structured VDM Header 135
Table 6-26 Structured VDM Commands 136
Table 6-27 SVID Values 136
Table 6-28 Commands and Responses 138
Table 6-29 ID Header VDO 140
Table 6-30 Product Types (UFP) 141
Table 6-31 Product Types (Cable Plug) 141
Table 6-32 Product Types (DFP) 141
Table 6-33 Cert Stat VDO 142
Table 6-34 Product VDO 142
Table 6-35 UFP VDO 1 143
Table 6-36 UFP VDO 2 144
Table 6-37 DFP VDO 145
Table 6-38 Passive Cable VDO 146
Table 6-39 Active Cable VDO 1 148
Table 6-40 Active Cable VDO 2 150
Table 6-41 AMA VDO 152
Table 6-42 VPD VDO 153
Table 6-43 Discover SVIDs Responder VDO 154
Table 6-44 Battery Status Data Object (BSDO) 162
Table 6-45 Alert Data Object 163
Table 6-46 Country Code Data Object 165
Table 6-47 Enter_USB Data Object 166
Table 6-48 Extended Message Types 167
Table 6-49 Source Capabilities Extended Data Block (SCEDB) 168
Table 6-50 SOP Status Data Block (SDB) 172
Table 6-51 SOP'/SOP'' Status Data Block (SDB) 175
Table 6-52 Get Battery Cap Data Block (GBCDB) 175
Table 6-53 Get Battery Status Data Block (GBSDB) 176
Table 6-54 Battery Capability Data Block (BCDB) 176
Table 6-55 Get Manufacturer Info Data Block (GMIDB) 177
Table 6-56 Manufacturer Info Data Block (MIDB) 178
Table 6-57 PPS Status Data Block (PPSSDB) 180

Table 6-58 Country Codes Data Block (CCDB) 181
Table 6-59 Country Info Data Block (CIDB) 182
Table 6-60 Sink Capabilities Extended Data Block (SKEDB) 183
Table 6-61 Time Values 197
Table 6-62 Timers 198
Table 6-63 Counter parameters 201
Table 6-64 Counters 202
Table 6-65 Response to an incoming Message (except VDM) 203
Table 6-66 Response to an incoming VDM 204
Table 6-67 Message discarding 206
Table 6-68 Protocol Layer States 229
Table 6-69 Applicability of Control Messages 232
Table 6-70 Applicability of Data Messages 233
Table 6-71 Applicability of Extended Messages 234
Table 6-72 Applicability of Structured VDM Commands 235
Table 6-73 Applicability of Reset Signaling 236
Table 6-74 Applicability of Fast Role Swap signal 236
Table 6-75 Value Parameters 237
Table 7-1 Sequence Description for Increasing the Current 263
Table 7-2 Sequence Description for Increasing the Voltage 265
Table 7-3 Sequence Diagram for Increasing the Voltage and Current 267
Table 7-4 Sequence Description for Increasing the Voltage and Decreasing the Current 269
Table 7-5 Sequence Description for Decreasing the Voltage and Increasing the Current 271
Table 7-6 Sequence Description for Decreasing the Current 273
Table 7-7 Sequence Description for Decreasing the Voltage 275
Table 7-8 Sequence Description for Decreasing the Voltage and the Current 277
Table 7-9 Sequence Description for a Sink Requested Power Role Swap 279
Table 7-10 Sequence Description for a Source Requested Power Role Swap 282
Table 7-11 Sequence Description for a GotoMin Current Decrease 285
Table 7-12 Sequence Description for a Source Initiated Hard Reset 287
Table 7-13 Sequence Description for a Sink Initiated Hard Reset 289
Table 7-14 Sequence Description for no change in Current or Voltage 291
Table 7-15 Sequence Description for Fast Role Swap 292
Table 7-16 Sequence Description for Increasing the Programmable Power Supply Voltage 294
Table 7-17 Sequence Description for Decreasing the Programmable Power Supply Voltage 297
Table 7-18 Sequence Description for Changing the Source PDO or APDO 298
Table 7-19 Sequence Description for increasing the Current in PPS mode 301
Table 7-20 Sequence Description for decreasing the Current in PPS mode 303

Table 7-21 Sequence Description for increasing the Current in PPS mode 304
Table 7-22 Source Electrical Parameters 305
Table 7-23 Sink Electrical Parameters 309
Table 7-24 Common Source/Sink Electrical Parameters 311
Table 8-1 Basic Message Flow 320
Table 8-2 Potential issues in Basic Message Flow 321
Table 8-3 Basic Message Flow with CRC failure 322
Table 8-4 Interruptible and Non-interruptible AMS 324
Table 8-5 Steps for a successful Power Negotiation 326
Table 8-6 Steps for a GotoMin Negotiation 329
Table 8-7 Steps for a successful Power Negotiation 331
Table 8-8 Steps for a Soft Reset 334
Table 8-9 Steps for a DFP Initiated Data Reset where the DFP is the Vconn Source 337
Table 8-10 Steps for a DFP Receiving a Data Reset where the DFP is the Vconn Source 340
Table 8-11 Steps for a DFP Initiated Data Reset where the UFP is the Vconn Source 343
Table 8-12 Steps for a DFP Receiving a Data Reset where the UFP is the Vconn Source 347
Table 8-13 Steps for Source initiated Hard Reset 351
Table 8-14 Steps for Sink initiated Hard Reset 354
Table 8-15 Steps for Source initiated Hard Reset – Sink long reset 357
Table 8-16 Steps for a Successful Source Initiated Power Role Swap Sequence 360
Table 8-17 Steps for a Successful Sink Initiated Power Role Swap Sequence 366
Table 8-18 Steps for a Successful Fast Role Swap Sequence 371
Table 8-19 Steps for Data Role Swap, UFP operating as Sink initiates 374
Table 8-20 Steps for Data Role Swap, UFP operating as Source initiates 376
Table 8-21 Steps for Data Role Swap, DFP operating as Source initiates 378
Table 8-22 Steps for Data Role Swap, DFP operating as Sink initiates 380
Table 8-23 Steps for Source to Sink VCONN Source Swap 383
Table 8-24 Steps for Sink to Source VCONN Source Swap 386
Table 8-25 Steps for Source Alert to Sink 389
Table 8-26 Steps for Sink Alert to Source 390
Table 8-27 Steps for a Sink getting Source Status Sequence 391
Table 8-28 Steps for a Source getting Sink Status Sequence 393
Table 8-29 Steps for a Sink getting Source PPS status Sequence 395
Table 8-30 Steps for a Sink getting Source Capabilities Sequence 397
Table 8-31 Steps for a Dual-Role Source getting Dual-Role Sink's capabilities as a Source Sequence 399
Table 8-32 Steps for a Source getting Sink Capabilities Sequence 401
Table 8-33 Steps for a Dual-Role Sink getting Dual-Role Source capabilities as a Sink Sequence 403
Table 8-34 Steps for a Sink getting Source extended capabilities Sequence 405

Table 8-35 Steps for a Dual-Role Source getting Dual-Role Sink extended capabilities Sequence 407
Table 8-36 Steps for a Sink getting Source Battery capabilities Sequence 409
Table 8-37 Steps for a Source getting Sink Battery capabilities Sequence 411
Table 8-38 Steps for a Sink getting Source Battery status Sequence 413
Table 8-39 Steps for a Source getting Sink Battery status Sequence 415
Table 8-40 Steps for a Source getting Sink's Port Manufacturer Information Sequence 417
Table 8-41 Steps for a Source getting Sink's Port Manufacturer Information Sequence 419
Table 8-42 Steps for a Source getting Sink's Battery Manufacturer Information Sequence 421
Table 8-43 Steps for a Source getting Sink's Battery Manufacturer Information Sequence 423
Table 8-44 Steps for a VCONN Source getting Sink's Port Manufacturer Information Sequence 425
Table 8-45 Steps for a Source getting Country Codes Sequence 427
Table 8-46 Steps for a Source getting Sink's Country Codes Sequence 429
Table 8-47 Steps for a VCONN Source getting Sink's Country Codes Sequence 431
Table 8-48 Steps for a Source getting Country Information Sequence 433
Table 8-49 Steps for a Source getting Sink's Country Information Sequence 435
Table 8-50 Steps for a VCONN Source getting Sink's Country Information Sequence 437
Table 8-51 Steps for a Source requesting a security exchange with a Sink Sequence 439
Table 8-52 Steps for a Sink requesting a security exchange with a Source Sequence 441
Table 8-53 Steps for a Vconn Source requesting a security exchange with a Cable Plug Sequence 443
Table 8-54 Steps for a Source requesting a firmware update exchange with a Sink Sequence 445
Table 8-55 Steps for a Sink requesting a firmware update exchange with a Source Sequence 447
Table 8-56 Steps for a Vconn Source requesting a firmware update exchange with a Cable Plug Sequence 449
Table 8-57 Steps for DFP to UFP Discover Identity 451
Table 8-58 Steps for Source Port to Cable Plug Discover Identity 453
Table 8-59 Steps for DFP to Cable Plug Discover Identity 455
Table 8-60 Steps for DFP to UFP Enter Mode 457
Table 8-61 Steps for DFP to UFP Exit Mode 459
Table 8-62 Steps for DFP to Cable Plug Enter Mode 462
Table 8-63 Steps for DFP to Cable Plug Exit Mode 464
Table 8-64 Steps for UFP to DFP Attention 466
Table 8-65 Steps for BIST Carrier Mode Test 468
Table 8-66 Steps for BIST Test Data Test 470
Table 8-67 Steps for UFP USB4 Mode Entry (Valid) 472
Table 8-68 Steps for Cable Plug USB4 Mode Entry (Valid) 474
Table 8-69 Steps for UFP USB4 Mode Entry (Invalid) 476
Table 8-70 Steps for Cable Plug USB4 Mode Entry (Invalid) 478
Table 8-71 Policy Engine States 560

Table 9-1 USB Power Delivery Type Codes....574
Table 9-2 USB Power Delivery Capability Descriptor....574
Table 9-3 Battery Info Capability Descriptor....575
Table 9-4 PD Consumer Port Descriptor....576
Table 9-5 PD Provider Port Descriptor....576
Table 9-6 PD Requests....578
Table 9-7 PD Request Codes....578
Table 9-8 PD Feature Selectors....578
Table 9-9 Battery Status Structure....579
Table 9-10 Battery Wake Mask....580
Table 9-11 Charging Policy Encoding....581
Table 10-1 Considerations for Sources....582
Table 10-2 Normative Voltages and Minimum Currents....583
Table 10-3 Fixed Supply PDO – Source 5V....585
Table 10-4 Fixed Supply PDO – Source 9V....585
Table 10-5 Fixed Supply PDO – Source 15V....585
Table 10-6 Fixed Supply PDO – Source 20V....585
Table 10-7 Programmable Power Supply PDOs and APDOs based on the PDP....587
Table 10-8 Programmable Power Supply Voltage Ranges....587
Table B-1 External power is supplied downstream....594
Table B-2 External power is supplied upstream....597
Table B-3 Giving back power....604
Table C-1 Discover Identity Command request from Initiator Example....616
Table C-2 Discover Identity Command response from Active Cable Responder Example....616
Table C-3 Discover Identity Command response from Hub Responder Example....618
Table C-4 Discover SVIDs Command request from Initiator Example....619
Table C-5 Discover SVIDs Command response from Responder Example....619
Table C-6 Discover Modes Command request from Initiator Example....621
Table C-7 Discover Modes Command response from Responder Example....621
Table C-8 Enter Mode Command request from Initiator Example....623
Table C-9 Enter Mode Command response from Responder Example....623
Table C-10 Enter Mode Command request from Initiator Example....624
Table C-11 Exit Mode Command request from Initiator Example....625
Table C-12 Exit Mode Command response from Responder Example....625
Table C-13 Attention Command request from Initiator Example....627
Table C-14 Attention Command request from Initiator with additional VDO Example....627
Table E-1: Sequence Table for setup of a Fast Role Swap (Hub connected to Power Adapter first)....637

Table E-2 Sequence Table for setup of a Fast Role Swap (Hub connected to Notebook before Power Adapter) 638
Table E-3 Sequence Table for slow Vbus discharge (it discharges after FR_Swap message is sent) 640
Table E-4 Vbus discharges quickly after adapter disconnected. 641

List of Figures

Figure 2-1 Logical Structure of USB Power Delivery Capable Devices 54
Figure 2-2 Example SOP' Communication between VCONN Source and Cable Plug(s) 56
Figure 2-3 USB Power Delivery Communications Stack 62
Figure 2-4 USB Power Delivery Communication Over USB 62
Figure 2-5 High Level Architecture View 64
Figure 5-1 Interpretation of ordered sets 72
Figure 5-2 Transmit Order for Various Sizes of Data 73
Figure 5-3 USB Power Delivery Packet Format 74
Figure 5-4 CRC 32 generation 77
Figure 5-5 Line format of Hard Reset 79
Figure 5-6 Line format of Cable Reset 79
Figure 5-7 BMC Example 80
Figure 5-8 BMC Transmitter Block Diagram 81
Figure 5-9 BMC Receiver Block Diagram 81
Figure 5-10 BMC Encoded Start of Preamble 81
Figure 5-11 Transmitting or Receiving BMC Encoded Frame Terminated by Zero with High-to-Low Last Transition 82
Figure 5-12 Transmitting or Receiving BMC Encoded Frame Terminated by One with High-to-Low Last Transition 82
Figure 5-13 Transmitting or Receiving BMC Encoded Frame Terminated by Zero with Low to High Last Transition 83
Figure 5-14 Transmitting or Receiving BMC Encoded Frame Terminated by One with Low to High Last Transition 83
Figure 5-15 BMC Tx 'ONE' Mask 84
Figure 5-16 BMC Tx 'ZERO' Mask 84
Figure 5-17 BMC Rx 'ONE' Mask when Sourcing Power 86
Figure 5-18 BMC Rx 'ZERO' Mask when Sourcing Power 87
Figure 5-19 BMC Rx 'ONE' Mask when Power neutral 87
Figure 5-20 BMC Rx 'ZERO' Mask when Power neutral 88
Figure 5-21 BMC Rx 'ONE' Mask when Sinking Power 88
Figure 5-22 BMC Rx 'ZERO' Mask when Sinking Power 89
Figure 5-23 Transmitter Load Model for BMC Tx from a Source 90
Figure 5-24 Transmitter Load Model for BMC Tx from a Sink 90
Figure 5-25 Transmitter diagram illustrating zDriver 92
Figure 5-26 Inter-Frame Gap Timings 93
Figure 5-27 Example Multi-Drop Configuration showing two DRPs 95
Figure 5-28 Example Multi-Drop Configuration showing a DFP and UFP 95
Figure 5-29 Test Data Frame 97

Figure 6-1 USB Power Delivery Packet Format including Control Message Payload 98
Figure 6-2 USB Power Delivery Packet Format including Data Message Payload 99
Figure 6-3 USB Power Delivery Packet Format including an Extended Message Header and Payload 99
Figure 6-4 Example Security_Request sequence Unchunked (Chunked bit = 0) 105
Figure 6-5 Example byte transmission for Security_Request Message of Data Size 7 (Chunked bit is set to 0) 105
Figure 6-6 Example byte transmission for Security_Response Message of Data Size 7 (Chunked bit is set to 0) 106
Figure 6-7 Example Security_Request sequence Chunked (Chunked bit = 1) 106
Figure 6-8 Example Security_Request Message of Data Size 7 (Chunked bit set to 1) 107
Figure 6-9 Example Chunk 0 of Security_Response Message of Data Size 30 (Chunked bit set to 1) 107
Figure 6-10 Example byte transmission for a Security_Response Message Chunk request (Chunked bit is set to 1) 107
Figure 6-11 Example Chunk 1 of Security_Response Message of Data Size 30 (Chunked bit set to 1) 108
Figure 6-12 Example Capabilities Message with 2 Power Data Objects 117
Figure 6-13 BIST Message 131
Figure 6-14 Vendor Defined Message 134
Figure 6-15 Discover Identity Command response 139
Figure 6-16 Discover Identity Command response for a DRD 139
Figure 6-17 Example Discover SVIDs response with 3 SVIDs 155
Figure 6-18 Example Discover SVIDs response with 4 SVIDs 155
Figure 6-19 Example Discover SVIDs response with 12 SVIDs followed by an empty response 155
Figure 6-20 Example Discover Modes response for a given SVID with 3 Modes 155
Figure 6-21 Successful Enter Mode sequence 157
Figure 6-22 Enter Mode sequence Interrupted by Source Capabilities and then Re-run 157
Figure 6-23 Unsuccessful Enter Mode sequence due to NAK 158
Figure 6-24 Exit Mode sequence 159
Figure 6-25 Attention Command request/response sequence 159
Figure 6-26 Command request/response sequence 160
Figure 6-27 Enter/Exit Mode Process 161
Figure 6-28 Battery_Status Message 162
Figure 6-29 Alert Message 163
Figure 6-30 Get_Country_Info Message 165
Figure 6-31 Enter_USB Message 165
Figure 6-32 Source_Capabilities_Extended Message 168
Figure 6-33 SOP Status Message 172
Figure 6-34 SOP'/SOP'' Status Message 174
Figure 6-35 Get_Battery_Cap Message 175
Figure 6-36 Get_Battery_Status Message 176

Figure 6-37 Battery_Capabilities Message 176
Figure 6-38 Get_Manufacturer_Info Message 177
Figure 6-39 Manufacturer_Info Message 178
Figure 6-40 Security_Request Message 179
Figure 6-41 Security_Response Message 179
Figure 6-42 Firmware_Update_Request Message 179
Figure 6-43 Firmware_Update_Response Message 180
Figure 6-44 PPS_Status Message 180
Figure 6-45 Country_Codes Message 181
Figure 6-46 Country_Info Message 182
Figure 6-47 Sink_Capabilities_Extended Message 182
Figure 6-48 Outline of States 207
Figure 6-49 References to states 207
Figure 6-50 Chunking architecture Showing Message and Control Flow 208
Figure 6-51 Chunked Rx State Diagram 210
Figure 6-52 Chunked Tx State Diagram 213
Figure 6-53 Chunked Message Router State Diagram 216
Figure 6-54 Common Protocol Layer Message Transmission State Diagram 218
Figure 6-55 Source Protocol Layer Message Transmission State Diagram 221
Figure 6-56 Sink Protocol Layer Message Transmission State Diagram 222
Figure 6-57 Protocol layer Message reception 224
Figure 6-58 Hard/Cable Reset 226
Figure 7-1 Placement of Source Bulk Capacitance 238
Figure 7-2 Transition Envelope for Positive Voltage Transitions 239
Figure 7-3 Transition Envelope for Negative Voltage Transitions 240
Figure 7-4 PPS Positive Voltage Transitions 241
Figure 7-5 PPS Negative Voltage Transitions 242
Figure 7-6 Expected PPS Ripple Relative to an LSB 242
Figure 7-7 PPS Programmable Voltage and Current Limit 244
Figure 7-8 iPpsCLOperatingDetail 245
Figure 7-9 PPS Programmable Voltage and Current Limit 246
Figure 7-10 Source V_{BUS} and V_{CONN} Response to Hard Reset 247
Figure 7-11 Application of vSrcNew and vSrcValid limits after tSrcReady 249
Figure 7-12 Source Peak Current Overload 250
Figure 7-13 Holdup Time Measurement 252
Figure 7-14 V_{BUS} Power during Fast Role Swap 253
Figure 7-15 V_{BUS} detection and timing during Fast Role Swap, initial V_{BUS} (at new source) > vSafe5V (min). 254

Figure 7-16 V_{BUS} detection and timing during Fast Role Swap, initial V_{BUS} (at new source) < vSafe5V (min). 254
Figure 7-17 Data Reset UFP V_{CONN} Power Cycle 255
Figure 7-18 Data Reset DFP V_{CONN} Power Cycle 256
Figure 7-19 Placement of Sink Bulk Capacitance 257
Figure 7-20 Transition Diagram for Increasing the Current 262
Figure 7-21 Transition Diagram for Increasing the Voltage 264
Figure 7-22 Transition Diagram for Increasing the Voltage and Current 266
Figure 7-23 Transition Diagram for Increasing the Voltage and Decreasing the Current 268
Figure 7-24 Transition Diagram for Decreasing the Voltage and Increasing the Current 270
Figure 7-25 Transition Diagram for Decreasing the Current 272
Figure 7-26 Transition Diagram for Decreasing the Voltage 274
Figure 7-27 Transition Diagram for Decreasing the Voltage and the Current 276
Figure 7-28 Transition Diagram for a Sink Requested Power Role Swap 278
Figure 7-29 Transition Diagram for a Source Requested Power Role Swap 281
Figure 7-30 Transition Diagram for a GotoMin Current Decrease 284
Figure 7-31 Transition Diagram for a Source Initiated Hard Reset 286
Figure 7-32 Transition Diagram for a Sink Initiated Hard Reset 288
Figure 7-33 Transition Diagram for no change in Current or Voltage 290
Figure 7-34 Transition Diagram for Fast Role Swap 292
Figure 7-35 Transition Diagram for Increasing the Programmable Power Supply Voltage 294
Figure 7-36 Transition Diagram for Decreasing the Programmable Power Supply Voltage 296
Figure 7-37 Transition Diagram for Changing the Source PDO or APDO 298
Figure 7-38 Transition Diagram for Increasing the Current in PPS mode 300
Figure 7-39 Transition Diagram for decreasing the Current in PPS mode 302
Figure 7-40 Transition Diagram for no change in Current or Voltage in PPS mode 304
Figure 8-1 Example of daisy chained displays 317
Figure 8-2 Basic Message Exchange (Successful) 320
Figure 8-3 Basic Message flow indicating possible errors 321
Figure 8-4 Basic Message Flow with Bad CRC followed by a Retry 322
Figure 8-5 Successful Fixed, Variable or Battery Power Negotiation 326
Figure 8-6 Successful GotoMin operation 329
Figure 8-7 PPS Keep Alive 331
Figure 8-8 Soft Reset 334
Figure 8-9 DFP Initiated Data Reset where the DFP is the Vconn Source 336
Figure 8-10 DFP Receives Data Reset where the DFP is the Vconn Source 339
Figure 8-11 DFP Initiated Data Reset where the UFP is the Vconn Source 342
Figure 8-12 DFP Receives a Data Reset where the UFP is the Vconn Source 346

Figure 8-13 Source initiated Hard Reset 350
Figure 8-14 Sink Initiated Hard Reset 353
Figure 8-15 Source initiated reset - Sink long reset 356
Figure 8-16 Successful Power Role Swap Sequence Initiated by the Source 360
Figure 8-17 Successful Power Role Swap Sequence Initiated by the Sink 365
Figure 8-18 Successful Fast Role Swap Sequence 370
Figure 8-19 Data Role Swap, UFP operating as Sink initiates 374
Figure 8-20 Data Role Swap, UFP operating as Source initiates 376
Figure 8-21 Data Role Swap, DFP operating as Source initiates 378
Figure 8-22 Data Role Swap, DFP operating as Sink initiates 380
Figure 8-23 Source to Sink VCONN Source Swap 382
Figure 8-24 Sink to Source VCONN Source Swap 385
Figure 8-25 Source Alert to Sink 388
Figure 8-26 Sink Alert to Source 390
Figure 8-27 Sink Gets Source Status 391
Figure 8-28 Source Gets Sink Status 393
Figure 8-29 Sink Gets Source PPS Status 395
Figure 8-30 Sink Gets Source's Capabilities 397
Figure 8-31 Dual-Role Source Gets Dual-Role Sink's Capabilities as a Source 399
Figure 8-32 Source Gets Sink's Capabilities 401
Figure 8-33 Dual-Role Sink Gets Dual-Role Source's Capabilities as a Sink 403
Figure 8-34 Sink Gets Source's Extended Capabilities 405
Figure 8-35 Dual-Role Source Gets Dual-Role Sink's Extended Capabilities 407
Figure 8-36 Sink Gets Source's Battery Capabilities 409
Figure 8-37 Source Gets Sink's Battery Capabilities 411
Figure 8-38 Sink Gets Source's Battery Status 413
Figure 8-39 Source Gets Sink's Battery Status 415
Figure 8-40 Source Gets Sink's Port Manufacturer Information 417
Figure 8-41 Sink Gets Source's Port Manufacturer Information 419
Figure 8-42 Source Gets Sink's Battery Manufacturer Information 421
Figure 8-43 Sink Gets Source's Battery Manufacturer Information 423
Figure 8-44 VCONN Source Gets Cable Plug's Manufacturer Information 425
Figure 8-45 Source Gets Sink's Country Codes 427
Figure 8-46 Sink Gets Source's Country Codes 429
Figure 8-47 VCONN Source Gets Cable Plug's Country Codes 431
Figure 8-48 Source Gets Sink's Country Information 433
Figure 8-49 Sink Gets Source's Country Information 435
Figure 8-50 VCONN Source Gets Cable Plug's Country Information 437

Figure 8-51 Source requests security exchange with Sink 439
Figure 8-52 Sink requests security exchange with Source 441
Figure 8-53 Vconn Source requests security exchange with Cable Plug 443
Figure 8-54 Source requests firmware update exchange with Sink 445
Figure 8-55 Sink requests firmware update exchange with Source 447
Figure 8-56 Vconn Source requests firmware update exchange with Cable Plug 449
Figure 8-57 DFP to UFP Discover Identity 451
Figure 8-58 Source Port to Cable Plug Discover Identity 453
Figure 8-59 DFP to Cable Plug Discover Identity 455
Figure 8-60 DFP to UFP Enter Mode 457
Figure 8-61 DFP to UFP Exit Mode 459
Figure 8-62 DFP to Cable Plug Enter Mode 461
Figure 8-63 DFP to Cable Plug Exit Mode 464
Figure 8-64 UFP to DFP Attention 466
Figure 8-65 BIST Carrier Mode Test 467
Figure 8-66 BIST Test Data Test 469
Figure 8-67 UFP Entering USB4 Mode (Valid) 472
Figure 8-68 Cable Plug Entering USB4 Mode (Valid) 474
Figure 8-69 UFP Entering USB4 Mode (Invalid) 476
Figure 8-70 Cable Plug Entering USB4 Mode (Invalid) 478
Figure 8-71 Outline of States 479
Figure 8-72 References to states 480
Figure 8-73 Example of state reference with conditions 480
Figure 8-74 Example of state reference with the same entry and exit 480
Figure 8-75 Source Port Policy Engine State Diagram 481
Figure 8-76 Sink Port State Diagram 487
Figure 8-77 Source Port Soft Reset and Protocol Error State Diagram 492
Figure 8-78 Sink Port Soft Reset and Protocol Error Diagram 493
Figure 8-79 DFP Data_Reset Message State Diagram 495
Figure 8-80 UFP Data_Reset Message State Diagram 497
Figure 8-81 Source Port Not Supported Message State Diagram 499
Figure 8-82 Sink Port Not Supported Message State Diagram 500
Figure 8-83 Source Port Ping State Diagram 501
Figure 8-84 Source Port Source Alert State Diagram 501
Figure 8-85 Sink Port Source Alert State Diagram 502
Figure 8-86 Sink Port Sink Alert State Diagram 502
Figure 8-87 Source Port Sink Alert State Diagram 502
Figure 8-88 Sink Port Get Source Capabilities Extended State Diagram 503

Figure 8-89 Source Give Source Capabilities Extended State Diagram 503
Figure 8-90 Sink Port Get Source Status State Diagram 504
Figure 8-91 Source Give Source Status State Diagram 504
Figure 8-92 Source Port Get Sink Status State Diagram 505
Figure 8-93 Sink Give Sink Status State Diagram 505
Figure 8-94 Sink Port Get Source PPS Status State Diagram 506
Figure 8-95 Source Give Source PPS Status State Diagram 506
Figure 8-96 Get Battery Capabilities State Diagram 507
Figure 8-97 Give Battery Capabilities State Diagram 507
Figure 8-98 Get Battery Status State Diagram 508
Figure 8-99 Give Battery Status State Diagram 508
Figure 8-100 Get Manufacturer Information State Diagram 509
Figure 8-101 Give Manufacturer Information State Diagram 509
Figure 8-102 Get Country Codes State Diagram 510
Figure 8-103 Give Country Codes State Diagram 511
Figure 8-104 Get Country Information State Diagram 511
Figure 8-105 Give Country Information State Diagram 512
Figure 8-106 DFP Enter_USB Message State Diagram 512
Figure 8-107 UFP Enter_USB Message State Diagram 513
Figure 8-108 Send security request State Diagram 513
Figure 8-109 Send security response State Diagram 514
Figure 8-110 Security response received State Diagram 514
Figure 8-111 Send firmware update request State Diagram 515
Figure 8-112 Send firmware update response State Diagram 515
Figure 8-113 Firmware update response received State Diagram 516
Figure 8-114: DFP to UFP Data Role Swap State Diagram 517
Figure 8-115: UFP to DFP Data Role Swap State Diagram 519
Figure 8-116: Dual-Role Port in Source to Sink Power Role Swap State Diagram 521
Figure 8-117: Dual-role Port in Sink to Source Power Role Swap State Diagram 524
Figure 8-118: Dual-Role Port in Source to Sink Fast Role Swap State Diagram 527
Figure 8-119: Dual-role Port in Sink to Source Fast Role Swap State Diagram 530
Figure 8-120 Dual-Role (Source) Get Source Capabilities diagram 532
Figure 8-121 Dual-Role (Source) Give Sink Capabilities diagram 533
Figure 8-122 Dual-Role (Sink) Get Sink Capabilities State Diagram 533
Figure 8-123 Dual-Role (Sink) Give Source Capabilities State Diagram 534
Figure 8-124 Dual-Role (Source) Get Source Capabilities Extended State Diagram 534
Figure 8-125 Dual-Role (Source) Give Sink Capabilities diagram 535
Figure 8-126 VCONN Swap State Diagram 536

Figure 8-127 Initiator to Port VDM Discover Identity State Diagram....539
Figure 8-128 Initiator VDM Discover SVIDs State Diagram....540
Figure 8-129 Initiator VDM Discover Modes State Diagram....541
Figure 8-130 Initiator VDM Attention State Diagram....542
Figure 8-131 Responder Structured VDM Discover Identity State Diagram....543
Figure 8-132 Responder Structured VDM Discover SVIDs State Diagram....544
Figure 8-133 Responder Structured VDM Discover Modes State Diagram....545
Figure 8-134 Receiving a Structured VDM Attention State Diagram....546
Figure 8-135 DFP VDM Mode Entry State Diagram....546
Figure 8-136 DFP VDM Mode Exit State Diagram....548
Figure 8-137 UFP Structured VDM Enter Mode State Diagram....549
Figure 8-138 UFP Structured VDM Exit Mode State Diagram....550
Figure 8-139 Cable Ready VDM State Diagram....551
Figure 8-140 Cable Plug Soft Reset State Diagram....551
Figure 8-141 Cable Plug Hard Reset State Diagram....552
Figure 8-142 VCONN Source Soft Reset or Cable Reset of a Cable Plug or VPD State Diagram....553
Figure 8-143 Source Startup Structured VDM Discover Identity State Diagram....554
Figure 8-144 Cable Plug Structured VDM Enter Mode State Diagram....556
Figure 8-145 Cable Plug Structured VDM Exit Mode State Diagram....557
Figure 8-146 BIST Carrier Mode State Diagram....558
Figure 9-1 Example PD Topology....568
Figure 9-2 Mapping of PD Topology to USB....569
Figure 9-3 USB Attached to USB Powered State Transition....570
Figure 9-4 Any USB State to USB Attached State Transition (When operating as a Consumer)....571
Figure 9-5 Any USB State to USB Attached State Transition (When operating as a Provider)....571
Figure 9-6 Any USB State to USB Attached State Transition (After a USB Type-C Data Role Swap)....572
Figure 9-7 Software stack on a PD aware OS....572
Figure 9-8 Enumeration of a PDUSB Device....573
Figure 10-1 Source Power Rule Illustration....584
Figure 10-2 Source Power Rule Example....584
Figure B-1 External Power supplied downstream....593
Figure B-2 External Power supplied upstream....597
Figure B-3 Giving Back Power....604
Figure D-1 Circuit Block of BMC Finite Difference Receiver....629
Figure D-2 BMC AC and DC noise from VBUS at Power Sink....630
Figure D-3 Sample BMC Signals (a) without *[USB 2.0]* SE0 Noise (b) with *[USB 2.0]* SE0 Noise....630
Figure D-4 Scaled BMC Signal Derivative with 50ns Sampling Rate....631
Figure D-5 BMC Signal and Finite Difference Output with Various Time Steps....631

Figure D-6 Output of Finite Difference in dash line and Edge Detector in solid line....632
Figure D-7 Noise Zone and Detect Zone of BMC Receiver....632
Figure D-8 Circuit Block of BMC Subtraction Receiver....633
Figure D-9 (a) Output of LPF1 and LPF2 (b) Subtraction of LPF1 and LPF2 Output....633
Figure D-10 Output of the BMC LPF1 in blue dash curve and the Subtractor in red solid curve....634
Figure E-1 Example FRS Capable System....635
Figure E-2 Slow V_{BUS} Discharge....636
Figure E-3 Fast V_{BUS} Discharge....637
Figure E-4 Sequence Diagram for slow V_{BUS} discharge (it discharges after FR_Swap message is sent)....640

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 ***[USBBC 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 the USB Type-C® CC wire as the communications channel) than are defined in the ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** or ***[USBBC 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:

- Works seamlessly with legacy USB Devices
- Compatible with existing spec-compliant USB cables
- Minimizes potential damage from non-compliant cables (e.g. 'Y' cables etc.)
- 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.

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.

To facilitate optimum charging, the specification defines two mechanisms a USB Charger can advertise for the Device to use:

1. A list of fixed voltages each with a maximum current. The Device selects a voltage and current from the list. This is the traditional model used by Devices that use internal electronics to manage the charging of their battery including modifying the voltage and current actually supplied to the battery. The side-effect of this model is that the charging circuitry generates heat that may be problematic for small form factor devices.
2. A list of programmable voltage ranges each with a maximum current (PPS). The Device requests a voltage (in 20 mV increments) that is within the advertised range and a maximum current. The USB Charger delivers the requested voltage until the maximum current is reached at which time the USB charger reduces its output voltage so as not to supply more than the requested maximum current. During the high current portion of the charge cycle, the USB Charger can be directly connected (through an appropriate safety device) to the battery. This model is used by Devices that want to minimize the thermal impact of their internal charging circuitry.

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 3.2]*** in band requests for high power interfaces.
- ***[USBBC 1.2]*** mechanisms for supplying higher power (not mandated by this specification).
- ***[USB Type-C 2.0]*** mechanisms for supplying higher power

Initial operating conditions remain the USB Default Operation as defined in ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** or ***[USBBC 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 3.2]***, ***[USB Type-C 2.0]*** and ***[USBBC 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 3.2]***, ***[USB Type-C 2.0]*** and ***[USBBC 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 implementation.

1.4 Conventions

1.4.1 Precedence

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

1.4.2 Keywords

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

1.4.2.1 Conditional Normative

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

1.4.2.2 Deprecated

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

1.4.2.3 Discarded

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.

1.4.2.7 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.

1.4.2.8 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.

1.4.2.9 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.

1.4.2.10 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.

1.4.2.11 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.

1.4.2.12 Shall Not

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

1.4.2.13 Should

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

1.4.2.14 Should Not

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

1.4.2.15 Valid

Valid is a keyword that is the inverse of ***Invalid*** indicating either a Packet or 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 3.2]*** – Universal Serial Bus 3.2 Specification, Revision 1.0, September 22, 2017. www.usb.org/developers/docs.
- ***[USBTypeCAuthentication 1.0]***, Universal Serial Bus Type-C Authentication Specification, Revision 1.0, March 25, 2016. www.usb.org/developers/docs.
- ***[USBPDFirmwareUpdate 1.0]***, Universal Serial Bus Power Delivery Firmware Update Specification, Revision 1.0, September 15, 2016. http://www.usb.org/developers/powerdelivery/
- ***[USBBC 1.2]*** – Universal Serial Bus 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.
- ***[USBBridge 1.1]*** – Universal Serial Bus Type-C Bridge Specification, Revision 1.1, October 10, 2017. www.usb.org/developers/docs.
- ***[USBTypeCBridge 1.0]*** – Universal Serial Bus Type-C Bridge Specification, Revision 1.0, March 25, 2016. www.usb.org/developers/docs.
- ***[USBPD 2.0]*** – Universal Serial Bus Power Delivery Specification, Revision 2, Version 1.2, March 25, 2016. www.usb.org/developers/docs.

- ***[USBPDCompliance]*** – USB Power Delivery Compliance Plan Revision 1.02, Version 2.0, 8 March 2017 http://www.usb.org/developers/docs/devclass_docs/.
- ***[USB Type-C 2.0]*** – Universal Serial Bus Type-C Cable and Connector Specification, Revision 2.0, TBD August 2019. 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 60950-1]*** IEC 60950-1:2005 Information technology equipment – Safety – Part 1: General requirements: Amendment 1:2009, Amendment 2:2013
- ***[IEC 62368-1]*** IEC 62368-1 Audio/Video, information and communication technology equipment – Part 1: Safety requirements
- ***[IEC 63002]*** Draft CD for IEC 63002 Identification and Communication Interoperability Method for External DC Power Supplies Used with Portable Computing Devices.
- ***[ISO 3166]*** ISO 3166 international Standard for country codes and codes for their subdivisions. http://www.iso.org/iso/home/standards/country_codes.htm.
- ***[USB4]*** – Universal Serial Bus 4 Specification (USB4™), Version 1.0, August 2019. www.usb.org/developers/docs.
- ***[DPTC1.0]*** DisplayPort™ Alt Mode on USB Type-C® Standard, Version 1.0b, 03 November 2017. www.vesa.org.
- ***[TBT3]*** see ***[USB4]*** Chapter 13 for Thunderbolt™ 3 device operation.

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 3.2]***, ***[USB Type-C 2.0]*** and ***[USBBC 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 2.0]***).
Active Mode	A Mode which has been entered and not exited.
Alternate Mode	As defined in ***[USB Type-C 2.0]***. Equivalent to Mode in the PD Specification.
Alternate Mode Adapter (AMA)	A PDUSB Device which supports Alternate Modes as defined in ***[USB Type-C 2.0]***. Note that since an AMA is a PDUSB Device it has a single UFP that is only addressable by SOP Packets.
Alternate Mode Controller (AMC)	A DFP that supports connection to AMAs as defined in ***[USB Type-C 2.0]***. A DFP that is an AMC can also be a PDUSB Host.
Augmented Power Data Object (APDO)	Data Object used to expose a Source Port's power capabilities or a Sink's power requirements as part of a ***Source_Capabilities*** or ***Sink_Capabilities*** Message respectively. Programmable Power Supply Data Object is defined.
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: ***PE_SRC_Ready***, ***PE_SNK_Ready*** or ***PE_CBL_Ready***. 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 Slot	A physical location where a Hot Swappable Battery can be installed. A Battery Slot might or might not have a Hot Swappable Battery present in a Battery Slot at any given time.
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 6.4.1.2.4)

Term	Description
Binary Frequency Shift Keying (BFSK)	A Signaling Scheme now ***Deprecated*** in this specification. BFSK used a pair of discrete frequencies to transmit binary (0s and 1s) information over V_{BUS}. See ***[USBPD 2.0]*** for further details.
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 ***BIST*** Messages.
BIST Mode	A BIST receiver or transmitter test mode enabled by a ***BIST*** 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 might reside anywhere in the cable.
Cable Reset	This is initiated by ***Cable Reset*** Signaling from the 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.
Charge Through	A mechanism for a V_{CONN}-powered USB Device (VPD) to pass power and CC communication from one Port to the other without any interference or re-regulation.
Charge Through Port	The USB Type-C receptacle on a USB Device that is designed to allow a Source to be connected through the USB Device to charge a system it is Attached to. Most common use is to allow a single Port Host to support a USB device while being charged.
Chunk	A ***MaxExtendedMsgChunkLen*** (26 byte) or less portion of a Data Block. Data Blocks can be sent either as a single Message or as a series of Chunks.
Chunking	The process of breaking up a Data Block larger than ***MaxExtendedMsgLegacyLen*** (26-bytes) into two of more Chunks.
Cold Socket	A Port that does not apply ***vSafe5V*** on V_{BUS} until a Sink is 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 2.0]***).
Connected	USB Power Delivery ports that have exchanged a Message and a ***GoodCRC*** Message response using the USB Power Delivery protocol so that both Port Partners know 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 USB Type-C Port with Rd asserted on its CC Wire.
Consumer/Provider	A Consumer with the additional capability to act as a Provider. This corresponds to a Dual-Role Port with Rd 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.
Constant Voltage (CV)	A mode in which the Source output Voltage remains constant as the load changes.
Contract	An agreement on both power level and direction reached between a Port Pair. A Contract could be explicitly negotiated between the Port Pair or could 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 ***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.
Current Limit (CL)	A current limiting feature for a Source. When the Sink attempts to draw more current from the Source than the requested Current Limit value, the Source reduces its output voltage so the current it supplies remains at or below the requested value.

Term	Description
Data Block	An Extended Message payload data unit. The size of each type of Data Block is specified as a series of bytes up to ***MaxExtendedMsgLen*** bytes in length. This is distinct from a Data Object used by a Data Message which is always a 32-bit object.
Data Message	A Data Message consists of a Message Header followed by one or more Data Objects. Data Messages are easily identifiable because the ***Number of Data Objects*** field in the Message Header is a non-zero value.
Data Object	A Data Message payload data unit. This 32-bit object 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 ***[USB Type-C 2.0]*** 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 (DPM)	Module running in a Source or Sink 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 ***[USB Type-C 2.0]***. At connection the Port defaults to operation as a USB Host (when USB Communication is supported) 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 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 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 ***[USB Type-C 2.0]***.
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 ***Accept*** Message in response to a ***Request*** Message sent by a Sink followed by a ***PS_RDY*** Message indicating that the power supply is ready; this corresponds to the ***PE_SRC_Ready*** state for a Source Policy Engine and the ***PE_SNK_Ready*** state for a Sink Policy Engine. The Explicit Contract can be altered through the re-negotiation process. All Port pairs are required to make an Explicit Contract.
Extended Message (EM)	A Message containing Data Blocks. The Extended Message is defined by the ***Extended*** field in the Message Header being set to one and contains an Extended Message Header immediately following the Message Header.
Extended Message Header	Every Extended Message contains a 16-bit Extended Message Header immediately following the Message Header containing information about the Data Block and any Chunking being applied.
Fast Role Swap	Process of exchanging the Source and Sink roles between Port Partners rapidly due to the disconnection of an external power supply.

Term	Description
Fast Role Swap Request	An indication from an initial Source to the initial Sink that a Fast Role Swap is needed. The Fast Role Swap Request is indicated by driving the CC line to Ground; it is not a Message or a Signal.
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 6.4.1.2.2)
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 ***Hard Reset*** 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. 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.
Hot Swappable Battery	A Battery that is easily accessible for a user to remove or change for another Battery.
ID Header VDO	The VDO in a ***Discover Identity*** 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 a Power Role Swap or Fast Role Swap. Implicit Contracts are transitory since the Port pair is required to immediately negotiate an Explicit Contract after the Power Role Swap. An Implicit Contract ***Shall*** be limited to USB Type-C Current (see ***[USB Type-C 2.0]***).
Initiator	The initial sender of a Command request in the form of a query.
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 ***GoodCRC*** 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 might be the default policy, hard coded or modified by changes in operating parameters or one provided by the system Host or some combination of these. The Local Policy ***Optionally*** can be changed by a System Policy Manager.
LPS	Limited Power Supply as defined in ***[IEC 62368-1]***.
Message	The packet payload consisting of a Message Header for Control Messages and a Message Header and data for Data Messages and Extended Messages as defined in Section 6.
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 Chapter 6.
Modal Operation	State where there are one or more Active Modes. Modal Operation ends when there are no longer any Active Modes.

Term	Description
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 2.0]***.
Multi-Drop	Refers to a Power Delivery system with one or more Cable Plugs where communication is to the Cable Plugs rather than the Port Partner. Multi-Drop systems share the Power Delivery communication channel with the Port Partners.
Negotiation	This is the PD process whereby: 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. The result of the negotiation is a Contract for power delivery/consumption between the Port Pair.
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 ***GoodCRC*** Message has been received).
OCP	Over-Current Protection
OTP	Over-Temperature Protection
OVP	Over-Voltage Protection
Packet	One entire unit of PD communication including a Preamble, ***SOP****, payload, CRC and ***EOP*** 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 ***Discover Identity*** to determine its characteristics (Electronically Marked Cable see ***[USB Type-C 2.0]***). Note this specification does not discuss Passive Cables which are not Electronically Marked Cables.
PD	USB Power Delivery
PD Capable	A Port that supports USB Power Delivery.
PD Connection	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.
PDP Rating	Manufacturer declared PDP for a Source.
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 the USB Type-C CC wire 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.

Term	Description
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 ***Source_Capabilities*** or ***Sink_Capabilities*** 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.
Product Type	Product categorization returned as part of the ***Discover Identity*** Command.
Product Type VDO	VDO identifying a certain Product Type in the ID Header VDO of a ***Discover Identity*** Command.
Programmable Power Supply (PPS)	A power supply whose output voltage can be programmatically adjusted in small increments over its advertised range. The PPS also has a programmable output current fold back. The capabilities of the PPS are exposed by the Programmable Power Supply APDO (see Section 6.4.1.2.5).
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.
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 USB Type-C Port with Rp asserted on its CC Wire.
Provider/Consumer	A Provider with the additional capability to act as a Consumer. This corresponds to a Dual-Role Power Port with Rp asserted on its CC Wire.
PS1, PS2	Classification of electrical power as defined in ***[IEC 62368-1].***
PSD	Sink which draws power but has no other USB or Alternate Mode communication function e.g. a power bank.
Rd	Pull-down resistor on the USB Type-C CC wire used to indicate that the Port is a Sink (see ***[USB Type-C 2.0]***).
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 ***Request*** 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 2.0]***).
Safe Operation	Sources must have the ability to tolerate ***vSafe5V*** 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. ***Hard Reset*** as defined in Section 5.4.

Term	Description
Signaling Scheme	Physical mechanism used to transmit bits. Only the BMC Signaling Scheme is defined in this specification. Note: the BFSK Signaling Scheme supported in previous Revisions of this specification has been ***Deprecated***.
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.
Sink Directed Charge	A charging scheme whereby the Sink connects the Source to its battery through safety and other circuitry. When the Current Limit feature is activated, the Source automatically controls its output current by adjusting its output voltage.
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 ***SOP***.
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 ***SOP***, ***SOP'*** or ***SOP''***.
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 ***SOP'*** 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 ***SOP''*** 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 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".
Start of Packet (SOP)	K-code marker used to delineate the start of a packet. Three start of packet sequences are defined: ***SOP, SOP'*** and ***SOP''***, with ***SOP**** used to refer to all three in place of ***SOP***/***SOP'***/***SOP''***.
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, ***SOP****, 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 ReservedMessage 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 2.0]***. At connection the Port defaults to operation as a USB Device (when USB Communication is supported) and Sink.

Term	Description
USB Attached State	Synonymous with the ***[USB 2.0]*** and ***[USB 3.2]*** definition of the Attached state
USB Default Operation	Operation of a Port at Attach or after a Hard Reset where the DFP Source applies ***vSafe0V*** or ***vSafe5V*** on V_{BUS} and the UFP Sink is operating at ***vSafe5V*** as defined in ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** or ***[USBBC 1.2]***.
USB Device	Either a hub or a peripheral device as defined in ***[USB 2.0]*** and ***[USB 3.2]***.
USB Host	The host computer system where the USB host controller is installed as defined in ***[USB 2.0]*** and ***[USB 3.2]***.
USB Powered State	Synonymous with the ***[USB 2.0]*** and ***[USB 3.2]*** definition of the powered state.
USB Safe State	State of the USB Type-C connector when there are pins to be re-purposed (see ***[USB Type-C 2.0]***) 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 USB Micro-A plugs and USB Standard-A plugs and receptacles. USB Micro-AB receptacles are assumed to be a combination of USB Type-A and USB Type-B.
USB Type-B	Terms used to refer to any B-plug or receptacle including USB Micro-B plugs and USB Standard-B plugs and receptacles, including the PD and non-PD versions. USB 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 2.0]***.
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 6.4.1.2.3).
VCONN Powered Accessory	An accessory that is powered from VCONN to operate in a Mode (see ***[USB Type-C 2.0]***).
VCONN Powered USB Device (VPD)	A captive cable USB Device that may be powered by either VCONN or V_{BUS} as defined in ***[USB Type-C 2.0]***. A VPD is a captive cable USB device that may be powered by either VCONN or V_{BUS} and only responds to SOP' messages as defined in the Tables in Section 6.12 (Message Applicability). It only responds to messages sent with a Specification Revision of at least Revision 3.0. A VPD is not allowed to support Alternate Modes. The term VPD refers to either a VPD or a CT-VPD with no charger connected.
VCONN Powered USB Charge Through Device (CT-VPD)	A CT-VPD is a VPD with an additional port for connecting a Source (e.g., a charger) as defined in ***[USB Type-C 2.0]***. When no charger is connected, a CT-VPD behaves as a VPD. When a charger is connected, no PD communication to the CT-VPD itself is possible as CC is connected to the charger port. Hence all PD communication then is with the charger and the cable with which it is connected.
VCONN Source	The USB Type-C Port responsible for sourcing VCONN.
VCONN 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 ***Vendor_Defined*** Message.
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]*.

1.8 Changes from Revision 2.0

This specification includes the following updates:

- PD Power rules (also applied to *[USBPD 2.0]*).
- Mechanisms to avoid collisions and simplify communication.
- Support for *[IEC 63002]* included extended power supply capabilities and status.
- Battery capabilities and status.
- Ability to perform a fast power role swap.
- Support for *[USBTypeCAuthentication 1.0]* and *[USBPDFirmwareUpdate 1.0]*.
- Programmable Power Supply (PPS) to support Sink Directed Charging

The following have been ***Deprecated*** from this specification:

- The BFSK signaling scheme.
- Definitions of Standard/Micro A/B cables and connectors.
- Dead battery operation for A/B Ports.
- Profiles (replaced by PD Power Rules).

The following have been moved to other specifications:

- System Policy which is now defined in *[USBTypeCBridge 1.0]*.

For more details see Section 9.

1.9 Compatibility with Revision 2.0

Revision 3.0 of the USB Power Delivery specification is designed to be fully interoperable with *[USBPD 2.0]* systems using BMC signaling over the *[USB Type-C 2.0]* connector and to be compatible with Revision 2.0 hardware.

Please see Section 2.3 for more details of the mechanisms defined to enable compatibility.

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 the USB Type-C® connector's CC wire as the communications channel. The mechanisms used, operate independently of other USB methods used to negotiate power.

USB Power Delivery also acts as a side-band channel enabling communications with the cable assembly connecting the ports. 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 3.2]***, ***[USB Type-C 2.0]*** or ***[USBBC 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 or Fast Role Swap when the initial Source removes power in order 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 or Fast Role Swap). Implicit Contracts are temporary; Port Pairs are required to immediately negotiate an Explicit Contract.

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 3.2]***, ***[USB Type-C 2.0]*** or ***[USBBC 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 or Fast Role Swap to exchange the power supply roles such that the DFP receives power and the UFP supplies power, to perform a Data Role Swap such that the DFP becomes the UFP and vice-versa and to perform a VCONN Swap to change the end supplying VCONN to the cable.

Prior to an Explicit Contract only the Source Port, that is also the VCONN Source, can communicate with the Attached cable assembly. This is important for ***[USB Type-C 2.0]*** where 5A cabling is marked as well as other details of the cable assembly such as the supported speed.

Cable discovery, determining whether the cable can communicate, can occur on initial Attachment of a Port Pair, before an Explicit Contract has been established. It is also possible to carry out cable discovery after a Power Role Swap or Fast Role Swap prior to re-establishing an Explicit Contract, where the UFP is the Source and an Implicit Contract is in place. Cable discovery can be carried out after an Explicit Contract has been established, if the Cable has not yet been discovered.

Once an Explicit Contract is in place either the Source or Sink Port, provided it is also the VCONN Source, is permitted to communicate with the Attached cable assembly. This communication can consist of:

- cable discovery (when the cable has not already been discovered)
- discovering capabilities
- discovery of SVIDs
- discovery of Modes
- entering Modes supported by the cable assembly

- exiting 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 ***Deprecated***. See ***[USBPD 2.0]*** for legacy PD connector specification.
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	Example CRC calculations.
Appendix B	Scenarios illustrating Device Policy Manager operation.
Appendix C	Examples of Structured VDM usage.

2.3 Compatibility with Revision 2.0

Revision 3.0 of the USB Power Delivery specification is designed to be fully interoperable with ***[USBPD 2.0]*** systems using BMC signaling over the ***[USB Type-C 2.0]*** connector and to be compatible with Revision 2.0 hardware.

This specification mandates that all Revision 3.0 systems fully support Revision 2.0 operation. They must discover the supported Revision used by their Port Partner and any connected Cable Plugs and revert to operation using the lowest common Revision number (see Section 6.2.1.1.5).

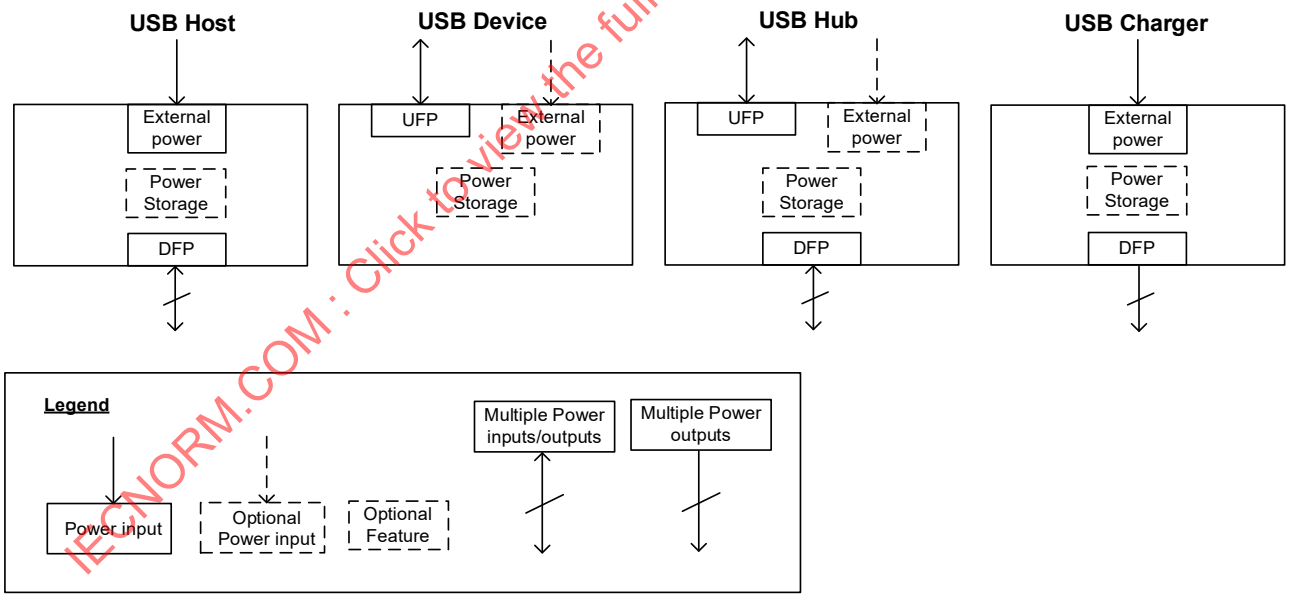
This specification defines Extended Messages containing data of up to 260 bytes (see Section 6.2.1.2). These Messages will be larger than expected by existing PHY HW. To accommodate Revision 2.0 based systems a Chunking mechanism is mandated such that Messages are limited to Revision 2.0 sizes unless it is discovered that both systems support the longer Message lengths.

This specification includes changes to the Vendor Defined Objects (VDO) used in the discovery of passive/active marked cables and Alternate Mode Adapters (AMA) (see Section 6.4.4.2). To enable systems to determine which VDO format is being used the Structured Vendor Defined Message (SVDM) version number has been incremented to 2.0. Version numbers have also been incorporated into the VDOs themselves to facilitate future changes if these become necessary.

2.4 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).
  - Is the only Port allowed to talk to the Cable Plug(s) at any given time.

2.5 SOP* Communication

2.5.1 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 might or might 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.

2.5.2 SOP* Collision Avoidance

For all SOP* the Source co-ordinates communication in order to avoid bus collisions by allowing the Sink to initiate messaging when it does not need to communicate itself. Once an Explicit Contract is in place the Source indicates to the Sink that it can initiate a message sequence. This sequence can be communication with the Source or with one of the Cable Plugs. As soon as the Source itself needs to initiate a message sequence this will be indicated to the Sink. The Source then waits for any outstanding Sink SOP* Communication to complete before initiating a message sequence itself.

2.5.3 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.

2.5.4 SOP'/SOP'' Communication with Cable Plugs

SOP' Communication is recognized by electronics in one Cable Plug (see ***[USB Type-C 2.0]***). SOP'' Communication can also be supported when SOP' Communication is also supported. SOP' and SOP'' assignment is fixed and does not dynamically change.

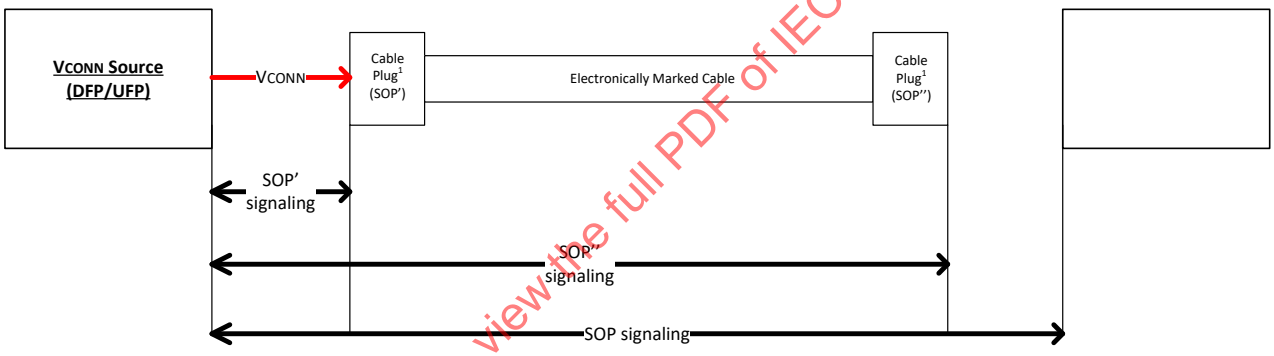
SOP Communication between the Port Partners is not recognized by the Cable Plug. Figure 2-2 outlines the usage of SOP* Communications between a VCONN Source (DFP/UFP) and the Cable Plugs.

All SOP* Communications take place over a single wire (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 or Fast Role Swap) only the Source port that is supplying VCONN is allowed to send packets to a Cable Plug (SOP'/SOP'') and is allowed to respond to packets from the Cable Plug (SOP'/SOP'') with a GoodCRC in order to discover the Cable Plug's 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 VCONN Source (either the DFP or the UFP) can communicate with the Cable Plug(s) using SOP'/SOP'' Packets (see Figure 2-2). During this phase all communication with the Cable Plug is initiated and controlled by the VCONN Source which acts to prevent conflicts between SOP* Packets. The Port that is not the VCONN Source does not communicate with the Cable Plug and does not recognize any SOP'/SOP'' Packets received. Only the DFP, when acting as a VCONN Source, is allowed to send SOP* in order to control the entry and exiting of Modes and to manage Modal Operation.

Figure 2-2 Example SOP' Communication between VCONN Source and Cable Plug(s)



[1] Cable Plug can be physically Attached to either the DFP or UFP.

2.6 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 2.0]***) 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 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 PDUSB Device see Section 9.1.2.

2.6.1 Source Operation

The Source operates differently depending on Attachment status:

- At Attach (no PD Connection or Contract):
  - 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 sets VBUS 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:
    - The Source attempts 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***.
- 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 or Fast Role Swap):
  - The Source receives a ***Request*** Message from the Sink and, if this is a ***Valid*** request, responds with an ***Accept*** Message followed by a ***PS_RDY*** Message when its power supply is ready to source power at the agreed level. At this point an Explicit Contract has been agreed.
  - A DFP that 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.

- The Source when it is the VCONN Source can communicate with a Cable Plug using SOP' or SOP'' Communication at any time it is not engaged in any other SOP Communications:
  - If SOP Packets are received by the 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.
- When the Source Port is also a DFP:
  - The Source can control the entry and exiting of modes in the Cable Plug(s) and control Modal Operation.
  - 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.
- 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.
  - When the Source operating in a PPS mode fails to receive periodic communication (e.g. a ***Request*** Message) from the Sink within ***tPPSTimeout***:
    - Source issues a Hard Reset and takes V_{BUS} to ***vSafe5V***
  - 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, 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 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 2.0]*** , by entering the ***ErrorRecovery*** state.

2.6.2 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 ***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 or Fast 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 may request one of the capabilities offered by the Source, even if this is the ***vSafe5V*** output offered by ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** or ***[USBBC 1.2]***, in order to enable future power negotiation:
      - 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 3.2]***, ***[USB Type-C 2.0]*** or ***[USBBC 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).
  - A Sink operating in a PPS mode periodically sends ***Request*** Message within ***tPPSRequest*** even if its request is unchanged.
  - The Sink will always have Rd asserted on its CC wire.

- When this Port is a DRP the Sink can initiate or receive a request for the exchange of power roles. After the Power Role Swap this Port will be a Source and an Implicit Contract will be in place until an Explicit Contract is negotiated immediately afterwards.
- When this Port is a DRD the Sink 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 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.
- The Sink when it is the VCONN Source can communicate with a Cable Plug using SOP' or SOP'' Communication at any time it is not engaged in any other SOP Communications:
  - If SOP Packets are received by the Sink, during SOP' or SOP'' Communication, the SOP' or SOP'' Communication is immediately terminated (the Cable Plug times out and does not retry)
  - If the Sink 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.
  - When the Sink Port is also a DFP the Sink can control the entry and exiting of modes in the Cable Plug(s) and control Modal Operation.
- 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 either a Hard Reset, 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.

2.6.3 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/UFP 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 VBUS 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).

2.7 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.

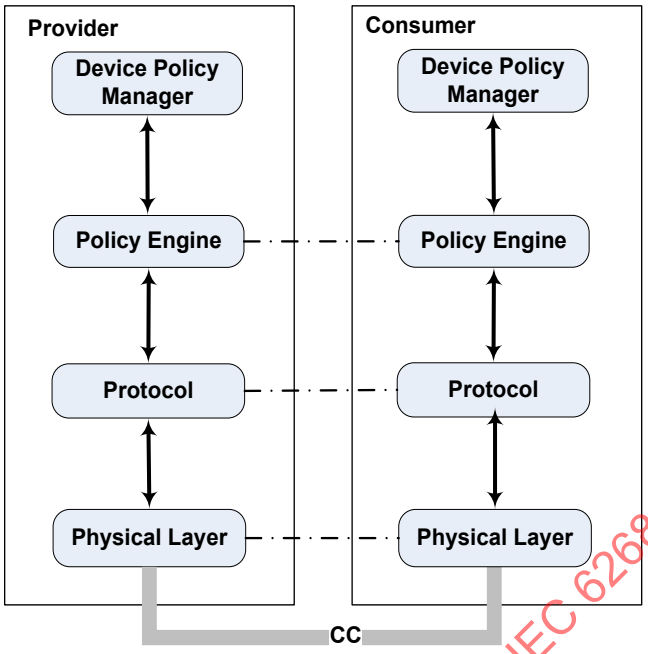
This section outlines the high-level logical architecture of USB Power Delivery referenced throughout this specification. In practice various implementation options are possible based on many different possible types of PD device. PD devices can have many different configurations e.g. USB or non-USB 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 2-3 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 6) 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 2-3 USB Power Delivery Communications Stack



Additionally, USB Power Delivery devices which can operate as USB devices can communicate over USB (see Figure 2-4). 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 2-4 USB Power Delivery Communication Over USB

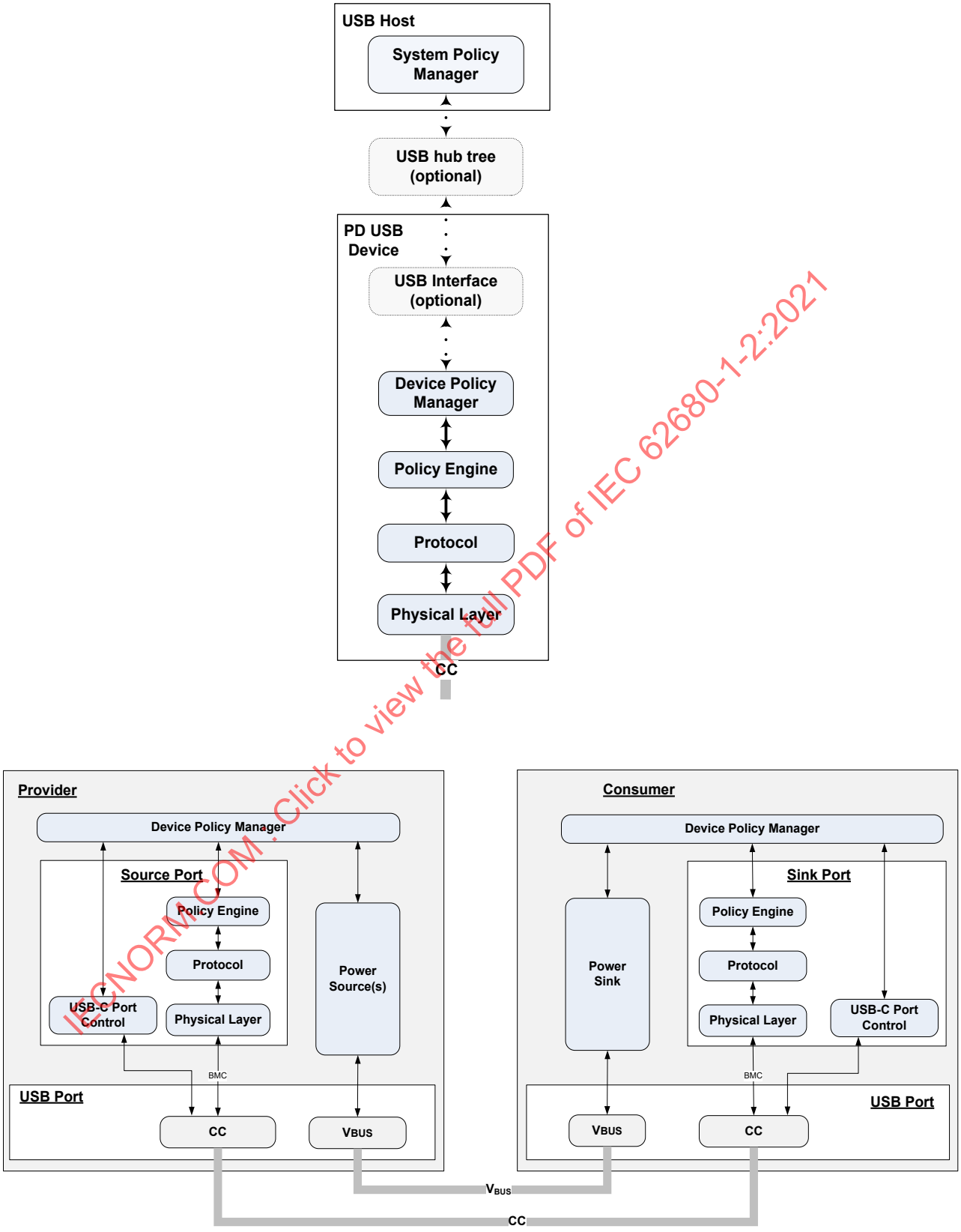
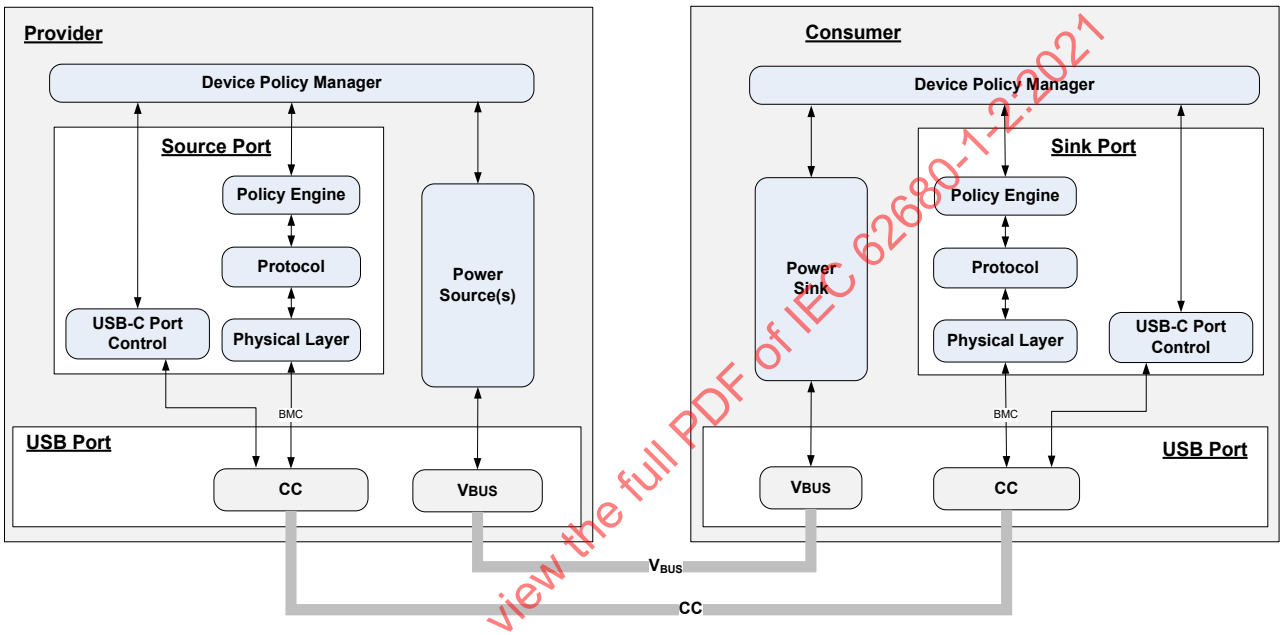


Figure 2-5 shows the logical blocks between two Attached PD ports. In addition to the communication stack described above there are also:

- 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 **USB-C Port Control** module (see Section 4.4) that detects cable Attach/Detach as defined in ***[USB Type-C 2.0]***.
- USB Power Delivery uses standard cabling as defined in ***[USB Type-C 2.0]***.

The **Device Policy Manager** talks to the communication stack, Source/Sink and the USB-C Port Control block in order to manage the resources in the Provider or Consumer..

Figure 2-5 High Level Architecture View



2.7.1 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).

2.7.1.1 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.

2.7.1.2 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 USB-C Port Control for that Port.

2.7.1.3 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).

2.7.2 Message Formation and Transmission

2.7.2.1 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 it 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.

Extended Messages of up to a Data Size of ***MaxExtendedMsgLen*** can be sent and received provided the Protocol Layer determines that both Port Partners support this capability. When one of both Port Partners do not support Extended Messages of Data Size greater than ***MaxExtendedMsgLegacyLen*** then the Protocol Layer supports a Chunking mechanism to break larger Messages into smaller Chunks of size ***MaxExtendedMsgChunkLen***.

2.7.2.2 PHY Layer

The PHY Layer is responsible for sending and receiving Messages across the USB Type-C CC wire and for managing data. It tries to avoid collisions on the wire, recovering from them when they occur. It also detects errors in the Messages using a CRC.

2.7.3 Collision Avoidance

2.7.3.1 Policy Engine

The Policy Engine in a Source will indicate to the Protocol Layer the start and end of each Atomic Message Sequence (AMS) that the Source initiates. The Policy Engine in a Sink will indicate to the Protocol Layer the start of each AMS the Sink initiates. This enables co-ordination of AMS initiation between the Port Partners.

2.7.3.2 Protocol Layer

The Protocol Layer in the Source will request the PHY to set the Rp value to ***SinkTxOk*** to indicate that the Sink can initiate an AMS by sending the first Message in the sequence. The Protocol Layer in the Source will request the PHY to set the Rp value to ***SinkTxNG*** to indicate that the Sink cannot initiate an AMS since the Source is about to initiate an AMS.

The Protocol Layer in the Sink, when the Policy Engine indicates that an AMS is being initiated, will wait for the Rp value to be set to ***SinkTxOk*** before initiating the AMS by sending the first Message in the sequence.

2.7.3.3 PHY Layer

The PHY Layer in the Source will set the Rp value to either ***SinkTxOk*** or ***SinkTxNG*** as directed by the Protocol Layer. The PHY Layer in the Sink will detect the present Rp value and inform the Protocol Layer.

2.7.4 Power supply

2.7.4.1 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.

2.7.4.2 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.

2.7.4.3 Dual-Role Power Ports

Dual-Role Power Ports have the ability to operate as either a Source or a Sink and to swap between the two roles using Power Role Swap or Fast Role Swap.

2.7.4.4 Dead Battery or Lost Power Detection

[USB Type-C 2.0] defines mechanisms intended to communicate with and charge a Sink or DRP with a Dead Battery.

2.7.4.5 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 only the VCONN Source is permitted to communicate with the Cable Plugs. Prior to a Power Role Swap, Data Role Swap or Fast 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.

2.7.5 DFP/UFP

2.7.5.1 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.

2.7.5.2 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.

2.7.5.3 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.

2.7.6 Cable and Connectors

2.7.6.1 USB-C Port Control

The USB-C Port Control block provides mechanisms to inform the Device Policy Manager of cable Attach/Detach events.

The USB Power Delivery specification assumes certified USB cables and associated detection mechanisms as defined in the ***[USB Type-C 2.0]*** specification.

2.7.7 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 3.2]***, ***[USBBC 1.2]*** or ***[USB Type-C 2.0]*** 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 3.2]***, ***[USBBC 1.2]*** or ***[USB Type-C 2.0]*** 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 3.2]***, ***[USBBC 1.2]*** or ***[USB Type-C 2.0]*** specifications.

2.7.8 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

This section has been ***Deprecated***. Please refer to *[USBPD 2.0]* for details of cables and connectors used in scenarios utilizing the BFSK Signaling scheme in conjunction with USB Type-A or USB Type-B connectors.

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 3.2]***, ***[USBBC 1.2]*** and ***[USB Type-C 2.0]*** (USB Type-C®) 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 6.8.3.

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.

Dead Battery charging operation for connections between USB Type-C connectors is defined in ***[USB Type-C 2.0]***.

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 due to excessive current draw. Certified USB cabling is specified such that such errors don't typically occur (See ***[USB Type-C 2.0]***).

4.4 Cable Type Detection

Standard USB Type-C cable assemblies are rated for PD voltages higher than ***vSafe5V*** and current levels of at least 3A (See ***[USB Type-C 2.0]***). The Source ***Shall*** limit maximum capabilities it offers so as not to exceed the capabilities of the type of cabling detected.

Sources capable of offering more than 3A ***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 2.5) 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 Fast 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 currents in excess of 3A and/or voltages in excess of 20V.
- 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.
- Sinks with the DRP bit set, ***Shall*** respond to a ***Get_Source_Cap*** message by declaring their full Source Capabilities, without limiting them based on the cable's capabilities.

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 (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 Biphase Mark Coding (BMC) over CC.

The receiver performs the following functions:

- 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

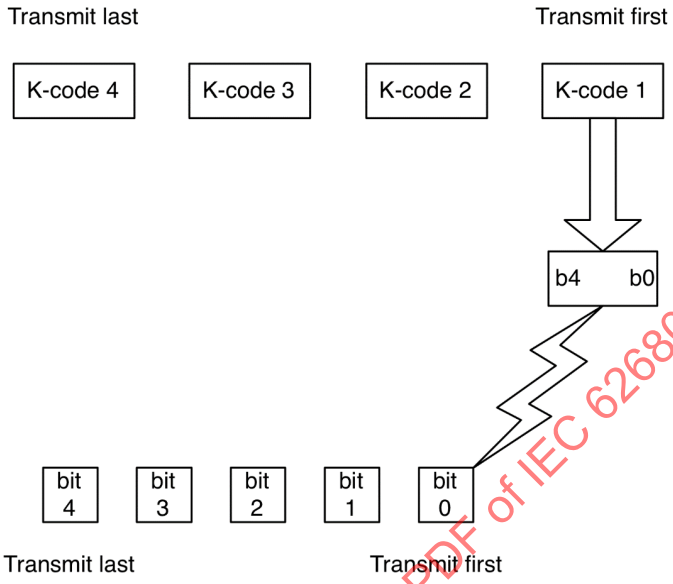
Name	4b	5b Symbol	Description
0	0000	11110	hex data 0
1	0001	01001	hex data 1
2	0010	10100	hex data 2
3	0011	10101	hex data 3
4	0100	01010	hex data 4
5	0101	01011	hex data 5
6	0110	01110	hex data 6
7	0111	01111	hex data 7
8	1000	10010	hex data 8
9	1001	10011	hex data 9
A	1010	10110	hex data A
B	1011	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
Cable Reset	Section 5.6.5
Hard Reset	Section 5.6.4
SOP	Section 5.6.1.2.1
SOP'	Section 5.6.1.2.2
SOP'_Debug	Section 5.6.1.2.4
SOP''	Section 5.6.1.2.3
SOP''_Debug	Section 5.6.1.2.5

The receiver ***Shall*** search for all four K-codes. When the receiver finds all four K-codes in the correct place, it ***Shall*** interpret this as a ***Valid*** ordered set. When the receiver finds three out of four K-codes in the correct place, it ***May*** interpret this as a ***Valid*** ordered set. The receiver ***Should*** ensure that all four K-codes are ***Valid*** to avoid ambiguity in detection (see Table 5-3).

Table 5-3 Validation of Ordered Sets

	1st code	2nd code	3rd code	4th code
Valid[1]	Corrupt	K-code	K-code	K-code
Valid[1]	K-code	Corrupt	K-code	K-code
Valid[1]	K-code	K-code	Corrupt	K-code
Valid[1]	K-code	K-code	K-code	Corrupt
Valid[2] (perfect)	K-code	K-code	K-code	K-code
Invalid (example)	K-code	Corrupt	K-code	Corrupt
1. ***May*** be interpreted as a ***Valid*** ordered set.				

	1st code	2nd code	3rd code	4th code
2. ***Shall*** be interpreted as a ***Valid*** ordered set.				

5.5 Transmitted Bit Ordering

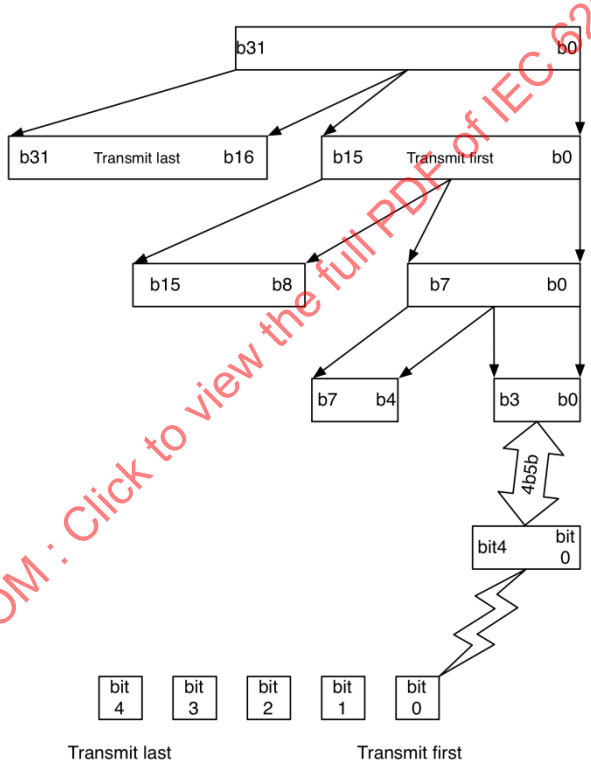
This section describes the order of bits on the wire that ***Shall*** be used when transmitting data of varying sizes. Table 5-4 shows the different data sizes that are possible.

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

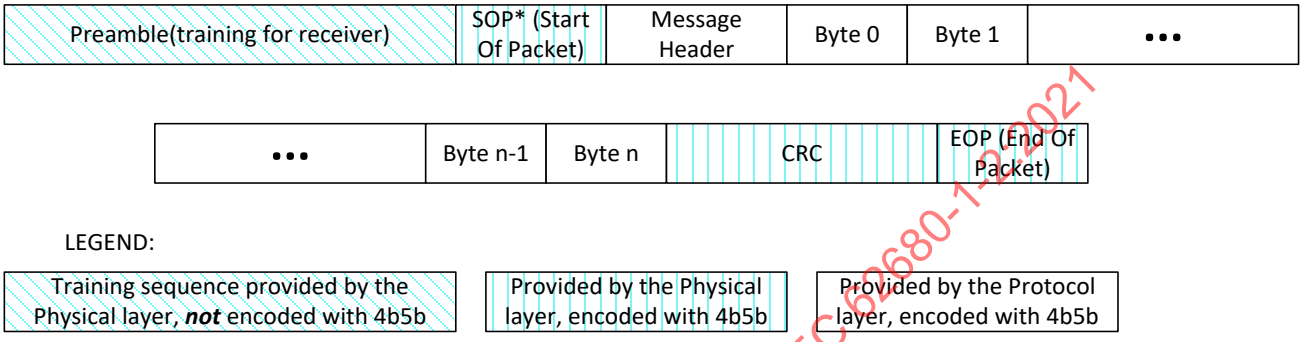
Figure 5-2 Transmit Order for Various Sizes of Data



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 using BMC over CC. Note that all the bits in the Packet, including the Preamble, are BMC encoded. See Section 6.2.1 for more details of the Packet construction for Control, Data and Extended Messages.

Figure 5-3 USB Power Delivery Packet Format



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 Source or Sink ***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 Ports on PDUSB Hosts and PDUSB Devices. Cable Plugs and VPDs ***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 VPD ***Shall*** have SOP' Communication capability. A VPD and a Cable Plug capable of SOP' Communications ***Shall*** only detect and communicate with packets starting with ***SOP'***.

A Port 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.

For a VPD or 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 a Port 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 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 VPD ***Shall Not*** have SOP'' Communication capability. 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 Port 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 Port 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 Port if a ***Valid SOP**** is not detected (see Table 5-3) then the whole transmission ***Shall*** be ***Discarded***.

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

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

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 3.2]***.

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

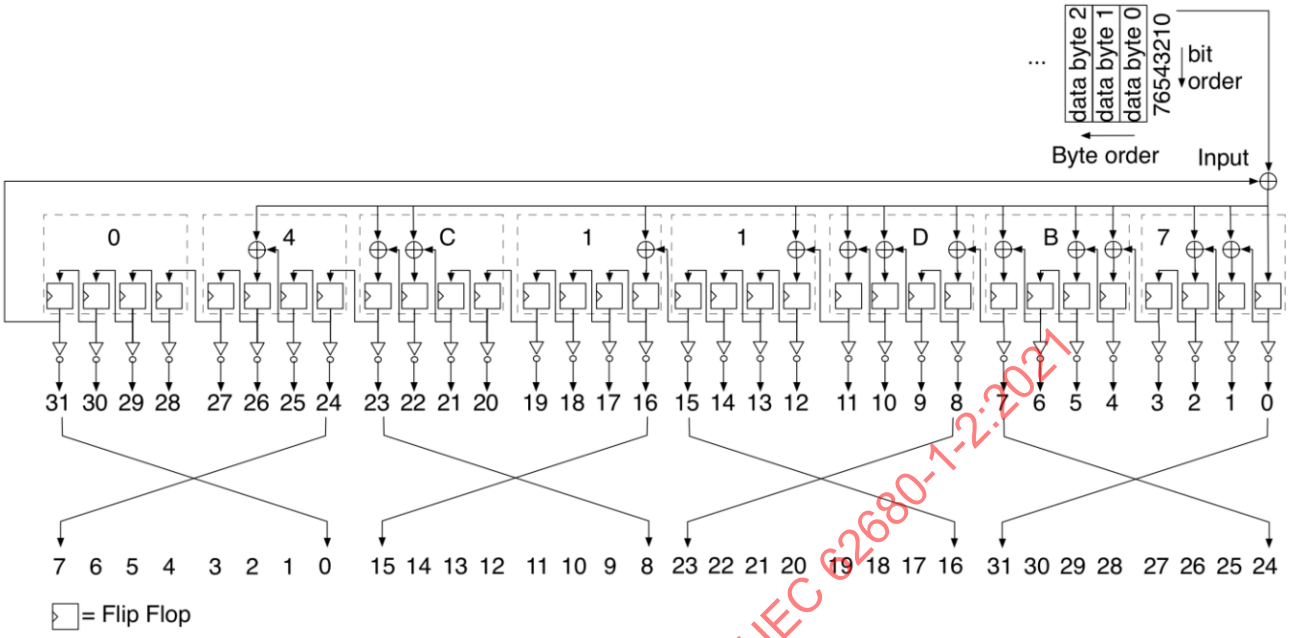


Table 5-10 CRC-32 Mapping

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
25	6
26	5
27	4

CRC-32	Result bit Position in CRC-32 Field
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 the CC-line becomes idle the receiver ***Shall*** stop processing the packet and ***Discard*** it (no ***GoodCRC*** Message is returned). See Section 5.8.6.1 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 6.8.3. If a ***Valid Hard Reset*** is not detected (see Table 5-3) then the whole transmission ***Shall*** be ***Discarded***.

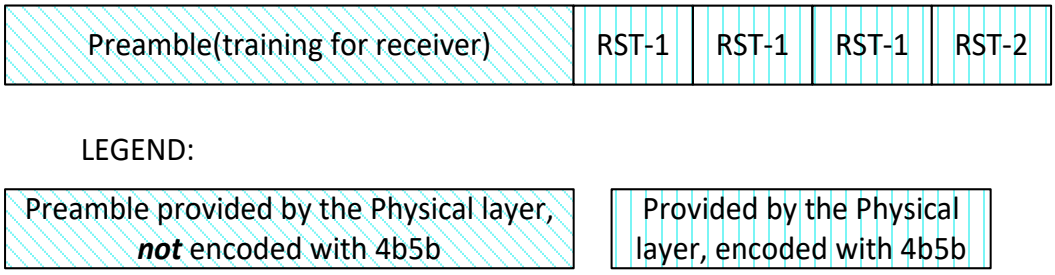
A Cable Plug ***Shall*** perform a Hard Reset when it detects ***Hard Reset*** Signaling being sent between the Port Partners. After receiving the ***Hard Reset*** Signaling, the device ***Shall*** reset as described in Section 6.8.3.

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 CC is not idle, wait for it to become idle (see Section 5.8.6.1).
3. Wait ***tInterFrameGap.***
4. If CC 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



5.6.5 Cable Reset

Cable Reset Signaling is an ordered set of bytes sent with the purpose to be recognized by 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).

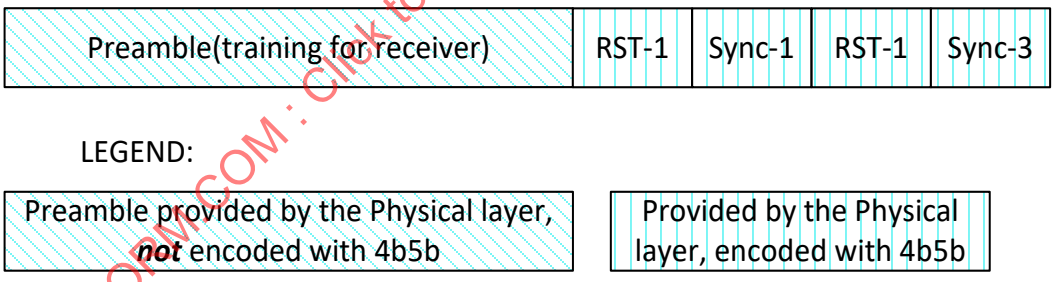
Table 5-12 Cable Reset ordered set

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



5.7 Collision Avoidance

The PHY Layer ***Shall*** monitor the channel for data transmission and only initiate transmissions when 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 5.8.5.4 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 CC is not idle, the PHY Layer ***Shall*** signal to the protocol layer that it has ***Discarded*** the Message as soon as CC becomes idle. See Section 5.8.6.1 for the definition of idle CC.

In addition, the PHY Layer ***Shall*** control the Rp resistor value to avoid collisions between Source and Sink transmissions. The Source ***Shall*** set an Rp value corresponding to a current of 3A to indicate to the Sink that it ***May*** initiate an AMS. The Source ***Shall*** set an Rp value corresponding to a current of 1.5A this ***Shall*** indicate to the Sink that it ***Shall Not*** initiate an AMS and ***Shall*** only respond to Messages as part of an AMS. See ***[USB Type-C 2.0]*** (USB Type-C®) for details of the corresponding Rp values.

Table 5-13 details the Rp values that ***Shall*** be used by the Source to control Sink initiation of an AMS.

Table 5-13 Rp values used for Collision Avoidance

Source Rp	Parameter	Description	Sink operation	Source operation
1.5A@5V	***SinkTxNG***	Sink Transmit "No Go",	Sink cannot initiate an AMS. Sink can only respond to Messages as part of an AMS	Source can initiate an AMS ***tSinkTx*** after setting Rp to this value.
3A@5V	***SinkTxOk***	Sink Transmit "Ok"	Sink can initiate an AMS.	Source cannot initiate an AMS while it has this value set.

See also Section 6.6.16 and Section 6.11.2.1.

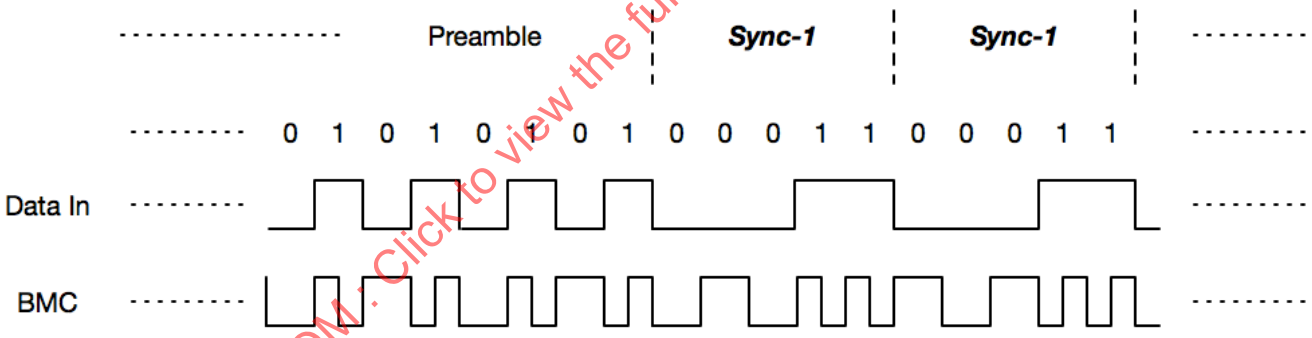
5.8 Biphase Mark Coding (BMC) Signaling Scheme

Biphase Mark Coding (BMC) is the physical layer Signaling Scheme 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 5-7 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 5-7 BMC Example



5.8.1 Encoding and signaling

BMC uses DC coupled baseband signaling on CC. Figure 5-8 shows a block diagram for a Transmitter and Figure 5-9 shows a block diagram for the corresponding Receiver.

Figure 5-8 BMC Transmitter Block Diagram

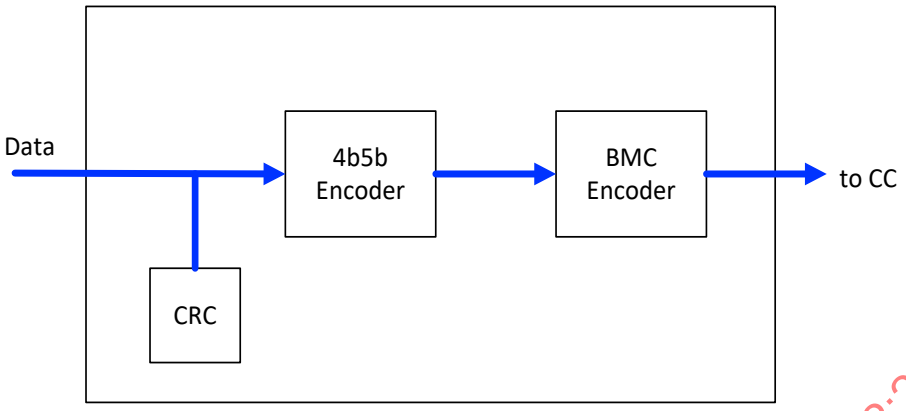
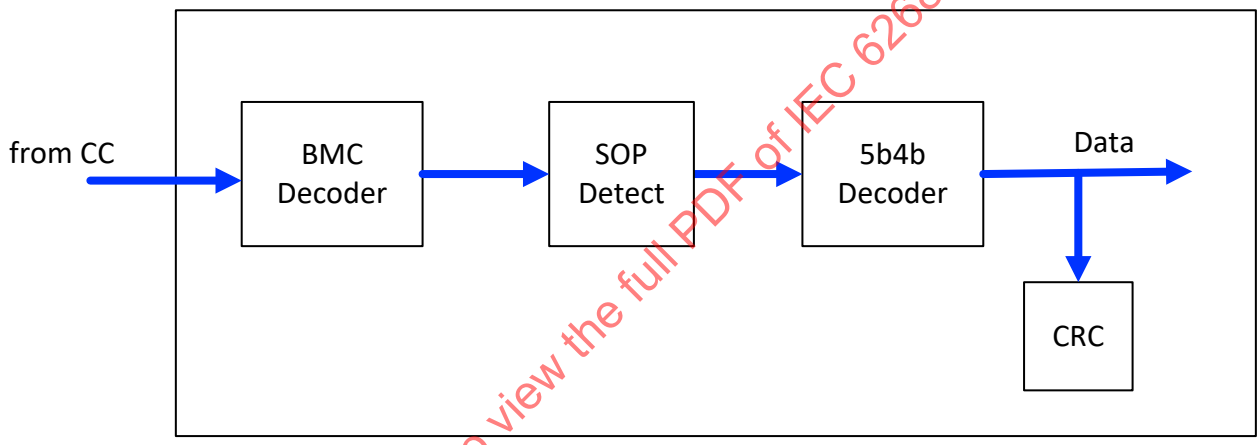


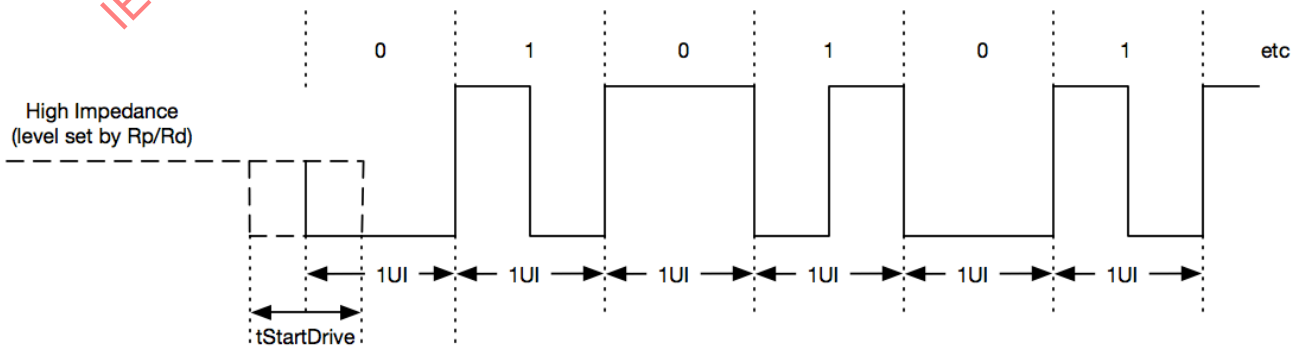
Figure 5-9 BMC Receiver Block Diagram



The USB PD baseband signal ***Shall*** be driven on the CC wire with a tristate driver that ***Shall*** cause a ***vSwing*** swing on CC. The tristate driver is slew rate limited (see min rise/fall time in Section 5.8.5) to limit coupling to D+/D- and to other signal lines in the USB Type-C® fully featured cables (see ***[USB Type-C 2.0]***). 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 5-10).

Figure 5-10 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 5-11 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 5-12 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 5.8.5.4).

Figure 5-11 Transmitting or Receiving BMC Encoded Frame Terminated by Zero with High-to-Low Last Transition

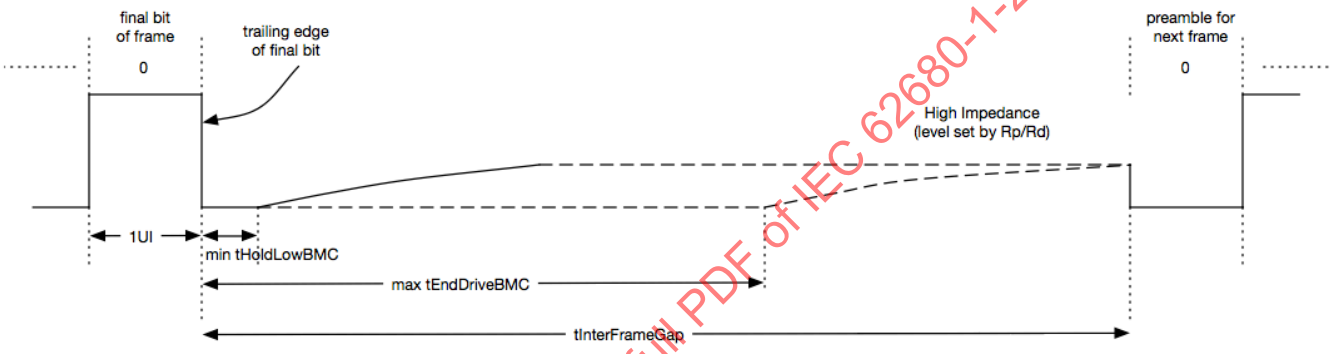
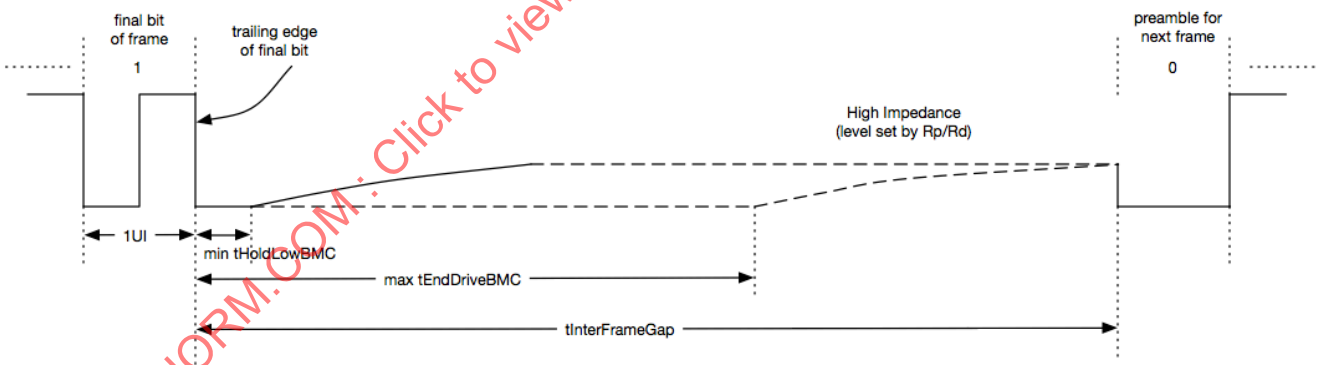


Figure 5-12 Transmitting or Receiving BMC Encoded Frame Terminated by One with High-to-Low Last Transition



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

Figure 5-13 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 5-14 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 ***tInterFrameGap*** timing requirement before the start of the next Frame when the Port has either been transmitting or receiving the previous Frame (see Section 5.8.5.4).

Figure 5-13 Transmitting or Receiving BMC Encoded Frame Terminated by Zero with Low to High Last Transition

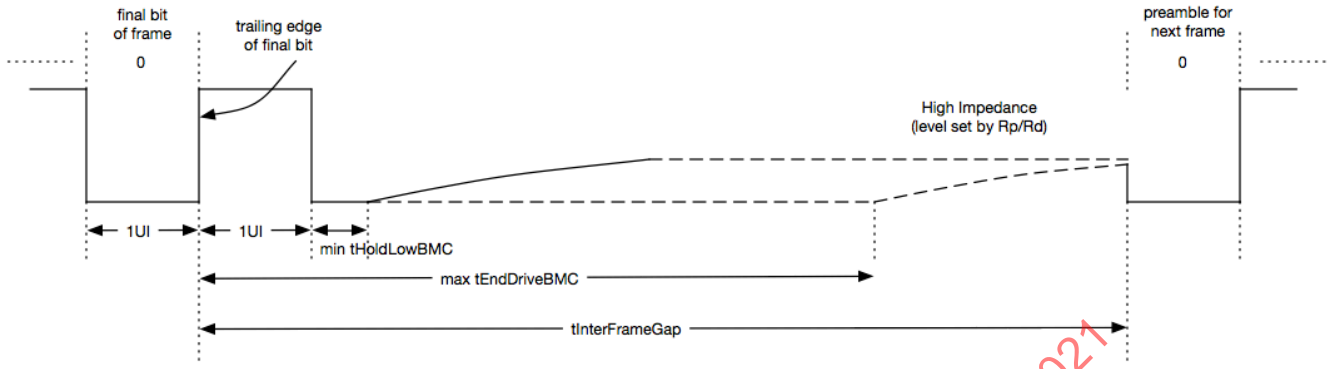
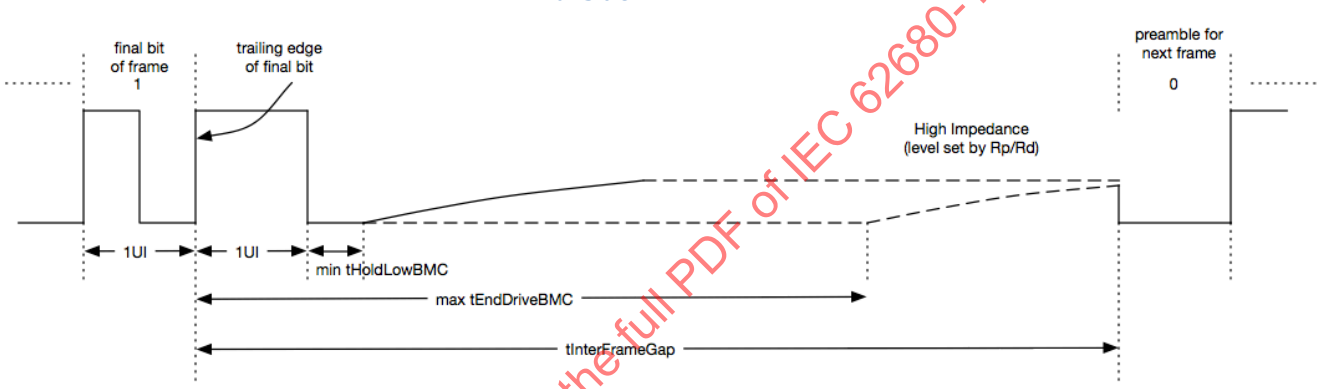


Figure 5-14 Transmitting or Receiving BMC Encoded Frame Terminated by One with Low to High Last Transition



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

5.8.2 Transmit and Receive Masks

5.8.2.1 Transmit Masks

The transmitted signal ***Shall Not*** violate the masks defined in Figure 5-15, Figure 5-16, Table 5-14 and Table 5-15 at the output of a load equivalent to the cable model and receiver load model described in Section 5.8.3. The masks apply to the full range of Rp/Rd values as defined in ***[USB Type-C 2.0]***. 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 ***tRise***. The transmitted signal ***Shall*** have a fall time no faster than ***tFall***. The maximum limits on the rise and fall times are enforced by the Tx inner masks.

Figure 5-15 BMC Tx 'ONE' Mask

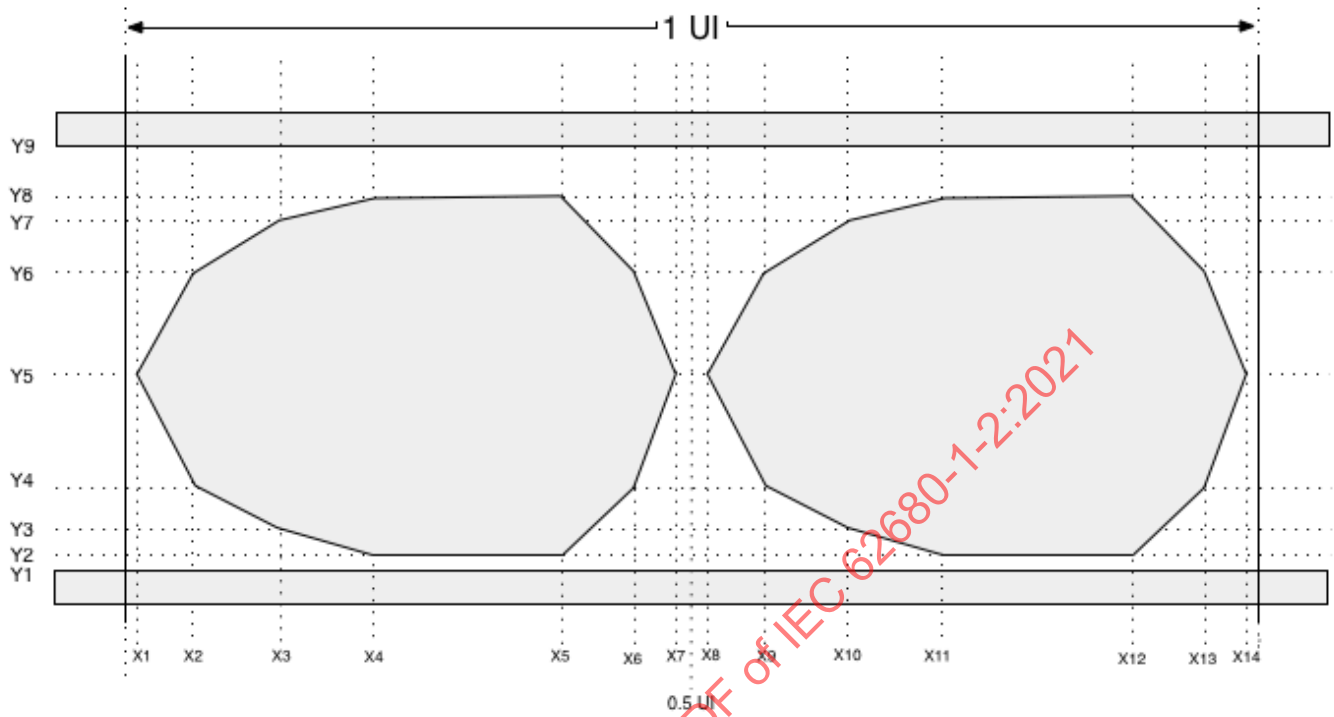


Figure 5-16 BMC Tx 'ZERO' Mask

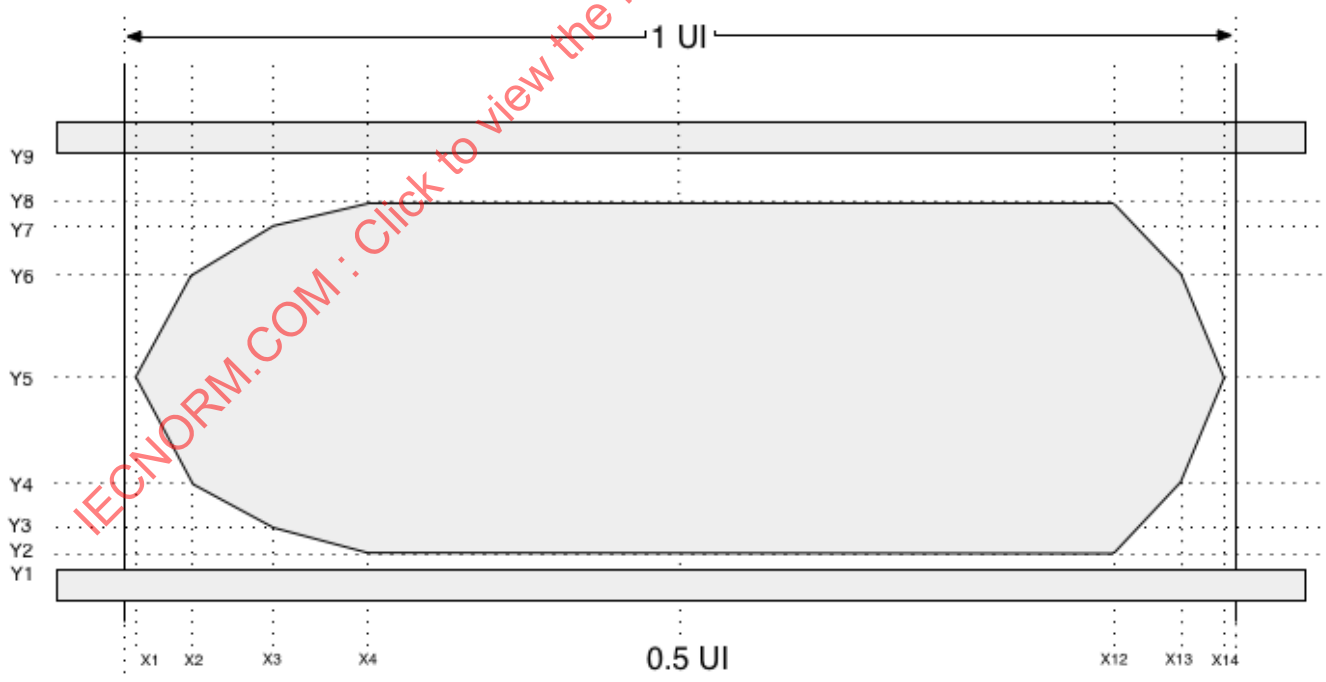


Table 5-14 BMC Tx Mask Definition, X Values

Name	Description	Value	Units
X1Tx	Left Edge of Mask	0.015	UI
X2Tx	see figure	0.07	UI

Name	Description	Value	Units
X3Tx	see figure	0.15	UI
X4Tx	see figure	0.25	UI
X5Tx	see figure	0.35	UI
X6Tx	see figure	0.43	UI
X7Tx	see figure	0.485	UI
X8Tx	see figure	0.515	UI
X9Tx	see figure	0.57	UI
X10Tx	see figure	0.65	UI
X11Tx	see figure	0.75	UI
X12Tx	see figure	0.85	UI
X13Tx	see figure	0.93	UI
X14Tx	Right Edge of Mask	0.985	UI

Table 5-15 BMC Tx Mask Definition, Y Values

Name	Description	Value	Units
Y1Tx	Lower bound of Outer mask	-0.075	V
Y2Tx	Lower bound of inner mask	0.075	V
Y3Tx	see figure	0.15	V
Y4Tx	see figure	0.325	V
Y5Tx	Inner mask vertical midpoint	0.5625	V
Y6Tx	see figure	0.8	V
Y7Tx	see figure	0.975	V
Y8Tx	see figure	1.04	V
Y9Tx	Upper Bound of Outer mask	1.2	V

5.8.2.2 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 5-17, Figure 5-18 and Table 5-16. The Source Rx mask is bounded by sweeping a Tx mask compliant signal, with added ***vNoiseActive*** between power neutral and Source offsets.

A Consumer using the BMC Signaling Scheme ***Shall*** be capable of receiving a signal that complies with the mask when sinking power as defined in Figure 5-21, Figure 5-22 and Table 5-16. The Consumer Rx mask is bounded by sweeping a Tx mask compliant signal, with added ***vNoiseActive*** between power neutral and Consumer offsets.

Every product using the BMC Signaling Scheme ***Shall*** be capable of receiving a signal that complies with the mask when power neutral as defined in Figure 5-19, Figure 5-20 and Table 5-16.

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 ***vIRDropGNDC*** max for sourcing power and is shifted down by half of ***vIRDropGNDC*** 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 5-17 BMC Rx 'ONE' Mask when Sourcing Power

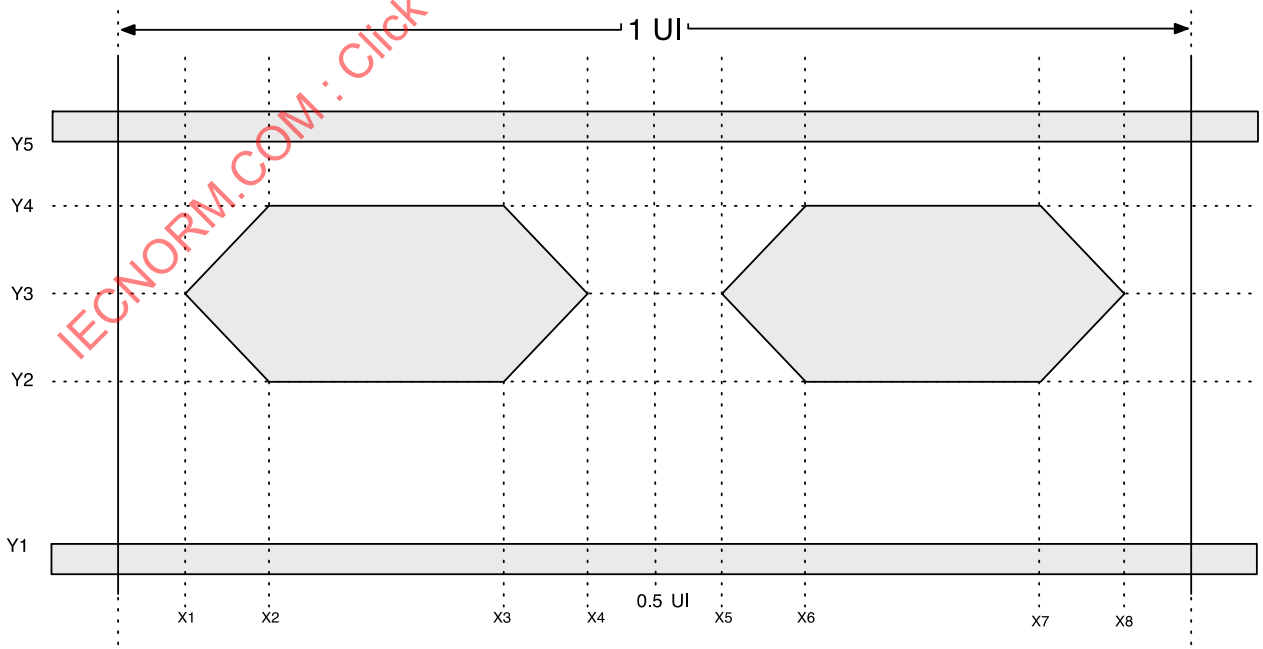


Figure 5-18 BMC Rx 'ZERO' Mask when Sourcing Power

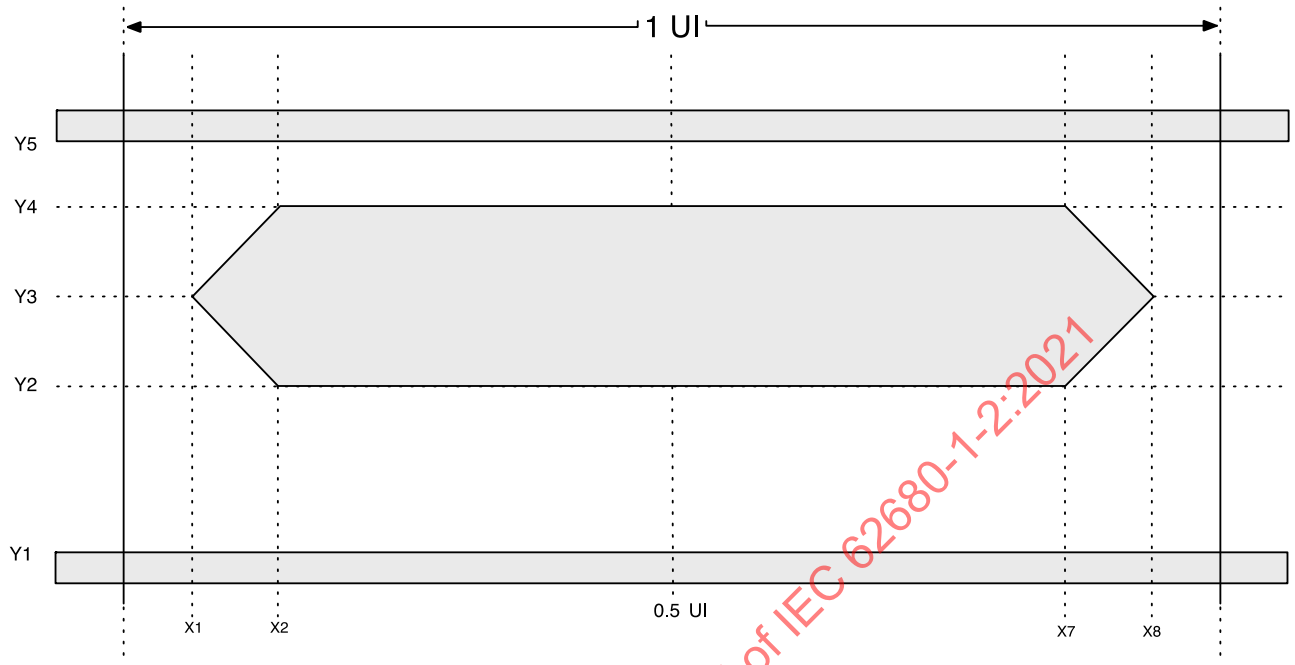


Figure 5-19 BMC Rx 'ONE' Mask when Power neutral

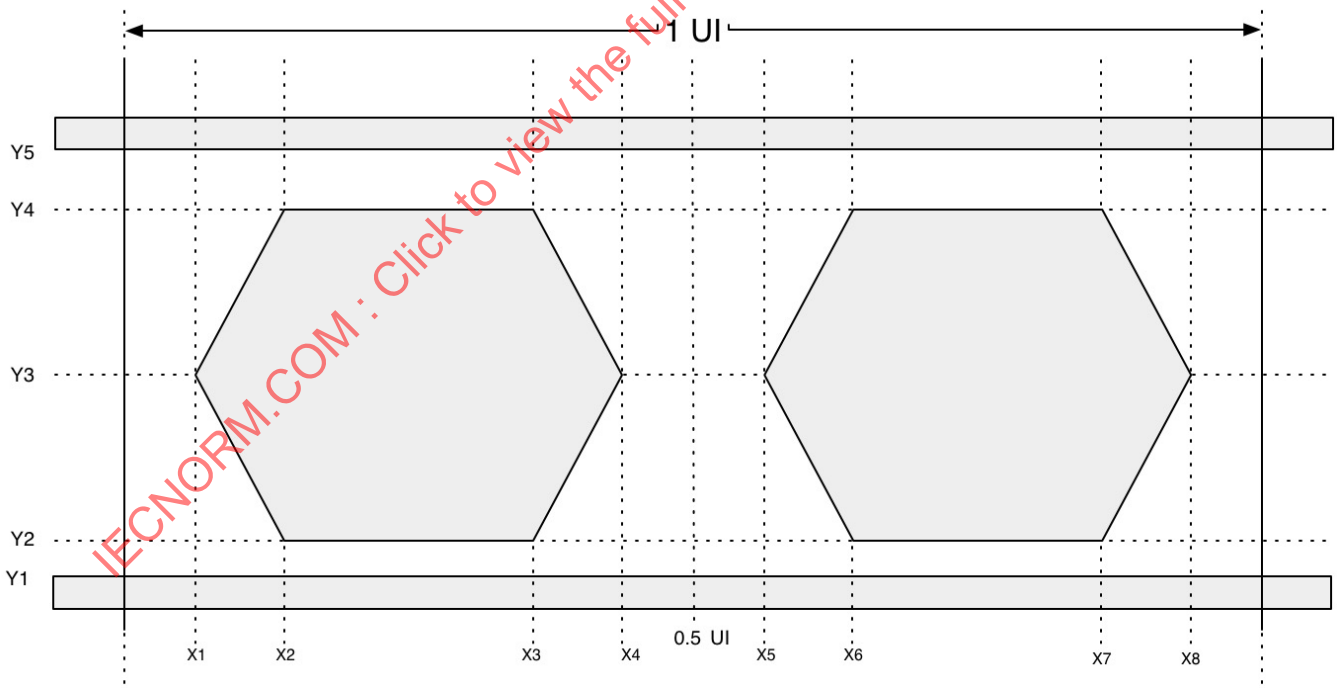


Figure 5-20 BMC Rx 'ZERO' Mask when Power neutral

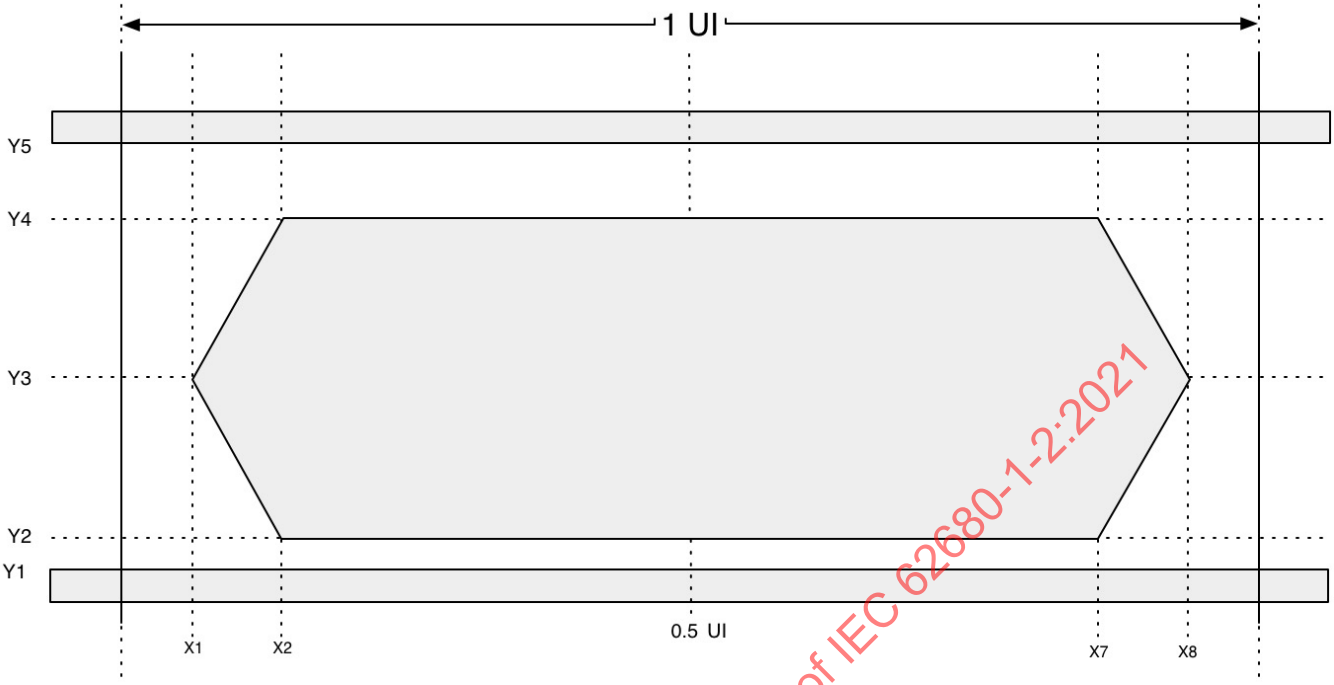


Figure 5-21 BMC Rx 'ONE' Mask when Sinking Power

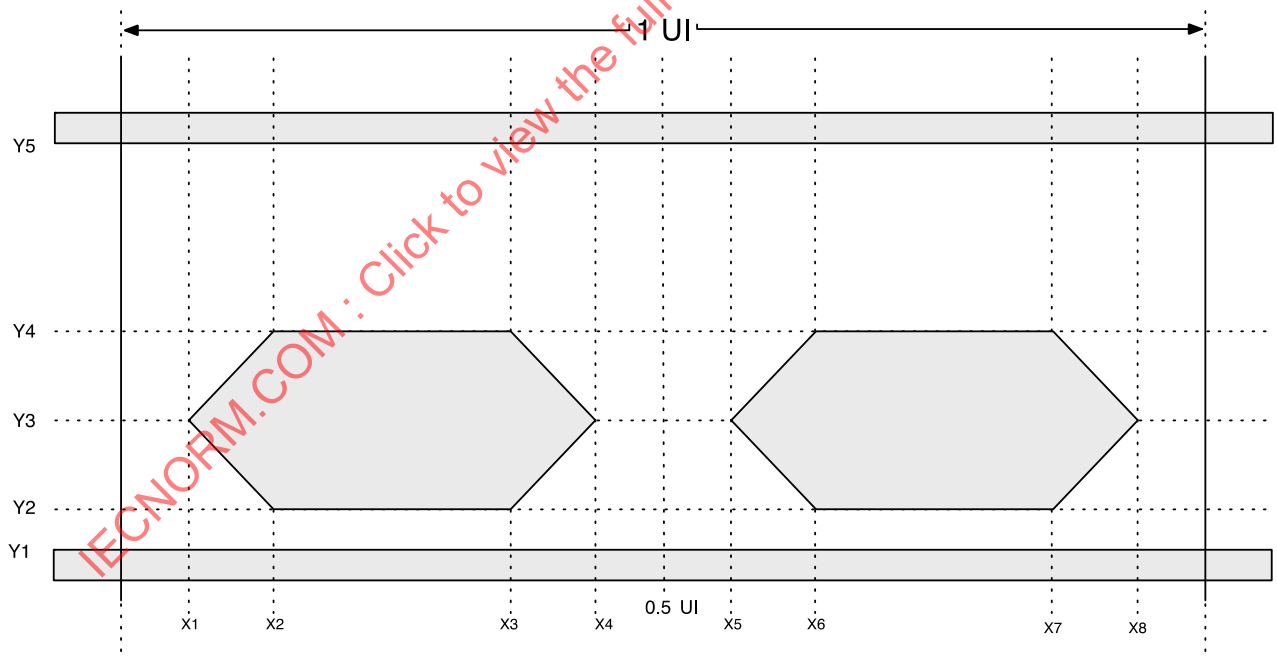


Figure 5-22 BMC Rx 'ZERO' Mask when Sinking Power

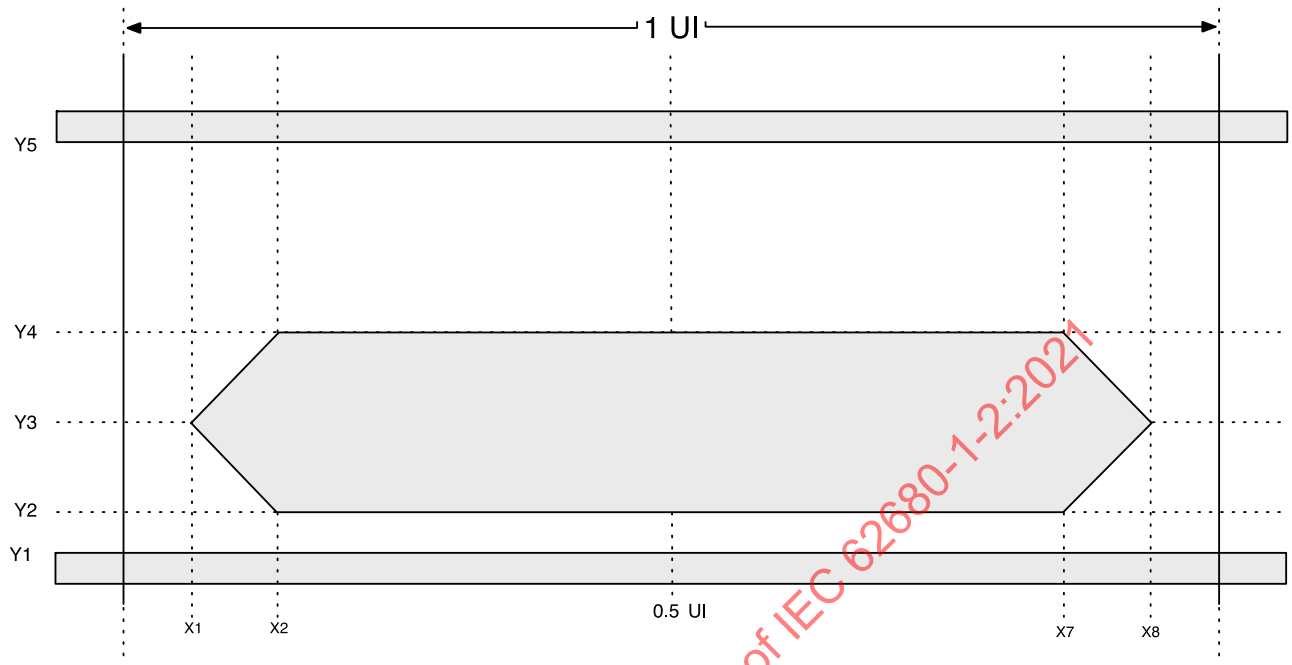


Table 5-16 BMC Rx Mask Definition

Name	Description	Value	Units
X1Rx	Left Edge of Mask	0.07	UI
X2Rx	Top Edge of Mask	0.15	UI
X3Rx	See figure	0.35	UI
X4Rx	See figure	0.43	UI
X5Rx	See figure	0.57	UI
X6Rx	See figure	0.65	UI
X7Rx	See figure	0.85	UI
X8Rx	See figure	0.93	UI
Y1Rx	Lower bound of Outer Mask	-0.3325	V
Y2Rx	Lower Bound of Inner Mask	***Y3Rx*** – 0.205 when sourcing power[1] or sinking power[1] ***Y3Rx*** – 0.33 when power neutral[1]	V
Y3Rx	Center line of Inner Mask	0.6875 Sourcing Power[1] 0.5625 Power Neutral[1] 0.4375 Sinking Power[1]	V
Y4Rx	Upper bound of Inner mask	***Y3Rx*** + 0.205 when sourcing power[1] or sinking power[1] ***Y3Rx*** + 0.33 when power neutral[1]	V
Y5Rx	Upper bound of the Outer mask	1.5325	V
Note 1: The position of the center line of the Inner Mask is dependent on whether the receiver is Sourcing or Sinking power or is Power Neutral (see earlier in this section).			

5.8.3 Transmitter Load Model

The transmitter load model ***Shall*** be equivalent to the circuit outlined in Figure 5-23 for a Source and Figure 5-24 for a Sink. It is formed by the concatenation of a cable load model and a receiver load model. See ***[USB Type-C 2.0]*** for details of the Rp and Rd resistors. Note the parameters *zCable_CC*, *tCableDelay_CC* and *cCablePlug_CC* are defined in ***[USB Type-C 2.0]***.

Figure 5-23 Transmitter Load Model for BMC Tx from a Source

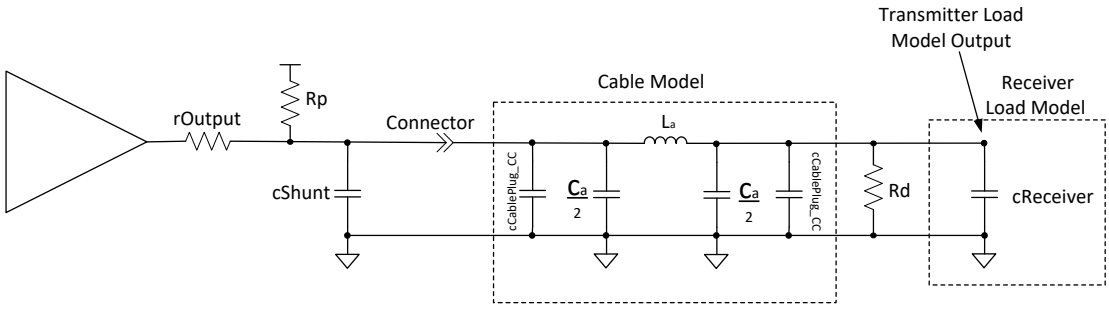
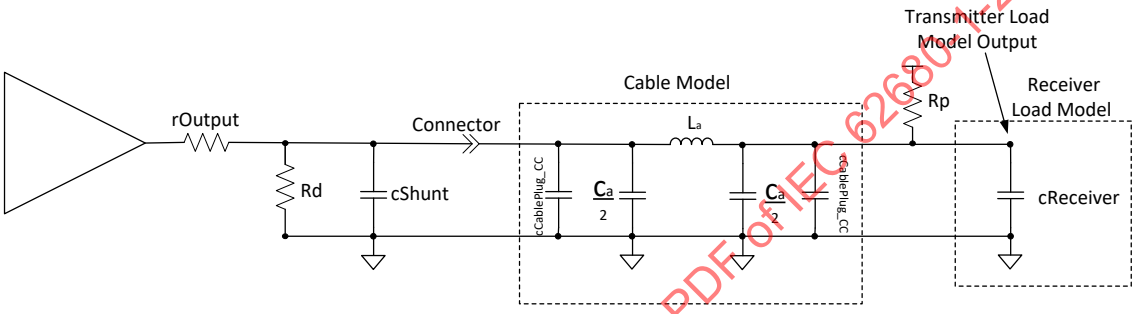


Figure 5-24 Transmitter Load Model for BMC Tx from a Sink



The transmitter system components *rOutput* and *cShunt* are illustrated for informative purposes, and do not form part of the transmitter load model. See Section 5.8.5 for a description of the transmitter system design.

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

$$La = tCableDelay\_CC_{max} * zCable\_CC_{min}$$

tCableDelay_CC is the modeled signal propagation delay through the cable, and *zCable_CC* is the modeled cable impedance.

The modeled cable inductance is 640 nH for a cable with $zCable\_CC_{min}$ = 32 Ω and $tCableDelay\_CC_{max}$ = 20 nS.

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

$$Ca = \frac{tCableDelay\_CC_{max}}{zCable\_CC_{min}}$$

The modeled cable capacitance is Ca = 625 pF for a cable with $zCable\_CC_{min}$ = 32 Ω and $tCableDelay\_CC_{max}$ = 20 nS. Therefore, Ca/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.

5.8.4 BMC Common specifications

This section defines the common receiver and transmitter requirements.

5.8.4.1 BMC Common Parameters

The electrical requirements specified in Table 5-17 ***Shall*** apply to both the transmitter and receiver.

Table 5-17 BMC Common Normative Requirements

Name	Description	Min	Nom	Max	Units	Comment
fBitRate	Bit rate	270	300	330	Kbps	
tUnitInterval[1]	Unit Interval	3.03		3.70	µs	1/*fBitRate*
Note 1: *tUnitInterval* denotes the time to transmit an unencoded data bit, not the shortest high or low times on the wire after encoding with BMC. A single data bit cell has duration of 1UI, but a data bit cell with value 1 will contain a centrally placed 01 or 10 transition in addition to the transition at the start of the cell.						

5.8.5 BMC Transmitter Specifications

The transmitter ***Shall*** meet the specifications defined in Table 5-18.

Table 5-18 BMC Transmitter Normative Requirements

Name	Description	Min	Nom	Max	Units	Comment
pBitRate	Maximum difference between the bitrate during the part of the packet following the Preamble and the reference bitrate.			0.25	%	The reference bit rate is the average bit rate of the last 32 bits of the Preamble.
rFRSwapTx	Fast Role Swap Request transmit driver resistance (excluding cable resistance)			5	Ω	Maximum driver resistance of a Fast Role Swap Request transmitter. Assumes a worst case cable resistance of 15Ω as defined in *[USB Type-C 2.0]*. Note: based on this value the maximum combined driver and cable resistance of a Fast Role Swap Request transmitter is 20Ω.
tEndDriveBMC	Time to cease driving the line after the end of the last bit of the Frame.			23	µs	Min value is limited by *tHoldLowBMC*.
tFall	Fall Time	300			ns	10 % and 90 % amplitude points, minimum is under an unloaded condition.
tHoldLowBMC	Time to cease driving the line after the final high-to-low transition.	1			µs	Max value is limited by *tEndDriveBMC*.
tInterFrameGap	Time from the end of last bit of a Frame until the start of the first bit of the next Preamble.	25			µs	
tFRSwapTx	Fast Role Swap Request transmit duration	60		120	µs	Fast Role Swap Request is indicated from the initial Source to the initial Sink by driving CC low for this time.
tRise	Rise time	300			ns	10 % and 90 % amplitude points, minimum is under an unloaded condition.
tStartDrive	Time before the start of the first bit of the Preamble when the transmitter ***Shall*** start driving the line.	-1		1	µs	
vSwing	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.

Name	Description	Min	Nom	Max	Units	Comment
zDriver	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.

5.8.5.1 Capacitance when not transmitting

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*.

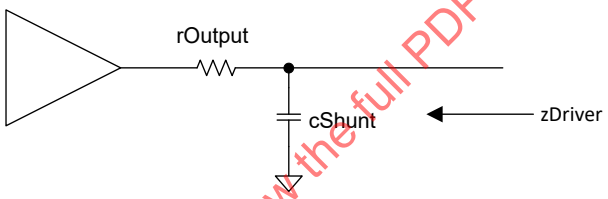
5.8.5.2 Source Output Impedance

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 ingression 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 5-25 Transmitter diagram illustrating zDriver



cShunt ***Shall Not*** cause a violation of *cReceiver* when not transmitting.

5.8.5.3 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:

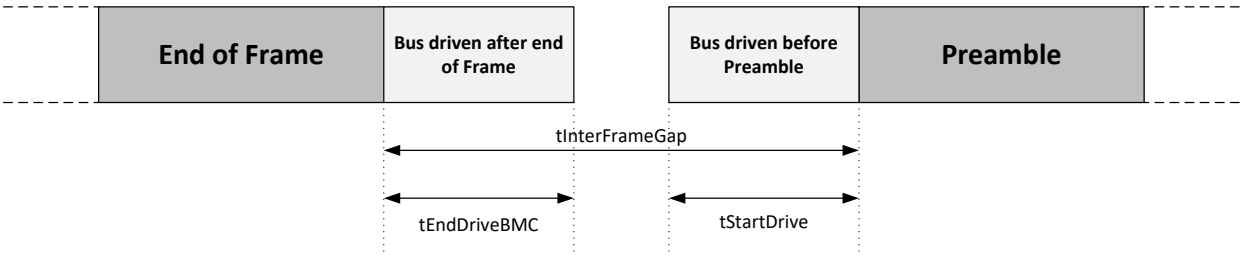
pBitRate = | *fBitRate* – refBitRate | / refBitRate x 100%

The transmitter ***Shall*** have the same *pBitRate* for all packet types. The *BIST Carrier Mode* 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 .

5.8.5.4 Inter-Frame Gap

Figure 5-26 illustrates the inter-Frame gap timings.

Figure 5-26 Inter-Frame Gap Timings



The transmitter ***Shall*** drive the bus for no longer than ***tEndDriveBMC*** after transmitting the final bit of the Frame.

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 5.8.6.1) 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 5.8.6.1).

The transmitter of the next Frame ***May*** vary the start of the Preamble by ***tStartDrive*** (see Section 5.8.1).

See also Section 5.8.1 for figures detailing the timings relating to transmitting, receiving and observing idle in relating to Frames.

5.8.5.5 Shorting of Transmitter Output

A Transmitter in a Port or Cable Plug ***Shall*** tolerate having its output be shorted to ground for ***tFRSwapTx*** max. This is due to the potential for Fast Role Swap to be signaled while the Transmitter is in the process of transmitting (see Section 5.8.5.6).

5.8.5.6 Fast Role Swap Transmission

The Fast Role Swap process is intended for use by a PDUSB HUB that presently has an external wall supply and is providing power both through its downstream Ports to USB Devices and upstream to a USB Host such as a notebook. On removal of the external wall supply Fast Role Swap enables a V_{BUS} supply to be maintained by allowing the USB Host to apply ***vSafe5V*** when it sees V_{BUS} droop below ***vSafe5V*** after having detected Fast Role Swap signaling. The Fast Role Swap AMS is then used to correctly assign Source/Sink roles and configure the Rp/Rd resistors (see Section 8.3.2.7).

The initial Source ***Shall*** signal a Fast Role Swap Request by driving CC to ground with a resistance of less than ***rFRSwapTx*** for ***tFRSwapTx***. The initial Source ***Shall*** only signal a Fast Role Swap when it has an Explicit Contract. The initial Source ***May*** signal a Fast Role Swap even if it has not yet had its Sink Capabilities queried by the initial Sink. On transmission of the Fast Role Swap signal any pending Messages ***Shall*** be ***Discarded*** (see Section 6.11.2.2.1).

The Fast Role Swap signal ***May*** override any active transmissions.

Since the initial Sink's response to the Fast Role Swap signal is to send an ***FR_Swap*** Message, the initial Source ***Shall*** ensure Rp is set to ***SinkTxOk*** once the Fast Role Swap signal is complete.

5.8.6 BMC Receiver Specifications

The receiver ***Shall*** meet the specifications defined in Table 5-19.

Table 5-19 BMC Receiver Normative Requirements

Name	Description	Min	Nom	Max	Units	Comment
cReceiver	CC receiver capacitance	200		600	pF	The DFP or UFP system ***Shall*** have capacitance within this range when not transmitting on the line.
nBER	Bit error rate, S/N = 25 dB			10^{-6}		
nTransitionCount	Transitions for signal detect	3				Number of transitions to be detected to declare bus non-idle.
tFRSwapRx	Fast Role Swap Request detection time	30		50	µs	A Fast Role Swap Request results in the receiver detecting a signal low for at least this amount of time.
tRxFilter	Rx bandwidth limiting filter (digital or analog)	100			ns	Time constant of a single pole filter to limit broad-band noise ingression[1].
tTransitionWindow	Time window for detecting non-idle	12		20	µs	
vFRSwapCableTx	Fast Role Swap Request voltage detection threshold	490	520	550	mV	The Fast Role Swap Request has to be below this voltage threshold to be detected.
vIRDropGNDC	Cable Ground IR Drop			250	mV	As specified in ***[USB Type-C 2.0]***
vNoiseActive	Noise amplitude when BMC is active.			165	mV	Peak-to-peak noise from V_{BUS}, USB 2.0 and SBU lines after the Rx bandwidth limiting filter with the time constant ***tRxFilter*** has been applied.
vNoiseIdle	Noise amplitude when BMC is idle.			300	mV	Peak-to-peak noise from V_{BUS}, USB 2.0 and SBU lines after the Rx bandwidth limiting filter with the time constant ***tRxFilter*** has been applied.
zBmcRx	Receiver Input Impedance	1			MΩ	
Note 1: Broad-band noise ingression is due to coupling in the cable interconnect.						

5.8.6.1 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.

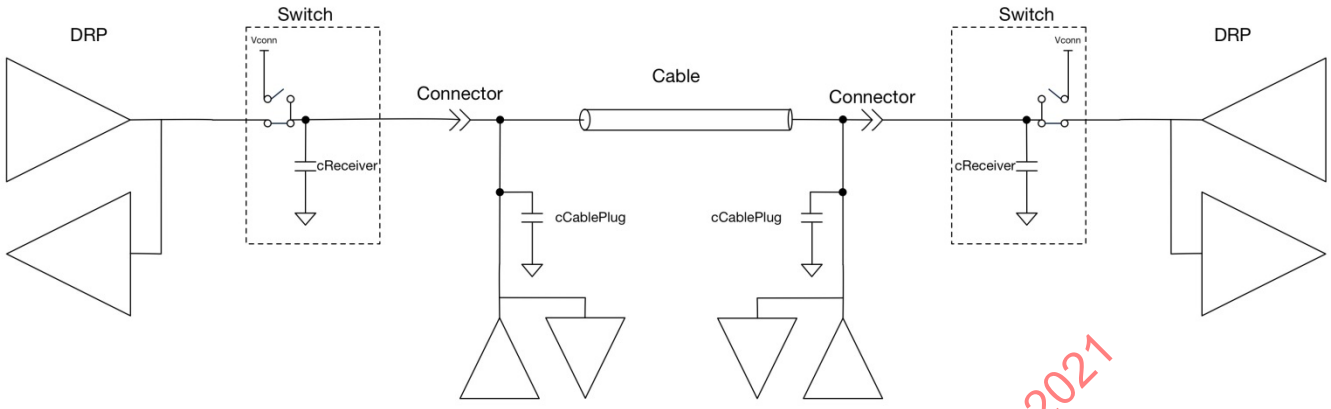
Refer to Section 5.8.5.4 for details of when transmissions ***May*** start.

5.8.6.2 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 5-27 below illustrates a typical Multi Drop configuration with two DRPs.

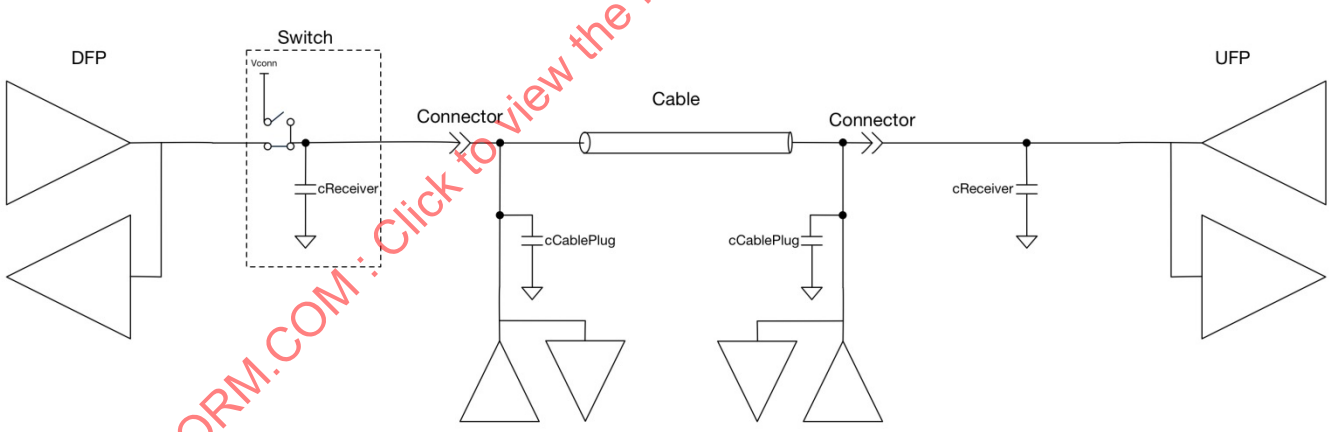
Figure 5-27 Example Multi-Drop Configuration showing two DRPs



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* defined in ***[USB Type-C 2.0]*** . 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 5-28. 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 5-28 Example Multi-Drop Configuration showing a DFP and UFP



5.8.6.3 Fast Role Swap Detection

An initial Sink prepares for a Fast Role Swap by ensuring that once it has detected the Fast Role Swap signal its power supply is ready to respond by applying ***vSafe5V*** according to the timing detailed in Section 7.1.13. The initial Sink ***Shall*** only respond to the Fast Role Swap signal when all of the following conditions have been met:

- An Explicit Contract has been established and the Sink Capabilities of the initial Source have been received by, and at the request of, the initial Sink
- The ***Sink_Capabilities*** Message received from the initial Source has at least one of the Fast Role Swap bits set in its 5V fixed PDO.
- The initial Sink is able and willing to source the current requested by the initial Source in the Fast Role Swap bits of its ***Sink_Capabilities*** Message.

On detection of the Fast Role Swap signal any pending Messages ***Shall*** be ***Discarded*** (see Section 6.11.2.2.1).

When the initial Sink is prepared for a Fast Role Swap and the bus is idle the CC voltage averaged over ***tFRSwapRx*** min remains above 0.7V (see ***[USB Type-C 2.0]***) since the Source Rp is either 1.5A or 3.0A. However, ***vNoiseIdle*** noise ***May*** cause the CC line voltage to reach 0.7V-***vNoiseIdle***/2 for short durations. When the initial Sink is prepared for a Fast Role swap while it is transmitting and the initial Source is signaling a Fast Role Swap Request, the transmission will be attenuated such that the peak CC voltage will not exceed ***vFRSwapCableTx*** min. Therefore, when the initial Sink is prepared for a Fast Role Swap, it ***Shall Not*** detect a Fast Swap signal when the CC voltage, averaged over ***tFRSwapRx*** min, is above 0.7V. When the initial Sink is prepared for a Fast Role Swap, it ***Shall*** detect a CC voltage lower than ***vFRSwapCableTx*** min for ***tFRSwapRx*** as a Fast Role Swap Request. Note: the initial Sink is not required to average the CC voltage to meet these requirements.

The initial Sink ***Shall*** initiate the Fast Role Swap AMS within ***tFRSwapInit*** of detecting the Fast Role Swap Request in order to assign the Rp/Rd resistors to the correct Ports and to re-synchronize the state machines (see Section 6.3.19).

The initial Sink ***Shall*** become the new Source and ***Shall*** start supplying ***vSafe5V*** at USB Type-C Current (see ***[USB Type-C 2.0]***) no later than ***tSrcFRSwap*** after V_{BUS} has dropped below ***vSafe5V***. An initial Sink ***Shall*** disable its V_{BUS} Disconnect Threshold detection circuitry while Fast Role Swap detection is active.

Note: while power is transitioning the V_{CONN} Source to the Cable Plug(s) cannot be guaranteed.

5.9 Built in Self-Test (BIST)

The following sections define BIST functionality which ***Shall*** be supported.

5.9.1 BIST Carrier Mode

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

Note that this transmission is a purely a sequence of alternating bits and ***Shall Not*** be formatted as a Packet.

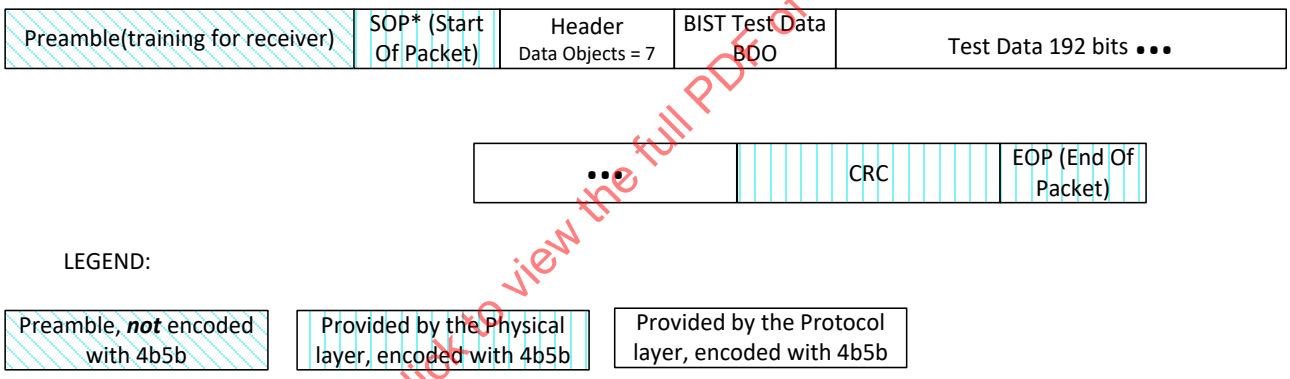
See also Section 6.4.3.

5.9.2 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 5-29 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 5-29 Test Data Frame



6. Protocol Layer

6.1 Overview

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

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

Refer to Section 2.6 for an overview of the theory of operation of USB Power Delivery.

6.2 Messages

This specification defines three types of Messages:

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

6.2.1 Message Construction

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

Figure 6-1 illustrates a Control Message as part of a Packet showing the parts are provided by the Protocol and PHY Layers.

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

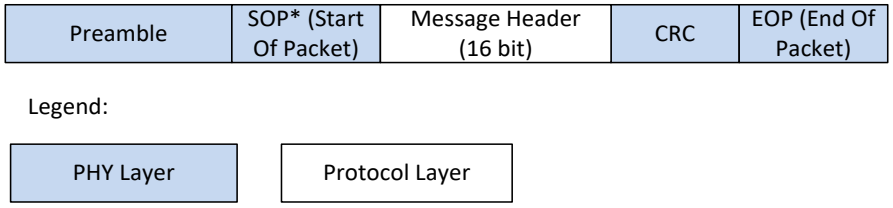


Figure 6-2 illustrates a Data Message as part of a Packet showing the parts are provided by the Protocol and PHY Layers.

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

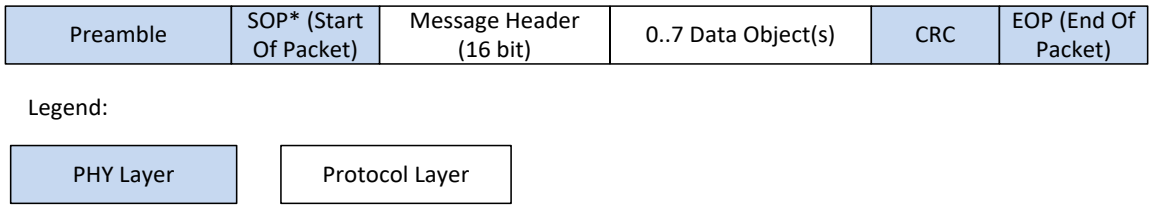
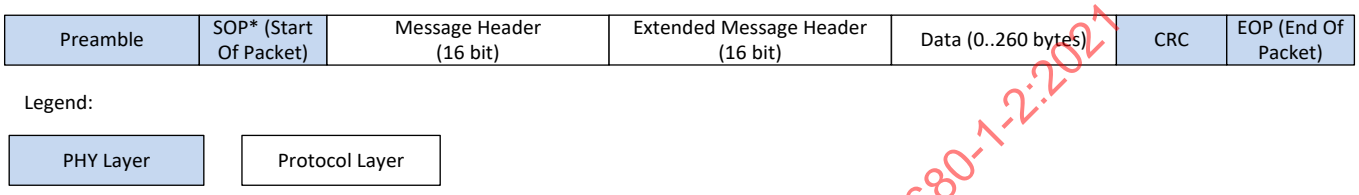


Figure 6-3 illustrates an Extended Message as part of a Packet showing the parts are provided by the Protocol and PHY Layers.

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



6.2.1.1 Message Header

Every Message ***Shall*** start with a Message Header as shown in Figure 6-1, Figure 6-2 and Figure 6-3 and as defined in Table 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	Reference
15	SOP*	***Extended***	Section 6.2.1.1.1
14…12	SOP*	***Number of Data Objects***	Section 6.2.1.1.2
11…9	SOP*	***MessageID***	Section 6.2.1.1.3
8	SOP only	***Port Power Role***	Section 6.2.1.1.4
	SOP'/SOP"	***Cable Plug***	Section 6.2.1.1.7
7…6	SOP*	***Specification Revision***	Section 6.2.1.1.5
5	SOP only	***Port Data Role***	Section 6.2.1.1.6
	SOP'/SOP"	***Reserved***	Section 1.4.2.10
4…0	SOP*	***Message Type***	Section 6.2.1.1.8

6.2.1.1.1 Extended

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

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

6.2.1.1.2 Number of Data Objects

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

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

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

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

6.2.1.1.3 MessageID

The 3-bit ***MessageID*** field is the value generated by a rolling counter maintained by the originator of the Message. The ***MessageIDCounter*** ***Shall*** be initialized to zero at power-on as a result of a Soft Reset, or a Hard Reset. The ***MessageIDCounter*** ***Shall*** be incremented when a Message is successfully received as indicated by receipt of a ***GoodCRC*** Message. Note: the usage of ***MessageID*** during testing with BIST Messages is defined in ***[USBPDCompliance]***.

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

6.2.1.1.4 Port Power Role

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

- 0b Sink
- 1b Source

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

During the Power Role Swap Sequence, for the initial Source Port, the ***Port Power Role*** field ***Shall*** be set to Sink in the ***PS_RDY*** Message indicating that the initial Source's power supply is turned off (see 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).

During the Fast Role Swap Sequence, for the initial Source Port, the ***Port Power Role*** field ***Shall*** be set to Sink in the ***PS_RDY*** Message indicating that V_{BUS} is not being driven by the initial Source and is within ***vSafe5V*** (see Figure 8-18).

During the Fast Role Swap Sequence, for the initial Sink Port, the ***Port Power Role*** field ***Shall*** be set to Source for Messages initiated by the Policy Engine after receiving the ***PS_RDY*** Message from the initial Source (see Figure 8-18).

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

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

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

6.2.1.1.5 Specification Revision

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

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

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

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

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

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

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

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

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

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

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

1. The VCONN Source sends a ***Discover Identity*** REQ to the Cable Plug (SOP') setting the ***Specification Revision*** field in the Message to the highest Revision of the Power Delivery Specification the VCONN Source supports. After a VCONN Swap the required ***Soft_Reset*** / ***Accept*** message exchange is used for the same purpose (see Section 6.3.13).
2. The Cable Plug responds with a ***Discover Identity*** ACK setting the ***Specification Revision*** field in the Message to the highest Revision of the Power Delivery Specification the VCONN Source supports that is equal to or lower than the ***Specification Revision*** it received from the Source Port.
3. The Cable Plug and VCONN Source ***Shall*** communicate using the lower of the two revisions until an Explicit Contract has been established.
4. Table 6-2 shows the ***Specification Revision*** that ***Shall*** be used between the Port Partners and the Cable Plugs when the ***Specification Revision*** has been discovered and an Explicit Contract is in place.

Notes:

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

A VCONN Source that supports Revision 3.0 of the Power Delivery Specification ***May*** communicate with a Cable Plug also supporting Revision 3.0 using Revision 3.0 Compliant Communications regardless of the ***Specification Revision*** of its Port Partner while no Explicit Contract exists. After an Explicit Contract has been established the Port Partners and Cable Plug(s) ***Shall*** use Table 6-2 to determine the Revision to be used.

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

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

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

See Table 6-2 for details of how various Revisions ***Shall*** interoperate.

Table 6-2 Revision Interoperability during an Explicit Contract

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

6.2.1.1.6 Port Data Role

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

- 0b UFP
- 1b DFP

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

If a USB Type-C® Port receives a Message with the ***Port Data Role*** field set to the same Data Role as its current Data Role, except for the ***GoodCRC*** Message, USB Type-C Error Recovery actions as defined in ***[USB Type-C 2.0]*** ***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 Rd asserted and 1b for a Port with Rp asserted.

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

6.2.1.1.7 Cable Plug

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

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

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

6.2.1.1.8 Message Type

The 5-bit ***Message Type*** field ***Shall*** indicate the type of Message being sent. To fully decode the ***Message Type***, the ***Number of Data Objects*** field is first examined to determine whether the Message is a Control Message or a Data Message. Then the specific ***Message Type*** can be found in Table 6-5 (Control Message) or Table 6-6 (Data Message).

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

6.2.1.2 Extended Message Header

Every Extended Message (indicated by the ***Extended*** field being set in the Message Header) ***Shall*** contain an Extended Message Header following the Message Header as shown in Figure 6-3 and defined in Table 6-3.

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

Table 6-3 Extended Message Header

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

6.2.1.2.1 Chunked

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

When either Port Partner only supports Chunked Extended Messages:

1. The ***Chunked*** bit in every Extended Message ***Shall*** be set to one
2. Every Extended Message of Data Size > ***MaxExtendedMsgLegacyLen*** ***Shall*** be transmitted between the Port Partners in Chunks
3. The ***Number of Data Objects*** in the Message Header ***Shall*** indicate the number of Data Objects in the Message padded to the 4-byte boundary including the Extended Header as part of the first Data Object.
4. Point 1, Point 2 and Point 3 above ***Shall*** apply until the Port Pair is Detached, there is a Hard Reset or the Source removes power (except during a Power Role Swap or Fast Role Swap when the initial Source removes power in order to for the new Source to apply power).

When both Port Partners support Unchunked Extended Messages:

1. The ***Chunked*** bit in every Extended Message ***Shall*** be set to zero.
2. Every Extended Message ***Shall*** be transmitted between the Port Partners Unchunked
3. The ***Number of Data Objects*** in the Message Header is ***Reserved***.
4. Point 1, Point 2 and Point 3 above ***Shall*** apply until the Port Pair is Detached, there is a Hard Reset or the Source removes power (except during a Power Role Swap or Fast Role Swap when the initial Source removes power in order to for the new Source to apply power).

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

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

6.2.1.2.2 Chunk Number

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

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

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

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

6.2.1.2.3 Request Chunk

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

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

6.2.1.2.4 Data Size

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

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

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

6.2.1.2.5 Extended Message Examples

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

During negotiation of the Explicit Contract after connection, the Port Partners use the Unchunked Extended Messages Supported fields in the ***Source_Capabilities*** Message and the ***Request*** Message to determine the value of the ***Chunked*** bit (see Table 6-4). When both Port Partners support Unchunked Messages then the ***Chunked*** bit is zero otherwise the ***Chunked*** bit is one.

The ***Chunked*** bit is used to determine whether or not:

- The Chunk request/response mechanism is used
- Extended Messages are Chunked
- Padding is applied
- The ***Number of Data Objects*** field is used

The following examples illustrate the expected usage in each case.

Table 6-4 Use of Unchunked Message Supported bit

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

6.2.1.2.5.1 Security_Request/Security_Response Unchunked Example

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

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

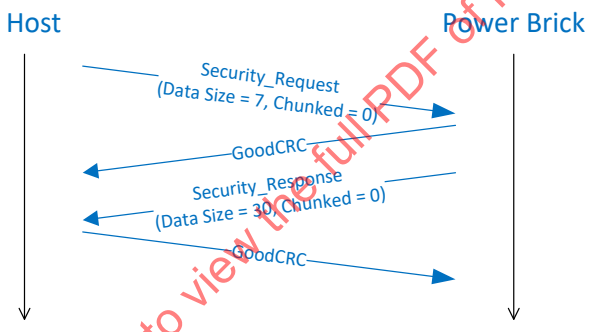


Figure 6-5 details the ***Security_Request*** Message shown in Figure 6-4. The figure shows the byte ordering on the bus as well as the fact that there is no padding in this case. The ***Number of Data Objects*** field has a value of 0 since it is ***Reserved*** when the ***Chunked*** bit is zero. The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

Figure 6-5 Example byte transmission for Security_Request Message of Data Size 7 (Chunked bit is set to 0)

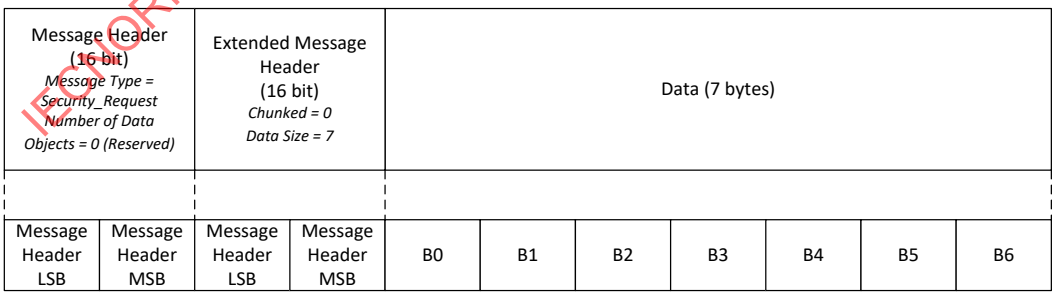
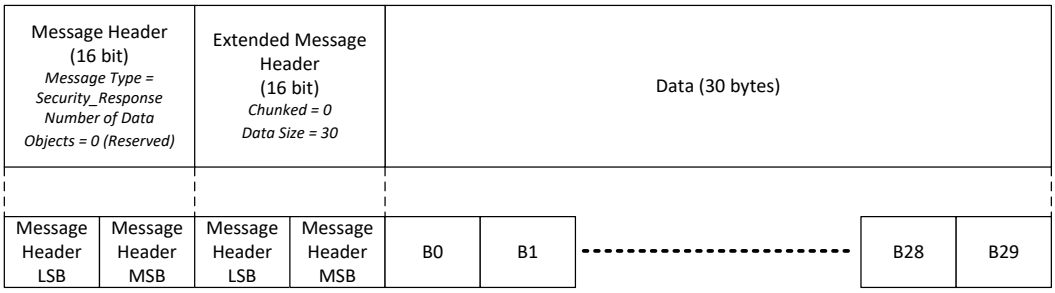


Figure 6-6 details the ***Security_Response*** Message shown in Figure 6-4. The figure shows the byte ordering on the bus as well as the fact that there is no padding in this case. The ***Number of Data Objects*** field has a value of 0 since it is ***Reserved*** when the ***Chunked*** bit is zero. The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

Figure 6-6 Example byte transmission for Security_Response Message of Data Size 7 (Chunked bit is set to 0)



6.2.1.2.5.2 Security_Request/Security_Response Chunked Example

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

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

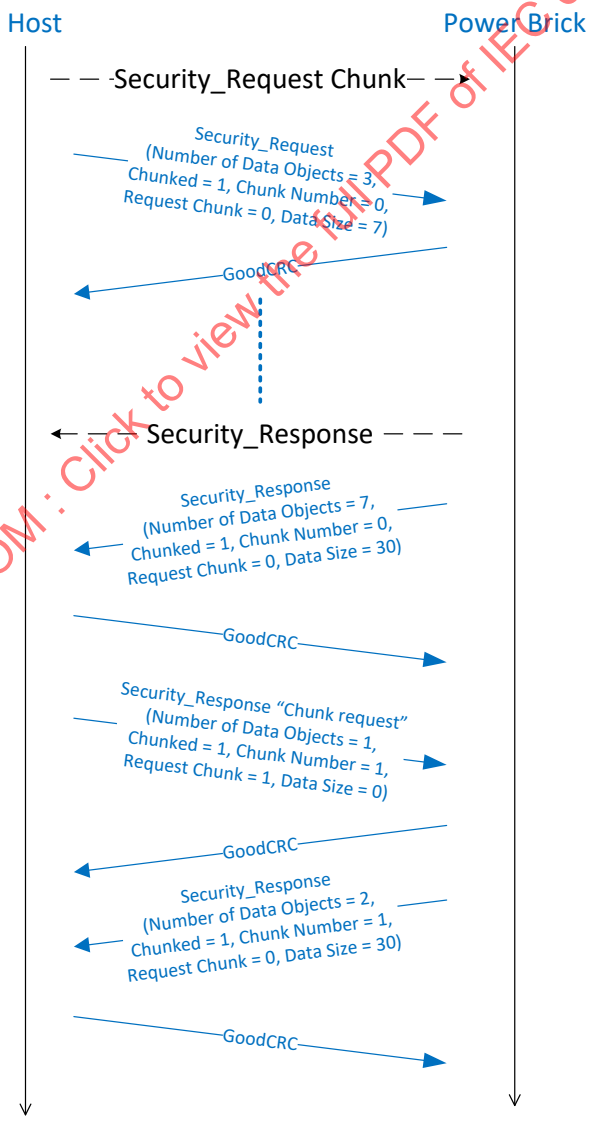


Figure 6-8 shows the ***Security_Request*** Message shown in Figure 6-7 in more detail including the byte ordering on the bus and padding. Three bytes of padding have been added to the Message so that the total number of bytes is a multiple of 32-bits, corresponding to 3 Data Objects. The ***Number of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

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

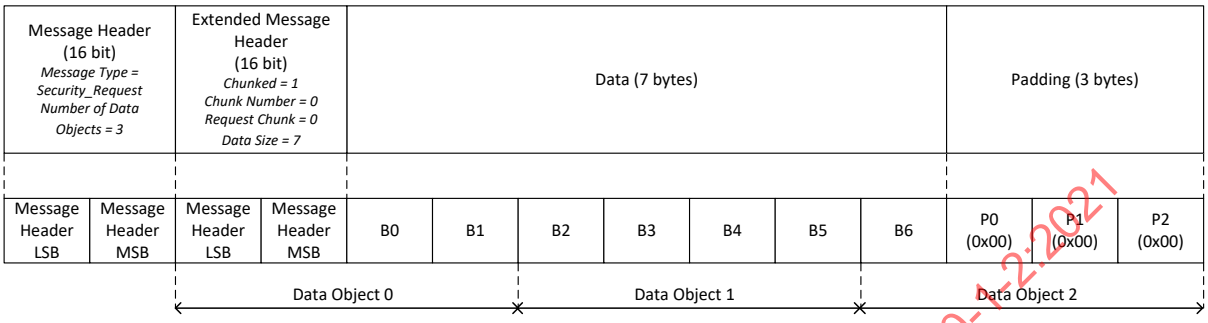


Figure 6-9 shows ***Chunk Number*** zero of the ***Security_Response*** Message shown in Figure 6-7 in more detail including the byte ordering on the bus and padding. No padding is need for this Chunk since the full 26-byte payload plus 2-byte Extended Message Header is a multiple of 32-bits, corresponding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

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

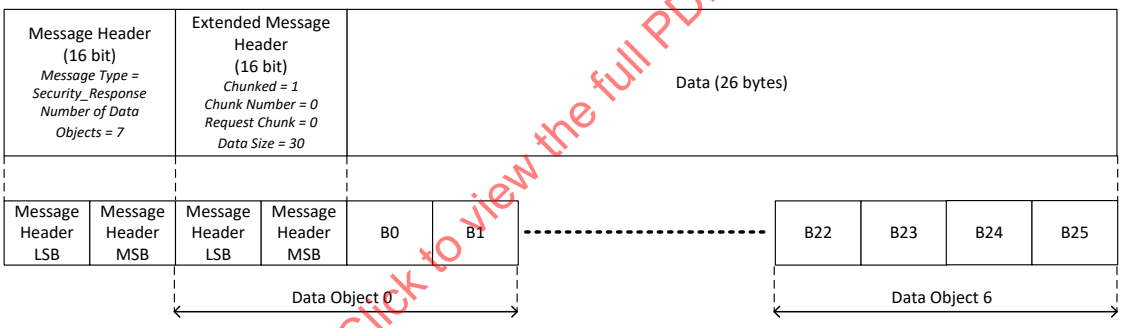


Figure 6-10 shows an example of the Message format, byte ordering and padding for the ***Security_Response*** Message Chunk request for ***Chunk Number*** one shown in Figure 6-7. In the Chunk request the ***Number of Data Objects*** field in the Message is set to 1 to indicate that the payload is 32 bits equivalent to 1 data object. Since the ***Chunked*** bit is set to 1 the Chunk request/Chunk response mechanism is used. The Message is a Chunk request so the ***Request Chunk*** bit is set to one, and in this case Chunk one is being requested so ***Chunk Number*** is set to one. ***Data Size*** is set to 0 indicating the length of the Data Block being transferred. Two bytes of padding are added to ensure that the payload is a multiple of 32 bits.

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

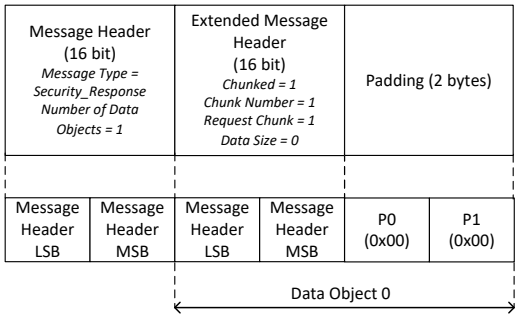
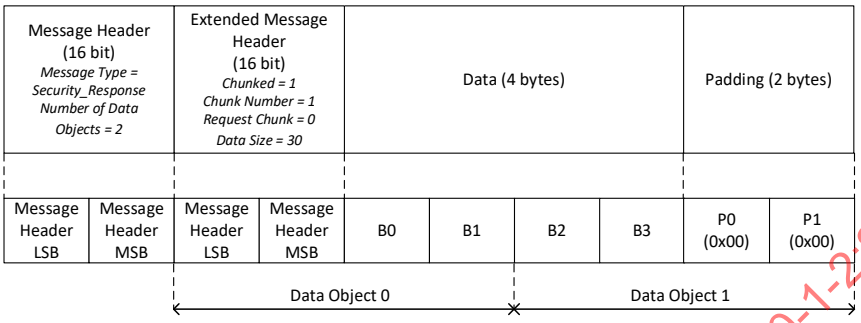


Figure 6-11 shows ***Chunk Number*** one of the ***Security_Response*** Message shown in Figure 6-7 in more detail including the byte ordering on the bus and padding. Two bytes of padding are added to ensure that the payload is a multiple of 32 bits, corresponding to 2 Data Objects. The ***Number of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

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



6.3 Control Message

A Message is defined as a Control Message when the ***Number of Data Objects*** field in the Message Header is set to 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 4…0) and are summarized in Table 6-5. The Sent by column indicates entities which ***May*** send the given Message (Source, Sink or Cable Plug); entities not listed ***Shall Not*** issue the corresponding Message. The "Valid Start of Packet" column indicates the Messages which ***Shall*** only be issued in SOP Packets and the Messages which ***May*** be issued in SOP* Packets.

Table 6-5 Control Message Types

Bits 4…0	Message Type	Sent by	Description	Valid Start of Packet
0 0000	***Reserved***	N/A	All values not explicitly defined are ***Reserved*** and ***Shall Not*** be used.	
0 0001	***GoodCRC***	Source, Sink or Cable Plug	See Section 6.3.1.	SOP*
0 0010	***GotoMin***	Source only	See Section 6.3.2.	SOP only
0 0011	***Accept***	Source, Sink or Cable Plug	See Section 6.3.3.	SOP*
0 0100	***Reject***	Source or Sink	See Section 6.3.4.	SOP only
0 0101	***Ping***	Source only	See Section 6.3.5.	SOP only
0 0110	***PS_RDY***	Source or Sink	See Section 6.3.6.	SOP only
0 0111	***Get_Source_Cap***	Sink or DRP	See Section 6.3.7.	SOP only
0 1000	***Get_Sink_Cap***	Source or DRP	See Section 6.3.8.	SOP only
0 1001	***DR_Swap***	Source or Sink	See Section 6.3.9	SOP only
0 1010	***PR_Swap***	Source or Sink	See Section 6.3.10	SOP only
0 1011	***VCONN_Swap***	Source or Sink	See Section 6.3.11	SOP only
0 1100	***Wait***	Source or Sink	See Section 6.3.12	SOP only
0 1101	***Soft_Reset***	Source or Sink	See Section 6.3.13	SOP*
01110	***Data_Reset***	Source or Sink	See Section 6.3.14	SOP only
0 1111	***Data_Reset_Complete***	Source or Sink	See Section 6.3.15	SOP only

Bits 4…0	Message Type	Sent by	Description	Valid Start of Packet
1 0000	***Not_Supported***	Source, Sink or Cable Plug	See Section 6.3.16	SOP*
1 0001	***Get_Source_Cap_Extended***	Sink or DRP	See Section 6.3.17	SOP only
1 0010	***Get_Status***	Source or Sink	See Section 6.3.18	SOP*
1 0011	***FR_Swap***	Sink[1]	See Section 6.3.19	SOP only
1 0100	***Get_PPS_Status***	Sink	See Section 6.3.20	SOP only
1 0101	***Get_Country_Codes***	Source or Sink	See Section 6.3.21	SOP only
1 0110	***Get_Sink_Cap_Extended***	Source or DRP	See Section 6.3.22	SOP only
1 0111-1 1111	***Reserved***	N/A	All values not explicitly defined are ***Reserved*** and ***Shall Not*** be used.	
Note 1: In this case the Port is providing ***vSafe5V*** however it will have Rd asserted rather than Rp and sets the ***Port Power Role*** field to Sink, until the Fast Role Swap AMS has completed.				

6.3.1 GoodCRC Message

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

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

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 ***FR_Swap*** Message to indicate that it has begun the Fast Role Swap sequence.
- It ***Shall*** be sent by the recipient of the ***Soft_Reset*** Message to indicate that it has completed its Soft Reset.

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

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 that is not presently the VCONN Source, to indicate it is unable to do a VCONN Swap.

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

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

6.3.5 Ping Message

The ***Ping*** Message was previously used on USB Type-A and USB Type-B connectors to determine the continued presence of the Sink when no other messaging was taking place. USB Type-C connectors have a mechanism to determine Sink presence so when the Port Partners are both connected using USB Type-C connectors the ***Ping*** Message is not necessary but ***May*** be sent by a Source if desired. A Sink using a USB Type-C connector ***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 or Fast Role Swap sequence) to indicate its power supply has reached the desired operating condition (see Section 8.3.2.2).

6.3.7 Get_Source_Cap Message

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

6.3.8 Get_Sink_Cap Message

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

6.3.9 DR_Swap Message

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

[USB Type-C 2.0] DRDs ***Shall*** have the capability to perform a Data Role Swap from the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** states. DFPs and UFPs ***May*** have the capability to perform a Data Role Swap from the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** states. A Data Role Swap ***Shall*** be regarded in the same way as a cable Detach/re-Attach in relation to any USB communication which is ongoing between the Port Partners. If there are any Active 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 Rp/Rd 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 2.0]*** will be necessary.

See Section 8.3.2.8, Section 8.3.3.18.1 and Section 8.3.3.18.2 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***.

The Source ***Shall*** have Rp asserted on the CC wire and the Sink ***Shall*** have Rd asserted on the CC wire as defined in ***[USB Type-C 2.0]***. When performing a Power Role Swap from Source to Sink, the Port ***Shall*** change its CC Wire resistor from Rp to Rd. When performing a Power Role Swap from Sink to Source, the Port ***Shall*** change its CC Wire resistor from Rd to Rp. The DFP (Host), UFP (Device) roles and VCONN Source ***Shall*** remain unchanged during the Power Role Swap process.

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

For more information regarding the Power Role Swap, refer to Section 7.3.9 and Section 7.3.10 in the Power Supply chapter, Section 8.3.2.6, Section 8.3.3.18.3 and Section 8.3.3.18.4 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 Port that can operate as a VCONN Source.

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

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

The DFP (Host), UFP (Device) roles and Source of V_{BUS} ***Shall*** remain unchanged as well as the Rp/Rd 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 ***Shall*** ensure that it is the VCONN Source and that the Cable Plugs are powered, by performing a VCONN swap if necessary. Since it cannot be guaranteed that the present VCONN Source is supplying VCONN, the only means to ensure that the Cable Plugs are powered is for a Port wishing to communicate with a Cable Plug to become the VCONN Source. If a ***Not_Supported*** Message is returned in response to the ***VCONN_Swap*** Message then the Port is allowed to become the VCONN Source until a Hard Reset or Detach.

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

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

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 that is not presently the VCONN Source 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 6.6.2).

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 ***[USBTypeCBridge 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. A ***Wait*** Message ***Shall*** only be sent by the Port that is not presently the Vconn Source in response to a ***VCONN_Swap*** Message. The Port that is presently the Vconn Source ***Shall Not*** send a ***Wait*** Message in response to ***VCONN_Swap*** Message. Once it has completed this evaluation one of the Port Partners ***Should*** initiate the VCONN Swap process again by sending a ***VCONN_Swap*** Message.

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

6.3.13 Soft Reset Message

A ***Soft_Reset*** Message ***May*** be initiated by either the Source or Sink to its Port Partner requesting a Soft Reset. The ***Soft_Reset*** Message ***Shall*** cause a Soft Reset of the connected Port Pair (see Section 6.8.1). 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 6.8).

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

6.3.14 Data_Reset Message

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

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

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

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

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

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

6.3.15 Data_Reset_Complete Message

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

6.3.16 Not_Supported Message

The ***Not_Supported*** Message ***Shall*** be sent by a Port or Cable Plug in response to any Message it does not support. Returning a ***Not_Supported*** Message is assumed in this specification and has not been called out explicitly except in Section 6.12 which defines cases where the ***Not_Supported*** Message is returned.

6.3.17 Get_Source_Cap_Extended Message

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

6.3.18 Get_Status Message

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

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

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

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

6.3.19 FR_Swap Message

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

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

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

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

Prior to the Fast Role Swap AMS the new Source ***Shall*** have Rd asserted on the CC wire and the new Sink ***Shall*** have Rp asserted on the CC wire. Note that this is an incorrect assignment of Rp/Rd (since Rp follows the Source and Rd follows the Sink as defined in ***[USB Type-C 2.0]***) that is corrected by the Fast Role Swap AMS.

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

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

Note: A Fast Role Swap is a "best effort" solution to a situation where a PDUSB Device has lost its external power. This process can occur at any time, even during a Non-interruptible AMS in which case error handling such as Hard Reset or ***[USB Type-C 2.0]*** Error Recovery will be triggered.

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

For more information regarding the Fast Role Swap process, refer to Section 7.1.13 and Section 7.2.10 in the Power Supply chapter, Section 8.3.3.18.5 and Section 8.3.3.18.6 in the Device Policy chapter and Section 9.1.2 for V_{BUS} mapping to USB states.

6.3.20 Get_PPS_Status

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

6.3.21 Get_Country_Codes

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

6.3.22 Get_Sink_Cap_Extended Message

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

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.
- Battery Status Data Object (BSDO) used to convey Battery status information.
- Alert Data Object (ADO) used to indicates events occurring on the Source or Sink.

The type of Data Object being used in a Data Message is defined by the Message Header's ***Message Type*** field and is summarized in Table 6-6. The Sent by column indicates entities which ***May*** send the given Message (Source, Sink or Cable Plug); entities not listed ***Shall Not*** issue the corresponding Message. The Valid Start of Packet column indicates the Messages which ***Shall*** only be issued in SOP Packets and the Messages which ***May*** be issued in SOP* Packets.

Table 6-6 Data Message Types

Bits 4…0	Type	Sent by	Description	Valid Start of Packet
0 0000	***Reserved***		All values not explicitly defined are ***Reserved*** and ***Shall Not*** be used.	
0 0001	***Source_Capabilities***	Source or Dual-Role Power	See Section 6.4.1.2	SOP only
0 0010	***Request***	Sink only	See Section 6.4.2	SOP only

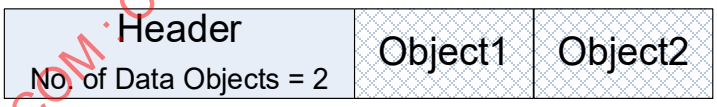
Bits 4...0	Type	Sent by	Description	Valid Start of Packet
0 0011	***BIST***	Tester, Source or Sink	See Section 6.4.3	SOP*
0 0100	***Sink_Capabilities***	Sink or Dual-Role Power	See Section 6.4.1.3	SOP only
0 0101	***Battery_Status***	Source or Sink	See Section 6.4.5	SOP only
0 0110	***Alert***	Source or Sink	See Section 6.4.6	SOP only
0 0111	***Get_Country_Info***	Source or Sink	See Section 6.4.7	SOP only
0 1000	***Enter_USB***	DFP	See Section 6.4.8	SOP*
0 1001 -0 1110	***Reserved***		All values not explicitly defined are ***Reserved*** and ***Shall Not*** be used.	
0 1111	***Vendor_Defined***	Source, Sink or Cable Plug	See Section 6.4.4	SOP*
1 0000-1 1111	***Reserved***		All values not explicitly defined are ***Reserved*** and ***Shall Not*** be used.	

6.4.1 Capabilities Message

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

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

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



In Figure 6-12, the ***Number of Data Objects*** field is 2: ***vSafe5V*** plus one other voltage.

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

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

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

There is one type of Augmented Power Data Object:

- Programmable Power Supply is used to expose a power supply whose output voltage can be programmatically adjusted over the advertised voltage range.

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

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

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

The Source and Sink ***Shall Not*** negotiate a power level that would allow the current to exceed the maximum current supported by their receptacles or the Attached plug (see ***[USB Type-C 2.0]***). The Source ***Shall*** limit its offered capabilities to the maximum current supported by its receptacle and Attached plug. A Sink ***Shall*** only make a request from any of the capabilities offered by the Source. For further details see Section 4.4.

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 6-7).

Table 6-7 Power Data Object

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

The Augmented Power Data Object (APDO) is defined to allow support for more than the four PDO types by extending the Power Data Object field from 2 to 4 bits when the B31…B30 are 11b. The generic APDO structure is shown in Table 6-8.

Table 6-8 Augmented Power Data Object

Bit(s)	Description
B31…30	11b – Augmented Power Datat Object (APDO)
B29…28	00b – Programmable Power Supply 01b-11b - ***Reserved***
B27…0	Specific Power Capabilities are described by the APDOs in the following sections.

6.4.1.1 Use of the Capabilities Message

6.4.1.1.1 Use by Sources

Sources send a ***Source_Capabilities*** Message (see Section 6.4.1) 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 by a Port in the following cases:

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

6.4.1.1.2 Use by Sinks

Sinks send a ***Sink_Capabilities*** Message (see Section 6.4.1.3) 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 6-7) as part of a ***Source_Capabilities*** Message (see Figure 6-12). Power Data Objects are used to convey a Source Port's capabilities to provide power including Dual-Role Power ports presently operating as a Sink.

Each Power Data Object ***Shall*** describe a specific Source capability such as a Battery (e.g. 2.8-4.1V) or a fixed power supply (e.g. 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.

Sinks with Accessory Support do not source V_{BUS} (see ***[USB Type-C 2.0]***). Sinks with Accessory Support are still considered Sources when sourcing V_{CONN} to an Accessory even though V_{BUS} is not applied; in this case they ***Shall*** advertise ***vSafe5V*** with the Maximum Current set to 0mA in the first Power Data Object. The main purpose of this is to enable the Sink with Accessory Support to get into the ***PE_SRC_Ready*** State in order to enter an Alternate Mode.

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.

A Sink that evaluates the ***Source_Capabilities*** Message it receives and identifies a PPS APDO ***Shall*** periodically re-request the PPS APDO at least every ***tPPSRequest*** until either:

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

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

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

6.4.1.2.1 Management of the Power Reserve

A Power Reserve ***May*** be allocated to a Sink when it makes a request from Source Capabilities which includes a Maximum Operating Current/Power. The size of the Power Reserve for a particular Sink is calculated as the difference between its Maximum Operating Current/Power field and its Operating Current/Power field. For a Hub with multiple ports this same Power Reserve ***May*** be shared between

several Sinks. The Power Reserve ***May*** also be temporarily used by a Sink which has indicated it can give back power by setting the GiveBack flag.

Where a Power Reserve has been allocated to a Sink the Source ***Shall*** indicate the Power Reserve as part of every ***Source_Capabilities*** Message it sends. When the same Power Reserve is shared between several Sinks the Source ***Shall*** indicate the Power Reserve as part of every ***Source_Capabilities*** Message it sends to every Sink. Every time a Source sends capabilities including the Power Reserve capability and then accepts a request from a Sink including the Power Reserve indicated by its Maximum Operating Current/Power it is confirming that the Power Reserve is part of the Explicit Contract with the Sink.

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

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

6.4.1.2.2 Fixed Supply Power Data Object

Table 6-9 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 6-9 Fixed Supply PDO - Source

Bit(s)	Description
B31…30	Fixed supply
B29	Dual-Role Power
B28	USB Suspend Supported
B27	Unconstrained Power
B26	USB Communications Capable
B25	Dual-Role Data
B24	Unchunked Extended Messages Supported
B23…22	***Reserved*** – ***Shall*** be set to zero.
B21…20	Peak Current
B19…10	Voltage in 50mV units
B9…0	Maximum Current in 10mA units

6.4.1.2.2.1 Dual-Role Power

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

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

zero in the ***Source_Capabilities*** Message the Dual-Role Power bit in the ***Sink_Capabilities*** Message ***Shall*** also be set to zero.

6.4.1.2.2.2 USB Suspend Supported

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

- If the USB Suspend Supported flag is set, then the Sink ***Shall*** follow the ***[USB 2.0]*** or ***[USB 3.2]*** rules for suspend and resume. A PDUSB Peripheral ***May*** draw up to ***pSnkSusp*** during suspend; a PDUSB Hub ***May*** draw up to ***pHubSusp*** during suspend (see Section 7.2.3).
- If the USB Suspend Supported flag is cleared, then the Sink ***Shall Not*** apply the ***[USB 2.0]*** or ***[USB 3.2]*** 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).

6.4.1.2.2.3 Unconstrained Power

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

To set the Unconstrained Power bit 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.

6.4.1.2.2.4 USB Communications Capable

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

6.4.1.2.2.5 Dual-Role Data

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

6.4.1.2.2.6 Unchunked Extended Messages Supported

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

6.4.1.2.2.7 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 6-10). Supplies that do not support an overload capability ***Shall*** set these bits to 00b in the corresponding Fixed Supply PDO. Supplies that support an extended overload capability specified in the PeakCurrent1…3 fields of the ***Source_Capabilities_Extended*** Message (see Section 6.5.1) ***Shall*** also set these bits to 00b. Sinks wishing to utilize these extended capabilities ***Shall*** first send the ***Get_Source_Cap_Extended*** Message to determine what capabilities, if any are supported by the Source.

Table 6-10 Fixed Power Source Peak Current Capability

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

6.4.1.2.3 Variable Supply (non-Battery) Power Data Object

Table 6-11 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 6-11 Variable Supply (non-Battery) PDO - Source

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

6.4.1.2.4 Battery Supply Power Data Object

Table 6-12 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 6-12 Battery Supply PDO - Source

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

6.4.1.2.5 Programmable Power Supply Augmented Power Data Object

Table 6-13 below describes a Programmable Power Supply (1100b) APDO for a Source. See Section 7.1.3 for the electrical requirements of the power supply. This APDO is used primarily for Sink Directed Charge of a Battery in the Sink. When applying a current to the Battery greater than the cable supports, a high efficiency fixed scaler ***May*** be used in the Sink to reduce the cable current.

The voltage fields define the output voltage range over which the power supply ***Shall*** be adjustable in 20mV steps. The Maximum Current field contains the current the Programmable Power Supply ***Shall*** be capable of delivering over the advertised voltage range.

Table 6-13 Programmable Power Supply APDO - Source

Bit(s)	Description
B31…30	11b – Augmented Power Data Object (APDO)
B29…28	00b – Programmable Power Supply 01b…11b - ***Reserved***, ***Shall Not*** be used
B27	PPS Power Limited
B26…25	***Reserved*** – ***Shall*** be set to zero
B24…17	Maximum Voltage in 100mV increments
B16	***Reserved*** – ***Shall*** be set to zero
B15…8	Minimum Voltage in 100mV increments
B7	***Reserved*** – ***Shall*** be set to zero
B6…0	Maximum Current in 50mA increments

6.4.1.2.5.1 PPS Power Limited

When the PPS Power Limited bit is set, the PPS Source ***Shall Not*** supply power that exceeds the Source's rated PDP; if the requested Output Voltage in the RDO exceeds the nominal Prog voltage (e.g. 5V for the 5VProg), the PPS ***Shall*** limit its output current such that for a PDP Rating of x Watts the output current limit is calculated as RoundDown(x/requested Output Voltage) to the nearest 50mA. . The PPS Source ***Shall Not*** reject an RDO with an Output Current that is less than or equal to the Maximum Current in the APDO even if the requested Output Current is greater than the Source's current limit. A PPS Source that sets the Power Output Limited bit ***Shall*** automatically limit its output current so as not to exceed its PDP Rating (See Figure 7-7).

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

6.4.1.3 Sink Capabilities Message

A Sink Port ***Shall*** report power levels it is able to operate at in a series of 32-bit Power Data Objects (see Table 6-7). These are returned as part of a ***Sink_Capabilities*** Message in response to a ***Get_Sink_Cap*** Message (see Figure 6-12). 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 6-14 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 6-14 Fixed Supply PDO - Sink

Bit(s)	Description
B31…30	Fixed supply
B29	Dual-Role Power
B28	Higher Capability
B27	Unconstrained Power
B26	USB Communications Capable
B25	Dual-Role Data
B24…23	Fast Role Swap required USB Type-C Current (see also ***[USB Type-C 2.0]***): **Value** / **Description** 00b – Fast Swap not supported (default) 01b – Default USB Power 10b – 1.5A @ 5V 11b – 3.0A @ 5V
B22…20	***Reserved*** – ***Shall*** be set to zero.
B19…10	Voltage in 50mV units
B9…0	Operational Current in 10mA units

6.4.1.3.1.1 Dual-Role Power

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

6.4.1.3.1.2 Higher Capability

In the case that the Sink needs more than ***vSafe5V*** (e.g. 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 Dual-Role data capable i.e. it supports the ***DR_Swap*** Message. This is a static capability which ***Shall*** remain fixed for a given device regardless of the device's present power role or data role. If the Dual-Role Data bit is set to one in the ***Source_Capabilities*** Message the Dual-Role Data bit in the ***Sink_Capabilities*** Message ***Shall*** also be set to one. If the Dual-Role Data bit is set to zero in the ***Source_Capabilities*** Message the Dual-Role Data bit in the ***Sink_Capabilities*** Message ***Shall*** also be set to zero.

6.4.1.3.1.6 Fast Role Swap USB Type-C Current

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

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

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

6.4.1.3.2 Variable Supply (non-Battery) Power Data Object

Table 6-15 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 6-15 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 6-16 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 6-16 Battery Supply PDO - Sink

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

6.4.1.3.4 Programmable Power Supply Augmented Power Data Object

Table 6-17 below describes a Programmable Power Supply (1100b) APDO used by a Sink. See Section 7.1.3 for the electrical requirements of the power supply.

The Maximum and Minimum Voltage fields ***Shall*** be set to the output voltage range that the Sink requires to operate. The Operational Current field ***Shall*** be set to the maximum current the Sink requires over the voltage range. The Operating Current is defined as the maximum amount of current the device needs to fully support its function (e.g., Sink Directed Charge).

Table 6-17 Programmable Power Supply APDO - Sink

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

6.4.2 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). 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 a different form depending on the kind of power requested. The Fixed Power Data Object and Variable Power Data Object share a common format shown in Table 6-18 and Table 6-19. The Battery Power Data Object uses the format shown in Table 6-20 and Table 6-21. The Programmable Request Object the format shown in Table 6-22.

Table 6-18 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	Unchunked Extended Messages Supported
B22…20	***Reserved*** - ***Shall*** be set to zero.
B19…10	Operating current in 10mA units
B9…0	Maximum Operating Current 10mA units

Table 6-19 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	Unchunked Extended Messages Supported
B22…20	***Reserved*** - ***Shall*** be set to zero.
B19…10	Operating Current in 10mA units
B9…0	Minimum Operating Current 10mA units

Table 6-20 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	GiveBackFlag = 0
B26	Capability Mismatch
B25	USB Communications Capable
B24	No USB Suspend
B23	Unchunked Extended Messages Supported
B22…20	***Reserved*** - ***Shall*** be set to zero.
B19…10	Operating Power in 250mW units
B9…0	Maximum Operating Power in 250mW units

Table 6-21 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	GiveBackFlag = 1
B26	Capability Mismatch
B25	USB Communications Capable
B24	No USB Suspend
B23	Unchunked Extended Messages Supported
B22…20	***Reserved*** - ***Shall*** be set to zero.
B19…10	Operating Power in 250mW units
B9…0	Minimum Operating Power in 250mW units

Table 6-22 Programmable 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	***Reserved*** – ***Shall*** be set to zero
B26	Capability Mismatch
B25	USB Communications Capable
B24	No USB Suspend
B23	Unchunked Extended Messages Supported
B22…20	***Reserved*** - ***Shall*** be set to zero.
B19…9	Output Voltage in 20mV units
B8…7	***Reserved*** - ***Shall*** be set to zero.
B6…0	Operating Current 50mA units

6.4.2.1 Object Position

The value in the Object Position field ***Shall*** indicate which object in the ***Source_Capabilities*** Message 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 3.2]*** 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 3.2]*** protocols. This is used by the Source to determine operation in certain cases such as USB suspend. If the USB Communications Capable flag has been set to zero by a Sink, then the Source needs to be aware that USB Suspend rules cannot be observed by the Sink.

6.4.2.5 No USB Suspend

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

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

6.4.2.6 Unchunked Extended Messages Supported

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

6.4.2.7 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.

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

The value in the Operating Current field ***Shall Not*** exceed the value in the Maximum Current field.

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

6.4.2.8 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.

6.4.2.9 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.

6.4.2.10 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.

6.4.2.11 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.

6.4.2.12 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 to intelligently distribute power.

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

6.4.2.13 Output Voltage

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

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

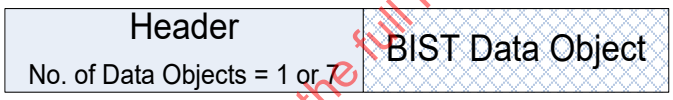
6.4.3 BIST Message

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

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

The Message format is as follows:

Figure 6-13 BIST Message



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

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

UUTs with Ports constituting a shared capacity group (see ***[USB Type-C 2.0]***) ***Shall*** support the following BIST Mode:

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

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

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

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

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

In Section 8.3.2.14 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 6-23.

Table 6-23 BIST Data Object

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

6.4.3.1 BIST Carrier Mode

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

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

6.4.3.2 BIST Test Data

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

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

6.4.3.3 BIST Shared Capacity Test Mode

A shared capacity group of Ports share a common power source that is not capable of simultaneously powering all the ports to their full Source Capabilities (see ***[USB Type-C 2.0]***). The BIST Shared Capacity Test Mode ***Shall*** only be implemented by ports in a shared capacity group.

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

6.4.3.3.1 BIST Shared Test Mode Entry

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

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

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

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

6.4.3.3.2 BIST Shared Test Mode Exit

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

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

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

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

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

6.4.4 Vendor Defined Message

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

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

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

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

Vendor_Defined Messages ***Shall Not*** be used for direct power negotiation. They ***May*** however be used to alter Local Policy, affecting what is offered or consumed via the normal PD Messages. 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 6-14.

Figure 6-14 Vendor Defined Message

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

The fields in the VDM Header for an Unstructured VDM, when the VDM Type Bit is set to zero, ***Shall*** be as defined in Table 6-24. 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 6-25.

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 or Fast Role Swap) is in place in order to discover Cable capabilities (see Section 8.3.3.24.3). 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.
- 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 “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.
- When a DFP or UFP does not support Unstructured VDMs or does not recognize the VID it ***Shall*** return a ***Not_Supported*** Message.

Table 6-24 illustrates the VDM Header bits.

Table 6-24 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 6-25.

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 8.3.3.24.3).
- Either Port ***May*** be an Initiator of Structured VDMs except for the ***Enter Mode*** and ***Exit Mode*** Commands which ***Shall*** only be initiated by the DFP.
- A Cable Plug ***Shall*** only be a Responder to Structured VDMs.
- Structured VDMs ***Shall Not*** be initiated or responded to under any other circumstances.
- When a DFP or UFP does not support Structured VDMs any Structured VDMs received ***Shall*** return a ***Not_Supported*** Message.
- When 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 6-25 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): • Version 1.0 = 00b (***Shall Not*** be used) • Version 2.0 = 01b • Values 2-3 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.
B10…8	Object Position	For the ***Enter Mode, Exit Mode*** and ***Attention*** Commands (Requests/Responses): • 000b = ***Reserved*** and ***Shall Not*** be used. • 001b…110b = Index into the list of VDOs to identify the desired Mode VDO • 111b = Exit all Active Modes (equivalent of a power on reset). ***Shall*** only be used with the ***Exit Mode*** Command. Commands 0…3, 7…15: • 000b • 001b…111b = ***Reserved*** and ***Shall Not*** be used. SVID Specific Commands (16…31) defined by the SVID.
B7…6	Command Type	00b = REQ (Request from Initiator Port) 01b = ACK (Acknowledge Response from Responder Port) 10b = NAK (Negative Acknowledge Response from Responder Port) 11b = BUSY (Busy Response from Responder Port)
B5	***Reserved***	***Shall*** be set to 0 and ***Shall*** be ***Ignored***

Bit(s)	Field	Description
B4…0	Command[1]	0 = ***Reserved***, ***Shall Not*** be used 1 = ***Discover Identity*** 2 = ***Discover SVIDs*** 3 = ***Discover Modes*** 4 = ***Enter Mode*** 5 = ***Exit Mode*** 6 = ***Attention*** 7-15 = ***Reserved***, ***Shall Not*** be used 16…31 = SVID Specific Commands
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 6-26 shows the Commands, which SVID to use with each Command and the ***SOP**** values which ***Shall*** be used.

Table 6-26 Structured VDM Commands

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

6.4.4.2.1 SVID

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

Table 6-27 lists specific SVID values referenced by this specification.

Table 6-27 SVID Values

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

6.4.4.2.2 VDM Type

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

6.4.4.2.3 Structured VDM Version

The Structured VDM Version 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. This field ***Shall*** be set to 01b to indicate Version 2.0.

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

On receipt of a VDM Header with a higher Version number than that supported, a Port ***Shall*** respond using the highest Version number it supports.

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

Detached. After discovering the Structure VDM Version, the Structured VDM Version field ***Shall*** match the agreed common Structured VDM Version.

6.4.4.2.4 Object Position

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

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

6.4.4.2.5 Command Type

6.4.4.2.5.1 Commands other than Attention

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

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

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

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

6.4.4.2.5.2 Attention Command

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

6.4.4.2.6 Command

6.4.4.2.6.1 Commands other than Attention

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

A Structured VDM Command consists of a Command request and a Command response (ACK, NAK or BUSY). A Structured VDM Command is deemed to be completed (and if applicable, the transition to the requested functionality is made) when the ***GoodCRC*** Message has been successfully received by the Responder in reply to its Command response.

If Structured VDMs are supported, but the Structured VDM Command request is an Unrecognized Message, it ***Shall*** be NAKed (see Table 6-28).

6.4.4.2.6.2 Attention Command

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

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

If Structured VDMs are supported, but the Structured VDM ***Attention*** Command request is an Unrecognized Message it ***Shall*** be ***Ignored*** (see Table 6-28).

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 6-28 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 6-28 Commands and Responses

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

Examples of Command usage can be found in Appendix 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 (VCONN Source) to identify the Responder (Cable Plug). The ***Discover 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*** only be sent to ***SOP*** when there is an Explicit Contract.

The ***Discover Identity*** Command ***Shall*** be used to determine whether a given Cable Plug is PD Capable (see Section 8.3.3.20.1 and Section 8.3.3.24.3). In this case a ***Discover Identity*** Command request sent to SOP' ***Shall Not*** cause a Soft Reset if a ***GoodCRC*** Message response is not returned since this can indicate a

non-PD Capable cable. Note that a Cable Plug will not be ready for PD Communication until tVCONNStable after VCONN has been applied (see *[USB Type-C 2.0]*). During Cable Plug discovery, when there is an Explicit Contract, ***Discover Identity*** Commands are sent at a rate defined by the ***DiscoverIdentityTimer*** (see Section 6.6.15) up to a maximum of ***nDiscoverIdentityCount*** times (see Section 6.7.5).

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

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

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

The SVID in the ***Discover Identity*** Command request ***Shall*** be set to the ***PD SID*** (see Table 6-27).

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

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

- Passive Cable VDO (see Section 6.4.4.3.1.6)
- Active Cable VDOs (see Section 6.4.4.3.1.7)
- Alternate Mode Adapter VDO (see Section 6.4.4.3.1.8)
- VCONN Powered USB Device VDO (see Section 6.4.4.3.1.9)
- UFP VDO (see Section 6.4.4.3.1.4)
- DFP 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 6-15 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.

If the product is a DRD both a Product Type (UFP) and a Product Type (DFP) are declared in the ID Header. These products ***Shall*** set both the PDUSB Host and PDUSB Peripheral bits in the ID Header and ***Shall*** return Product Type VDOs for both the PDUSB Peripheral and PDUSB Host beginning with the UFP VDOs followed by the DFP VDO as shown in Figure 6-16.

Figure 6-16 Discover Identity Command response for a DRD

Header No. of Data Objects = 7	VDM Header	ID Header VDO	Cert Stat VDO	Product VDO	Product Type VDO(s)		
					UFP1	UFP2	DFP

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 6-29.

Table 6-29 ID Header VDO

Bit(s)	Description	Reference
B31	USB Communications Capable as USB Host: • ***Shall*** be set to one if the product is capable of enumerating USB Devices. • ***Shall*** be set to zero otherwise	Section 6.4.4.3.1.1.1
B30	USB Communications Capable as a USB Device: • ***Shall*** be set to one if the product is capable of being enumerated as a USB Device. • ***Shall*** be set to zero otherwise	Section 6.4.4.3.1.1.2
B29…27	Product Type (UFP): • 000b – Undefined • 001b – PDUSB Hub • 010b – PDUSB Peripheral • 011b – PSD • 100b – ***Reserved***, ***Shall Not*** be used. • 101b – Alternate Mode Adapter (AMA) • 110b – VCONN-Powered USB Device (VPD) • 111b – ***Reserved***, ***Shall Not*** be used. Product Type (Cable Plug): • 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: • ***Shall*** be set to one if the product supports Modal Operation (Alternate Modes). • ***Shall*** be set to zero otherwise	Section 6.4.4.3.1.1.4
B25…23	Product Type (DFP): • 000b – Undefined • 001b – PDUSB Hub • 010b – PDUSB Host • 011b – Power Brick • 100b - Alternate Mode Controller (AMC) • 101b…111b – ***Reserved***, ***Shall Not*** be used.	
B22…16	***Reserved***. ***Shall*** be set to zero.	
B15…0	USB Vendor ID.	*[USB 2.0]*/*[USB 3.2]*/*[USB4]*

6.4.4.3.1.1.1 USB Communications Capable as a USB Host

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

6.4.4.3.1.1.2 USB Communications Capable as a USB Device

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

6.4.4.3.1.1.3 Product Type (UFP)

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

indicate the Product Type when in UFP role regardless of the present Data Role. Table 6-30 defines the Product Type VDOs which ***Shall*** be returned.

Table 6-30 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.	UFP VDO	Section 6.4.4.3.1.4
PDUSB Peripheral	***Shall*** be used when the Product is a PDUSB Device other than a PDUSB Hub.	UFP VDO	Section 6.4.4.3.1.4
PSD	***Shall*** be used when the Product is a PSD, e.g. power bank.	None	
Alternate Mode Adapter	***Shall*** be used when the Product is a PDUSB Device that supports one or more Alternate Modes.	AMA VDO	Section 6.4.4.3.1.8
VCONN Powered USB Device	***Shall*** be used when the Product is a PDUSB VCONN Powered USB Device.	VPD VDO	Section 6.4.4.3.1.9

6.4.4.3.1.4 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 6-31 defines the Product Type VDOs which ***Shall*** be returned.

Table 6-31 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 6.4.4.3.1.7
Passive Cable	***Shall*** be used when the Product is a cable that does not incorporate signal conditioning circuits.	Passive Cable VDO	Section 6.4.4.3.1.6

6.4.4.3.1.5 Modal Operation Supported

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

6.4.4.3.1.6 Product Type (DFP)

The Product Type (DFP) field indicates the type of Product when in DFP Data Role, whether a VDO will be returned and if so the type of VDO to be returned. The Product Type indicated in the Product Type (DFP) field ***Shall*** be the closest categorization of the main functionality of the Product in DFP Data Role or "Undefined" when there is no suitable category for the product. For DRD Products this field ***Shall*** always indicate the Product Type when in DFP role regardless of the present Data Role. Table 6-32 defines the Product Type VDOs which ***Shall*** be returned.

Table 6-32 Product Types (DFP)

Product Type	Description	Product Type VDO	Reference
Undefined	***Shall*** be used where no other Product Type value is appropriate.	None	
PDUSB Hub	***Shall*** be used when the Product is a PDUSB Hub.	DFP VDO	Section 6.4.4.3.1.5
PDUSB Host	***Shall*** be used when the Product is a PDUSB Host.	DFP VDO	Section 6.4.4.3.1.5
Power Brick	***Shall*** be used when the Product is a Power Brick/Wall Wart.	DFP VDO	Section 6.4.4.3.1.5

Product Type	Description	Product Type VDO	Reference
Alternate Mode Controller	***Shall*** be used when the Product is a PDUSB Host or DFP that supports one or more Alternate Modes.	None	

6.4.4.3.1.1.7 Vendor ID

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

6.4.4.3.1.2 Cert Stat VDO

The Cert Stat VDO ***Shall*** contain the XID assigned by USB-IF to the product before certification in binary format. The fields in the Cert Stat VDO ***Shall*** be as defined in Table 6-33.

Table 6-33 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 6-34.

Table 6-34 Product VDO

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

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

6.4.4.3.1.4 UFP VDOs

The UFP VDOs defined in this section ***Shall*** be returned by Ports capable of operating as a UFP including traditional USB peripherals, USB hub's upstream Port and DRD capable host Ports. The UFP VDOs defined in this section ***Shall*** be sent when the Product Type (UFP) field in the ID Header VDO is given as a PDUSB Peripheral or PDUSB Hub. Table 6-35 and 6-36 define the UFP VDOs that ***Shall*** be sent based on the Product Type.

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

Table 6-35 UFP VDO 1

Bit(s)	Field	Description
B31…29	UFP VDO Version	Version Number of the VDO (not this specification Version): • Version 1.0 = 000b Values 001b…111b are ***Reserved*** and ***Shall Not*** be used
B28	***Reserved***	***Shall*** be set to zero.
B27…24	Device Capability	**Bit** / **Description** 0 – *[USB 2.0]* Device Capable 1 – *[USB 2.0]* Device Capable (Billboard only) 2 – *[USB 3.2]* Device Capable 3 – *[USB4]* Device Capable
B23…6	***Reserved***	***Shall*** be set to zero.
B5…3	Alternate Modes	**Bit** / **Description** 0 – Supports *[TBT3]* Alternate Mode 1 – Supports Alternate Modes that reconfigure the signals on the *[USB Type-C 2.0]* connector – except for *[TBT3]*. 2 – Supports Alternate Modes that do not reconfigure the signals on the *[USB Type-C 2.0]* connector
B2…0	USB Highest Speed	000b = *[USB 2.0]* only, no SuperSpeed support 001b = *[USB 3.2]* Gen1 010b = *[USB 3.2]*/*[USB4]* Gen2 011b = *[USB4]* Gen3 100b…111b = ***Reserved***, ***Shall Not*** be used

6.4.4.3.1.4.1 VDO Version Field

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

6.4.4.3.1.4.2 Device Capability Field

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

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

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

6.4.4.3.1.4.3 Alternate Modes Field

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

6.4.4.3.1.4.4 USB Highest Speed Field

The USB Highest Speed field Shall indicate the port's highest signaling capability.

Table 6-36 UFP VDO 2

Bit(s)	Field	Description
B31…30	***Reserved***	***Shall*** be set to zero.
B29…23	USB4 Min Power	Minimum power in watts required to function in ***[USB4]*** operation.
B22…16	USB4 Max Power	Power in watts required for full functionality excluding any power required for battery charging or for redistribution in ***[USB4]*** operation.
B15…14	***Reserved***	***Shall*** be set to zero.
B13…7	USB3 Min Power	Minimum power in watts required to function in ***[USB 3.2]*** operation.
B6…0	USB3 Max Power	Power in watts required for full functionality excluding any power required for battery charging or for redistribution in ***[USB 3.2]*** operation.

6.4.4.3.1.4.5 USB4 Min Power Field

The USB4 Min Power field ***Shall*** contain the minimum amount of power, rounded-up the next integer value, a device operating in ***[USB4]*** needs to function. Minimally at this power level the device can be enumerated and at least one of its functions must operate although this may be at reduced performance.

6.4.4.3.1.4.6 USB4 Max Power Field

The USB4 Max Power field ***Shall*** contain the amount of power, rounded-up the next integer value, a device operating in ***[USB4]*** needs to be fully functional at maximum performance. However, this does not include any additional power required for charging a battery or for redistribution such as using some of the power to supply power to another port on a hub.

6.4.4.3.1.4.7 USB3 Min Power Field

The USB3 Min Power field ***Shall*** contain the minimum amount of power rounded-up the next integer value a device operating in ***[USB 3.2]*** needs to function. Minimally at this power level the device can be enumerated and at least one of its functions must operate although this may be at reduced performance.

6.4.4.3.1.4.8 USB3 Max Power Field

The USB3 Max Power field ***Shall*** contain the amount of power a device, rounded-up the next integer value, operating in ***[USB 3.2]*** needs to be fully functional at maximum performance. However, this does not include any additional power required for charging a battery or for redistribution such as using some of the power to supply power to another port on a hub.

6.4.4.3.1.5 DFP VDO

The DFP VDO ***Shall*** be returned by Ports capable of operating as a DFP; including those implemented by Hosts, Hubs and Power Bricks. The DFP VDO ***Shall*** be returned when the Product Type (DFP) field in the ID Header VDO is given as Power Brick, PDUSB Host or PDUSB Hub. Table 6-37 defines the DFP VDO that ***Shall*** be sent.

Table 6-37 DFP VDO

Bit(s)	Field	Description
B31…29	DFP VDO Version	Version Number of the VDO (not this specification Version): • Version 1.0 = 000b Values 001b…111b are ***Reserved*** and ***Shall Not*** be used
B28…27	***Reserved***	***Shall*** be set to zero.
B26…24	Host Capability	**Bit** / **Description** 0 / ***[USB 2.0]*** Host Capable 1 / ***[USB 3.2]*** Host Capable 2 / ***[USB4]*** Host Capable
B23…5	***Reserved***	***Shall*** be set to zero.
B4…0	Port Number	Unique port number to identify a specific port on a multi-port device.

6.4.4.3.1.5.1 VDO Version Field

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

6.4.4.3.1.5.2 Host Capability Field

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

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

6.4.4.3.1.5.3 Port Number Field

The Port Number field ***Shall*** be a static unique number that unambiguously identifies each ***[USB Type-C 2.0]*** DFP, including DRPs, on the device. Note that this number is independent of the USB port number.

6.4.4.3.1.6 Passive Cable VDO

The Passive Cable VDO defined in this section ***Shall*** be sent when the Product Type is given as Passive Cable. Table 6-38 defines the Cable VDO which ***Shall*** be sent.

A Passive Cable has a USB Plug on each end at least one of which is a Cable Plug supporting SOP' Communication. A Passive Cable ***Shall Not*** incorporate data bus signal conditioning circuits and hence has no concept of Super Speed Directionality. A Passive Cable ***Shall*** include a V_{BUS} wire and ***Shall*** only respond to SOP' Communication. Passive Cables ***Shall*** support the Structured VDM ***Discover Identity*** Command and ***Shall*** return the Passive Cable VDO in a ***Discover Identity*** Command ACK as shown in Table 6-38.

Table 6-38 Passive Cable VDO

Bit(s)	Field	Description
B31…28	HW Version	0000b…1111b assigned by the VID owner
B27…24	Firmware Version	0000b…1111b assigned by the VID owner
B23…21	VDO Version	Version Number of the VDO (not this specification Version): • Version 1.0 = 000b Values 001b…111b are ***Reserved*** and ***Shall Not*** be used
B20	***Reserved***	***Shall*** be set to zero.
B19…18	USB Type-C plug to USB Type-C/Captive	00b = ***Reserved***, ***Shall Not*** be used 01b = ***Reserved***, ***Shall Not*** be used 10b = USB Type-C 11b = Captive
B17	***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) 1001b ….1111b ***Reserved***, ***Shall Not*** be used Includes latency of electronics in Active Cable
B12…11	Cable Termination Type	00b = VCONN not required. Cable Plugs that only support ***Discover Identity*** Commands ***Shall*** set these bits to 00b. 01b = VCONN required 10b…11b = ***Reserved***, ***Shall Not*** be used
B10…9	Maximum V_{BUS} Voltage	Maximum Cable VBUS Voltage: 00b – 20V 01b – 30V 10b – 40V 11b – 50V
B8…7	***Reserved***	***Shall*** be set to zero.
B6…5	V_{BUS} Current Handling Capability	00b = ***Reserved***, ***Shall Not*** be used. 01b = 3A 10b = 5A 11b = ***Reserved***, ***Shall Not*** be used.
B4…3	***Reserved***	***Shall*** be set to zero.
B2…0	USB Highest Speed	000b = ***[USB 2.0]*** only, no SuperSpeed support 001b = ***[USB 3.2]*** Gen1 010b = ***[USB 3.2]***/***[USB4]*** Gen2 011b = ***[USB4]*** Gen3 100b…111b = ***Reserved***, ***Shall Not*** be used

6.4.4.3.1.6.1 HW Version Field

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

6.4.4.3.1.6.2 FW Version Field

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

6.4.4.3.1.6.3 VDO Version Field

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

6.4.4.3.1.6.4 Connector Type Field

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.

6.4.4.3.1.6.5 Cable Latency Field

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

6.4.4.3.1.6.6 Cable Termination Type Field

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.

6.4.4.3.1.6.7 Maximum VBUS Voltage Field

The Maximum V_{BUS} Voltage field (B10…9) ***Shall*** contain the maximum voltage that ***Shall*** be negotiated using a Fixed Supply over the cable as part of an Explicit Contract where the maximum voltage that ***Shall*** be applied to the cable is ***vSrcNew*** max + ***vSrcValid*** max. For example, when the Maximum V_{BUS} Voltage field is 20V, a Fixed Supply of 20V can be negotiated as part of an Explicit Contract where the absolute maximum voltage that can be applied to the cable is 21.55V.

6.4.4.3.1.6.8 VBUS Current Handling Capability Field

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

6.4.4.3.1.6.9 USB Highest Speed Field

The USB Highest Speed field (B2…0) ***Shall*** indicate the highest signaling rate the cable supports.

6.4.4.3.1.7 Active Cable VDOs

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

An Active Cable:

- ***Shall*** respond to SOP' Communication
- ***May*** respond to SOP'' Communication
- ***Shall*** support the Structured VDM ***Discover Identity*** Command
- In the ***Discover Identity*** Command ACK:
  - ***Shall*** set the Product Type in the ID Header VDO to Active Cable
  - ***Shall*** return the Active Cable VDOs defined in Table 6-39 and Table 6-40

Table 6-39 Active Cable VDO 1

Bit(s)	Field	Description
B31…28	HW Version	0000b…1111b assigned by the VID owner
B27…24	Firmware Version	0000b…1111b assigned by the VID owner
B23…21	VDO Version	Version Number of the VDO (not this specification Version): • Version 1.3 = 011b Values 000b, 100b…111b are ***Reserved*** and ***Shall Not*** be used
B20	***Reserved***	***Shall*** be set to zero.
B19…18	Connector Type	00b = ***Reserved***, ***Shall Not*** be used 01b = ***Reserved***, ***Shall Not*** be used 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) 1011b ….1111b ***Reserved***, ***Shall Not*** be used Includes latency of electronics in Active Cable
B12…11	Cable Termination Type	00b…01b = ***Reserved***, ***Shall Not*** be used 10b = One end Active, one end passive, VCONN required 11b = Both ends Active, VCONN required
B10…9	Maximum V_{BUS} Voltage	Maximum Cable V_{BUS} Voltage: 00b – 20V 01b – 30V 10b – 40V 11b – 50V
B8	SBU Supported	0 = SBUs connections supported 1 = SBU connections are not supported
B7	SBU Type	When SBU Supported = 1 this bit ***Shall*** be ***Ignored*** When SBU Supported = 0: 0 = SBU is passive 1 = SBU is active
B6…5	V_{BUS} Current Handling Capability	When V_{BUS} Through Cable is “No”, this field ***Shall*** be ***Ignored***. When V_{BUS} Though Cable is “Yes”: 00b = USB Type-C Default Current 01b = 3A 10b = 5A 11b = ***Reserved***, ***Shall Not*** be used.
B4	V_{BUS} Through Cable	0 = No 1 = Yes
B3	SOP” Controller Present	0 = No SOP” controller present 1 = SOP” controller present

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

6.4.4.3.1.7.1 HW Version Field

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

6.4.4.3.1.7.2 FW Version Field

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

6.4.4.3.1.7.3 VDO Version Field

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

6.4.4.3.1.7.4 Connector Type Field

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

6.4.4.3.1.7.5 Cable Latency Field

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

6.4.4.3.1.7.6 Cable Termination Type Field

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

6.4.4.3.1.7.7 Maximum VBUS Voltage Field

The Maximum V_{BUS} Voltage field (B10…9) ***Shall*** contain the maximum voltage that ***Shall*** be negotiated as part of an Explicit Contract where the maximum voltage that ***Shall*** be applied to the cable is ***vSrcNew*** max + ***vSrcValid*** max. When this field is set to 20V, the cable will safely carry a Programmable Power Supply APDO of 20V where the absolute maximum voltage that can be applied to the cable is 21.55V.

6.4.4.3.1.7.8 SBU Supported Field

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

6.4.4.3.1.7.9 SBU Type Field

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

6.4.4.3.1.7.10 VBUS Current Handling Capability Field

The V_{BUS} Current Handling Capability field (B6…5) ***Shall*** indicate whether the cable is capable of carrying default current (500mA USB2, 900mA USB3.2 x1, 1.5A USB3.2 x2), 3A or 5A. The V_{BUS} Current Handling Capability ***Shall*** only be ***Valid*** when the V_{BUS} Through Cable field indicates an end to end V_{BUS} wire.

6.4.4.3.1.7.11 VBUS Through Cable Field

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

6.4.4.3.1.7.12 SOP'' Controller Present Field

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

6.4.4.3.1.7.13 USB Highest Speed Field

The USB Highest Speed field (B2…0) ***Shall*** indicate the highest signaling rate the cable supports.

Table 6-40 Active Cable VDO 2

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

6.4.4.3.1.7.14 Maximum Operating Temperature Field

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

6.4.4.3.1.7.15 Shutdown Temperature Field

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

6.4.4.3.1.7.16 U3/CLd Power Field

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

6.4.4.3.1.7.17 U3 to U0 Transition Mode Field

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

6.4.4.3.1.7.18 Physical Connection Field

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

6.4.4.3.1.7.19 Active element Field

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

6.4.4.3.1.7.20 USB4 Supported Field

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

6.4.4.3.1.7.21 USB 2.0 Hub Hops Consumed field

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

6.4.4.3.1.7.22 USB 2.0 Supported Field

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

6.4.4.3.1.7.23 USB 3.2 Supported Field

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

6.4.4.3.1.7.24 USB Lanes Supported Field

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

6.4.4.3.1.7.25 Optically Isolated Active Cable Field

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

6.4.4.3.1.7.26 USB Gen Field

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

6.4.4.3.1.8 Alternate Mode Adapter VDO

The Alternate Mode Adapter (AMA) VDO defined in this section ***Shall*** be sent when the Product Type is given as Alternate Mode Adapter. Table 6-41 defines the AMA VDO which ***Shall*** be sent.

Table 6-41 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…21	VDO Version	Version Number of the VDO (not this specification Version): • Version 1.0 = 000b Values 001b…111b are ***Reserved*** and ***Shall Not*** be used
B20…8	***Reserved***.	***Shall*** be set to zero.
B7…5	VCONN power	When the VCONN required field is set to "Yes" VCONN 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 VCONN required field is set to "No" ***Reserved***, ***Shall*** be set to zero.
B4	VCONN required	0 = No 1 = Yes
B3	VBUS required	0 = No 1 = Yes
B2…0	USB Highest Speed	000b = ***[USB 2.0]*** only 001b = ***[USB 3.2]*** Gen1 and USB 2.0 010b = ***[USB 3.2]*** Gen1, Gen2 and USB 2.0 011b = ***[USB 2.0]*** billboard only 100b…111b = ***Reserved***, ***Shall Not*** be used

6.4.4.3.1.8.1 HW Version Field

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

6.4.4.3.1.8.2 FW Version Field

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

6.4.4.3.1.8.3 VDO Version Field

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

6.4.4.3.1.8.4 VCONN Required Field

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

The VCONN required field ***Shall*** indicate whether VCONN is needed for the AMA to operate.

6.4.4.3.1.8.5 VBUS Required Field

The VBUS required field ***Shall*** indicate whether VBUS is needed for the AMA to operate.

6.4.4.3.1.8.6 USB Highest Speed Field

The USB Highest Speed field (B2…0) ***Shall*** indicate whether the AMA supports only ***[USB 2.0]*** , or in addition Supports ***[USB 3.2]*** Gen1, or Gen1 and Gen2 or ***[USB 2.0]*** billboard only.

6.4.4.3.1.9 Vconn Powered USB Device VDO

The VCONN Powered USB Device (VPD) VDO defined in this section ***Shall*** be sent when the Product Type is given as VCONN Powered USB Device. Table 6-42 defines the VPD VDO which ***Shall*** be sent.

Table 6-42 VPD VDO

Bit(s)	Field	Description
B31…28	HW Version	0000b…1111b assigned by the VID owner
B27…24	Firmware Version	0000b…1111b assigned by the VID owner
B23…21	VDO Version	Version Number of the VDO (not this specification Version): • Version 1.0 = 000b Values 001b…111b are ***Reserved*** and ***Shall Not*** be used
B20…17	***Reserved***	***Shall*** be set to zero.
B16…15	Maximum VBUS Voltage	Maximum Cable VBUS Voltage: 00b – 20V 01b – 30V 10b – 40V 11b – 50V
B14	Charge Through Current Support	Charge Through Support bit=1b: 0b - 3A capable; 1b - 5A capable Charge Through Support bit = 0b: ***Reserved***, ***Shall*** be set to zero
B14…13	***Reserved***	***Shall*** be set to zero.
B12…7	VBUS Impedance	Charge Through Support bit = 1b: Vbus impedance through the VPD in 2 mΩ increments. Values less than 10 mΩ are Reserved and ***Shall Not*** be used. Charge Through Support bit = 0b: ***Reserved, Shall*** be set to zero
B6…1	Ground Impedance	Charge Through Support bit = 1b: Ground impedance through the VPD in 1 mΩ increments. Values less than 10 mΩ are ***Reserved*** and ***Shall Not*** be used. Charge Through Support bit = 0b: ***Reserved, Shall*** be set to zero
B0	Charge Through Support	1b – the VPD supports Charge Through 0b – the VPD does not support Charge Through

6.4.4.3.1.9.1 HW Version Field

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

6.4.4.3.1.9.2 FW Version Field

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

6.4.4.3.1.9.3 VDO Version Field

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

6.4.4.3.1.9.4 Maximum VBUS Voltage Field

The Maximum VBUS Voltage field (B16…15) ***Shall*** contain the maximum voltage that a Sink ***Shall*** negotiate through the VPD Charge Through port as part of an Explicit Contract. Note: the maximum voltage that will be applied to the cable is ***vSrcNew*** max + ***vSrcValid*** max. For example, when the Maximum VBUS Voltage field is 20V, a Fixed Supply of 20V can be negotiated as part of an Explicit Contract where the absolute maximum voltage that can be applied to the cable is 21.55V.

6.4.4.3.1.9.5 VBUS Impedance Field

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

6.4.4.3.1.9.6 Ground Impedance Field

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

6.4.4.3.1.9.7 Charge Through Field

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

6.4.4.3.2 Discover SVIDs

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

The SVID in the ***Discover SVIDs*** Command ***Shall*** be set to the ***PD SID*** (see Table 6-27) 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 6-43). 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 6-17 shows an example response to the ***Discover SVIDs*** Command request with two VDOs containing three SVIDs. Figure 6-18 shows an example response with two VDOs containing four SVIDs followed by an empty VDO to terminate the response. Figure 6-19 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 6-43 Discover SVIDs Responder VDO

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

Figure 6-17 Example Discover SVIDs response with 3 SVIDs

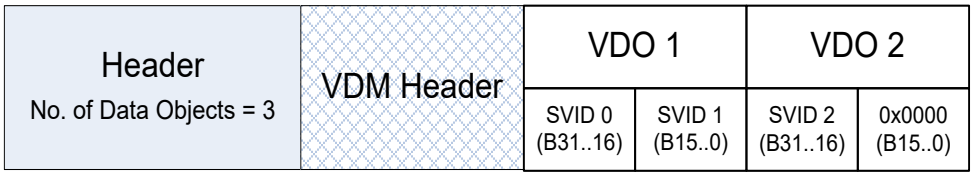


Figure 6-18 Example Discover SVIDs response with 4 SVIDs

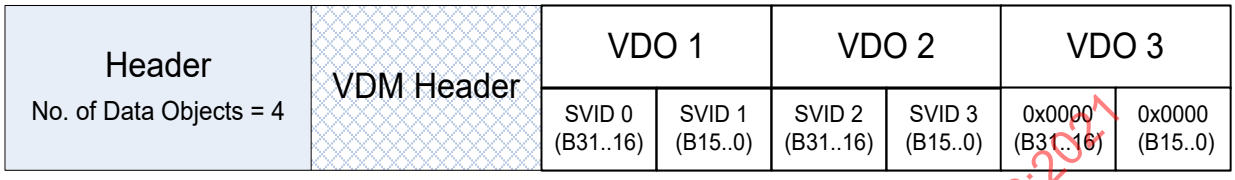
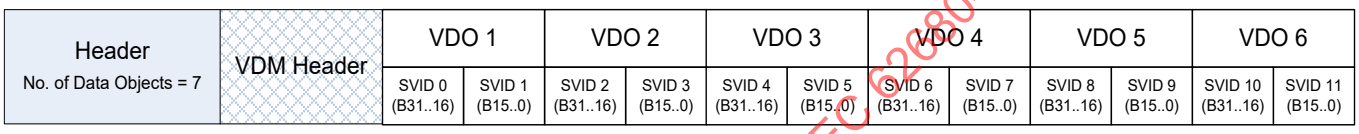


Figure 6-19 Example Discover SVIDs response with 12 SVIDs followed by an empty response



6.4.4.3.3 Discover Modes

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

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

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

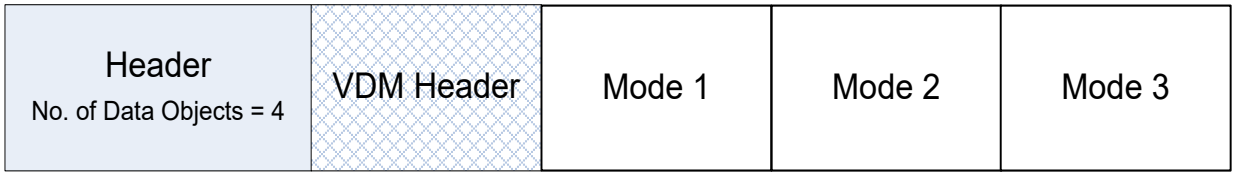
The ***Discover Modes*** Command ACK sent back by the Responder ***Shall*** contain one or more Modes. The ***Discover Modes*** Command ACK ***Shall*** contain a Message Header with the ***Number of Data Objects*** field set to a value of 1 to 7 (the actual value is the number of Mode objects plus one). 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 6-20 shows an example of a ***Discover Modes*** Command response from a Responder which supports three Modes for a given SVID.

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



6.4.4.3.4 Enter Mode Command

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

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

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

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

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

A device ***May*** support multiple Modes with one or more active at any point in time. Any interactions between them are the responsibility of the Standard or Vendor. Where there are multiple Active 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 6-21 shows the sequence of events during the transition between USB operation and entering a Mode. It illustrates when the Responder's Mode changes and when the Initiator's Mode changes. Figure 6-22 shows a sequence that is Interrupted by a ***Source_Capabilities*** Message, that completes a Contract Negotiation, and then the sequence is Re-run. Figure 6-23 illustrates that when the Responder returns a NAK the transition to a Mode do not take place and the Responder and Initiator remain in their default USB roles.

Figure 6-21 Successful Enter Mode sequence

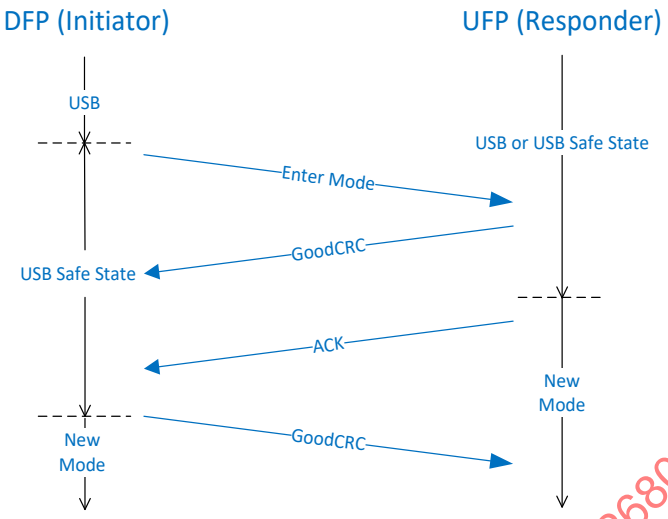


Figure 6-22 Enter Mode sequence Interrupted by Source Capabilities and then Re-run

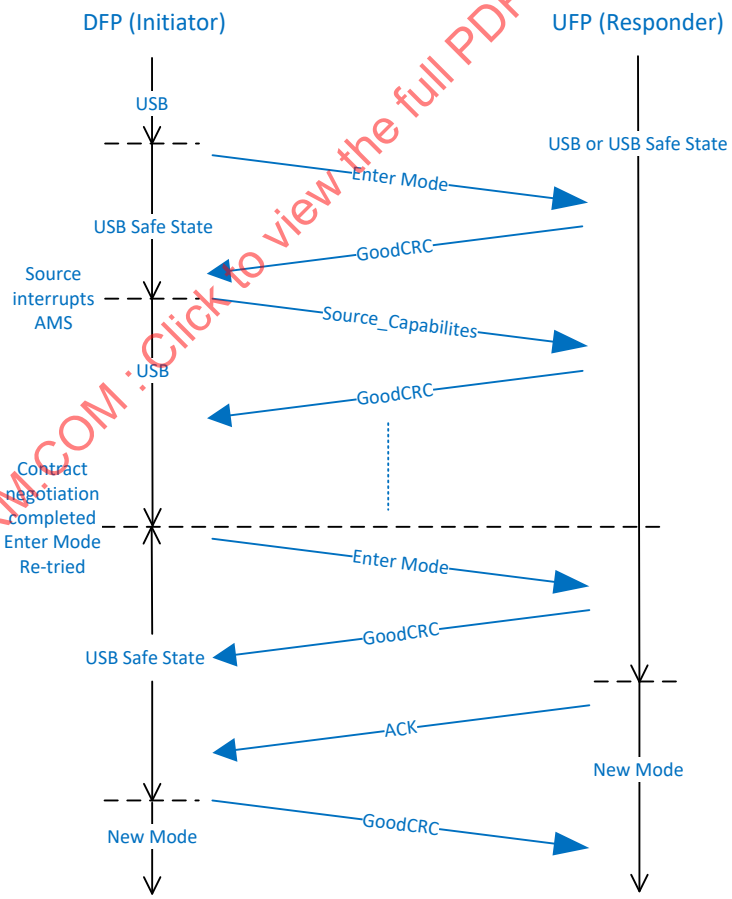
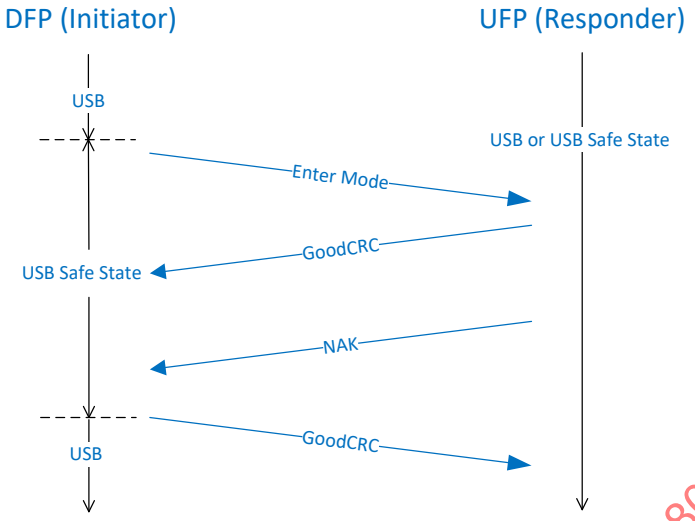


Figure 6-23 Unsuccessful Enter Mode sequence due to NAK



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 ***Shall*** initiate the Exit Mode Process.

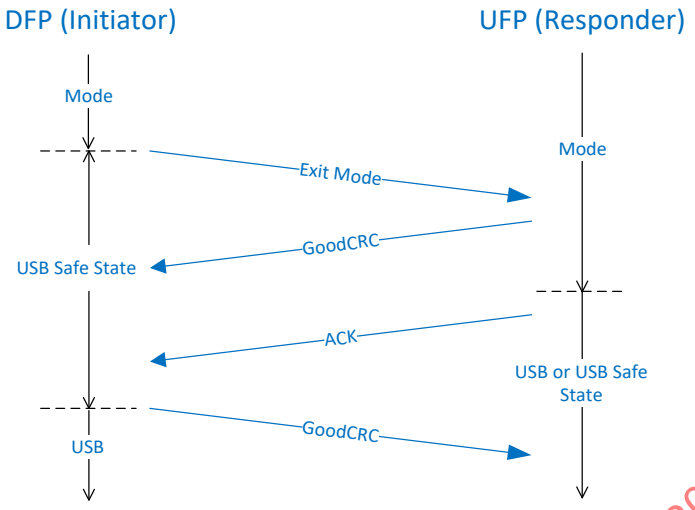
The value in the Object Position field ***Shall*** indicate to which Mode in the ***Discover Modes*** Command the VDO refers (see Figure 6-20) and ***Shall*** have been used previously in an ***Enter Mode*** Command request for an Active Mode. The value 1 always indicates the first Mode as it is the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

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

The Responder ***Shall*** exit its Active Mode before sending the response Message. The Initiator ***Shall*** exit its Active Mode 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 6-24 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 6-24 Exit Mode sequence



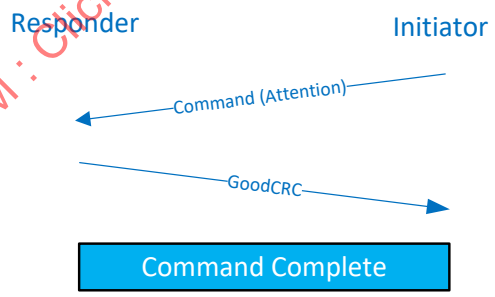
6.4.4.3.6 Attention

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

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

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

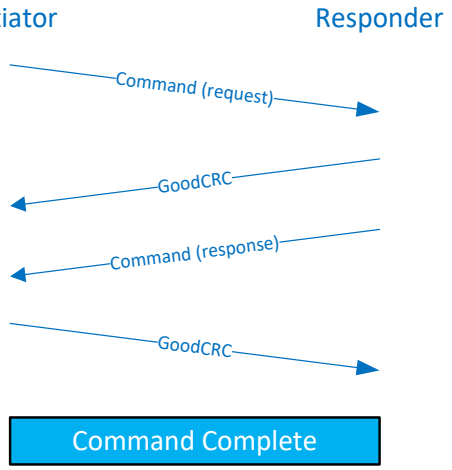
Figure 6-25 Attention Command request/response sequence



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 6-26 shows the request/response sequence including the ***GoodCRC*** Messages.

Figure 6-26 Command request/response sequence



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

- ACK.
- NAK.
- BUSY.

The Responder ***Shall*** complete:

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

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

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

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

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

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

6.4.4.4.1 Discovery Process

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

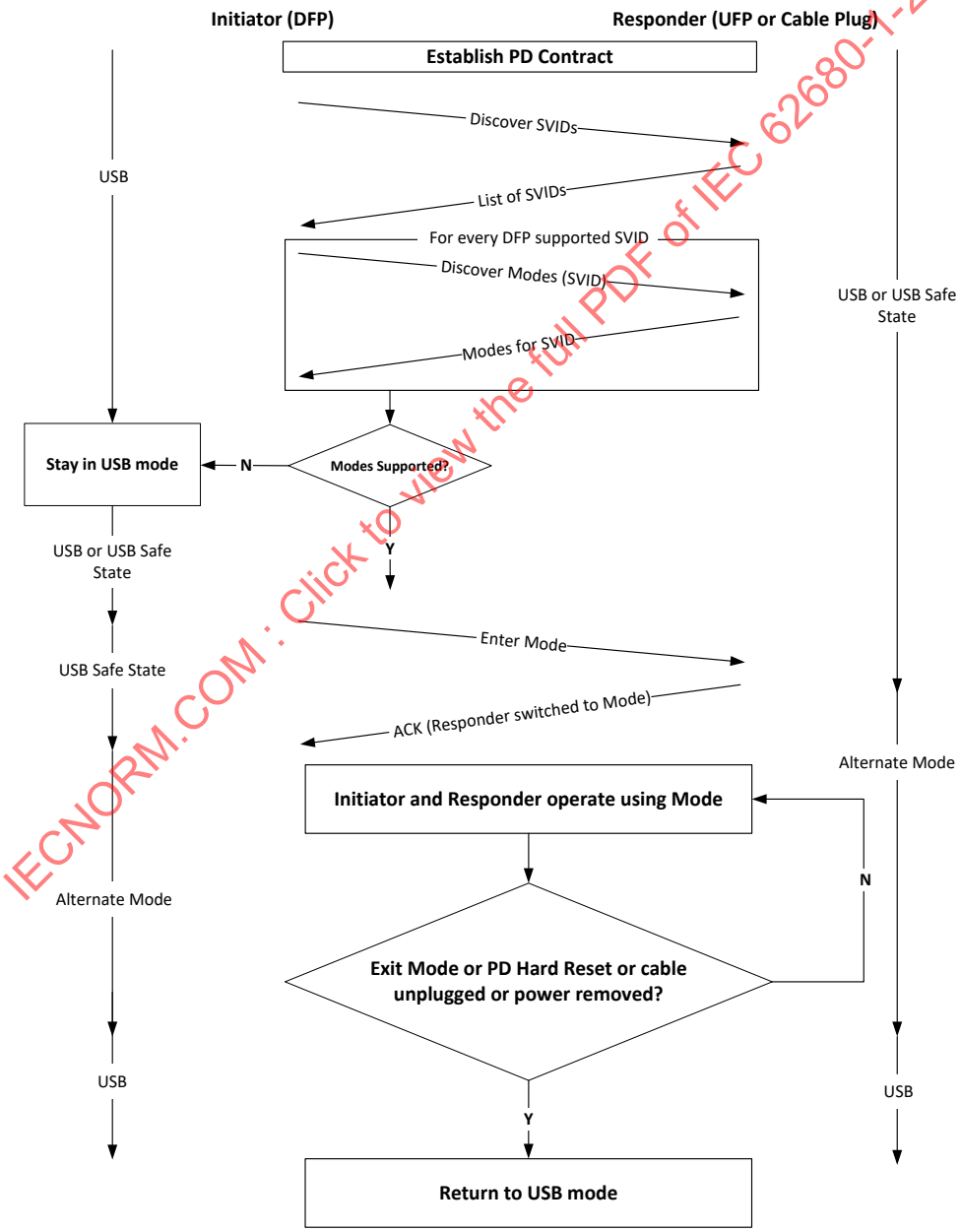
6.4.4.4.2 Enter Vendor Mode / Exit Vendor Mode Processes

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

The Responder (UFP or Cable Plug) and Initiator continue using the Active Mode until the Active Mode is exited. In a managed termination, using the ***Exit Mode*** Command, the Active Mode ***Shall*** be exited in a controlled manner as described in Section 6.4.4.3.5. In an unmanaged termination, triggered by a Power Delivery Hard Reset (i.e. ***Hard Reset*** Signaling sent by either Port Partner) or by cable Detach (device unplugged), the Active Mode ***Shall*** still be exited but there ***Shall Not*** be a transition through the USB Safe State. In both the managed and unmanaged terminations, the Initiator and Responder return to USB operation as defined in ***[USB Type-C 2.0]*** following an exit from a Mode.

The overall Message flow is illustrated in Figure 6-27.

Figure 6-27 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.

6.4.5 Battery_Status Message

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

The ***Battery_Status*** Message returns a BSDO whose format ***Shall*** be as shown in Figure 6-28 and Table 6-44. The ***Number of Data Objects*** field in the ***Battery_Status*** Message ***Shall*** be set to 1.

Figure 6-28 Battery_Status Message

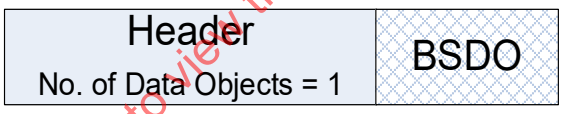


Table 6-44 Battery Status Data Object (BSDO)

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

6.4.5.1 Battery Present Capacity

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

6.4.5.2 Battery Info

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

6.4.5.2.1 Invalid Battery Reference

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

6.4.5.2.2 Battery is Present

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

6.4.5.2.3 Battery Charging Status

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

6.4.6 Alert Message

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

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

The ***Alert*** Message ***Shall*** contain exactly one Alert Data Object (ADO) and the format ***Shall*** be as shown in Figure 6-29 and Table 6-45.

Figure 6-29 Alert Message

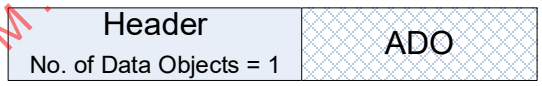


Table 6-45 Alert Data Object

Bit(s)	Field	Description
B31…24	***Type of Alert***	**Bit** / **Description** 0 — ***Reserved*** and ***Shall*** be set to zero 1 — Battery Status Change Event (Attach/Detach/charging/discharging/idle) 2 — OCP event when set (Source only, for Sink ***Reserved*** and ***Shall*** be set to zero) 3 — OTP event when set 4 — Operating Condition Change when set 5 — Source Input Change Event when set 6 — OVP event when set 7 — ***Reserved*** and ***Shall*** be set to zero

Bit(s)	Field	Description
B23…20	***Fixed Batteries***	When Battery Status Change bit set indicates which Fixed Batteries have had a status change. B20 corresponds to Battery 0 and B23 corresponds to Battery 3.
B19…16	***Hot Swappable Batteries***	When Battery Status Change bit set indicates which Hot Swappable Batteries have had a status change. B16 corresponds to Battery 4 and B19 corresponds to Battery 7.
B15…0	***Reserved***	***Shall*** be set to zero

6.4.6.1 Type of Alert

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

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

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

6.4.6.1.1 Battery Status Change

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

6.4.6.1.2 Over-Current Protection Event

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

6.4.6.1.3 Over-Temperature Protection Event

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

6.4.6.1.4 Operating Condition Change

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

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

6.4.6.1.5 Source Input Change Event

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

6.4.6.1.6 Over-Voltage Protection Event

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

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

6.4.6.2 Fixed Batteries

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

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

6.4.6.3 Hot Swappable Batteries

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

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

6.4.7 Get_Country_Info Message

The ***Get_Country_Info*** Message ***Shall*** be sent by a port to get country specific information from its port partner using the country's Alpha-2 Country Code defined by ***[ISO 3166]***. The port partner responds with a ***Country_Info*** Message that contains the country specific information. The ***Get_Country_Info*** Message ***Shall*** be as shown in Figure 6-30 and Table 6-46.

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

Figure 6-30 Get_Country_Info Message

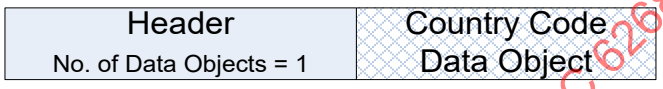


Table 6-46 Country Code Data Object

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

6.4.8 Enter_USB Message

The ***Enter_USB*** Message ***Shall*** be sent by the DFP to its UFP Port Partner and to the Cable Plug(s), when in an Explicit Contract, to enter a specified USB Mode of operation. The recipient of the Message ***Shall*** respond by sending either an ***Accept*** Message in response to a ***Valid*** request or a ***Reject*** Message in response to an ***Invalid*** request.

The ***Enter_USB*** Message ***Shall*** be sent by a ***[USB4]*** PDUSB Hub's DFP(s) or ***[USB4]*** PDUSB Host's DFP(s) within ***tEnterUSB*** following the initial power-on or a Data Reset to enter ***[USB4]*** operation.

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

The ***Enter_USB*** Message ***Shall*** be used by a PDUSB Hub's DFP(s) to speculatively train the USB links or enter ***[DPTC1.0]*** or ***[TBT3]*** Alternate Modes prior to the presence of a host. In this case, the Host Present bit ***Shall*** be cleared. When the Host is Connected the ***Enter_USB*** Message ***Shall*** be resent with the Host Present bit set. The ***Enter_USB*** Message's Enter USB Data Object (EUDO), received from the Root Hub when the USB Host is connected, ***Shall*** be propagated down through the hub tree.

See ***[USB Type-C 2.0]*** USB4 Hub Connection Requirements.

Figure 6-31 Enter_USB Message

Table 6-47 Enter_USB Data Object

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

6.4.8.1 USB Mode Field

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

6.4.8.2 USB4 DRD Field

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

6.4.8.3 USB3 DRD Field

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

6.4.8.4 Cable Speed Field

The ***Cable Speed*** field ***Shall*** be used to indicate the cable's maximum speed.

6.4.8.5 Cable Type Field

The *Cable Type* field ***Shall*** be used to indicate whether the cable is passive or active. Further if the cable is active, it indicates the type of active circuits in the cable and if the cable is optically isolated.

6.4.8.6 Cable Current Field

The *Cable Current* field ***Shall*** be used to indicate the cable's current carrying capability.

6.4.8.7 PCIe Support Field

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

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

6.4.8.8 DP Support Field

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

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

6.4.8.9 TBT Support Field

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

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

6.4.8.10 Host Present Field

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

6.5 Extended Message

An Extended Message ***Shall*** contain an Extended Message Header (indicated by the *Extended* field in the Message Header being set) and be followed by zero or more data bytes.

The format of the Extended Message is defined by the Message Header's *Message Type* field and is summarized in Table 6-48. 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 6-48 Extended Message Types

Bits 4…0	Type	Sent by	Description	Valid Start of Packet
0 0000	***Reserved***		All values not explicitly defined are ***Reserved*** and ***Shall Not*** be used.	
0 0001	*Source_Capabilities_Extended*	Source or Dual-Role Power	See Section 6.5.1	SOP only
0 0010	*Status*	Source or Sink	See Section 6.5.2	SOP*
0 0011	*Get_Battery_Cap*	Source or Sink	See Section 6.5.3	SOP only
0 0100	*Get_Battery_Status*	Source or Sink	See Section 6.5.4	
0 0101	*Battery_Capabilities*	Source or Sink	See Section 6.5.5	SOP only
0 0110	*Get_Manufacturer_Info*	Source or Sink	See Section 6.5.6	SOP*

Bits 4…0	Type	Sent by	Description	Valid Start of Packet
0 0111	***Manufacturer_Info***	Source, Sink or Cable Plug	See Section 6.5.7	SOP*
0 1000	***Security_Request***	Source or Sink	See Section 6.5.8.1	SOP*
0 1001	***Security_Response***	Source, Sink or Cable Plug	See Section 6.5.8.2	SOP*
0 1010	***Firmware_Update_Request***	Source or Sink	See Section 6.5.9.1	SOP*
0 1011	***Firmware_Update_Response***	Source, Sink or Cable Plug	See Section 6.5.9.2	SOP*
0 1100	***PPS_Status***	Source	See Section 6.5.10	SOP only
0 1101	***Country_Info***	Source or Sink	See Section 6.5.12	SOP only
0 1110	***Country_Codes***	Source or Sink	See Section 6.5.11	SOP only
0 1111	***Sink_Capabilities_Extended***	Sink or Dual-Role Power	See Section 6.5.13	SOP only
1 0000 - 1 1111	***Reserved***		All values not explicitly defined are ***Reserved*** and ***Shall Not*** be used.	

6.5.1 Source_Capabilities_Extended Message

The ***Source_Capabilities_Extended*** Message ***Should*** be sent in response to a ***Get_Source_Cap_Extended*** Message. The ***Source_Capabilities_Extended*** Message enables a Source or a DRP to inform the Sink about its capabilities as a Source.

The ***Source_Capabilities_Extended*** Message ***Shall*** return a 24-byte Source Capabilities Extended Data Block (SCEDB) whose format ***Shall*** be as shown in Figure 6-32 and Table 6-49.

Figure 6-32 Source_Capabilities_Extended Message

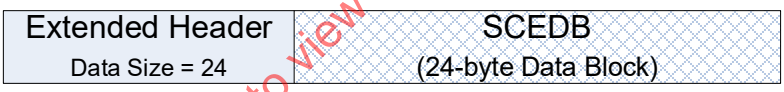


Table 6-49 Source Capabilities Extended Data Block (SCEDB)

Offset	Field	Description
0	VID	Vendor ID (assigned by the USB-IF)
2	PID	Product ID (assigned by the manufacturer)
4	XID	Value provided by the USB-IF assigned to the product
8	FW Version	Firmware version number
9	HW Version	Hardware version number
10	Voltage Regulation	**Bit** / **Description** 1…0: 00b: 150mA/µs Load Step (default); 01b: 500mA/µs Load Step; 11b…10b: ***Reserved*** and ***Shall Not*** be used 2: 0b: 25% IoC (default); 1b: 90% IoC 3…7: ***Reserved*** and ***Shall*** be set to zero
11	Holdup Time	Output will stay with regulated limits for this number of milliseconds after removal of the AC from the input. 0x00 = feature not supported Note: a value of 3ms ***Should*** be used

<table>
<tr><th>Offset</th><th>Field</th><th>Description</th></tr>
<tr><td>12</td><td>Compliance</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>LPS compliant when set</td></tr>
<tr><td>1</td><td>PS1 compliant when set</td></tr>
<tr><td>2</td><td>PS2 compliant when set</td></tr>
<tr><td>3…7</td><td>Reserved and Shall be set to zero</td></tr>
</table>
</td></tr>
<tr><td>13</td><td>Touch Current</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>Low touch Current EPS when set</td></tr>
<tr><td>1</td><td>Ground pin supported
when set</td></tr>
<tr><td>2</td><td>Ground pin intended for protective earth
when set</td></tr>
<tr><td>3…7</td><td>Reserved and Shall be set to zero</td></tr>
</table>
</td></tr>
<tr><td>14</td><td>Peak Current1</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0…4</td><td>Percent overload in 10% increments
Values higher than 25 (11001b) are clipped to 250%.</td></tr>
<tr><td>5…10</td><td>Overload period in 20ms</td></tr>
<tr><td>11.14</td><td>Duty cycle in 5% increments</td></tr>
<tr><td>15</td><td>V_{BUS} Voltage droop</td></tr>
</table>
</td></tr>
<tr><td>16</td><td>Peak Current2</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0…4</td><td>Percent overload in 10% increments
Values higher than 25 (11001b) are clipped to 250%.</td></tr>
<tr><td>5…10</td><td>Overload period in 20ms</td></tr>
<tr><td>11.14</td><td>Duty cycle in 5% increments</td></tr>
<tr><td>15</td><td>V_{BUS} Voltage droop</td></tr>
</table>
</td></tr>
<tr><td>18</td><td>Peak Current3</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0…4</td><td>Percent overload in 10% increments
Values higher than 25 (11001b) are clipped to 250%.</td></tr>
<tr><td>5…10</td><td>Overload period in 20ms</td></tr>
<tr><td>11.14</td><td>Duty cycle in 5% increments</td></tr>
<tr><td>15</td><td>V_{BUS} Voltage droop</td></tr>
</table>
</td></tr>
<tr><td>20</td><td>Touch Temp</td><td>Temperature conforms to:
0 = [IEC 60950-1] (default)
1 = [IEC 62368-1] TS1
2 = [IEC 62368-1] TS2
Note: All other values Reserved</td></tr>
</table>

<table>
<tr><th>Offset</th><th>Field</th><th>Description</th></tr>
<tr><td>21</td><td>Source Inputs</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>0b: No external supply
1b: External supply present</td></tr>
<tr><td>1</td><td>If bit 0 is set:

0b: External supply is constrained
1b: External supply is unconstrained

If bit 0 is not set Reserved and Shall be set to zero</td></tr>
<tr><td>2</td><td>0b: No internal Battery
1b: Internal Battery present</td></tr>
<tr><td>3…7</td><td>Reserved and Shall be set to zero</td></tr>
</table>
</td></tr>
<tr><td>22</td><td>Number of Batteries/Battery Slots</td><td>Upper Nibble = Number of Hot Swappable Battery Slots (0…4)
Lower Nibble = Number of Fixed Batteries (0…4)</td></tr>
<tr><td>23</td><td>Source PDP Rating</td><td>0…6: Source PDP rating
7: Reserved and Shall be set to zero</td></tr>
</table>

6.5.1.1 Vendor ID (VID) Field

The Vendor ID field ***Shall*** contain the 16-bit Vendor ID (VID) assigned to the Source's vendor by the USB-IF. If the vendor does not have a VID, the Vendor ID field ***Shall*** be set to zero. Devices that have a USB data interface ***Shall*** report the same VID as the idVendor in the Standard Device Descriptor (see ***[USB 2.0]*** and ***[USB 3.2]***).

6.5.1.2 Product ID (PID) Field

The Product ID field ***Shall*** contain the 16-bit Product ID (PID) assigned by the Source's vendor. Devices that have a USB data interface ***Shall*** report the same PID as the idProduct in the Standard Device Descriptor (see ***[USB 2.0]*** and ***[USB 3.2]***).

6.5.1.3 XID Field

The XID field ***Shall*** contain the 32-bit XID provided by the USB-IF to the vendor who in turns assigns it to a product. If the vendor does not have an XID, then it ***Shall*** return zero in this field (see ***[USB 2.0]*** and ***[USB 3.2]***).

6.5.1.4 Firmware Version Field

The Firmware Version field ***Shall*** contain an 8-bit firmware version number assigned to the device by the vendor.

6.5.1.5 Hardware Version Field

The Hardware Version field ***Shall*** contain an 8-bit hardware version number assigned to the device by the vendor.

6.5.1.6 Voltage Regulation Field

The Voltage Regulation field contains bits covering Load Step Slew Rate and Magnitude.

See Section 7.1.12.1 for further details.

6.5.1.6.1 Load Step Slew Rate

The Source ***Shall*** report its load step response capability in bits 0…1 of the Voltage Regulation bit field.

6.5.1.6.2 Load Step Magnitude

The Source ***Shall*** report its load step magnitude rate as a percentage of IoC in bit 2 of the Voltage Regulation field.

6.5.1.7 Holdup Time Field

The Holdup Time field ***Shall*** contain the Source's holdup time (see Section 7.1.12.2).

6.5.1.8 Compliance Field

The Compliance field ***Shall*** contain the standards the Source is compliant with (see Section 7.1.12.3).

6.5.1.9 Touch Current Field

The Touch Current field reports whether the Source meets certain leakage current levels and if it has a ground pin.

A Source ***Shall*** set the Touch Current bit (bit-0) when their leakage current is less than 65µA rms when Source's maximum capability is less than or equal to 30W, or when their leakage current is less than 100 µA rms when its power capability is between 30W and 100W. The total combined leakage current ***Shall*** be measured in accordance with ***[IEC 60950-1]*** when tested at 250VAC rms at 50 Hz.

A Source with a ground pin ***Shall*** set the Ground pin bit (bit 1).

A Source whose Ground pin is intended to be connected to a protective earth ***Shall*** set both bit1 and bit 2.

6.5.1.10 Peak Current Field

The Peak Current field ***Shall*** contain the combinations of Peak Current that the Source supports (see Section 7.1.12.4).

Peak Current provides a means for Source report its ability to provide current in excess of the negotiated amount for short periods. The Peak Current descriptor defines up to three combinations of % overload, duration and duty cycle defined as PeakCurrent1, PeakCurrent2 and PeakCurrent3 that the Source supports. A Source ***May*** offer no Peak Current capability. A Source ***Shall*** populate unused Peak Current bit fields with zero.

The Bit Fields within Peak Current1, Peak Current2, and Peak Current3 contain the following subfields:

- **Percentage Overload** ***Shall*** be the maximum peak current reported in 10% increments as a percentage of the negotiated operating current (IoC) offered by the Source. Values higher than 25 (11001b) are clipped to 250%.
- **Overload Period** ***Shall*** be the minimum rolling average time window in 20ms increments, where a value of 20ms is recommended.
- **Duty Cycle** ***Shall*** be the maximum percentage of overload period reported in 5% increments. The values ***Should*** be 5%, 10% and 50% for PeakCurrent1, PeakCurrent2 and PeakCurrent3 respectively.
- **V_{BUS} Droop** ***Shall*** be set to one to indicate there is an additional 5% voltage droop on V_{BUS} when the overload conditions occur as defined by ***vSrcPeak***. However, it is recommended that the Source ***Should*** provide V_{BUS} in the range of ***vSrcNew*** when overload conditions occur and set this bit to zero.

6.5.1.11 Touch Temp Field

The Touch Temp field ***Shall*** report the IEC standard used to determine the surface temperature of the Source's enclosure. Safety limits for the Source's touch temperature are set in applicable product safety standards (e.g. ***[IEC 60950-1]*** or ***[IEC 62368-1]***). The Source ***May*** report when its touch temperature performance conforms to the TS1 or TS2 limits described in ***[IEC 62368-1]***.

6.5.1.12 Source Inputs Field

The Source Inputs field ***Shall*** identify the possible inputs that provide power to the Source. Note some Sources are only powered by a Battery (e.g. an automobile) rather than the more common mains.

- When bit 0 is set, the Source can be sourced by an external power supply.
- When bits 0 and 1 are set, the Source can be sourced by an external power supply which is assumed to be effectively "infinite" i.e. it won't run down over time.
- When bit 2 is set the Source can be sourced by an internal Battery.

Bit 2 ***May*** be set independently of bits 0 and 1.

6.5.1.13 Number of Batteries/Battery Slots Field

The Number of Batteries/Battery Slots field ***Shall*** report the number of Fixed Batteries and Hot Swappable Battery Slots the Source supports. This field ***Shall*** independently report the number of Battery Slots and the number of Fixed Batteries.

A Source ***Shall*** have no more than 4 Fixed Batteries and no more than 4 Battery Slots.

Fixed Batteries ***Shall*** be numbered consecutively from 0 to 3. The number assigned to a given Fixed Battery ***Shall Not*** change between Attach and Detach.

Battery Slots ***Shall*** be numbered consecutively from 4 to 7. The number assigned to a given Battery Slot ***Shall Not*** change between Attach and Detach.

6.5.1.14 Source PDP Rating Field

The Source PDP Rating field ***Shall*** report the integer portion of the Source's Source PDP Rating as defined in Table 10-2.

The Source PDP Rating field that is reported ***Shall*** be invariant and ***Shall*** follow the ***[USB Type-C 2.0]*** requirements for single-port, Multi-port Assured Capacity Chargers, or Multi-port Shared Capacity Chargers.

6.5.2 Status Message

The ***Status*** Message ***Shall*** be sent in response to a ***Get_Status*** Message. The content of the ***Status*** Message depends on the target of the ***Get_Status*** Message. When sent to ***SOP*** the Status Message returns the status of the Port's Port Partner. When sent to ***SOP'*** or ***SOP"*** the ***Status*** Message returns the status of one of the Active Cable's Cable Plugs.

6.5.2.1 SOP Status Message

A ***Status*** Message, sent in response to ***Get_Status*** Message to ***SOP***, enables a Port to inform its Port Partner about the present status of the Source or Sink. Typically, a ***Get_Status*** Message will be sent by the Port after receipt of an ***Alert*** Message. Some of the reported events are critical such as OCP, OVP and OTP, while others are informative such as change in a Battery's status from charging to neither charging nor discharging.

The ***Status*** Message returns a 6-byte Status Data Block (SDB) whose format ***Shall*** be as shown in Figure 6-33 and Table 6-50.

Figure 6-33 SOP Status Message

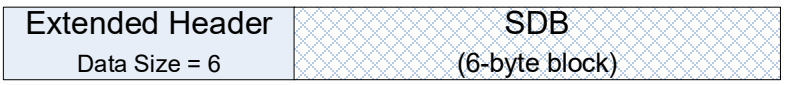


Table 6-50 SOP Status Data Block (SDB)

Offset (Byte)	Field	Description
0	Internal Temp	Source or Sink's internal temperature in degrees centigrade. 0 = feature not supported 1 = temperature is less than 2°C. 2-255 = temperature in °C.

<table>
<tr><th>Offset (Byte)</th><th>Field</th><th>Description</th></tr>
<tr><td>1</td><td>Present Input</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>Reserved and Shall be set to zero</td></tr>
<tr><td>1</td><td>External Power when set</td></tr>
<tr><td>2</td><td>External Power AC/DC (Valid when Bit 1 set)
0: DC
1: AC
Reserved when Bit 1 is zero</td></tr>
<tr><td>3</td><td>Internal Power from Battery when set</td></tr>
<tr><td>4</td><td>Internal Power from non-Battery power source when set</td></tr>
<tr><td>5…7</td><td>Reserved and Shall be set to zero</td></tr>
</table>
</td></tr>
<tr><td>2</td><td>Present Battery Input</td><td>When Present Input field bit 3 set Shall contain the bit corresponding to the Battery or Batteries providing power:

Upper nibble = Hot Swappable Battery (b7…4)
Lower nibble = Fixed Battery (b3…0)

When Present Source Input field bit 3 is not set this field is Reserved and Shall be set to zero.</td></tr>
<tr><td>3</td><td>Event Flags</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>Reserved and Shall be set to zero</td></tr>
<tr><td>1</td><td>OCP event when set</td></tr>
<tr><td>2</td><td>OTP event when set</td></tr>
<tr><td>3</td><td>OVP event when set</td></tr>
<tr><td>4</td><td>CP mode when set, CV mode when cleared</td></tr>
<tr><td>5…7</td><td>Reserved and Shall be set to zero</td></tr>
</table>
</td></tr>
<tr><td>4</td><td>Temperature Status</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>Reserved and Shall be set to zero</td></tr>
<tr><td>1…2</td><td>00 – Not Supported
01 – Normal
10 – Warning
11 – Over temperature</td></tr>
<tr><td>3…7</td><td>Reserved and Shall be set to zero</td></tr>
</table>
</td></tr>
<tr><td>5</td><td>Power Status</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>Reserved and Shall be set to zero</td></tr>
<tr><td>1</td><td>Source power limited due to cable supported current</td></tr>
<tr><td>2</td><td>Source power limited due to insufficient power available while sourcing other ports</td></tr>
<tr><td>3</td><td>Source power limited due to insufficient external power</td></tr>
<tr><td>4</td><td>Source power limited due to Event Flags in place (Event Flags must also be set)</td></tr>
<tr><td>5</td><td>Source power limited due to temperature</td></tr>
<tr><td>6…7</td><td>Reserved and Shall be set to zero</td></tr>
</table>
</td></tr>
</table>

6.5.2.1.1 Internal Temp Field

The Internal Temp field reports the instantaneous temperature of a portion of the Source or Sink.

6.5.2.1.2 Present Input Field

The Present Input field indicates which supplies are presently powering the Source or Sink.

The following bits are defined:

- Bit 1 indicates that an external Source is present.
- Bit 2 indicates whether the external unconstrained Source is AC or DC.
- Bit 3 indicates that power is being provided from Battery.
- Bit4 indicates an alternative internal source of power that is not a Battery.

6.5.2.1.3 Present Battery Input Field

The Present Battery Input field indicates which Battery or Batteries are presently supplying power to the Source or Sink. The Present Battery Input field is only ***Valid*** when the Present Input field indicates that there is Internal Power from Battery.

The upper nibble of the field indicates which Hot Swappable Battery/Batteries are supplying power with bit 4 in upper nibble corresponding to Battery 4 and bit 7 in the upper nibble corresponding to Battery 7 (see Section 6.5.3 and Section 6.5.4).

The lower nibble of the field indicates which Fixed Battery/Batteries are supplying power with bit 0 in lower nibble corresponding to Battery 0 and bit 3 in the lower nibble corresponding to Battery 3 (see Section 6.5.3 and Section 6.5.4).

6.5.2.1.4 Event Flags Field

The Event Flags field returns event flags. The OTP, OVP and OCP event flags ***Shall*** be set when there is an event and ***Shall*** only be cleared when read with the ***Get_Status*** Message.

When the OTP event flag is set the Temperature Status field ***Shall*** also be set to over temperature.

The CL/CV mode bit is only ***Valid*** when operating as a Programmable Power Supply and ***Shall*** be ***Ignored*** otherwise. When the Source is operating as a Programmable Power Supply the CL/CV mode bit ***Shall*** be set when operating in Current Limit mode (CL mode) and ***Shall*** be cleared when operating in Constant Voltage mode (CV mode).

6.5.2.1.5 Temperature Status Field

The Temperature Status field returns the current temperature status of the device either: normal, warning and over temperature. When the Temperature Status field is set to over temperature the OTP event flag ***Shall*** also be set.

6.5.2.1.6 Power Status Field

The Power Status field indicates the current status of a Source. A non-zero return of the field indicates advertised Source power is being reduced for either: the cable does not support the full Source current, the Source is supplying power to other ports and is unable to provide its full power, the external power to the Source is insufficient to support full power, or an Event has occurred that is causing the Source to reduce it advertised power.

A Sink ***Shall*** set this field to zero.

6.5.2.1 SOP'/SOP'' Status Message

A ***Status*** Message, sent in response to a ***Get_Status*** Message to ***SOP'*** or ***SOP''***, enables a Source or Sink to get the present status of the Active Cable's Cable Plug(s). Typically, a ***Get_Status*** Message will be used by the USB Host and/or USB Device to manage the Active Cable's Cable Plug(s) temperature. The ***Status*** Message returns a 2-byte Status Data Block (SDB) whose format ***Shall*** be as shown in Figure 6-34 and Table 6-51.

Figure 6-34 SOP'/SOP'' Status Message

Table 6-51 SOP'/SOP'' Status Data Block (SDB)

Offset (Byte)	Field	Value	Description
0	Internal Temp	Unsigned Int	Active Cable plug's internal temperature in °C. 0 = feature not supported 1 = temperature is less than 2°C. 2…255 = temperature in °C.
1	Flags	Bit field	**Bit** / **Description** 0 – Thermal Shutdown 1…7 – ***Reserved*** and ***Shall*** be set to zero

6.5.2.1.1 Internal Temp Field

The Internal Temp field reports the instantaneous temperature of the plug in °C. The internal temperature ***Shall*** be monotonic. The Active Cable ***Shall*** report its internal temperature every ***tACTempUpdate***.

6.5.2.1.2 Thermal Shutdown Field

The Thermal Shutdown flag ***Shall*** also be set when the plug's internal temperature exceeds the Internal Maximum Temperature reported in the Active Cable VDO. Once this bit has been set, it ***Shall*** remain set and the plug ***Shall*** remain in Thermal Shutdown until there is a Hard Reset or the Active Cable's power is removed. The Thermal Shutdown flag ***Shall Not*** be cleared by a Cable Reset.

6.5.3 Get_Battery_Cap Message

The ***Get_Battery_Cap*** (Get Battery Capabilities) Message is used to request the capability of a Battery present in its Port Partner. The Port ***Shall*** respond by returning a ***Battery_Capabilities*** Message (see Section 6.5.5) containing a Battery Capabilities Data Block (BCDB) for the targeted Battery.

The ***Get_Battery_Cap*** Message contains a 1 byte Get Battery Cap Data Block (GBCDB), whose format ***Shall*** be as shown in Figure 6-35 and Table 6-52. This block defines for which Battery the request is being made.

The ***Data Size*** field in the ***Get_Battery_Cap*** Message ***Shall*** be set to 1.

Figure 6-35 Get_Battery_Cap Message

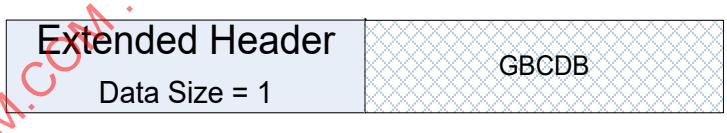


Table 6-52 Get Battery Cap Data Block (GBCDB)

Offset	Field	Description
0	***Battery Cap Ref***	Number of the Battery indexed from zero: • Values 0…3 represent the Fixed Batteries. • Values 4…7 represent the Hot Swappable Batteries. • Values 8…255 are ***Reserved*** and ***Shall Not*** be used.

6.5.4 Get_Battery_Status Message

The ***Get_Battery_Status*** (Get Battery Status) Message is used to request the status of a Battery present in its Port Partner. The port ***Shall*** respond by returning a ***Battery_Status*** Message (see Section 6.4.5) containing a Battery Status Data Object (BSDO) for the targeted Battery.

The ***Get_Battery_Status*** Message contains a 1 byte Get Battery Status Data Block (GBSDB) whose format ***Shall*** be as shown in Figure 6-36 and Table 6-53. This block contains details of the requested Battery. The ***Data Size*** field in the ***Get_Battery_Status*** Message ***Shall*** be set to 1.

Figure 6-36 Get_Battery_Status Message

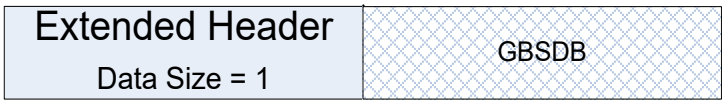


Table 6-53 Get Battery Status Data Block (GBSDB)

Offset	Field	Description
0	***Battery Status Ref***	Number of the Battery indexed from zero: • Values 0…3 represent the Fixed Batteries. • Values 4…7 represent the Hot Swappable Batteries. • Values 8…255 are ***Reserved*** and ***Shall Not*** be used.

6.5.5 Battery_Capabilities Message

The ***Battery_Capabilities*** Message is sent in response to a ***Get_Battery_Cap*** Message. The ***Battery_Capabilities*** Message contains one Battery Capability Data Block (BCDB) for one of the Batteries its supports as reported by Battery field in the ***Source_Capabilities_Extended*** Message. The returned BCDB ***Shall*** correspond to the Battery requested in the ***Battery Cap Ref*** field contained in the ***Get_Battery_Cap*** Message.

The ***Battery_Capabilities*** Message returns a 9-byte BCDB whose format ***Shall*** be as shown in Figure 6-37 and Table 6-50.

Figure 6-37 Battery_Capabilities Message

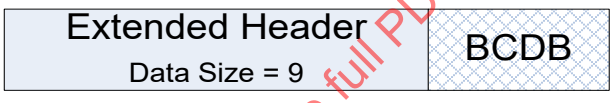


Table 6-54 Battery Capability Data Block (BCDB)

Offset (Byte)	Field	Description
0	VID	Vendor ID (assigned by the USB-IF)
2	PID	Product ID (assigned by the manufacturer)
4	Battery Design Capacity	Battery's design capacity in 0.1 WH Note: 0x0000 = Battery not present 0xFFFF = design capacity unknown
6	Battery Last Full Charge Capacity	Battery's last full charge capacity in 0.1 WH Note: 0x0000 = Battery not present 0xFFFF = last full charge capacity unknown
8	Battery Type	**Bit** / **Description** 0 – Invalid Battery reference 1-7 – ***Reserved***

6.5.5.1 Battery Design Capacity Field

The Battery Design Capacity field ***Shall*** return the Battery's design capacity in tenths of WH. If the Battery is Hot Swappable and is not present, the Battery Design Capacity field ***Shall*** be set to 0. If the Battery is unable to report its Design Capacity, it ***Shall*** return 0xFFFF.

6.5.5.2 Battery Last Full Charge Capacity Field

The Battery Last Full Charge Capacity field ***Shall*** return the Battery's last full charge capacity in tenths of WH. If the Battery is Hot Swappable and is not present, the Battery Last Full Charge Capacity field ***Shall*** be set to 0. If the Battery is unable to report its Design Capacity, the Battery Last Full Charge Capacity field ***Shall*** be set to 0xFFFF.

6.5.5.3 Battery Type Field

The Battery Type Field is used to report additional information about the Battery's capabilities.

6.5.5.3.1 Invalid Battery Reference

The Invalid Battery Reference bit ***Shall*** be set when the ***Get_Battery_Cap*** Message contains a reference to a Battery that does not exist.

6.5.6 Get_Manufacturer_Info Message

The ***Get_Manufacturer_Info*** (Get Manufacturer Info) Message is sent by a Port to request manufacturer specific information relating to its Port Partner or Cable Plug or of a Battery behind a Port. The Port ***Shall*** respond by returning a ***Manufacturer_Info*** Message (Section 6.5.7) containing a Manufacturer Info Data Block (MIDB). Support for this feature by the Cable Plug is ***Optional Normative***.

The ***Get_Manufacturer_Info*** Message contains a 2-byte Get Manufacturer Info Data Block (GMIDB). This block defines whether it is the Device or Battery manufacturer information being requested and for which Battery the request is being made.

The ***Get_Manufacturer_Info*** Message returns a GMIDB whose format ***Shall*** be as shown in Figure 6-36 and Table 6-55.

Figure 6-38 Get_Manufacturer_Info Message

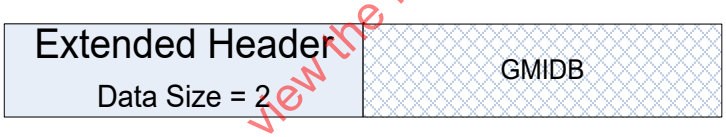


Table 6-55 Get Manufacturer Info Data Block (GMIDB)

Offset	Field	Description
0	***Manufacturer Info Target***	0: Port/Cable Plug 1: Battery 255…2: ***Reserved***, ***Shall Not*** be used.
1	***Manufacturer Info Ref***	If ***Manufacturer Info Target*** subfield is Battery (01b) the ***Manufacturer Info Ref*** field ***Shall*** contain the Battery number reference which is the number of the Battery indexed from zero: • Values 0…3 represent the Fixed Batteries. • Values 4…7 represent the Hot Swappable Batteries. Otherwise this field is ***Reserved*** and ***Shall*** be set to zero.

6.5.7 Manufacturer_Info Message

The ***Manufacturer_Info*** Message ***Shall*** be sent in response to a ***Get_Manufacturer_Info*** Message. The ***Manufacturer_Info*** Message contains the USB VID and the Vendor's PID to identify the device or Battery and the device or Battery's manufacturer byte array in a variable length Data Block of up to ***MaxExtendedMsgLegacyLen*** .

The ***Manufacturer_Info*** Message returns a Manufacturer Info Data Block (MIDB) whose format ***Shall*** be as shown in Figure 6-37 and Table 6-50.

Figure 6-39 Manufacturer_Info Message

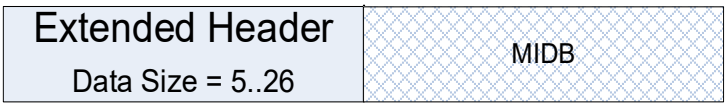


Table 6-56 Manufacturer Info Data Block (MIDB)

Offset	Field	Description
0	VID	Vendor ID (assigned by the USB-IF)
2	PID	Product ID (assigned by the manufacturer)
4	Manufacturer String	Vendor defined null terminated string of 0…21 characters If the Manufacturer Info Target field or Manufacturer Info Ref field in the ***Get_Manufacturer_Info*** Message is unrecognized the field Shall return a null terminated ascii text string "Not Supported".

6.5.7.1 Vendor ID (VID)

The VID field ***Shall*** contain the device's or Battery's manufacturer string as defined by the vendor.

If the ***Manufacturer Info Target*** field in the ***Get_Manufacturer_Info*** Message is ***Invalid***, this VID field ***Shall*** be 0xFFFF, and the associated PID field ***Should*** be set to 0x0000. If the ***Manufacturer Info Target*** field in the ***Get_Manufacturer_Info*** Message equals Battery (01b) and the ***Manufacturer Info Ref*** field is ***Invalid***, this VID field ***Shall*** be 0xFFFF, and the associated PID field ***Should*** be set to 0x0000.

6.5.7.2 Product ID (PID)

The PID field ***Shall*** contain the device's or Battery's 16-bit product identifier designated by the vendor.

If the ***Manufacturer Info Target*** field in the ***Get_Manufacturer_Info*** Message is ***Invalid***, this PID field ***Should*** be set to 0x0000. If the ***Manufacturer Info Target*** field in the ***Get_Manufacturer_Info*** Message equals Battery (01b) and the ***Manufacturer Info Ref*** field is ***Invalid***, this PID field ***Should*** be set to 0x0000.

On receiving a ***Manufacturer_Info*** Message, with the VID set to 0xFFFF, the PID field ***Shall*** be ***Ignored***.

6.5.7.3 Manufacturer String

This field ***Shall*** contain the device's or Battery's manufacturer string as defined by the vendor.

If the ***Manufacturer Info Target*** field or ***Manufacturer Info Ref*** field in the ***Get_Manufacturer_Info*** Message is unrecognized the field ***Shall*** return a null terminated ascii text string "Not Supported".

6.5.8 Security Messages

The authentication process between Port Partners or a Port and Cable Plug is fully described in ***[USBTypeCAuthentication 1.0]***. This specification describes two Extended Messages used by the authentication process when applied to PD.

In the authentication process described in ***[USBTypeCAuthentication 1.0]*** there are three basic exchanges that serve to:

- Get the Port or Cable Plug's certificates.
- Get the Port or Cable Plug's digest.
- Challenge the Port Partner or Cable Plug.

Certificates are used to convey information, attested to by a signer, which attests to the Port Partner's or Cable Plug's authenticity. The Port's or Cable Plug's certificates are needed when a Port encounters a Port Partner or Cable Plug it has not been Attached to before. To minimize calculations after the initial Attachment, a Port can also use a digest consisting of hashes of the certificates rather than the certificates themselves. Once the port has the certificates and has calculated the hashes, it stores the hashes and uses

the digest in future exchanges. After the port gets the certificates or digest, it challenges its Port Partner or the Cable Plug to detect replay attacks.

For further details refer to ***[USBTypeCAuthentication 1.0]***.

6.5.8.1 Security_Request

The ***Security_Request*** Message is used by a Port to pass a security data structure to its Port Partner or a Cable Plug.

The ***Security_Request*** Message contains a Security Request Data Block (SRQDB) whose format ***Shall*** be as shown in Figure 6-40. The contents of the SRQDB and its use are defined in ***[USBTypeCAuthentication 1.0]***.

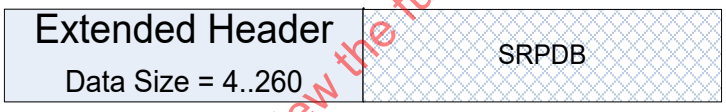
Figure 6-40 Security_Request Message

6.5.8.2 Security_Response

The ***Security_Response*** Message is used by a Port or Cable Plug to pass a security data structure to the Port that sent the ***Security_Request*** Message.

The ***Security_Response*** Message contains a Security Response Data Block (SRPDB) whose format ***Shall*** be as shown in Figure 6-41. The contents of the SRPDB and its use are defined in ***[USBTypeCAuthentication 1.0]***.

Figure 6-41 Security_Response Message



6.5.9 Firmware Update Messages

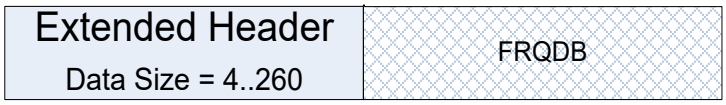
The firmware update process between Port Partners or a Port and Cable Plug is fully described in ***[USBPDFirmwareUpdate 1.0]***. This specification describes two Extended Messages used by the firmware update process when applied to PD.

6.5.9.1 Firmware_Update_Request

The ***Firmware_Update_Request*** Message is used by a Port to pass a firmware update data structure to its Port Partner or a Cable Plug.

The ***Firmware_Update_Request*** Message contains a Firmware Update Request Data Block (FRQDB) whose format ***Shall*** be as shown in Figure 6-42. The contents of the FRQDB and its use are defined in ***[USBPDFirmwareUpdate 1.0]***.

Figure 6-42 Firmware_Update_Request Message

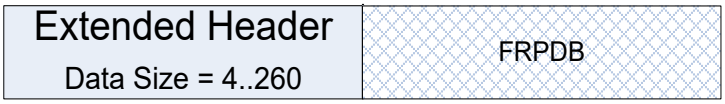


6.5.9.2 Firmware_Update_Response

The ***Firmware_Update_Response*** Message is used by a Port or Cable Plug to pass a firmware update data structure to the Port that sent the ***Firmware_Update_Request*** Message.

The ***Firmware_Update_Response*** Message contains a Firmware Update Response Data Block (FRPDB) whose format ***Shall*** be as shown in Figure 6-43. The contents of the FRPDB and its use are defined in ***[USBPDFirmwareUpdate 1.0]***.

Figure 6-43 Firmware_Update_Response Message



6.5.10 PPS_Status Message

The ***PPS_Status*** Message ***Shall*** be sent in response to a ***Get_PPS_Status*** Message. The ***PPS_Status*** Message enables a Sink to query the Source to get additional information about its operational state. The ***Get_PPS_Status*** Message and the ***PPS_Status*** Message ***Shall*** only be supported when the ***Alert*** Message is also supported.

The ***PPS_Status*** Message ***Shall*** return a 4-byte PPS Status Data Block (PPSSDB) whose format ***Shall*** be as shown in Figure 6-44 and Table 6-57.

Figure 6-44 PPS_Status Message

Table 6-57 PPS Status Data Block (PPSSDB)

Offset	Field	Size	Description
0	Output Voltage	2	Source's output voltage in 20mV units. When set to 0xFFFF, the Source does not support this field.
2	Output Current	1	Source's output current in 50mA units. When set to 0xFF, the Source does not support this field.
3	Real Time Flags	1	**Bit** / **Description** 0 – ***Reserved*** and ***Shall*** be set to zero 1…2 – PTF: 00 – Not Supported PTF: 01 – Normal PTF: 10 – Warning PTF: 11 – Over temperature 3 – OMF set when operating in Current Limit mode and cleared when operating in Constant Voltage mode. 4…7 – ***Reserved*** and ***Shall*** be set to zero

6.5.10.1 Output Voltage Field

The Output Voltage field ***Shall*** return the Source's output voltage at the time of the request. The output voltage is measured either at the Source's receptacle or, if the Source has a captive cable, where the voltage is applied to the cable.

The measurement accuracy ***Shall*** be +/-3% rounded to the nearest 20mV.

If the Source does not support the Output Voltage field, the field ***Shall*** be set to 0xFFFF.

6.5.10.2 Output Current Field

The Output Current field ***Shall*** return the Source's output current at the time of the request measured at the Source's receptacle.

The measurement accuracy ***Shall*** be +/-150mA.

If the Source does not support the Output Current field, the field ***Shall*** be set to 0xFF.

6.5.10.3 Real Time Flags Field

Real Time flags provide a real-time indication of the Source's operating state.

- The PTF (Present Temperature Flag) ***Shall*** provide a real-time indication of the Source's internal thermal status. If the PTF is not supported, it will be set to zero.
  - Normal indicates that that the Source is operating within its normal thermal envelope.
  - Warning indicates that the Source is over-heating but is not in imminent danger of shutting down.
  - Over Temperature indicates that the Source is over heated and will shut down soon or has already shutdown and has sent an OTP in an ***Alert*** Message.
- The OMF (Operating Mode Flag) ***Shall*** provide a real-time indication of the Source's operating mode. When set, the Source is operating in Current Limit mode; when cleared it is operating Constant Voltage mode.

6.5.11 Country_Codes Message

The ***Country_Codes*** Message ***Shall*** be sent in response to a ***Get_Country_Codes*** Message. The ***Country_Codes*** Message enables a Port to query its Port partner to get a list of alpha-2 country codes as defined in ***[ISO 3166]*** for which the Port Partner has country specific information.

The Country_Codes Message ***Shall*** contain a 4…26-byte Country Code Data Block (CCDB) whose format ***Shall*** be as shown in Figure 6-45 and Table 6-58.

Figure 6-45 Country_Codes Message

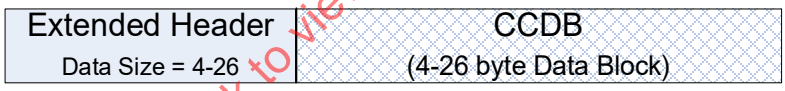


Table 6-58 Country Codes Data Block (CCDB)

Offset	Field	Description
0	Length	Number of country codes in the message
1	***Reserved***	***Shall*** be set to 0.
2	1st Country Code	First character of the Alpha-2 Country Code defined by ***[ISO 3166]***
3		Second character of the Alpha-2 Country Code defined by ***[ISO 3166]***
4	2nd Country Code	First character of the Alpha-2 Country Code defined by ***[ISO 3166]***
5		Second character of the Alpha-2 Country Code defined by ***[ISO 3166]***
	…	
Length * 2n	nth Country Code	

6.5.11.1 Country Code Field

The Country Code field ***Shall*** contain the Alpha-2 Country Code defined by ***[ISO 3166]***.

6.5.12 Country_Info Message

The ***Country_Info*** Message ***Shall*** be sent in response to a ***Get_Country_Info*** Message. The ***Country_Info*** Message enables a Port to get additional country specific information from its Port Partner.

The ***Country_Info*** Message ***Shall*** contain a 4-26 byte Country Info Data Block (CIDB) whose format ***Shall*** be as shown in Figure 6-46 and Table 6-59.

Figure 6-46 Country_Info Message

Table 6-59 Country Info Data Block (CIDB)

Offset	Field	Description
0	Country Code	First character of the Alpha-2 Country Code received in the corresponding ***Get_Country_Info*** Message.
1		Second character of the Alpha-2 Country Code received in the corresponding ***Get_Country_Info*** Message
2…3	***Reserved***	***Shall*** be set to 0.
4	Country Specific Data	0…22 bytes of content defined by the country's authority.

6.5.12.1 Country Code Field

The Country Code field ***Shall*** contain the Alpha-2 Country Code received in the corresponding ***Get_Country_Info*** Message.

6.5.12.2 Country Specific Data Field

The Country Specific Data field ***Shall*** contain content defined by and formatted in a manner determined by an official agency of the country indicated in the Country Code field.

If the Country Code field in the ***Get_Country_Info*** Message is unrecognized the Country Specific Data field ***Shall*** return the null terminated ascii text string "Unsupported Country Code".

6.5.13 Sink_Capabilities_Extended Message

The ***Sink_Capabilities_Extended*** Message ***Shall*** be sent in response to a ***Get_Sink_Cap_Extended*** Message. The ***Sink_Capabilities_Extended*** Message enables a Sink or a DRP to inform the Source about its capabilities as a Sink.

The ***Sink_Capabilities_Extended*** Message ***Shall*** return a 21-byte Sink Capabilities Extended Data Block (SKEDB) whose format ***Shall*** be as shown in Figure 6-46 and Table 6-60.

Figure 6-47 Sink_Capabilities_Extended Message

Table 6-60 Sink Capabilities Extended Data Block (SKEDB)

Offset	Field	Size	Value	Description
0	VID	2	Numeric	Vendor ID (assigned by the USB-IF)
2	PID	2	Numeric	Product ID (assigned by the manufacturer)
4	XID	4	Numeric	Value provided by the USB-IF assigned to the product
8	FW Version	1	Numeric	Firmware version number
9	HW Version	1	Numeric	Hardware version number
10	SKEDB Version	1	Numeric	SKEDB Version (not the specification Version): Version 1.0 = 1 Values 0 and 2-255 are ***Reserved*** and ***Shall Not*** be used
11	Load Step	1	Bit Field	**Bit** / **Description** 1…0: 00b: 150mA/µs Load Step (default) 01b: 500mA/µs Load Step 11b…10b: ***Reserved*** and ***Shall Not*** be used 2…7: ***Reserved*** and ***Shall*** be set to zero
12	Sink Load Characteristics	2	Bit field	**Bit** / **Description** 0…4: Percent overload in 10% increments Values higher than 25 (11001b) are clipped to 250%. 00000b is the default. 5…10: Overload period in 20ms when bits 0-4 non-zero. 11.14: Duty cycle in 5% increments when bits 0-4 are non-zero 15: Can tolerate V_{BUS} Voltage droop
14	Compliance	1	Bit Field	**Bit** / **Description** 0: Requires LPS Source when set 1: Requires PS1 Source when set 2: Requires PS2 Source when set 3…7: ***Reserved*** and ***Shall*** be set to zero
15	Touch Temp	1	Value	Temperature conforms to: 0 = Not applicable 1 = ***[IEC 60950-1]*** (default) 2 = ***[IEC 62368-1]*** TS1 3 = ***[IEC 62368-1]*** TS2 Note: All other values ***Reserved***
16	Battery Info	1	Byte	Upper Nibble = Number of Hot Swappable Battery Slots (0…4) Lower Nibble = Number of Fixed Batteries (0…4)
17	Sink Modes	1	Bit field	**Bit** / **Description** 0: 1: PPS charging supported 1: 1: V_{BUS} powered 2: 1: Mains powered 3: 1: Battery powered 4: 1: Battery essentially unlimited 5…7: ***Reserved*** and ***Shall*** be set to zero

Offset	Field	Size	Value	Description
18	Sink Minimum PDP	1	Byte	**Bit** / **Description** 0…6: The Minimum PDP required by the Sink to operate without consuming any power from its Battery(s) should it have one. 7: ***Reserved*** and ***Shall*** be set to zero
19	Sink Operational PDP	1	Byte	**Bit** / **Description** 0…6: The PDP the Sink requires to operate normally. For Sinks with a Battery, it is the PDP Rating of the charger supplied with it or recommended for it. 7: ***Reserved*** and ***Shall*** be set to zero
20	Sink Maximum PDP	1	Byte	**Bit** / **Description** 0…6: The Maximum PDP the Sink can consume to operate and charge its Battery(s) should it have one. 7: ***Reserved*** and ***Shall*** be set to zero

6.5.13.1 Vendor ID (VID) Field

The Vendor ID field ***Shall*** contain the 16-bit Vendor ID (VID) assigned to the Sink's vendor by the USB-IF. If the vendor does not have a VID, the Vendor ID field Shall be set to zero. Devices that have a USB data interface ***Shall*** report the same VID as the idVendor in the Standard Device Descriptor (see ***[USB 2.0]*** and ***[USB 3.2]***).

6.5.13.2 Product ID (PID) Field

The Product ID field ***Shall*** contain the 16-bit Product ID (PID) assigned by the Sink's vendor. Devices that have a USB data interface ***Shall*** report the same PID as the idProduct in the Standard Device Descriptor (see ***[USB 2.0]*** and ***[USB 3.2]***).

6.5.13.3 XID Field

The XID field ***Shall*** contain the 32-bit XID provided by the USB-IF to the vendor who in turns assigns it to a product. If the vendor does not have an XID, then it ***Shall*** return zero in this field (see ***[USB 2.0]*** and ***[USB 3.2]***).

6.5.13.4 Firmware Version Field

The Firmware Version field ***Shall*** contain an 8-bit firmware version number assigned to the device by the vendor.

6.5.13.5 Hardware Version Field

The Hardware Version field ***Shall*** contain an 8-bit hardware version number assigned to the device by the vendor.

6.5.13.6 SKEDB Version Field

The SKEDB version field contains version level of the SKEDB. Currently only Version 1 is defined.

6.5.13.7 Load Step Field

The Voltage Regulation field contains bits indicating the Load Step Slew Rate and Magnitude that this Sink prefers. See Section 7.1.12.1 for further details.

6.5.13.8 Sink Load Characteristics Field

The Sink ***Shall*** report its preferred load characteristics. Regardless of this value, in operation its load ***Shall Not*** exceed the capabilities reported in the ***Source_Capabilities_Extended*** message.

6.5.13.9 Compliance Field

The Compliance field ***Shall*** contain the types of Sources the Sink has been tested and certified with (see Section 7.1.12.3).

6.5.13.10 Touch Temp

The Touch Temp field ***Shall*** report the IEC standard used to determine the surface temperature of the Sink's enclosure. Safety limits for the Sink's touch temperature are set in applicable product safety standards (e.g. ***[IEC 60950-1]*** or ***[IEC 62368-1]***). The Sink ***May*** report when its touch temperature performance conforms to the TS1 or TS2 limits described in ***[IEC 62368-1]***.

6.5.13.11 Battery Info

The Batteries Info field ***Shall*** report the number of Fixed Batteries and Hot Swappable Battery Slots the Sink supports. This field ***Shall*** independently report the number of Battery Slots and the number of Fixed Batteries. The information reported in the Battery Info field ***Shall*** match that reported in the Battery Info field of the ***Source_Capabilities_Extended*** Message.

A Sink ***Shall*** have no more than 4 Fixed Batteries and no more than 4 Battery Slots.

Fixed Batteries ***Shall*** be numbered consecutively from 0 to 3. The number assigned to a given Fixed Battery ***Shall Not*** change between Attach and Detach.

Battery Slots ***Shall*** be numbered consecutively from 4 to 7. The number assigned to a given Battery Slot ***Shall Not*** change between Attach and Detach.

6.5.13.12 Sink Modes

The Sink Modes bit field ***Shall*** identify the charging capabilities and the power sources that can be used by the Sink. When bit 0 is set, the Sink has the ability to use a PPS Source for fast charging.

The source of power a Sink can use:

- When bit 1 is set, the Sink has the ability to be sourced by Vbus.
- When bit 2 is set, the Sink has the ability to be sourced by an external mains power supply.
- When bit 3 is set, the Sink has the ability to be sourced by a battery.
- When bit 4 is set, the Sink has the ability to be sourced by a battery with essentially infinite energy (e.g. a car battery).

Bits 1-4 ***May*** be set independently of one another. The combination indicates what sources of power the Sink can utilize. For example, some Sinks are only powered by a Battery (e.g. an automobile battery) rather than the more common mains and some Sinks are only powered from VBUS or VCONN.

6.5.13.13 Sink Minimum PDP

The Sink Minimum PDP field ***Shall*** contain the minimum power required by the Sink, rounded up to the next integer, to operate all its functional modes except charging its battery if present. The Sink Minimum PDP field ***Shall*** be less than or equal to the Sink Operational PDP. The value is used by the Source to determine whether or not it has sufficient power to minimally support the attached Sink.

6.5.13.14 Sink Operational PDP

The Sink Operational PDP field ***Shall*** contain the manufacturer recommended PDP of the Sink, rounded up to the next integer. This corresponds to the PDP Rating of Sources that the Sink is designed to operate with (See Section 10.3.2). The Sink Operational PDP ***Shall*** be sufficient to operate all the Sink's functional modes normally AND charge the Sink's battery if present. For Sinks with a battery(s), it ***Shall*** correspond to the PDP Rating of the charger shipped with the Sink or the recommended charger's PDP Rating.

6.5.13.15 Sink Maximum PDP

The Sink Maximum PDP ***Shall*** be highest amount of power the Sink consumes under any operating condition, rounded up to the next integer, including charging its battery if present. The Sink Maximum PDP field ***Shall Not*** be less than the Sink Operational PDP, but ***May*** be the same. The value is used by the Source to determine the maximum amount of power it has to budget for the attached Sink.

6.6 Timers

All the following timers are defined in terms of bits on the bus regardless of where they are implemented in terms of the logical architecture. This is to ensure a fixed reference for the starting and stopping of timers. It is left to the implementer to ensure that this timing is observed in a real system.

6.6.1 CRCReceiveTimer

The ***CRCReceiveTimer*** ***Shall*** be used by the sender's Protocol Layer to ensure that a Message has not been lost. Failure to receive an acknowledgement of a Message (a ***GoodCRC*** Message) whether caused by a bad CRC on the receiving end or by a garbled Message within ***tReceive*** is detected when the ***CRCReceiveTimer*** expires.

The sender's Protocol Layer response when a ***CRCReceiveTimer*** expires ***Shall*** be to retry ***nRetryCount*** times. Note: that Cable Plugs do not retry Messages and large Extended Messages that are not Chunked are not retried (see Section 6.7.2). Sending of the Preamble corresponding to the retried Message ***Shall*** start within ***tRetry*** of the ***CRCReceiveTimer*** expiring.

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

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

6.6.2 SenderResponseTimer

The ***SenderResponseTimer*** ***Shall*** be used by the sender's Policy Engine to ensure that a Message requesting a response (e.g. ***Get_Source_Cap*** Message) is responded to within a bounded time of ***tSenderResponse***. Failure to receive the expected response is detected when the ***SenderResponseTimer*** expires.

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.

6.6.3 Capability Timers

Sources and Sinks use Capability Timers to determine Attachment of a PD Capable device. By periodically sending or requesting capabilities it is possible to determine PD device Attachment when a response is received.

6.6.3.1 SourceCapabilityTimer

Prior to a successful negotiation a Source ***Shall*** use the ***SourceCapabilityTimer*** to periodically send out a ***Source_Capabilities*** Message every ***tTypeCSendSourceCap*** while:

- The Port is Attached.
- The Source is not in an active connection with a PD Sink Port.

Whenever there is a ***SourceCapabilityTimer*** timeout the Source ***Shall*** send a ***Source_Capabilities*** Message. It ***Shall*** then re-initialize and restart the ***SourceCapabilityTimer***. The ***SourceCapabilityTimer*** ***Shall*** be stopped when the last bit of the ***EOP*** corresponding to the ***GoodCRC*** Message has been received by the Physical Layer since a PD connection has been established. At this point the Source waits for a ***Request*** Message or a response timeout.

See Section 8.3.3.2 more details of when ***Source_Capabilities*** Messages are transmitted.

6.6.3.2 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*** the Sink ***Shall*** issue ***Hard Reset*** Signaling in order to restart the sending of ***Source_Capabilities*** Messages by the Source (see Section 6.7.4).

See Section 8.3.3.3 for more details of when the ***SinkWaitCapTimer*** are run.

6.6.3.3 tFirstSourceCap

After Port Partners are Attached or after a Hard Reset or after a Power Role Swap or after a Fast Role Swap a Source ***Shall*** send its first ***Source_Capabilities*** Message within ***tFirstSourceCap*** of V_{BUS} reaching ***vSafe5V***. This ensures that the Sink receives a ***Source_Capabilities*** Message before the Sink's ***SinkWaitCapTimer*** expires.

6.6.4 Wait Timers and Times

6.6.4.1 SinkRequestTimer

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

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

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

6.6.4.2 tPRSwapWait

The time before the next ***PR_Swap*** Message, after a ***Wait*** Message has been received in response to a ***PR_Swap*** Message is a minimum of ***tPRSwapWait*** min (see 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.

6.6.4.3 tDRSwapWait

The time before the next ***DR_Swap*** Message, after a ***Wait*** Message has been received in response to a ***DR_Swap*** Message is a minimum of ***tDRSwapWait*** min (see 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.

6.6.4.4 tVconnSwapWait

The time before the next ***VCONN_Swap*** Message, after a ***Wait*** Message has been received in response to a ***VCONN_Swap*** Message is a minimum of ***tVCONNSwapWait*** min (see 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.

6.6.5 Power Supply Timers

6.6.5.1 PSTransitionTimer

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

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.

6.6.5.2 PSSourceOffTimer

6.6.5.2.1 Use during Power Role Swap

The ***PSSourceOffTimer*** is used by the Policy Engine in Dual-Role Power Device that is currently acting as a Sink to timeout on a ***PS_RDY*** Message during a Power Role Swap sequence. This condition leads to USB Type-C Error Recovery.

If a ***PR_Swap*** Message request has been sent by the Dual-Role Power Device currently acting as a Source the Sink can respond with an ***Accept*** Message. When the last bit of the ***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.

6.6.5.2.2 Use during Fast Role Swap

The ***PSSourceOffTimer*** is used by the Policy Engine in Dual-Role Power Device that is the initial Sink (currently providing ***vSafe5V***) to timeout on a ***PS_RDY*** Message during a Fast Role Swap sequence. This condition leads to USB Type-C Error Recovery.

When the ***FR_Swap*** Message request has been sent by the initial Sink, the initial Source ***Shall*** respond with an ***Accept*** Message. When the last bit of the ***EOP*** of the ***GoodCRC*** Message corresponding to this ***Accept*** Message is received by the initial 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 initial Source to stop supplying power and for V_{BUS} to revert to ***vSafe5V*** (see also Section 7.2.10 and Section 7.3.15). The timer ***Shall*** time out if a ***PS_RDY*** Message has not been received from the initial Source within ***tPSSourceOff*** indicating this has occurred.

6.6.5.3 PSSourceOnTimer

6.6.5.3.1 Use during Power Role Swap

The ***PSSourceOnTimer*** is used by the Policy Engine in Dual-Role Power Device that has just stopped sourcing power and is waiting to start sinking power to timeout on a ***PS_RDY*** Message during a Power Role Swap. This condition leads to USB Type-C Error Recovery.

The ***PSSourceOnTimer*** ***Shall*** be started when:

- The last bit of the ***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***.

6.6.5.3.2 Use during Fast Role Swap

The ***PSSourceOnTimer*** is used by the Policy Engine in Dual-Role Power Device that has just stopped sourcing power and is waiting to start sinking power to timeout on a ***PS_RDY*** Message during a Fast Role Swap. This condition leads to USB Type-C Error Recovery.

The ***PSSourceOnTimer*** ***Shall*** be started when:

- The last bit of the ***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 7.2.10 and Section 7.3.15) and will time out if a ***PS_RDY*** Message indicating this has not been received within ***tPSSourceOn***.

6.6.6 NoResponseTimer

The ***NoResponseTimer*** is used by the Policy Engine in a Source to determine that its Port Partner is not responding after a Hard Reset. When the ***NoResponseTimer*** times out, the Policy Engine ***Shall*** issue up to ***nHardResetCount*** additional Hard Resets before determining that the Port Partner is non-responsive to USB Power Delivery messaging.

If the Source fails to receive a ***GoodCRC*** Message in response to a ***Source_Capabilities*** Message within ***tNoResponse*** of:

- The last bit of a ***Hard Reset*** Signaling being sent by the PHY Layer if the ***Hard Reset*** Signaling was initiated by the Sink.
- The last bit of a ***Hard Reset*** Signaling being received by the PHY Layer if the ***Hard Reset*** Signaling was initiated by the Source.

Then the Source ***Shall*** issue additional Hard Resets up to ***nHardResetCount*** times (see Section 6.8.3).

For a non-responsive device, the Policy Engine in a Source ***May*** either decide to continue sending ***Source_Capabilities*** Messages or to go to non-USB Power Delivery operation and cease sending ***Source_Capabilities*** Messages.

6.6.7 BIST Timers

6.6.7.1 tBISTCarrierMode

tBISTCarrierMode is used to define the maximum time that a UUT has to enter BIST Carrier Mode when requested by a Tester.

A UUT ***Shall*** enter BIST Carrier Mode within ***tBISTCarrierMode*** of the last bit of the ***EOP*** of the ***BIST*** Message used to initiate the test is received by the Physical Layer. In ***BIST Carrier Mode*** when transmitting a continuous carrier signal transmission ***Shall*** start as soon as the UUT enters BIST mode.

6.6.7.2 BISTContModeTimer

The ***BISTContModeTimer*** is used by a UUT to ensure that a Continuous BIST Mode (i.e. ***BIST Carrier Mode***) is exited in a timely fashion. A UUT that has been put into a Continuous BIST Mode ***Shall*** return to normal operation (either ***PE_SRC_Transition_to_default***, ***PE_SNK_Transition_to_default***, or ***PE_CBL_Ready***) within ***tBISTContMode*** of starting to transmit a continuous carrier signal.

6.6.7.3 tBISTSharedTestMode

tBISTSharedTestMode is used to define the maximum time that a UUT has to enter BIST Shared Capacity Test Mode when requested by a Tester.

A UUT ***Shall*** enter BIST Shared Capacity Test Mode and send a new ***Source_Capabilities*** Message from all Ports within the shared capacity group within ***tBISTSharedTestMode*** of the last bit of the ***EOP*** of the ***BIST*** Message used to initiate the test is received by the Physical Layer.

6.6.8 Power Role Swap Timers

6.6.8.1 SwapSourceStartTimer

The ***SwapSourceStartTimer*** ***Shall*** be used by the new Source, after a Power Role Swap or Fast Role Swap, to ensure that it does not send ***Source_Capabilities*** Message before the new Sink is ready to receive the ***Source_Capabilities*** Message. The new Source ***Shall Not*** send the ***Source_Capabilities*** Message earlier than ***tSwapSourceStart*** after the last bit of the ***EOP*** of ***GoodCRC*** Message sent in response to the ***PS_RDY*** Message sent by the new Source indicating that its power supply is ready. The Sink ***Shall*** be ready to receive a ***Source_Capabilities*** Message ***tSwapSinkReady*** after having sent the last bit of the ***EOP*** of ***GoodCRC*** Message sent in response to the ***PS_RDY*** Message sent by the new Source indicating that its power supply is ready.

6.6.9 Soft Reset Timers

6.6.9.1 tSoftReset

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

6.6.9.2 tProtErrSoftReset

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

6.6.10 Data Reset Timers

6.6.10.1 VCONNDischargeTimer

The ***VCONNDischargeTimer*** is used by the Policy Engine in the DFP to ensure the UFP actively discharges VCONN in a timely manner to ensure the cable will restore Ra. Once the UFP has discharged VCONN below vRaReconnect (see ***[USB Type-C 2.0]***) it sends a ***PS_RDY*** Message (see also Section 7.1.15.1).

If the DFP does not receive a ***PS_RDY*** Message from the UFP within ***tVCONNSourceDischarge*** of the last bit of the ***GoodCRC*** acknowledging the ***Accept*** message in response to the ***Data_Reset*** Message, the ***VCONNDischargeTimer*** will time out and the Policy Engine ***Shall*** enter the ***ErrorRecovery*** State.

6.6.10.2 tDataReset

The DFP ***Shall*** complete the Data_Reset process (as defined in Section 6.3.14) within ***tDataReset*** of either:

- The last bit of the ***GoodCRC*** acknowledging the ***Accept*** Message when the DFP sent the ***Data_Reset*** Message.
- The last bit of the ***Accept*** Message when the UFP sent the ***Data_Reset*** Message.

6.6.10.3 DataResetFailTimer

The ***DataResetFailTimer*** ***Shall*** be used by the DFP's Policy Engine to ensure the Data Reset process completes within ***tDataResetFail*** of the last bit of the ***GoodCRC*** acknowledging the ***Accept*** Message in response to the ***Data_Reset*** Message. If the DFP's ***DataResetFailTimer*** expires, the DFP ***Shall*** enter the ***ErrorRecovery*** State.

6.6.11 Hard Reset Timers

6.6.11.1 HardResetCompleteTimer

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

6.6.11.2 PSHardResetTimer

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

When a Hard Reset occurs the Source, stops driving VCONN, removes Rp from the VCONN pin and starts to transition the VBUS voltage to ***vSafe0V*** either:

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

See Section 7.1.5.

6.6.11.3 tDRSwapHardReset

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

6.6.11.4 tProtErrHardReset

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

6.6.12 Structured VDM Timers

6.6.12.1 VDMResponseTimer

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

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

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.

6.6.12.2 VDMModeEntryTimer

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

The Policy Engine's response when the ***VDMModeEntryTimer*** expires is to inform the Device Policy Manager (see Section 8.3.3.22.1).

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.

6.6.12.3 VDMModeExitTimer

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

The Policy Engine's response when the ***VDMModeExitTimer*** expires is to inform the Device Policy Manager (see Section 8.3.3.22.2).

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.

6.6.12.4 tVDMBusy

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

6.6.13 VCONN Timers

6.6.13.1 VCONNOnTimer

The ***VCONNOnTimer*** is used during a VCONN Swap.

The ***VCONNOnTimer*** ***Shall*** be started when:

- The last bit of the ***EOP*** of the ***Accept*** Message is received.
- The last bit of the ***EOP*** of ***GoodCRC*** Message corresponding to the ***Accept*** Message is received.

The ***VCONNOnTimer*** ***Shall*** be stopped when:

- The last bit of the ***EOP*** of the ***PS_RDY*** Message is received.

Prior to sending the ***PS_RDY*** Message, the Port ***Shall*** have turned VCONN On.

6.6.13.2 tVCONNSourceOff

The ***tVCONNSourceOff*** time applies during a Vconn Swap. The initial VCONN Source ***Shall*** cease sourcing VCONN within ***tVCONNSourceOff*** of receipt of the last bit of the ***EOP*** of the ***PS_RDY*** Message.

6.6.14 tCableMessage

Ports compliant with this Revision of the specification ***Shall Not*** wait ***tCableMessage*** before sending an SOP' or SOP'' Packet even when communicating using ***[USBPD 2.0]*** with a Cable Plug. This specification defines collision avoidance mechanisms that obviate the need for this time.

Cable Plugs ***Shall*** only wait ***tCableMessage*** before sending an SOP' or SOP'' Packet when operating at ***[USBPD 2.0]***. When operating at Revisions higher than ***[USBPD 2.0]*** Cable Plugs ***Shall Not*** wait ***tCableMessage*** before sending an SOP' or SOP'' Packet.

6.6.15 DiscoverIdentityTimer

The ***DiscoverIdentityTimer*** is used during an Explicit Contract when discovering whether a Cable Plug is PD Capable using SOP'. When performing cable discovery during an Explicit Contract the ***Discover Identity*** Command request ***Shall*** be sent every ***tDiscoverIdentity***. No more than ***nDiscoverIdentityCount Discover Identity*** Messages without a ***GoodCRC*** Message response ***Shall*** be sent. If no ***GoodCRC*** Message response is received after ***nDiscoverIdentityCount Discover Identity*** Command requests have been sent by a Port, the Port ***Shall Not*** send any further SOP'/SOP'' Messages.

6.6.16 Collision Avoidance Timers

The ***SinkTxTimer*** is used by the Protocol Layer in a Source to allow the Sink to complete its transmission before initiating an AMS.

The Source ***Shall*** wait a minimum of ***tSinkTx*** after changing Rp from ***SinkTxOk*** to ***SinkTxNG*** before initiating an AMS by sending a Message.

A Sink ***Shall*** only initiate an AMS when it has determined that Rp is set to ***SinkTxOk***.

6.6.17 Fast Role Swap Timers

6.6.17.1 tFRSwap5V

During a Fast Role Swap, the initial Source ***Shall*** start the ***PS_RDY*** Message within ***tFRSwap5V*** after it has sent the ***Accept*** Message and V_{BUS} is at ***vSafe5V***. The ***tFRSwap5V*** time Shall be measured from the later of the last bit of the ***EOP*** for the ***GoodCRC*** Message corresponding to the ***Accept*** message and V_{BUS} being within ***vSafe5V***, until the first bit of the response ***PS_RD*** Message Preamble has been transmitted by the Physical Layer.

6.6.17.2 tFRSwapComplete

During a fast-role swap, the initial Sink ***Shall*** respond with a the ***PS_RDY*** Message within ***tFRSwapComplete*** after it has received the ***PS_RDY*** Message from the Initial Source. The ***tFRSwapComplete*** time Shall be measured from the time the last bit of the ***PS_RDY*** Message ***EOP*** has been received by the Physical Layer until the first bit of the response ***PS_RD*** Message Preamble has been transmitted by the Physical Layer.

6.6.17.3 tFRSwapInit

That last bit of the ***EOP*** of the ***FR_Swap*** Message ***Shall*** be transmitted by the new Source no later than ***tFRSwapInit*** after the Fast Role Swap Request has been detected (see Section 5.8.6.3).

6.6.18 Chunking Timers

6.6.18.1 ChunkingNotSupportedTimer

The ***ChunkingNotSupportedTimer*** is used by a Source or Sink which does not support multi-chunk Chunking but has received a Message Chunk.

The ***ChunkingNotSupportedTimer*** ***Shall*** be started when:

- The last bit of the EOP of a Message Chunk of a multi-chunk Message is received. The Policy Engine ***Shall Not*** send its Not_Supported Message before the ChunkingNotSupportedTimer expires.

6.6.18.2 ChunkSenderRequestTimer

The ***ChunkSenderRequestTimer*** is used during a Chunked Message transmission.

The ***ChunkSenderRequestTimer*** ***Shall*** be used by the sender's Chunking state machine to ensure that a Chunk Response is responded to within a bounded time of ***tChunkSenderRequest***. Failure to receive the expected response is detected when the ***ChunkSenderRequestTimer*** expires.

The ***ChunkSenderRequestTimer*** ***Shall*** be started when:

- The last bit of the ***EOP*** of the ***GoodCRC*** Message corresponding to the Chunk Response Message is received.

The ***ChunkSenderRequestTimer*** ***Shall*** be stopped when:

- The last bit of the ***EOP*** of the Chunk Request Message is received.
- A Message other than a Chunk Request is received from the Protocol Layer Rx.

The receiver of a Chunk Response requiring a Chunk Request ***Shall*** respond with a Chunk Request within ***tChunkReceiverRequest*** in order to ensure that the sender's ***ChunkSenderRequestTimer*** does not expire.

The *tChunkReceiverRequest* time ***Shall*** be measured from the time the last bit of the Message *EOP* has been received by the Physical Layer until the first bit of the response Message Preamble has been transmitted by the Physical Layer.

6.6.18.3 ChunkSenderResponseTimer

The *ChunkSenderResponseTimer* is used during a Chunked Message transmission.

The *ChunkSenderResponseTimer* ***Shall*** be used by the sender's Chunking state machine to ensure that a Chunk Request is responded to within a bounded time of *tChunkSenderResponse*. Failure to receive the expected response is detected when the ChunkSenderResponseTimer expires.

The *ChunkSenderResponseTimer* ***Shall*** be started when:

- The last bit of the EOP of GoodCRC Message corresponding to the Chunk Request Message is received.

The *ChunkSenderResponseTimer* ***Shall*** be stopped when:

- The last bit of the EOP of the Chunk Response Message is received.
- A Message other than a Chunk is received from the Protocol Layer.

The receiver of a Chunk Request requiring a Chunk Response ***Shall*** respond with a Chunk Response within *tChunkReceiverResponse* in order to ensure that the sender's *ChunkSenderResponseTimer* does not expire.

The *tChunkReceiverResponse* time ***Shall*** be measured from the time the last bit of the Message *EOP* has been received by the Physical Layer until the first bit of the response Message Preamble has been transmitted by the Physical Layer.

6.6.19 Programmable Power Supply Timers

6.6.19.1 SinkPPSPeriodicTimer

The *SinkPPSPeriodicTimer* ***Shall*** be used by the Sink's Policy Engine to ensure that communication between the Sink and Source occurs within a bounded time of *tPPSRequest* when in PPS operation. In the absence of any other traffic, a *Request* Message requesting a PPS APDO is sent periodically as a keep alive mechanism.

SinkPPSPeriodicTimer ***Shall*** be re-initialized and restarted when the last bit of the *EOP* of any Message is received that causes the Sink to enter the *PE_SNK_Ready* state.

The Sink ***Shall*** stop the *SinkPPSPeriodicTimer* when the last bit of the *EOP* of any Message or the last bit of any Signaling is received from the Source and by the Sink that causes the Sink to leave the *PE_SNK_Ready* state.

6.6.19.2 SourcePPSCommTimer

The *SourcePPSCommTimer* ***Shall*** be used by the Source's Policy Engine to ensure that communication between the Sink and Source occurs within a bounded time of *tPPSTimeout* when in PPS operation. In the absence of any other traffic, a *Request* Message requesting a PPS APDO is received periodically as a keep alive mechanism.

SourcePPSCommTimer ***Shall*** be re-initialized and restarted when the last bit of the *EOP* of any Message is received that causes the Source to enter the *PE_SRC_Ready* state.

The Source ***Shall*** stop the *SourcePPSCommTimer* when the last bit of the *EOP* of any Message or the last bit of any Signaling is received from the Sink by the Source that causes the Source to leave the *PE_SRC_Ready* state.

When the *SourcePPSCommTimer* times out the Source ***Shall*** issue *Hard Reset* Signaling.

6.6.20 tEnterUSB

The DFP ***Shall*** send the *Enter_USB* Message within *tEnterUSB* of either:

- The last bit of the ***GoodCRC*** acknowledging the ***Accept*** Message in response to the ***Data_Reset*** Message or
- Initial power-on.

Failure to meet this timeout parameter may result in the ports not transitioning into ***[USB4]*** operation.

6.6.21 Time Values and Timers

Table 6-61 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 6-62 lists the timers.

Table 6-61 Time Values

Parameter	Value (min)	Value (max)	Units	Reference
tACTempUpdate		500	ms	Section 6.5.2.1.1
tBISTContMode	30	60	ms	Section 6.6.7.2
tBISTCarrierMode		300	ms	Section 6.6.7.1
tBISTSharedTestMode		1	s	Section 6.6.7.3
tCableMessage	750		µs	Section 6.6.14
tChunkingNotSupported	40	50	ms	Section 6.6.18.1
tChunkReceiverRequest		15	ms	Section 6.6.18.2
tChunkReceiverResponse		15	ms	Section 6.6.18.3
tChunkSenderRequest	24	30	ms	Section 6.6.18.2
tChunkSenderResponse	24	30	ms	Section 6.6.18.3
tDataReset	200	250	ms	Section 6.6.10.2
tDataResetFail	300		ms	Section 6.6.10.3
tDiscoverIdentity	40	50	ms	Section 6.6.14
tDRSwapHardReset		15	ms	Section 6.6.11.3
tDRSwapWait	100		ms	Section 6.6.4.3
tEnterUSB		500	ms	Section 6.6.20
tFirstSourceCap		250	ms	Section 6.6.3.3
tFRSwap5V		15	ms	Section 6.6.17.1
tFRSwapComplete		15	ms	Section 6.6.17.2
tFRSwapInit		15	ms	Section 6.6.17.3
tHardReset		5	ms	Section 6.3.13
tHardResetComplete	4	5	ms	Section 6.6.9
tNoResponse	4.5	5.5	s	Section 6.6.6
tPPSRequest		10	s	Section 6.6.19.1
tPPSTimeout	12	15	s	Section 6.6.19.2
tProtErrHardReset		15	ms	Section 6.6.11.4
tProtErrSoftReset		15	ms	Section 6.6.9.2
tPRSwapWait	100		ms	Section 6.6.4.2
tPSHardReset	25	35	ms	Section 6.6.11.2
tPSSourceOff	750	920	ms	Section 6.6.5.2
tPSSourceOn	390	480	ms	Section 6.6.5.3
tPSTransition	450	550	ms	Section 6.6.5.1
tReceive	0.9	1.1	ms	Section 6.6.1
tReceiverResponse		15	ms	Section 6.6.2
tRetry		195	µs	Section 6.6.1
tSenderResponse	24	30	ms	Section 6.6.2
tSinkRequest	100		ms	Section 6.6.4.1
tSinkTx	16	20	ms	Section 6.6.16
tSoftReset		15	ms	Section 6.8.1
tSwapSinkReady		15	ms	Section 6.6.8.1

Parameter	Value (min)	Value (max)	Units	Reference
tSwapSourceStart	20		ms	Section 6.6.8.1
tTransmit		195	µs	Section 6.6.1
tTypeCSendSourceCap	100	200	ms	Section 6.6.3.1
tTypeCSinkWaitCap	310	620	ms	Section 6.6.3.2
tVCONNSourceDischarge	160	240	ms	Section 6.6.10.1
tVCONNSourceOff		25	ms	Section 6.6.13
tVCONNSourceOn		50	ms	Section 6.3.11
tVCONNSourceTimeout	100	200	ms	Section 6.6.13
tVCONNSwapWait	100		ms	Section 6.6.4.4
tVDMBusy	50		ms	Section 6.6.12.4
tVDMEnterMode		25	ms	Section 6.6.12.2
tVDMExitMode		25	ms	Section 6.6.12.3
tVDMReceiverResponse		15	ms	Section 6.6.12.1
tVDMSenderResponse	24	30	ms	Section 6.6.12.1
tVDMWaitModeEntry	40	50	ms	Section 6.6.12.2
tVDMWaitModeExit	40	50	ms	Section 6.6.12.3

Table 6-62 Timers

Timer	Parameter	Used By	Reference
BISTContModeTimer	*tBISTContMode*	Policy Engine	Section 6.6.7.2
ChunkingNotSupportedTimer	*tChunkingNotSupported*	Policy Engine	Section 6.6.18.1
ChunkSenderRequestTimer	*tChunkSenderRequest*	Protocol	Section 6.6.18.2
ChunkSenderResponseTimer	*tChunkSenderResponse*	Protocol	Section 6.6.18.3
CRCReceiveTimer	*tReceive*	Protocol	Section 6.6.1
DataResetFailTimer	*tDataResetFail*	Policy Engine	Section 6.6.10.3
DiscoverIdentityTimer	*tDiscoverIdentity*	Policy Engine	Section 6.6.15
HardResetCompleteTimer	*tHardResetComplete*	Protocol	Section 6.6.9
NoResponseTimer	*tNoResponse*	Policy Engine	Section 6.6.6
PSHardResetTimer	*tPSHardReset*	Policy Engine	Section 6.6.11.2
PSSourceOffTimer	*tPSSourceOff*	Policy Engine	Section 6.6.5.2
PSSourceOnTimer	*tPSSourceOn*	Policy Engine	Section 6.6.5.3
PSTransitionTimer	*tPSTransition*	Policy Engine	Section 6.6.5.1
SenderResponseTimer	*tSenderResponse*	Policy Engine	Section 6.6.2
SinkPPSPeriodicTimer	*tPPSRequest*	Policy Engine	Section 6.6.19.1
SinkRequestTimer	*tSinkRequest*	Policy Engine	Section 6.6.4
SinkWaitCapTimer	*tTypeCSinkWaitCap*	Policy Engine	Section 6.6.3.2
SourceCapabilityTimer	*tTypeCSendSourceCap*	Policy Engine	Section 6.6.3.1
SourcePPSCommTimer	*tPPSTimeout*	Policy Engine	Section 6.6.19.2
SinkTxTimer	*tSinkTx*	Protocol Layer	Section 6.6.16
SwapSourceStartTimer	*tSwapSourceStart*	Policy Engine	Section 6.6.8.1
VCONNDischargeTimer	*tVCONNSourceDischarge*	Policy Engine	Section 6.6.10.1

Timer	Parameter	Used By	Reference
VCONNOnTimer	*tVCONNSourceTimeout*	Policy Engine	Section 6.6.13.1
VDMModeEntryTimer	*tVDMWaitModeEntry*	Policy Engine	Section 6.6.12.2
VDMModeExitTimer	*tVDMWaitModeExit*	Policy Engine	Section 6.6.12.3
VDMResponseTimer	*tVDMSenderResponse*	Policy Engine	Section 6.6.12.1

6.7 Counters

6.7.1 MessageID Counter

The ***MessageIDCounter*** is a rolling counter, ranging from 0 to ***nMessageIDCount***, used to detect duplicate Messages. This value is used for the ***MessageID*** field in the Message Header of each transmitted Message.

Each Port ***Shall*** maintain a copy of the last ***MessageID*** value received from its Port Partner. Devices that support multiple ports, such as Hubs, ***Shall*** maintain copies of the last ***MessageID*** on a per Port basis. A Port which communicates using SOP* Packets ***Shall*** maintain copies of the last ***MessageID*** for each type of ***SOP**** it uses.

The transmitter ***Shall*** use the ***MessageID*** in a ***GoodCRC*** Message to verify that a particular Message was received correctly. The receiver ***Shall*** use the ***MessageID*** to detect duplicate Messages.

6.7.1.1 Transmitter Usage

The Transmitter ***Shall*** use the ***MessageID*** as follows:

- Upon receiving either ***Hard Reset*** Signaling, or a ***Soft_Reset*** Message, the transmitter ***Shall*** set its ***MessageIDCounter*** to zero and re-initialize its retry mechanism.
- If a ***GoodCRC*** Message with a ***MessageID*** matching the ***MessageIDCounter*** is not received before the ***CRCReceiveTimer*** expires, it ***Shall*** retry the same packet up to ***nRetryCount*** times using the same ***MessageID***.
- If a ***GoodCRC*** Message is received with a ***MessageID*** matching the current ***MessageIDCounter*** before the ***CRCReceiveTimer*** expires, the transmitter ***Shall*** re-initialize its retry mechanism and increment its ***MessageIDCounter***.
- If the Message is aborted by the Policy Engine, the transmitter ***Shall*** delete the Message from its transmit buffer, re-initialize its retry mechanism and increment its ***MessageIDCounter***.

6.7.1.2 Receiver Usage

The Receiver ***Shall*** use the ***MessageID*** as follows:

- When the first good packet is received after a reset, the receiver ***Shall*** store a copy of the received ***MessageID*** value.
- For subsequent Messages, if ***MessageID*** value in a received Message is the same as the stored value, the receiver ***Shall*** return a ***GoodCRC*** Message with that ***MessageID*** value and drop the Message (this is a retry of an already received Message). Note: this ***Shall Not*** apply to the ***Soft_Reset*** Message which always has a ***MessageID*** value of zero.
- If ***MessageID*** value in the received Message is different than the stored value, the receiver ***Shall*** return a ***GoodCRC*** Message with the new ***MessageID*** value, store a copy of the new ***MessageID*** value and process the Message.

6.7.2 Retry Counter

The ***RetryCounter*** is used by a Port whenever there is a Message transmission failure (timeout of ***CRCReceiveTimer***). If the ***nRetryCount*** retry fails, then the link ***Shall*** be reset using the Soft Reset mechanism.

The following rules apply to retries when there is a Message transmission failure (see also Section 6.11.2.1):

- Cable Plugs ***Shall Not*** retry Messages.
- Extended Messages of ***Data Size*** > ***MaxExtendedMsgLegacyLen*** that are not Chunked (***Chunked*** flag set to zero) ***Shall Not*** be retried.
- Extended Messages of ***Data Size*** ≤ ***MaxExtendedMsgLegacyLen*** (***Chunked*** flag set to zero or one) ***Shall*** be retried.

- Extended Messages of ***Data Size*** > ***MaxExtendedMsgLegacyLen*** that are Chunked (***Chunked*** flag set to one) individual Chunks ***Shall*** be retried.

When messages are not retried, then the ***RetryCounter*** is not used. Higher layer protocols are expected to accommodate message delivery failure or failure to receive a ***GoodCRC*** Message.

6.7.3 Hard Reset Counter

The ***HardResetCounter*** is used to retry the Hard Reset whenever there is no response from the remote device (see Section 6.6.6). Once the Hard Reset has been retried ***nHardResetCount*** times then it ***Shall*** be assumed that the remote device is non-responsive.

6.7.4 Capabilities Counter

The ***CapsCounter*** is used to count the number of ***Source_Capabilities*** Messages which have been sent by a Source at power up or after a Hard Reset. Implementation of the ***CapsCounter*** is ***Optional*** but ***May*** be used by any Source which wishes to preserve power by not sending ***Source_Capabilities*** Messages after a period of time.

When the ***CapsCounter*** is implemented and the Source detects that a Sink is Attached then after ***nCapsCount Source_Capabilities*** Messages have been sent the Source ***Shall*** decide that the Sink is non-responsive, stop sending ***Source_Capabilities*** Messages and disable PD.

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

6.7.5 Discover Identity Counter

When sending ***Discover Identity*** Messages to a Cable Plug a Port ***Shall*** maintain a count of Messages sent (***DiscoverIdentityCounter***). No more than ***nDiscoverIdentityCount Discover Identity*** Messages ***Shall*** be sent by the Port without receiving a ***GoodCRC*** Message response. A VCONN Swap ***Shall*** reset the ***DiscoverIdentityCounter*** to zero.

6.7.6 VDMBusyCounter

When sending Responder Busy responses to a Structured ***Vendor_Defined*** Message a UFP or Cable Plug ***Shall*** maintain a count of Messages sent (***VDMBusyCounter***). No more than ***nBusyCount*** Responder Busy responses ***Shall*** be sent. The ***VDMBusyCounter*** ***Shall*** be reset on sending a non-Busy response. Products wishing to meet ***[USB Type-C 2.0]*** requirements for Mode entry ***Should*** use an ***nBusyCount*** of 1.

6.7.7 Counter Values and Counters

Table 6-64 lists the counters used in this section and Table 6-63 shows the corresponding parameters.

Table 6-63 Counter parameters

Parameter	Value	Reference
nBusyCount	5	Section 6.7.6
nCapsCount	50	Section 6.7.4
nDiscoverIdentityCount	20	Section 6.7.5
nHardResetCount	2	Section 6.7.3
nMessageIDCount	7	Section 6.7.1
nRetryCount	2	Section 6.7.2

Table 6-64 Counters

Counter	Max	Reference
CapsCounter	***nCapsCount***	Section 6.7.4
DiscoverIdentityCounter	***nDiscoverIdentityCount***	Section 6.7.5
HardResetCounter	***nHardResetCount***	Section 6.7.3
MessageIDCounter	***nMessageIDCount***	Section 6.7.1
RetryCounter	***nRetryCount***	Section 6.7.2
VDMBusyCounter	***nBusyCount***	Section 6.7.6

6.8 Reset

Resets are a necessary response to protocol or other error conditions. USB Power Delivery defines four different types of reset:

- Soft Reset, which resets protocol
- Data Reset which resets the USB communications
- Hard Reset which resets both the power supplies and protocol
- Cable Reset which resets the cable.

6.8.1 Soft Reset and Protocol Error

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

Protocol Errors are any unexpected Message during an AMS. If the first Message in an AMS has been passed to the Protocol Layer by the Policy Engine but has not yet been sent (***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 (see Section 8.3.3.4) except when the voltage is transition when a Protocol Error ***Shall*** lead to a Hard Reset (see Section 6.6.11.4 and Section 8.3.3.2). Details of Interruptible and Non-interruptible AMS's can be found in Section 8.3.2.1.3.

An Unrecognized or Unsupported Message received in the ***PE_SNK_Ready*** or ***PE_SRC_Ready*** states, ***Shall Not*** cause a ***Soft_Reset*** Message to be generated but instead a ***Not_Supported*** Message ***Shall*** be generated.

A ***Soft_Reset*** Message ***Shall*** be sent regardless of the Rp value either ***SinkTxOk*** or ***SinkTxNG*** if it is the correct response in that state. Note: this means that a ***Soft_Reset*** Message can be sent during an AMS regardless of the Rp value either ***SinkTxOk*** or ***SinkTxNG*** when responding to a Protocol Error.

Table 6-65 and Table 6-66 summarize the responses that ***Shall*** be made to an incoming Message including VDMs.

Table 6-65 Response to an incoming Message (except VDM)

Recipient's Power Role	Recipient's state	Incoming Message			
		Recognized			Unrecognized
		Supported		Unsupported	
		Expected	Unexpected		
Source	*PE_SRC_Ready*	Process Message	*Soft_Reset* Message[2]	*Not_Supported* Message[3]	*Not_Supported* Message[3] (except for VDM) See 6.4.4.1 for UVDM, 6.12.4 for SVDM
	During Interruptible AMS (In Explicit Contract)	Process Message	return to *PE_SRC_Ready* state and process Message		
	During Interruptible AMS (Not in Explicit Contract)	Process Message	*Soft_Reset* Message[2]		
	During Non-interruptible AMS (power not transitioning[1])	Process Message	*Soft_Reset* Message[2]		
	During Non-interruptible AMS (power transitioning[1])	Process Message	*Hard Reset* Signaling		
Sink	*PE_SNK_Ready*	Process Message	*Soft_Reset* Message[2]	*Not_Supported* Message[3]	*Not_Supported* Message[3] (except for VDM) See 6.4.4.1 for UVDM, 6.12.4 for SVDM
	During Interruptible AMS (In Explicit Contract)	Process Message	return to *PE_SNK_Ready* state and process Message		
	During Interruptible AMS (Not in Explicit Contract)	Process Message	*Soft_Reset* Message[2]		
	During Non-interruptible AMS (not power transitioned)	Process Message	*Soft_Reset* Message[2]		
	During Non-interruptible AMS (power transitioned)	Process Message	*Hard Reset* Signaling		

1. "power transitioning" means the policy engine is in *PE_SRC_Transition_Supply* State or *PE_SNK_Transition_Sink* State or *PE_FRS_SNK_SRC_Send_Swap* State.
2. The *Soft_Reset* Message ***Shall*** be sent using the SOP* of the incoming message.
3. The *Not_Supported* Message ***Shall*** be sent using the SOP* of the incoming message.

Table 6-66 Response to an incoming VDM

Recipient's Role	Supported UVDM	Unsupported UVDM	Unrecognized UVDM	Supported SVDM	Unsupported SVDM	Unrecognized SVDM
DFP or UFP	Defined by vendor	*Not_Supported* Message	*Not_Supported* Message	See 6.12.4	*Not_Supported* Message	NAK Command
Cable Plug	Defined by vendor	Message ***Ignored***	Message ***Ignored***	See 6.12.4	Message ***Ignored***	NAK Command

A failure to see a *GoodCRC* Message in response to any Message within *tReceive* (after *nRetryCount* retries), when a Port Pair is Connected, is indicative of a communications failure resulting in a Soft Reset (see Section 6.6.9.1).

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 8-8). 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 6.8.3 and Section 6.8.4).

6.8.2 Data Reset

A *Data_Reset* Message is used by a Port to reset its USB data connection and to exit all Alternate Modes both with its Port Partner and in the Cable Plug(s).

The Data Reset process ***May*** be initiated by either Port Partner sending a *Data_Reset* Message.

A Data Reset impacts USB Power Delivery in the following ways:

- ***Shall Not*** change the Port Power Roles (Source/Sink) or Port Data Roles (DFP/UFP).
- ***Shall Not*** change the existing Explicit Contract.
- ***Shall*** cause all Active Modes to be exited.
- ***Shall*** reset the cable by Power cycling VCONN.
- The DFP ***Shall*** become the VCONN Source.

If the Data Reset process fails then the Port ***Shall*** enter the *ErrorRecovery* State as defined in *[USB Type-C 2.0]*.

See Section 6.3.14 for details of Data Reset operation.

6.8.3 Hard Reset

Hard Resets are signaled by an ordered set as defined in Section 5.6.4. Both the sender and recipient ***Shall*** cause their power supplies to return to their default states (see Section 7.3.12 and Section 7.3.13 for details of voltage transitions). In addition, their respective Protocol Layers ***Shall*** be reset as for the Soft Reset. This allows the Attached devices to be in a state where they can re-establish USB PD communication. Hard Reset is retried up to *nHardResetCount* times (see also Section 6.6.6 and Section 6.7.3). 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.

A Hard Reset ***Shall Not*** cause any change to either the Rp/Rd 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 Rp resistor asserted and UFP for a Port with the Rd resistor asserted.

When VCONN is supported (see ***[USB Type-C 2.0]***) the Hard Reset ***Shall*** cause the Port with the Rp resistor asserted to supply VCONN and the Port with the Rd resistor asserted to turn off VCONN.

In effect the Hard Reset will revert the Ports to their default state based on their CC line resistors. Removing and reapplying VCONN from the Cable Plugs also ensures that they re-establish their configuration as either SOP' or SOP'' based on the location of VCONN (see ***[USB Type-C 2.0]***).

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 2.0]***.

A Sink ***Shall*** be able to send ***Hard Reset*** signaling regardless of the value of Rp (see Section 5.7).

6.8.3.1 Cable Plugs and Hard Reset

Cable Plugs ***Shall Not*** generate ***Hard Reset*** Signaling but ***Shall*** monitor for ***Hard Reset*** Signaling between the Port Partners and ***Shall*** reset when this is detected (see Section 8.3.3.24.2.2). 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.

6.8.3.2 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).

6.8.4 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 VCONN prior to a Cable Reset. If VCONN has been turned off the DFP ***Shall*** turn on VCONN prior to generating ***Cable Reset*** Signaling. If there has been a VCONN Swap and the UFP is currently supplying VCONN, the DFP ***Shall*** perform a VCONN Swap such that it is supplying VCONN prior to generating ***Cable Reset*** Signaling.

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

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

6.9 Collision Avoidance

In order to avoid message collisions due to asynchronous Messaging sent from the Sink, the Source sets Rp to ***SinkTxOk*** to indicate to the Sink that it is ok to initiate an AMS. When the Source wishes to initiate an AMS it sets Rp to ***SinkTxNG***. When the Sink detects that Rp is set to ***SinkTxOk*** it ***May*** initiate an AMS. When the Sink detects that Rp is set to ***SinkTxNG*** it ***Shall Not*** initiate an AMS and ***Shall*** only send Messages that are part of an AMS the Source has initiated. Note that this restriction applies to SOP* AMS's i.e. for both Port to Port and Port to Cable Plug communications.

Note: A Sink can still send ***Hard Reset*** signaling at any time.

6.10 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 6-67 for details of the Messages that ***Shall*** / ***Shall Not*** be ***Discarded***.

Table 6-67 Message discarding

Message pending transmission	Message received	Discard pending transmission?
SOP	SOP	Yes
SOP	SOP'/SOP''	No
SOP'	SOP	Yes
SOP'	SOP'	No
SOP'	SOP''	No
SOP''	SOP	Yes
SOP''	SOP'	No
SOP''	SOP''	No

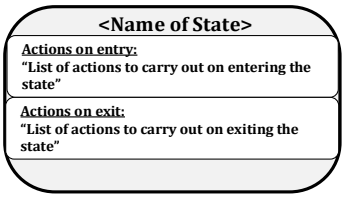
6.11 State behavior

6.11.1 Introduction to state diagrams used in Chapter 6

The state diagrams defined in Section 6.11 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 6-48 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 6-48 Outline of States

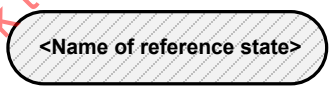


Transitions from one state to another are indicated by arrows with the conditions listed on the arrow. Where there are multiple conditions these are connected using either a logical OR "|" or a logical AND "&". The inverse of a condition is shown with a "NOT" in front of the condition.

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 6-49 indicates how such references are made. The reference is indicated with a hatched box. The box contains the name of the referenced state.

Figure 6-49 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.)

6.11.2 State Operation

The following section details Protocol Layer State Operation when sending and receiving SOP* Packets.

For each SOP* Communication being sent and received there ***Shall*** be separate Protocol Layer Transmission and Protocol Layer Reception and Hard Reset State Machine instances, with their own counter and timer instances. When Chunking is supported there ***Shall*** be separate Chunked Tx, Chunked Tx and Chunked Message Router State Machine instances.

Soft Reset ***Shall*** only apply to the State Machine instances it is targeted at based on the type of SOP* Packet used to send the ***Soft_Reset*** Message. The Hard-Reset State Machine (including Cable Reset) ***Shall***

apply simultaneously to all Protocol Layer State Machine instances active in the DFP, UFP and Cable Plug (if present).

6.11.2.1 Protocol Layer Chunking

6.11.2.1.1 Architecture of Device Including Chunking Layer

The Chunking component resides in the Protocol Layer between the Policy Engine and Protocol Tx/Rx. Figure 6-50 illustrates the relationship between components.

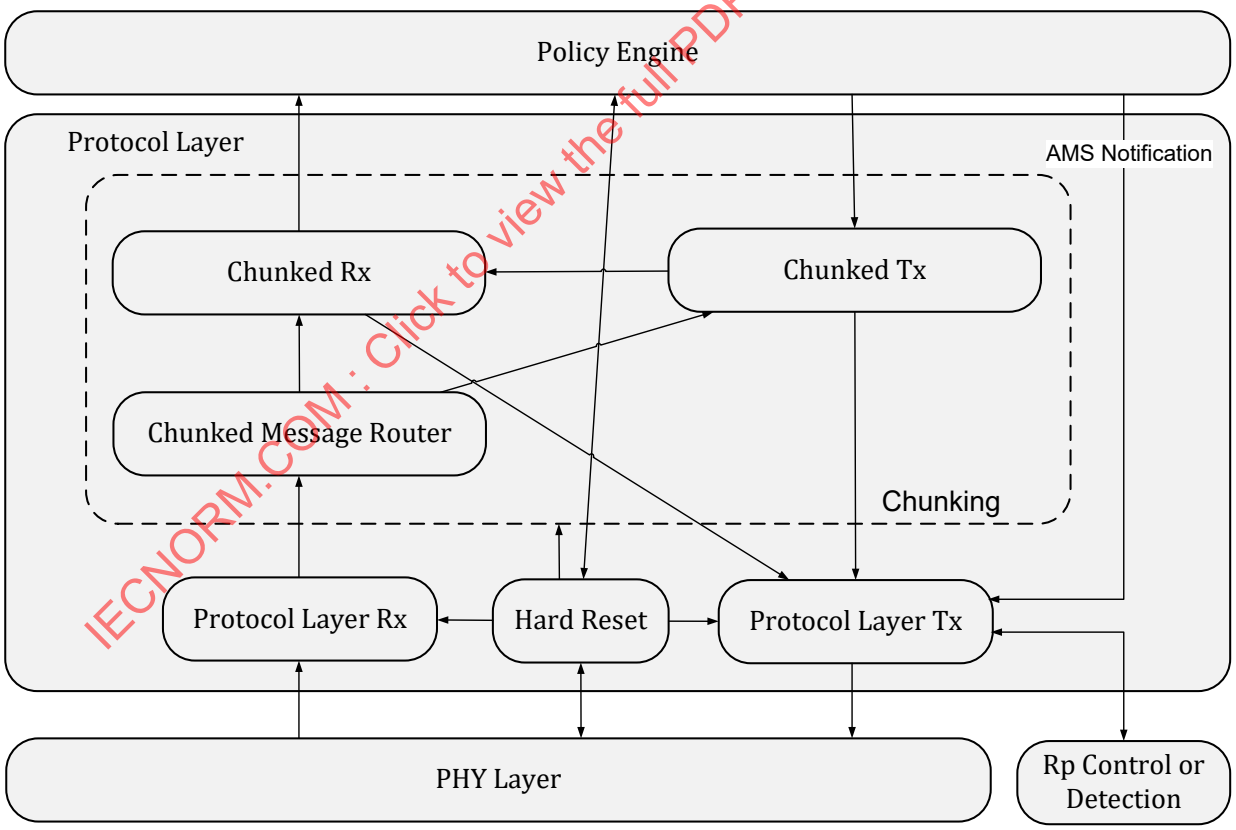
The Chunking Layer comprises three related state machines:

- Chunked Rx.
- Chunked Tx.
- Chunked Message Router.

Note that the consequence of this architecture is that the Policy Engine deals entirely in un-chunked messages. It will not receive (and might not respond to) a message until all the related chunks have been collated.

If a PD Device or Cable Marker has no requirement to handle any message requiring more than one Chunk of any Extended Message, it ***May*** omit the Chunking Layer. In this case it ***Shall*** implement the ***ChunkingNotSupportedTimer*** to ensure compatible operation with partners which support Chunking (see Section 6.6.18.1 and Section 8.3.3.6).

Figure 6-50 Chunking architecture Showing Message and Control Flow



6.11.2.1.1.1 Optional Abort Mechanism

Long Chunked Messages bring with them the potential problem that they could prevent urgent messages from being transmitted in a timely manner. An optional Abort mechanism is provided to remedy this problem.

The Abort Flag referred to in the diagrams below ***May*** be set and examined by the Policy Engine. The specific means are left to the implementer.

6.11.2.1.1.2 Aborting Sending a Long-Chunked Message

A long-Chunked Message being sent ***May*** be aborted by setting the ***Optional*** Abort Flag. The message ***Shall*** be considered aborted when the Abort Flag is again cleared by the Chunked Tx state machine.

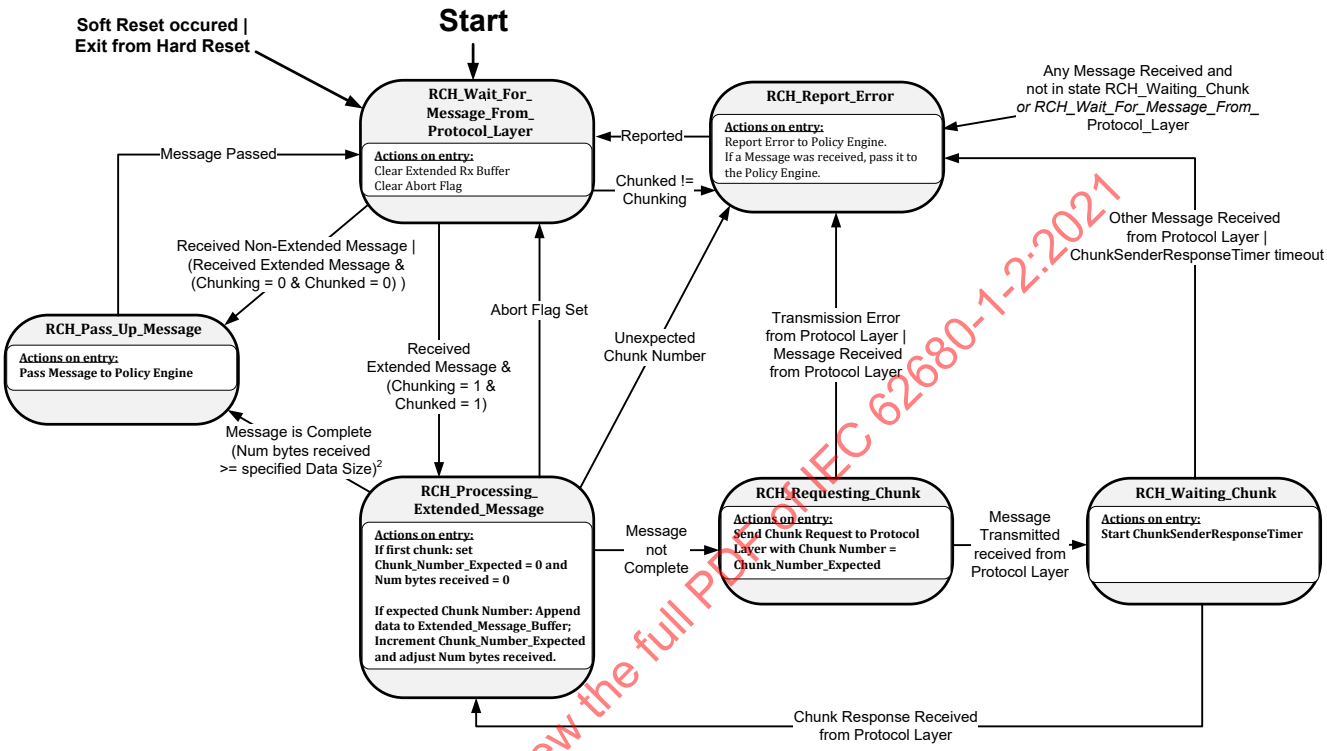
6.11.2.1.1.3 Aborting Receiving a Long-Chunked Message

If the optional Abort mechanism has been implemented, any message sent while a Chunked Message receive is in progress will result in an error report being received by the Policy Engine, to indicate that the message request has been ***Discarded***. If the message was urgent the Policy Engine might set the Abort Flag, which will result in the incoming Chunked Message being aborted. The Abort Flag being cleared by the Chunked Rx state machine indicates that the urgent message can now be sent.

6.11.2.1.2 Chunked Rx State Diagram

Figure 6-51 shows the state behavior for the Chunked Rx State Machine. This recognizes whether chunked received messages are involved and deals with requesting chunks when they are. It also performs validity checks on all messages related to chunking.

Figure 6-51 Chunked Rx State Diagram



[1] Chunking is an internal state that is set to 1 if the 'Unchunked Extended Messages Supported' bit in either Source Capabilities or Request is 0. It defaults to 1 and is set after the first exchange of Source Capabilities and Request. It is also set to 1 for ***SOP'*** or ***SOP''*** communication.

[2] Additional bytes received over specified ***Data Size*** will be as a result of padding in the last chunk.

6.11.2.1.2.1 RCH_Wait_For_Message_From_Protocol_Layer State

The Chunked Rx State Machine ***Shall*** enter the ***RCH_Wait_For_Message_From_Protocol_Layer*** state:

- At startup.
- As a result of a Soft Reset occurring.
- On exit from a Hard Reset.

On entry to the ***RCH_Wait_For_Message_From_Protocol_Layer*** state the Chunked Rx state machine clears the Extended Rx Buffer, and clears the optional Abort Flag.

In the ***RCH_Wait_For_Message_From_Protocol_Layer*** state the Chunked Rx state machine waits until the Chunked Message Router passes up a received message.

The Chunked Rx State Machine ***Shall*** transition to the ***RCH_Pass_Up_Message*** state when:

- A non-Extended Message is passed up from the Chunked Message Router.
- An Extended Message is passed up from the Chunked Message Router, and the Policy Engine has determined that we are not doing Chunking, and the Message has its ***Chunked*** bit set to 0b.

The Chunked Rx State Machine ***Shall*** transition to the ***RCH_Processing_Extended_Message*** state when:

- An Extended Message is passed up from the Chunked Message Router, and the Policy Engine has determined that we are doing Chunking, and the Message has its ***Chunked*** bit set to 1b.

6.11.2.1.2.2 RCH_Pass_Up_Message State

On entry to the ***RCH_Pass_Up_Message*** state the Chunked Rx state machine ***Shall*** pass the received message to the Policy Engine.

The Chunked Rx State Machine ***Shall*** transition to the ***RCH_Wait_For_Message_From_Protocol_Layer*** state when:

- The Message has been passed.

6.11.2.1.2.3 RCH_Processing_Extended_Message State

On entry to the ***RCH_Processing_Extended_Message*** state the Chunked Rx state machine ***Shall***:

- If this is the first chunk:
  - Set Chunk_Number_Expected = 0.
  - Set Num bytes received = 0.
- If chunk contains the expected Chunk Number:
  - Append its data to the Extended_Message_Buffer.
  - Increment Chunk_Number_Expected.
  - Adjust Num bytes received.

The Chunked Rx State Machine ***Shall*** transition to the ***RCH_Pass_Up_Message*** state when:

- The message is complete (i.e. Num bytes received >= specified ***Data Size***. Note that the inequality allows for padding bytes in the last chunk, which are not actually part of the extended message).

The Chunked Rx State Machine ***Shall*** transition to the ***RCH_Requesting_Chunk*** state when:

- The Message is not yet complete.

The Chunked Rx State Machine ***Shall*** transition to the ***RCH_Report_Error*** state when:

- An unexpected Chunk Number is received.

The Chunked Rx State Machine ***Shall*** transition to the ***RCH_Wait_For_Message_From_Protocol_Layer*** state when:

- The optional Abort Flag is set.

6.11.2.1.2.4 RCH_Requesting_Chunk State

On entry to the ***RCH_Requesting_Chunk*** state the Chunked Rx state machine ***Shall***:

- Send Chunk Request to Protocol Layer with ***Chunk Number*** = Chunk_Number_Expected.

The Chunked Rx State Machine ***Shall*** transition to the ***RCH_Waiting_Chunk*** state when:

- Message Transmitted is received from the Protocol Layer.

The Chunked Rx State Machine ***Shall*** transition to the ***RCH_Report_Error*** state when:

- Transmission Error is received from the Protocol Layer, or
- A Message is received from the Protocol Layer.

6.11.2.1.2.5 RCH_Waiting_Chunk State

On entry to the ***RCH_Waiting_Chunk*** state the Chunked Rx state machine ***Shall***:

- Start the ***ChunkSenderResponseTimer.***

The Chunked Rx State Machine ***Shall*** transition to the ***RCH_Processing_Extended_Message*** state when:

- A Chunk is received from the Protocol Layer.

The Chunked Rx State Machine ***Shall*** transition to the ***RCH_Report_Error*** state when:

- A Message, other than a Chunk, is received from the Protocol Layer, or
- The ***ChunkSenderResponseTimer*** expires.

6.11.2.1.2.6 RCH_Report_Error State

The Chunked Rx State Machine ***Shall*** enter the ***RCH_Report_Error*** state:

- When any Message is received and the Chunked Rx State Machine is not in one of the states ***RCH_Waiting_Chunk*** or ***RCH_Wait_For_Message_From_Protocol_Layer.***

On entry to the ***RCH_Report_Error*** state the Chunked Rx state machine ***Shall***:

- Report the error to the Policy Engine.
- If the state was entered because a Message was received, this Message ***Shall*** be passed to the Policy Engine.

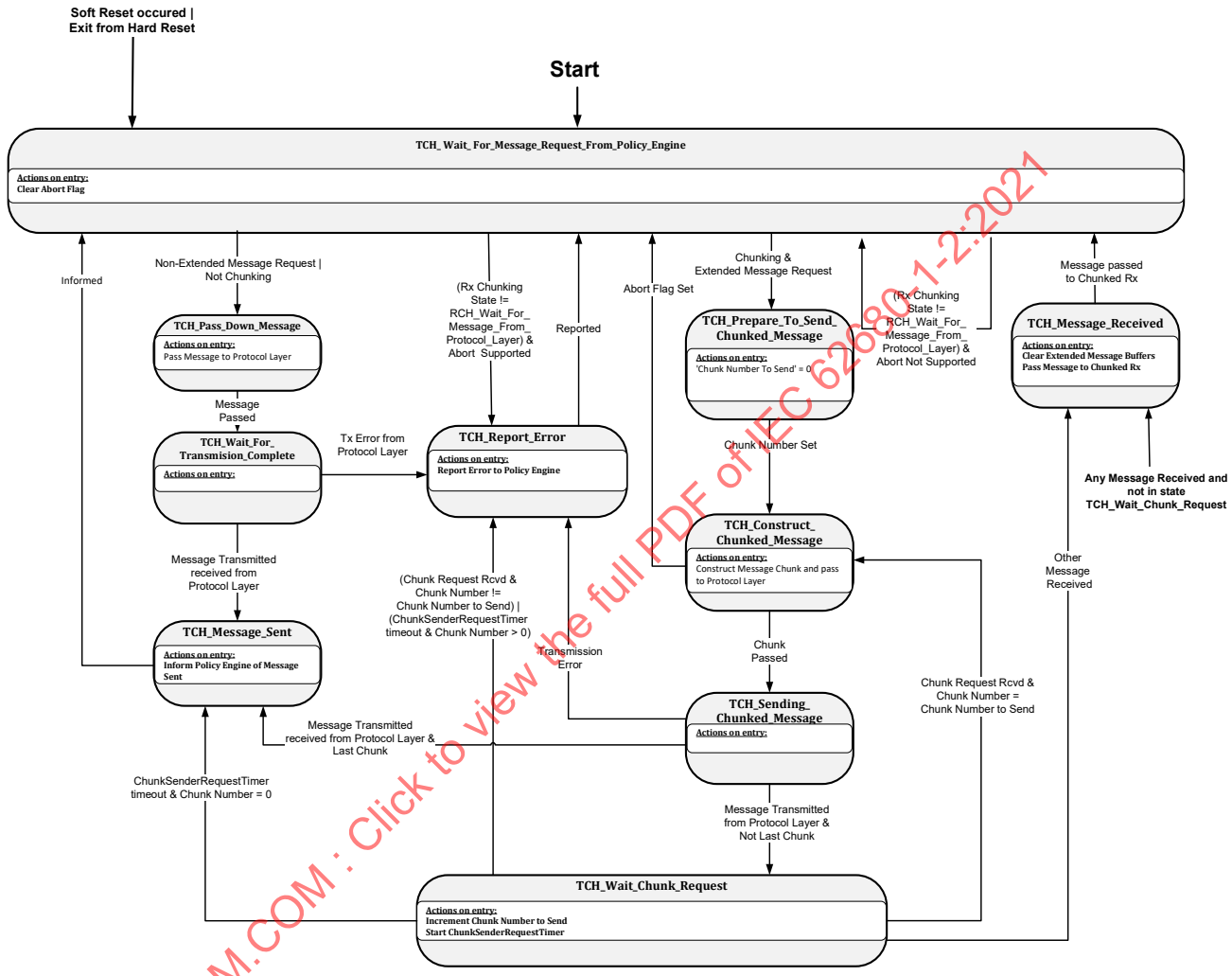
The Chunked Rx State Machine ***Shall*** transition to the ***RCH_Wait_For_Message_From_Protocol_Layer*** state when:

- The error has been reported.
- Any message received was passed to the Policy Engine.

6.11.2.1.3 Chunked Tx State Diagram

Figure 6-52 shows the state behavior for the Chunked Tx State Machine. This recognizes whether chunked transmitted messages are involved and deals with sending chunks and waiting for chunk requests when they are. It also performs validity checks on all related messages related to chunking.

Figure 6-52 Chunked Tx State Diagram



6.11.2.1.3.1 TCH_Wait_For_Message_Request_From_Policy_Engine State

The Chunked Tx State Machine ***Shall*** enter the ***TCH_Wait_For_Message_Request_From_Policy_Engine*** state:

- At startup.
- As a result of a Soft Reset occurring.
- On exit from a Hard Reset.

On entry to the ***TCH_Wait_For_Message_Request_From_Policy_Engine*** state the Chunked Tx state machine clears the optional Abort Flag.

In the ***TCH_Wait_For_Message_Request_From_Policy_Engine*** state the Chunked Tx State Machine waits until the Policy Engine sends it a Message Request.

The Chunked Tx State Machine ***Shall*** transition to the ***TCH_Pass_Down_Message*** state when:

- A non-Extended Message Request is received from the Policy Engine, or

- A Message Request is received from the Policy Engine and the link is not Chunking.

The Chunked Tx State Machine ***Shall*** transition to the ***TCH_Prepare_To_Send_Chunked_Message*** state when:

- An Extended Message Request is received from the Policy Engine, and the link is Chunking.

The Chunked Tx State Machine ***Shall Discard*** the Message Request and remain in the ***TCH_Wait_For_Message_Request_From_Policy_Engine*** state when:

- The Chunked Rx state is any other than ***TCH_Wait_For_Message_Request_From_Policy_Engine***, and the optional Abort Flag has not been implemented.

The Chunked Tx State Machine ***Shall Discard*** the Message Request and enter the ***TCH_Report_Error*** state when:

- The Chunked Rx state is any other than ***TCH_Wait_For_Message_Request_From_Policy_Engine***, and the optional Abort Flag has been implemented.

6.11.2.1.3.2 TCH_Pass_Down_Message State

On entry to the ***TCH_Pass_Down_Message*** state the Chunked Tx State Machine ***Shall*** pass the message to the Protocol Layer.

The Chunked Tx State Machine ***Shall*** transition to the ***TCH_Wait_For_Transmision_Complete*** state when:

- The message has been passed to the Protocol Layer.

6.11.2.1.3.3 TCH_Wait_For_Transmision_Complete State

The Chunked Tx State Machine ***Shall*** transition to the ***TCH_Message_Sent*** state when:

- Message Transmitted has been received from the Protocol Layer.

The Chunked Tx State Machine ***Shall*** transition to the ***TCH_Report_Error*** state when:

- Transmission Error has been received from the Protocol Layer.

6.11.2.1.3.4 TCH_Message_Sent State

On entry to the ***TCH_Message_Sent*** state the Chunked Tx State Machine ***Shall***:

- Inform the Policy Engine that the Message has been sent.

The Chunked Tx State Machine ***Shall*** transition to the ***TCH_Wait_For_Message_Request_From_Policy_Engine*** state when:

- The Policy Engine has been informed.

6.11.2.1.3.5 TCH_Prepare_To_Send_Chunked_Message State

On entry to the ***TCH_Prepare_To_Send_Chunked_Message*** state the Chunked Tx State Machine ***Shall***:

- Set 'Chunk Number To Send' to zero.

The Chunked Tx State Machine ***Shall*** transition to the ***TCH_Construct_Chunked_Message*** state when:

- 'Chunk Number To Send' has been set to zero.

6.11.2.1.3.6 TCH_Construct_Chunked_Message State

On entry to the ***TCH_Construct_Chunked_Message*** state the Chunked Tx State Machine ***Shall***:

- Construct a Message Chunk and pass it to the Protocol Layer.

The Chunked Tx State Machine ***Shall*** transition to the ***TCH_Sending_Chunked_Message*** state when:

- The Message Chunk has been passed to the Protocol Layer.

The Chunked Tx State Machine ***Shall*** transition to the ***TCH_Wait_For_Message_Request_From_Policy_Engine*** state when:

- The optional Abort Flag is set.

6.11.2.1.3.7 TCH_Sending_Chunked_Message State

The Chunked Tx State Machine ***Shall*** transition to the ***TCH_Wait_Chunk_Request*** state when:

- Message Transmitted is received from Protocol Layer and this was not the last chunk.

The Chunked Tx State Machine ***Shall*** transition to the ***TCH_Message_Sent*** state when:

- Message Transmitted is received from Protocol Layer and this was the last chunk.

The Chunked Tx State Machine ***Shall*** transition to the ***TCH_Report_Error*** state when:

- Transmission Error has been received from the Protocol Layer.

6.11.2.1.3.8 TCH_Wait_Chunk_Request State

On entry to the ***TCH_Wait_Chunk_Request*** state the Chunked Tx State Machine ***Shall***:

- Increment Chunk Number to Send.
- Start ***ChunkSenderRequestTimer.***

The Chunked Tx State Machine ***Shall*** transition to the ***TCH_Report_Error*** state when:

- A Chunk Request has been received and the Chunk Number does not equal Chunk Number to Send) or
- ***ChunkSenderRequestTimer*** has expired and Chunk Number is greater than zero.

The Chunked Tx State Machine ***Shall*** transition to the ***TCH_Message_Sent*** state when:

- ***ChunkSenderRequestTimer*** has expired and Chunk Number equals zero.

Note that this is the mechanism which allows the remote port partner or cable marker to omit the chunking layer. The Policy Engine will receive a Message Sent signal if the remote port partner or cable marker is present (***GoodCRC*** Message received) but does not sent a Chunk Request. After this the remote port partner will send a ***Not_Supported*** Message, or the Cable Marker will ***Ignore*** the Chunked Message.

The Chunked Tx State Machine ***Shall*** transition to the ***TCH_Message_Received*** state when:

- Any other message than Chunk Request is received.

6.11.2.1.3.9 TCH_Message_Received State

The Chunked Tx State Machine ***Shall*** enter the ***TCH_Message_Received*** state:

- When any Message is received and the Chunked Tx State Machine is not in the ***TCH_Wait_Chunk_Request*** state.

On entry to the ***TCH_Message_Received*** state the Chunked Tx State Machine ***Shall***:

- Clear the Extended Message Buffers.
- Pass the received Message to Chunked Rx Engine.

The Chunked Tx State Machine ***Shall*** transition to the ***TCH_Wait_For_Message_Request_From_Policy_Engine*** state when:

- The received message has been passed to the Chunked Rx Engine.

6.11.2.1.3.10 TCH_Report_Error State

On entry to the ***TCH_Report_Error*** state the Chunked Tx State Machine ***Shall***:

- Report the error to the Policy Engine.

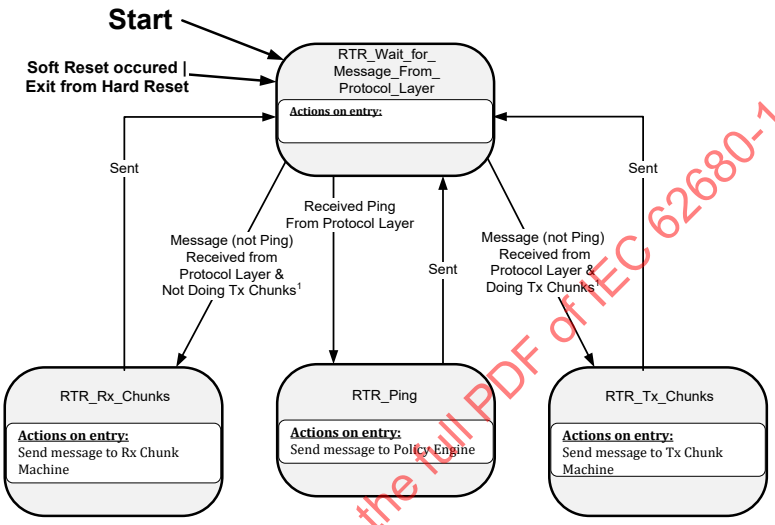
The Chunked Tx State Machine ***Shall*** transition to the ***TCH_Wait_For_Message_Request_From_Policy_Engine*** state when:

- The error has been reported.

6.11.2.1.4 Chunked Message Router State Diagram

Figure 6-53 shows the state behavior for the Chunked Message Router. This determines to which state machine an incoming message is routed to (Chunked Rx, Chunked Tx or direct to Policy Engine).

Figure 6-53 Chunked Message Router State Diagram



[1] Doing Tx Chunks means that Chunked Tx State Machine is not in the ***TCH_Wait_For_Message_Request_From_Policy_Engine*** state

[2] Messages are taken to include notification about transmission success or otherwise of Messages

6.11.2.1.4.1 RTR_Wait_for_Message_From_Protocol_Layer State

In the ***RTR_Wait_for_Message_From_Protocol_Layer*** state the Chunked Message Router waits until the Protocol Layer sends it a received Message.

The Chunked Message Router ***Shall*** transition to the ***RTR_Rx_Chunks*** state when:

- A Message other than a ***Ping*** Message is received from the Protocol Layer, and the combined Chunking is not doing Tx Chunks.

The Chunked Message Router ***Shall*** transition to the ***RTR_Tx_Chunks*** state when:

- A Message other than a ***Ping*** Message is received from the Protocol Layer, and the combined Chunking is doing Tx Chunks.

The Chunked Message Router ***Shall*** transition to the ***RTR_Ping*** state when:

- A ***Ping*** Message is received from the Protocol Layer.

6.11.2.1.4.2 RTR_Rx_Chunks State

On entry to the ***RTR_Rx_Chunks*** state the Chunked Message Router ***Shall***:

- Send the message to the Chunked Rx State Machine.
- Transition to the ***RTR_Wait_for_Message_From_Protocol_Layer*** state.

6.11.2.1.4.3 RTR_Ping State

On entry to the ***RTR_Ping*** state the Chunked Message Router ***Shall***:

- Send the message to the Policy Engine.
- Transition to the ***RTR_Wait_for_Message_From_Protocol_Layer*** state.

6.11.2.1.4.4 RTR_Tx_Chunks State

On entry to the ***RTR_Tx_Chunks*** state the Chunked Message Router ***Shall***:

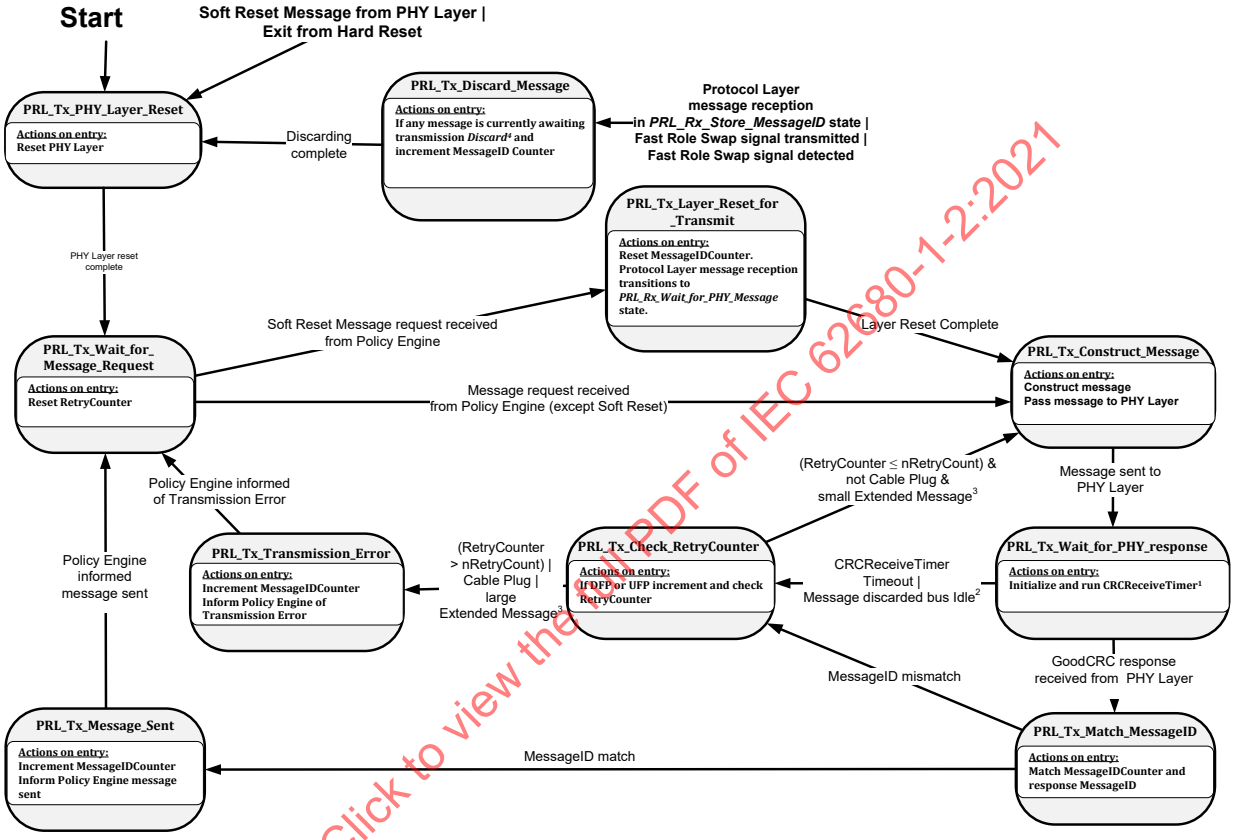
- Send the message to the Chunked Tx State Machine.
- Transition to the ***RTR_Wait_for_Message_From_Protocol_Layer*** state.

6.11.2.2 Protocol Layer Message Transmission

6.11.2.2.1 Common Protocol Layer Message Transmission State Diagram

Figure 6-54 shows the state behavior, common between the Source and the Sink, for the Protocol Layer when transmitting a Message.

Figure 6-54 Common Protocol Layer Message Transmission State Diagram



[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 6.6.1).

[2] This indication is sent by the PHY Layer when a message has been ***Discarded*** due to CC being busy, and after CC becomes idle again (see Section 5.7). The ***CRCReceiveTimer*** is not running in this case since no message has been sent.

[3] A "small" Extended Message is either an Extended Message with ***Data Size*** ≤ ***MaxExtendedMsgLegacyLen*** bytes or an Extended Message with ***Data Size*** > ***MaxExtendedMsgLegacyLen*** bytes that has been Chunked. A "large" Extended Message is an Extended Message with ***Data Size*** > ***MaxExtendedMsgLegacyLen*** bytes that has not been Chunked.

[4] See Section 6.10 for details of when Messages are ***Discarded***.

6.11.2.2.1.1 PRL_Tx_PHY_Layer_Reset State

The Protocol Layer ***Shall*** enter the ***PRL_Tx_PHY_Layer_Reset*** state:

- At startup.
- As a result of a Soft Reset request being received by the PHY Layer.
- On exit from a Hard Reset.

On entry to the ***PRL_Tx_PHY_Layer_Reset*** state the Protocol Layer ***Shall*** reset the PHY Layer (clear any outstanding Messages and enable communications).

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

- When the PHY Layer reset is complete.

6.11.2.2.1.2 PRL_Tx_Wait_for_Message_Request State

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

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

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

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

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

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

6.11.2.2.1.3 PRL_Tx_Layer_Reset_for_Transmit State

On entry to the ***PRL_Tx_Layer_Reset_for_Transmit*** state the Protocol Layer ***Shall*** reset the ***MessageIDCounter***. The Protocol Layer ***Shall*** transition Protocol Layer Message reception to the ***PRL_Rx_Wait_for_PHY_Message*** state (see Section 6.11.2.3.1) 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.

6.11.2.2.1.4 PRL_Tx_Construct_Message State

On entry to the ***PRL_Tx_Construct_Message*** state the Protocol Layer ***Shall*** construct the Message requested by the Policy Engine, or resend a previously constructed Message, and then pass this Message to the PHY Layer.

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

- The Message has been sent to the PHY Layer.

6.11.2.2.1.5 PRL_Tx_Wait_for_PHY_Response State

On entry to the ***PRL_Tx_Wait_for_PHY_Response*** state, once the Message has been sent, the Protocol Layer ***Shall*** initialize and run the ***CRCReceiveTimer*** (see Section 6.6.1).

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).

6.11.2.2.1.6 PRL_Tx_Match_MessageID State

On entry to the ***PRL_Tx_Match_MessageID*** state the Protocol Layer ***Shall*** compare the ***MessageIDCounter*** and the ***MessageID*** of the received ***GoodCRC*** Message.

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

- The MessageIDCounter and the ***MessageID*** of the received ***GoodCRC*** Message match.

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

- The MessageIDCounter and the ***MessageID*** of the received ***GoodCRC*** Message do not match.

6.11.2.2.1.7 PRL_Tx_Message_Sent State

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

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

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

6.11.2.2.1.8 PRL_Tx_Check_RetryCounter State

On entry to the ***PRL_Tx_Check_RetryCounter*** state the Protocol Layer in a DFP or UFP ***Shall*** increment the value of the ***RetryCounter*** and then check it in order to determine whether it is necessary to retry sending the Message. Note that Cable Plugs do not retry Messages and so do not use the ***RetryCounter***.

The Protocol Layer ***Shall*** transition to the ***PRL_Tx_Construct_Message*** state in order to retry Message sending when:

- ***RetryCounter*** ≤ ***nRetryCount*** and
- This is not a Cable Plug and
- This is an Extended Message with ***Data Size*** ≤ ***MaxExtendedMsgLegacyLen*** or
- This is an Extended Message that has been Chunked.

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

- ***RetryCounter*** > ***nRetryCount*** or
- This is a Cable Plug, which does not retry.
- This is an Extended Message with ***Data Size*** > ***MaxExtendedMsgLegacyLen*** that has not been Chunked.

6.11.2.2.1.9 PRL_Tx_Transmission_Error State

On entry to the ***PRL_Tx_Transmission_Error*** state the Protocol Layer ***Shall*** increment the ***MessageIDCounter*** and inform the Policy Engine of the transmission error.

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

- The Policy Engine has been informed of the transmission error.

6.11.2.2.1.10 PRL_Tx_Discard_Message State

Protocol Layer Message transmission ***Shall*** enter the ***PRL_Tx_Discard_Message*** state whenever:

- Protocol Layer Message reception receives an incoming Message or
- The Fast Role Swap signal is being transmitted (see Section 5.8.5.6)
- The Fast Role Swap signal is detected (see Section 5.8.6.3).

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 6.10 and increment the ***MessageIDCounter***.

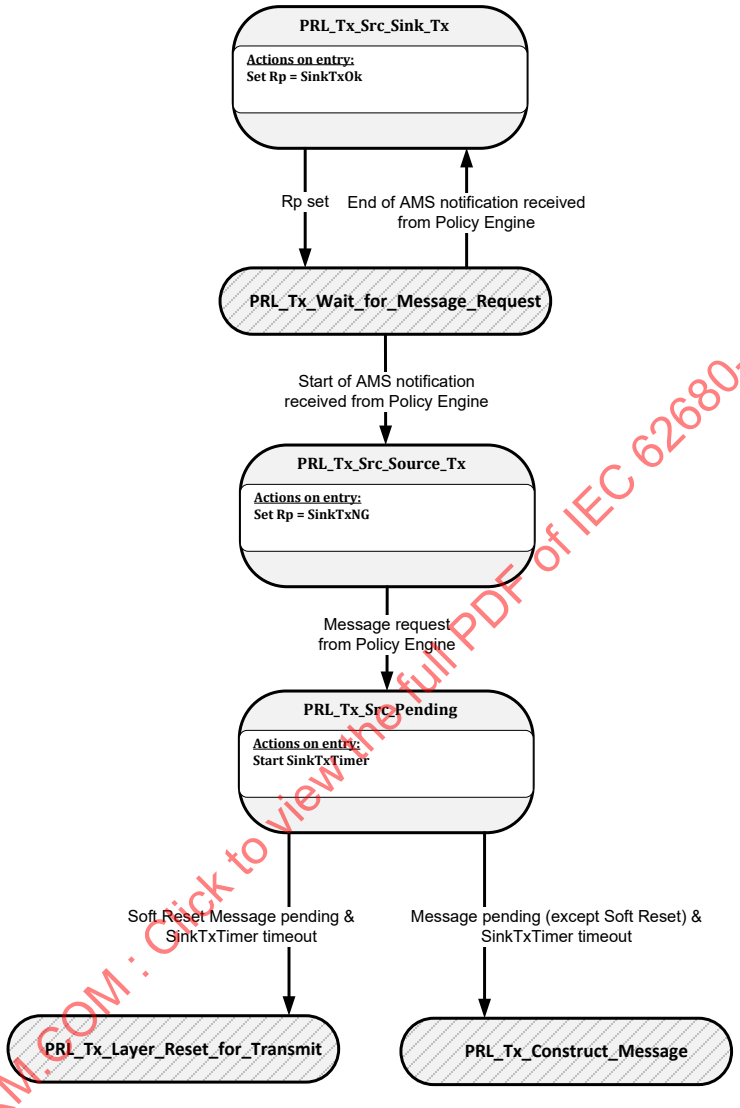
The Protocol Layer ***Shall*** transition to the ***PRL_Tx_PHY_Layer_Reset*** state when:

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

6.11.2.2.2 Source Protocol Layer Message Transmission State Diagram

Figure 6-55 shows the state behavior for the Protocol Layer in a Source when transmitting a Message.

Figure 6-55 Source Protocol Layer Message Transmission State Diagram



6.11.2.2.2.1 PRL_Tx_Src_Sink_Tx State

In the ***PRL_Tx_Src_Sink_Tx*** state the Source sets Rp to ***SinkTxOk*** allowing the Sink to start an Atomic Message Sequence (AMS).

The Protocol Layer in a Source ***Shall*** transition from the ***PRL_Tx_Wait_for_Message_Request*** state to the ***PRL_Tx_Src_Sink_Tx*** state when:

- A notification is received from the Policy Engine that the end of an AMS has been reached.

On entry to the ***PRL_Tx_Src_Sink_Tx*** state the Protocol Layer ***Shall*** request the PHY Layer to Rp to ***SinkTxOk***.

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

- Rp has been set

6.11.2.2.2.2 PRL_Tx_Src_Source_Tx State

In the ***PRL_Tx_Src_Source_Tx*** state the Source sets Rp to ***SinkTxNG*** allowing the Source to start an Atomic Message Sequence (AMS).

The Protocol Layer in a Source ***Shall*** transition from the ***PRL_Tx_Wait_for_Message_Request*** state to the ***PRL_Tx_Src_Source_Tx*** state when:

- A notification is received from the Policy Engine that an AMS will be starting.

On entry to the ***PRL_Tx_Src_Source_Tx*** state the Protocol Layer ***Shall*** set Rp to ***SinkTxNG***.

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

- A Message request is received from the Policy Engine.

6.11.2.2.2.3 PRL_Tx_Src_Pending State

In the ***PRL_Tx_Src_Pending*** state the Protocol Layer has a Message buffered ready for transmission.

On entry to the ***PRL_Tx_Src_Pending*** state the ***SinkTxTimer*** ***Shall*** be initialized and run.

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

- The pending Message request from the Policy Engine is not a ***Soft_Reset*** Message and
- The ***SinkTxTimer*** times out.

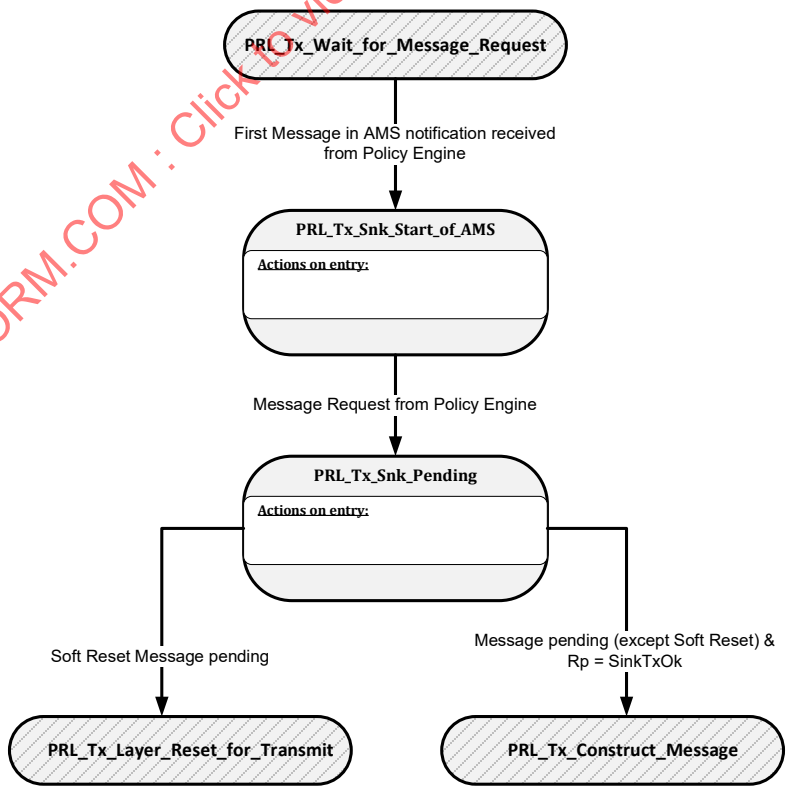
The Protocol Layer ***Shall*** transition to the ***PRL_Tx_Layer_Reset_for_Transmit*** state when:

- The pending Message request from the Policy Engine is a ***Soft_Reset*** Message and
- The ***SinkTxTimer*** times out.

6.11.2.2.3 Sink Protocol Layer Message Transmission State Diagram

Figure 6-56 shows the state behavior for the Protocol Layer in a Source when transmitting a Message.

Figure 6-56 Sink Protocol Layer Message Transmission State Diagram



6.11.2.2.3.1 PRL_Tx_Snk_Start_of_AMS State

In the ***PRL_Tx_Snk_Start_of_AMS*** state the Protocol Layer waits for the first Message in a Sink initiated AMS.

The Protocol Layer in a Sink ***Shall*** transition from the ***PRL_Tx_Wait_for_Message_Request*** state to the ***PRL_Tx_Snk_Start_of_AMS*** state when:

- A notification is received from the Policy Engine that the next Message the Sink will send is the start of an AMS.

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

- A Message request is received from the Policy Engine.

6.11.2.2.3.2 PRL_Tx_Snk_Pending State

In the ***PRL_Tx_Snk_Pending*** state the Protocol Layer has the first Message in a Sink initiated AMS ready to send and is waiting for Rp to transition to ***SinkTxOk*** before sending the Message.

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

- A Message is Pending that is not a ***Soft_Reset*** Message and
- Rp is set to ***SinkTxOk.***

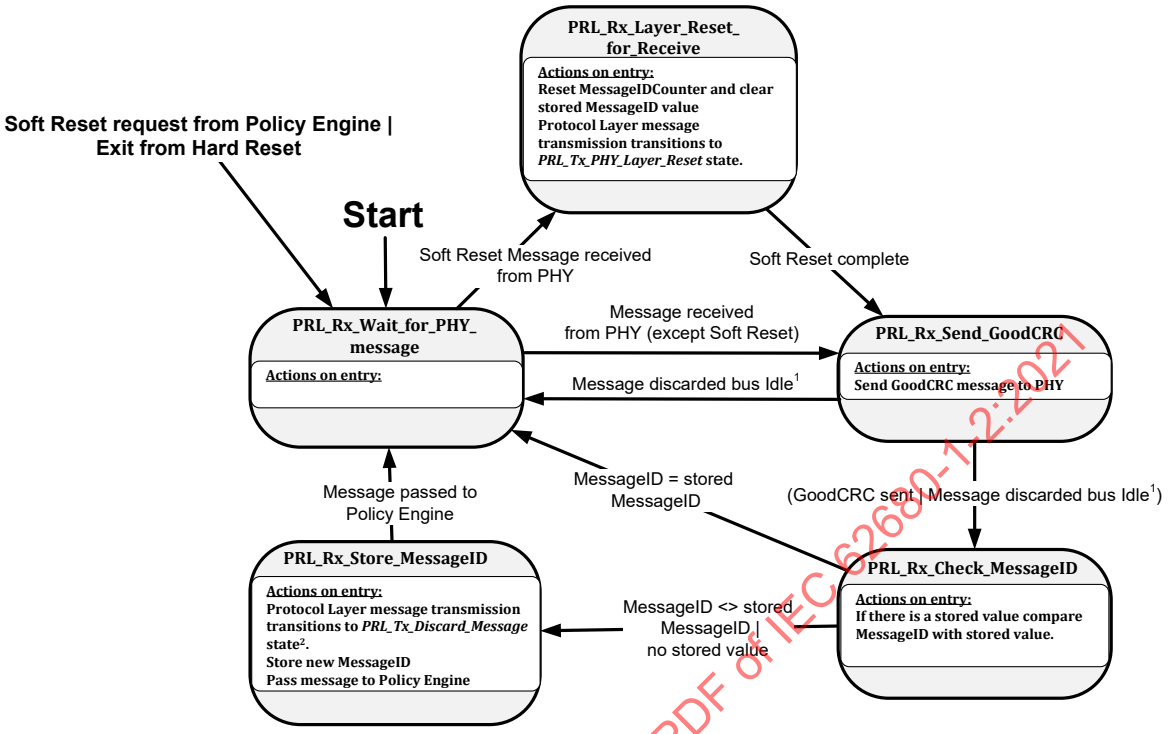
The Protocol Layer ***Shall*** transition to the ***PRL_Tx_Layer_Reset_for_Transmit*** state when:

- A ***Soft_Reset*** Message is pending

6.11.2.3 Protocol Layer Message Reception

Figure 6-57 shows the state behavior for the Protocol Layer when receiving a Message.

Figure 6-57 Protocol layer Message reception



[1] This indication is sent by the PHY when a message has been ***Discarded*** due to CC being busy, and after CC becomes idle again (see Section 5.7). Two alternate allowable transitions are shown.

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

6.11.2.3.1 PRL_Rx_Wait_for_PHY_Message state

The Protocol Layer ***Shall*** enter the ***PRL_Rx_Wait_for_PHY_Message*** state:

- At startup.
- As a result of a Soft Reset request from the Policy Engine.
- On exit from a Hard Reset.

In the ***PRL_Rx_Wait_for_PHY_Message*** state the Protocol Layer waits until the PHY Layer passes up a received Message.

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

- A Message is passed up from the PHY Layer.

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

- A ***Soft_Reset*** Message is received from the PHY Layer.

6.11.2.3.2 PRL_Rx_Layer_Reset_for_Receive state

On entry to the ***PRL_Rx_Layer_Reset_for_Receive*** state the Protocol Layer ***Shall*** reset the ***MessageIDCounter*** and clear the stored ***MessageID***. The Protocol Layer ***Shall*** transition Protocol Layer Message transmission to the ***PRL_Tx_Wait_for_Message_Request*** state (see Section 6.11.2.2.1.1).

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

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

6.11.2.3.3 PRL_Rx_Send_GoodCRC state

On entry to the ***PRL_Rx_Send_GoodCRC*** state the Protocol Layer ***Shall*** construct a ***GoodCRC*** Message and request the PHY Layer to transmit it.

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

- The ***GoodCRC*** Message has been passed to the PHY Layer.

When the PHY Layer indicates that a Message has been ***Discarded*** due to CC being busy but CC is now idle (see Section 5.7), the Protocol Layer ***Shall*** either:

- Transition to the ***PRL_Rx_Check_MessageID state*** or
- Transition to the ***PRL_Rx_Wait_for_PHY_Message*** state.

6.11.2.3.4 PRL_Rx_Check_MessageID state

On entry to the ***PRL_Rx_Check_MessageID*** state the Protocol Layer ***Shall*** compare the ***MessageID*** of the received Message with its stored value if a value has previously been stored.

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

- The ***MessageID*** of the received Message equals the stored ***MessageID*** value since this is a Message retry which ***Shall*** be ***Discarded***.

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

- The ***MessageID*** of the received Message does not equal the stored ***MessageID*** value since this is a new Message or
- This is the first received Message and no ***MessageID*** value is currently stored.

6.11.2.3.5 PRL_Rx_Store_MessageID state

On entry to the ***PRL_Rx_Store_MessageID*** state the Protocol Layer ***Shall*** transition Protocol Layer Message transmission to the ***PRL_Tx_Discard_Message*** state (except when a ***Ping*** Message has been received in which case the ***PRL_Tx_Discard_Message*** state ***Should Not*** be entered), replace the stored value of ***MessageID*** with the value of ***MessageID*** in the received Message and pass the Message up to the Policy Engine.

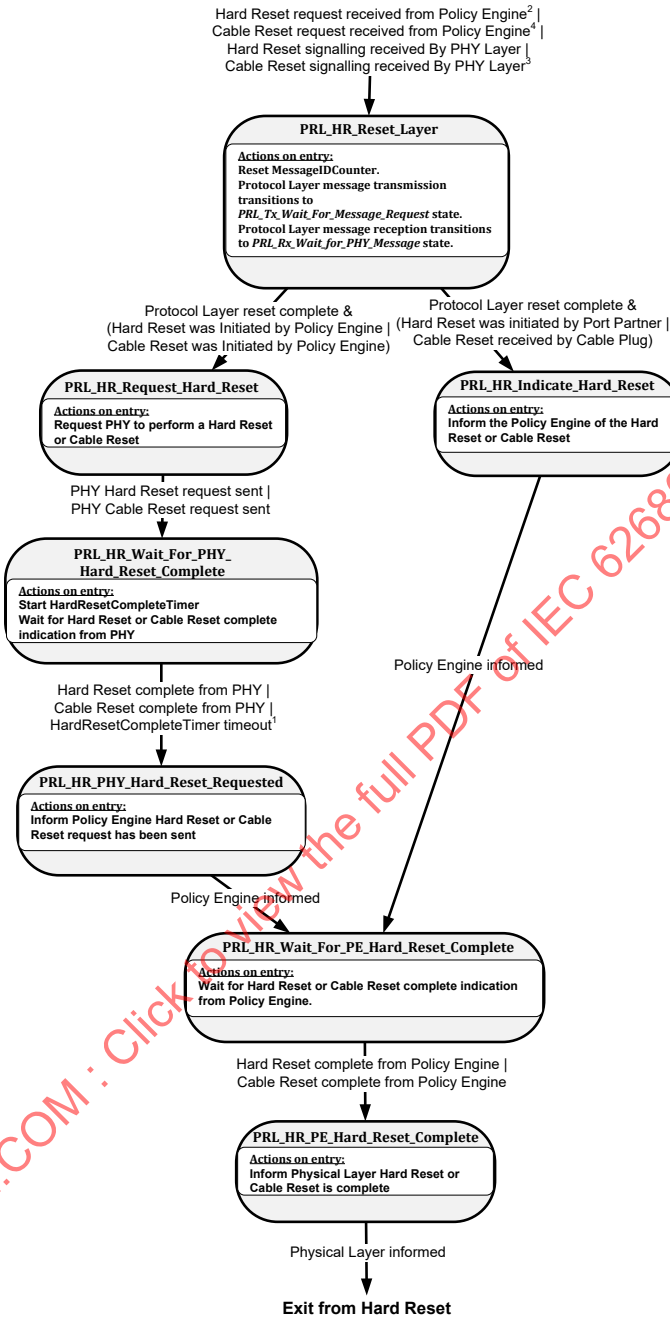
The Protocol Layer ***Shall*** transition to the ***PRL_Rx_Wait_for_PHY_Message*** state when:

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

6.11.2.4 Hard Reset operation

Figure 6-58 shows the state behavior for the Protocol Layer when receiving a Hard Reset or Cable Reset request from the Policy Engine or ***Hard Reset*** Signaling or ***Cable Reset*** Signaling from the Physical Layer (see also Section 6.8.3 and Section 6.8.4).

Figure 6-58 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

6.11.2.4.1 PRL_HR_Reset_Layer state

The ***PRL_HR_Reset_Layer*** State defines the mode of operation of both the Protocol Layer transmission and reception state machines during a Hard Reset or Cable Reset. During Hard Reset no USB Power Delivery Protocol Messages are sent or received; only ***Hard Reset*** Signaling is present after which the communication channel is assumed to have been disabled by the Physical Layer until completion of the Hard Reset. During Cable Reset no USB Power Delivery Protocol Messages are sent to or received by the Cable Plug but other USB Power Delivery communication ***May*** continue.

The Protocol Layer ***Shall*** enter the ***PRL_HR_Reset_Layer*** state from any other state when:

- A Hard-Reset Request is received from the Policy Engine or
- ***Hard Reset*** Signaling is received from the Physical Layer or
- A Cable Reset Request is received from the Policy Engine or
- ***Cable Reset*** Signaling is received from the Physical Layer.

On entry to the ***PRL_HR_Reset_Layer*** state the Protocol Layer ***Shall*** reset the ***MessageIDCounter***. It ***Shall*** also reset the states of the Protocol Layer transmission and reception state machines to their starting points. The Protocol Layer transmission state machine ***Shall*** transition to the ***PRL_Tx_Wait_for_Message_Request*** state. The Protocol Layer reception state machine ***Shall*** transition to the ***PRL_Rx_Wait_for_PHY_Message*** state.

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

- The Protocol Layer's reset is complete and
  - The Hard-Reset request has originated from the Policy Engine or
  - The Cable Reset request has originated from the Policy Engine.

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

- The Protocol Layer's reset is complete and
  - The Hard-Reset request has been passed up from the Physical Layer or
  - A Cable Reset request has been passed up from the Physical Layer (Cable Plug only).

6.11.2.4.2 PRL_HR_Indicate_Hard_Reset state

On entry to the ***PRL_HR_Indicate_Hard_Reset*** state the Protocol Layer ***Shall*** indicate to the Policy Engine that either ***Hard Reset*** Signaling or ***Cable Reset*** Signaling has been received.

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

- The Indication to the Policy Engine has been sent.

6.11.2.4.3 PRL_HR_Request_Hard_Reset state

On entry to the ***PRL_HR_Request_Hard_Reset*** state the Protocol Layer ***Shall*** request the Physical Layer to send either ***Hard Reset*** Signaling or ***Cable Reset*** signaling.

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

- The Physical Layer ***Hard Reset*** Signaling request has been sent or
- The Physical Layer ***Cable Reset*** Signaling request has been sent.

6.11.2.4.4 PRL_HR_Wait_for_PHY_Hard_Reset_Complete state

In the ***PRL_HR_Wait_for_PHY_Hard_Reset_Complete*** state the Protocol Layer ***Shall*** start the ***HardResetCompleteTimer*** and wait for the PHY Layer to indicate that the Hard Reset or Cable Reset has been completed.

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

- A Hard-Reset complete indication is received from the PHY Layer or
- A Cable Reset complete indication is received from the PHY Layer or
- The ***HardResetCompleteTimer*** times out.

6.11.2.4.5 PRL_HR_PHY_Hard_Reset_Requested state

On entry to the ***PRL_HR_PHY_Hard_Reset_Requested*** state the Protocol Layer ***Shall*** inform the Policy Engine that the PHY Layer has been requested to perform a Hard Reset or Cable Reset.

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

- The Indication to the Policy Engine has been sent.

6.11.2.4.6 PRL_HR_Wait_for_PE_Hard_Reset_Complete state

In the ***PRL_HR_Wait_for_PE_Hard_Reset_Complete*** state the Protocol Layer ***Shall*** wait for the Policy Engine to indicate that the Hard Reset or Cable Reset has been completed.

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

- A Hard-Reset complete indication is received from the Policy Engine or
- A Cable Reset complete indication is received from the Policy Engine.

6.11.2.4.7 PRL_HR_PE_Hard_Reset_Complete

On entry to the ***PRL_HR_PE_Hard_Reset_Complete*** state the Protocol Layer ***Shall*** inform the Physical Layer that the Hard Reset or Cable Reset is complete.

The Protocol Layer ***Shall*** exit from the Hard Reset and return to normal operation when:

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

6.11.3 List of Protocol Layer States

Table 6-68 lists the states used by the various state machines.

Table 6-68 Protocol Layer States

State name	Reference
Protocol Layer Message Transmission	
Common Protocol Layer Message Transmission	
PRL_Tx_PHY_Layer_Reset	Section 6.11.2.2.1.1
PRL_Tx_Wait_for_Message_Request	Section 6.11.2.2.1.2
PRL_Tx_Layer_Reset_for_Transmit	Section 6.11.2.2.1.3
PRL_Tx_Construct_Message	Section 6.11.2.2.1.4
PRL_Tx_Wait_for_PHY_Response	Section 6.11.2.2.1.5
PRL_Tx_Match_MessageID	Section 6.11.2.2.1.6
PRL_Tx_Message_Sent	Section 6.11.2.2.1.7
PRL_Tx_Check_RetryCounter	Section 6.11.2.2.1.8
PRL_Tx_Transmission_Error	Section 6.11.2.2.1.9
PRL_Tx_Discard_Message	Section 6.11.2.2.1.10
Source Protocol Layer Message Transmission	
PRL_Tx_Src_Sink_Tx	Section 6.11.2.2.2.1
PRL_Tx_Src_Source_Tx	Section 6.11.2.2.2.2
PRL_Tx_Src_Pending	Section 6.11.2.2.2.3
Sink Protocol Layer Message Transmission	
PRL_Tx_Snk_Start_of_AMS	Section 6.11.2.2.3.1
PRL_Tx_Snk_Pending	Section 6.11.2.2.3.2
Protocol Layer Message Reception	
PRL_Rx_Wait_for_PHY_Message	Section 6.11.2.3.1
PRL_Rx_Layer_Reset_for_Receive	Section 6.11.2.3.2
PRL_Rx_Send_GoodCRC	Section 6.11.2.3.3
PRL_Rx_Check_MessageID	Section 6.11.2.3.4
PRL_Rx_Store_MessageID	Section 6.11.2.3.5
Hard Reset Operation	
PRL_HR_Reset_Layer	Section 6.11.2.4.1
PRL_HR_Indicate_Hard_Reset	Section 6.11.2.4.2
PRL_HR_Request_Hard_Reset	Section 6.11.2.4.3
PRL_HR_Wait_for_PHY_Hard_Reset_Complete	Section 6.11.2.4.4
PRL_HR_PHY_Hard_Reset_Requested	Section 6.11.2.4.5
PRL_HR_Wait_for_PE_Hard_Reset_Complete	Section 6.11.2.4.6
PRL_HR_PE_Hard_Reset_Complete	Section 6.11.2.4.7
Chunking	
Chunked Rx	
RCH_Wait_For_Message_From_Protocol_Layer	Section 6.11.2.1.2.1
RCH_Pass_Up_Message	Section 6.11.2.1.2.2
RCH_Processing_Extended_Message	Section 6.11.2.1.2.3
RCH_Requesting_Chunk	Section 6.11.2.1.2.4

State name	Reference
RCH_Waiting_Chunk	Section 6.11.2.1.2.5
RCH_Report_Error	Section 6.11.2.1.2.6
Chunked Tx	
TCH_Wait_For_Message_Request_From_Policy_Engine	Section 6.11.2.1.3.1
TCH_Pass_Down_Message	Section 6.11.2.1.3.2
TCH_Wait_For_Transmision_Complete	Section 6.11.2.1.3.3
TCH_Message_Sent	Section 6.11.2.1.3.4
TCH_Prepare_To_Send_Chunked_Message	Section 6.11.2.1.3.5
TCH_Construct_Chunked_Message	Section 6.11.2.1.3.6
TCH_Sending_Chunked_Message	Section 6.11.2.1.3.7
TCH_Wait_Chunk_Request	Section 6.11.2.1.3.8
TCH_Message_Received	Section 6.11.2.1.3.9
TCH_Report_Error	Section 6.11.2.1.3.10
Chunked Message Router	
RTR_Wait_for_Message_From_Protocol_Layer	Section 6.11.2.1.4.1
RTR_Rx_Chunks	Section 6.11.2.1.4.2
RTR_Ping	Section 6.11.2.1.4.3
RTR_Tx_Chunks	Section 6.11.2.1.4.4

6.12 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
- NS – Not Supported; ***Shall*** result in a ***Not_Supported*** Message response by this Port/Cable Plug when received.
- I – ***Ignore***; ***Shall*** be ***Ignored*** by this Port/Cable Plug when received.
- NK – NAK; this Port/Cable Plug ***Shall*** return Responder NAK to this Command when received
- NA – Not allowed; ***Shall Not*** be transmitted by this Port/Cable Plug.
- DR – Don't Recognize; there ***Shall*** no response at all (i.e. not even a ***GoodCRC*** Message) from this Port/Cable Plug when received.

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

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

Note: that where NS/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 6.8.1).

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.

6.12.1 Applicability of Control Messages

Table 6-69 details Control Messages that ***Shall*/*Should*/ *Shall Not*** be transmitted and received by a Source, Sink, Cable Plug or VPD. Requirements for Dual-Role Power Ports and Dual-Role Data Ports ***Shall*** override any requirements for Source-only or Sink-Only Ports.

Table 6-69 Applicability of Control Messages

Message Type	Source	Sink	Dual-Role Power	Dual-Role Data	Cable Plug	VPD[12]
Transmitted Message						
Accept	N	N			N	N
DR_Swap	O	O		N	NA	NA
FR_Swap	NA	NA	R		NA	NA
Get_Country_Codes	CN[10]/NA	CN[10]/NA			NA	NA
Get_PPS_Status	NA	CN[9]			NA	NA
Get_Sink_Cap	R	NA	N		NA	NA
Get_Sink_Cap_Extended	R	NA	R		NA	NA
Get_Source_Cap	NA	R	N		NA	NA
Get_Source_Cap_Extended	NA	R	R		NA	NA
Get_Status	R	R			NA	NA
GoodCRC	N	N			N	N
GotoMin	CN[1]/O	NA			NA	NA
Not_Supported	N	N			N	NA
Ping	O	NA			NA	NA
Data_Reset	CN[13]/R	CN[13]/R			NA	NA
PR_Swap	NA	NA	N		NA	NA
PS_RDY	N	CN[4]/NA	N		NA	NA
Reject	N	NA	O	O	NA	NA
Soft_Reset	N	N			NA	NA
VCONN_Swap	R	R			NA	NA
Wait	CN[2]/O	NA	O	O	NA	NA
Received Message						
Accept	N	N	N	N	I	I
DR_Swap	O/NS	O/NS		N	I	I
FR_Swap	NS	NS	CN[7]/NS		I	I
Get_Country_Codes	CN[10]/NS	CN[10]/NS			I	I
Get_PPS_Status	CN[9]/NS	NS			I	I
Get_Sink_Cap	NS	N	N		I	I
Get_Sink_Cap_Extended	NS	N	N		I	I
Get_Source_Cap	N	NS	N		I	I
Get_Source_Cap_Extended	CN[5]/NS	NS	CN[5]/NS		I	I
Get_Status	CN[6]/NS	CN[6]/NS	CN[6]/NS		CN[11]/I	I
GoodCRC	N	N			N	N
GotoMin	NS	R[3]			I	I

Message Type	Source	Sink	Dual-Role Power	Dual-Role Data	Cable Plug	VPD[12]
Not_Supported	N	N			CN[11]/I	I
Ping	NS	I			I	I
Data_Reset	CN[13]/R	CN[13]/R			I	I
PR_Swap	NS	NS	N		I	I
PS_RDY	CN[4]/NS	N	N		I	I
Reject	CN[8]/NS	N	N	N	I	I
Soft_Reset	N	N			N	N
VCONN_Swap	CN[4]/ NS	CN[4]/ NS			I	I
Wait	CN[8]/NS	N	N	N	I	I

Note 1: ***Shall*** be supported by a Hub with multiple Downstream Ports. ***Should*** be supported by a Host with multiple Downstream Ports.
Note 2: ***Shall*** be supported when transmission of ***GotoMin*** Messages is supported.
Note 3: ***Should*** be supported by Sinks which use PD power for charging.
Note 4: ***Shall*** be supported by any Port that can supply VCONN.
Note 5: ***Shall*** be supported products that support the ***Source_Capabilities_Extended*** Message.
Note 6: ***Shall*** be supported by Sources that support the ***Alert*** Message.
Note 7: ***Shall*** be supported when the Fast Role Swap signal is supported.
Note 8: ***Shall*** be supported when ***VCONN_Swap*** is supported.
Note 9: ***Shall*** be supported when PPS is supported.
Note 10: ***Shall*** be supported when required by a country authority.
Note 11: ***Shall*** be supported by Active Cables.
Note 12: VPD includes CT-VPDs when not Connected to a Charger. PD communication with a CT-VPD ***Shall*** only take place when not Connected to a Charger.
Note 13: ***Shall*** be supported by products that support ***[USB4]***.

6.12.2 Applicability of Data Messages

Table 6-70 details Data Messages (except for VDM Commands) that ***Shall***/***Should***/ ***Shall Not*** be transmitted and received by a Source, Sink, Cable Plug or VPD. Requirements for Dual-Role Power Ports ***Shall*** override any requirements for Source-only or Sink-Only Ports.

Table 6-70 Applicability of Data Messages

Message Type	Source	Sink	Dual-Role Power	Cable Plug SOP'	Cable Plug SOP''	VPD[6]
Transmitted Message						
Source_Capabilities	N	NA	N	NA	NA	NA
Request	NA	N		NA	NA	NA
Get_Country_Info	CN[5]/O	CN[5]/O		NA	NA	NA
BIST	N[1]	N[1]		NA	NA	NA
Sink_Capabilities	NA	N	N	NA	NA	NA
Battery_Status	CN[2]	CN[2]		NA	NA	NA
Alert	R	R		NA	NA	NA
Enter_USB	CN[7]/O	CN[7]/O		NA	NA	NA
Received Message						
Source_Capabilities	NS	N	N	I	I	I
Request	N	NS		I	I	I
Get_Country_Info	CN[5]/NS	CN[5]/NS		I	I	I

Message Type	Source	Sink	Dual-Role Power	Cable Plug SOP'	Cable Plug SOP''	VPD[6]
BIST	N[1]	N[1]		N[1]	N[1]	N[1]
Sink_Capabilities	CN[4]	NS	CN[4]	I	I	I
Battery_Status	CN[3]/NS	CN[3]/NS		I	I	I
Alert	R/NS	R/NS		I	I	I
Enter_USB	CN[7]/O	CN[7]/O		CN[7]/O	I	I

Note 1: For details of which BIST Modes and Messages ***Shall*** be supported see Section 5.9 and Section 6.4.3.
Note 2: ***Shall*** be supported by products that contain batteries.
Note 3: ***Shall*** be supported by products that support the *Get_Battery_Status* Message.
Note 4: ***Shall*** be supported by products that support the *Get_Sink_Cap* Message.
Note 5: ***Shall*** be supported when required by a country authority.
Note 6: VPD includes CT-VPDs when not Connected to a Charger. PD communication with a CT-VPD ***Shall*** only take place when not Connected to a Charger.
Note 7: ***Shall*** be supported by products that support *[USB4]*.

6.12.3 Applicability of Extended Messages

Table 6-71 details Extended Messages (except for Extended VDM Commands) that ***Shall*/*Should*/ *Shall Not*** be transmitted and received by a Source, Sink, Cable Plug or VPD. Requirements for Dual-Role Power Ports ***Shall*** override any requirements for Source-only or Sink-Only Ports.

Table 6-71 Applicability of Extended Messages

Message Type	Source	Sink	Dual-Role Power	Cable Plug SOP'	Cable Plug SOP''	VPD[1,3]
Transmitted Message						
Battery_Capabilities	CN[1]/NA	CN[1]/NA		NA	NA	NA
Country_Codes	CN[10]/NA	CN[10]/NA		NA	NA	NA
Country_Info	CN[10]/NA	CN[10]/NA		NA	NA	NA
Firmware_Update_Request	CN[7]/NA	CN[7]/NA		NA	NA	NA
Firmware_Update_Response	CN[7]/NA	CN[7]/NA		CN[7]/NA	O	NA
Get_Battery_Cap	R	R		NA	NA	NA
Get_Battery_Status	R	R		NA	NA	NA
Get_Manufacturer_Info	R	R		NA	NA	NA
Manufacturer_Info	R	R		R	NA	NA
PPS_Status	CN[8]/NA	NA		NA	NA	NA
Security_Request	CN[6]/NA	CN[6]/NA		NA	NA	NA
Security_Response	CN[6]/NA	CN[6]/NA		CN[6]/NA	NA	NA
Sink_Capabilities_Extended	NA	N	N	NA	NA	NA
Source_Capabilities_Extended	R	NA	R	NA	NA	NA
Status	R	R	R	CN[12]/NA	CN[12]/NA	NA
Received Message						
Battery_Capabilities	CN[4]/NS	CN[4]/NS		I	I	I
Country_Codes	CN[10]/NS	CN[10]/NS		I	I	I
Country_Info	CN[10]/NS	CN[10]/NS		I	I	I
Firmware_Update_Request	CN[7]/NS	CN[7]/NS		CN[7]/I	O	I
Firmware_Update_Response	CN[7]/NS	CN[7]/NS		I	I	I

Message Type	Source	Sink	Dual-Role Power	Cable Plug SOP'	Cable Plug SOP''	VPD[1,3]
Get_Battery_Cap	CN[1]/NS	CN[1]/NS		I	I	I
Get_Battery_Status	CN[1]/NS	CN[1]/NS		I	I	I
Get_Manufacturer_Info	R/NS	R/NS		R/I	I	I
Manufacturer_Info	CN[5]/NS	CN[5]/NS		I	I	I
PPS_Status	NS	CN9/NS		I	I	I
Security_Request	CN[6]/NS	CN[6]/NS		CN[6]/I	I	I
Security_Response	CN[6]/NS	CN[6]/NS		I	I	I
Sink_Capabilities_Extended	CN[11]/NS	NS	CN[11]/NS	I	I	I
Source_Capabilities_Extended	NS	CN[2]/NS	CN[2]/NS	I	I	I
Status	CN[3]/NS	CN[3]/NS		I	I	I

Note 1: ***Shall*** be supported by products that contain batteries.
Note 2: ***Shall*** be supported by products that can transmit the *Get_Source_Cap_Extended* Message.
Note 3: ***Shall*** be supported by products that can transmit the *Get_Status* Message.
Note 4: ***Shall*** be supported by products that can transmit the *Get_Battery_Cap* Message.
Note 5: ***Shall*** be supported by products that can transmit the *Get_Manufacturer_Info* Message
Note 6: ***Shall*** be supported by products that support USB security communication as defined in *[USBTypeCAuthentication 1.0]*
Note 7: ***Shall*** be supported by products that support USB firmware update communication as defined in *[USBPDFirmwareUpdate 1.0]*
Note 8: ***Shall*** be supported when PPS is supported.
Note 9: ***Shall*** be supported by products that can transmit the *Get_PPS_Status*.
Note 10: ***Shall*** be supported when required by a country authority.
Note 11: ***Shall*** be supported by products that can transmit the *Get_Sink_Cap_Extended* Message.
Note 12: ***Shall*** be supported by Active Cables.
Note 13: VPD includes CT-VPDs when not Connected to a Charger. PD communication with a CT-VPD ***Shall*** only take place when not Connected to a Charger.

6.12.4 Applicability of Structured VDM Commands

Table 6-72 details Structured VDM Commands that ***Shall***/***Should***/ ***Shall Not*** be transmitted and received by a DFP, UFP, Cable Plug or VPD. If Structured VDMs are not supported, the DFP or UFP receiving a VDM Command ***Shall*** send a *Not_Supported* Message in response.

Table 6-72 Applicability of Structured VDM Commands

Command Type	DFP	UFP	Cable Plug SOP'	Cable Plug SOP''	VPD[4]
Transmitted Command Request					
Discover Identity	CN[1,6]/R	R[2]	NA	NA	NA
Discover SVIDs	CN[1]/O	O	NA	NA	NA
Discover Modes	CN[1]/O	O	NA	NA	NA
Enter Mode	CN[1]/NA	NA	NA	NA	NA
Exit Mode	CN[1]/NA	NA	NA	NA	NA
Attention	O	O	NA	NA	NA
Received Command Request/Transmitted Command Response					
Discover Identity	CN[5,6]/R/NK[3]	CN[1,6]/R/NK[3]	N	I	N
Discover SVIDs	O/NK[3]	CN[1]/NK[3]	CN[1]/NK	I	NK
Discover Modes	O/NK[3]	CN[1]/NK[3]	CN[1]/NK	I	NK
Enter Mode	NK[3]	CN[1]/NK[3]	CN[1]/NK	O	NK

Command Type	DFP	UFP	Cable Plug SOP'	Cable Plug SOP''	VPD[4]
Exit Mode	NK[3]	CN[1]/NK[3]	CN[1]/NK	O	NK
Attention	O/I[3]	O/I[3]	I	I	I

Note 1: ***Shall*** be supported when Modal Operation is supported.
Note 2: ***May*** be transmitted by a UFP/Source during discovery (see Section 6.4.4.3.1 and Section 8.3.3.24.3).
Note 3: If Structured VDMs are not supported, the DFP or UFP receiving a VDM Command ***Shall*** send a ***Not_Supported*** Message in response.
Note 4: VPD includes CT-VPDs when not Connected to a Charger. PD communication with a CT-VPD ***Shall*** only take place when not Connected to a Charger.
Note 5: ***Shall*** be supported by products with more than one DFP.
Note 6: ***Shall*** be supported by products that support ***[USB4]***.

6.12.5 Applicability of Reset Signaling

Table 6-73 details Reset Signaling that ***Shall***/***Should***/ ***Shall Not*** be transmitted and received by a DFP/UFP or Cable Plug.

Table 6-73 Applicability of Reset Signaling

Signaling Type	DFP	UFP	Cable Plug SOP'	Cable Plug SOP''	VPD[2]
Transmitted Message/Signaling					
Soft_Reset	N	N	NA	NA	NA
Hard Reset	N	N	NA	NA	NA
Cable Reset	CN[1]	CN[1]	NA	NA	NA
Received Message/Signaling					
Soft_Reset	N	N	N	N	N
Hard Reset	N	N	N	N	N
Cable Reset	DR	DR	N	N	N

Note 1: ***Shall*** be supported when transmission of SOP' Packets are supported, and the Port can supply VCONN.
Note 2: VPD includes CT-VPDs when not Connected to a Charger. PD communication with a CT-VPD ***Shall*** only take place when not Connected to a Charger.

6.12.6 Applicability of Fast Role Swap signal

Table 6-73 details the Fast Role Swap signal that ***Shall***/***Should***/ ***Shall Not*** be transmitted and received by a Source or Sink.

Table 6-74 Applicability of Fast Role Swap signal

Command Type	Source	Sink	Dual-Role Power
Transmitted Message/Signaling			
Fast Role Swap	NA	NA	R
Received Message/Signaling			
Fast Role Swap	NA	NA	R

6.13 Value Parameters

Table 6-75 contains value parameters used in this section.

Table 6-75 Value Parameters

Parameter	Description	Value	Unit	Reference
MaxExtendedMsgLen	Maximum length of an Extended Message as expressed in the ***Data Size*** field.	260	Byte	Section 6.4.8
MaxExtendedMsgChunkLen		26	Byte	Section 6.4.8
MaxExtendedMsgLegacyLen		26	Byte	Section 6.4.8

7. Power Supply

7.1 Source Requirements

7.1.1 Behavioral Aspects

A USB PD Source exhibits the following behaviors:

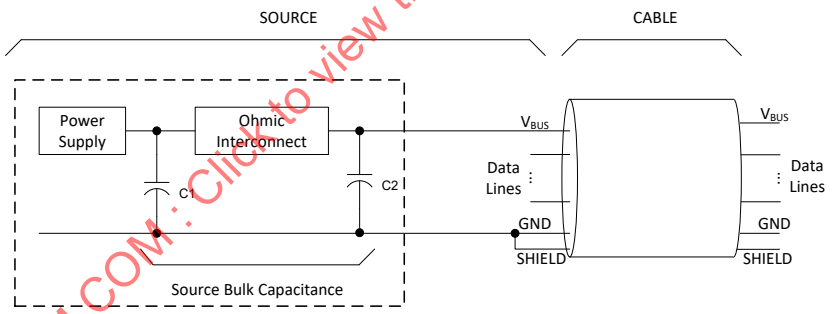
- ***Shall*** supply the default *[USB 2.0]*, *[USB 3.2]*, *[USB Type-C 2.0]* (USB Type-C®) or *[USBBC 1.2]* voltage and current to V_{BUS} when a Contract does not exist (USB Default Operation).
- ***Shall*** follow the requirements as specified in Section 7.1.5 when ***Hard Reset*** Signaling is received.
- ***Shall*** control V_{BUS} voltage transitions as bound by undershoot, overshoot and transition time requirements.

7.1.2 Source Bulk Capacitance

The Source 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



7.1.3 Types of Sources

Consistent with the Power Data Objects discussed in Section 6.4.1, the 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 7.1.8.
- 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 than 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 exposed in the Battery Supply PDO.

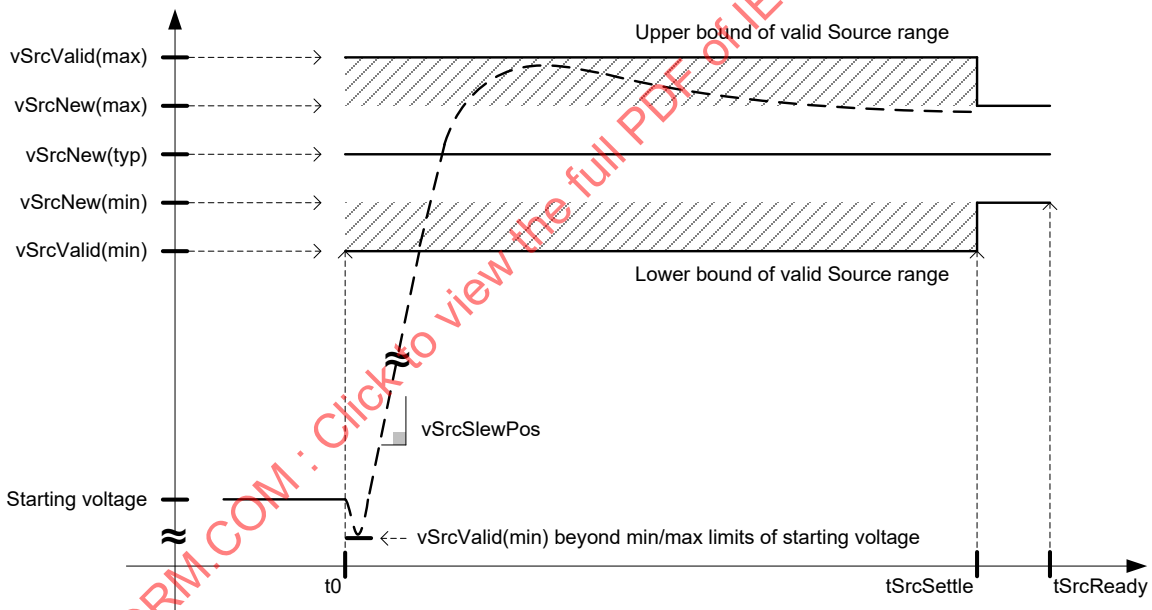
- The Programmable Power Supply (PPS) Augmented PDO (APDO) exposes a Source with an output voltage that can be adjusted programmatically over a defined range. The output voltage of the Programmable Power Supply ***Shall*** remain within a range defined by the relative tolerance ***vPpsNew*** and the absolute band ***vPpsValid***.

7.1.4 Source Transitions

7.1.4.1 Fixed Supply 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. 5V, 9V, 15V 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, t0, 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



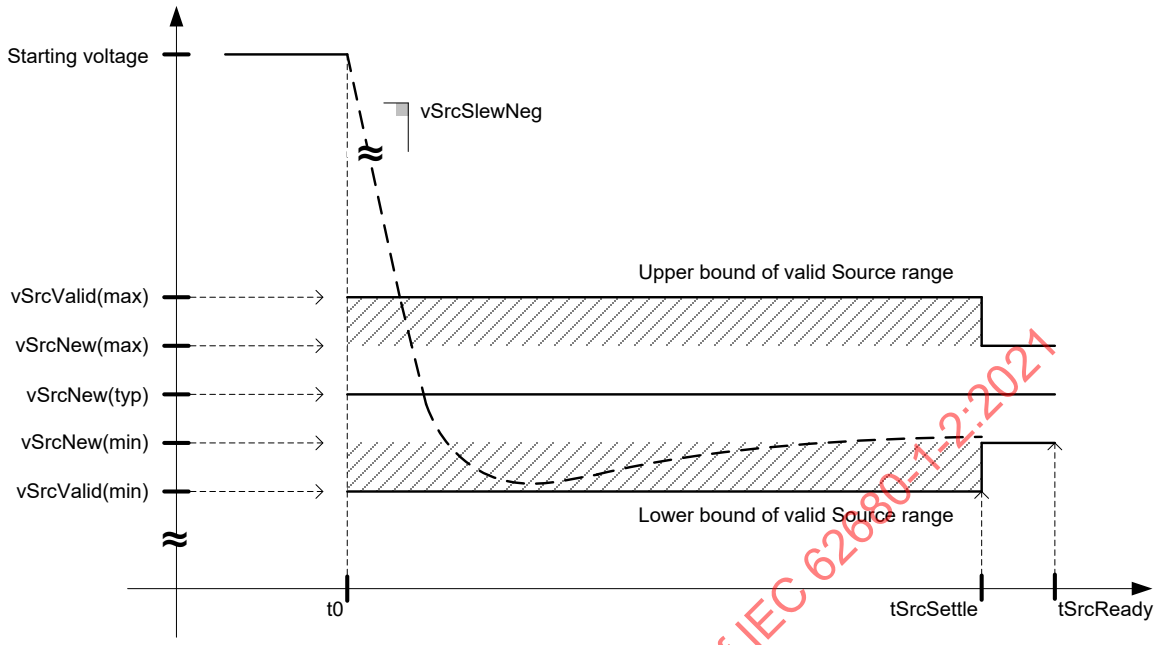
At the start of the positive voltage transition the V_{BUS} voltage level ***Shall Not*** droop ***vSrcValid*** min below either ***vSrcNew*** (i.e., if the starting V_{BUS} voltage level is not ***vSafe5V***) or ***vSafe5V*** as applicable.

Section 7.1.14 lists transitions that are exempt from the ***vSrcSlewPos*** limit.

7.1.4.2 Fixed Supply 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, t0, 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***.

Section 7.1.14 lists transitions that are exempt from the ***vSrcSlewNeg*** limit.

7.1.4.3 Programmable Power Supply Voltage Transitions

The Programmable Power Supply (PPS) ***Shall*** transition V_{BUS} over the defined voltage range in a controlled manner. The Output Voltage value in the Programmable RDO defines the nominal value of the PPS output voltage after completing a voltage change and ***Shall*** settle within the limits defined by ***vPpsNew*** by ***tPpsSrcTransSmall*** for steps smaller than or equal to ***vPpsSmallStep***, or else, within the limits defined by ***vPpsNew*** by ***tPpsSrcTransLarge***, but only in case the Programmable Power Supply is not in CL mode. Any overshoot beyond ***vPpsNew*** ***Shall Not*** exceed ***vPpsValid*** at any time. Any undershoot beyond ***vPpsNew*** ***Shall Not*** exceed ***vPpsValid*** for currents not resulting in CL mode. The PPS output voltage ***May*** change in a step-wise or linear manner and the slew rate of either type of change ***Shall Not*** exceed ***vPpsSlewPos*** for voltage increases or ***vPpsSlewNeg*** for voltage decreases. The nominal requested voltage of all linear voltage changes ***Shall*** equate to an integer number of LSB changes. An LSB change of the PPS output voltage is defined as ***vPpsStep***. A PPS ***Shall*** be able to supply the negotiated current level as it changes its output voltage to the requested level. All PPS voltage increases ***Shall*** result in a voltage that is greater than the previous PPS output voltage. Likewise, all PPS voltage decreases ***Shall*** result in a voltage that is less than the previous PPS output voltage.

Since a Sink can draw current up to the negotiated APDO current level in case of a voltage step, the voltage might not increase to the requested level due to the power supply operating in CL mode. Likewise, since a Sink can have a battery connected to VBUS, the voltage might not decrease to the requested level due to the battery voltage being higher than the output voltage set-point the Source is transitioning to. Were the Source to rely on checking the voltage on VBUS, in either case, to determine when its power supply is ready a PS_RDY would never be sent.

When the PPS voltage steps up or down, a ***PS_RDY*** Message ***Shall*** be sent within:

- ***tPpsSrcTransLarge*** after the last bit of the ***GoodCRC*** Message following the ***Accept*** Message for steps larger than ***vPpsSmallStep.***
- ***tPpsSrcTransSmall*** after the last bit of the ***GoodCRC*** Message following the ***Accept*** Message for steps less than or equal to ***vPpsSmallStep*** provided that either the voltage on V_{BUS} has reached ***vPpsNew*** or the power supply is in CL mode.

When ***vPpsNew*** is lower than the battery voltage, or the Source's primary power is cut off the Sink ***Shall*** immediately disconnect its battery from V_{BUS}. In these situations, the output current could reverse polarity and the Sink is not allowed to source current (see Sections 7.2.1 and 7.2.9).

Figure 7-4 and Figure 7-5 below show the output voltage behavior of a Programmable Power Supply in response to positive and negative voltage change requests while operating with a PPS. The parameters ***vPpsMinVoltage*** and ***vPpsMaxVoltage*** define the lower and upper limits of the PPS range respectively (see Table 10-8 for required ranges). ***vPpsMinVoltage*** corresponds to Minimum Voltage field in the PPS APDO and ***vPpsMaxVoltage*** corresponds to Maximum Voltage field in the PPS APDO. If the Sink negotiates for a new PPS APDO, then the transition between the two PPS APDOs ***Shall*** occur as described in Section 7.3.18.

Figure 7-4 PPS Positive Voltage Transitions

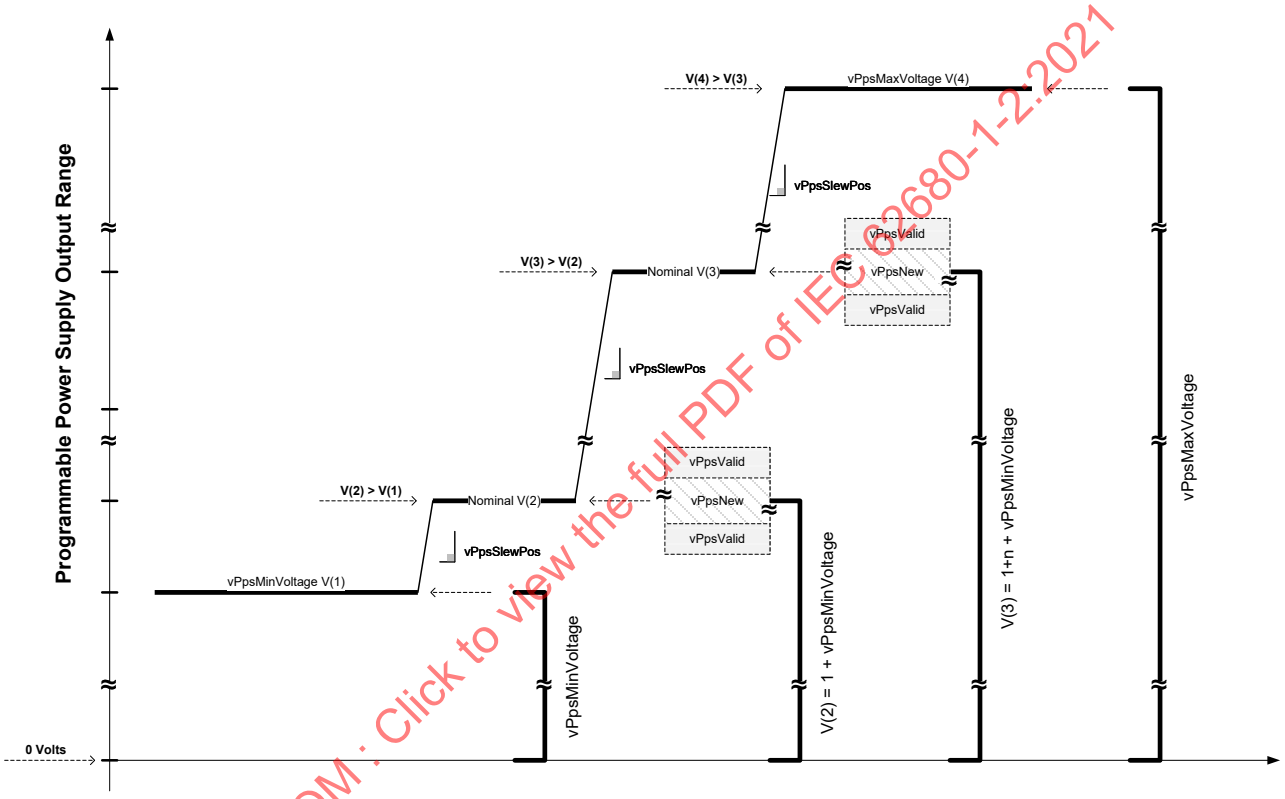
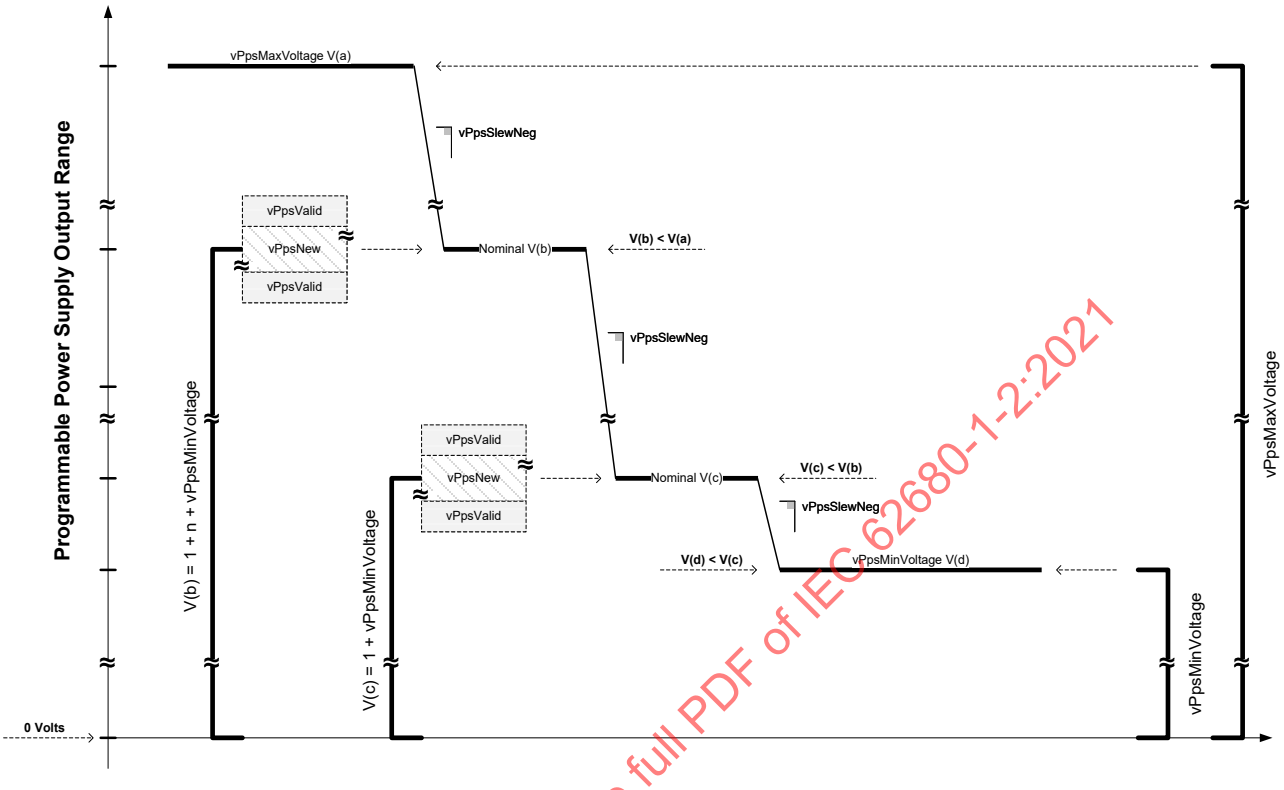
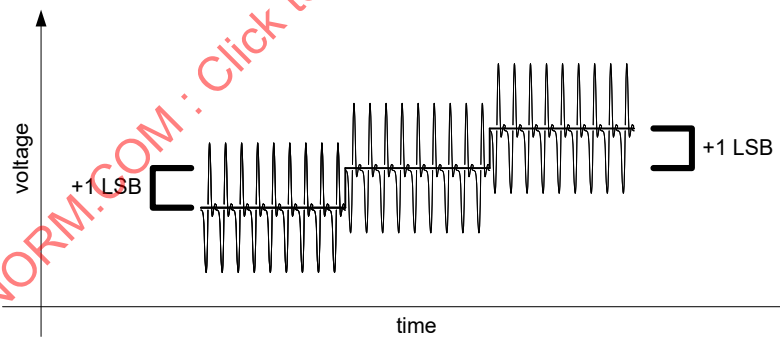


Figure 7-5 PPS Negative Voltage Transitions



The PPS output voltage ripple is expected to exceed the magnitude of one or more LSB as show in the Figure 7-6.

Figure 7-6 Expected PPS Ripple Relative to an LSB



Section 7.1.14 lists transitions that are exempt from the ***vPpsSlewNeg*** and ***vPpsSlewPos*** limits.

7.1.4.4 Programmable Power Supply Current Limit

The Programmable Power Supply ***Shall*** limit its output current to the Operating Current value in the Programmable RDO when the Sink attempts to draw more current than the Output Current level. The programming step size for the Output Current is ***iPpsCLStep***. All programming changes of the Operating Current ***Shall*** settle to the new Operating Current value within ***tPpsCLProgramSettle***. The PPS Operating Current regulation accuracy during Current Limit is defined as ***iPpsCLNew***. The minimum programmable Current Limit level is ***iPpsCLMin***. A Source that supports PPS ***Shall*** support Current Limit programmability between ***iPpsCLMin*** and the Maximum Current value in the PPS APDO.

The response of a PPS to a load change depends on the Operating mode of the PPS and the magnitude of the load change. These dependencies lead to one of four possible responses of a PPS to any load change. They are differentiated by the value of the PPS Status OMF Flag before and after the load change:

- If the PPS Status OMF Flag is cleared both before and after the load change, the PPS responds solely by maintaining the output voltage. The PPS output voltage shall remain within ***vPpsValid*** range. The PPS response to the load change ***Shall*** settle within the ***vPpsNew*** tolerance band by the time ***tPpsTransient***. The Operating Mode Flag ***Shall*** remain cleared during the load change response of the PPS.
- If the PPS Status OMF Flag is cleared before the load change and set after the load change, the PPS responds by reducing its output voltage to limit the PPS output current. The PPS output current ***Shall*** stay within the ***iPpsCVCLTransient*** range once it reaches the ***iPpsCVCLTransient*** range. The PPS response to the load change ***Shall*** settle within the ***iPpsCLNew*** tolerance band by the time ***tPpsCVCLTransient***. The Operating Mode Flag ***Shall*** be set when the PPS load change response settles.
- If the PPS Status OMF Flag is set both before and after the load change, the PPS responds by adjusting its output voltage to maintain the output current. The PPS output current ***Shall*** stay within the ***iPpsCLTransient*** range. The PPS response to the load change ***Shall*** settle within the ***iPpsCLNew*** tolerance band by the time ***tPpsCLSettle***. The Operating Mode Flag ***Shall*** remain set during the load change response of the PPS.
- If the PPS Status OMF Flag is set before the load change and cleared after the load change, the PPS responds to the load change by increasing its output voltage to ***vPpsNew*** and then maintaining it. The PPS output voltage ***Shall*** stay within the ***tPpsCLCVTransient*** range. The PPS response to the load change ***Shall*** settle within the ***vPpsNew*** tolerance band by the time ***tPpsCLCVTransient***. The Operating Mode Flag ***Shall*** be cleared when the PPS load change response settles.

The PPS Shall maintain its output voltage at the value requested in the PPS RDO for all static and dynamic load conditions except when in Current Limit operation. In response to any static or dynamic load condition during Current Limit operation that causes the PPS output voltage to drop below ***vPpsShutdown*** the Source ***May*** send ***Hard Reset*** Signaling and ***Shall*** discharge V_{BUS} to ***vSafe0V*** then resume default operation at ***vSafe5V***.

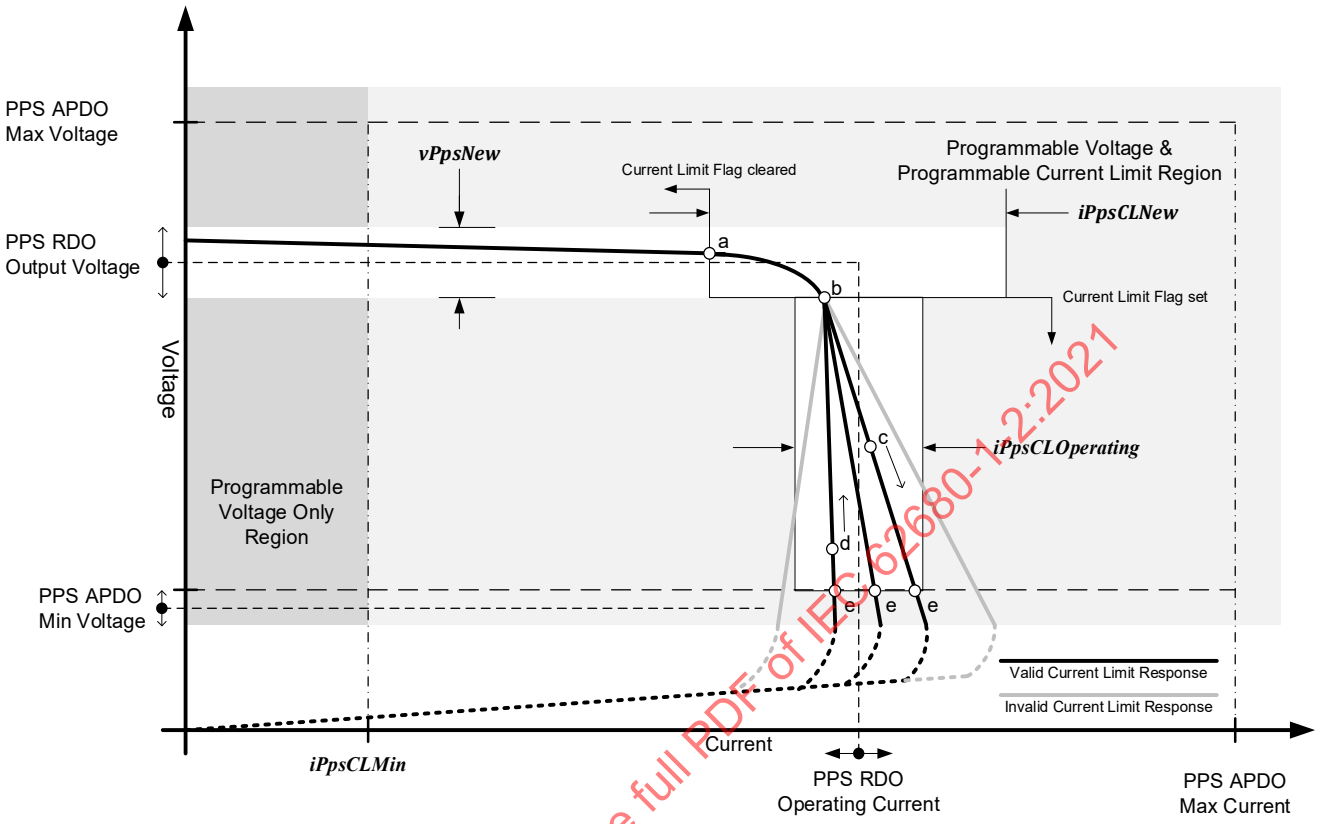
When the Sink attempts to draw more current than the Operating Current in the RDO, the Source ***Shall*** limit its output current. The current available from the Source during Current Limit mode shall meet ***iPpsCLNew*** plus ***iPpsCLOperating***. The Sink ***May Not*** reduce its Operating Current request in the RDO when the PPS Status OM Flag is set.

Current limiting ***Shall*** be performed by the PPS Source. Sinks that rely on PPS Current Limiting ***Shall*** meet the requirements of Section 7.2.9. The Source ***Shall Not*** shutdown or otherwise disrupt the available output power while in Current Limit mode unless another protection mechanism as outlined in Section 7.1.7 is engaged to protect the Source from damage.

The relationship between PPS programmable output voltage and PPS programmable Current Limit ***Shall*** be as shown in Figure 7-7. The transition between the Constant Voltage mode and the Current Limit mode occurs between points *a* and *b*. The PPS Status OM Flag shall be set or cleared within this region. In Current Limit mode when the load resistance changes the output current of the Source stay within ***iPpsCLOperating***, which is determined by point b (a measured value). As the load resistance decreases the output current should stay the same or increase slightly and as the load resistance increases the output current should stay the same or decrease slightly. The amount of allowable increase and decrease Shall Not exceed ***iPpsCLTolerance*** relative to a straight line drawn between points b and e as illustrated in Figure 7-8.

The proper behavior is represented by point *c*. Likewise, as the load resistance increases, the output current of the Source ***Shall Not*** increase. The proper behavior is represented by point *d*.

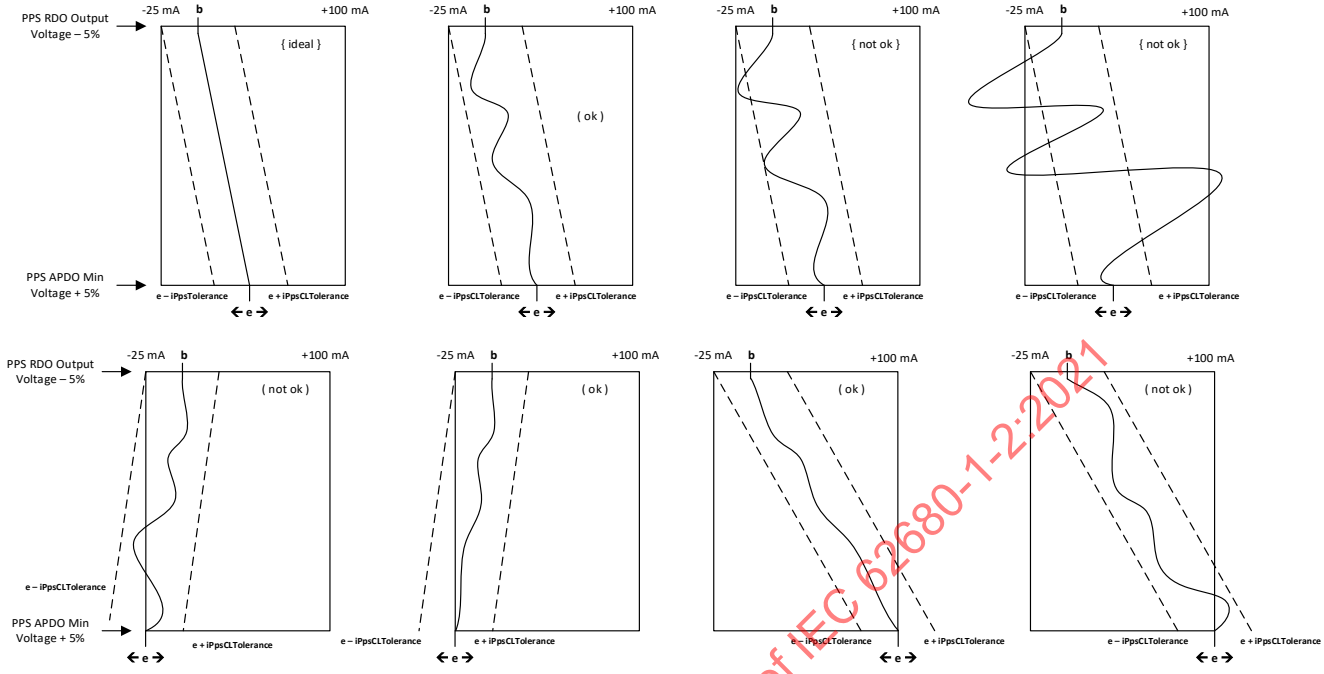
Figure 7-7 PPS Programmable Voltage and Current Limit



Notes:

- Point *a* represents entry into the transition region between Constant Voltage mode and Current Limit mode.
- Point *b* represents exit from the transition region between Constant Voltage mode and Current Limit mode.
- Point *b* is where the allowable increase in current up to ***iPpsCLOperating*** begins.
- Point *c* represents the behavior as the load resistance decreases during Current Limit mode. See Table 7-22 for the allowed change in Operating Current (***iPpsCLOperating***) during this behavior.
- Point *d* represents the behavior as the load resistance increases during Current Limit mode. See Table 7-22 for the allowed change in Operating Current (***iPpsCLOperating***) during this behavior.
- Point e represents the exit from the ***iPpsCLOperating*** region.

Figure 7-8 iPpsCLOperatingDetail

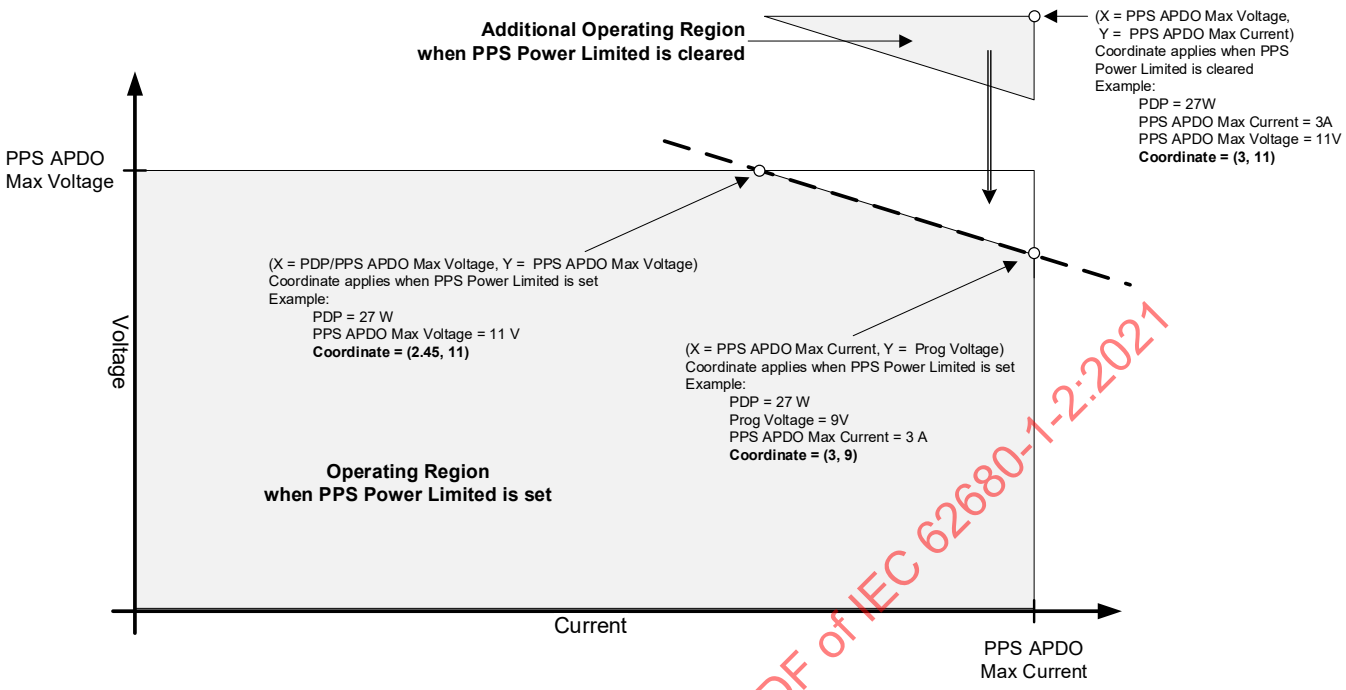


7.1.4.4.1 Constant Power Mode

In Constant Power mode (when the PPS Power Limited bit is set) the Source ***Shall*** limit its output current so that the product of the output current times the output voltage does not exceed the Source's PDP Rating. Sinks ***May Not*** limit their Operating Current request in the RDO but ***Shall*** meet the requirements of Section 7.2.9.

The relationship between PPS programmable output voltage and PPS programmable Current Limit in the Constant Power mode Shall be as shown in Figure 7-9.

Figure 7-9 PPS Programmable Voltage and Current Limit



7.1.5 Response to Hard Resets

Hard Reset Signaling indicates a communication failure has occurred and the Source ***Shall*** stop driving VCONN, ***Shall*** remove Rp from the VCONN pin and ***Shall*** drive V_{BUS} to ***vSafe0V*** as shown in Figure 7-10. 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 VCONN and restoring V_{BUS} to ***vSafe5V***. A Source ***Shall*** conform to the VCONN timing as specified in ***[USB Type-C 2.0]***.

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

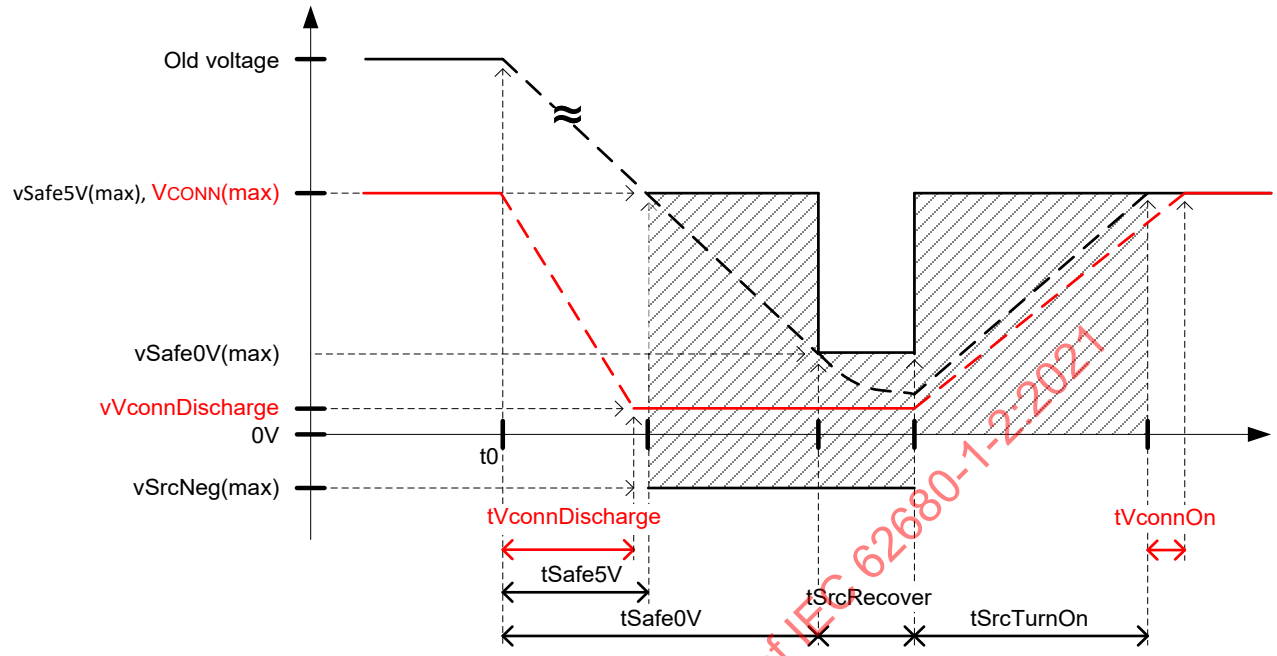
- Self-powered devices ***Should Not*** disconnect from USB during a Hard Reset (see Section 9.1.2).
- Self-powered devices operating at more than ***vSafe5V*** ***May Not*** maintain full functionality after a ***Hard Reset***.
- Bus powered devices will disconnect from USB during a Hard Reset due to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

When a Hard Reset occurs the Source ***Shall*** stop driving VCONN, ***Shall*** remove Rp from the VCONN 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 ***tSafe5V*** and ***tSafe0V*** relative to the start of the voltage transition as shown in Figure 7-10.

Figure 7-10 Source V_{BUS} and VCONN Response to Hard Reset



VCONN will meet ***tVconnDischarge*** relative to the start of the voltage transition as shown in Figure 7-10 due to the discharge circuitry in the Cable Plug. VCONN ***Shall*** meet ***tVconnOn*** relative to V_{BUS} reaching ***vSafe5V***. Note ***tVconnOn*** and ***tVconnDischarge*** are defined in ***[USB Type-C 2.0]***.

7.1.6 Changing the Output Power Capability

Some USB Power Delivery negotiations will require the Source to adjust its output power capability without changing the output voltage. In this case the Source ***Shall*** be able to supply a higher or lower load current within ***tSrcReady***.

7.1.7 Robust Source Operation

7.1.7.1 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 engages followed by an ***Alert*** Message indicating an OCP event once an Explicit Contract has been established. 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***.

7.1.7.2 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 engages followed by an ***Alert*** Message indicating an OTP event once an Explicit Contract has been established. 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***.

7.1.7.3 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.

7.1.7.4 Detach

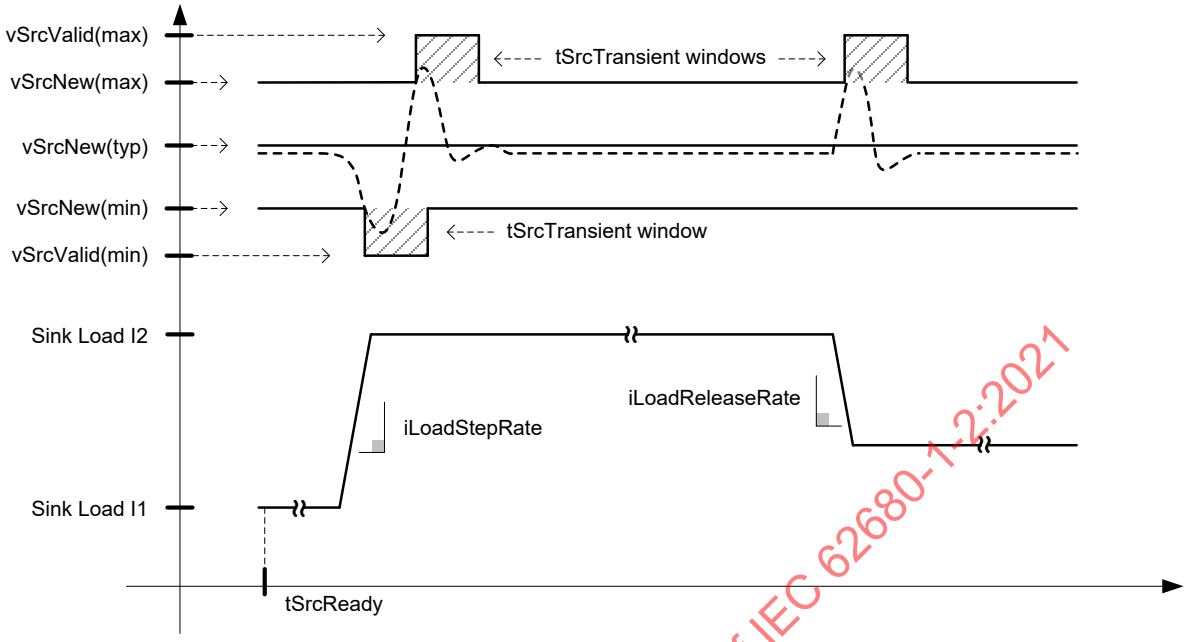
A USB Detach is detected electrically using CC detection on the USB Type-C® connector. When the Source is Detached the Source ***Shall*** transition to ***vSafe0V*** by ***tSafe0V*** relative to when the Detach event occurred. During the transition to ***vSafe0V*** the V_{BUS} voltage ***Shall*** be below ***vSafe5V*** max by ***tSafe5V*** relative to when the Detach event occurred and ***Shall Not*** exceed ***vSafe5V*** max after this time.

7.1.8 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 7-11 illustrates the application of ***vSrcNew*** and ***vSrcValid*** after the voltage transition is complete.

The ***vSrcNew*** and ***vSrcValid*** limits ***Shall Not*** apply to V_{BUS} during the V_{BUS} discharge and switchover that occurs during a Fast Role Swap as described in Section 7.1.13.

Figure 7-11 Application of vSrcNew and vSrcValid limits after tSrcReady



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.

7.1.8.1 Programmable Power Supply Output Voltage Tolerance and Range

After a voltage transition of a Programmable Power Supply is complete (i.e. after ***tPpsSrcTransSmall*** or ***tPpsSrcTransLarge***) and during static load conditions the Source output voltage ***Shall*** remain within the ***vPpsNew*** limits. The range defined by ***vPpsNew*** accounts for DC regulation accuracy, line regulation, load regulation and output ripple. After a voltage transition is complete (i.e. after ***tPpsSrcTransSmall*** or ***tPpsSrcTransLarge***) and during transient load conditions the Source output voltage ***Shall Not*** go beyond the range specified by ***vPpsValid***. The amount of time the Source output voltage can be in the band between ***vPpsNew*** and ***vPpsValid*** ***Shall Not*** exceed ***tPpsTransient***.

7.1.9 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 ***tSrcReady***.

7.1.10 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 ***vSafe0V*** 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 ***vSafe0V***.
- Any discharge circuitry that was used to achieve ***vSafe0V*** ***Shall*** be removed from V_{BUS}.
- The Dual-Role Power Port ***Shall*** be configured as a Sink.

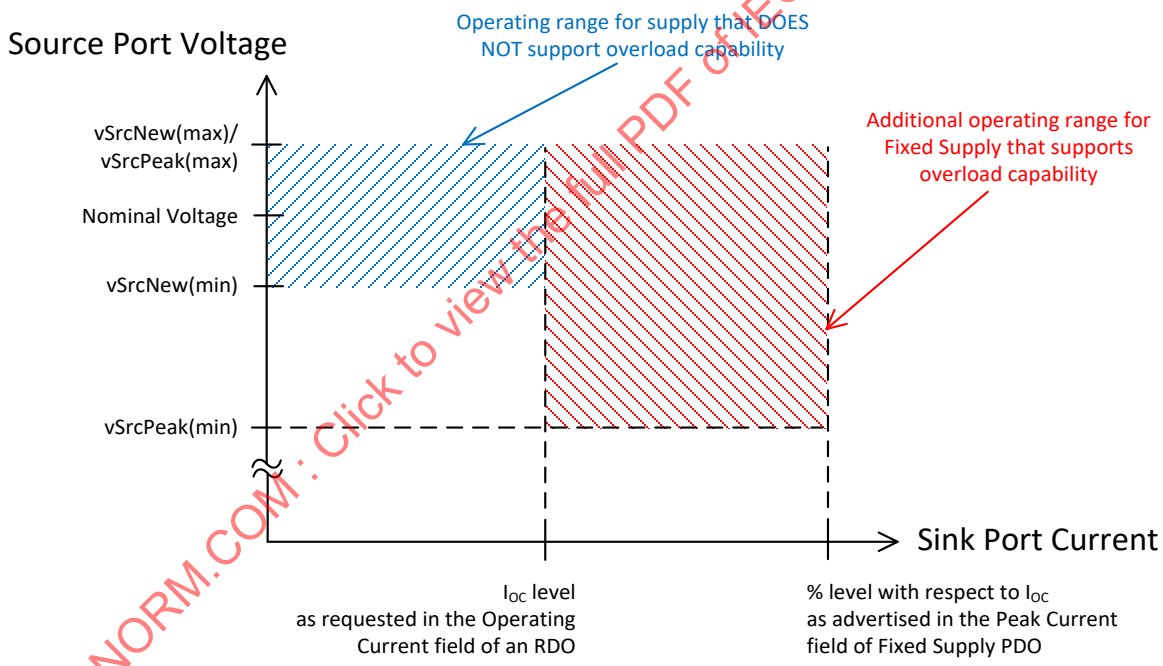
- The USB connection ***Shall Not*** reset even though ***vSafe5V*** is no longer present on V_{BUS} (see Section 9.1.2).

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 ***tSrcSwapStdby***. 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 ***tNewSnk***. The new Sink ***May*** start using power after the new Source sends the ***PS_RDY*** Message.

7.1.11 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 6-10. The overload conditions are bound in magnitude, duration and duty cycle as listed in Table 6-10. 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 7-12). 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 7-12 Source Peak Current Overload



7.1.12 Source Capabilities Extended Parameters

Implementers can choose to make available certain characteristics of a USB PD Source as a set of static and/or dynamic parameters to improve interoperability between external power sources and portable computing devices. The complete list of reportable static parameters are described in full in Section 6.5.1 and listed in Figure 6-32. The subset of parameters listed below directly represent Source capabilities and are described in the rest of this section.

- Voltage Regulation.
- Holdup Time.
- Compliance.
- Peak Current.
- Source Inputs.
- Batteries.

7.1.12.1 Voltage Regulation Field

The power consumption of a device can change dynamically. The ability of the Source to regulate its voltage output might be important if the device is sensitive to fluctuations in voltage. The Voltage Regulation bit field is used to convey information about the Sources output regulation and tolerance to various load steps.

7.1.12.1.1 Load Step Slew Rate

The default load step slew rate is established at 150mA/µs. A Source ***Shall*** meet the following requirements under the load step reported in the Extended Source Capabilities:

- The Source ***Shall*** maintain V_{BUS} regulation within the ***vSrcValid*** range.
- The noise on the CC line ***Shall*** remain below ***vNoiseIdle*** and ***vNoiseActive.***

Test conditions require a change in both positive and negative load steps from 1Hz to 5000Hz, up to the advertised Load Step Magnitude of the full load output including from both 10 mA and 10% initial load. The Source ***Shall*** ensure that PD Communications meet the transmit and receive masks as specified in Section 5.8.2 under all load conditions.

7.1.12.1.2 Load Step Magnitude

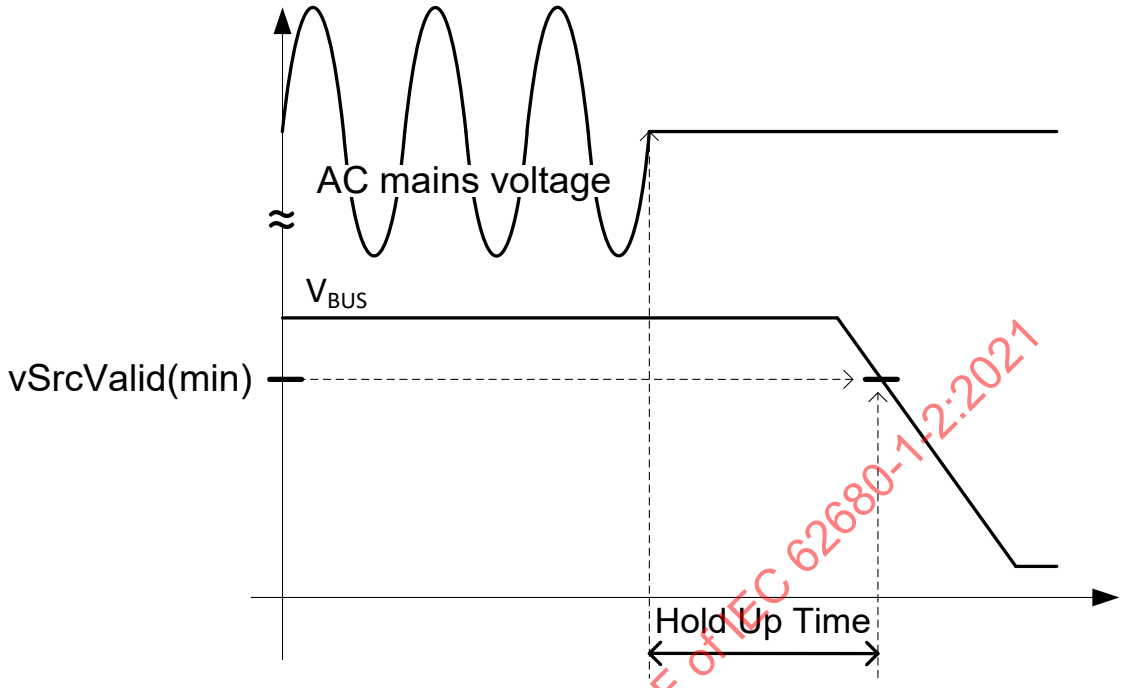
The default load step magnitude rate ***Shall*** be 25% of IoC. The Source ***May*** report higher capability tolerating a load step of 90% of IoC.

7.1.12.2 Holdup Time Field

The Holdup Time field ***Shall*** return a numeric value of the number of milliseconds the output voltage stays in regulation upon a short interruption of AC mains.

A mains supplied Source ***Shall*** report its holdup time in this field. The holdup time is measured with the load at rated maximum, with AC mains at 115VAC rms and 60Hz (or at 230VAC rms and 50Hz for a Source that does not support 115VAC mains). The reported time describes the minimum length of time from the last completed AC mains input cycle (zero-degree phase angle) until when the output voltage decays below ***vSrcValid*** (min). Power sources are recommended to support a minimum of 3ms and are preferred to support over 10 milliseconds holdup time (equivalent to a half cycle drop from the AC Mains).

Figure 7-13 Holdup Time Measurement



7.1.12.3 Compliance Field

A Source claiming LPS, PS1 or PS2 compliance (see *[IEC 62368-1]*) ***Shall*** report its capabilities in the Compliance field. Since the Source ***May*** have several potential output voltage and current settings, every Source supply (indicated by a PDO) ***Shall*** be compliant to LPS requirements.

Note: according to the requirements of *[IEC 60950-1]*, a device tested and certified with an LPS Source is prohibited to use a non-LPS Source. Alternatively, *[IEC 62368-1]*, classifies power sources according to their maximum, constrained power output (15watts or 100watts).

7.1.12.4 Peak Current

The Source reports its ability to source peak current delivery in excess of the negotiated amount in the Peak Current field. The duration of peak current ***Shall*** be followed by a current consumption below the Operating Current (IoC) in order to maintain average power delivery below the IoC current.

A Source ***May*** have greater capability to source peak current than can be reported using the Peak Current field in the Fixed Supply PDO. In this case the Source ***Shall*** report its additional capability in the Peak Current field in the ***Source_Capabilities_Extended*** Message.

Each overload period ***Shall*** be followed by a period of reduced current draw such that the rolling average current over the Overload Period field value with the specified Duty Cycle field value (see Section 6.5.1.10) ***Shall Not*** exceed the negotiated current. This is calculated as:

*Period of reduced current = (1 – value in Duty Cycle field/100) * value in Overload Period field*

7.1.12.5 Source Inputs

The Source Inputs field identifies the possible inputs that provide power to the Source. Note some Sources are only powered by a Battery (e.g., an automobile) rather than the more common mains.

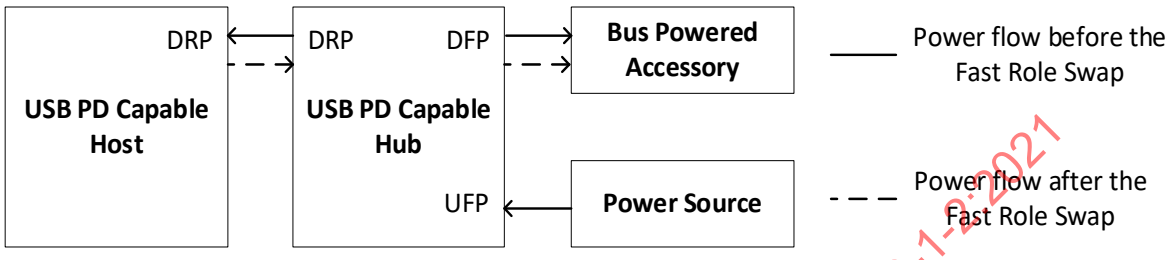
7.1.12.6 Batteries

The Batteries field ***Shall*** report the number of Batteries the Source supports. The Source ***Shall*** independently report the number of Hot Swappable Batteries and the number of Fixed batteries.

7.1.13 Fast Role Swap

A Fast Role Swap limits the interruption of V_{BUS} power to a bus powered accessory connected to a Hub DFP that has a UFP attached to a power source and a DRP attached to a Host port supporting DRP as shown in Figure 7-14 V_{BUS} Power during Fast Role Swap.

Figure 7-14 V_{BUS} Power during Fast Role Swap



When the power source connected to the Hub UFP stops sourcing power and V_{BUS} at the Hub DRP connector discharges below ***vSrcValid***(min), if V_{BUS} has been negotiated to a higher voltage than***vSafe5V***, or ***vSafe5V*** (min) the Fast Role Swap signal ***Shall*** be sent from the Hub DRP to the Host DRP and the Hub DRP ***Shall*** sink power. In the Fast Role Swap use case, the Hub DRP behaves like a bidirectional power path. The Hub DRP ***Shall Not*** enable V_{BUS} discharge circuitry when changing operation from initial Source to new Sink. The Hub DFP Port(s) ***Shall*** support default USB Type-C Current (see ***[USB Type-C 2.0]***) until a new Explicit Contract is negotiated.

After sending the FRS signal and while V_{BUS} > ***vSafe5V*** (min), the new Sink ***Shall Not*** draw more than ***iNewFrsSink*** until the new Source has applied its Rp. The new Sink ***Shall Not*** draw more than ***pSnkStdby*** from V_{BUS} until ***tSnkFRSwap*** after it has started sending the FRS signal or V_{BUS} has fallen below ***vSafe5V*** (min). The ***tSnkFRSwap*** time ***Shall*** start at the beginning of the FRS signal or when V_{BUS} falls below ***vSafe5V*** (min), whichever comes later. After waiting for ***tSnkFRSwap***, the new Sink ***Shall Not*** draw more than ***iNewFrsSink*** until the new Source has applied its Rp. After the new Source has applied its Rp, the new Sink ***Shall*** be limited to USB Type-C Current (see ***[USB Type-C 2.0]***) in an Implicit Contract until a new Explicit Contract is negotiated. All Sink requirements ***Shall*** apply to the new Sink after the Fast Role Swap is complete. The Fast Role Swap response of the Host DRP is described in Section 7.2.10 since the Host DRP is operating as the initial Sink prior to the Fast Role Swap.

After the V_{BUS} voltage level at the Hub DRP connector drops below ***vSafe5V*** a ***PS_RDY*** Message ***Shall*** be sent to the Host DRP as shown in the Fast Role Swap transition diagram of Section 7.3.15.

Figure 7-15 shows the V_{BUS} detection and timing for the new Source during a Fast Role Swap after the Fast Role Swap signal has been received. The new Source ***May*** turn on the V_{BUS} output switch once V_{BUS} is below ***vSafe5V*** (max). In this case, the new Source prevents V_{BUS} from falling below ***vSafe5V*** (min). The new source ***Shall*** turn on the V_{BUS} output switch within ***tSrcFRSwap*** of falling below ***vSafe5V*** (min).

V_{BUS} might have started at ***vSafe5V*** or at higher voltage. When the Fast Role Swap Signal is detected, V_{BUS} could therefore be either above ***vSafe5V*** (max), within the ***vSafe5V*** range, or below ***vSafe5V*** (min). If the Fast Role Swap Signal is detected when V_{BUS} is below ***vSafe5V*** (min), then the new source ***Shall*** turn on the V_{BUS} output switch within ***tSrcFRSwap*** of detecting the Fast Role Swap Signal. In this case, the maximum time from the beginning of the Fast Role Swap signal to V_{BUS} being sourced ***May*** be ***tSrcFRSwap*** (max) + ***tFRSwapRx*** (max).

Figure 7-15 V_{BUS} detection and timing during Fast Role Swap, initial V_{BUS} (at new source) > vSafe5V (min).

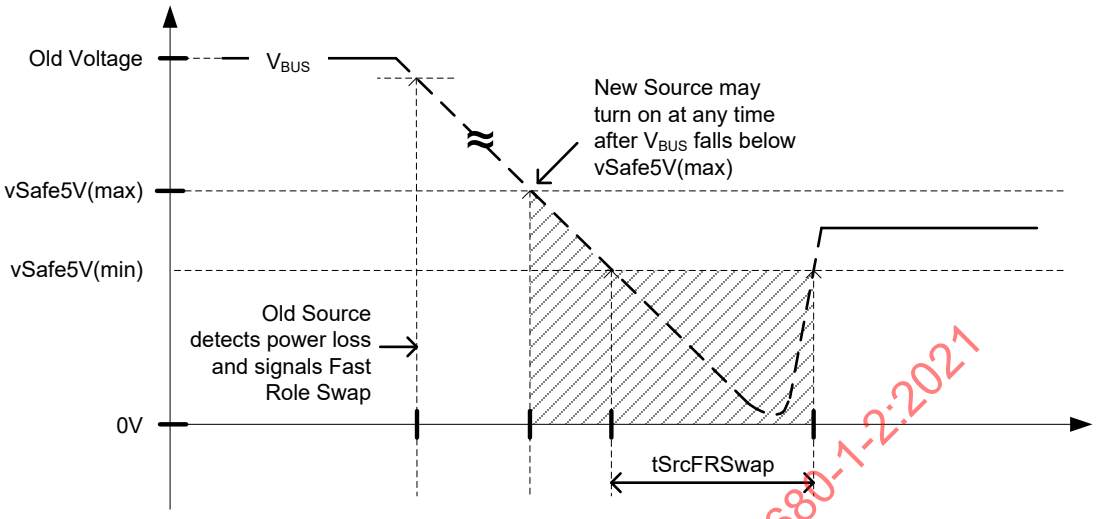
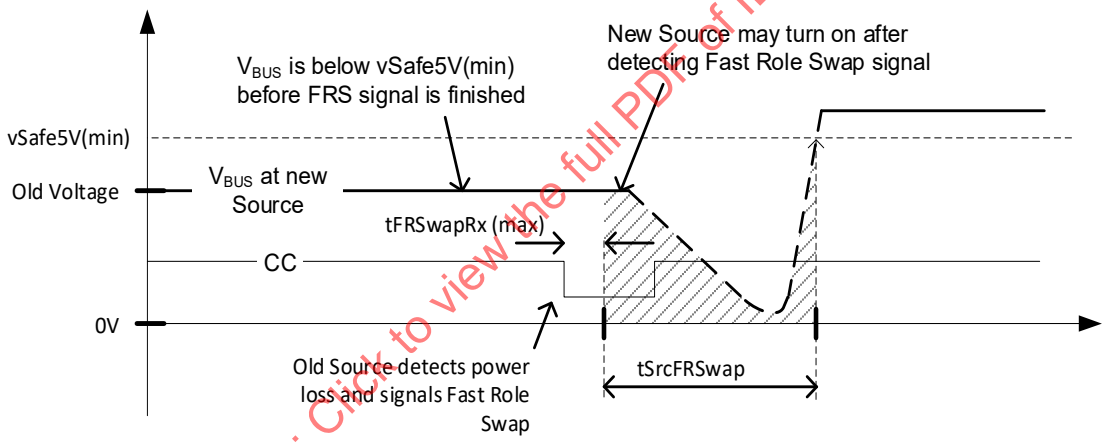


Figure 7-16 V_{BUS} detection and timing during Fast Role Swap, initial V_{BUS} (at new source) < vSafe5V (min).



7.1.14 Non-application of V_{BUS} Slew Rate Limits

Scenarios where ***vSrcSlewPos*** and ***vPpsSlewPos*** V_{BUS} slew rate limits do not apply and V_{BUS} ***May*** transition faster than specified are as follows:

- When first applying V_{BUS} after an Attach.
- When increasing V_{BUS} from ***vSafe0V*** to ***vSafe5V*** during a Hard Reset.
- During a Fast Role Swap when the initial Sink applies V_{BUS}.

Scenarios where ***vSrcSlewNeg*** and ***vPpsSlewNeg*** V_{BUS} slew rate limits do not apply and V_{BUS} ***May*** transition faster than specified are as follows:

- When discharging V_{BUS} to ***vSafe0V*** during a Hard Reset.
- When discharging V_{BUS} to ***vSafe0V*** after a Detach.
- During a Fast Role Swap when the V_{BUS} power source connected to the Hub UFP stops sourcing power.

7.1.15 VCONN Power Cycle

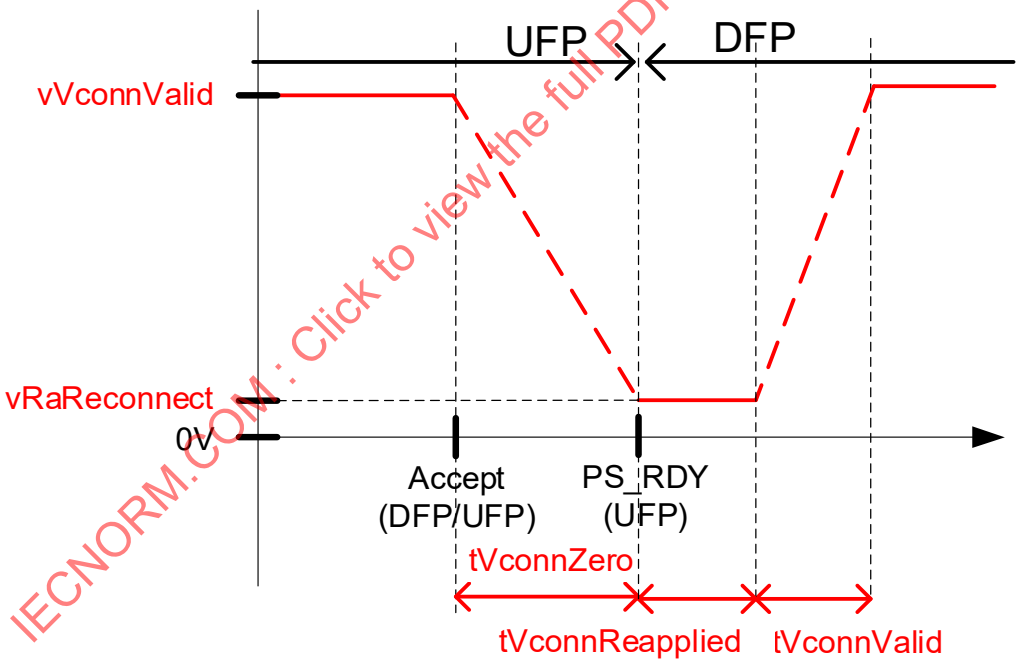
7.1.15.1 UFP VCONN Power Cycle

The Data Reset process requires the DFP to be the VCONN source by the end of the process. In the case where the UFP is the VCONN source, the following steps ***Shall*** be followed:

1. Following the last bit of the ***GoodCRC*** acknowledging the ***Accept*** Message in response to the ***Data_Reset*** Message, the UFP Shall turn off VCONN and ensure it is below vRaReconnect (see ***[USB Type-C 2.0]***) within ***tVconnZero***.
2. When VCONN is below vRaReconnect, the UFP ***Shall*** send a ***PS_RDY*** Message. Note if the UFP was not sourcing VCONN, it still sends the ***PS_RDY*** Message.
3. The DFP Shall wait ***tVconnReapplied*** following the last bit of the ***GoodCRC*** acknowledging the ***PS_RDY*** Message before sourcing VCONN. The DFP Shall ensure VCONN is within vVconnValid (see ***[USB Type-C 2.0]***) within ***tVconnValid***.

Figure 7-17 below illustrates the UFP VCONN Power Cycle process.

Figure 7-17 Data Reset UFP VCONN Power Cycle



7.1.15.2 DFP VCONN Power Cycle

The Data Reset process requires the DFP to be the VCONN source by the end of the process. In the case where the DFP is the VCONN source, the following steps ***Shall*** be followed:

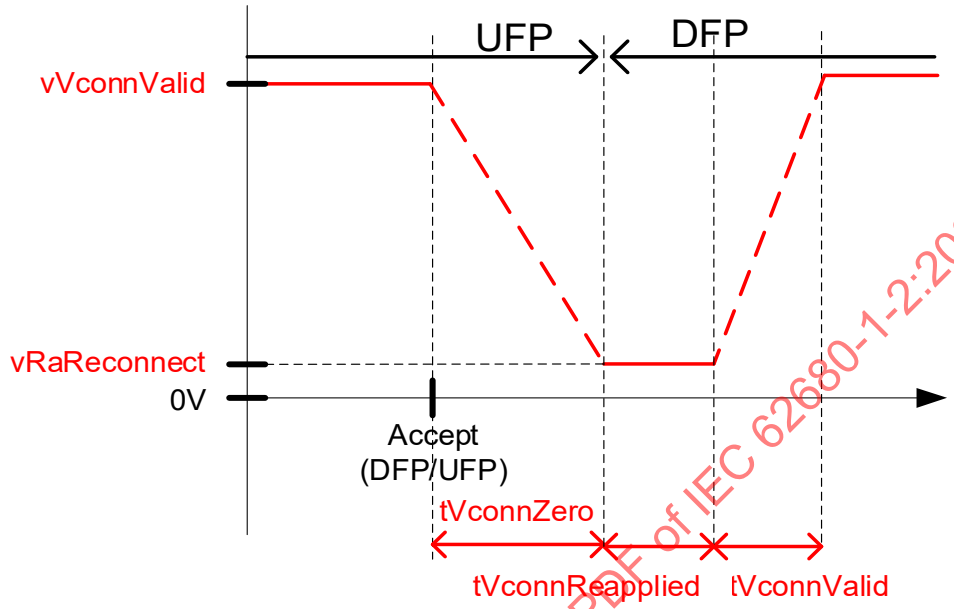
1. If the DFP sent the ***Data_Reset*** Message and is sourcing VCONN then it ***Shall*** turn off VCONN and ensure it is below vRaReconnect (see ***[USB Type-C 2.0]***) within ***tVconnZero*** of the last bit of the ***GoodCRC*** acknowledging the ***Accept*** message in response to the ***Data_Reset*** Message.

2. If the UFP sent the ***Data_Reset*** Message then the DFP Shall turn off VCONN and ensure it is below vRaReconnect (see ***[USB Type-C 2.0]***) within ***tVconnZero*** following the last bit of the ***GoodCRC*** acknowledging the ***Accept*** Message in response to the ***Data_Reset*** Message.

3. When VCONN is below vRaReconnect, the DFP Shall wait ***tVconnReapplied*** before sourcing VCONN.
4. The DFP Shall ensure VCONN is within vVconnValid (see ***[USB Type-C 2.0]***) within ***tVconnValid***.

Figure 7-18 below illustrates the DFP VCONN Power Cycle process.

Figure 7-18 Data Reset DFP VCONN Power Cycle



7.2 Sink Requirements

7.2.1 Behavioral Aspects

A USB PD Sink exhibits the following behaviors:

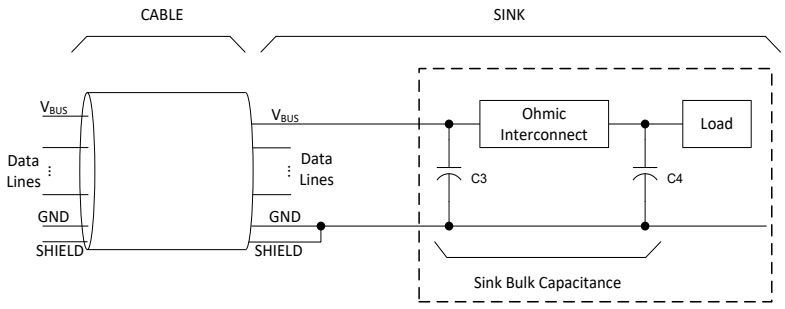
- ***Shall*** not draw more than the default ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** or ***[USBBC 1.2]*** VBUS current when a Contract does not exist (USB Default Operation).
- ***Shall*** follow the requirements as specified in Section 7.1.5 when ***Hard Reset*** Signaling is received.
- ***Shall*** control VBUS in-rush current when increasing current consumption.

7.2.2 Sink Bulk Capacitance

The Sink bulk capacitance consists of C3 and C4 as shown in Figure 7-19. 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 VBUS 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 VBUS 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 7-19 Placement of Sink Bulk Capacitance



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.

7.2.3.1 Programmable Power Supply Sink Standby

A Sink is not required to transition to Sink Standby when operating within the negotiated PPS APDO. A Sink ***May*** consume the Operating Current value in the Programmable RDO during PPS output voltage changes. However, prior to operating the PPS in Current Limit, the Sink ***Shall*** program the PPS Operating Voltage to the lowest practical level that satisfies the Sink load requirement. Doing so will minimize the inrush current that occurs when the transition to Current Limit occurs. When operating with a PPS that is in Current Limit, the Sink ***Shall Not*** change its load in a manner that exceeds ***iPpsCLLoadStepRate*** or ***iPpsCLLoadReleaseRate***. The load change magnitude ***Shall Not*** exceed ***iPpsCLLoadStep*** or ***iPpsCLLoadRelease***.

If the Sink negotiates for a new PPS APDO, then the Sink ***Shall*** transition to Sink Standby while changing between PPS APDOs as described in Section 7.3.18.

7.2.4 Suspend Power Consumption

When Source has set its USB Suspend Supported flag (see Section 6.4.1.2.2.2), 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.

The Sink's operating current ***Shall Not*** change faster than the value reported in the Source's Load Step Slew Rate field and ***Shall*** ensure that PD Communications meet the transmit and receive masks as specified in Section 5.8.2.

7.2.7 Swap Standby for Sinks

The Sink capability in a Dual-Role Power 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 Power 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 6.4.1.2.2.7). 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 Robust Sink Operation

7.2.9.1 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 = 21.55V.
- Maximum bulk capacitance = 30µF.
- 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 21.55V 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.

7.2.9.2 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 followed by an ***Alert*** Message indicating an OVP event once an Explicit Contract has been established. 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.

7.2.9.3 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 followed by an ***Alert*** Message indicating an OTP event once an Explicit Contract has been established. 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.

7.2.9.4 Over Current Protection

Sinks that operate with a Programmable Power Supply ***Shall*** implement their own internal current protection mechanism to protect against internal V_{BUS} current faults as well as erratic Source current regulation. The Sink ***Shall*** never draw higher current than the Maximum Current value in the PPS APDO.

7.2.10 Fast Role Swap

As described in Section 7.1.13 a Fast Role Swap limits the interruption of V_{BUS} power to a bus powered accessory connected to a Hub DFP that has a UFP attached to a power source and a DRP attached to a Host port that supports DRP. This configuration is shown in Figure 7-14 V_{BUS} Power during Fast Role Swap.

The Host DRP, upon establishing an explicit contract, ***Shall*** query the initial Source's Sink Capabilities to determine whether the initial Source supports Fast Role Swap, and what level of current it requires. If the ***Sink_Capabilities*** Message received from the initial Source has at least one of the Fast Role Swap bits set, and the Host DRP is able to source the requested current at 5V, the Host DRP May arm itself for Fast Role

Swap. If the Host DRP has not queried the Sink Capabilities from the initial Source, or if the ***Sink_Capabilities*** Message reports no Fast Role Swap support or a current that is beyond what the Host DRP is able or willing to source in the event of a Fast Role Swap, the Host DRP ***Shall Not*** arm itself for Fast Role Swap and ***Shall Ignore*** any Fast Role Swap signals that may be detected.

When the Host DRP that supports Fast Role Swap detects the Fast Role Swap signal, the Host DRP ***Shall*** stop sinking current and ***Shall*** be ready and able to source ***vSafe5V*** if the residual V_{BUS} voltage level at the Host DRP connector is greater than ***vSafe5V***. When the residual V_{BUS} voltage level at the Host DRP connector discharges below ***vSafe5V***(min) the Host DRP as the new Source ***Shall*** supply ***vSafe5V*** to the Hub DRP within ***tSrcFRSwap***. The Host DRP ***Shall Not*** enable V_{BUS} discharge circuitry when changing roles from initial Sink to new Source.

The new Source ***Shall*** supply ***vSafe5V*** at USB Type-C Current (see ***[USB Type-C 2.0]***) at the value advertised in the Fast Role Swap USB Type-C Current field (see Section 6.4.1.3.1.6). All Source requirements ***Shall*** apply to the new Source after the Fast Role Swap is complete The Fast Role Swap response of the Hub DRP is described in Section 7.1.13 since the Hub DRP is operating as the initial Source prior to the Fast Role Swap.

After the Host DRP is providing V_{BUS} power to the Hub DRP, a ***PS_RDY*** Message ***Shall*** be sent to the Hub DRP as defined by the Fast Role Swap signaling and messaging sequence detailed in Section 7.3.15.

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
- Goto Minimum Current Transition
- Response to ***Hard Reset*** Signaling
  - Source issues ***Hard Reset*** Signaling
  - Sink issues ***Hard Reset*** Signaling
- No change in Current or Voltage.

The transition from ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** or ***[USBBC 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 3.2]***, ***[USB Type-C 2.0]*** or ***[USBBC 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 7-20. The sequence that ***Shall*** be followed is described in Table 7-1. 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 7-20 Transition Diagram for Increasing the Current

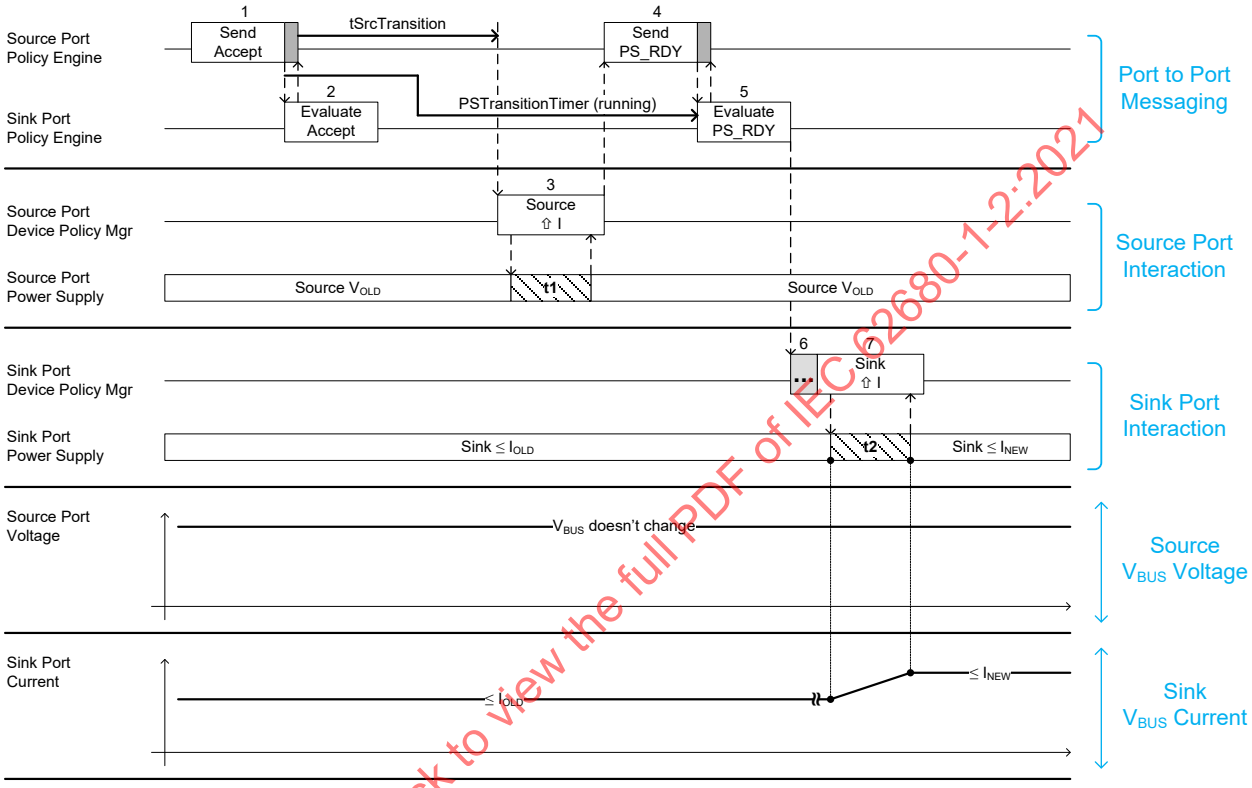


Table 7-1 Sequence Description for Increasing the Current

Step	Source Port	Sink Port
1	Policy Engine sends the ***Accept*** Message to the Sink.	Policy Engine receives the ***Accept*** Message and starts the ***PSTransitionTimer***.
2	Protocol Layer receives the ***GoodCRC*** 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 ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***Accept*** Message.
3	***tSrcTransition*** after the ***GoodCRC*** Message was 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 ***tSrcReady*** (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 ***PS_RDY*** Message to the Sink.	The Policy Engine receives the ***PS_RDY*** Message from the Source.
5	Protocol Layer receives the ***GoodCRC*** Message from the Sink.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***PS_RDY*** 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 ***PS_RDY*** 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 7-21. The sequence that ***Shall*** be followed is described in Table 7-2. 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 7-21 Transition Diagram for Increasing the Voltage

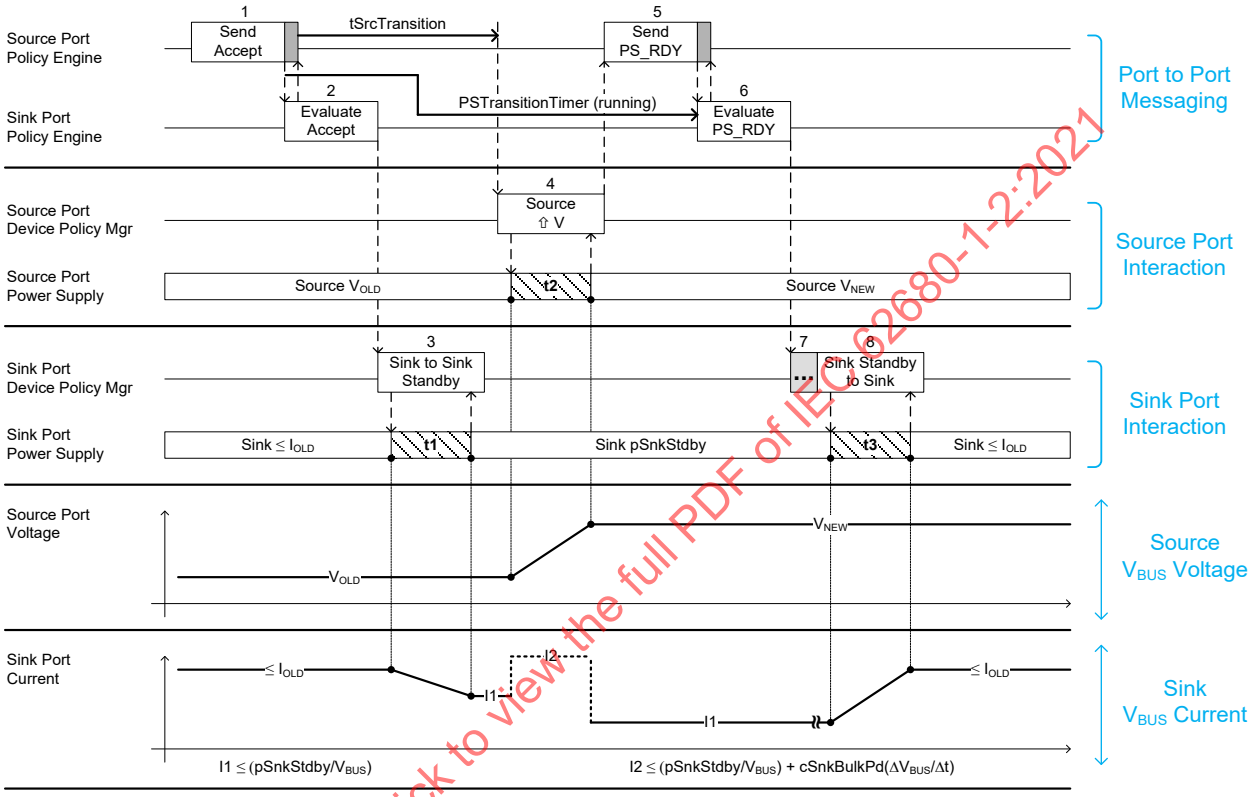


Table 7-2 Sequence Description for Increasing the Voltage

Step	Source Port	Sink Port
1	Policy Engine sends the ***Accept*** Message to the Sink.	Policy Engine receives the ***Accept*** Message and starts the ***PSTransitionTimer***.
2	Protocol Layer receives the ***GoodCRC*** 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 ***GoodCRC*** Message to the Source. Policy Engine. Policy Engine then evaluates the ***Accept*** Message.
3		Policy Engine tells the Device Policy Manager to instruct the power supply to reduce power consumption to ***pSnkStdby*** within ***tSnkStdby*** (t1); t1 ***Shall*** complete before ***tSrcTransition***. 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	***tSrcTransition*** after the ***GoodCRC*** Message was 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 ***tSrcReady*** (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 ***PS_RDY*** Message to the Sink.	The Policy Engine receives the ***PS_RDY*** Message from the Source.
6	Protocol Layer receives the ***GoodCRC*** Message from the Sink.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***PS_RDY*** Message from the Source and tells the Device Policy Manager it is okay to operate at the new power level.
7		The Sink ***May*** begin operating at the new power level any time after evaluation of the ***PS_RDY*** Message. This time duration is indeterminate.
8		The Sink ***Shall Not*** violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. The time duration (t3) depends on the magnitude of the load change.

7.3.3 Increasing the Voltage and Current

The interaction of the System Policy, Device Policy, and power supply that ***Shall*** be followed when increasing the voltage and current is shown in Figure 7-22. 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, Table 7-23 and Table 7-24. Note in this figure, the Sink has previously sent a ***Request*** Message to the Source.

Figure 7-22 Transition Diagram for Increasing the Voltage and Current

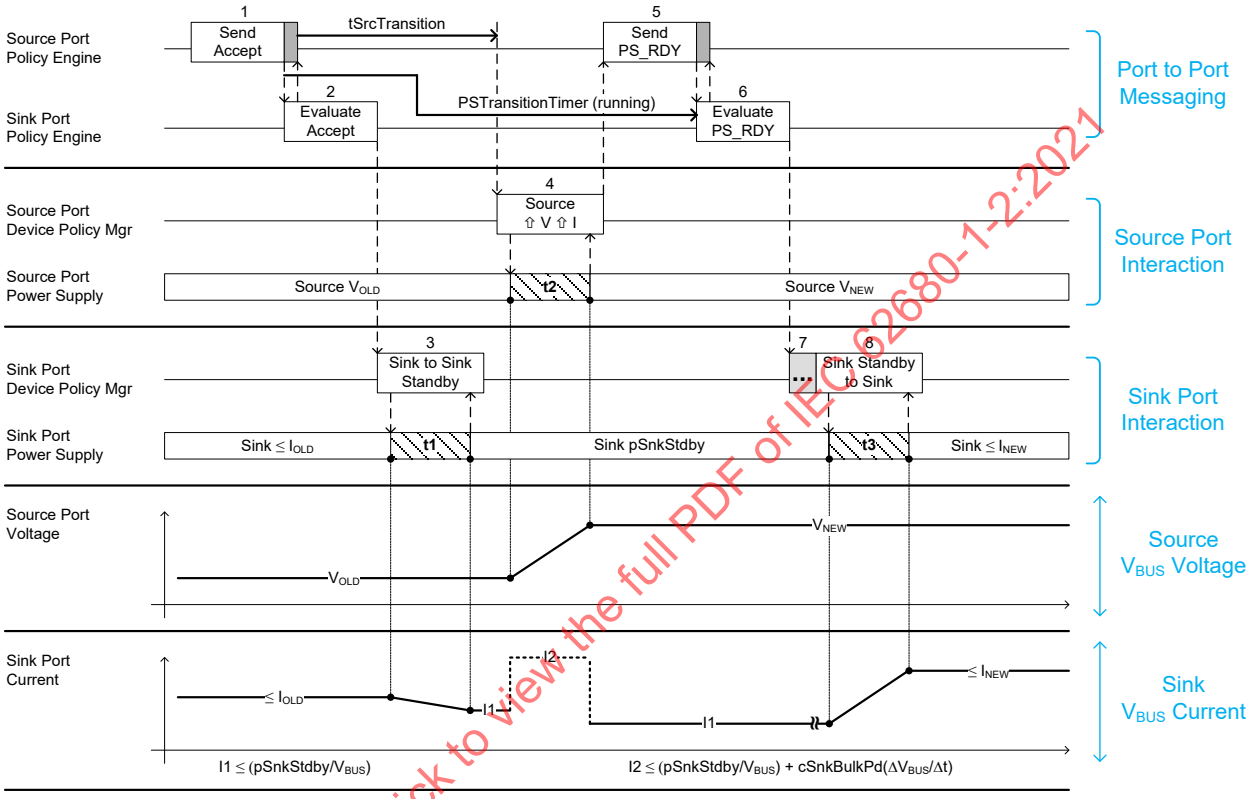


Table 7-3 Sequence Diagram for Increasing the Voltage and Current

Step	Source Port	Sink Port
1	Policy Engine sends the ***Accept*** Message to the Sink.	Policy Engine receives the ***Accept*** Message and starts the ***PSTransitionTimer***.
2	Protocol Layer receives the ***GoodCRC*** 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 ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***Accept*** Message.
3		Policy Engine tells the Device Policy Manager to instruct the power supply to reduce power consumption to ***pSnkStdby*** within ***tSnkStdby*** (t1); t1 ***Shall*** complete before ***tSrcTransition***. 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	***tSrcTransition*** after the ***GoodCRC*** Message was 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 ***tSrcReady*** (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 ***PS_RDY*** Message to the Sink.	The Policy Engine receives the ***PS_RDY*** Message from the Source.
6	Protocol Layer receives the ***GoodCRC*** Message from the Sink.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***PS_RDY*** Message from the Source and tells the Device Policy Manager it is okay to operate at the new power level.
7		The Sink ***May*** begin operating at the new power level any time after evaluation of the ***PS_RDY*** Message. This time duration is indeterminate.
8		The Sink ***Shall Not*** violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. The time duration (t3) depends on the magnitude of the load change.

7.3.4 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 7-23. 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 7-23 Transition Diagram for Increasing the Voltage and Decreasing the Current

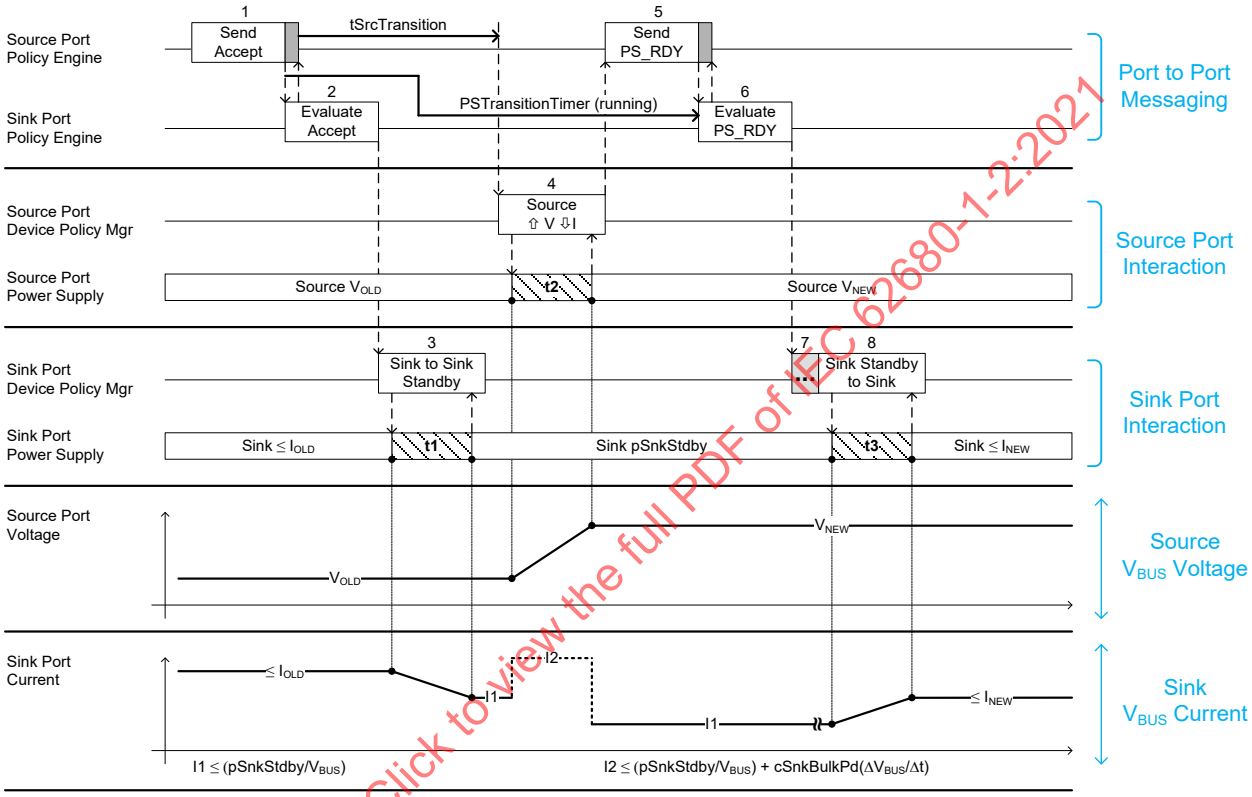


Table 7-4 Sequence Description for Increasing the Voltage and Decreasing the Current

Step	Source Port	Sink Port
1	Policy Engine sends the ***Accept*** Message to the Sink.	Policy Engine evaluates the ***Accept*** Message and starts the ***PSTransitionTimer***.
2	Protocol Layer receives the ***GoodCRC*** 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 ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***Accept*** Message.
3		Policy Engine tells the Device Policy Manager to instruct the power supply to reduce power consumption to ***pSnkStdby*** within ***tSnkStdby*** (t1); t1 ***Shall*** complete before ***tSrcTransition***. 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	***tSrcTransition*** after the ***GoodCRC*** Message was 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 ***tSrcReady*** (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 ***PS_RDY*** Message to the Sink.	The Policy Engine receives the ***PS_RDY*** Message from the Source.
6	Protocol Layer receives the ***GoodCRC*** Message from the Sink.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***PS_RDY*** Message from the Source and tells the Device Policy Manager it is okay to operate at the new power level.
7		The Sink ***May*** begin operating at the new power level any time after evaluation of the ***PS_RDY*** Message. This time duration is indeterminate.
8		The Sink ***Shall Not*** violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. The time duration (t3) depends on the magnitude of the load change.

7.3.5 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 7-24. 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 7-24 Transition Diagram for Decreasing the Voltage and Increasing the Current

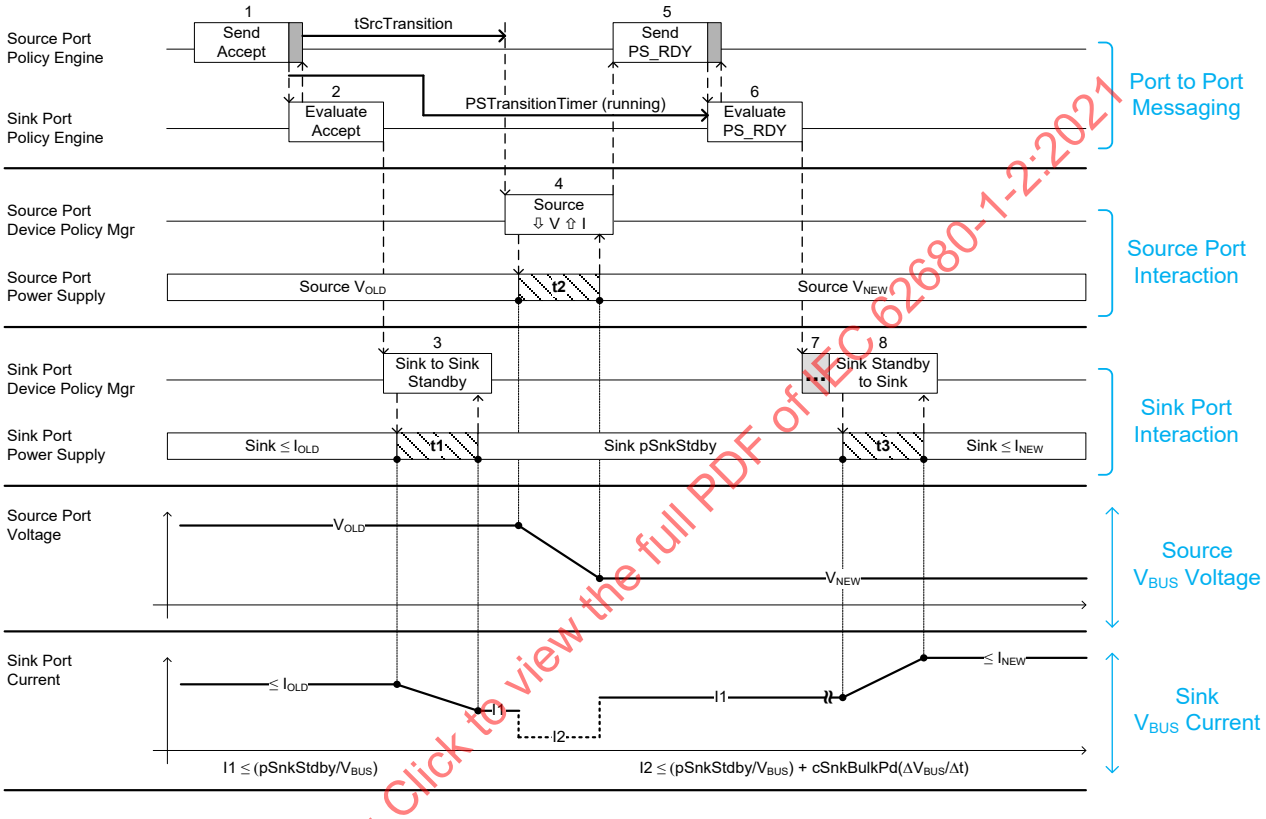


Table 7-5 Sequence Description for Decreasing the Voltage and Increasing the Current

Step	Source Port	Sink Port
1	Policy Engine sends the ***Accept*** Message to the Sink.	Policy Engine receives the ***Accept*** Message and starts the ***PSTransitionTimer***.
2	Protocol Layer receives the ***GoodCRC*** 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 ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***Accept*** Message.
3		Policy Engine tells the Device Policy Manager to instruct the power supply to reduce power consumption to ***pSnkStdby*** within ***tSnkStdby*** (t1); t1 ***Shall*** complete before ***tSrcTransition***. 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	***tSrcTransition*** after the ***GoodCRC*** Message was 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 ***tSrcReady*** (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 ***PS_RDY*** Message to the Sink.	The Policy Engine receives the ***PS_RDY*** Message from the Source.
6	Protocol Layer receives the ***GoodCRC*** Message from the Sink.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***PS_RDY*** Message from the Source and tells the Device Policy Manager it is okay to operate at the new power level.
7		The Sink ***May*** begin operating at the new power level any time after evaluation of the ***PS_RDY*** Message. This time duration is indeterminate.
8		The Sink ***Shall Not*** violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. The time duration (t3) depends on the magnitude of the load change.

7.3.6 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 7-25. 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 7-25 Transition Diagram for Decreasing the Current

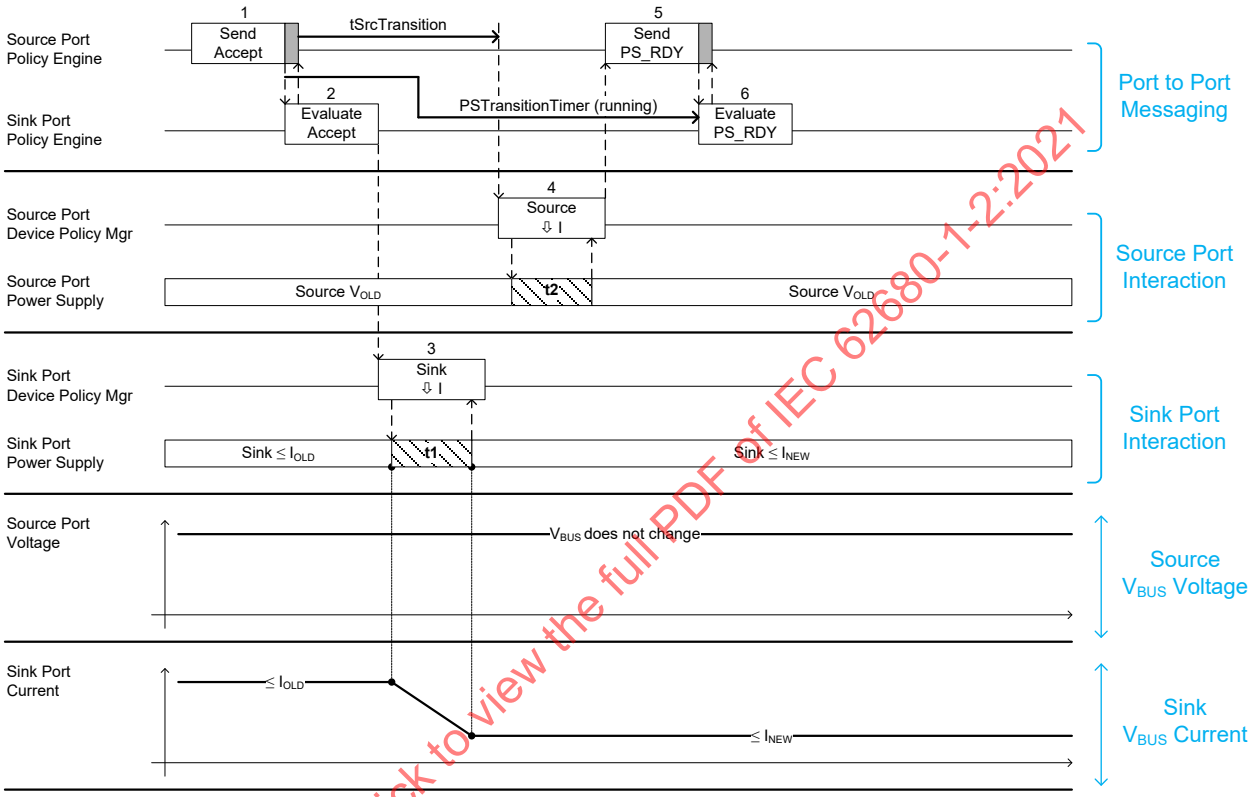


Table 7-6 Sequence Description for Decreasing the Current

Step	Source Port	Sink Port
1	Policy Engine sends the ***Accept*** Message to the Sink.	Policy Engine receives the ***Accept*** Message starts ***PSTransitionTimer***.
2	Protocol Layer receives the ***GoodCRC*** 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 ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***Accept*** 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 ***tSnkNewPower*** (t1); t1 ***Shall*** complete before ***tSrcTransition***.
4	***tSrcTransition*** after the ***GoodCRC*** Message was 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 ***tSrcReady*** (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 ***PS_RDY*** Message to the Sink.	The Policy Engine receives the ***PS_RDY*** Message from the Source.
6	Protocol Layer receives the ***GoodCRC*** Message from the Sink.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine evaluates the ***PS_RDY*** 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 7-26. 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 7-26 Transition Diagram for Decreasing the Voltage

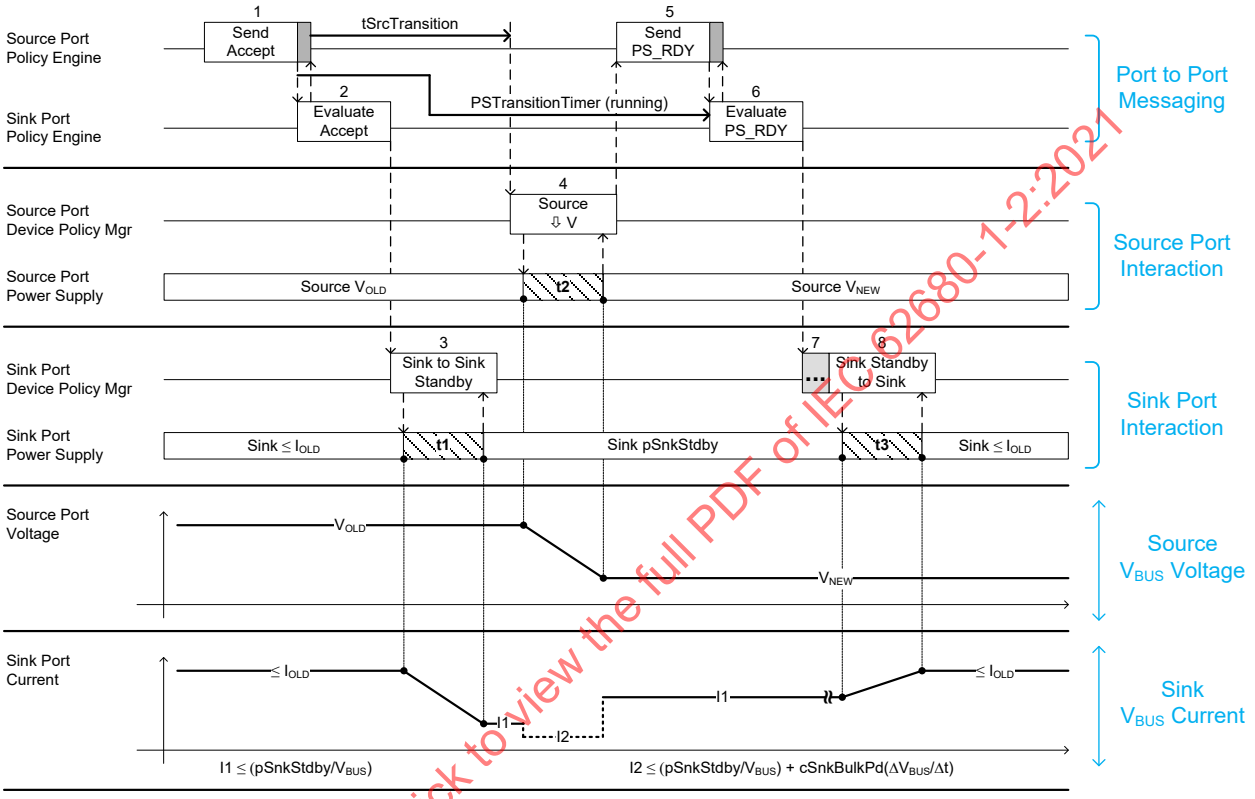


Table 7-7 Sequence Description for Decreasing the Voltage

Step	Source Port	Sink Port
1	Policy Engine sends the ***Accept*** Message to the Sink.	Policy Engine receives the ***Accept*** Message and starts the ***PSTransitionTimer***.
2	Protocol Layer receives the ***GoodCRC*** 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 ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***Accept*** Message.
3		Policy Engine tells the Device Policy Manager to instruct the power supply to reduce power consumption to ***pSnkStdby*** within ***tSnkStdby*** (t1); t1 ***Shall*** complete before ***tSrcTransition***. 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	***tSrcTransition*** after the ***GoodCRC*** Message was 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 ***tSrcReady*** (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 ***PS_RDY*** Message to the Sink.	The Policy Engine receives the ***PS_RDY*** Message from the Source.
6	Protocol Layer receives the ***GoodCRC*** Message from the Sink.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***PS_RDY*** Message from the Source and tells the Device Policy Manager it is okay to operate at the new power level.
7		The Sink ***May*** begin operating at the new power level any time after evaluation of the ***PS_RDY*** Message. This time duration is indeterminate.
8		The Sink ***Shall Not*** violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. The time duration (t3) depends on the magnitude of the load change.

7.3.8 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 7-27. 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 7-27 Transition Diagram for Decreasing the Voltage and the Current

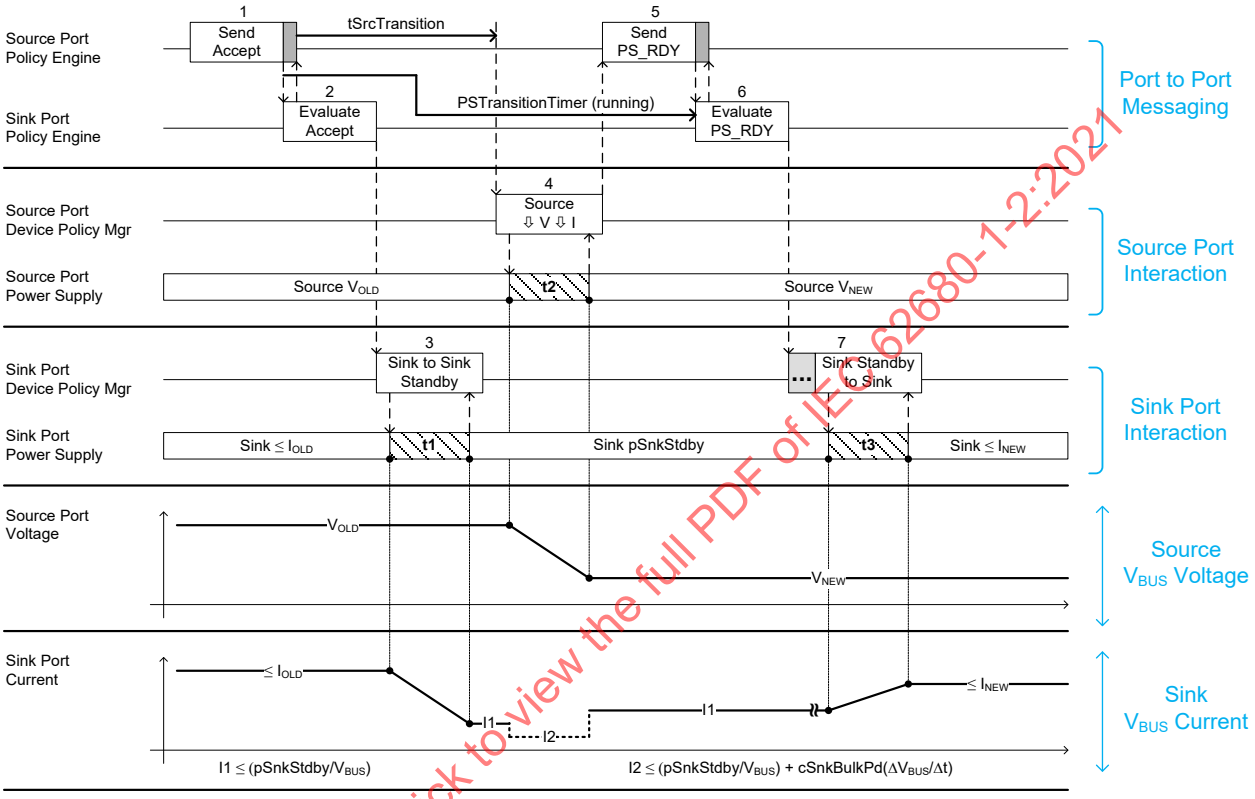


Table 7-8 Sequence Description for Decreasing the Voltage and the Current

Step	Source Port	Sink Port
1	Policy Engine sends the ***Accept*** Message to the Sink.	Policy Engine receives the ***Accept*** Message and starts the ***PSTransitionTimer***.
2	Protocol Layer receives the ***GoodCRC*** 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 ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***Accept*** Message.
3		Policy Engine tells the Device Policy Manager to instruct the power supply to reduce power consumption to ***pSnkStdby*** within ***tSnkStdby*** (t1); t1 ***Shall*** complete before ***tSrcTransition***. 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	***tSrcTransition*** after the ***GoodCRC*** Message was 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 ***tSrcReady*** (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 ***PS_RDY*** Message to the Sink.	The Policy Engine receives the ***PS_RDY*** Message from the Source.
6	Protocol Layer receives the ***GoodCRC*** Message from the Sink.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***PS_RDY*** Message from the Source and tells the Device Policy Manager it is okay to operate at the new power level.
7		The Sink ***May*** begin operating at the new power level any time after evaluation of the ***PS_RDY*** Message. This time duration is indeterminate.
8		The Sink ***Shall Not*** violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. The time duration (t3) depends on the magnitude of the load change.

7.3.9 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 7-28. The sequence that ***Shall*** be followed is described in Table 7-9. 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 7-28 Transition Diagram for a Sink Requested Power Role Swap

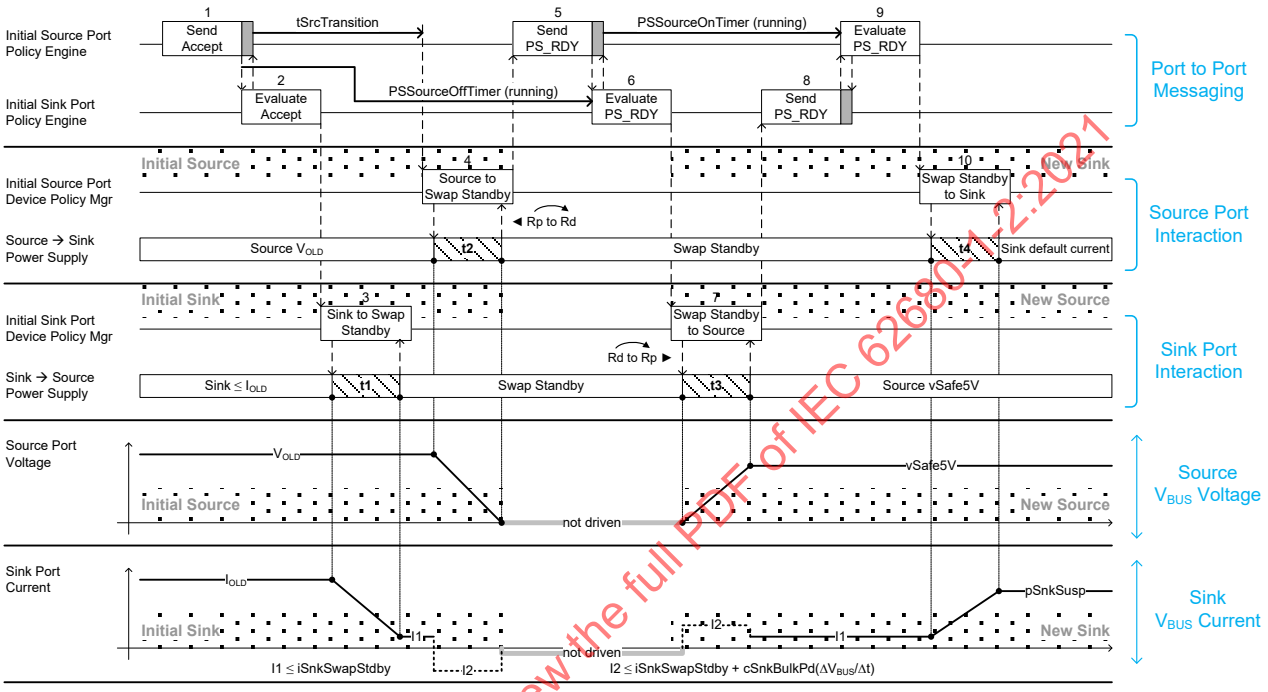


Table 7-9 Sequence Description for a Sink Requested Power Role Swap

Step	Initial Source Port → New Sink Port	Initial Sink Port → New Source Port
1	Policy Engine sends the ***Accept*** Message to the Initial Sink.	Policy Engine receives the ***Accept*** and starts the ***PSSourceOffTimer***.
2	Protocol Layer receives the ***GoodCRC*** 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 ***GoodCRC*** Message to the Initial Source. Policy Engine then evaluates the ***Accept*** Message.
3		Policy Engine tells the Device Policy Manager to instruct the power supply to transition to Swap Standby within ***tSnkStdby*** (t1); t1 ***Shall*** complete before ***tSrcTransition***. When in Sink Standby the Initial Sink ***Shall Not*** draw more than ***iSnkSwapStdby*** (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	***tSrcTransition*** after the ***GoodCRC*** Message was received the power supply starts to change its output power capability to Swap Standby (see Section 7.1.10). The power supply ***Shall*** complete the transition to Swap Standby within ***tSrcSwapStdby*** (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 2.0]***). The power supply status is passed to the Policy Engine.	
5	The power supply is ready, and the Policy Engine sends the ***PS_RDY*** Message to the device that will become the new Source.	
6	Protocol Layer receives the ***GoodCRC*** Message from the device that will become the new Source. Policy Engine starts the ***PSSourceOnTimer***. Upon sending the ***PS_RDY*** Message and receiving the ***GoodCRC*** Message the Initial Source is ready to be the new Sink.	Policy Engine stops the ***PSSourceOffTimer.*** The Protocol Layer sends the ***GoodCRC*** 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 2.0]***). The power supply as the new Source transitions from Swap Standby to sourcing default ***vSafe5V*** within ***tNewSrc*** (t3). The power supply informs the Device Policy Manager that it is operating as the new Source.
8	Policy Engine receives the ***PS_RDY*** Message from the Source.	Device Policy Manager informs the Policy Engine the power supply is ready, and the Policy Engine sends the ***PS_RDY*** Message to the new Sink.
9	Policy Engine stops the ***PSSourceOnTimer***. Protocol Layer sends the ***GoodCRC*** Message to the new Source. Policy Engine evaluates the ***PS_RDY*** 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 ***GoodCRC*** Message from the new Sink.

Step	Initial Source Port → New Sink Port	Initial Sink Port → New Source Port
10	The power supply as the new Sink transitions from Swap Standby to drawing ***pSnkSusp*** within ***tNewSnk*** (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.	

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 7-29. 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. Note in this figure, the Sink has previously sent a ***PR_Swap*** Message to the Source.

Figure 7-29 Transition Diagram for a Source Requested Power Role Swap

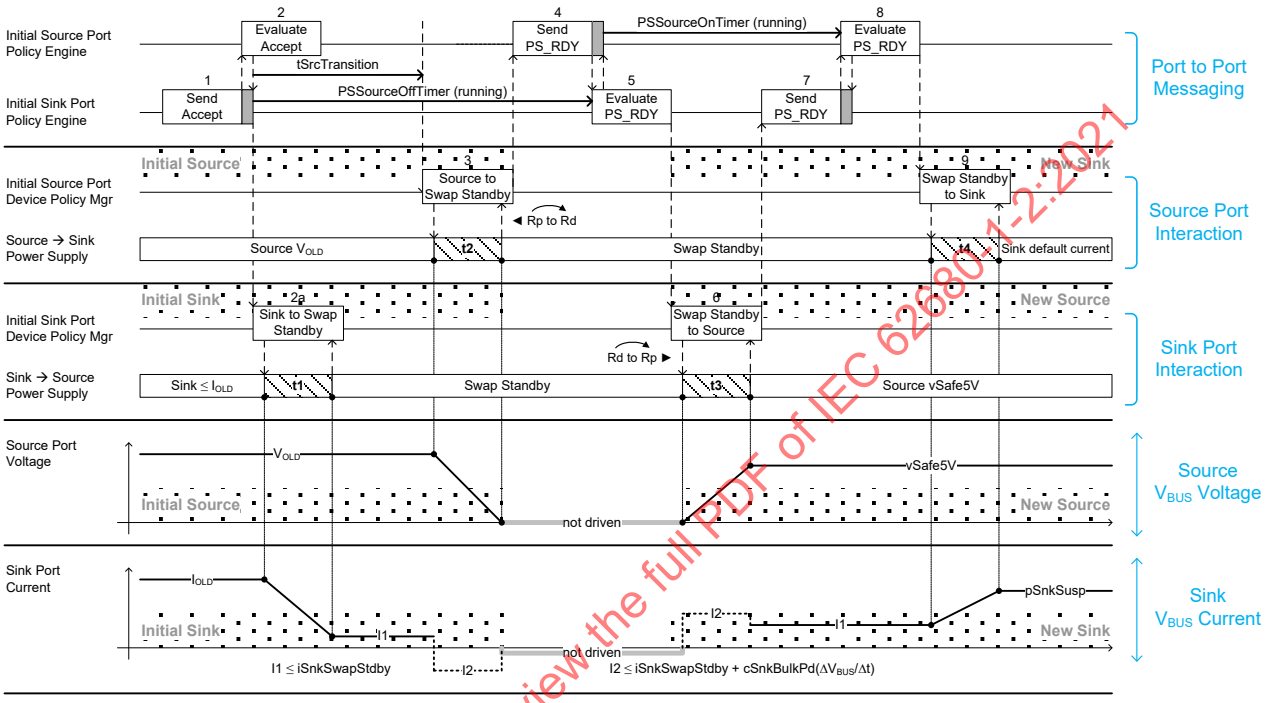


Table 7-10 Sequence Description for a Source Requested Power Role Swap

Step	Initial Source Port→ New Sink Port	Initial Sink Port → New Source Port
1	Policy Engine receives the ***Accept*** Message.	Policy Engine sends the ***Accept*** Message to the Initial Source.
2	Protocol Layer receives the ***GoodCRC*** 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 ***GoodCRC*** Message from the Initial Source. Policy Engine starts the ***PSSourceOffTimer***.
2a		The Policy Engine tells the Device Policy Manager to instruct the power supply to transition to Swap Standby. The power supply ***Shall*** complete the transition to Swap Standby within ***tSnkStdby*** (t1); t1 ***Shall*** complete before ***tSrcTransition***. The Sink ***Shall Not*** violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. Policy Engine starts ***PSSourceOffTimer***. When in Sink Standby the Initial Sink ***Shall Not*** draw more than ***iSnkSwapStdby*** (I1).
3	***tSrcTransition*** after the ***GoodCRC*** Message was received the power supply starts to change its output power capability to Swap Standby (see Section 7.1.10). The power supply ***Shall*** complete the transition to Swap Standby within ***tSrcSwapStdby*** (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 2.0]***). The power supply status is passed to the Policy Engine.	
4	The Policy Engine sends the ***PS_RDY*** Message to the soon to be new Source.	Policy Engine receives the ***PS_RDY*** Message and stops the ***PSSourceOffTimer***.
5	Protocol Layer receives the ***GoodCRC*** Message from the soon to be new Source. Policy Engine starts the ***PSSourceOnTimer***. At this point the Initial Source is ready to be the new Sink.	Protocol Layer sends the ***GoodCRC*** Message to the new Sink. Upon evaluating the ***PS_RDY*** 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 ***[USB Type-C 2.0]***). The power supply as the new Source transitions from Swap Standby to sourcing default ***vSafe5V*** within ***tNewSrc*** (t3). The power supply informs the Device Policy Manager that it is operating as the new Source.
7	Policy Engine receives the ***PS_RDY*** Message and stops the ***PSSourceOnTimer***.	Device Policy Manager informs the Policy Engine the power supply is ready, and the Policy Engine sends the ***PS_RDY*** Message to the new Sink.
8	Protocol Layer sends the ***GoodCRC*** Message to the new Source. Policy Engine evaluates the ***PS_RDY*** 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 ***GoodCRC*** Message from the new Sink.

Step	Initial Source Port→ New Sink Port	Initial Sink Port → New Source Port
9	The power supply as the new Sink transitions from Swap Standby to drawing ***pSnkSusp*** within ***tNewSnk*** (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.	

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 7-30. The sequence that ***Shall*** be followed is described in Table 7-11. The timing parameters that ***Shall*** be followed are listed in Table 7-22 and Table 7-11.

Figure 7-30 Transition Diagram for a GotoMin Current Decrease

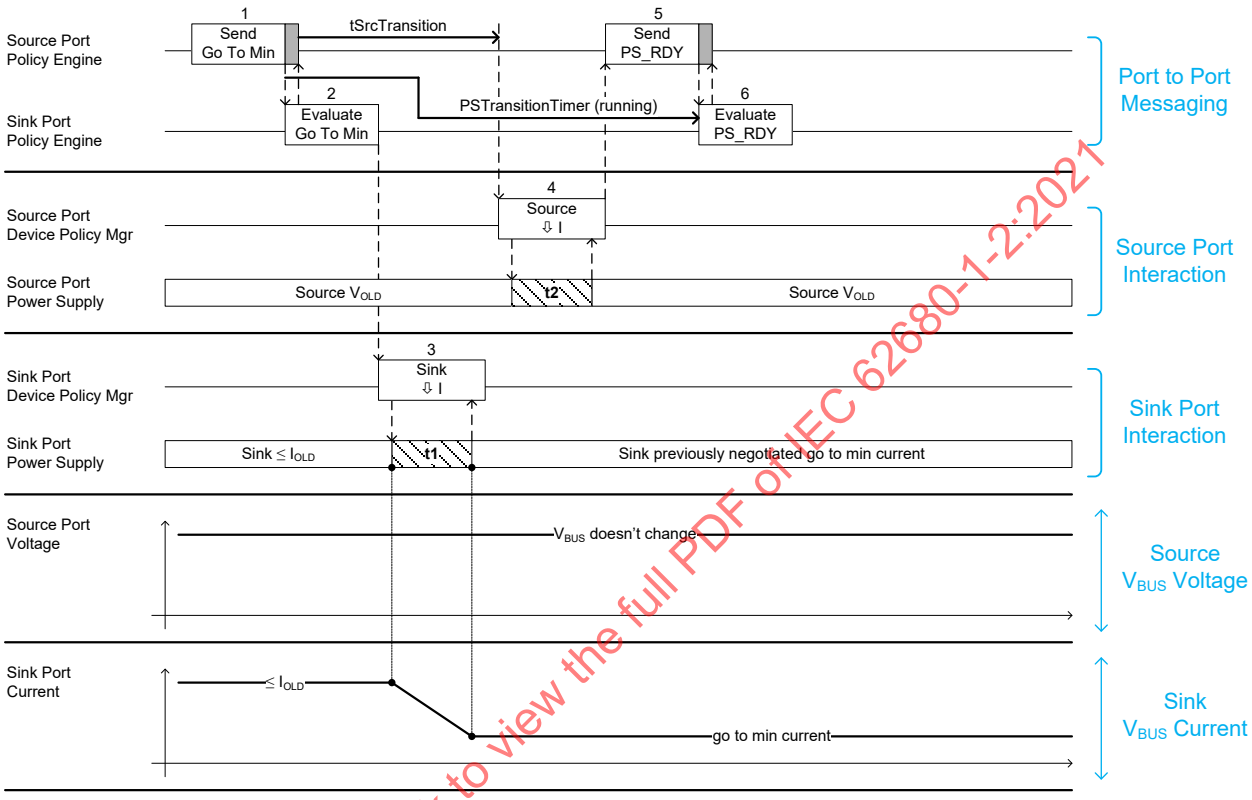


Table 7-11 Sequence Description for a GotoMin Current Decrease

Step	Source Port	Sink Port
1	Policy Engine sends the ***GotoMin*** Message to the Sink.	Policy Engine receives the ***GotoMin*** Message and starts the ***PSTransitionTimer.***
2	Protocol Layer receives the ***GoodCRC*** 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 ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***GotoMin*** Message.
3		Policy Engine tells the Device Policy Manager to instruct the power supply to reduce power consumption, within ***tSnkNewPower*** (t1), to the pre-negotiated go to reduced power level); t1 ***Shall*** complete before ***tSrcTransition***. 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	***tSrcTransition*** after the ***GoodCRC*** Message was 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 ***tSrcReady*** (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 ***PS_RDY*** Message to the Sink.	The Policy Engine receives the ***PS_RDY*** Message.
6	Protocol Layer receives the ***GoodCRC*** Message from the Sink.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine evaluates the ***PS_RDY*** 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 7-31. The sequence that ***Shall*** be followed is described in Table 7-12. The timing parameters that ***Shall*** be applied are listed in Table 7-22 and Table 7-23.

Figure 7-31 Transition Diagram for a Source Initiated Hard Reset

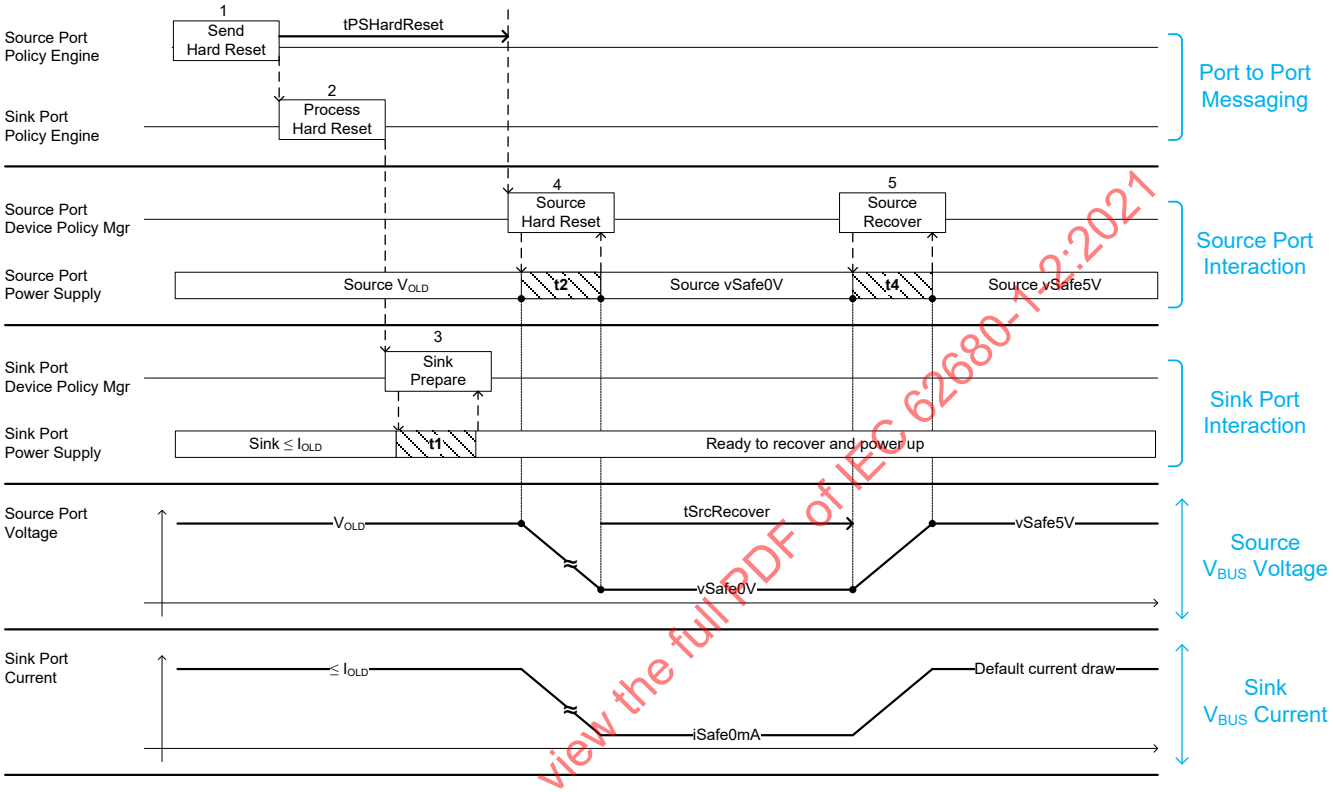


Table 7-12 Sequence Description for a Source Initiated Hard Reset

Step	Source Port	Sink Port
1	Policy Engine sends ***Hard Reset*** Signaling to the Sink.	Sink receives ***Hard Reset*** 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 ***tSnkHardResetPrepare*** (t1) and passes an indication to the Device Policy Manger The Sink ***Shall Not*** draw more than ***iSafe0mA*** when V_{BUS} is driven to ***vSafe0V***.
4	Policy Engine waits ***tPSHardReset*** after sending ***Hard Reset*** Signaling and then tells the Device Policy Manager to instruct the power supply to perform a Hard Reset. The transition to ***vSafe0V*** ***Shall*** occur within ***tSafe0V*** (t2).	
5	After ***tSrcRecover*** 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 ***vSafe5V*** ***Shall*** occur within ***tSrcTurnOn*** (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 7-32. 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 7-23.

Figure 7-32 Transition Diagram for a Sink Initiated Hard Reset

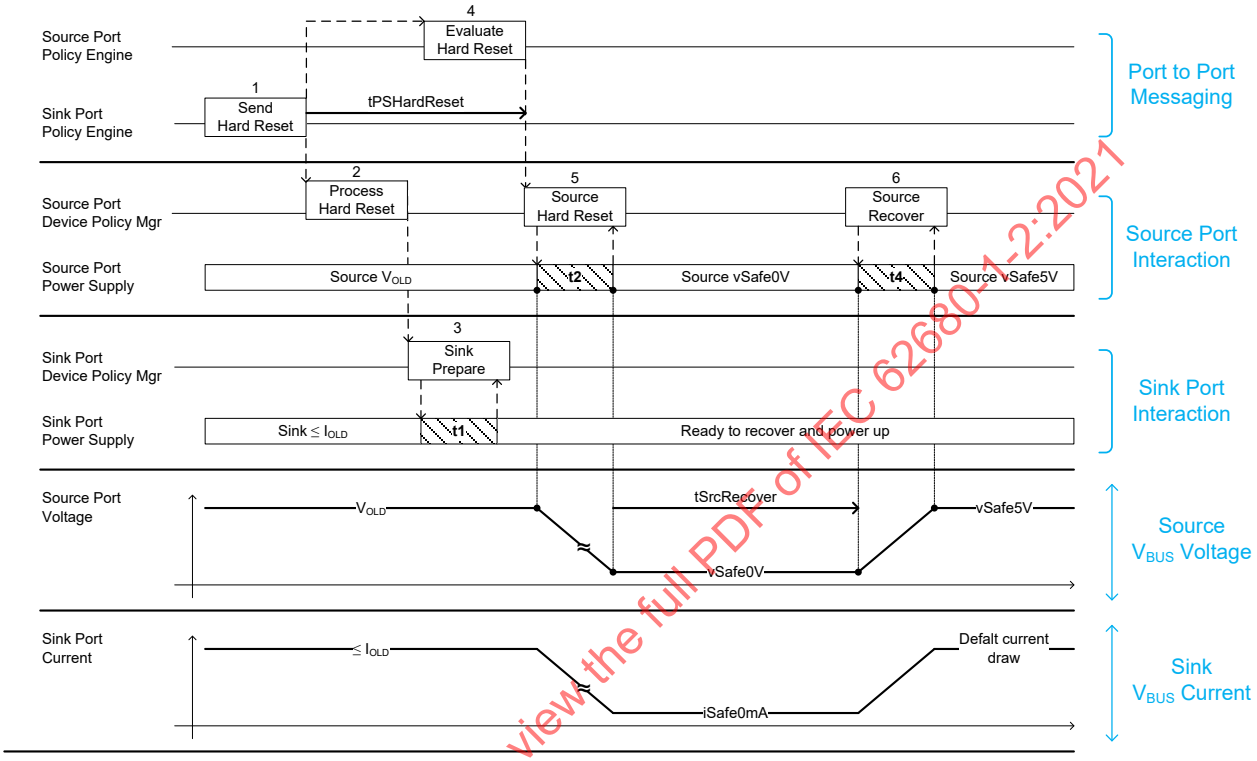


Table 7-13 Sequence Description for a Sink Initiated Hard Reset

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 ***tSnkHardResetPrepare*** (t1) and passes an indication to the Device Policy Manger. The Sink ***Shall Not*** draw more than ***iSafe0mA*** when V_{BUS} is driven to ***vSafe0V***.
4	Policy Engine is informed of the Hard Reset.	
5	Policy Engine waits ***tPSHardReset*** 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 ***vSafe0V*** ***Shall*** occur within ***tSafe0V*** (t2).	
6	After ***tSrcRecover*** 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 ***vSafe5V*** ***Shall*** occur within ***tSrcTurnOn*** (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 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 7-33. The sequence that ***Shall*** be followed is described in Table 7-14. The timing parameters that ***Shall*** be followed are listed in Table 7-22 and Table 7-23.

Figure 7-33 Transition Diagram for no change in Current or Voltage

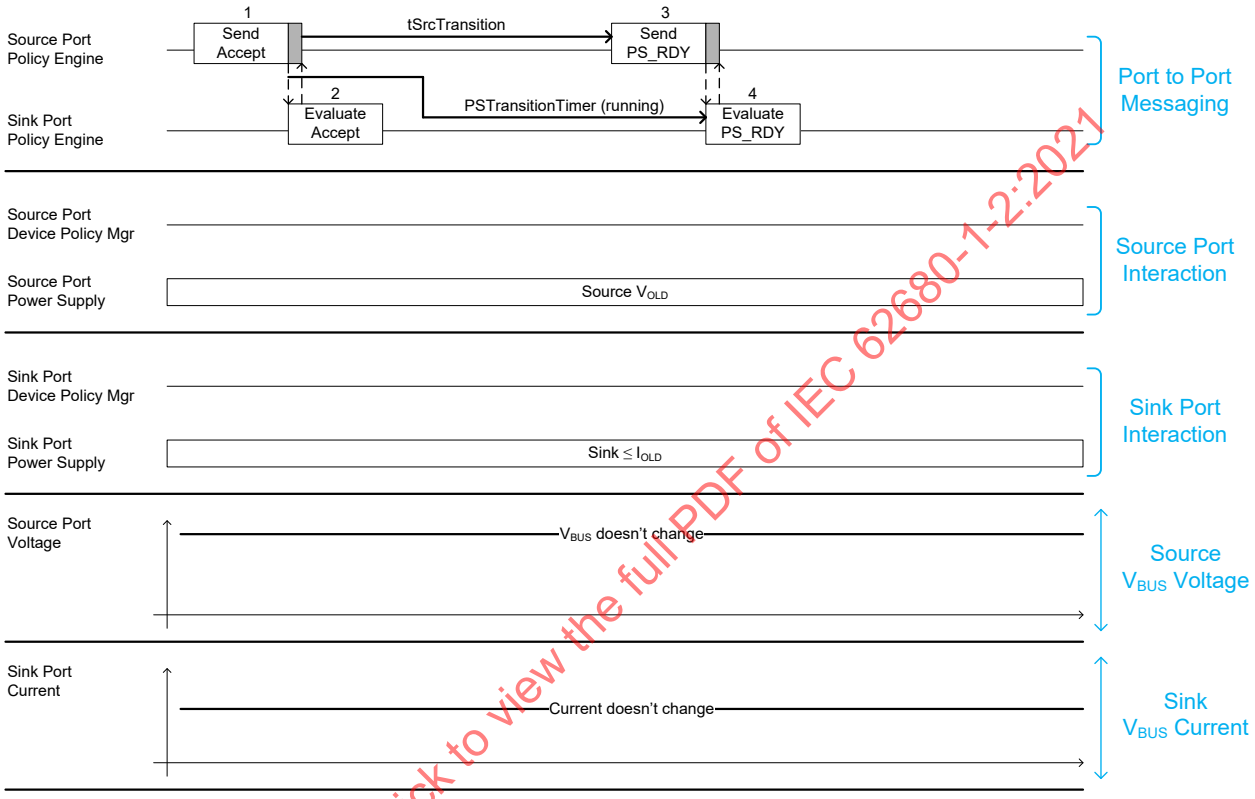


Table 7-14 Sequence Description for no change in Current or Voltage

Step	Source Port	Sink Port
1	Policy Engine sends the ***Accept*** Message to the Sink.	Policy Engine receives the ***Accept*** Message and starts the ***PSTransitionTimer***.
2	Protocol Layer receives the ***GoodCRC*** Message from the Sink.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***Accept*** Message.
3	The Policy Engine waits ***tSrcTransition*** then sends the ***PS_RDY*** Message to the Sink.	Policy Engine receives the ***PS_RDY*** Message.
4	Policy Engine receives the ***GoodCRC*** 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 ***GoodCRC*** Message to the Source. Policy Engine evaluates the ***PS_RDY*** Message.

7.3.15 Fast Role Swap

The interaction of the System Policy, Device Policy, and power supply that ***Shall*** be followed during a Fast Role Swap is shown in Figure 7-34. The parallel sequences that ***Shall*** be followed are described in Table 7-15. The timing parameters that ***Shall*** be followed are listed in Table 7-22 and Table 7-23. Negotiations between the new Source and the new Sink ***May*** occur after the new Source sends the final ***PS_RDY*** Message. Note: in Figure 7-34 and Table 7-15 numbers are used to indicate Message related steps and letters are used to indicate other events.

Figure 7-34 Transition Diagram for Fast Role Swap

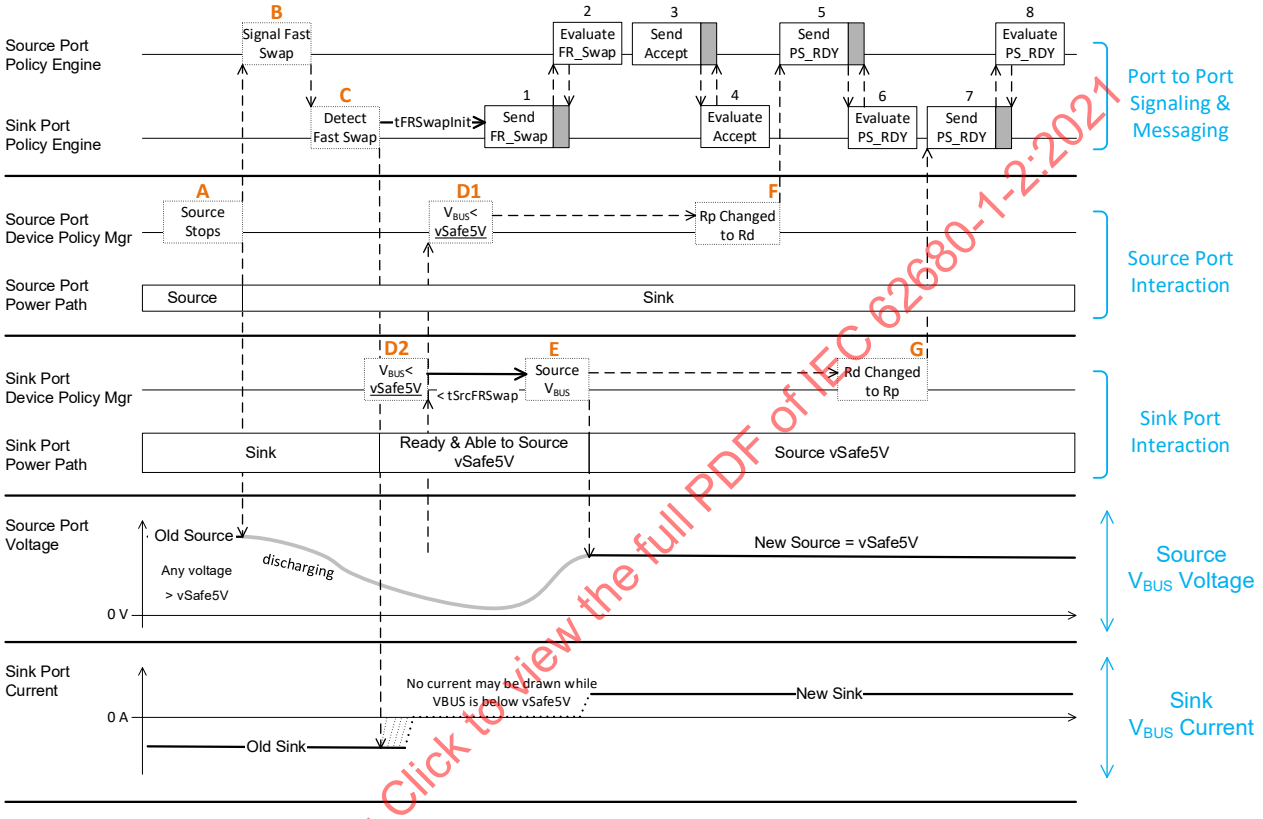


Table 7-15 Sequence Description for Fast Role Swap

Step	Initial Source Port→ New Sink Port	Initial Sink Port → New Source Port
Fast Role Swap Signaling and Power Transition		
A	The Source connected to the Hub UFP (see Figure 7-14) stops sourcing V_{BUS}.	
B	Policy Engine signals the Fast Role Swap to the initial Sink on the CC wire. When V_{BUS} < ***vSafe5V*** (min), it tells the Device Policy Manager not to draw more than ***pSnkStdby*** until the ***tSnkFRSwap*** timer has elapsed.	
C		Policy Engine detects the Fast Role swap signal on the CC wire from the initial Source and ***Shall*** send the ***FR_Swap*** Message back to the initial Source (that is no longer powering V_{BUS}) within time ***tFRSwapInit***.
D1	The Policy engine monitors for V_{BUS} ≤ ***vSafe5V*** so that a ***PS_RDY*** Message can be sent to the new Source at Step 5 of the messaging sequence.	
D2		The Policy engine monitors for V_{BUS} ≤ ***vSafe5V*** so the initial Sink can assume the role of new Source and begin to source V_{BUS}.

Step	Initial Source Port→ New Sink Port	Initial Sink Port → New Source Port
E		When V_{BUS} = ***vSafe5V*** the new Source ***May*** provide power to V_{BUS}. When V_{BUS} < ***vSafe5V*** the new Source ***Shall*** provide power to V_{BUS} within ***tSrcFRSwap*** and the ***PS_RDY*** Message can be sent to the new Sink at Step 7 of the messaging sequence.
F	The CC termination is changed from Rp to Rd (see ***[USB Type-C 2.0]***) before the new Sink sends the ***PS_RDY*** Message of Step 5 to the new Source.	
G		The CC termination is changed from Rd to Rp (see ***[USB Type-C 2.0]***) before the new Source sends the ***PS_RDY*** Message of Step 7 to the new Sink.
Fast Role Swap Message Sequence		
1	Policy Engine receives the ***FR_Swap*** Message from the initial Sink that is transitioning to be the new Source.	Policy Engine sends the ***FR_Swap*** Message to the initial Source(that is no longer powering V_{BUS}) after detecting the Fast Role Swap signal of Step C.
2	Protocol Layer sends the ***GoodCRC*** Message to the initial Sink. Policy Engine then evaluates the ***FR_Swap*** Message.	Protocol Layer receives the ***GoodCRC*** Message from the initial Source.
3	Policy Engine sends an ***Accept*** Message to the initial Sink that is transitioning to be the new Source.	Policy Engine receives the ***Accept*** Message from the initial Source that is transitioning to be the new Sink.
4	Protocol Layer receives the ***GoodCRC*** Message from the initial Sink that is transitioning to be the new Source.	Protocol Layer sends the ***GoodCRC*** Message to the initial Source that is transitioning to be the new Sink.
5	Policy Engine sends a ***PS_RDY*** Message to the initial Sink that is transitioning to be the new Source. The Policy Engine ***Shall*** wait for Step D1 before sending the ***PS_RDY*** Message, and ***Shall*** send the ***PS_RDY*** Message within ***tFRSwap5V*** of sending the ***Accept*** Message.	Policy Engine receives the ***PS_RDY*** Message from the new Sink.
6	Protocol Layer receives the ***GoodCRC*** Message from the new Source.	Protocol Layer sends the ***GoodCRC*** Message from the initial Sink that has completed the transition to new Source. Policy Engine then evaluates the ***PS_RDY*** Message.
7	Policy Engine receives the ***PS_RDY*** Message from the new Source.	Policy Engine sends a ***PS_RDY*** Message to the new Sink. The Policy Engine ***Shall*** wait for Step E before sending the ***PS_RDY*** Message, and ***Shall*** send the ***PS_RDY*** Message within ***tFRSwapComplete*** of receiving the ***PS_RDY*** Message from the Initial Source Port.

7.3.16 Increasing the Programmable Power Supply Voltage

The interaction of the System Policy, Device Policy, and power supply that ***Shall*** be followed when increasing the voltage is shown in Figure 7-35. 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. Note in this figure, the Sink has previously sent a ***Request*** Message to the Source.

Figure 7-35 Transition Diagram for Increasing the Programmable Power Supply Voltage

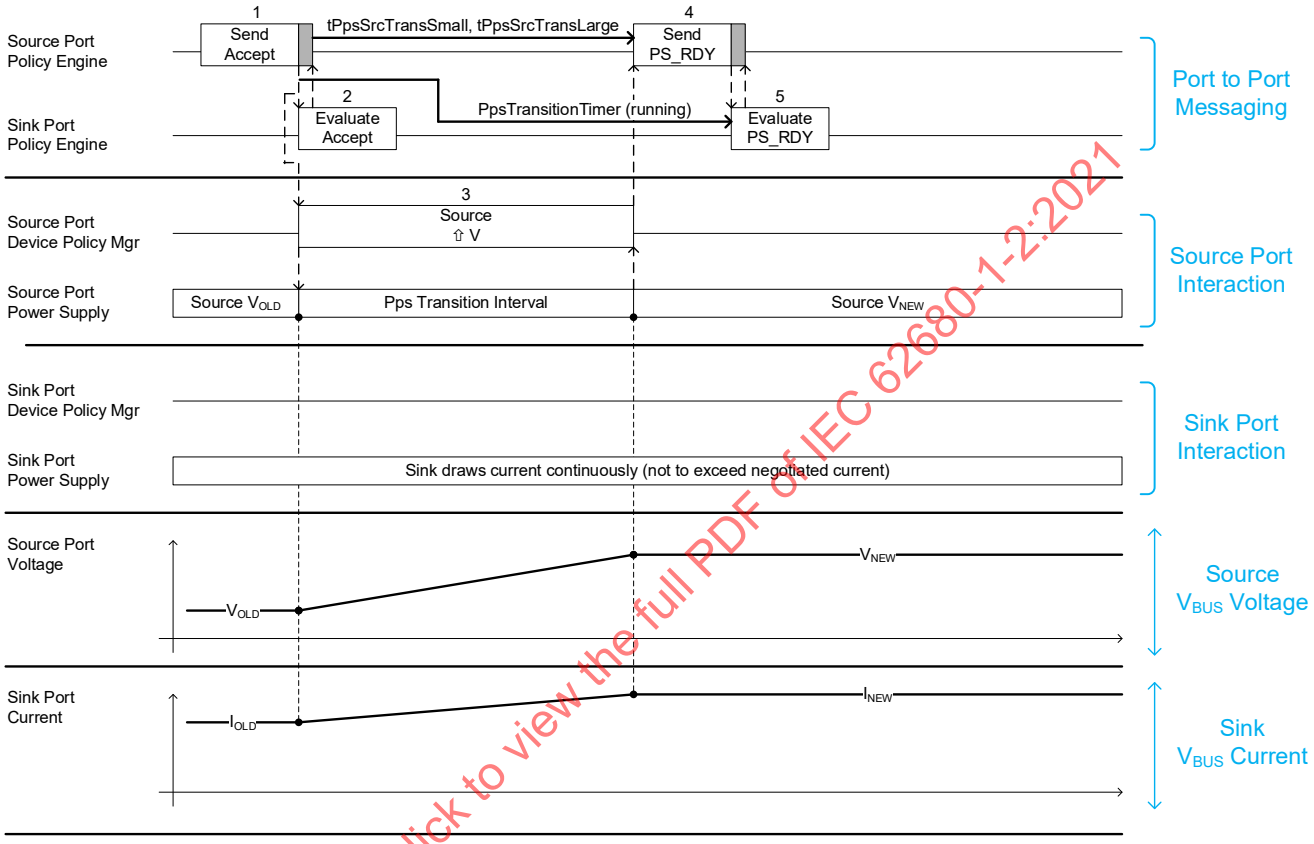


Table 7-16 Sequence Description for Increasing the Programmable Power Supply Voltage

Step	Source Port	Sink Port
1	Policy Engine sends the ***Accept*** Message to the Sink.	Policy Engine receives the ***Accept*** Message and starts the ***PSTransitionTimer***.
2	Protocol Layer receives the ***GoodCRC*** Message from the Sink. The Policy Engine tells the Device Policy Manager to instruct the power supply to increase its output voltage.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine. Policy Engine then evaluates the ***Accept*** Message.
3	After sending the ***Accept*** Message, the Programmable Power Supply starts to increase its output voltage. The Programmable Power Supply new voltage set-point ***Shall*** be reached by ***tPpsSrcTransLarge*** for steps larger than ***vPpsSmallStep*** or else by ***tPpsSrcTransSmall*** The power supply informs the Device Policy Manager that it has reached the new set-point and whether V_{BUS} is at the corresponding new level, or if the supply is operating in CL mode. The power supply status is passed to the Policy Engine.	
4	The Policy Engine sends the ***PS_RDY*** Message to the Sink.	The Policy Engine receives the ***PS_RDY*** Message from the Source.

Step	Source Port	Sink Port
5	Protocol Layer receives the ***GoodCRC*** Message from the Sink.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***PS_RDY*** Message from the Source and tells the Device Policy Manager that the Programmable Power Supply is operating at the new voltage set-point.

7.3.17 Decreasing the Programmable Power Supply Voltage

The interaction of the System Policy, Device Policy, and power supply that ***Shall*** be followed when decreasing the voltage is shown in Figure 7-36. 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. Note in this figure, the Sink has previously sent a ***Request*** Message to the Source.

Figure 7-36 Transition Diagram for Decreasing the Programmable Power Supply Voltage

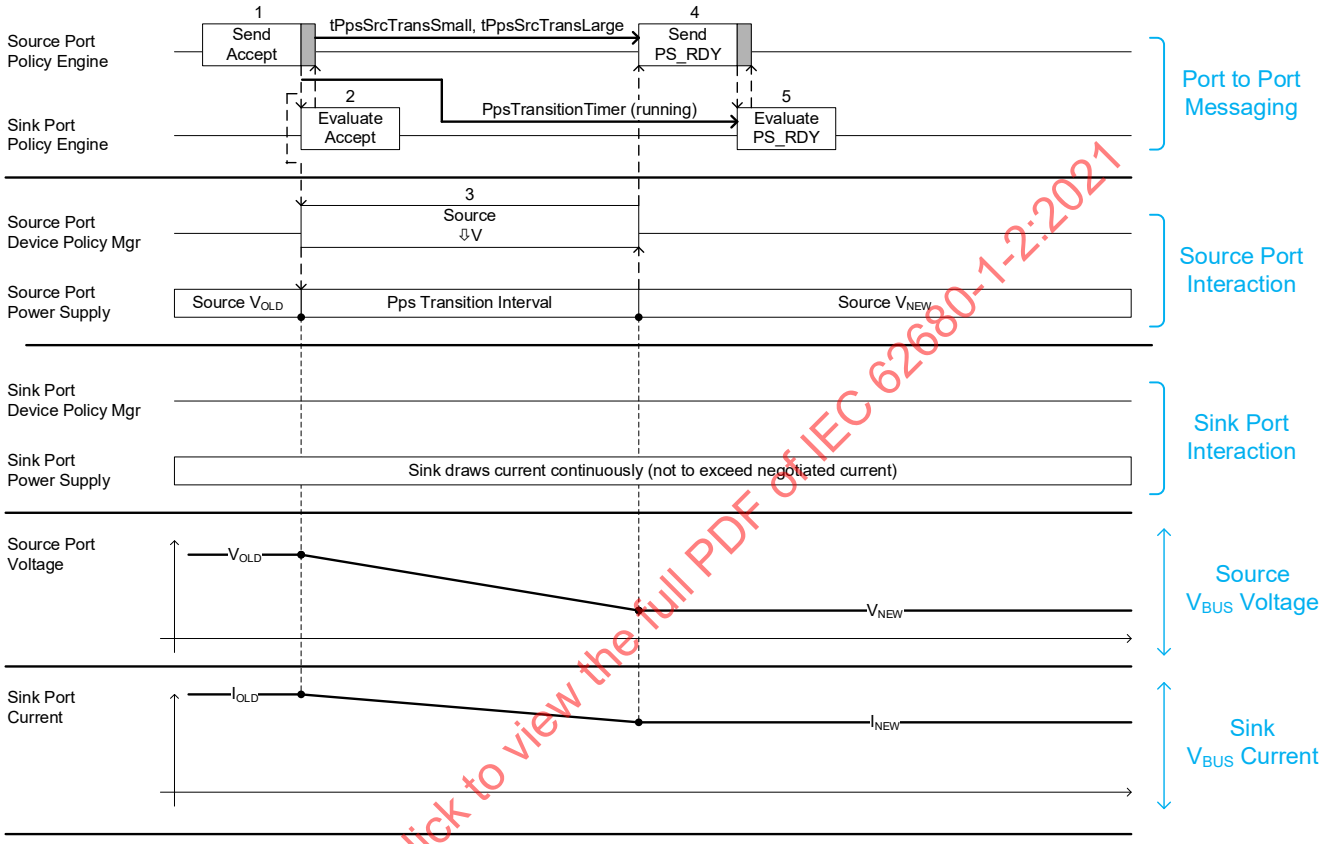


Table 7-17 Sequence Description for Decreasing the Programmable Power Supply Voltage

Step	Source Port	Sink Port
1	Policy Engine sends the ***Accept*** Message to the Sink.	Policy Engine receives the ***Accept*** Message and starts the ***PSTransitionTimer***.
2	Protocol Layer receives the ***GoodCRC*** Message from the Sink. The Policy Engine tells the Device Policy Manager to instruct the power supply to decrease its output voltage.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine. Policy Engine then evaluates the ***Accept*** Message.
3	After sending the ***Accept*** Message, the Programmable Power Supply starts to decrease its output voltage. The Programmable Power Supply new voltage set-point (corresponding to ***vPpsNew***) ***Shall*** be reached by ***tPpsSrcTransLarge*** for steps larger than ***vPpsSmallStep*** or else by ***tPpsSrcTransSmall***. The power supply informs the Device Policy Manager that it is has reached the new level. The power supply status is passed to the Policy Engine.	
4	The Policy Engine sends the ***PS_RDY*** Message to the Sink.	The Policy Engine receives the ***PS_RDY*** Message from the Source.
5	Protocol Layer receives the ***GoodCRC*** Message from the Sink.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***PS_RDY*** Message from the Source and tells the Device Policy Manager that the Programmable Power Supply is operating at the new voltage set-point (corresponding to ***vPpsNew***).

7.3.18 Changing the Source PDO or APDO

The interaction of the Device Policy Manager, the port Policy Engine and the Power Supply when changing *between* Source PDOs and APDOs, as listed below, is shown in Figure 7-37.

- PDO to PDO
- PDO to APDO
- APDO to APDO
- APDO to PDO

The Source voltage as the transition starts ***Shall*** be any voltage within the ***Valid*** V_{BUS} range of the previous Source PDO or APDO. The Source voltage after the transition is complete ***Shall*** be any voltage within the ***Valid*** V_{BUS} range of the new Source PDO or APDO. 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. Note in this figure, the Sink has previously sent a ***Request*** Message to the Source.

Figure 7-37 Transition Diagram for Changing the Source PDO or APDO

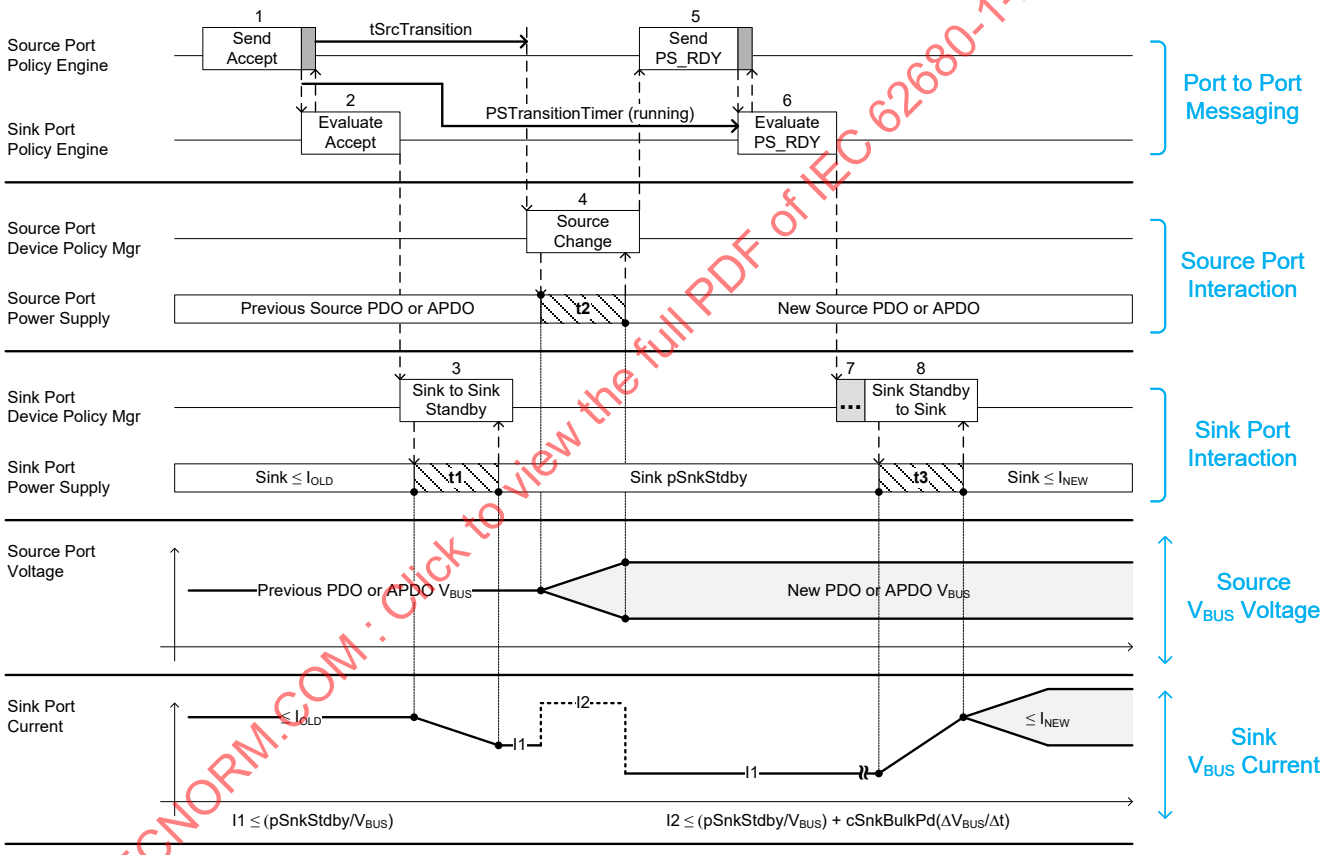


Table 7-18 Sequence Description for Changing the Source PDO or APDO

Step	Source Port	Sink Port
1	Policy Engine sends the ***Accept*** Message to the Sink.	Policy Engine receives the ***Accept*** Message and starts the ***PSTransitionTimer***.
2	Protocol Layer receives the ***GoodCRC*** Message from the Sink. The Policy Engine tells the Device Policy Manager to instruct the power supply to change to the new Source PDO or APDO.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine. Policy Engine then evaluates the ***Accept*** Message.

Step	Source Port	Sink Port
3		Policy Engine tells the Device Policy Manager to instruct the power supply to reduce power consumption to ***pSnkStdby*** within ***tSnkStdby*** (t1); t1 Shall complete before ***tSrcTransition***. 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	***tSrcTransition*** after the ***GoodCRC*** Message was received the Source starts to change to the new PDO or APDO. The Source ***Shall*** be ready to operate at the new power level within ***tSrcReady*** (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 ***PS_RDY*** Message to the Sink.	The Policy Engine receives the ***PS_RDY*** Message from the Source.
6	Protocol Layer receives the ***GoodCRC*** Message from the Sink.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***PS_RDY*** Message from the Source and tells the Device Policy Manager that the Source is operating at the new PDO or APDO.
7		The Sink ***May*** begin operating at the new power level any time after evaluation of the ***PS_RDY*** Message. This time duration is indeterminate.
8		The Sink ***Shall Not*** violate the transient load behavior defined in Section 7.2.6 while transitioning to and operating at the new power level. The time duration (t3) depends on the magnitude of the load change.

7.3.19 Increasing the Programmable Power Supply Current

The interaction of the System Policy, Device Policy, and power supply that ***Shall*** be followed when increasing the current limit in the same APDO, not exceeding the maximum for that APDO and without changing the requested voltage is shown in Figure 7-38. 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. Note in this figure, the Sink has previously sent a ***Request*** Message to the Source.

The Sink ***May*** draw current equal to the increasing Current Limit of the Source before it has received the ***PS_RDY*** Message for the new request.

Figure 7-38 Transition Diagram for increasing the Current in PPS mode

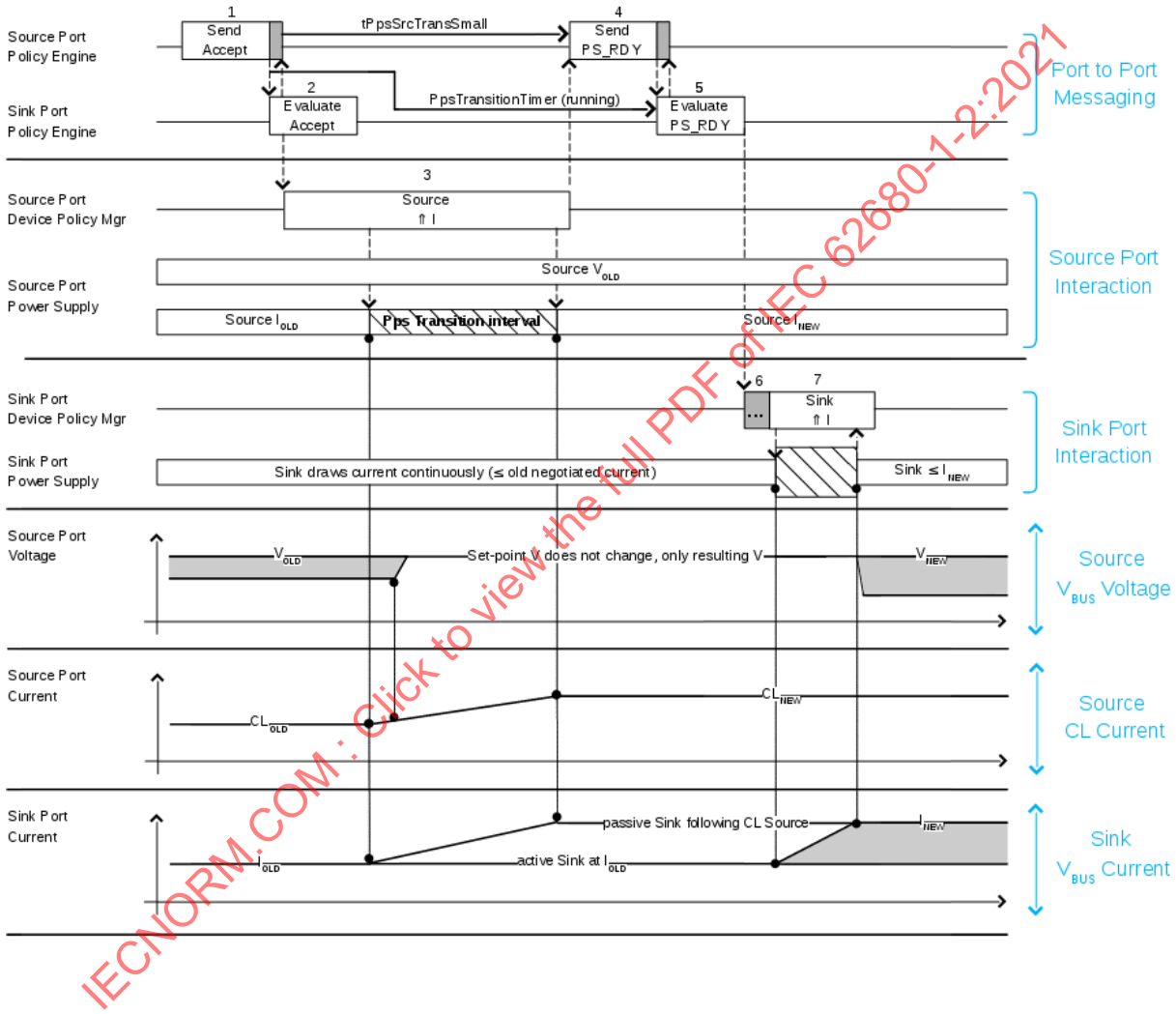


Table 7-19 Sequence Description for increasing the Current in PPS mode

Step	Source Port	Sink Port
1	Policy Engine sends the ***Accept*** Message to the Sink.	Policy Engine receives the ***Accept*** Message and starts the ***PSTransitionTimer***.
2	Protocol Layer receives the ***GoodCRC*** Message from the Sink. The Policy Engine tells the Device Policy Manager to instruct the power supply to increase its set-point for the current limit.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***Accept*** Message.
3	The Power Supply increases its Current Limit set-point to the new requested value.	The Sink draws current according to the increased Current Limit of the Source.
4	The Policy Engine waits ***tPpsSrcTransSmall*** then sends the ***PS_RDY*** Message to the Sink.	Policy Engine receives the ***PS_RDY*** Message.
5	Policy Engine receives the ***GoodCRC*** Message from the Sink.	Protocol Layer sends the ***GoodCRC*** Message to the Source.
6		Policy Engine evaluates the ***PS_RDY*** Message and tells the Device Policy Manager it can increase the current up to the requested value without the Source going into CL mode.
7		The Sink increases its current.

7.3.20 Decreasing the Programmable Power Supply Current

The interaction of the System Policy, Device Policy, and power supply that ***Shall*** be followed when decreasing the current limit in the same APDO, not exceeding the minimum for that APDO and without changing the requested voltage is shown in Figure 7-39. The sequence that ***Shall*** be followed is described in Table 7-20. 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 7-39 Transition Diagram for decreasing the Current in PPS mode

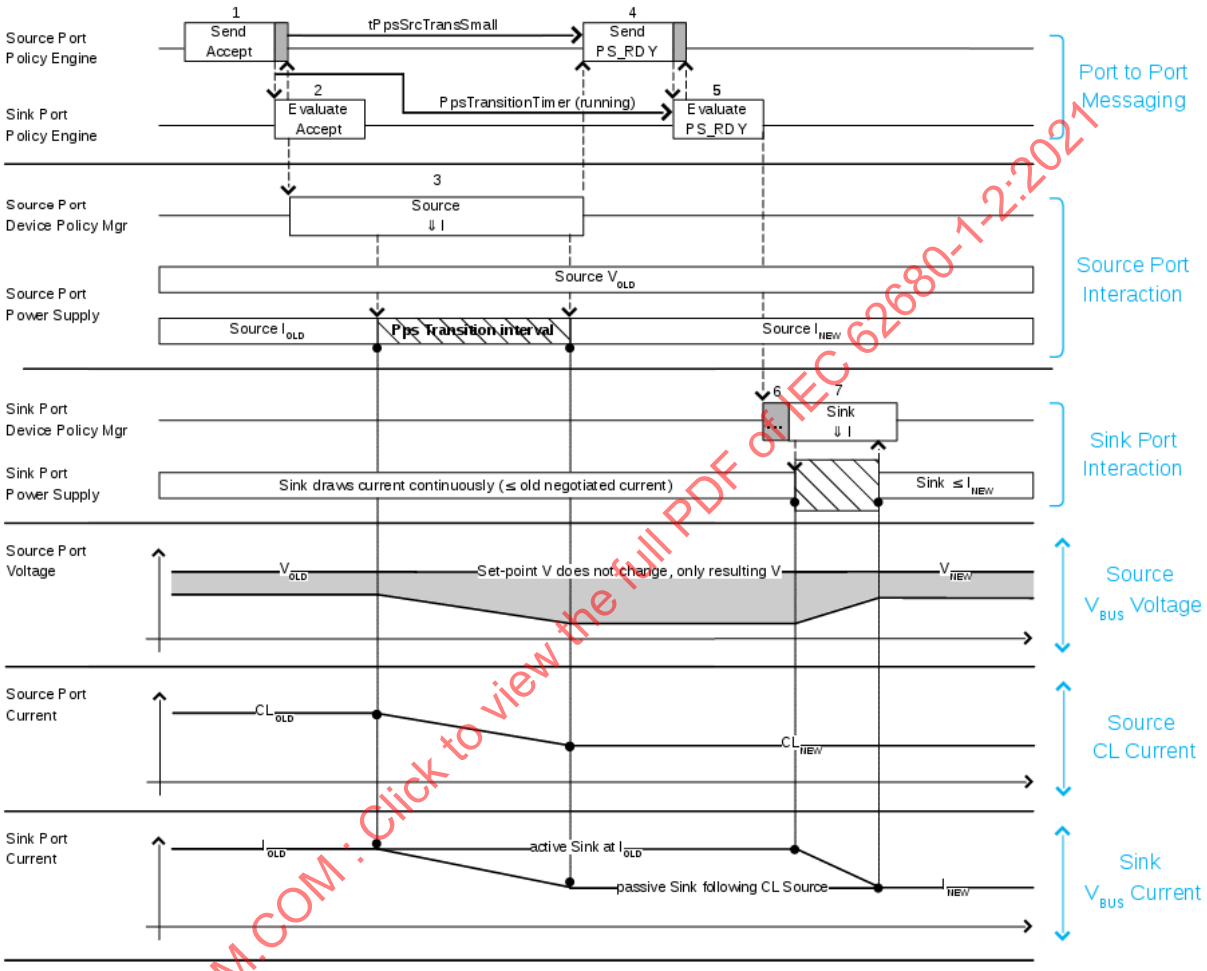


Table 7-20 Sequence Description for decreasing the Current in PPS mode

Step	Source Port	Sink Port
1	Policy Engine sends the ***Accept*** Message to the Sink.	Policy Engine receives the ***Accept*** Message and starts the PSTransitionTimer.
2	Protocol Layer receives the ***GoodCRC*** Message from the Sink. The Policy Engine tells the Device Policy Manager to instruct the power supply to decrease its set-point for the current limit.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***Accept*** Message and instructs the Sink to reduce its current to below the new negotiated current level.
3	The Power Supply decreases its Current Limit set-point to the new negotiated value.	The Sink reduces its current to less than the new negotiated current to prevent the Source from going into Current Limit.
4	The Policy Engine waits ***tPpsSrcTransSmall*** then sends the ***PS_RDY*** Message to the Sink.	
5	Policy Engine receives the ***GoodCRC*** Message from the Sink.	Policy Engine receives the ***PS_RDY*** Message.
6		Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine evaluates the ***PS_RDY*** Message.
7		The Sink is allowed to draw I_{NEW} but must be aware the voltage on V_{BUS} can drop doing so.

7.3.21 Same Request Programmable Power Supply

The interaction of the System Policy, Device Policy, and power supply that ***Shall*** be followed when the Sink requests the same voltage and current levels as the present negotiated levels for voltage and current is shown in Figure 7-40. 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. Note in this figure, the Sink has previously sent a ***Request*** Message to the Source.

Figure 7-40 Transition Diagram for no change in Current or Voltage in PPS mode

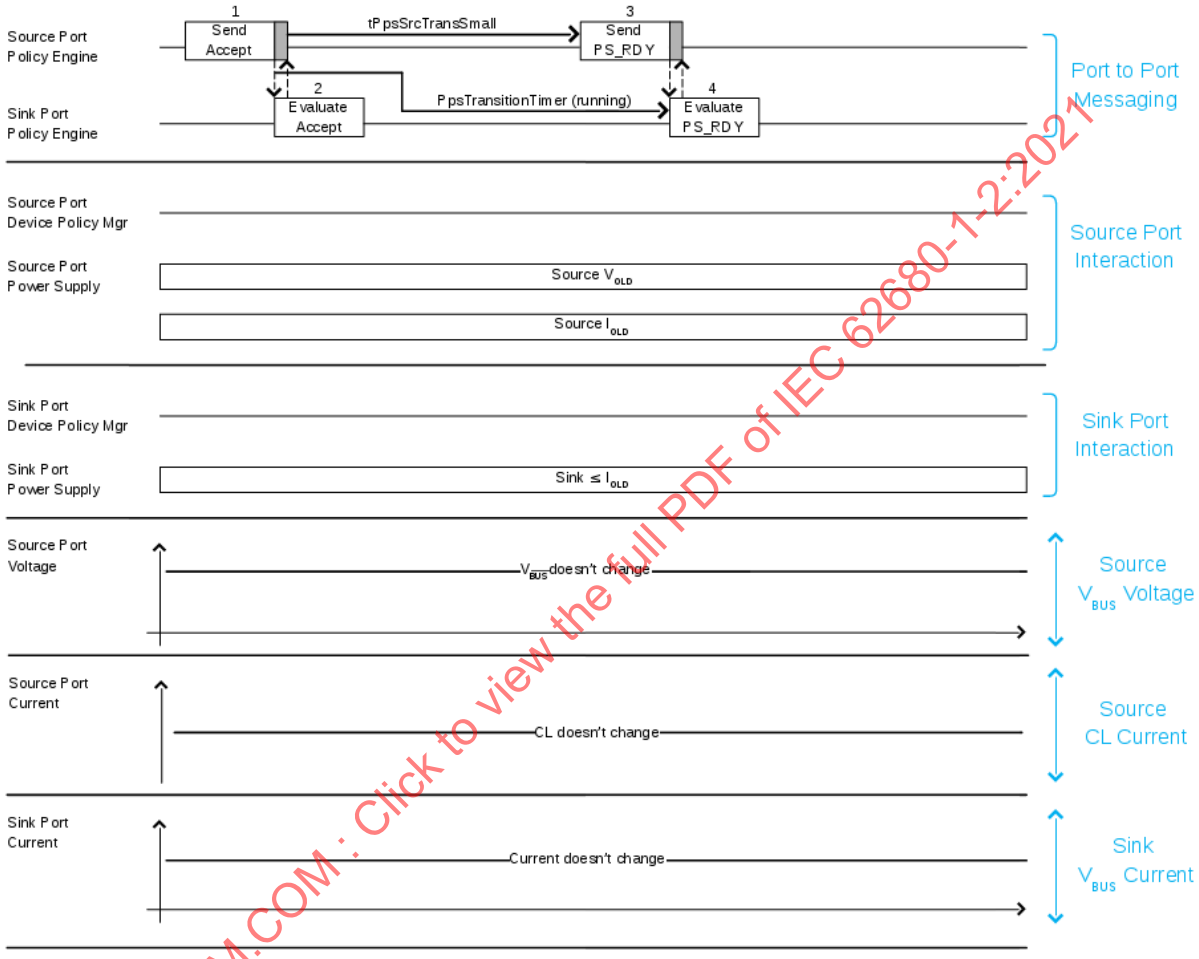


Table 7-21 Sequence Description for increasing the Current in PPS mode

Step	Source Port	Sink Port
1	Policy Engine sends the ***Accept*** Message to the Sink.	Policy Engine receives the ***Accept*** Message and starts the PSTransitionTimer.
2	Protocol Layer receives the ***GoodCRC*** Message from the Sink.	Protocol Layer sends the ***GoodCRC*** Message to the Source. Policy Engine then evaluates the ***Accept*** Message.
3	The Policy Engine then sends the ***PS_RDY*** Message to the Sink within ***tPpsSrcTransSmall***.	Policy Engine receives the ***PS_RDY*** Message.
4	Policy Engine receives the ***GoodCRC*** 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 ***GoodCRC*** Message to the Source. Policy Engine evaluates the ***PS_RDY*** 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.

Table 7-22 Source Electrical Parameters

Parameter	Description	MIN	TYP	MAX	UNITS	Reference
cSrcBulk[1]	Source bulk capacitance when a Port is powered from a dedicated supply.	10			µF	Section 7.1.2
cSrcBulkShared[1]	Source bulk capacitance when a Port is powered from a shared supply.	120			µF	Section 7.1.2
iPpsCLMin	Minimum Current Limit setting.	1			A	Section 7.1.4.4
iPpsCLNew	Current Limit accuracy					Section 7.1.4.4
	1A ≤ Operating Current ≤ 3A	-150		150	mA	
	Operating current > 3A	-5		5	%	
iPpsCLOperating	Total allowed change in Operating Current from point b in Figure 7-7 as the load resistance changes during Current Limit mode.	-25		100	mA	Figure 7-7
iPpsCLStep	PPS Current Limit programming step size.		50		mA	Section 7.1.4.4
iPpsCLTolerance	Allowable deviation of the operating current along the load line between the point b and e as shown in Figure 7-8.	-25		25	mA	Figure 7-8
iPpsCLTransient	Allowed output current overshoot when a load increase occurs while in CL mode.			New load + 100	mA	Section 7.1.4.4
	Allowed output current undershoot when a load decrease occurs while in CL mode.	New load – 100				
iPpsCVCLTransient	CV to CL transient current bounds assuming the Operating Voltage reduction of Section 7.2.3.1.	***iPpsCLNew*** - 100		New load + 500	mA	Section 7.1.4.4
tNewSnk	Time allowed for an initial Source in Swap Standby to transition new Sink operation.			15	ms	Figure 7-28, Figure 7-29
tPpsCLCVTransient	CL to CV transient voltage settling time.			25	ms	Section 7.1.4.4

Parameter	Description	MIN	TYP	MAX	UNITS	Reference
tPpsCLProgramSettle	PPS Current Limit programming settling time			250	ms	Section 7.1.4.4
tPpsCLSettle	CL load transient current settling time.			250	ms	Section 7.1.4.4
tPpsCVCLTransient	CV to CL transient settling time.			250	ms	Section 7.1.8.1
tPpsSrcTransLarge	The time the Programmable Power Supply's set-point ***Shall*** transition between requested voltages for steps larger than ***vPpsSmallStep***.	0		275	ms	Section 7.3.16 Section 7.3.17
tPpsSrcTransSmall	The time the Programmable Power Supply's set-point ***Shall*** transition between requested voltages for steps less than or equal to ***vPpsSmallStep***.	0		25	ms	Section 7.3.16 Section 7.3.17
tPpsTransient	The maximum time for the Programmable Power Supply to be between ***vPpsNew*** and ***vPpsValid*** in response to a load transient when target load is greater than or equal to 60mA.			5	ms	Section 7.1.8.1
	The maximum time for the Programmable Power Supply to be between ***vPpsNew*** and ***vPpsValid*** in response to a load transient when target load is less than 60mA.			150	ms	Section 7.1.8.1
tSrcFRSwap	Time from the initial Sink detecting that V_{BUS} has dropped below ***vSafe5V*** until the initial Sink/new Source is able to supply USB Type-C Current (see ***[USB Type-C 2.0]***)			150	µs	Section 7.1.13
tSrcReady	Time from positive/negative transition start (t0) to when the Source is ready to provide the newly negotiated power level.			285	ms	Figure 7-2, Figure 7-3
tSrcRecover	Time allotted for the Source to recover.	0.66		1	s	Section 7.1.5
tSrcSettle	Time from positive/negative transition start (t0) to when the transitioning voltage is within the range ***vSrcNew.***			275	ms	Figure 7-2

Parameter	Description	MIN	TYP	MAX	UNITS	Reference
tSrcSwapStdby	The maximum time for the Source to transition to Swap Standby.			650	ms	Table 7-9 Table 7-10
tSrcTransient	The maximum time for the Source output voltage to be between ***vSrcNew*** and ***vSrcValid*** in response to a load transient when target load is greater or equal to than 60mA.			5	ms	Section 7.1.8
	The maximum time for the Source output voltage to be between ***vSrcNew*** and ***vSrcValid*** in response to a load transient when target load is less than 60mA.			150	ms	Section 7.1.8
tSrcTransition	The time the Source ***Shall*** wait before transitioning the power supply to ensure that the Sink has sufficient time to prepare.	25		35	ms	Section 7.3
tSrcTurnOn	Transition time from ***vSafe0V*** to ***vSafe5V***.			275	ms	Table 7-12 Table 7-13
vPpsCLCVTransient	CL to CV load transient voltage bounds.	Operating Voltage * 0.95 - 0.1V		Operating Voltage * 1.05 + 0.1V	V	Section 7.1.4.4
vPpsCVCLTransient	CL to CF transient voltage bounds assuming the Operating Voltage reduction of Section 7.2.3.1.	Operating Voltage – 1.0V		Operating Voltage + 0.5V	V	Section 7.1.8.1
vPpsMaxVoltage	Maximum Voltage Field in the Programmable Power Supply APDO.	APDO Voltage *0.95		APDO Voltage * 1.05	V	Section 7.1.4.3
vPpsMinVoltage	Minimum Voltage Field in the Programmable Power Supply APDO.	APDO Voltage *0.95		APDO Voltage * 1.05	V	Section 7.1.4.3
vPpsNew	Programmable RDO Output Voltage measured at the Source receptacle.	RDO Output Voltage *0.95	RDO Output Voltage	RDO Output Voltage *1.05	V	Section 7.1.8.1
vPpsShutdown	The voltage at which the PPS shuts down when operating in CL.	APDO Minimum Voltage * 0.85		APDO Minimum Voltage * 0.95	V	Section 7.1.4.4
vPpsSlewNeg	Programmable Power Supply maximum slew rate for negative voltage changes			-30	mV/µs	Section 7.1.8.1
vPpsSlewPos	Programmable Power Supply maximum slew rate for positive voltage changes			30	mV/µs	Section 7.1.8.1

Parameter	Description	MIN	TYP	MAX	UNITS	Reference
vPpsSmallStep	PPS Step size defined as a small step relative to the previous ***vPpsNew***.	-500		500	mV	Section 7.1.4.3
vPpsStep	PPS voltage programming step size.		20		mV	Section 7.1.8.1
vPpsValid	The range in addition to ***vPpsNew*** which the Programmable Power Supply output is considered ***Valid*** in response to a load step.	-0.1		0.1	V	Section 7.1.8.1
vSrcNeg	Most negative voltage allowed during transition.			-0.3	V	Figure 7-10
vSrcNew	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	
vSrcPeak	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 6-10 Figure 7-12
vSrcSlewNeg	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 7.1.4.2 Figure 7-3
vSrcSlewPos	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
vSrcValid	The range in addition to ***vSrcNew*** which a newly negotiated voltage is considered ***Valid*** during and after a transition. This range also applies to ***vSafe5V***.	-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.

Table 7-23 Sink Electrical Parameters

Parameter	Description	MIN	TYP	MAX	UNITS	Reference
cSnkBulk[1]	Sink bulk capacitance on V_{BUS} at Attach and during FRS after the old Source stops sourcing and prior to establishing an Explicit Contract (see Appendix E for an example).	1		10	µF	Section 7.2.2
cSnkBulkPd	Bulk capacitance on V_{BUS} a Sink is allowed after a successful negotiation.	1		100	µF	Section 7.2.2
iLoadReleaseRate	Load release di/dt. Refer to ***[USB Type-C 2.0]*** Section 3.7.3.3.2 for cable details.	-150			mA/µs	Section 7.2.6
iLoadStepRate	Load step di/dt. Refer to ***[USB Type-C 2.0]*** Section 3.7.3.3.2 for cable details.			150	mA/µs	Section 7.2.6
iNewFrsSink	Maximum current the new Sink can draw during a Fast Role Swap until the new Source applies Rp. Matches the required USB Type-C Current field of the Fixed Supply PDO of the old Source's ***Sink_Capabilities*** Message.			Default USB current or 1.5 or 3.0	A	Section7.1.13
iOvershoot	Positive or negative overshoot when a load change occurs less than or equal to ***iLoadStepRate***; relative to the settled value after the load change. Refer to USB ***[USB Type-C 2.0]*** Section 3.7.3.3.2 for cable details.	-230		230	mA	Section 7.2.6
iPpsCLLoadRelease	Maximum load release decrease during Current Limit.	-500			mA	Section 7.2.3.1
iPpsCLLoadReleaseRate	Maximum load decrease slew rate during Current Limit.	-150			mA/µs	Section 7.2.3.1
iPpsCLLoadStep	Maximum load step increase during Current Limit.			500	mA	Section 7.2.3.1
iPpsCLLoadStepRate	Maximum load increase slew rate during Current Limit.			150	mA/µs	Section 7.2.3.1
iSafe0mA	Maximum current a Sink is allowed to draw when V_{BUS} is driven to ***vSafe0V***.			1.0	mA	Figure 7-31 Figure 7-32
iSnkSwapStdby	Maximum current a Sink can draw during Swap Standby. Ideally this current is very near to 0 mA largely			2.5	mA	Section 7.2.7

Parameter	Description	MIN	TYP	MAX	UNITS	Reference
	influenced by Port leakage current.					
pHubSusp	Suspend power consumption for a hub. 25mW + 25mW per downstream Port for up to 4 ports.			125	mW	Section 7.2.3
pSnkStdby	Maximum power consumption while in Sink Standby.			2.5	W	Section 7.2.3
pSnkSusp	Suspend power consumption for a peripheral device.			25	mW	Section 7.2.3
tNewSrc	Maximum time allowed for an initial Sink in Swap Standby to transition to new Source operation.			275	ms	Section 7.2.7 Table 7-9 Table 7-10
tSnkFRSwap	Time during a Fast Role Swap when the new Sink can draw no more than ***pSnkStdby***.			200	µs	Section 7.1.13
tSnkHardResetPrepare	Time allotted for the Sink power electronics to prepare for a Hard Reset.			15	ms	Table 7-13
tSnkNewPower	Maximum transition time between power levels.			15	ms	Section 7.2.3
tSnkRecover	Time for the Sink to resume USB Default Operation.			150	ms	Table 7-12
tSnkStdby	Time to transition to Sink Standby from Sink.			15	ms	Section 7.2.3
tSnkSwapStdby	Maximum time for the Sink to transition to Swap Standby.			15	ms	Section 7.2.7
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 3.2]*** 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
tSafe0V	Time to reach ***vSafe0V*** max.			650	ms	Section 7.1.5 Figure 7-10 Table 7-12 Table 7-13
tSafe5V	Time to reach ***vSafe5V*** max.			275	ms	Section 7.1.4.2 Figure 7-10
tVconnReapplied	• When the UFP is the VCONN source: time from the last bit of the ***GoodCRC*** acknowledging the ***PS_RDY*** Message before reapplying VCONN • When the DFP is the VCONN source: time from when VCONN drops below vRaReconnect.	10		20	ms	Figure 7-17 Figure 7-18
tVconnValid[1]	Time from ***tVconnReapplied*** until VCONN is within vVconnValid (see ***[USB Type-C 2.0]***).	0		5	ms	Figure 7-17 Figure 7-18
tVconnZero	Time from the last bit of the ***GoodCRC*** acknowledging the ***Accept*** Message in response to the ***Data_Reset*** Message until VCONN is below vRaReconnect (see ***[USB Type-C 2.0]***).			125	ms	Figure 7-17 Figure 7-18
vSafe0V	Safe operating voltage at “zero volts”.	0		0.8	V	Section 7.1.5
vSafe5V	Safe operating voltage at 5V. See ***[USB 2.0]*** and ***[USB 3.2]*** for allowable VBUS voltage range.	4.75		5.5	V	Section 7.1.5
Note 1: tVconnStable (See ***[USB Type-C 2.0]***) still applies.						

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 2.7.

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 USB-C Port Control 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 look at the capabilities offered by the Source. 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.

USB PD defines two types of power sources:

- Pre-defined voltage sources (Fixed, Variable and Battery)
- Programmable voltage source (PPS).

The first are generally used for classic charging wherein the charger electronics reside inside the Sink. The Device Policy Manager in the Sink requests a fixed voltage from the list of PDOs offered by the Source and which is converted internally to charge the Sink's battery and/or power its function.

The second moves the charger electronics that manage the voltage control outside the Sink and back into the Source itself. The Device Policy Manager in the Sink requests a specific voltage with a 20mV accuracy and sets a current limit. Unlike traditional USB where Sinks are responsible for limiting the current, they consume, the PPS Source limits the current to what the Sink has requested.

The process to request power is the same for both types of power Sources although the actual format and contents of the request are slightly different. The primary operational difference is that a Sink that is using PPS is required to periodically sent requests to let the Source know it is still alive and communicating. When this communication fails a Hard-Reset results.

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 2.7 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 relatively 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 USB-C Port Control 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 ***[USBTypeCBridge 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 an 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.

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 USB-C Port Control module and ***Shall*** be able to use the USB-C Port Control module to determine the Attachment status.

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 USB-C Port Control module and ***Shall*** be able to use the USB-C Port Control module to determine the Attachment status.

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 USB-C Port Control 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 USB-C Port Control 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 6.4.1.3.4). 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 6.4.1.3.4). 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 via 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.

Because the capability mismatch may not cause operational failure, the Provider's Device Policy Manager ***Should Not*** display a message to the user if the power offered to the Sink meets or exceeds the Sink Minimum PDP advertised in the ***Sink_Capabilities_Extended*** Message (see Section 6.5.13). If a message is displayed, it ***Should Not*** be shown as an error unless the power offered to the Sink is less than the Sink Minimum PDP advertised in the ***Sink_Capabilities_Extended*** Message.

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 ***[USBBridge 1.1]***. 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 6.4.1.2.2.3 and 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 ***[USBTypeCBridge 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

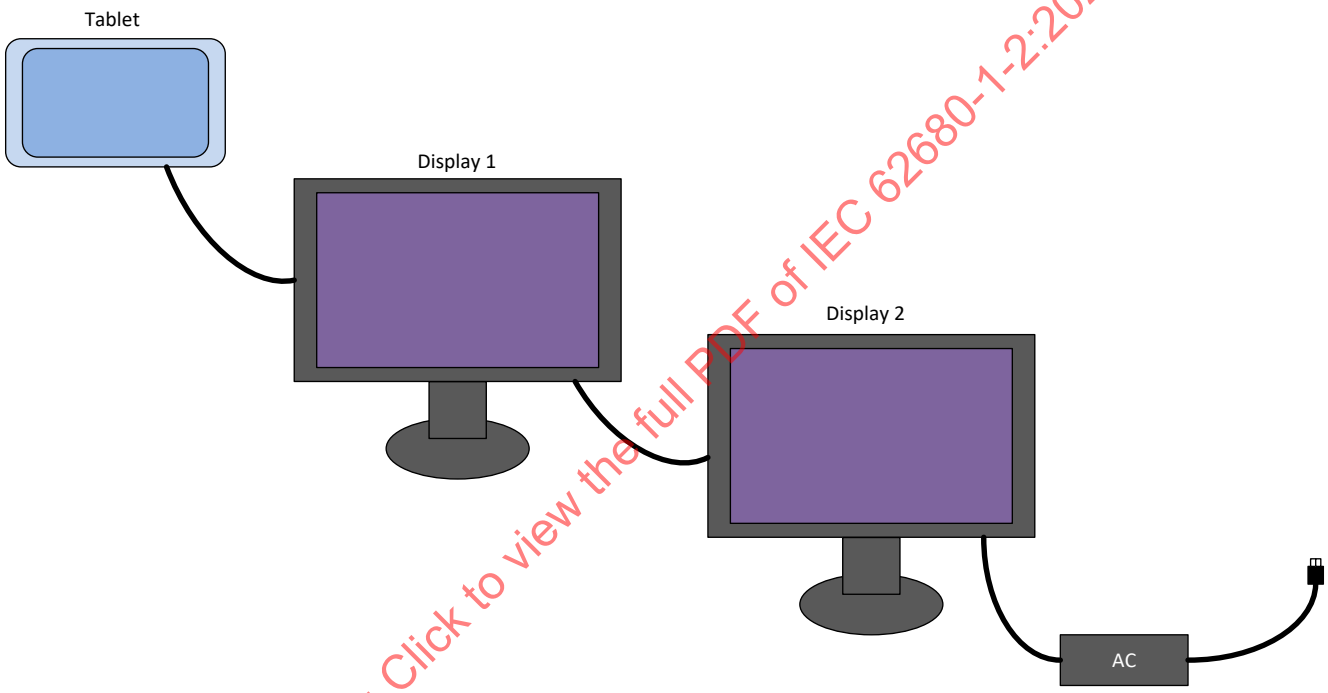
The Unconstrained Power bit provided by Sources and Sinks (see Section 6.4.1.2.2.3 and Section 6.4.1.3.1.3) notifies a connected device that it is acceptable to use the advertised power for charging as well as for what is needed for normal operation. A device that sets the Unconstrained Power bit has 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 6.4.1.2.2.3 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 Dual-Role Power Device

The Device Policy Manager in a Dual-Role Power Device ***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 Dual-Role Power Device Dead Battery handling

The Device Policy Manager in a Dual-Role Power Device with a Dead Battery ***Should***:

- Switch Ports to Sink-only or Sinking DFP operation to obtain power from the next Attached Source
- Use V_{BUS} from the Attached Source to power the USB Power Delivery communications as well as charging to enable the negotiation of higher input power.

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

8.3.2.1 Introduction

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 8.3.3.24).

In addition, the Cable Plug discovery sequence specified in Section 8.3.3.24.3 ***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 8.3.2.1.3 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.

8.3.2.1.1 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)

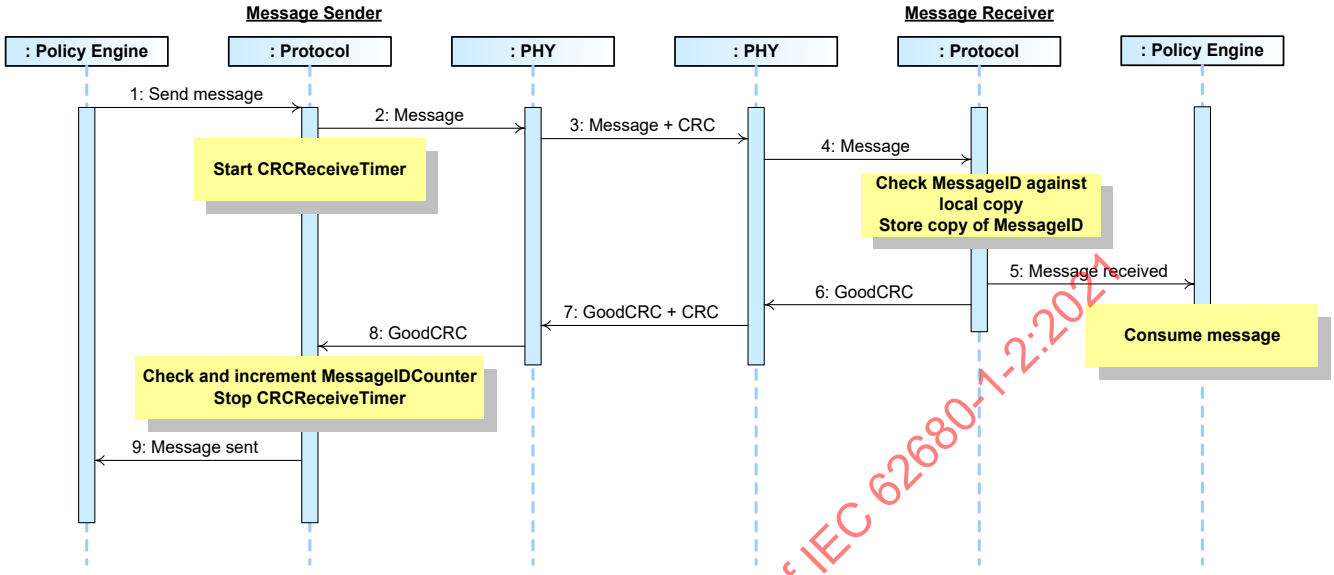


Table 8-1 Basic Message Flow

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 ***CRCReceiveTimer***.	
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 ***MessageID*** 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 ***GoodCRC*** 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 ***GoodCRC*** Message.
8	Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer. Protocol Layer checks and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***.	
9	Protocol Layer informs the Policy Engine that the Message was successfully sent.	

8.3.2.1.2 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

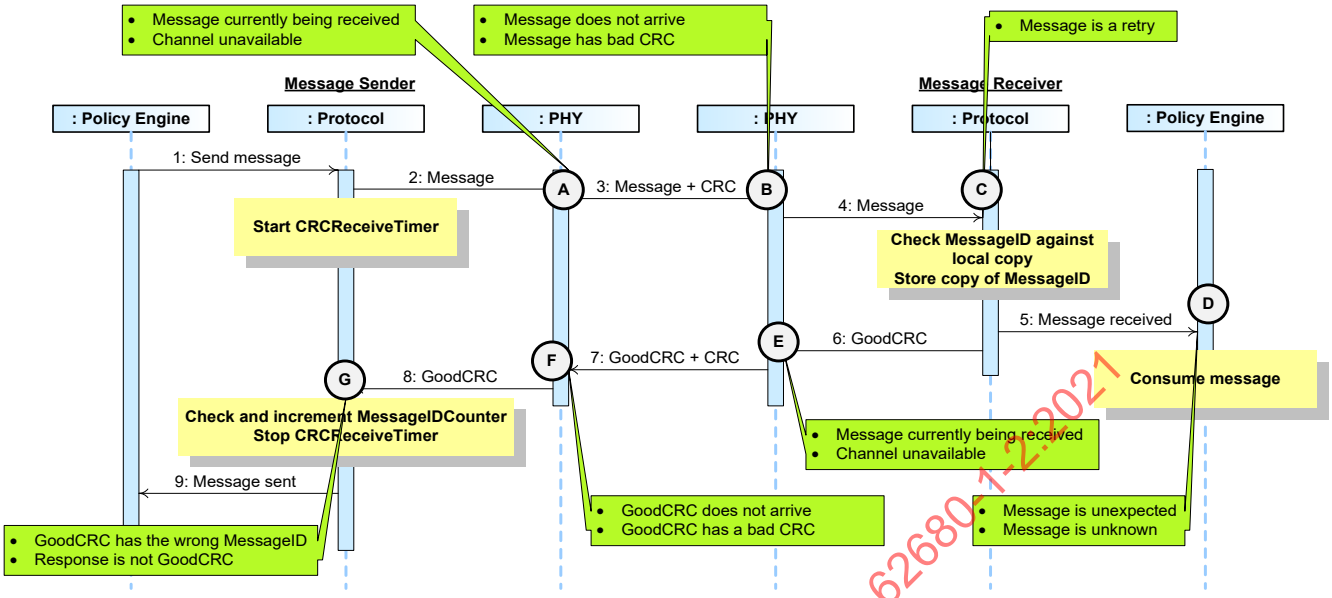


Table 8-2 Potential issues in Basic Message Flow

Point	Possible issues
A	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	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. 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.
C	1. ***MessageID*** of received Message matches stored ***MessageID*** so this is a retry. Message is not passed up to the Policy Engine.
D	1. Policy Engine receives a known Message that it was not expecting. 2. Policy Engine receives an Unrecognized Message. These cases are errors in the protocol which could lead to the generation of a ***Soft_Reset*** Message.
E	Same as point A but at the Message Receiver side.
F	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. A ***GoodCRC*** Message is not received by the Message Sender's Protocol Layer. This leads to a ***CRCReceiveTimer*** timeout in the Message Sender.
G	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). 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).

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

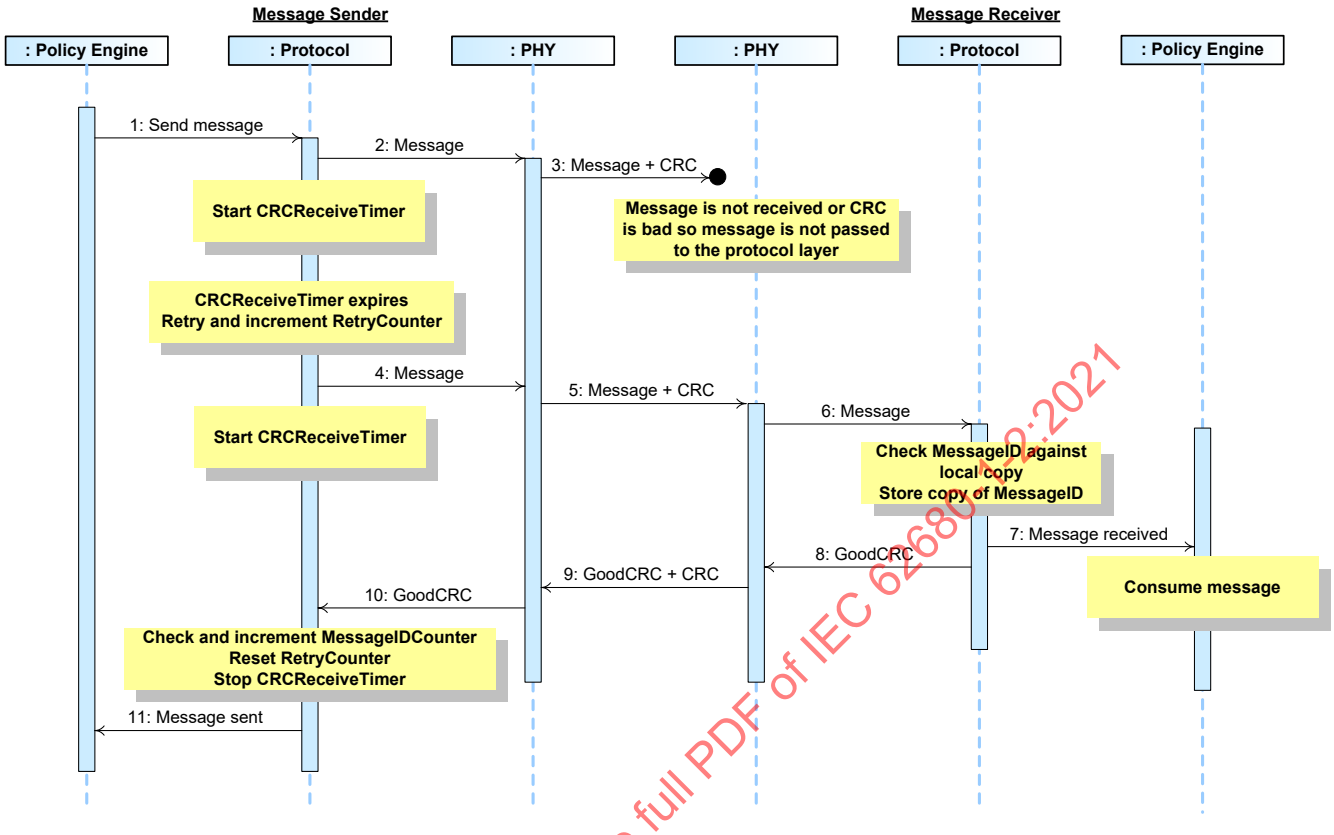


Table 8-3 Basic Message Flow with CRC failure

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 ***CRCReceiveTimer***.	
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 ***CRCReceiveTimer*** will expire and trigger the first retry by the Protocol Layer. The ***RetryCounter*** is incremented. Protocol Layer passes the Message to the Physical Layer. Starts ***CRCReceiveTimer***.	
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 ***MessageID*** 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 ***GoodCRC*** Message and passes it to the Physical Layer.
9	Physical Layer receives the Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the ***GoodCRC*** Message.
10	Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.	

Step	Message Sender	Message Receiver
11	Protocol Layer verifies the ***MessageID***, stops ***CRCReceiveTimer*** and resets the ***RetryCounter***. Protocol Layer informs the Policy Engine that the Message was successfully sent.	

8.3.2.1.3 Interruptible and Non-Interruptible Atomic Message Sequences

Table 8-4 details which AMS (as defined in Section 8.3.2) ***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, 8.3.3.3
GotoMin	No	Section 8.3.3.2, Section 8.3.3.3
Soft Reset	No	Section 8.3.3.4
Data Reset	No	Section 8.3.2.4
Hard Reset	No	Section 8.3.3.2, Section 8.3.3.3
Cable Reset	No	Section 8.3.3.24.2.3
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 8.3.3.18.3, Section 8.3.3.18.4
Fast Role Swap	No	Section 8.3.3.18.5, Section 8.3.3.18.6
Data Role Swap	No	Section 8.3.3.18.1, Section 8.3.3.18.2
VCONN Swap	No	Section 8.3.3.19
Source Alert	N/A	Section 8.3.3.8
Getting Source Extended Capabilities	No	Section 8.3.3.9
Getting Source/Sink Status	No	Section 8.3.3.10
Getting Battery Capabilities	No	Section 8.3.3.11
Getting Battery Status	No	Section 8.3.3.12
Getting Manufacturer Information	No	Section 8.3.3.13
Security	Yes	Section 8.3.3.14
Firmware Update	Yes	Section 8.3.3.17
Discover Identity	Yes	Section 8.3.3.20.1, Section 8.3.3.21.1
Source startup Cable Plug Discover Identity	Yes	Section 8.3.3.20.1, Section8.3.3.24.3
Discover SVIDs	Yes	Section 8.3.3.20.2, Section 8.3.3.21.2
Discover Modes	Yes	Section 8.3.3.20.3, Section 8.3.3.21.3
DFP to UFP Enter Mode	Yes	Section 8.3.3.22.1, Section 8.3.3.23.1
DFP to UFP Exit Mode	Yes	Section 8.3.3.22.2, Section 8.3.3.23.2
DFP to Cable Plug Enter Mode	Yes	Section 8.3.3.22.1, Section 8.3.3.24.4.1
DFP to Cable Plug Exit Mode	Yes	Section 8.3.3.22.1, Section 8.3.3.24.4.2
Attention	N/A	Section 8.3.3.20.4
Built in Self-Test (BIST)	No	Section 8.3.2.14
Sequence of Unstructured VDMs	Yes	Section 6.4.4.1
Sequence of Structured VDMs using Vendor Commands	Yes	Section 6.4.4.2
Country Info	Yes	Section 8.3.2.10.8
Enter USB	No	Section 8.3.2.15
Country Codes	Yes	Section 8.3.2.10.7

8.3.2.2 Power Negotiation

8.3.2.2.1 Explicit Contract Negotiation

Figure 8-5 illustrates an example of a successful Message flow while negotiating an Explicit Contract. 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.
- For PPS operation:
  - the Source starts its keep alive timer
  - the Sink starts its request timer to send periodic ***Request*** Messages

Figure 8-5 Successful Fixed, Variable or Battery Power Negotiation

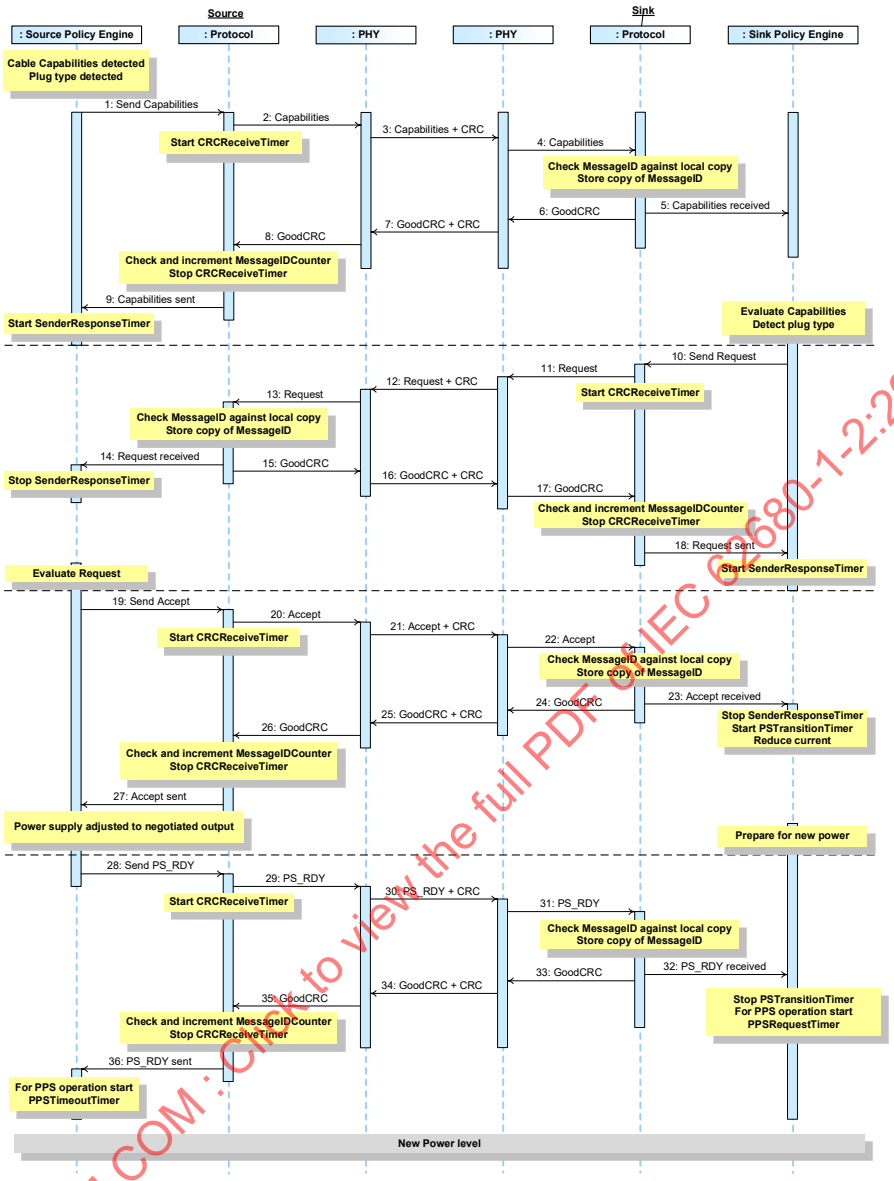


Table 8-5 below provides a detailed explanation of what happens at each labeled step in Figure 8-5 above.

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 4.4). Policy Engine directs the Protocol Layer to send a ***Source_Capabilities*** Message that represents the power supply's present capabilities.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	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.
4		Physical Layer removes the CRC and forwards the ***Source_Capabilities*** Message to the Protocol Layer.

Step	Source	Sink
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 ***Source_Capabilities*** 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 ***Source_Capabilities*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine evaluates the ***Source_Capabilities*** Message sent by the Source, detects the plug type if this is necessary (see Section 4.4) and selects which power it would like. It tells the Protocol Layer to form the data (e.g. Power Data Object) that represents its Request into a Message.
11		Protocol Layer creates the ***Request*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
12	Physical Layer receives the ***Request*** Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the ***Request*** Message.
13	Physical Layer removes the CRC and forwards the ***Request*** Message to the Protocol Layer.	
14	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 passes the Request information to the Policy Engine. Policy Engine stops ***SenderResponseTimer***.	
15	The Protocol Layer generates a ***GoodCRC*** 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.
17		Physical Layer forwards the ***GoodCRC*** Message to the Protocol Layer.
18		The protocol Layer verifies and increments the ***MessageIDCounter***. It informs the Policy Engine that the ***Request*** Message was successfully sent. The Protocol Layer stops the ***CRCReceiveTimer***. The Policy Engine starts ***SenderResponseTimer***.
19	Policy Engine evaluates the ***Request*** Message sent by the Sink and decides if it can meet the request. It tells the Protocol Layer to form an ***Accept*** Message.	
20	The Protocol Layer forms the ***Accept*** Message that is passed to the Physical Layer and starts the ***CRCReceiveTimer***.	
21	Physical Layer appends CRC and sends the ***Accept*** Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.

Step	Source	Sink
22		Physical Layer forwards the ***Accept*** Message to the Protocol Layer.
23		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. Protocol Layer informs the Policy Engine that an ***Accept*** Message has been received. The Policy Engine stops ***SenderResponseTimer***, starts the ***PSTransitionTimer*** 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 ***GoodCRC*** 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 ***GoodCRC*** Message to the Protocol Layer. The Protocol Layer verifies and increments the ***MessageIDCounter*** and stops the ***CRCReceiveTimer***.	
27	The Protocol Layer informs the Policy Engine that an ***Accept*** 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 ***PS_RDY*** Message.	
29	The Protocol Layer forms the ***PS_RDY*** Message and starts the ***CRCReceiveTimer***.	
30	Physical Layer appends CRC and sends the ***PS_RDY*** Message.	Physical Layer receives the ***PS_RDY*** Message and compares the CRC it calculated with the one sent to verify the Message.
31		Physical Layer forwards the ***PS_RDY*** Message to the Protocol Layer.
32		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. Protocol Layer informs the Policy Engine that a RS_RDY has been received. The Policy Engine stops the ***PSTransitionTimer***. When in PPS operation the Policy Engine starts the ***SinkPPSPeriodicTimer***.
33		The Protocol Layer generates a ***GoodCRC*** 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 ***GoodCRC*** Message to the Protocol Layer. The Protocol Layer verifies and increments the ***MessageIDCounter***. Stops the ***CRCReceiveTimer***.	
36	The Protocol Layer informs the Policy Engine that the ***PS_RDY*** Message was successfully sent.	
37	When in PPS operation the Policy Engine starts the ***SourcePPSCommTimer***.	

8.3.2.2.2 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

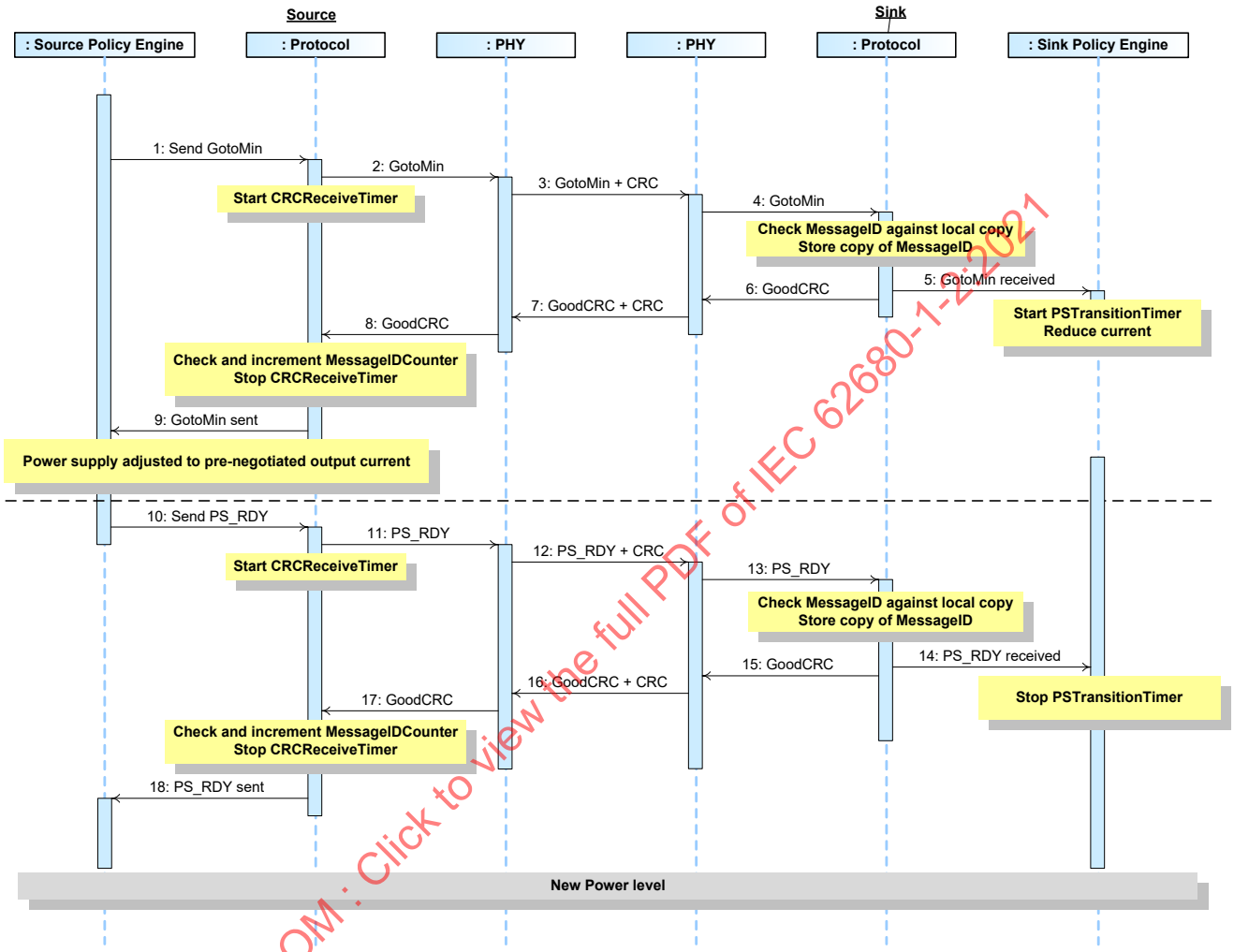


Table 8-6 provides a detailed explanation of what happens at each labeled step in Figure 8-6 above.

Table 8-6 Steps for a GotoMin Negotiation

Step	Source	Sink
1	Policy Engine tells the Protocol Layer to form a ***GotoMin*** Message.	
2	The Protocol Layer forms the ***GotoMin*** Message that is passed to the Physical Layer and starts the ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***GotoMin*** 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 ***GotoMin*** Message to the Protocol Layer.

Step	Source	Sink
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. Protocol Layer informs the Policy Engine that a ***GotoMin*** Message has been received. The Policy starts the ***PSTransitionTimer*** 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 ***GoodCRC*** 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 ***GoodCRC*** Message to the Protocol Layer. The Protocol Layer verifies and increments the ***MessageIDCounter*** and stops the ***CRCReceiveTimer***.	
9	The Protocol Layer informs the Policy Engine that a ***GotoMin*** 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 ***PS_RDY*** Message.	
11	The Protocol Layer forms the ***PS_RDY*** Message and starts the ***CRCReceiveTimer***.	
12	Physical Layer appends CRC and sends the ***PS_RDY*** 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 ***PS_RDY*** Message to the Protocol Layer.
14		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. Protocol Layer informs the Policy Engine that a ***PS_RDY*** Message has been received. The Policy Engine stops the ***PSTransitionTimer***.
15		The Protocol Layer generates a ***GoodCRC*** 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 ***GoodCRC*** Message to the Protocol Layer. The Protocol Layer verifies and increments the ***MessageIDCounter*** and stops the ***CRCReceiveTimer***.	
18	The Protocol Layer informs the Policy Engine that the ***PS_RDY*** Message was successfully sent.	

8.3.2.2.3 PPS Keep Alive

This is an example of PPS keep alive operation during an Explicit Contract with PPS as the APDO. Figure 8-7 shows the Messages as they flow across the bus and within the devices to accomplish the keep alive.

Figure 8-7 PPS Keep Alive

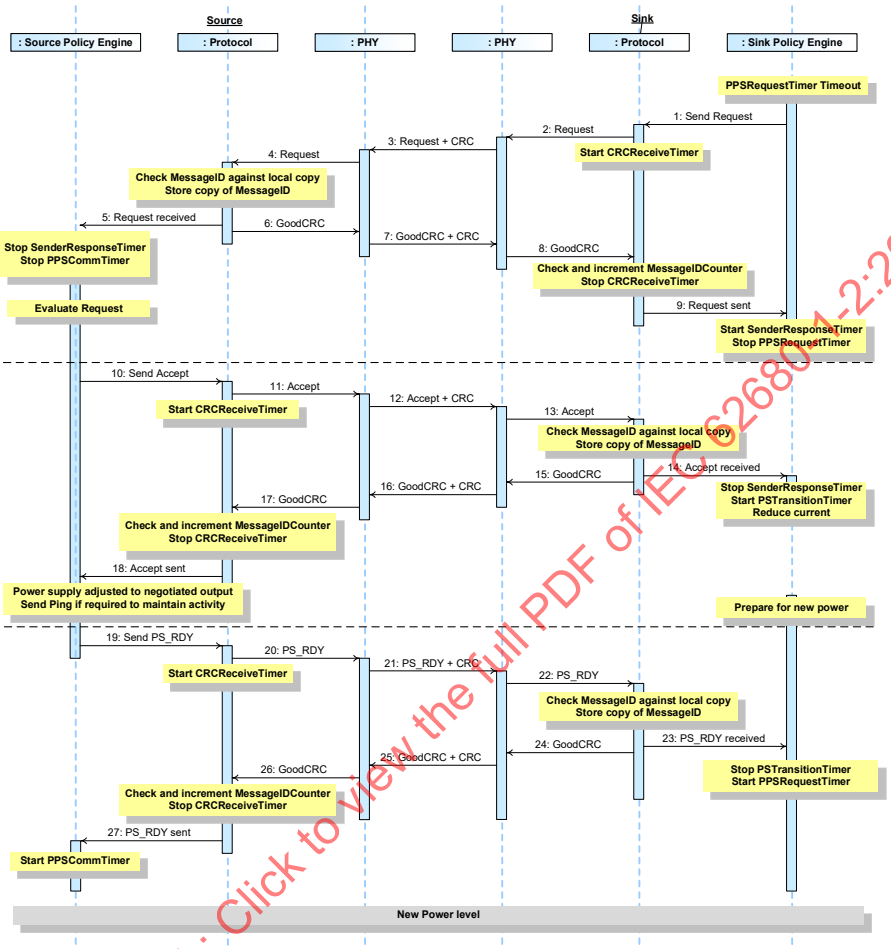


Table 8-7 below provides a detailed explanation of what happens at each labeled step in Figure 8-5 above.

Table 8-7 Steps for a successful Power Negotiation

Step	Source	Sink
1		The ***SinkPPSPeriodicTimer*** times out in the Policy Engine. The Policy Engine tells the Protocol Layer to form a ***Request*** Message. The Protocol Layer creates the ***Request*** Message and passes it to Physical Layer. The Protocol Layer starts the ***CRCReceiveTimer***.
2	Physical Layer receives the ***Request*** Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the ***Request*** Message.
3	Physical Layer removes the CRC and forwards the ***Request*** Message to the Protocol Layer.	

Step	Source	Sink
4	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 passes the Request information to the Policy Engine. Policy Engine stops the ***SourcePPSCommTimer.***	
5	The Protocol Layer generates a ***GoodCRC*** Message and passes it to its Physical Layer.	
6	Physical Layer appends CRC and sends the ***GoodCRC*** Message.	Physical Layer receives the ***GoodCRC*** Message and compares the CRC it calculated with the one sent to verify the Message.
7		Physical Layer forwards the ***GoodCRC*** Message to the Protocol Layer.
8		The protocol Layer verifies and increments the ***MessageIDCounter***. It informs the Policy Engine that the ***Request*** Message was successfully sent. The Protocol Layer stops the ***CRCReceiveTimer***. The Policy Engine starts ***SenderResponseTimer***.
9	Policy Engine requests the Device Policy Manager to evaluate the ***Request*** Message sent by the Sink and decides if the Source can meet the request. The Policy Engine tells the Protocol Layer to form an ***Accept*** Message.	
10	The Protocol Layer forms the ***Accept*** Message that is passed to the Physical Layer and starts the ***CRCReceiveTimer***.	
11	Physical Layer appends CRC and sends the ***Accept*** Message.	Physical Layer receives the ***Accept*** Message and compares the CRC it calculated with the one sent to verify the Message.
12		Physical Layer forwards the ***Accept*** Message to the Protocol Layer.
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. Protocol Layer informs the Policy Engine that an ***Accept*** Message has been received. The Policy Engine stops ***SenderResponseTimer***, starts the ***PSTransitionTimer*** and reduces its current draw. The Device Policy Manager prepares the Power supply for transition to the new power level.
14		The Protocol Layer generates a ***GoodCRC*** Message and passes it to its Physical Layer.
15	Physical Layer receives the ***GoodCRC*** Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the ***GoodCRC*** Message.
16	Physical Layer forwards the ***GoodCRC*** Message to the Protocol Layer. The Protocol Layer verifies and increments the ***MessageIDCounter*** and stops the ***CRCReceiveTimer***.	
17	The Protocol Layer informs the Policy Engine that an ***Accept*** Message was successfully sent.	
power supply Adjusts its Output to the Negotiated Value		
18	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 ***PS_RDY*** Message.	
19	The Protocol Layer forms the ***PS_RDY*** Message and starts the ***CRCReceiveTimer***.	

Step	Source	Sink
20	Physical Layer appends CRC and sends the ***PS_RDY*** Message.	Physical Layer receives the ***PS_RDY*** Message and compares the CRC it calculated with the one sent to verify the Message.
21		Physical Layer forwards the ***PS_RDY*** Message to the Protocol Layer.
22		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. Protocol Layer informs the Policy Engine that a RS_RDY has been received. The Policy Engine stops the ***PSTransitionTimer***. When in PPS operation the Policy Engine starts the ***SinkPPSPeriodicTimer***.
23		The Protocol Layer generates a ***GoodCRC*** Message and passes it to its Physical Layer.
24	Physical Layer receives the ***GoodCRC*** Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the ***GoodCRC*** Message.
25	Physical Layer forwards the ***GoodCRC*** Message to the Protocol Layer. The Protocol Layer verifies and increments the ***MessageIDCounter***. Stops the ***CRCReceiveTimer***.	
26	The Protocol Layer informs the Policy Engine that the ***PS_RDY*** Message was successfully sent.	
27	When in PPS operation the Policy Engine starts the ***SourcePPSCommTimer***.	

8.3.2.3 Soft Reset

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

Figure 8-8 Soft Reset

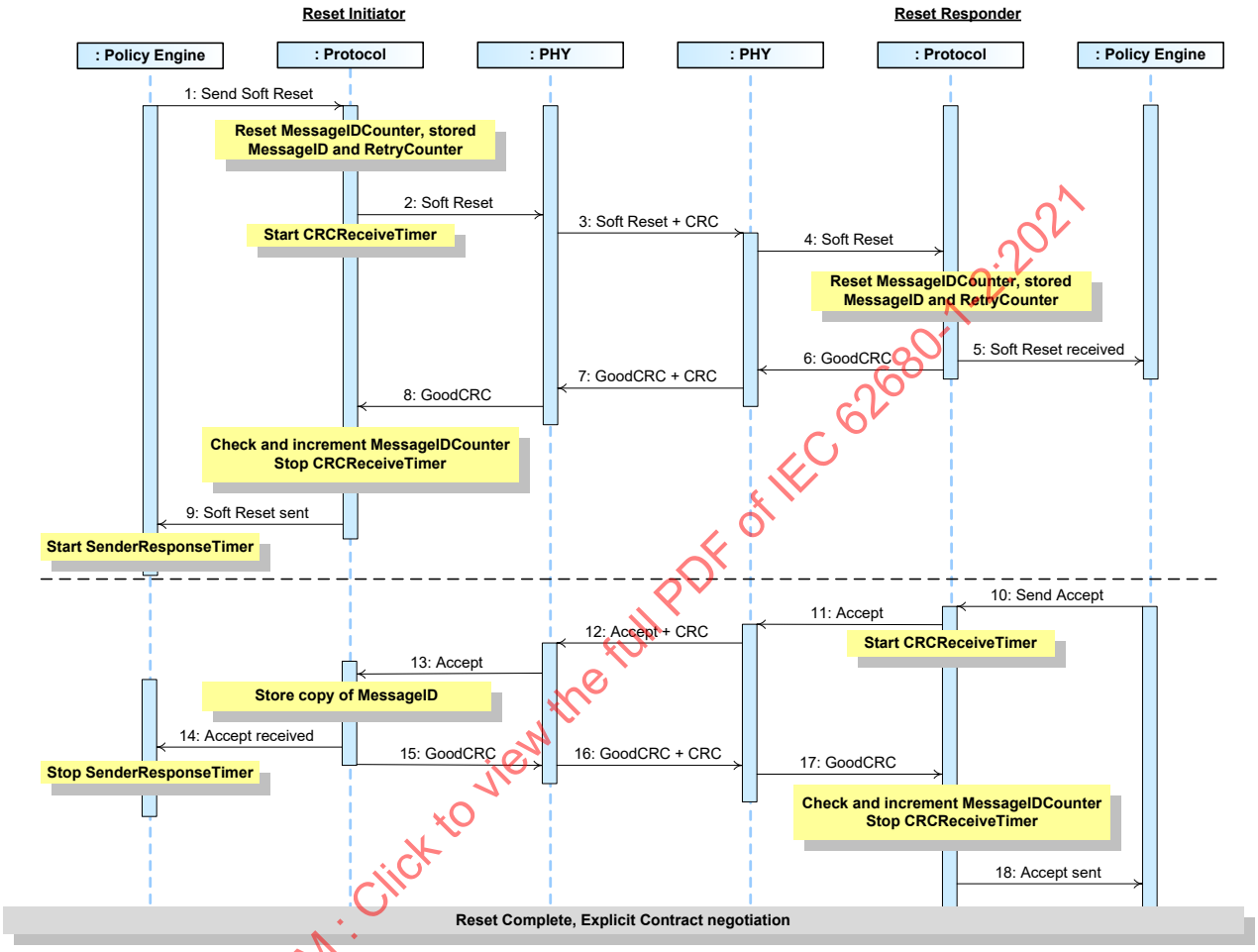


Table 8-8 below provides a detailed explanation of what happens at each labeled step in Figure 8-8 above.

Table 8-8 Steps for a Soft Reset

Step	Reset Initiator	Reset Responder
1	The Policy Engine directs the Protocol Layer to generate a ***Soft_Reset*** Message to request a Soft Reset.	
2	Protocol Layer resets ***MessageIDCounter***, stored ***MessageID*** and ***RetryCounter***. Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Soft_Reset*** Message.	Physical Layer receives the ***Soft_Reset*** Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Soft_Reset*** Message to the Protocol Layer.

Step	Reset Initiator	Reset Responder
5		Protocol Layer does not check the ***MessageID*** in the incoming Message and resets ***MessageIDCounter***, stored ***MessageID*** and ***RetryCounter***. The Protocol Layer forwards the received ***Soft_Reset*** 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 ***Soft_Reset*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine 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 Message.	Physical Layer appends a CRC and sends the Message.
13	Protocol Layer stores the ***MessageID*** of the incoming Message.	
14	The Protocol Layer forwards the received ***Accept*** Message information to the Policy Engine that consumes it.	
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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Accept*** 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.	

8.3.2.4 Data Reset

8.3.2.4.1 DFP Initiated Data Reset where the DFP is the VCONN Source

This is an example of a Data Reset operation where the DFP is also the VCONN Source and initiates a Data Reset. Figure 8-9 shows the Messages as they flow across the bus and within the devices to accomplish the Data Reset.

Figure 8-9 DFP Initiated Data Reset where the DFP is the Vconn Source

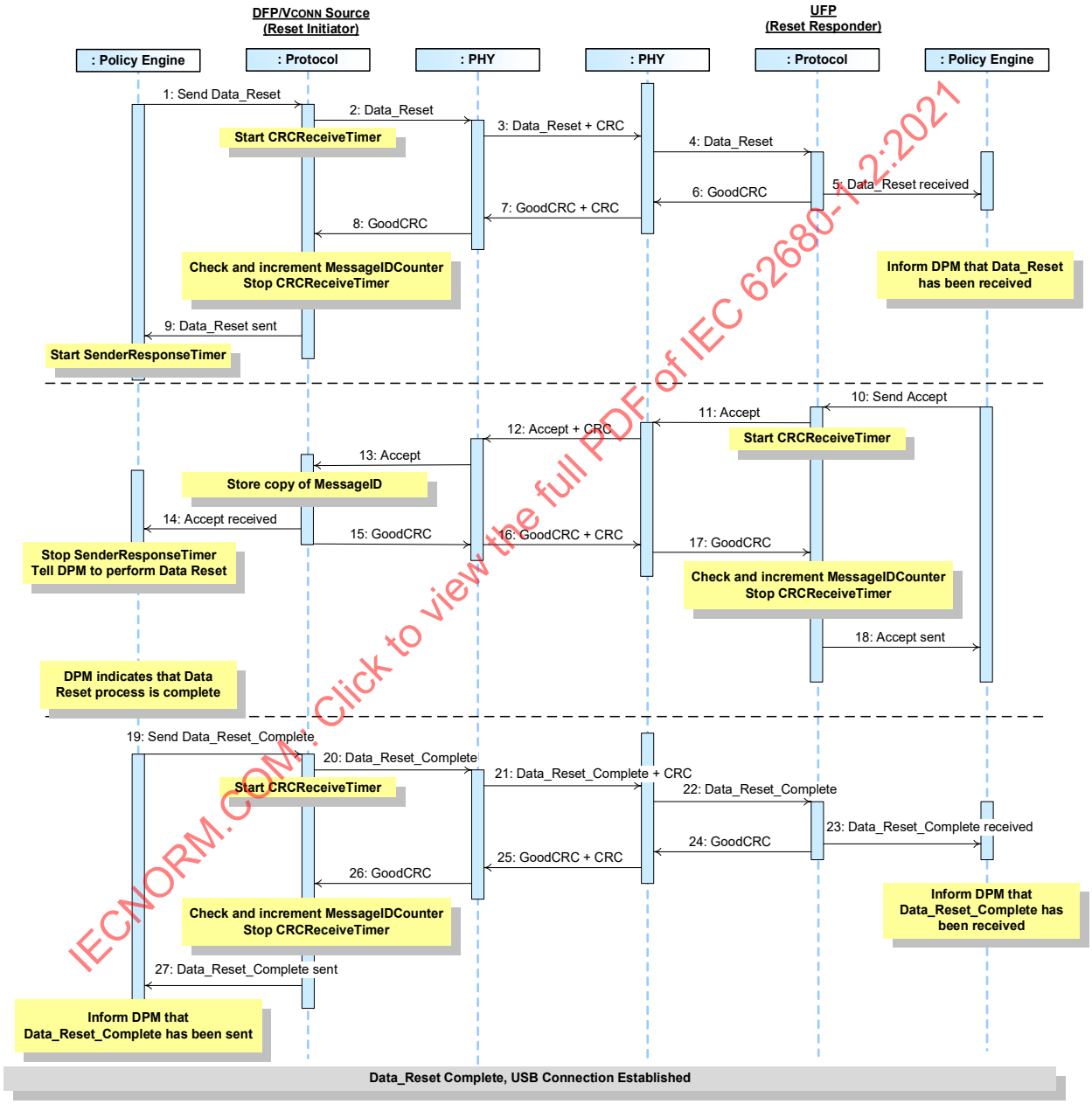


Table 8-9 below provides a detailed explanation of what happens at each labeled step in Figure 8-9 above.

Table 8-9 Steps for a DFP Initiated Data Reset where the DFP is the Vconn Source

Step	DFP/VCONN Source (Reset Initiator)	UFP (Reset Responder)
1	The Policy Engine directs the Protocol Layer to generate a ***Data_Reset*** Message to request a Data Reset.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Data_Reset*** Message.	Physical Layer receives the ***Data_Reset*** Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Data_Reset*** 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 ***Data_Reset*** Message information to the Policy Engine that consumes it. The Policy Engine informs the Device Policy Manager that a ***Data_Reset*** Message has been received.
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 ***Data_Reset*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine 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 Message.	Physical Layer appends a CRC and sends the Message.
13	Protocol Layer stores the ***MessageID*** of the incoming Message.	
14	The Protocol Layer forwards the received ***Accept*** Message information to the Policy Engine that consumes it. The Policy Engine stops the ***SenderResponseTimer*** and tells the Device Policy Manager to perform a Data Reset. The Device Policy Manager proceeds to cycle VCONN and then reset the data connection.	
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	DFP/VCONN Source (Reset Initiator)	UFP (Reset Responder)
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Accept*** Message was successfully sent.
19	The Device Policy Manager indicates that the Data Reset process is complete. The Policy Engine directs the Protocol Layer to generate a ***Data_Reset_Complete*** Message.	
20	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
21	Physical Layer appends CRC and sends the ***Data_Reset_Complete*** Message.	Physical Layer receives the ***Data_Reset_Complete*** Message and compares the CRC it calculated with the one sent to verify the Message.
22		Physical Layer removes the CRC and forwards the ***Data_Reset_Complete*** Message to the Protocol Layer.
23		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 ***Data_Reset_Complete*** Message information to the Policy Engine that consumes it. The Policy Engine informs the Device Policy Manager that a ***Data_Reset_Complete*** Message has been received.
24		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.
25	Physical Layer receives the ***GoodCRC*** and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the ***GoodCRC*** Message.
26	Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.	
27	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Data_Reset_Complete*** Message was successfully sent. The Policy Engine informs the Device Policy Manager that the ***Data_Reset_Complete*** Message was successfully sent.	
The data reset is complete as defined in Section 6.3.14 Step 5. Port Partners re-establish a USB data connection.		

8.3.2.4.2 DFP Receives Data Reset where the DFP is the VCONN Source

This is an example of a Data Reset operation where the DFP receives a Data Reset Message and is the VCONN Source. Figure 8-10 shows the Messages as they flow across the bus and within the devices to accomplish the Data Reset.

Figure 8-10 DFP Receives Data Reset where the DFP is the Vconn Source

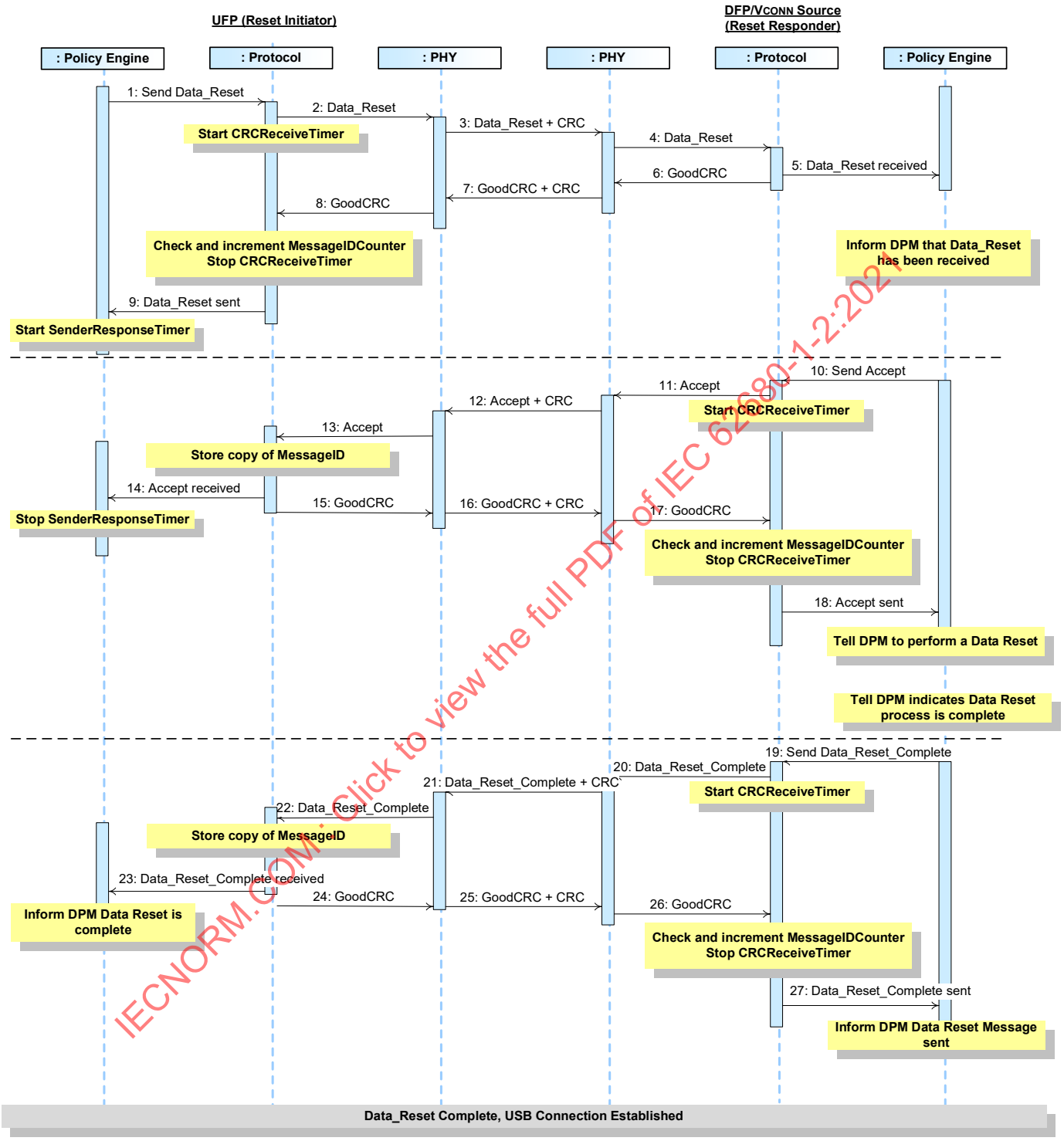


Figure 8-10 below provides a detailed explanation of what happens at each labeled step in Figure 8-10 above.

Table 8-10 Steps for a DFP Receiving a Data Reset where the DFP is the Vconn Source

Step	UFP (Reset Initiator)	DFP/VCONN Source (Reset Responder)
1	The Policy Engine directs the Protocol Layer to generate a ***Data_Reset*** Message to request a Data Reset.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Data_Reset*** Message.	Physical Layer receives the ***Data_Reset*** Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Data_Reset*** 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 ***Data_Reset*** Message information to the Policy Engine that consumes it. The Policy Engine informs the Device Policy Manager that a ***Data_Reset*** Message has been received.
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 ***Data_Reset*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine 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 Message.	Physical Layer appends a CRC and sends the Message.
13	Protocol Layer stores the ***MessageID*** of the incoming Message.	
14	The Protocol Layer forwards the received ***Accept*** Message information to the Policy Engine that consumes it. The Policy Engine stops the ***SenderResponseTimer***. The Device Policy Manager proceeds to cycle VCONN and then reset the data connection.	
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	UFP (Reset Initiator)	DFP/VCONN Source (Reset Responder)
18		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 tells the Device Policy Manager to perform a Data Reset.
19		The Device Policy Manager indicates that the Data Reset process is complete. The Policy Engine directs the Protocol Layer to generate a ***Data_Reset_Complete*** Message.
20		Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
21	Physical Layer receives the ***Data_Reset_Complete*** Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the ***Data_Reset_Complete*** Message.
22	Physical Layer removes the CRC and forwards the ***Data_Reset_Complete*** Message to the Protocol Layer.	
23	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 ***Data_Reset_Complete*** Message information to the Policy Engine that consumes it. The Policy Engine informs the Device Policy Manager that a ***Data_Reset_Complete*** Message has been received.	
24	Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.	
25	Physical Layer appends CRC and sends the ***GoodCRC*** Message.	Physical Layer receives the ***GoodCRC*** and checks the CRC to verify the Message.
26		Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.
27		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Data_Reset_Complete*** Message was successfully sent. The Policy Engine informs the Device Policy Manager that the ***Data_Reset_Complete*** Message was successfully sent.
	The reset is complete as defined in Section 6.3.14 Step 5. Port Partners re-establish a USB data connection.	

8.3.2.4.3 DFP Initiated Data Reset where the UFP is the VCONN Source

This is an example of a Data Reset operation where the DFP initiates a Data Reset and the UFP is the VCONN Source. Figure 8-11 shows the Messages as they flow across the bus and within the devices to accomplish the Data Reset.

Figure 8-11 DFP Initiated Data Reset where the UFP is the Vconn Source

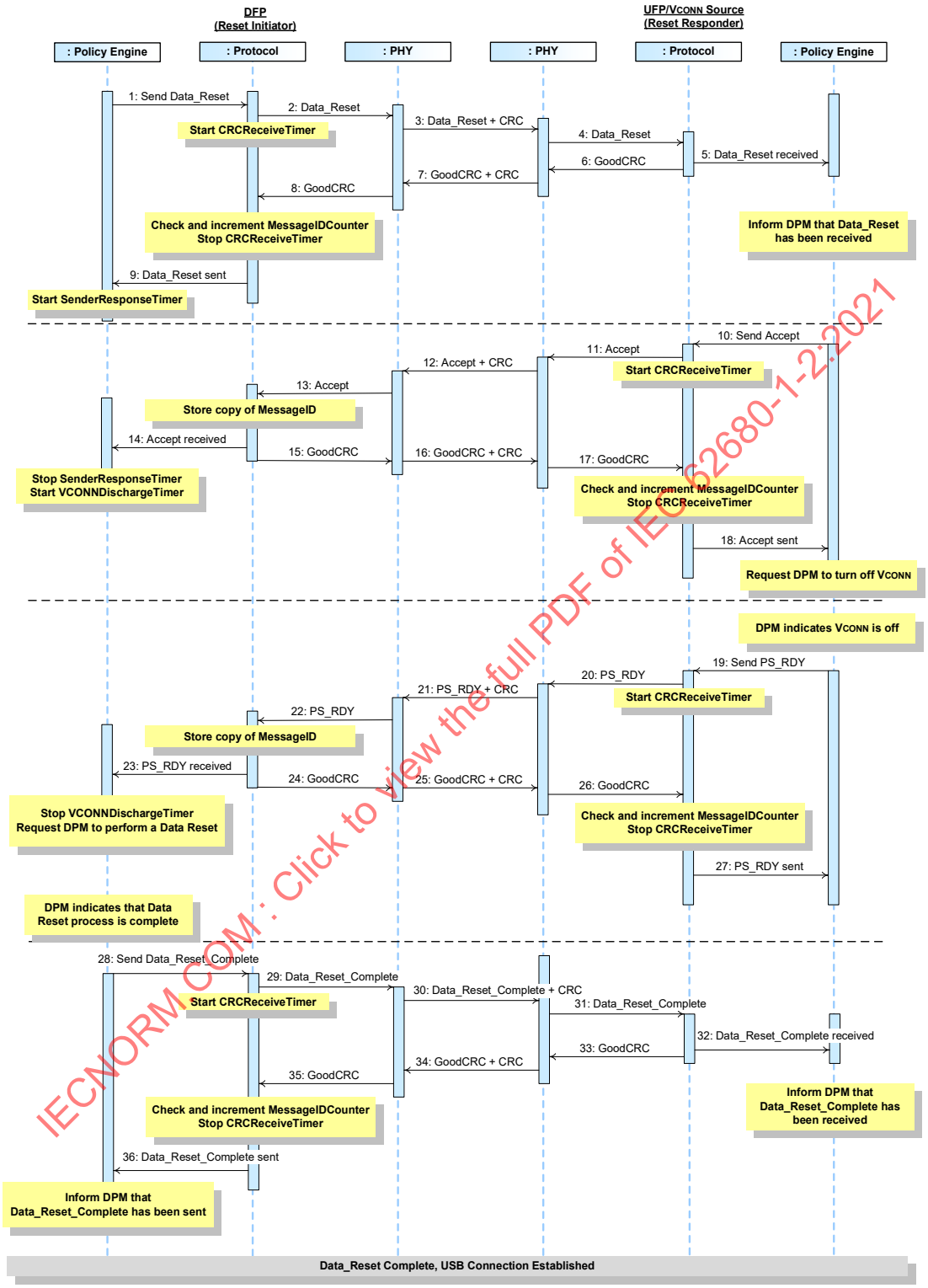


Table 8-11 Steps for a DFP Initiated Data Reset where the UFP is the Vconn Source below provides a detailed explanation of what happens at each labeled step in Figure 8-11 above.

Table 8-11 Steps for a DFP Initiated Data Reset where the UFP is the Vconn Source

Step	DFP (Reset Initiator)	UFP/VCONN Source (Reset Responder)
1	The Policy Engine directs the Protocol Layer to generate a ***Data_Reset*** Message to request a Soft Reset.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Data_Reset*** Message.	Physical Layer receives the ***Data_Reset*** Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Data_Reset*** 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 ***Data_Reset*** Message information to the Policy Engine that consumes it. The Policy Engine informs the Device Policy Manager that a ***Data_Reset*** Message has been received.
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 ***Data_Reset*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine 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 Message.	Physical Layer appends a CRC and sends the Message.
13	Protocol Layer stores the ***MessageID*** of the incoming Message.	
14	The Protocol Layer forwards the received ***Accept*** Message information to the Policy Engine that consumes it. The Policy Engine stops the ***SenderResponseTimer*** and starts the ***VCONNDischargeTimer***.	
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.
18		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 the Device Policy Manager to turn off VCONN.

Step	DFP (Reset Initiator)	UFP/VCONN Source (Reset Responder)
19		When the Device Policy Manager indicates VCONN has been turned off the Policy Engine tells the Protocol Layer to form an ***PS_RDY*** Message.
20		Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
21	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.
22	Protocol Layer stores the ***MessageID*** of the incoming Message.	
23	The Protocol Layer forwards the received ***PS_RDY*** Message information to the Policy Engine that consumes it. The Policy Engine stops the ***VCONNDischargeTimer*** and tells the Device Policy Manager to perform a Data Reset. The Device Policy Manager proceeds to turn on VCONN and then reset the data connection.	
24	Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.	
25	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.
26		Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.
27		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***PS_RDY*** Message was successfully sent.
28	The Device Policy Manager indicates that the Data Reset process is complete. The Policy Engine directs the Protocol Layer to generate a ***Data_Reset_Complete*** Message.	
29	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
30	Physical Layer appends CRC and sends the ***Data_Reset_Complete*** Message.	Physical Layer receives the ***Data_Reset_Complete*** Message and compares the CRC it calculated with the one sent to verify the Message.
31		Physical Layer removes the CRC and forwards the ***Data_Reset_Complete*** Message to the Protocol Layer.
32		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 ***Data_Reset_Complete*** Message information to the Policy Engine that consumes it. The Policy Engine informs the Device Policy Manager that a ***Data_Reset_Complete*** Message has been received.
33		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.
34	Physical Layer receives the ***GoodCRC*** and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the ***GoodCRC*** Message.
35	Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.	

Step	DFP (Reset Initiator)	UFP/VCONN Source (Reset Responder)
36	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Data_Reset_Complete*** Message was successfully sent. The Policy Engine informs the Device Policy Manager that the ***Data_Reset_Complete*** Message was successfully sent.	
The reset is complete as defined in Section 6.3.14 Step 5. Port Partners re-establish a USB data connection.		

8.3.2.4.4 DFP Receives Data Reset where the UFP is the VCONN Source

This is an example of a Data Reset operation where the DFP receives a Data Reset Message and the UFP is the VCONN Source. Figure 8-12shows the Messages as they flow across the bus and within the devices to accomplish the Data Reset.

Figure 8-12 DFP Receives a Data Reset where the UFP is the Vconn Source

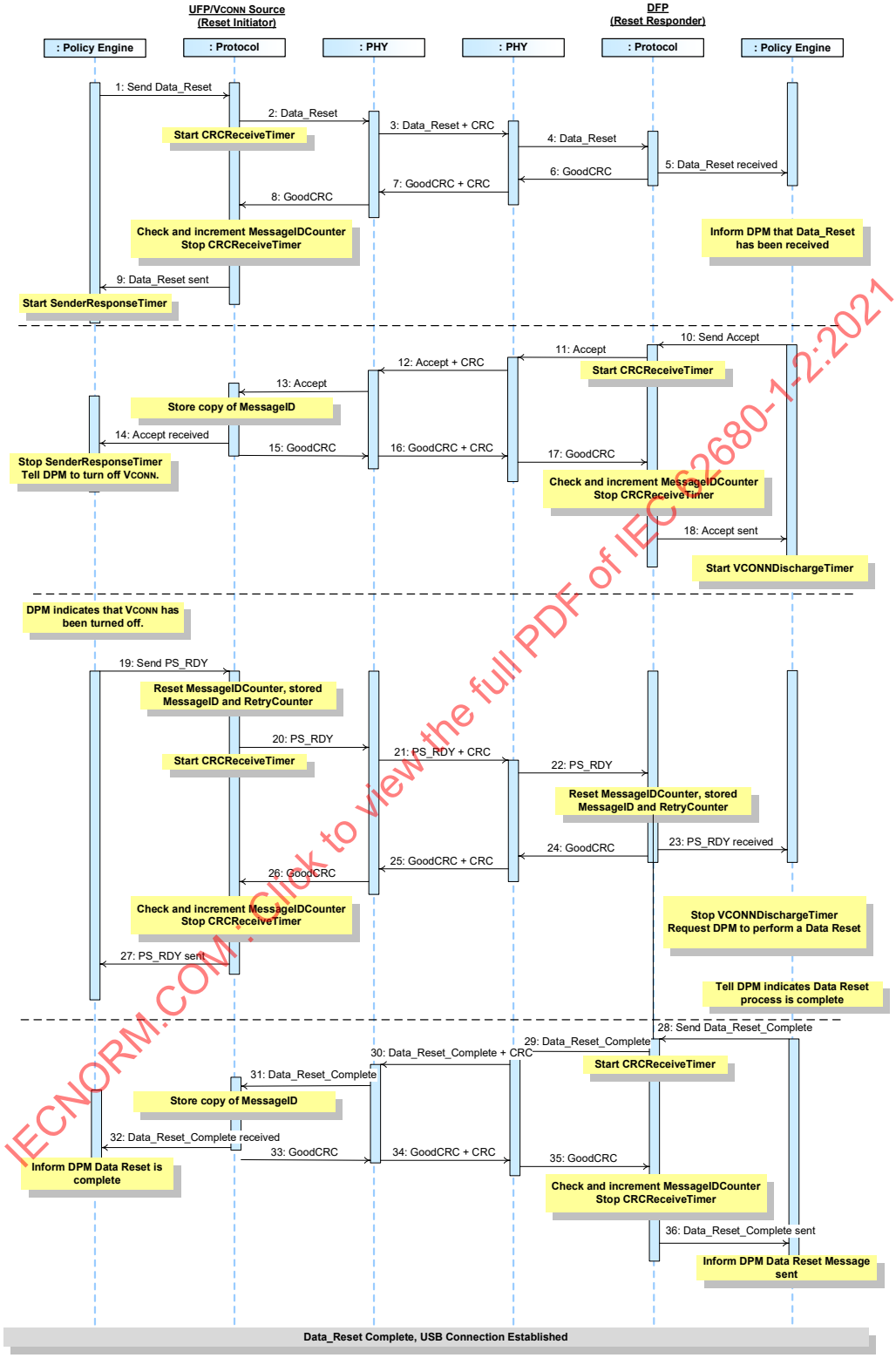


Table 8-12 below provides a detailed explanation of what happens at each labeled step in Figure 8-12 above.

Table 8-12 Steps for a DFP Receiving a Data Reset where the UFP is the Vconn Source

Step	UFP/VCONN Source (Reset Initiator)	DFP (Reset Responder)
1	The Policy Engine directs the Protocol Layer to generate a ***Data_Reset*** Message to request a Soft Reset.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Data_Reset*** Message.	Physical Layer receives the ***Data_Reset*** Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Data_Reset*** 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 ***Data_Reset*** Message information to the Policy Engine that consumes it. The Policy Engine informs the Device Policy Manager that a ***Data_Reset*** Message has been received.
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 ***Data_Reset*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine 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 Message.	Physical Layer appends a CRC and sends the Message.
13	Protocol Layer stores the ***MessageID*** of the incoming Message.	
14	The Protocol Layer forwards the received ***Accept*** Message information to the Policy Engine that consumes it. The Policy Engine stops the SenderResponseTimer and tells the Device Policy Manager to turn off VCONN.	
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.
18		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 starts the VCONNDischargeTimer.

Step	UFP/VCONN Source (Reset Initiator)	DFP (Reset Responder)
19	When the Device Policy Manager indicates that VCONN has been turned off the Policy Engine directs the Protocol Layer to generate a ***PS_RDY*** Message to request a Soft Reset.	
20	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
21	Physical Layer appends CRC and sends the ***PS_RDY*** Message.	Physical Layer receives the ***PS_RDY*** Message and compares the CRC it calculated with the one sent to verify the Message.
22		Physical Layer removes the CRC and forwards the ***PS_RDY*** Message to the Protocol Layer.
23		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 ***PS_RDY*** Message information to the Policy Engine that consumes it. The Policy Engine stops the VCONNDischargeTimer and requests the Device Policy Manager perform a Data Reset. The Device Policy Manager proceeds to turn on VCONN and then reset the data connection.
24		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.
25	Physical Layer receives ***GoodCRC*** Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the ***GoodCRC*** Message.
26	Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.	
27	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***PS_RDY*** Message was successfully sent.	
28		The Device Policy Manager indicates that the Data Reset process is complete. The Policy Engine directs the Protocol Layer to generate a ***Data_Reset_Complete*** Message.
29		Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
30	Physical Layer receives the ***Data_Reset_Complete*** Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the ***Data_Reset_Complete*** Message.
31	Physical Layer removes the CRC and forwards the ***Data_Reset_Complete*** Message to the Protocol Layer.	
32	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 ***Data_Reset_Complete*** Message information to the Policy Engine that consumes it. The Policy Engine informs the Device Policy Manager that a ***Data_Reset_Complete*** Message has been received.	
33	Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.	
34	Physical Layer appends CRC and sends the ***GoodCRC*** Message.	Physical Layer receives the ***GoodCRC*** and checks the CRC to verify the Message.

Step	UFP/VCONN Source (Reset Initiator)	DFP (Reset Responder)
35		Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.
36		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Data_Reset_Complete*** Message was successfully sent. The Policy Engine informs the Device Policy Manager that the ***Data_Reset_Complete*** Message was successfully sent.
The reset is complete as defined in Section 6.3.14 Step 5. Port Partners re-establish a USB data connection.		

8.3.2.5 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.

8.3.2.5.1 Source Initiated Hard Reset

This is an example of a Hard-Reset operation when initiated by a Source. Figure 8-13 shows the Messages as they flow across the bus and within the devices to accomplish the Hard Reset.

Figure 8-13 Source initiated Hard Reset

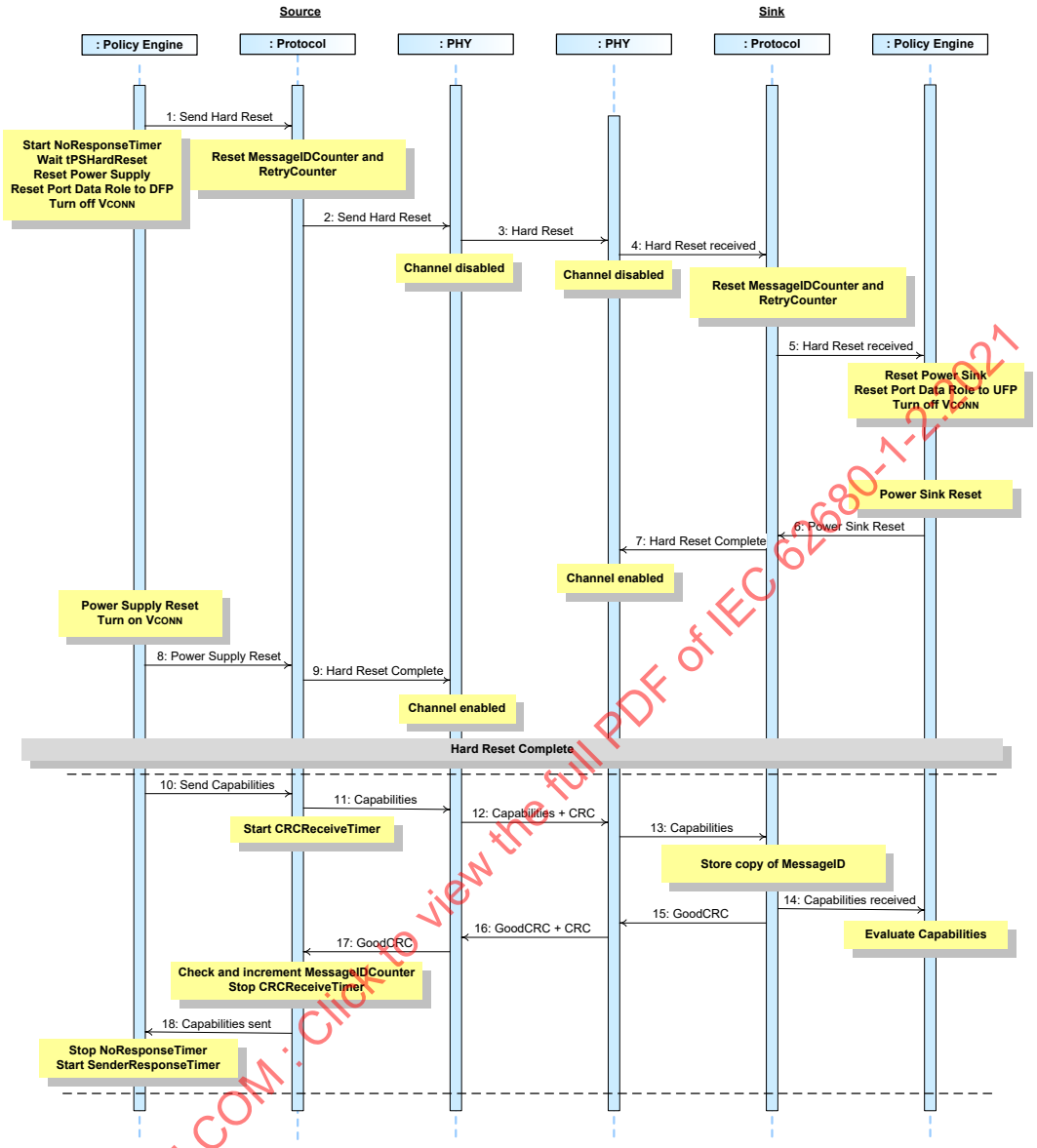


Table 8-13 Steps for Source initiated Hard Reset

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. 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*** and ***RetryCounter.*** Protocol Layer requests the Physical Layer send ***Hard Reset*** Signaling.	
3	Physical Layer sends ***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*** 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. 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 and 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 ***MessageID*** of the incoming Message. The Protocol Layer forwards the received ***Source_Capabilities*** Message information to the Policy Engine that consumes it.
15		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.
16	Physical Layer receives the ***GoodCRC*** Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the ***GoodCRC*** Message.
17	Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Source_Capabilities*** Message was successfully sent. Policy Engine stops the ***NoResponseTimer*** and starts the ***SenderResponseTimer***.	
	USB Power Delivery communication is re-established.	

8.3.2.5.2 Sink Initiated Hard Reset

This is an example of a Hard-Reset operation when initiated by a Sink. Figure 8-14 shows the Messages as they flow across the bus and within the devices to accomplish the Hard Reset.

Figure 8-14 Sink Initiated Hard Reset

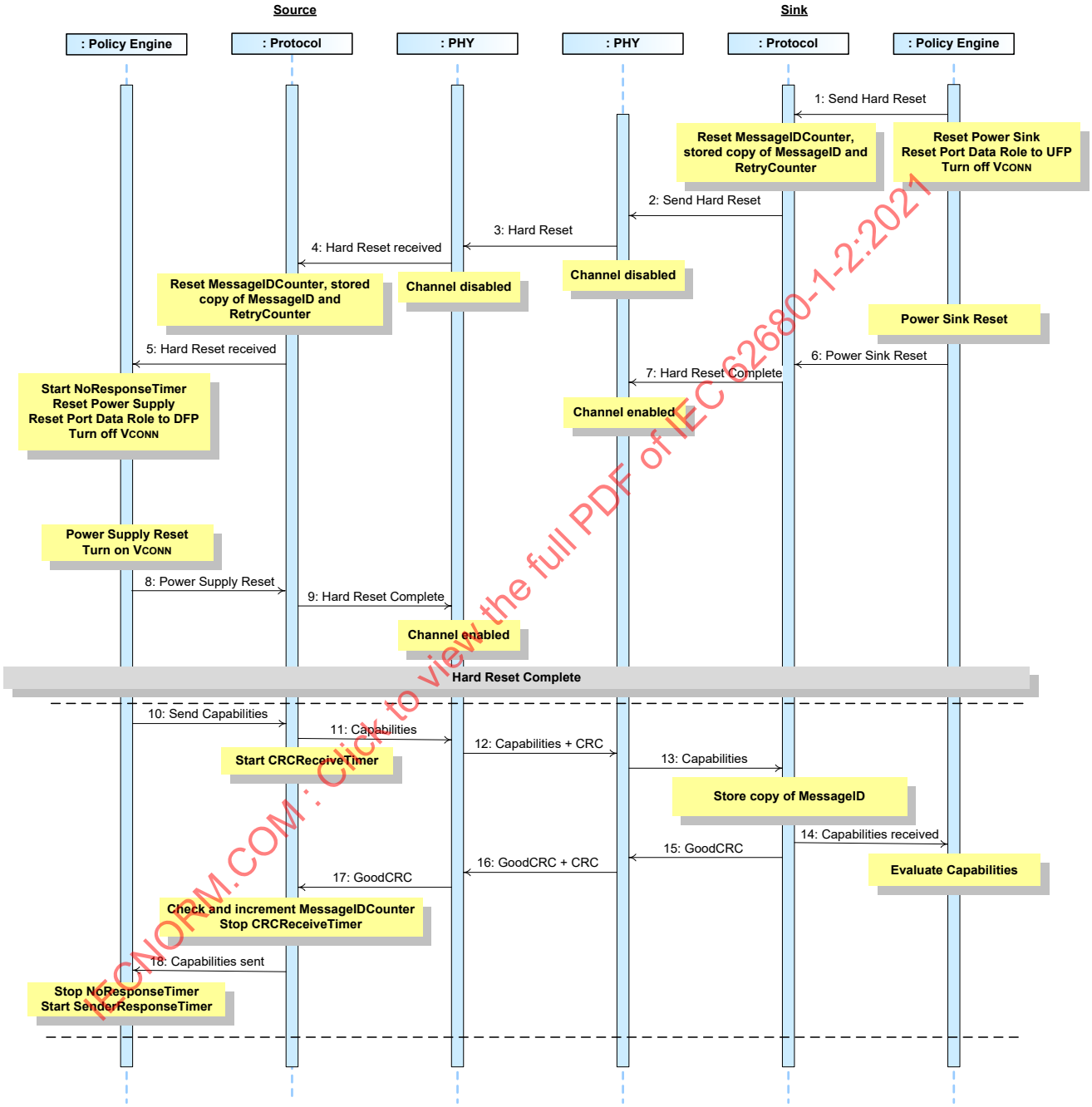


Table 8-14 Steps for Sink initiated Hard Reset

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. 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. 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 and 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.
14		Protocol Layer stores the ***MessageID*** of the incoming Message. The Protocol Layer forwards the received ***Source_Capabilities*** Message information to the Policy Engine that consumes it.

Step	Source	Sink
15		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.
16	Physical Layer receives the ***GoodCRC*** Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the ***GoodCRC*** Message.
17	Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Source_Capabilities*** Message was successfully sent. Policy Engine stops the ***NoResponseTimer*** and starts the ***SenderResponseTimer***.	
	USB Power Delivery communication is re-established.	

8.3.2.5.3 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 8-15 shows the Messages as they flow across the bus and within the devices to accomplish the Hard Reset.

Figure 8-15 Source initiated reset - Sink long reset

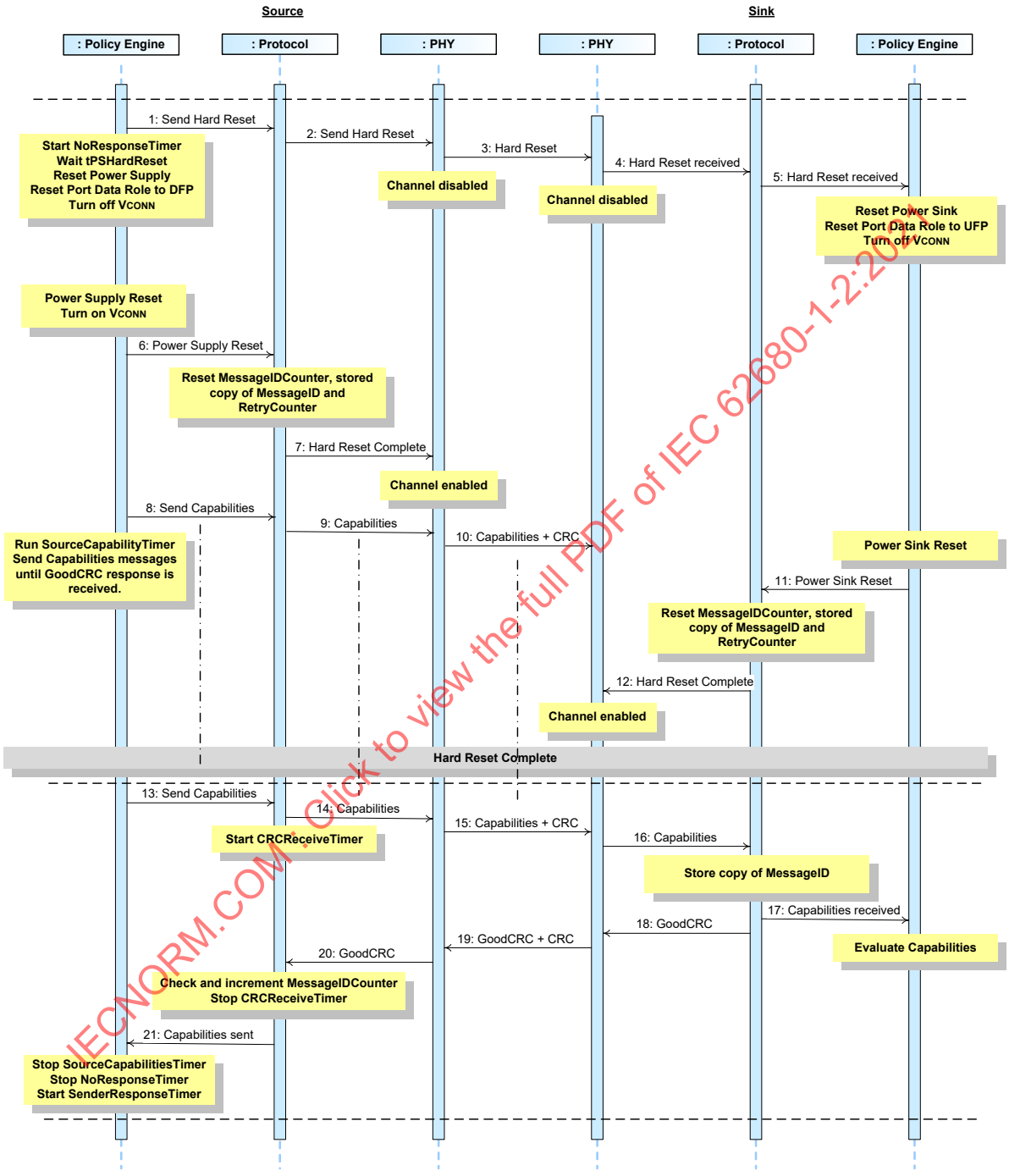


Table 8-15 Steps for Source initiated Hard Reset – Sink long reset

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. 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. 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 and 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.
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.	

Step	Source	Sink
13	Policy Engine directs the Protocol Layer to send a ***Source_Capabilities*** Message that represents the power supply's present capabilities. Starts the ***SourceCapabilityTimer***.	
14	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
15	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.
16		Physical Layer removes the CRC and forwards the ***Source_Capabilities*** Message to the Protocol Layer.
17		Protocol Layer stores the ***MessageID*** of the incoming Message. The Protocol Layer forwards the received ***Source_Capabilities*** Message information to the Policy Engine that consumes it.
18		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.
19	Physical Layer receives the ***GoodCRC*** Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the ***GoodCRC*** Message.
20	Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.	
21	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Source_Capabilities*** Message was successfully sent. Policy Engine stops the ***SourceCapabilityTimer***, stops the ***NoResponseTimer*** and starts the ***SenderResponseTimer***.	
	USB Power Delivery communication is re-established.	

8.3.2.6 Power Role Swap

8.3.2.6.1 Source Initiated Power Role Swap without subsequent Power Negotiation

This is an example of a successful Power Role Swap operation initiated by a Port which initially, at the start of this Message sequence, is acting as a Source and therefore has Rp 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 Section 8.3.2.2).

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 new Sink sets its power output to ***vSafe0V***, then asserts Rd and sends a ***PS_RDY*** Message when this process is complete.
4. The new Source asserts Rp, 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 sequence.

Figure 8-16 Successful Power Role Swap Sequence Initiated by the Source

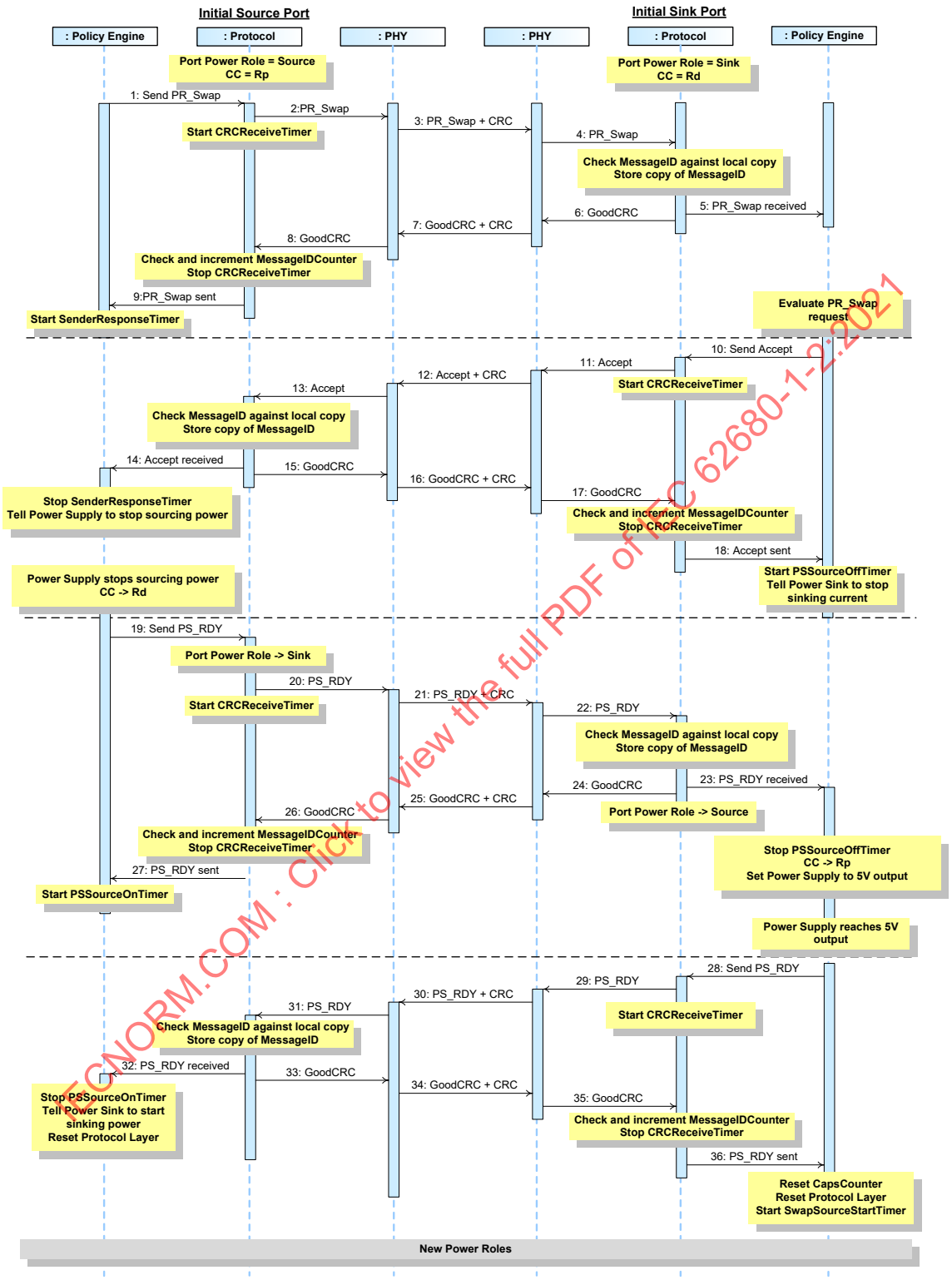


Table 8-16 below provides a detailed explanation of what happens at each labeled step in Figure 8-16 above.

Table 8-16 Steps for a Successful Source Initiated Power Role Swap Sequence

Step	Initial Source Port	Initially Sink Port
1	The Port has ***Port Power Role*** set to Source and the Rp pull up on its CC wire. Policy Engine directs the Protocol Layer to send a ***PR_Swap*** Message.	The Port has ***Port Power Role*** set to Sink with the Rd 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.
18		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 starts the ***PSSourceOffTimer*** and tells the power supply to stop sinking current.

Step	Initial Source Port	Initially Sink Port
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 Rd pull down on the CC wire. The Policy Engine then directs the Protocol Layer to generate a ***PS_RDY*** Message, with the ***Port Power Role*** 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 ***Port Power Role*** bit in the Message Header set to "Sink", creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
21	Physical Layer appends CRC and sends the ***PS_RDY*** Message.	Physical Layer receives the ***PS_RDY*** Message and checks the CRC to verify the Message.
22		Physical Layer removes the CRC and forwards the ***PS_RDY*** Message to the Protocol Layer.
23		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 ***PS_RDY*** Message information to the Policy Engine that consumes it. The Policy Engine stops the ***PSSourceOffTimer***, directs the Device Policy Manager to apply the Rp pull up and then starts switching the power supply to ***vSafe5V*** Source operation.
24		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.
25	Physical Layer receives the ***GoodCRC*** Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the ***GoodCRC*** Message.
26	Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.	
27	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***PS_RDY*** Message was successfully sent. Policy Engine starts ***PSSourceOnTimer***.	
28		Policy Engine, when its power supply is ready to supply power, tells the Protocol Layer to form a ***PS_RDY*** Message. The ***Port Power Role*** bit used in this and subsequent Message Headers is now set to "Source".
29		Protocol Layer creates the ***PS_RDY*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
30	Physical Layer receives the ***PS_RDY*** Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the ***PS_RDY*** Message.
31	Physical Layer removes the CRC and forwards the ***PS_RDY*** Message to the Protocol Layer.	
32	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 ***PS_RDY*** Message information to the Policy Engine that consumes it.	
33	Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.	

Step	Initial Source Port	Initially Sink Port
34	Physical Layer appends a CRC and sends the ***GoodCRC*** Message. The Policy Engine stops the ***PSSourceOnTimer***, informs the power supply it can now Sink power and resets the Protocol Layer.	Physical Layer receives ***GoodCRC*** Message and compares the CRC it calculated with the one sent to verify the Message.
35		Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.
36		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***PS_RDY*** Message was successfully sent. The Policy Engine resets the ***CapsCounter***, resets the Protocol Layer and starts the ***SwapSourceStartTimer*** which must timeout before sending any ***Source_Capabilities*** Messages.
	The Power Role Swap is complete, the roles have been reversed and the Port Partners are free to negotiate for more power.	

8.3.2.6.2 Sink Initiated Power Role Swap without subsequent Power Negotiation

This is an example of a successful Power Role Swap operation initiated by a Port which initially, at the start of this Message sequence, is acting as a Sink and therefore has Rd pulled down on 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).

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 new Sink sets its power output to ***vSafe0V***, then asserts Rd and sends a ***PS_RDY*** Message when this process is complete.
4. The new Source asserts Rp, then sets its power output to ***vSafe5V*** and sends a ***PS_RDY*** Message when it is ready to supply power.

Figure 8-17 shows the Messages as they flow across the bus and within the devices to accomplish the Power Role Swap.

Figure 8-17 Successful Power Role Swap Sequence Initiated by the Sink

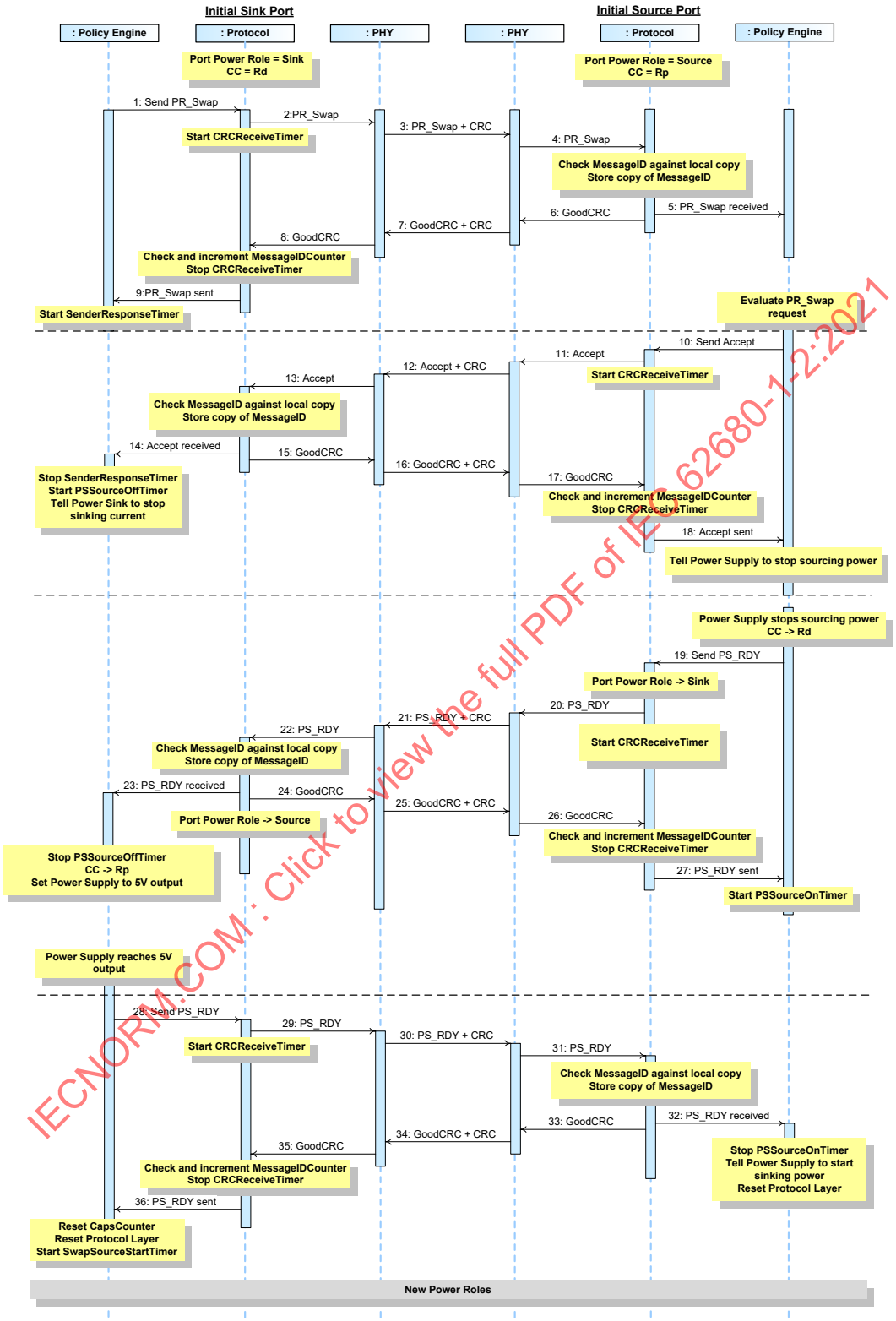


Table 8-17 Steps for a Successful Sink Initiated Power Role Swap Sequence

Step	Initial Sink Port	Initial Source Port
1	The Port has ***Port Power Role*** set to Sink with the Rd pull down on its CC wire. Policy Engine directs the Protocol Layer to send a ***PR_Swap*** Message.	The Port has ***Port Power Role*** set to Source and the Rp 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	Initial Sink Port	Initial Source Port
18		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 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 Rd pull down on the CC wire. The Policy Engine then directs the Protocol Layer to generate a ***PS_RDY*** Message, with the ***Port Power Role*** 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 ***Port Power Role*** bit in the Message Header set to "Sink", creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
21	Physical Layer receives the ***PS_RDY*** Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the ***PS_RDY*** Message.
22	Physical Layer removes the CRC and forwards the ***PS_RDY*** Message to the Protocol Layer.	
23	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 ***PS_RDY*** Message information to the Policy Engine that consumes it. The Policy Engine stops the ***PSSourceOffTimer***, directs the Device Policy Manager to apply the Rp pull up and then starts switching the power supply to ***vSafe5V*** Source operation.	
24	Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.	
25	Physical Layer appends CRC and sends the ***GoodCRC*** Message.	Physical Layer receives the ***GoodCRC*** Message and checks the CRC to verify the Message.
26		Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.
27		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***PS_RDY*** Message was successfully sent. Policy Engine starts ***PSSourceOnTimer***.
28	Policy Engine, when its power supply is ready to supply power, tells the Protocol Layer to form a ***PS_RDY*** Message. The ***Port Power Role*** bit used in this and subsequent Message Headers is now set to "Source".	
29	Protocol Layer creates the ***PS_RDY*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
30	Physical Layer appends a CRC and sends the ***PS_RDY*** Message.	Physical Layer receives the ***PS_RDY*** Message and compares the CRC it calculated with the one sent to verify the Message.
31		Physical Layer removes the CRC and forwards the ***PS_RDY*** Message to the Protocol Layer.

Step	Initial Sink Port	Initial Source Port
32		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 ***PS_RDY*** Message information to the Policy Engine that consumes it. The Policy Engine stops the ***PSSourceOnTimer***, informs the power supply that it can start consuming power.
33		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.
34	Physical Layer receives ***GoodCRC*** Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the ***GoodCRC*** Message. The Policy Engine stops the ***PSSourceOnTimer***, informs the power supply it can now Sink power and resets the Protocol Layer.
35	Physical Layer removes the CRC and forwards the ***GoodCRC*** to the Protocol Layer.	
36	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***PS_RDY*** Message was successfully sent. The Policy Engine resets the ***CapsCounter***, resets the Protocol Layer and starts the ***SwapSourceStartTimer*** which must timeout before sending any ***Source_Capabilities*** 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 Fast Role Swap

This is an example of a successful Fast Role Swap operation initiated by a Port that is initially a Source and therefore has Rp pulled up on its CC Wire and which has lost power and needs to get ***vSafe5V*** quickly. It does not include any subsequent Power Negotiation which is required in order to establish an Explicit Contract (see Section 8.3.2.2).

There are several distinct phases to the Fast Role Swap negotiation:

1. The initial Source stops driving its power output which starts transitioning to ***vSafe0V*** and signals Fast Role Swap on the CC Wire; these could occur in either order or simultaneously.
2. The initial Sink stops Sinking power. At this point the new Source still has Rd asserted and the new Sink still has Rp asserted.
3. An ***FR_Swap*** Message is sent by the new Source within ***tFRSwapInit*** of detecting the Fast Swap signal.
4. An ***Accept*** Message is sent by the new Sink in response to the ***FR_Swap*** Message.
5. The new Sink asserts Rd and sends a ***PS_RDY*** Message indicating that the voltage on V_{BUS} is at or below ***vSafe5V***.
6. The new Source asserts Rp and sends a ***PS_RDY*** Message indicating that it is acting as a Source and is supplying ***vSafe5V***. Note: that the new Source can start applying V_{BUS} when V_{BUS} is at or below ***vSafe5V*** (max) but will start driving V_{BUS} to ***vSafe5V*** no later than ***tSrcFRSwap*** after detecting both the FRS signal and that V_{BUS} has dropped below ***vSafe5V*** (min).

Figure 8-18 shows the Messages as they flow across the bus and within the devices to accomplish the Fast Role Swap.

Figure 8-18 Successful Fast Role Swap Sequence

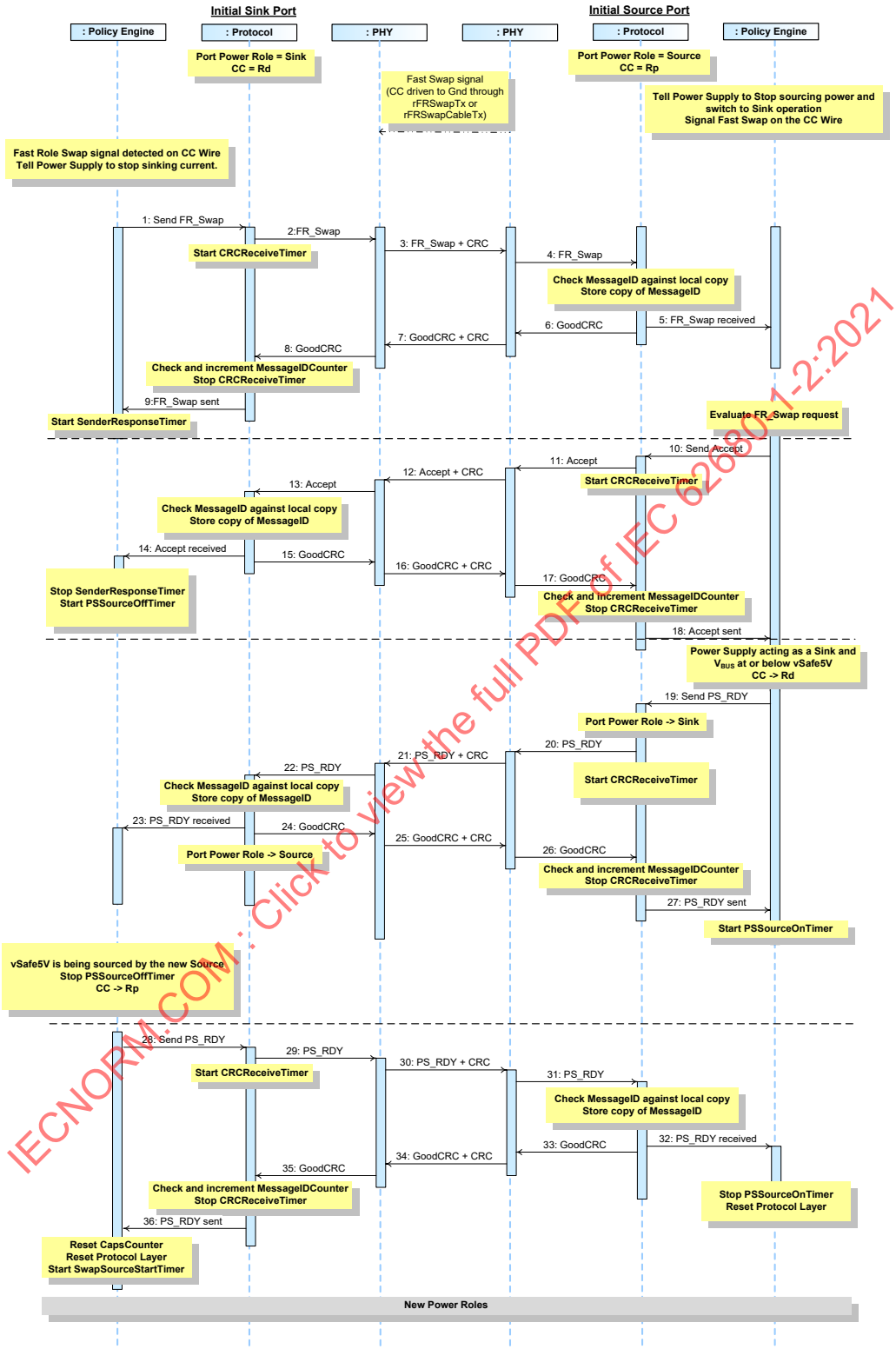


Table 8-18 Steps for a Successful Fast Role Swap Sequence

Step	Initial Sink Port	Initial Source Port
1	The Port has ***Port Power Role*** set to Sink with the Rd pull down on its CC wire. The Device Policy Manager detects Fast Swap on the CC Wire and tells the power supply to stop sinking current. The Policy Engine directs the Protocol Layer to send an ***FR_Swap*** Message within ***tFRSwapInit*** of detecting the Fast Swap signal.	The Port has ***Port Power Role*** set to Source and the Rp pull up on its CC wire. The Device Policy Manager tells the Power Supply to stop sourcing power and switch to Sink operation. The Device Policy Manager signals Fast Swap on the CC Wire by driving CC to ground with a resistance of less than ***rFRSwapTx*** for at least ***tFRSwapTx***.
2	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***FR_Swap*** Message.	Physical Layer receives the ***FR_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 ***FR_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 ***FR_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***.	
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	Initial Sink Port	Initial Source Port
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Accept*** Message was successfully sent.
19		The Policy Engine determines its power supply is no longer supplying V_{BUS} and is acting as a Sink. The Policy Engine requests the Device Policy Manager to assert the Rd pull down on the CC wire. The Policy Engine then directs the Protocol Layer to generate a ***PS_RDY*** Message, with the ***Port Power Role*** 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 ***Port Power Role*** bit in the Message Header set to "Sink", creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
21	Physical Layer receives the ***PS_RDY*** Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the ***PS_RDY*** Message.
22	Physical Layer removes the CRC and forwards the ***PS_RDY*** Message to the Protocol Layer.	
23	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 ***PS_RDY*** Message information to the Policy Engine that consumes it. The Policy Engine stops the ***PSSourceOffTimer***.	
24	Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.	
25	Physical Layer appends CRC and sends the ***GoodCRC*** Message.	Physical Layer receives the ***GoodCRC*** Message and checks the CRC to verify the Message.
26		Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.
27		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***PS_RDY*** Message was successfully sent. Policy Engine starts ***PSSourceOnTimer***.
28	The Policy Engine directs the Device Policy Manager to apply the Rp pull up. Note: at some point (either before or after receiving the ***PS_RDY*** Message) the new Source has applied ***vSafe5V*** no later than ***tSrcFRSwap*** after detecting the FRS signal and that V_{BUS} has dropped below ***vSafe5V***. Policy Engine, when its power supply is ready to supply power, tells the Protocol Layer to form a ***PS_RDY*** Message. The ***Port Power Role*** bit used in this and subsequent Message Headers is now set to "Source".	
29	Protocol Layer creates the ***PS_RDY*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
30	Physical Layer appends a CRC and sends the ***PS_RDY*** Message.	Physical Layer receives the ***PS_RDY*** Message and compares the CRC it calculated with the one sent to verify the Message.
31		Physical Layer removes the CRC and forwards the ***PS_RDY*** Message to the Protocol Layer.

Step	Initial Sink Port	Initial Source Port
32		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 ***PS_RDY*** Message information to the Policy Engine that consumes it. The Policy Engine stops the ***PSSourceOnTimer***.
33		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.
34	Physical Layer receives ***GoodCRC*** Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the ***GoodCRC*** Message. The Policy Engine resets the Protocol Layer.
35	Physical Layer removes the CRC and forwards the ***GoodCRC*** to the Protocol Layer.	
36	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***PS_RDY*** Message was successfully sent. The Policy Engine resets the ***CapsCounter***, resets the Protocol Layer and starts the ***SwapSourceStartTimer*** which must timeout before sending any ***Source_Capabilities*** Messages.	
	The Fast Role Swap is complete, the roles have been reversed and the Port Partners are free to negotiate for more power.	

8.3.2.8 Data Role Swap

8.3.2.8.1 Data Role Swap, Initiated by UFP Operating as Sink

Figure 8-19 shows an example sequence between a Port, which is initially a UFP (Device) and a Sink (Rd asserted), and a Port which is initially a DFP (Host) and a Source (Rp 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 Rp/Rd remain constant) but exchange data roles between DFP (Host) and UFP (Device).

Figure 8-19 Data Role Swap, UFP operating as Sink initiates

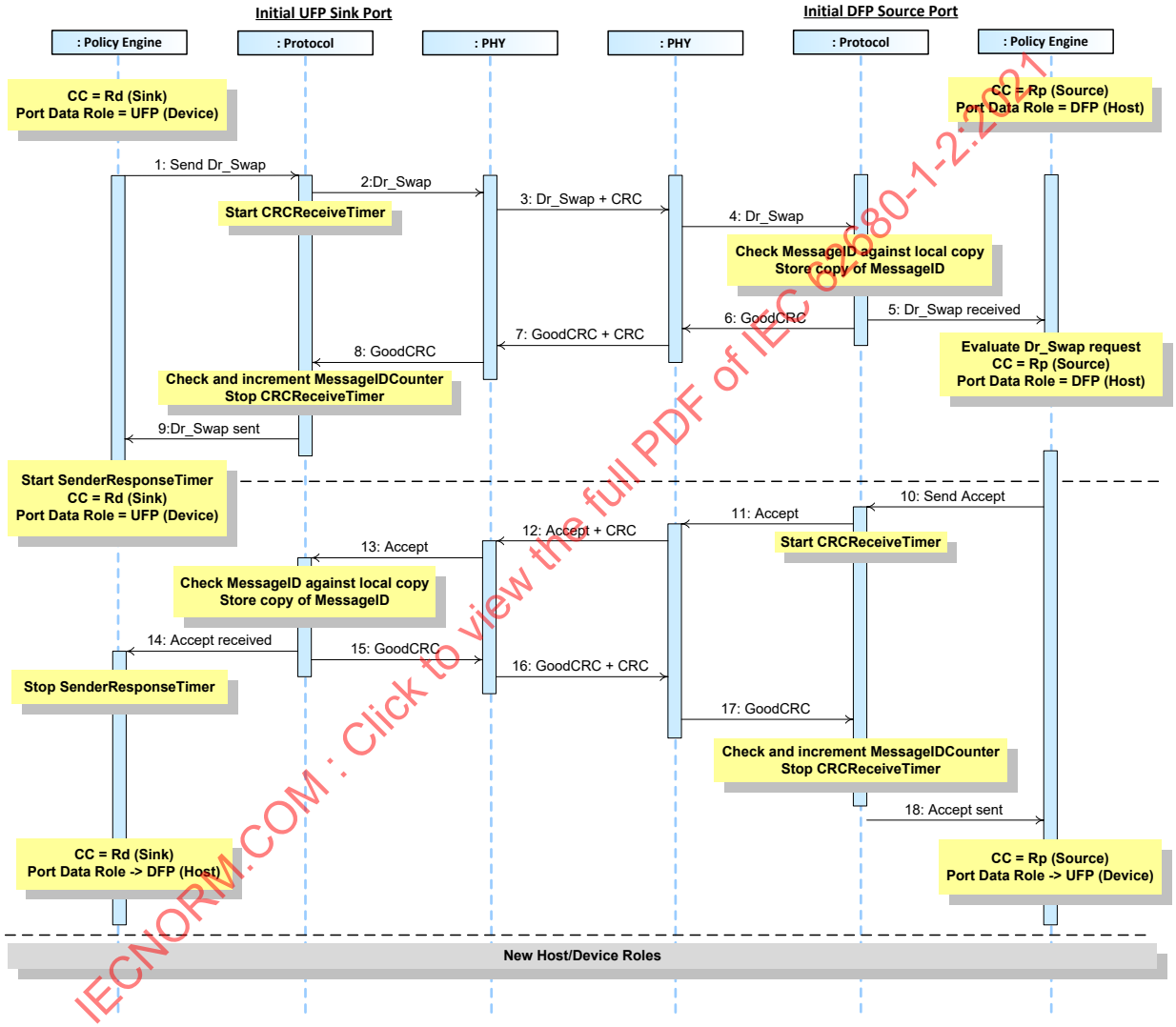


Table 8-19 below provides a detailed explanation of what happens at each labeled step in Figure 8-19 above.

Table 8-19 Steps for Data Role Swap, UFP operating as Sink initiates

Step	Initial UFP Sink Port	Initial DFP Source Port
1	Port starts as a UFP (Device) operating as a Sink with Rd asserted and ***Port Data Role*** set to UFP. The Policy Engine directs the Protocol Layer to send a ***DR_Swap*** Message.	Port starts as a DFP (Host) operating as Source with Rp asserted and ***Port Data Role*** set to DFP.
2	Protocol Layer creates the ***DR_Swap*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	

Step	Initial UFP Sink Port	Initial DFP Source Port
3	Physical Layer appends CRC and sends the ***DR_Swap*** Message.	Physical Layer receives the ***DR_Swap*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***DR_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 ***DR_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 ***DR_Swap*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine evaluates the ***DR_Swap*** Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form an ***Accept*** Message.
11		Protocol Layer creates the ***Accept*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
12	Physical Layer receives the ***Accept*** Message and checks the CRC to verify the 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 ***Accept*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18	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 (Rd asserted).	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 (Rp 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 Data Role Swap, Initiated by UFP Operating as Source

Figure 8-20 shows an example sequence between a Port, which is initially a UFP (Device) and a Source (Rp asserted), and a Port which is initially a DFP (Host) and a Sink (Rd 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 Rp/Rd remain constant) but exchange data roles between DFP (Host) and UFP (Device).

Figure 8-20 Data Role Swap, UFP operating as Source initiates

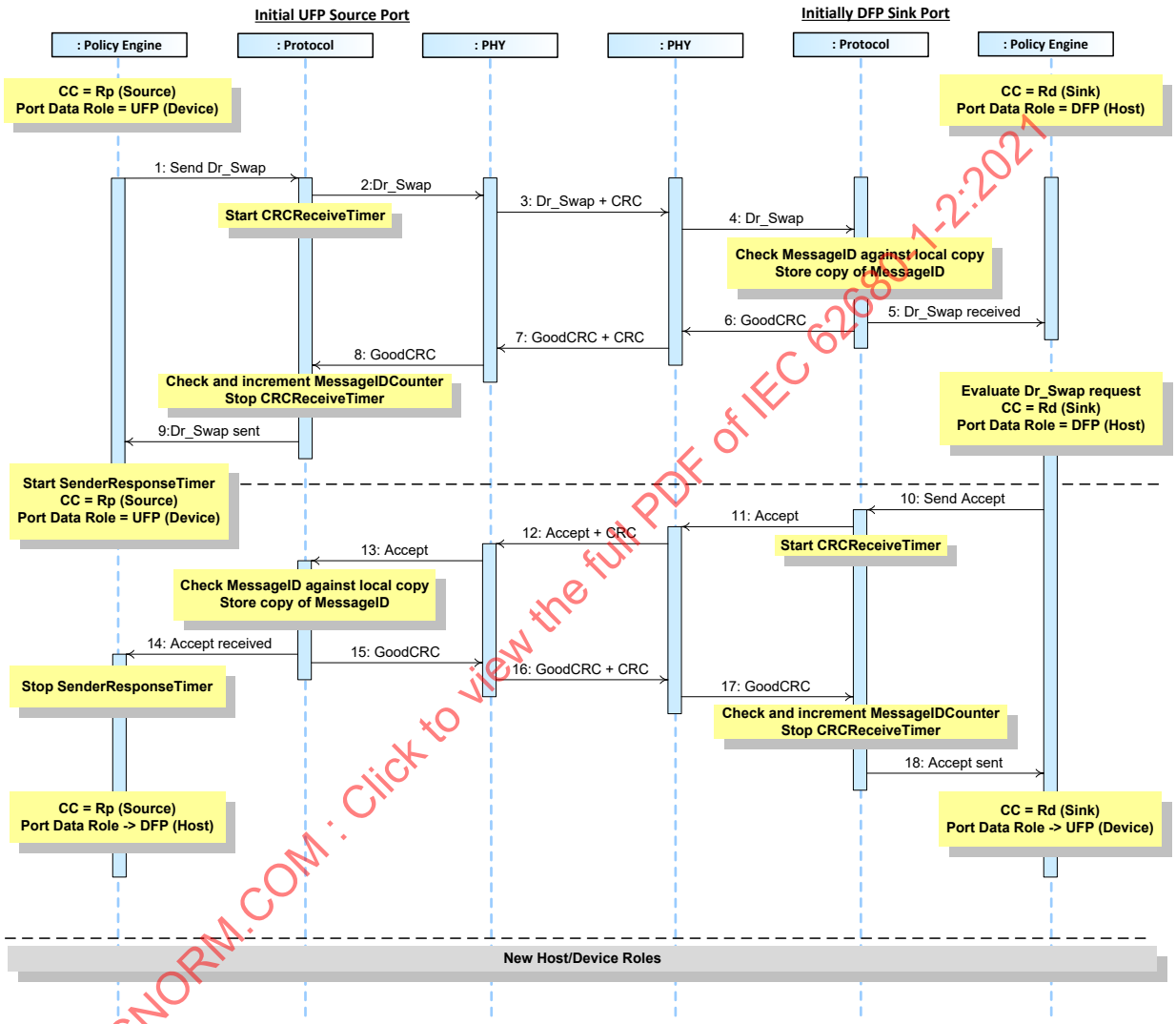


Table 8-20 below provides a detailed explanation of what happens at each labeled step in Figure 8-20 above.

Table 8-20 Steps for Data Role Swap, UFP operating as Source initiates

Step	Initial UFP Source Port	Initial DFP Sink Port
1	Port starts as a UFP (Device) operating as Source with Rp asserted and ***Port Data Role*** set to UFP. The Policy Engine directs the Protocol Layer to send a ***DR_Swap*** Message.	Port starts as a DFP (Host) operating as a Sink with Rd asserted and ***Port Data Role*** set to DFP.
2	Protocol Layer creates the ***DR_Swap*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***DR_Swap*** Message.	Physical Layer receives the ***DR_Swap*** Message and checks the CRC to verify the Message.

Step	Initial UFP Source Port	Initial DFP Sink Port
4		Physical Layer removes the CRC and forwards the ***DR_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 ***DR_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 ***DR_Swap*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine evaluates the ***DR_Swap*** Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form an ***Accept*** Message.
11		Protocol Layer creates the ***Accept*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
12	Physical Layer receives the ***Accept*** Message and checks the CRC to verify the 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 ***Accept*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18	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 (Rp asserted).	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 (Rp 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 Data Role Swap, Initiated by DFP Operating as Source

Figure 8-21 shows an example sequence between a Port, which is initially a UFP (Device) and a Sink (Rd asserted), and a Port which is initially a DFP and a Source (Rp 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 Rp/Rd remain constant) but exchange data roles between DFP (Host) and UFP (Device).

Figure 8-21 Data Role Swap, DFP operating as Source initiates

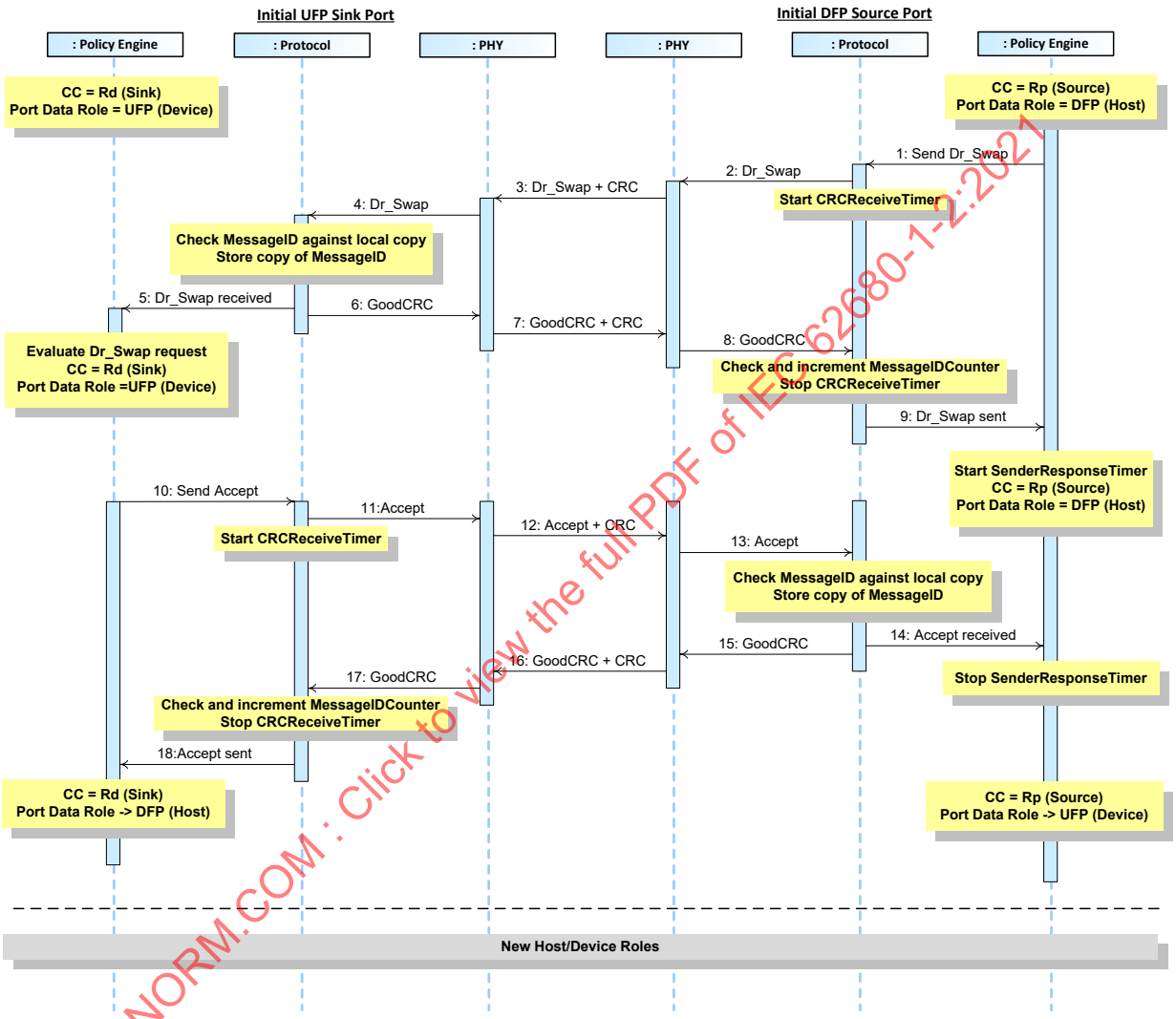


Table 8-21 below provides a detailed explanation of what happens at each labeled step in Figure 8-21 above.

Table 8-21 Steps for Data Role Swap, DFP operating as Source initiates

Step	Initial UFP Sink Port	Initial DFP Source Port
1	Port starts as a UFP (Device) operating as a Sink with Rd asserted and ***Port Data Role*** set to UFP.	Port starts as a DFP (Host) operating as Source with Rp asserted and ***Port Data Role*** set to DFP. The Policy Engine directs the Protocol Layer to send a ***DR_Swap*** Message.
2		Protocol Layer creates the ***DR_Swap*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
3	Physical Layer receives the ***DR_Swap*** Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the ***DR_Swap*** Message.

Step	Initial UFP Sink Port	Initial DFP Source Port
4	Physical Layer removes the CRC and forwards the ***DR_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 ***DR_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 appends CRC and sends the ***GoodCRC*** Message.	Physical Layer receives the ***GoodCRC*** Message and checks the CRC to verify the 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 ***DR_Swap*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.
10	Policy Engine evaluates the ***DR_Swap*** Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form an ***Accept*** Message.	
11	Protocol Layer creates the ***Accept*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
12	Physical Layer appends a CRC and sends the ***Accept*** Message.	Physical Layer receives the ***Accept*** Message and checks the CRC to verify the 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 ***Accept*** Message information to the Policy Engine that consumes it.
14		The Policy Engine stops the ***SenderResponseTimer***.
15		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.
16	Physical Layer receives ***GoodCRC*** Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the ***GoodCRC*** Message.
17	Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.	
18	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 DFP (Host), with ***Port Data Role*** set to DFP, still operating as a Sink (Rd 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 ***Port Data Role*** set to UFP, and continues supplying power as a Source (Rp asserted).
	The Data Role Swap is complete; the data roles have been reversed while maintaining the direction of power flow.	

8.3.2.8.4 Data Role Swap, Initiated by DFP Operating as Sink

Figure 8-22 shows an example sequence between a Port, which is initially a UFP (Device) and a Source (Rp asserted), and a Port which is initially a DFP (Host) and a Sink (Rd 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 Rp/Rd remain constant) but exchange data roles between DFP (Host) and UFP (Device).

Figure 8-22 Data Role Swap, DFP operating as Sink initiates

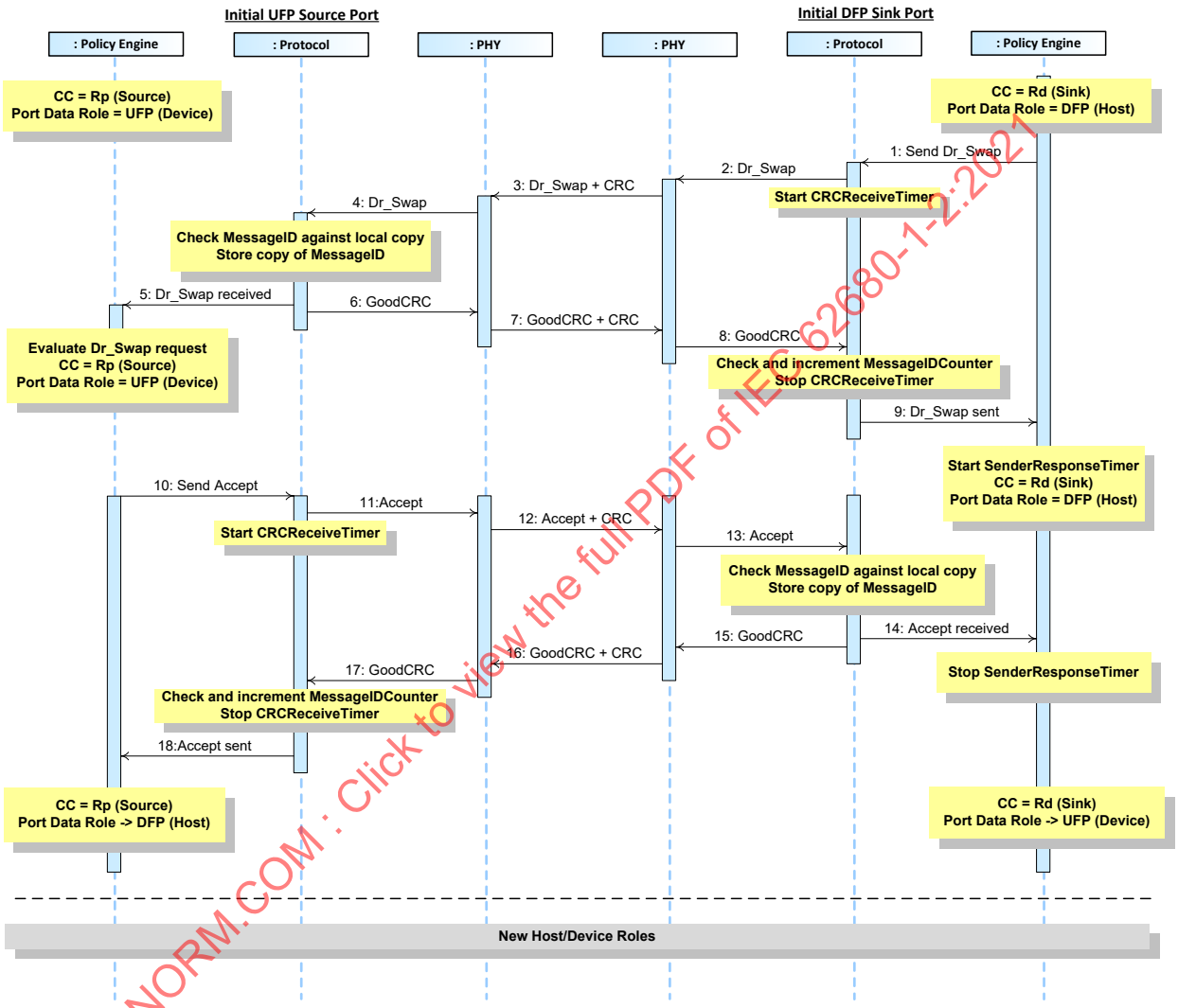


Table 8-22 below provides a detailed explanation of what happens at each labeled step in Figure 8-22 above.

Table 8-22 Steps for Data Role Swap, DFP operating as Sink initiates

Step	Initial UFP Source Port	Initial DFP Sink Port
1	Port starts as a UFP (Device) operating as Source with Rp asserted and ***Port Data Role*** set to UFP.	Port starts as a DFP (Host) operating as a Sink with Rd asserted and ***Port Data Role*** set to DFP. The Policy Engine directs the Protocol Layer to send a ***DR_Swap*** Message.
2		Protocol Layer creates the ***DR_Swap*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
3	Physical Layer receives the ***DR_Swap*** Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the ***DR_Swap*** Message.

Step	Initial UFP Source Port	Initial DFP Sink Port
4	Physical Layer removes the CRC and forwards the ***DR_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 ***DR_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 appends CRC and sends the ***GoodCRC*** Message.	Physical Layer receives the ***GoodCRC*** Message and checks the CRC to verify the 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 ***DR_Swap*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.
10	Policy Engine evaluates the ***DR_Swap*** Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form an ***Accept*** Message.	
11	Protocol Layer creates the ***Accept*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
12	Physical Layer appends a CRC and sends the ***Accept*** Message.	Physical Layer receives the ***Accept*** Message and checks the CRC to verify the 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 ***Accept*** Message information to the Policy Engine that consumes it.
14		The Policy Engine stops the ***SenderResponseTimer***.
15		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.
16	Physical Layer receives ***GoodCRC*** Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the ***GoodCRC*** Message.
17	Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.	
18	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 DFP (Host), with ***Port Data Role*** set to DFP, and continues supplying power as a Source (Rp 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 ***Port Data Role*** set to UFP, still operating as a Sink (Rd asserted).
	The Data Role Swap is complete; the data roles have been reversed while maintaining the direction of power flow.	

8.3.2.9 VCONN Swap

8.3.2.9.1 Source to Sink VCONN Source Swap

Figure 8-23 shows an example sequence between a 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 8-23 Source to Sink VCONN Source Swap

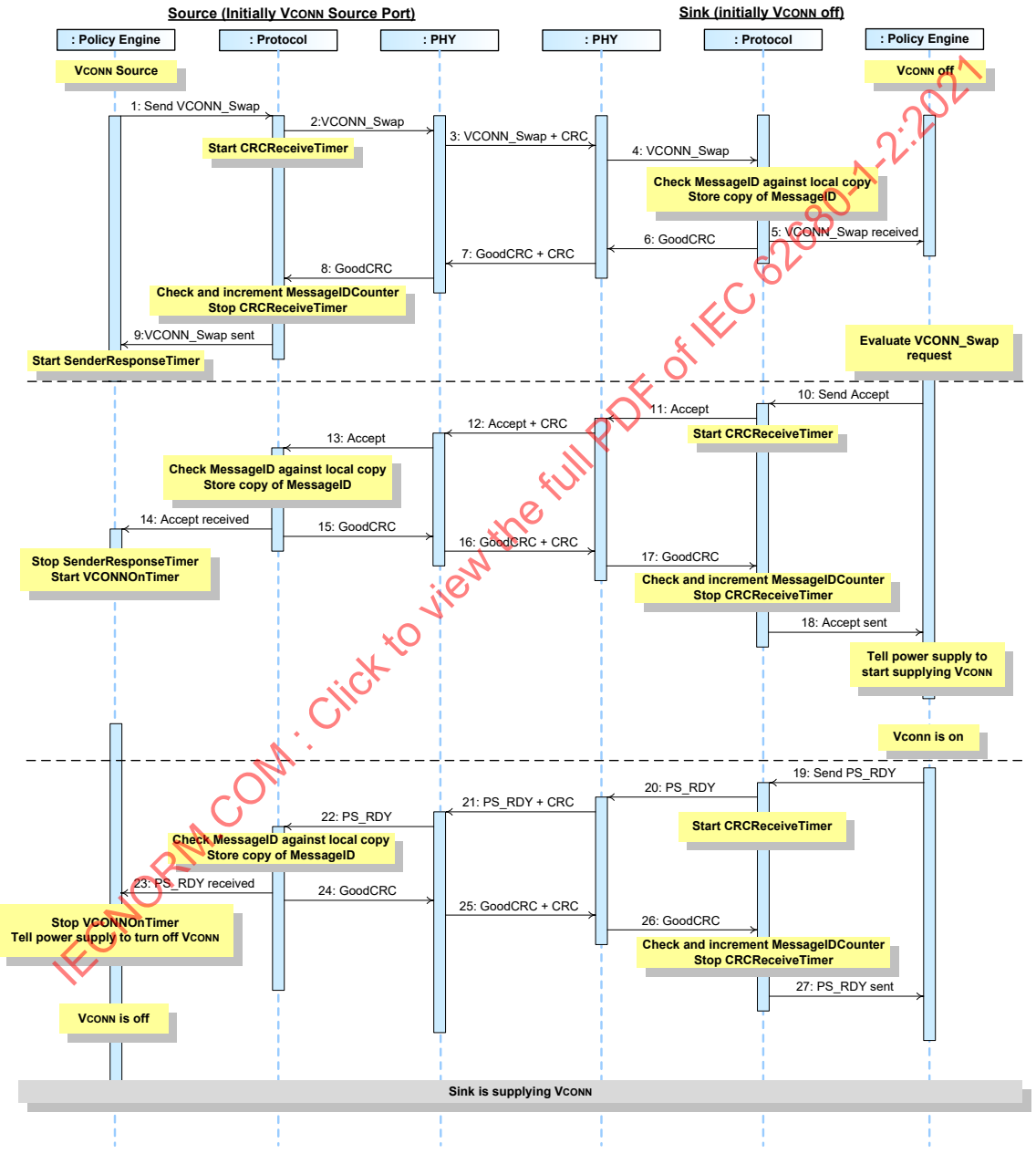


Table 8-23 below provides a detailed explanation of what happens at each labeled step in Figure 8-23 above.

Table 8-23 Steps for 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 ***VCONN_Swap*** Message.	The Sink starts with VCONN off.
2	Protocol Layer creates the ***VCONN_Swap*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***VCONN_Swap*** Message.	Physical Layer receives the ***VCONN_Swap*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***VCONN_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 ***VCONN_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 ***VCONN_Swap*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine evaluates the ***VCONN_Swap*** 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 ***Accept*** Message.
11		Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
12	Physical Layer receives the ***Accept*** Message and compares the CRC it calculated with the one sent to verify the 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 ***Accept*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the ***SenderResponseTimer*** and starts the ***VCONNOnTimer***.	
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	Source (initially VCONN Source)	Sink (Initially VCONN off)
18		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 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 ***PS_RDY*** Message to tell the Source it can turn off VCONN.
20		Protocol Layer creates the ***PS_RDY*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
21	Physical Layer receives the ***PS_RDY*** Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the ***PS_RDY*** Message.
22	Physical Layer removes the CRC and forwards the ***PS_RDY*** Message to the Protocol Layer.	
23	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 ***PS_RDY*** Message information to the Policy Engine that consumes it.	
24	Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.	
25	Physical Layer appends a CRC and sends the ***GoodCRC*** Message. The Policy Engine stops the ***VCONNOnTimer***, and tells the power supply to stop sourcing VCONN.	Physical Layer receives ***GoodCRC*** Message and compares the CRC it calculated with the one sent to verify the Message.
26		Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.
27	VCONN is off.	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***PS_RDY*** Message was successfully sent.
	The Sink is now the VCONN Source and the Source has VCONN turned off.	

8.3.2.9.2 Sink to Source VCONN Source Swap

Figure 8-24 shows an example sequence between a 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 8-24 Sink to Source VCONN Source Swap

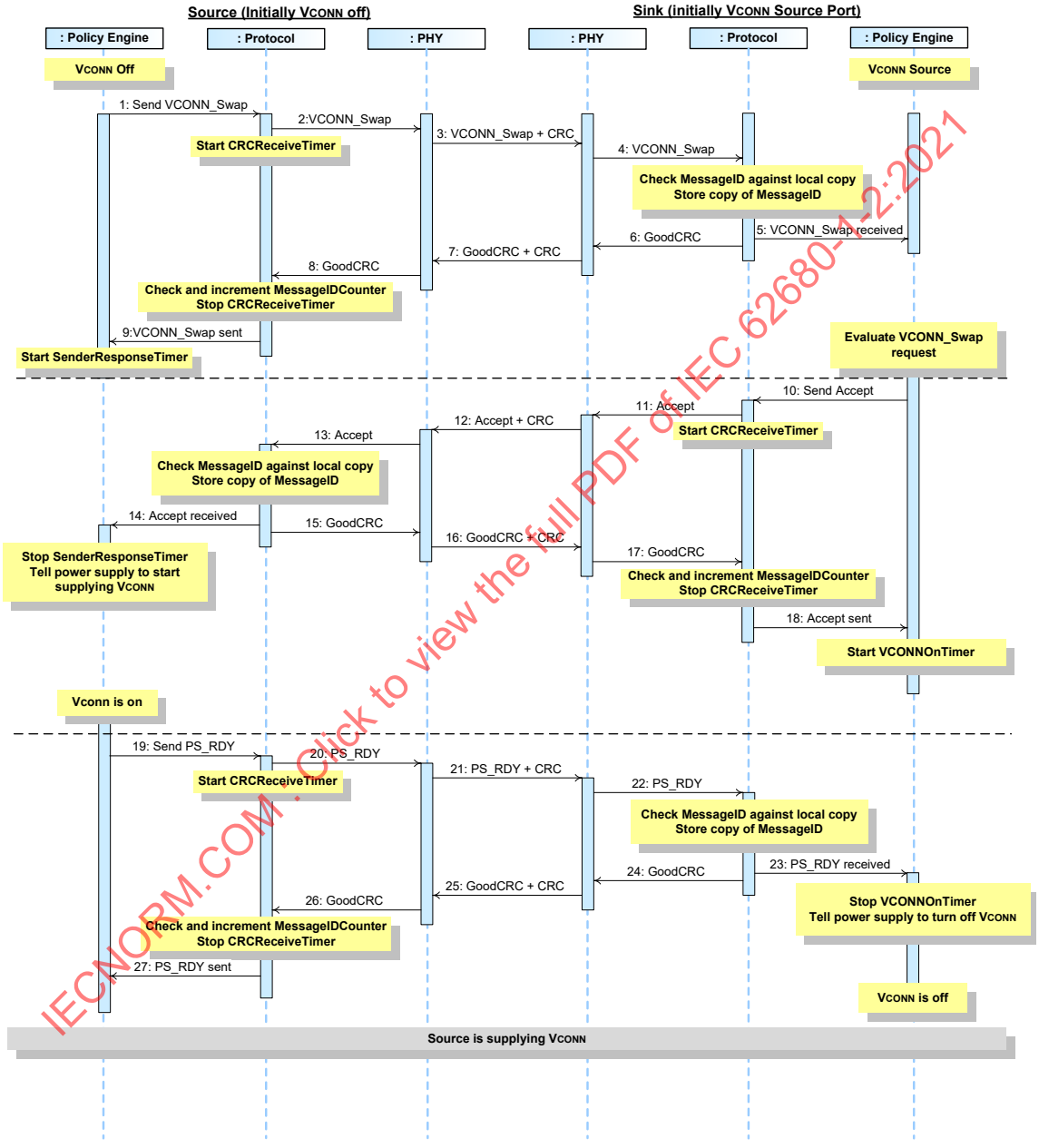


Table 8-24 below provides a detailed explanation of what happens at each labeled step in Figure 8-24 above.

Table 8-24 Steps for 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 ***VCONN_Swap*** Message.	The Sink starts as the VCONN Source.
2	Protocol Layer creates the ***VCONN_Swap*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***VCONN_Swap*** Message.	Physical Layer receives the ***VCONN_Swap*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***VCONN_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 ***VCONN_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 ***VCONN_Swap*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine evaluates the ***VCONN_Swap*** 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 ***Accept*** Message.
11		Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
12	Physical Layer receives the ***Accept*** Message and compares the CRC it calculated with the one sent to verify the 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 ***Accept*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the ***SenderResponseTimer***. The Policy Engine tells the Device Policy Manger to turn on VCONN.	
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	Source	Sink
18		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 starts the ***VCONNOnTimer***.
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 ***PS_RDY*** Message to tell the Sink it can turn off VCONN.	
20	Protocol Layer creates the ***PS_RDY*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
21	Physical Layer appends a CRC and sends the ***PS_RDY*** Message.	Physical Layer receives the ***PS_RDY*** Message and compares the CRC it calculated with the one sent to verify the Message.
22		Physical Layer removes the CRC and forwards the ***PS_RDY*** Message to the Protocol Layer.
23		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 ***PS_RDY*** Message information to the Policy Engine that consumes it.
24		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.
25	Physical Layer receives ***GoodCRC*** Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the ***GoodCRC*** Message. The Policy Engine stops the ***VCONNOnTimer***, and tells the power supply to stop sourcing VCONN.
26	Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.	
27	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***PS_RDY*** Message was successfully sent.	VCONN is off.
	The Source is now the VCONN Source and the Sink has VCONN turned off.	

8.3.2.10 Additional Capabilities, Status and Information

8.3.2.10.1 Alert

8.3.2.10.1.1 Source sends Alert to a Sink

Figure 8-25 shows an example sequence between a Source and a Sink where the Source alerts the Sink that there has been a status change. This AMS will be followed by getting the Source status to determine further details of the alert (see Section 8.3.2.10.2).

Figure 8-25 Source Alert to Sink

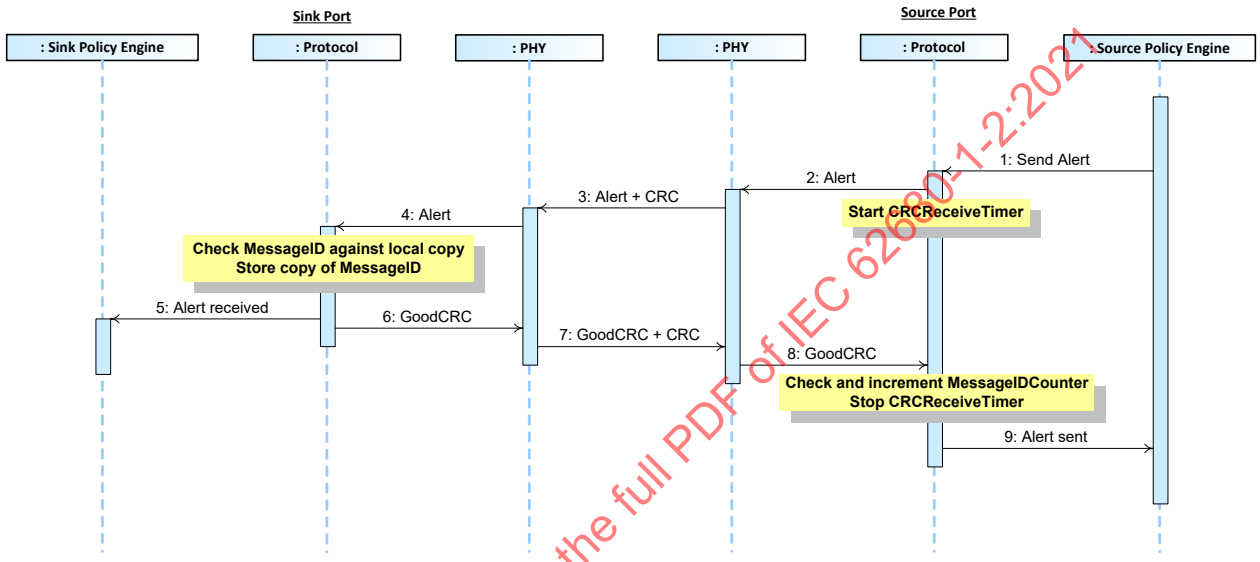


Table 8-64 below provides a detailed explanation of what happens at each labeled step in Figure 8-25 above.

Table 8-25 Steps for Source Alert to Sink

Step	Sink	Source
1		The Device Policy Manager indicates a Source alert condition. The Policy Engine tells the Protocol Layer to form an ***Alert*** Message.
2		Protocol Layer creates the ***Alert*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
3	Physical Layer receives the ***Alert*** Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the ***Alert*** Message.
4	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 ***Alert*** Message to the Policy Engine that consumes it.	
5	The Policy Engine informs the Device Policy Manager.	
6	Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.	
7	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.
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 ***Alert*** Message was successfully sent.

8.3.2.10.1.1 Sink sends Alert to a Source

Figure 8-25 shows an example sequence between a Source and a Sink where the Sink alerts the Source that there has been a status change. This AMS will be followed by getting the Sink status to determine further details of the alert (see Section 8.3.2.10.2).

Figure 8-26 Sink Alert to Source

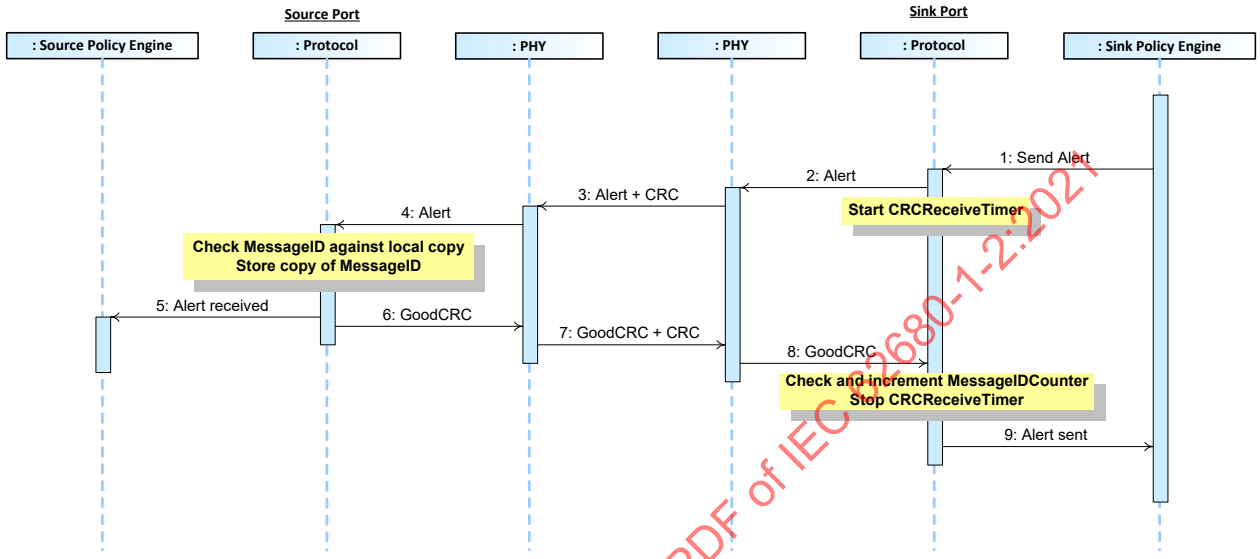


Table 8-64 below provides a detailed explanation of what happens at each labeled step in Figure 8-25 above.

Table 8-26 Steps for Sink Alert to Source

Step	Source	Sink
1		The Device Policy Manager indicates a Sink alert condition. The Policy Engine tells the Protocol Layer to form an ***Alert*** Message.
2		Protocol Layer creates the ***Alert*** Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
3	Physical Layer receives the ***Alert*** Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the ***Alert*** Message.
4	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 ***Alert*** Message to the Policy Engine that consumes it.	
5	The Policy Engine informs the Device Policy Manager.	
6	Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.	
7	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.
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 ***Alert*** Message was successfully sent.

8.3.2.10.2 Status

8.3.2.10.2.1 Sink Gets Source Status

Figure 8-27 shows an example sequence between a Source and a Sink where, after the Sink has received an alert (see Section 8.3.2.10.1) that there has been a status change, the Sink gets more details on the change.

Figure 8-27 Sink Gets Source Status

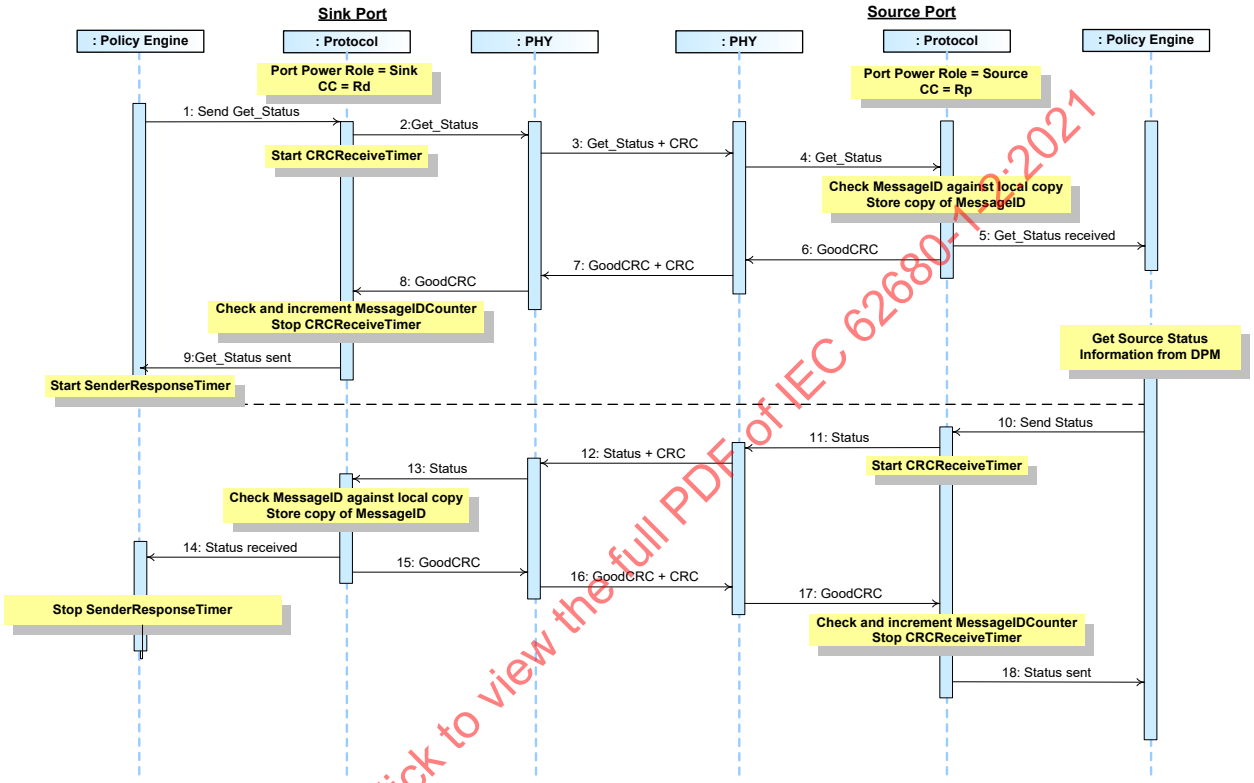


Table 8-27 below provides a detailed explanation of what happens at each labeled step in Figure 8-27 above.

Table 8-27 Steps for a Sink getting Source Status Sequence

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

Step	Sink Port	Source Port
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 ***Get_Status*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the present Source status which is provided. The Policy Engine tells the Protocol Layer to form a ***Status*** 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 ***Status*** Message.	Physical Layer appends a CRC and sends the ***Status*** 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 ***Status*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Status*** Message was successfully sent.
	The Source has informed the Sink of its present status.	

8.3.2.10.2.2 Source Gets Sink Status

Figure 8-27 shows an example sequence between a Source and a Sink where, after the Source has received an alert (see Section 8.3.2.10.1) that there has been a status change, the Source gets more details on the change.

Figure 8-28 Source Gets Sink Status

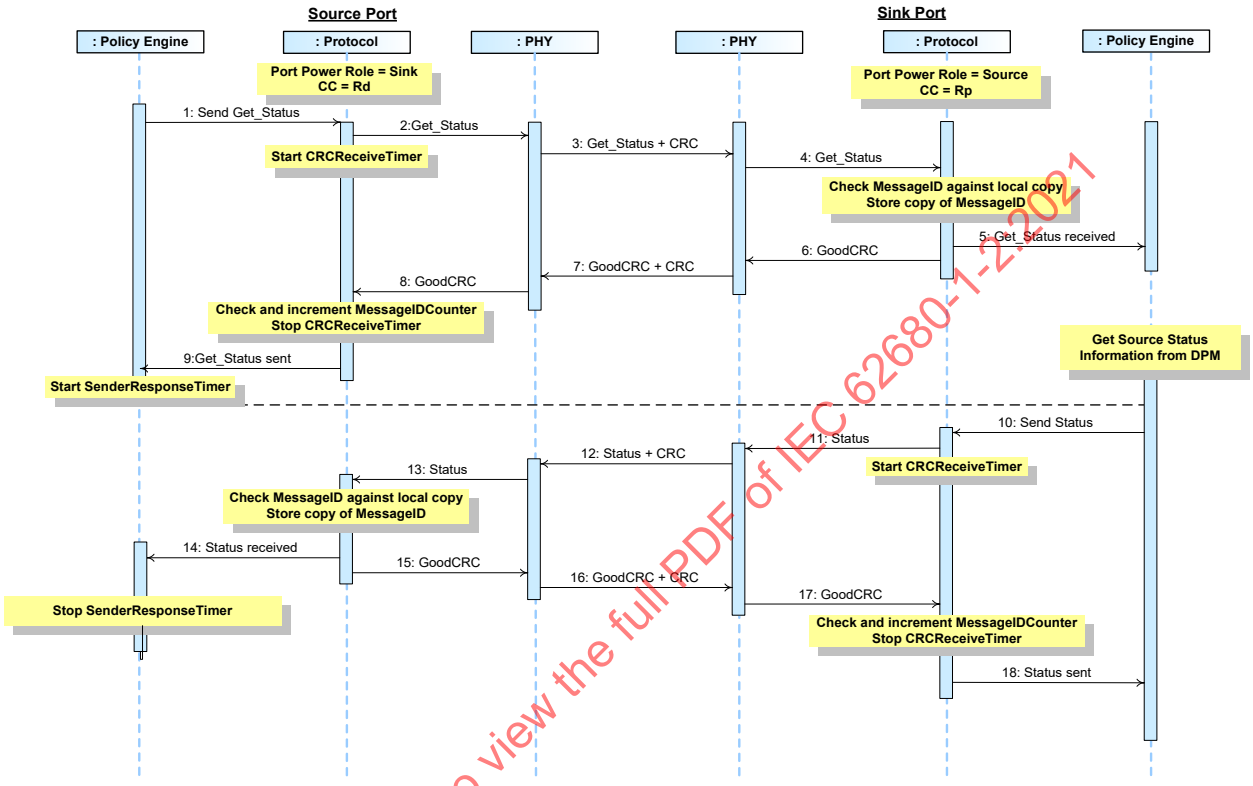


Table 8-27 below provides a detailed explanation of what happens at each labeled step in Figure 8-27 above.

Table 8-28 Steps for a Source getting Sink Status Sequence

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

Step	Source Port	Sink Port
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 ***Get_Status*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the present Source status which is provided. The Policy Engine tells the Protocol Layer to form a ***Status*** 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 ***Status*** Message.	Physical Layer appends a CRC and sends the ***Status*** 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 ***Status*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Status*** Message was successfully sent.
	The Sink has informed the Source of its present status.	

8.3.2.10.2.3 Sink Gets Source PPS Status

Figure 8-29 shows an example sequence between a Source and a Sink where, after the Sink has received an alert (see Section 8.3.2.10.1) that there has been a PPS status change, the Sink gets more details on the change.

Figure 8-29 Sink Gets Source PPS Status

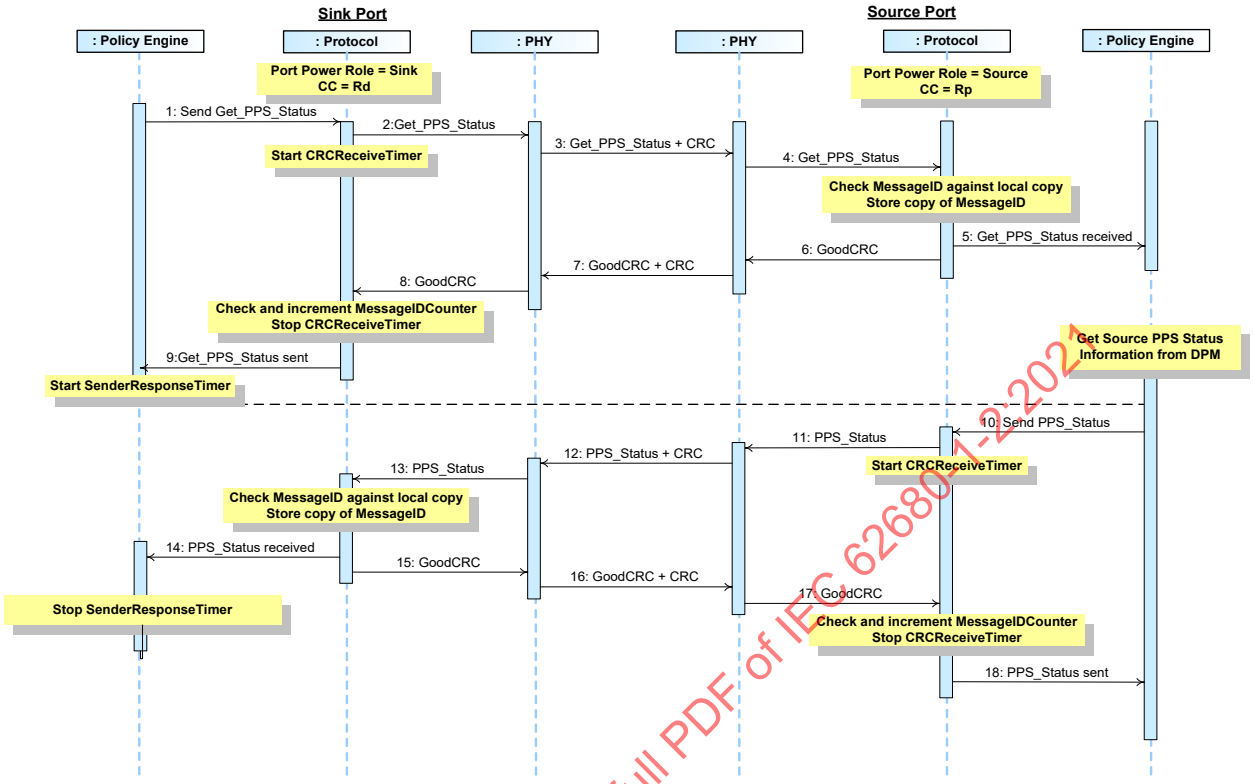


Table 8-29 below provides a detailed explanation of what happens at each labeled step in Figure 8-29 above.

Table 8-29 Steps for a Sink getting Source PPS status Sequence

Step	Sink Port	Source Port
1	The Port has ***Port Power Role*** set to Sink with the Rd pull down on its CC wire. Policy Engine directs the Protocol Layer to send a ***Get_PPS_Status*** Message.	The Port has ***Port Power Role*** set to Source and the Rp 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 ***Get_PPS_Status*** Message.	Physical Layer receives the ***Get_PPS_Status*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Get_Status*** 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 ***Get_PPS_Status*** 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.	

Step	Sink Port	Source Port
9	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Get_PPS_Status*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the present Source status which is provided. The Policy Engine tells the Protocol Layer to form a ***PPS_Status*** 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 ***PPS_Status*** Message.	Physical Layer appends a CRC and sends the ***PPS_Status*** 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 ***PPS_Status*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***PPS_Status*** Message was successfully sent.
	The Source has informed the Sink of its present PPS status.	

8.3.2.10.3 Source/Sink Capabilities

8.3.2.10.3.1 Sink Gets Source Capabilities

Figure 8-30 shows an example sequence between a Source and a Sink when the Sink gets the Source's capabilities.

Figure 8-30 Sink Gets Source's Capabilities

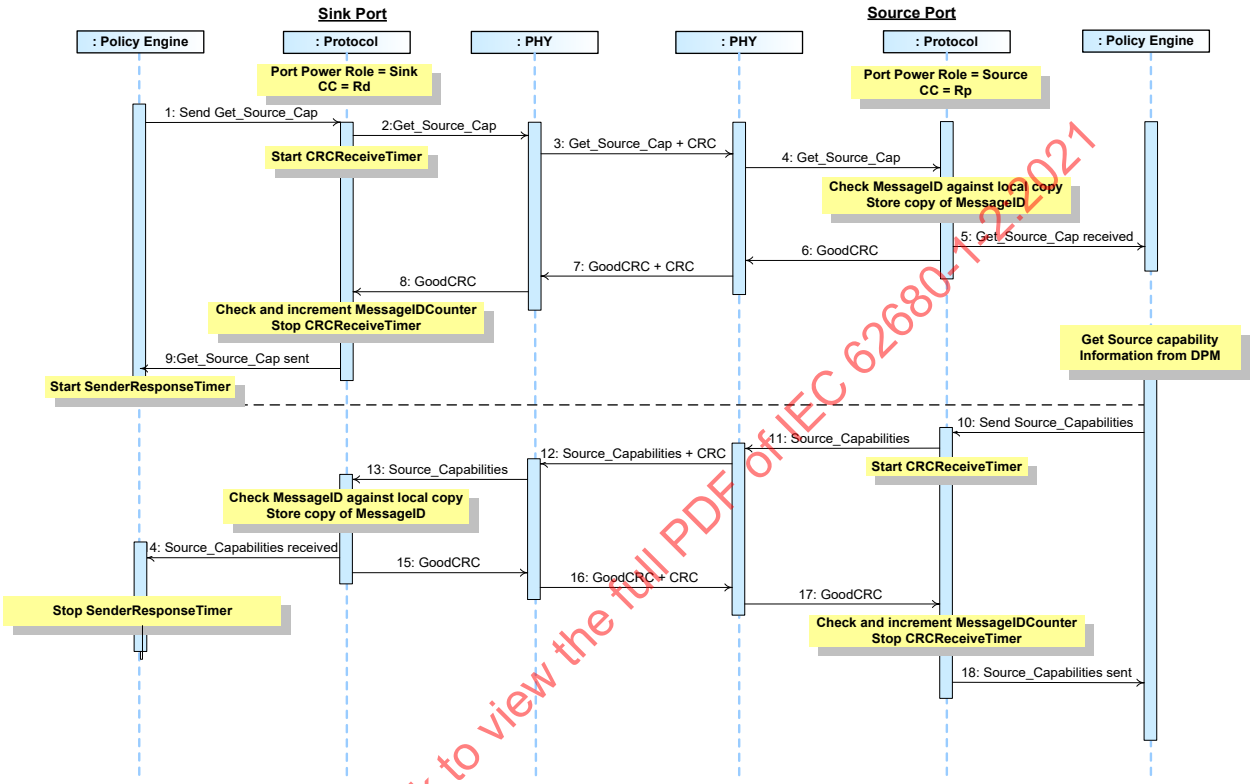


Table 8-30 below provides a detailed explanation of what happens at each labeled step in Figure 8-30 above.

Table 8-30 Steps for a Sink getting Source Capabilities Sequence

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

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

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

Figure 8-31 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 8-31 Dual-Role Source Gets Dual-Role Sink's Capabilities as a Source

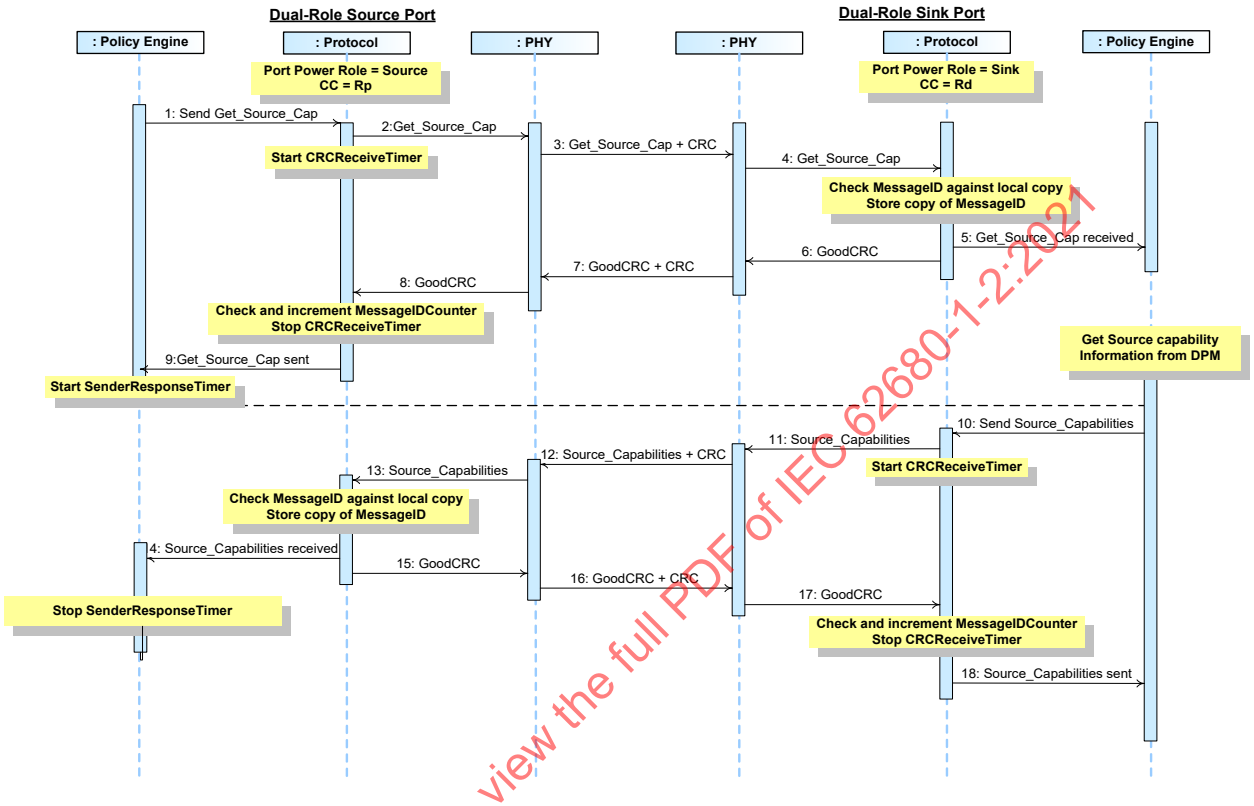


Table 8-31 below provides a detailed explanation of what happens at each labeled step in Figure 8-31 above.

Table 8-31 Steps for 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 ***Port Power Role*** set to Source and the Rp pull up on its CC wire. Policy Engine directs the Protocol Layer to send a ***Get_Source_Cap*** Message.	The Port has ***Port Power Role*** set to Sink with the Rd 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 ***Get_Source_Cap*** Message.	Physical Layer receives the ***Get_Source_Cap*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Get_Source_Cap*** 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 ***Get_Source_Cap*** Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.

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

8.3.2.10.3.3 Source Gets Sink Capabilities

Figure 8-32 shows an example sequence between a Source and a Sink when the Source gets the Sink's capabilities.

Figure 8-32 Source Gets Sink's Capabilities

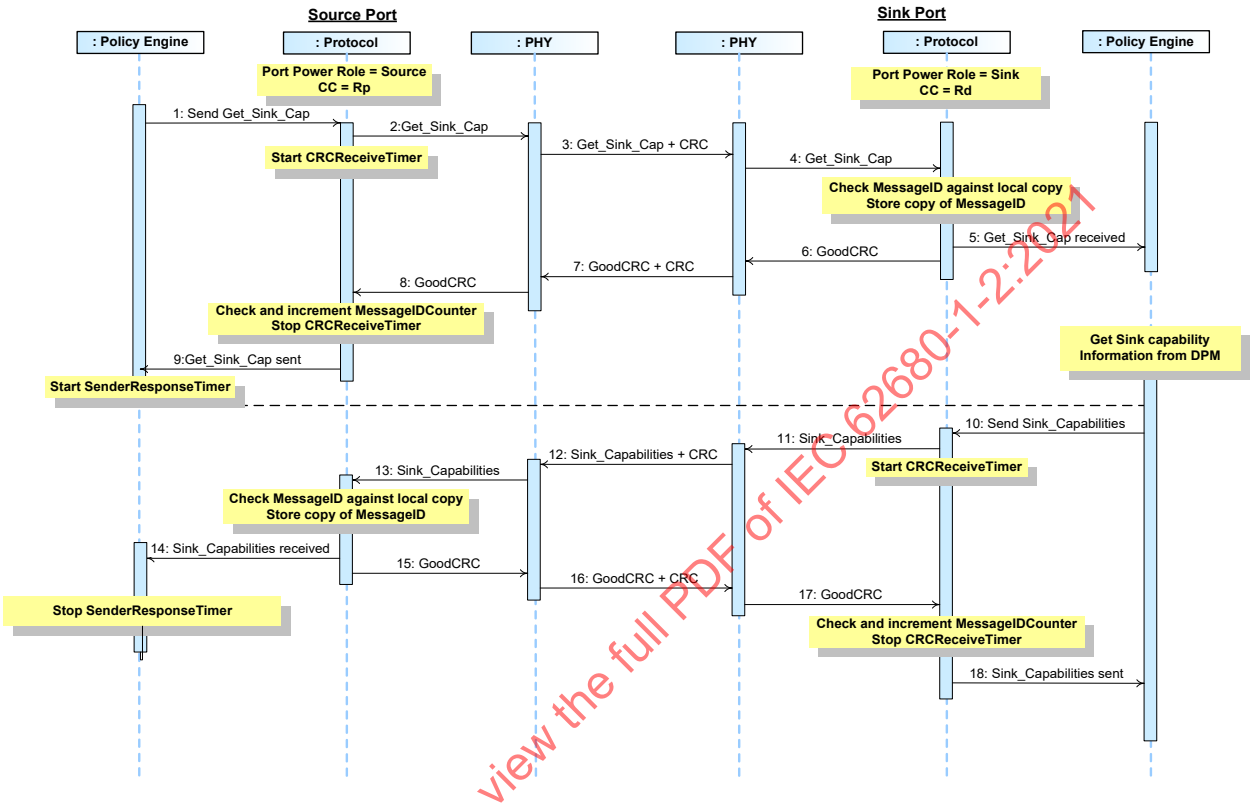


Table 8-32 below provides a detailed explanation of what happens at each labeled step in Figure 8-32 above.

Table 8-32 Steps for a Source getting Sink Capabilities Sequence

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

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

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

Figure 8-33 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 8-33 Dual-Role Sink Gets Dual-Role Source's Capabilities as a Sink

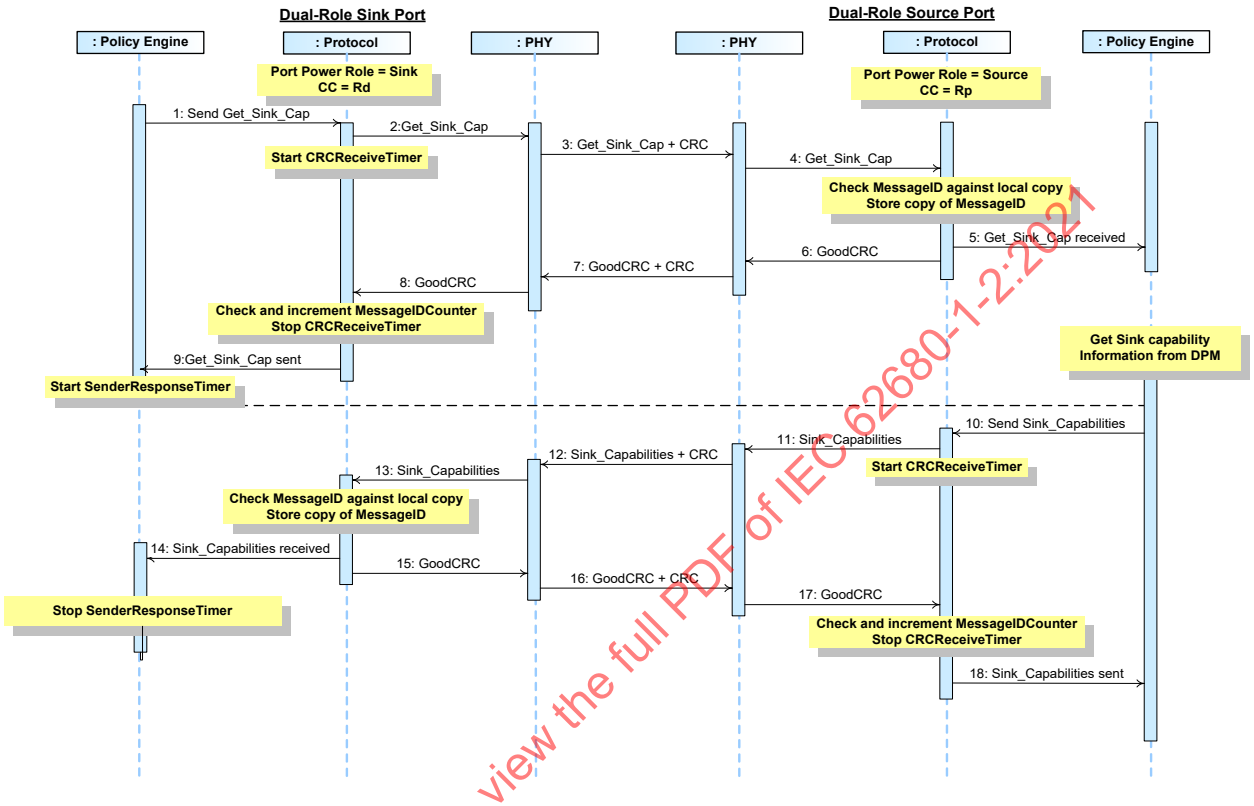


Table 8-33 below provides a detailed explanation of what happens at each labeled step in Figure 8-33 above.

Table 8-33 Steps for 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 ***Port Power Role*** set to Dual-Role Sink with the Rd pull down on its CC wire. Policy Engine directs the Protocol Layer to send a ***Get_Sink_Cap*** Message.	The Port has ***Port Power Role*** 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 ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Get_Sink_Cap*** Message.	Physical Layer receives the ***Get_Sink_Cap*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Get_Sink_Cap*** 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 ***Get_Sink_Cap*** Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.

Step	Dual-Role Sink Port	Dual-Role Source Port
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 ***Get_Sink_Cap*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
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 ***Sink_Capabilities*** 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 ***Sink_Capabilities*** Message.	Physical Layer appends a CRC and sends the ***Sink_Capabilities*** 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 ***Sink_Capabilities*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Sink_Capabilities*** Message was successfully sent.
	The Dual-Role Source has informed the Dual-Role Sink of its capabilities as a Sink.	

8.3.2.10.4 Extended Capabilities

8.3.2.10.4.1 Sink Gets Source Extended Capabilities

Figure 8-34 shows an example sequence between a Source and a Sink when the Sink gets the Source's extended capabilities.

Figure 8-34 Sink Gets Source's Extended Capabilities

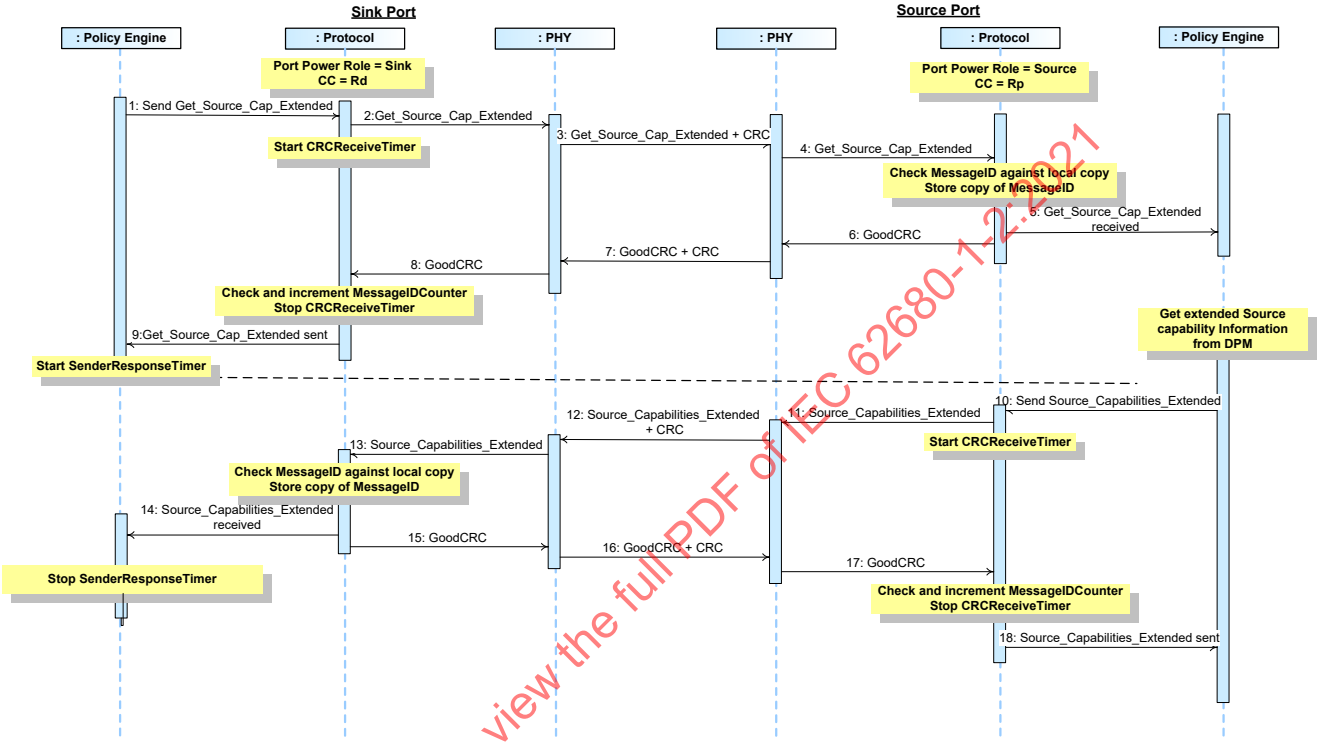


Table 8-34 below provides a detailed explanation of what happens at each labeled step in Figure 8-34 above.

Table 8-34 Steps for a Sink getting Source extended capabilities Sequence

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

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

8.3.2.10.4.2 Dual-Role Source Gets Source Capabilities Extended from a Dual-Role Sink

Figure 8-34 shows an example sequence between a Source and a Sink when the Dual-Role Source gets the Dual-Role Sink's extended capabilities as a Source.

Figure 8-35 Dual-Role Source Gets Dual-Role Sink's Extended Capabilities

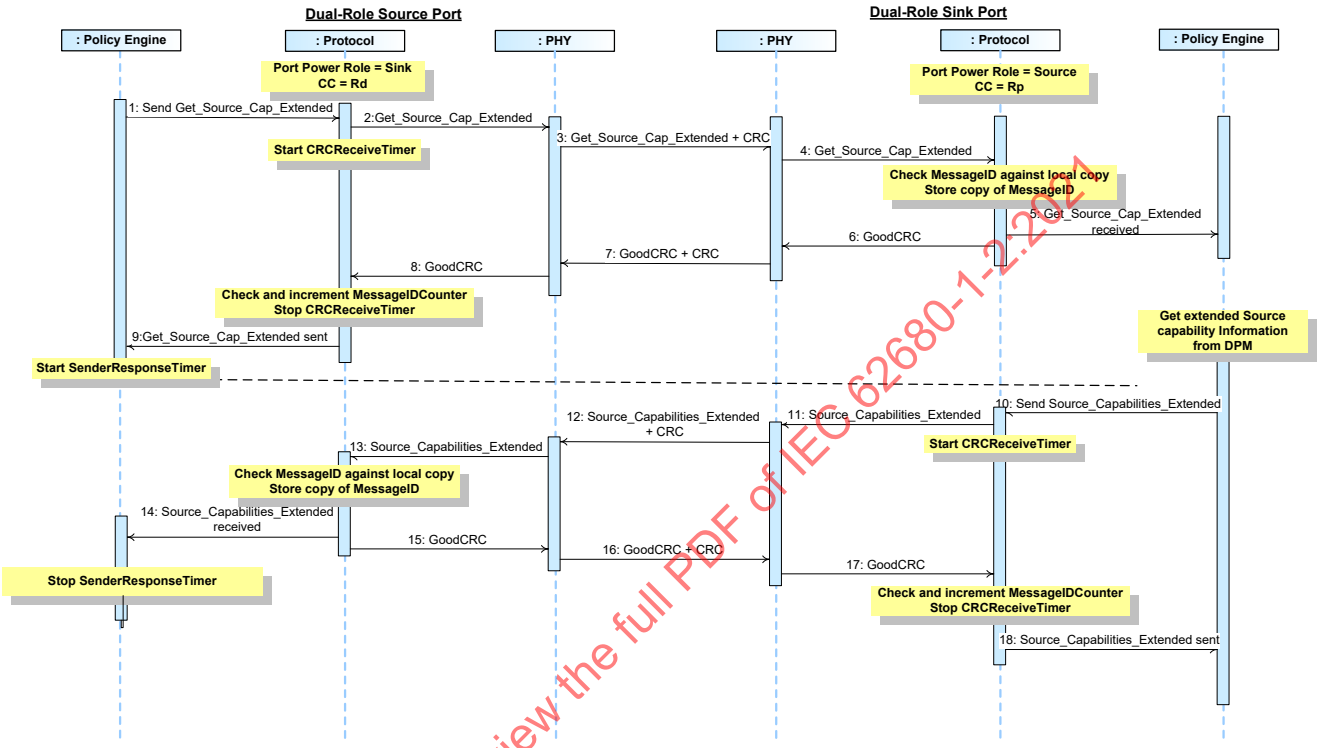


Table 8-34 below provides a detailed explanation of what happens at each labeled step in Figure 8-34 above.

Table 8-35 Steps for a Dual-Role Source getting Dual-Role Sink extended capabilities Sequence

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

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

8.3.2.10.5 Battery Capabilities and Status

8.3.2.10.5.1 Sink Gets Battery Capabilities

Figure 8-36 shows an example sequence between a Source and a Sink when the Sink gets the Source's Battery capabilities for a given Battery.

Figure 8-36 Sink Gets Source's Battery Capabilities

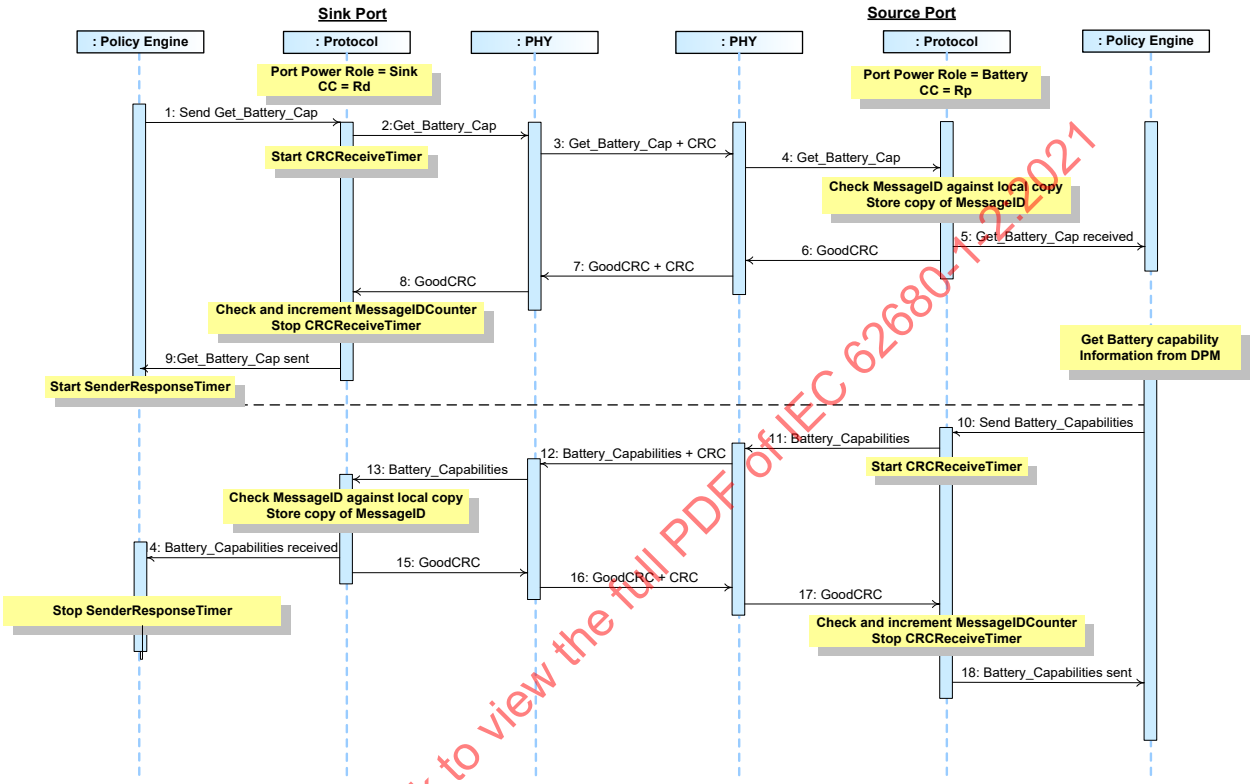


Table 8-36 below provides a detailed explanation of what happens at each labeled step in Figure 8-36 above.

Table 8-36 Steps for a Sink getting Source Battery capabilities Sequence

Step	Sink Port	Source Port
1	The Port has ***Port Power Role*** set to Sink with the Rd pull down on its CC wire. Policy Engine directs the Protocol Layer to send a ***Get_Battery_Cap*** Message containing the number of the Battery for which capabilities are being requested.	The Port has ***Port Power Role*** set to Source and the Rp 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 ***Get_Battery_Cap*** Message.	Physical Layer receives the ***Get_Battery_Cap*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Get_Battery_Cap*** 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 ***Get_Battery_Cap*** Message information to the Policy Engine that consumes it.

Step	Sink Port	Source Port
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 ***Get_Battery_Cap*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the present Source Battery capabilities, for the requested Battery number, which are provided. The Policy Engine tells the Protocol Layer to form a ***Battery_Capabilities*** 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 ***Battery_Capabilities*** Message.	Physical Layer appends a CRC and sends the ***Battery_Capabilities*** 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 ***Battery_Capabilities*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Battery_Capabilities*** Message was successfully sent.
	The Source has informed the Sink of the Battery capabilities for the requested Battery.	

8.3.2.10.5.2 Source Gets Battery Capabilities

Figure 8-37 shows an example sequence between a Source and a Sink when the Source gets the Sink's Battery capabilities for a given Battery.

Figure 8-37 Source Gets Sink's Battery Capabilities

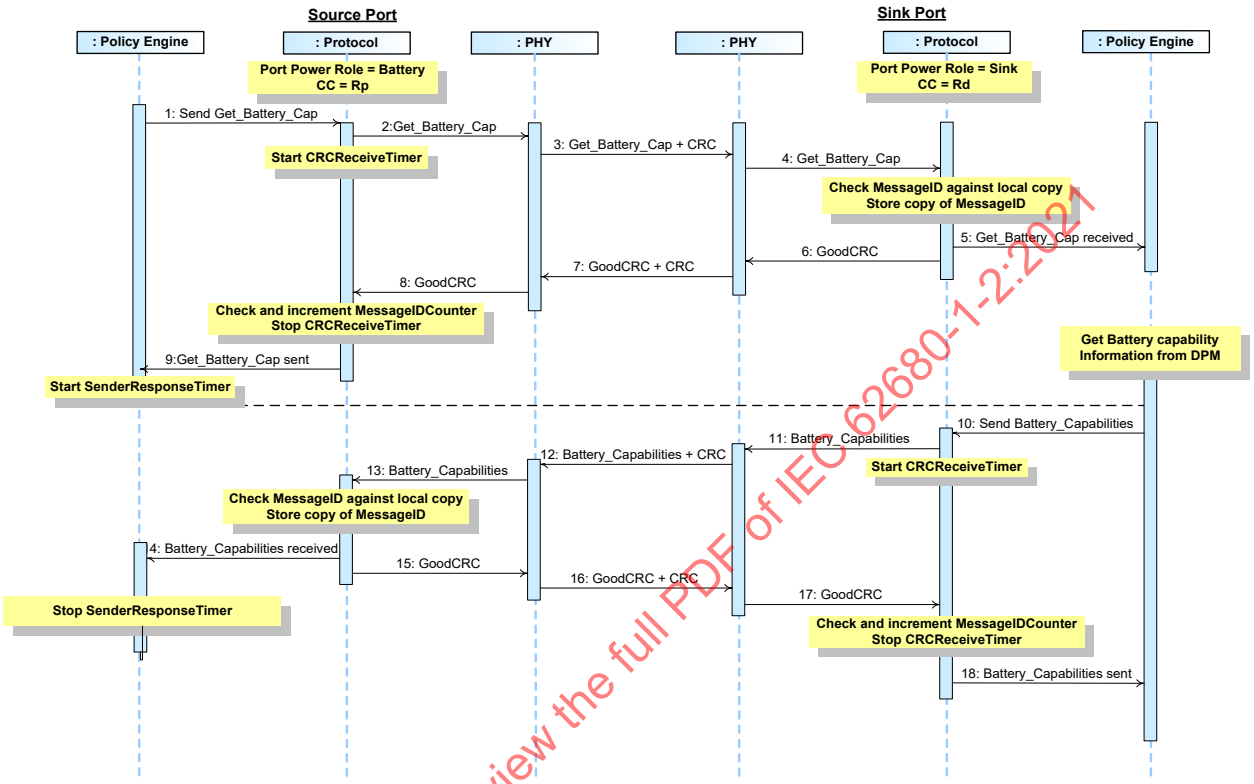


Table 8-37 below provides a detailed explanation of what happens at each labeled step in Figure 8-37 above.

Table 8-37 Steps for a Source getting Sink Battery capabilities Sequence

Step	Source Port	Sink Port
1	The Port has ***Port Power Role*** set to Source and the Rp pull up on its CC wire. Policy Engine directs the Protocol Layer to send a ***Get_Battery_Cap*** Message containing the number of the Battery for which capabilities are being requested.	The Port has ***Port Power Role*** set to Sink with the Rd 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 ***Get_Battery_Cap*** Message.	Physical Layer receives the ***Get_Battery_Cap*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Get_Battery_Cap*** 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 ***Get_Battery_Cap*** Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.

Step	Source Port	Sink Port
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 ***Get_Battery_Cap*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the present Source Battery capabilities, for the requested Battery number, which are provided. The Policy Engine tells the Protocol Layer to form a ***Battery_Capabilities*** 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 ***Battery_Capabilities*** Message.	Physical Layer appends a CRC and sends the ***Battery_Capabilities*** 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 ***Battery_Capabilities*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Battery_Capabilities*** Message was successfully sent.
	The Sink has informed the Source of the Battery capabilities for the requested Battery.	

8.3.2.10.5.3 Sink Gets Battery Status

Figure 8-38 shows an example sequence between a Source and a Sink when the Sink gets the Source's Battery status for a given Battery.

Figure 8-38 Sink Gets Source's Battery Status

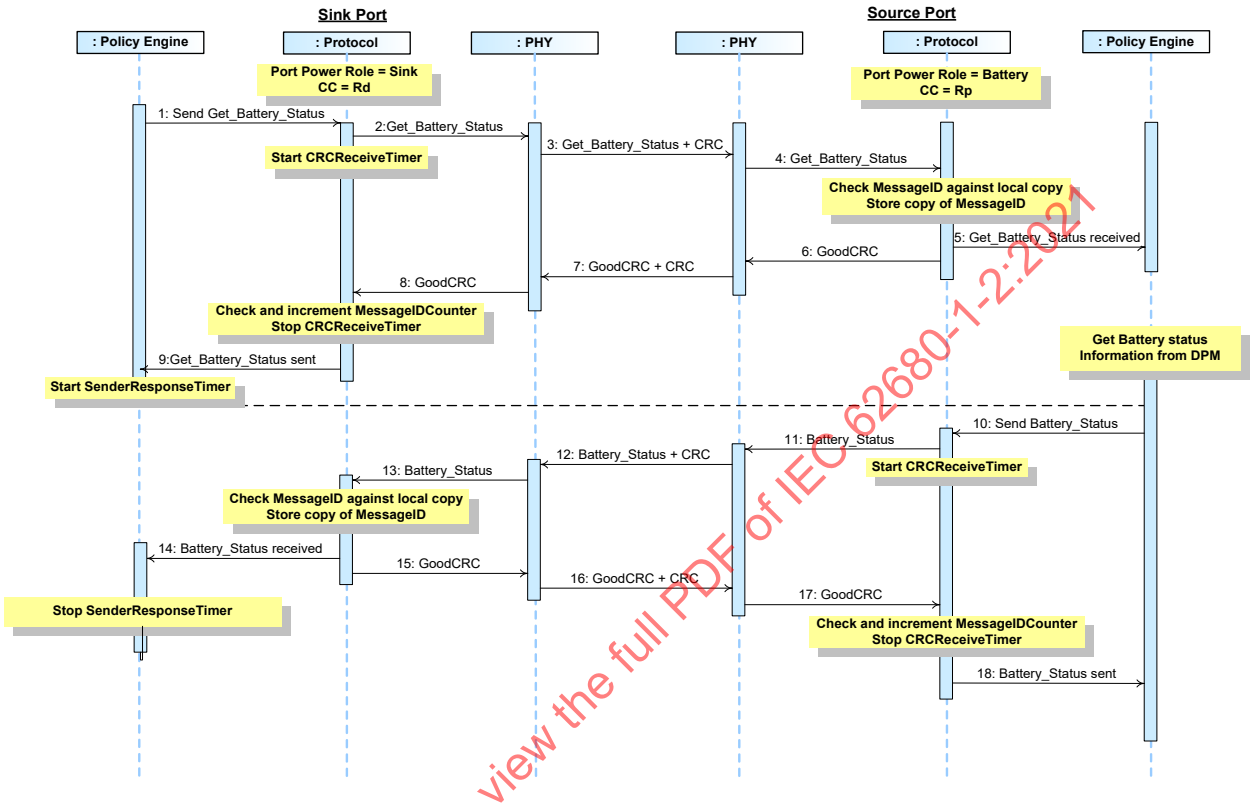


Table 8-38 below provides a detailed explanation of what happens at each labeled step in Figure 8-38 above.

Table 8-38 Steps for a Sink getting Source Battery status Sequence

Step	Sink Port	Source Port
1	The Port has ***Port Power Role*** set to Sink with the Rd pull down on its CC wire. Policy Engine directs the Protocol Layer to send a ***Get_Battery_Status*** Message containing the number of the Battery for which status is being requested.	The Port has ***Port Power Role*** set to Source and the Rp 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 ***Get_Battery_Status*** Message.	Physical Layer receives the ***Get_Battery_Status*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Get_Battery_Status*** 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 ***Get_Battery_Status*** Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.

Step	Sink Port	Source Port
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 ***Get_Battery_Status*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the present Source Battery status, for the requested Battery number, which are provided. The Policy Engine tells the Protocol Layer to form a ***Battery_Status*** 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 ***Battery_Status*** Message.	Physical Layer appends a CRC and sends the ***Battery_Status*** 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 ***Battery_Status*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Battery_Status*** Message was successfully sent.
	The Source has informed the Sink of the Battery status for the requested Battery.	

8.3.2.10.5.4 Source Gets Battery Status

Figure 8-39 shows an example sequence between a Source and a Sink when the Source gets the Sink's Battery status for a given Battery.

Figure 8-39 Source Gets Sink's Battery Status

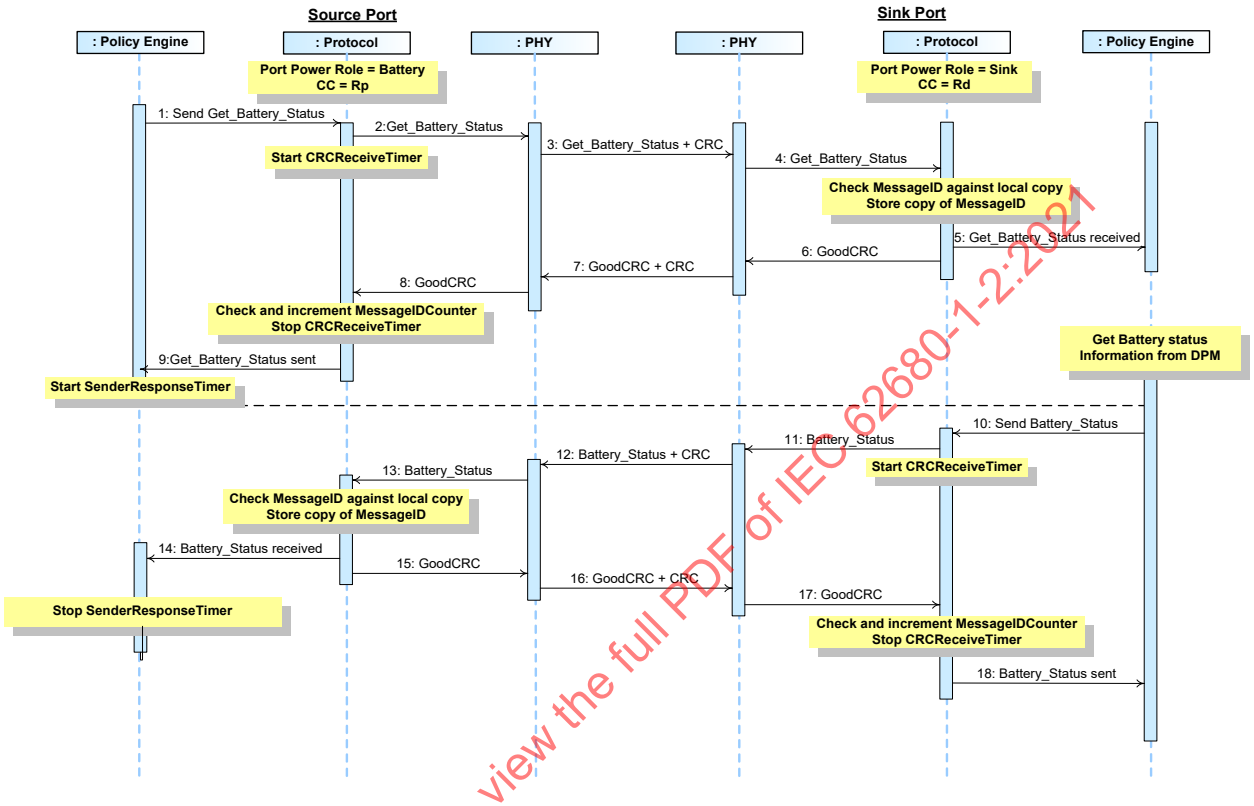


Table 8-39 below provides a detailed explanation of what happens at each labeled step in Figure 8-39 above.

Table 8-39 Steps for a Source getting Sink Battery status Sequence

Step	Source Port	Sink Port
1	The Port has ***Port Power Role*** set to Source and the Rp pull up on its CC wire. Policy Engine directs the Protocol Layer to send a ***Get_Battery_Status*** Message containing the number of the Battery for which status is being requested.	The Port has ***Port Power Role*** set to Sink with the Rd 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 ***Get_Battery_Status*** Message.	Physical Layer receives the ***Get_Battery_Status*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Get_Battery_Status*** 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 ***Get_Battery_Status*** Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.

Step	Source Port	Sink Port
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 ***Get_Battery_Status*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the present Source Battery status, for the requested Battery number, which are provided. The Policy Engine tells the Protocol Layer to form a ***Battery_Status*** 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 ***Battery_Status*** Message.	Physical Layer appends a CRC and sends the ***Battery_Status*** 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 ***Battery_Status*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Battery_Status*** Message was successfully sent.
	The Sink has informed the Source of the Battery status for the requested Battery.	

8.3.2.10.6 Manufacturer Information

8.3.2.10.6.1 Source Gets Port Manufacturer Information from a Sink

Figure 8-40 shows an example sequence between a Source and a Sink when the Source gets the Sink's Manufacturer information for the Port.

Figure 8-40 Source Gets Sink's Port Manufacturer Information

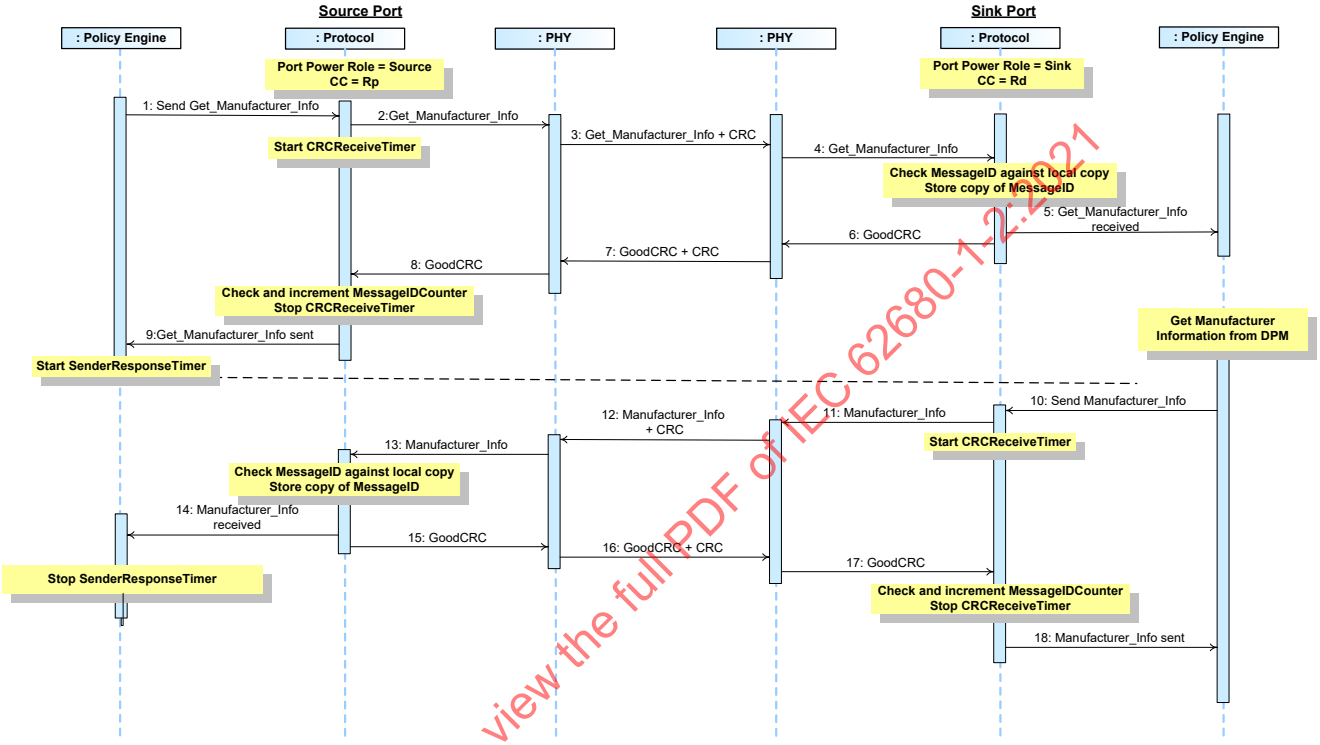


Table 8-37 below provides a detailed explanation of what happens at each labeled step in Figure 8-40 above.

Table 8-40 Steps for a Source getting Sink's Port Manufacturer Information Sequence

Step	Source Port	Sink Port
1	The Port has ***Port Power Role*** set to Source and the Rp pull up on its CC wire. Policy Engine directs the Protocol Layer to send a ***Get_Manufacturer_Info*** Message with a request for Port information.	The Port has ***Port Power Role*** set to Sink with the Rd 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 ***Get_Manufacturer_Info*** Message.	Physical Layer receives the ***Get_Manufacturer_Info*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Get_Manufacturer_Info*** 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 ***Get_Manufacturer_Info*** Message information to the Policy Engine that consumes it.

Step	Source Port	Sink Port
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 ***Get_Manufacturer_Info*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the Port's manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a ***Manufacturer_Info*** 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 ***Manufacturer_Info*** Message.	Physical Layer appends a CRC and sends the ***Manufacturer_Info*** 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 ***Manufacturer_Info*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Manufacturer_Info*** Message was successfully sent.
	The Sink has informed the Source of the manufacturer information for the Port.	

8.3.2.10.6.2 Sink Gets Port Manufacturer Information from a Source

Figure 8-41 shows an example sequence between a Source and a Sink when the Source gets the Sink's Manufacturer information for the Port.

Figure 8-41 Sink Gets Source's Port Manufacturer Information

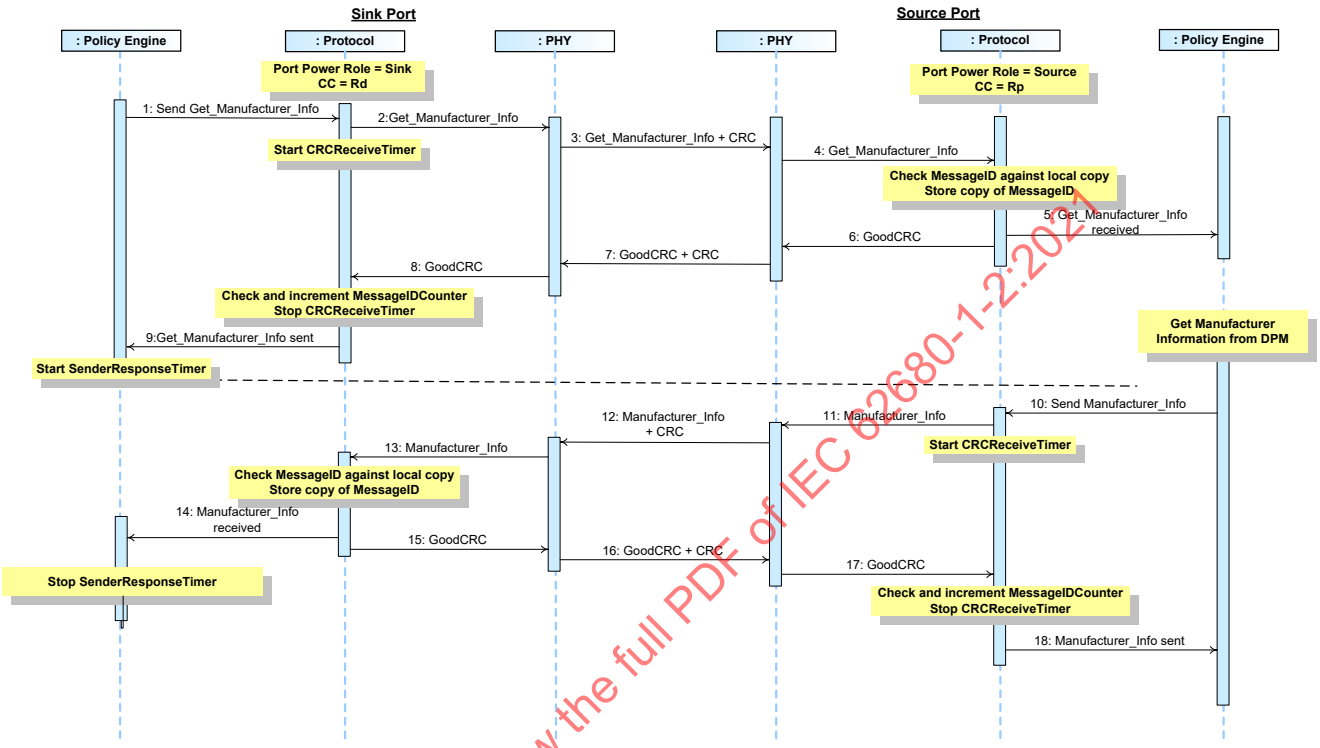


Table 8-41 below provides a detailed explanation of what happens at each labeled step in Figure 8-41 above.

Table 8-41 Steps for a Source getting Sink's Port Manufacturer Information Sequence

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

Step	Sink Port	Source Port
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 ***Get_Manufacturer_Info*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the Port's manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a ***Manufacturer_Info*** 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 ***Manufacturer_Info*** Message.	Physical Layer appends a CRC and sends the ***Manufacturer_Info*** 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 ***Manufacturer_Info*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Manufacturer_Info*** Message was successfully sent.
	The Sink has informed the Source of the manufacturer information for the Port.	

8.3.2.10.6.3 Source Gets Battery Manufacturer Information from a Sink

Figure 8-42 shows an example sequence between a Source and a Sink when the Source gets the Sink's Manufacturer information for one of its Batteries.

Figure 8-42 Source Gets Sink's Battery Manufacturer Information

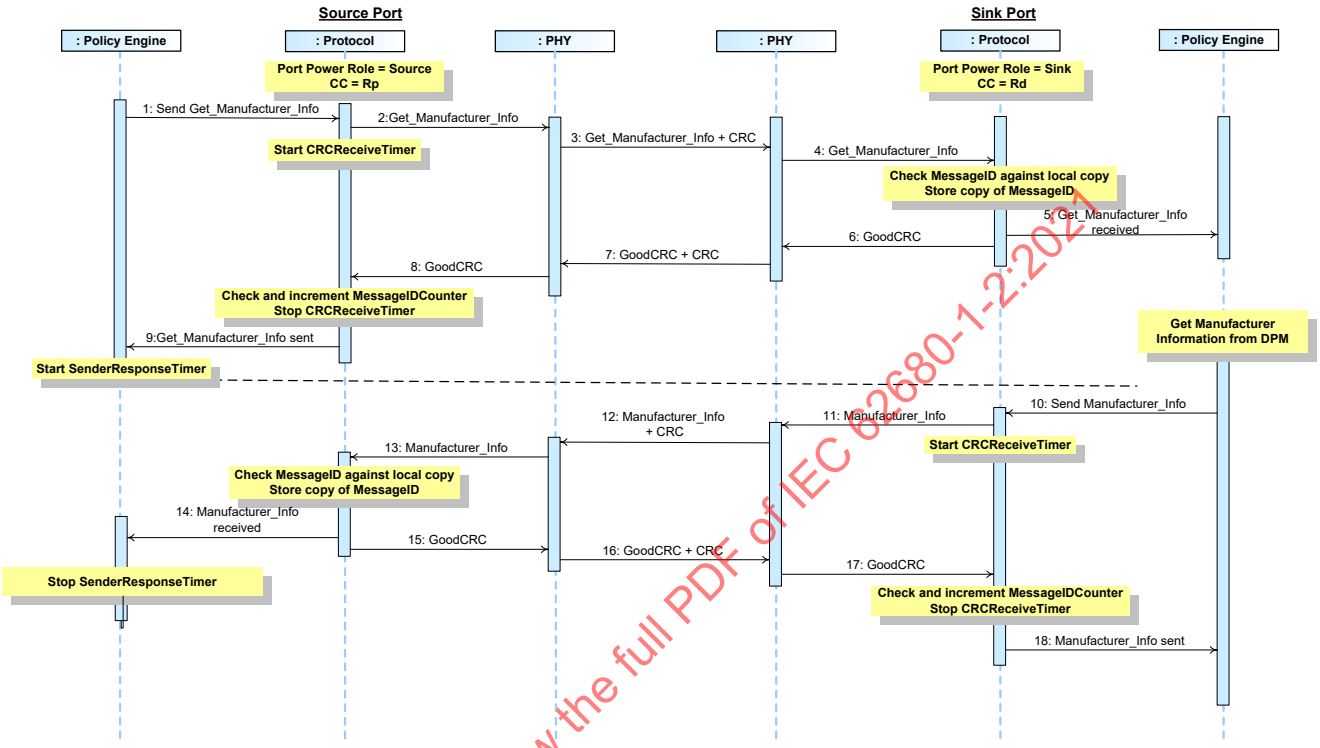


Table 8-42 below provides a detailed explanation of what happens at each labeled step in Figure 8-42 above.

Table 8-42 Steps for a Source getting Sink's Battery Manufacturer Information Sequence

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

Step	Source Port	Sink Port
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 ***Get_Manufacturer_Info*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the Battery's manufacturer information for a given Battery which is provided. The Policy Engine tells the Protocol Layer to form a ***Manufacturer_Info*** 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 ***Manufacturer_Info*** Message.	Physical Layer appends a CRC and sends the ***Manufacturer_Info*** 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 ***Manufacturer_Info*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Manufacturer_Info*** Message was successfully sent.
	The Sink has informed the Source of the manufacturer information for the requested Battery.	

8.3.2.10.6.4 Sink Gets Battery Manufacturer Information from a Source

Figure 8-43 shows an example sequence between a Source and a Sink when the Source gets the Sink's Manufacturer information for the Port.

Figure 8-43 Sink Gets Source's Battery Manufacturer Information

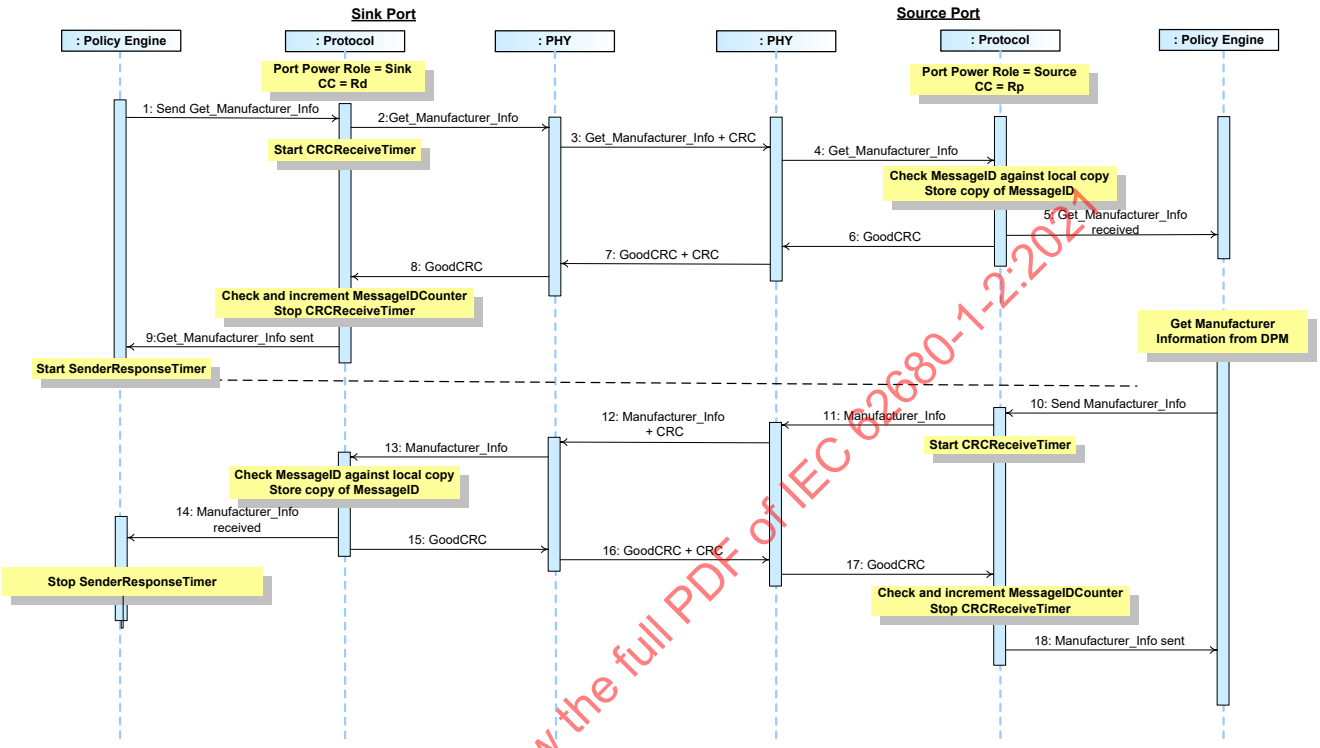


Table 8-43 below provides a detailed explanation of what happens at each labeled step in Figure 8-43 above.

Table 8-43 Steps for a Source getting Sink's Battery Manufacturer Information Sequence

Step	Sink Port	Source Port
1	The Port has ***Port Power Role*** set to Sink with the Rd pull down on its CC wire. Policy Engine directs the Protocol Layer to send a ***Get_Manufacturer_Info*** Message with a request for Battery information for a given Battery.	The Port has ***Port Power Role*** set to Source and the Rp 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 ***Get_Manufacturer_Info*** Message.	Physical Layer receives the ***Get_Manufacturer_Info*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Get_Manufacturer_Info*** 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 ***Get_Manufacturer_Info*** Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.

Step	Sink Port	Source Port
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 ***Get_Manufacturer_Info*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the Battery's manufacturer information for a given Battery which is provided. The Policy Engine tells the Protocol Layer to form a ***Manufacturer_Info*** 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 ***Manufacturer_Info*** Message.	Physical Layer appends a CRC and sends the ***Manufacturer_Info*** 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 ***Manufacturer_Info*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Manufacturer_Info*** Message was successfully sent.
	The Sink has informed the Source of the manufacturer information for the requested Battery.	

8.3.2.10.6.5 VCONN Source Gets Manufacturer Information from a Cable Plug

Figure 8-44 shows an example sequence between a VCONN Source (Source or Sink) and a Cable Plug when the VCONN Source gets the Cable Plug's Manufacturer information.

Figure 8-44 VCONN Source Gets Cable Plug's Manufacturer Information

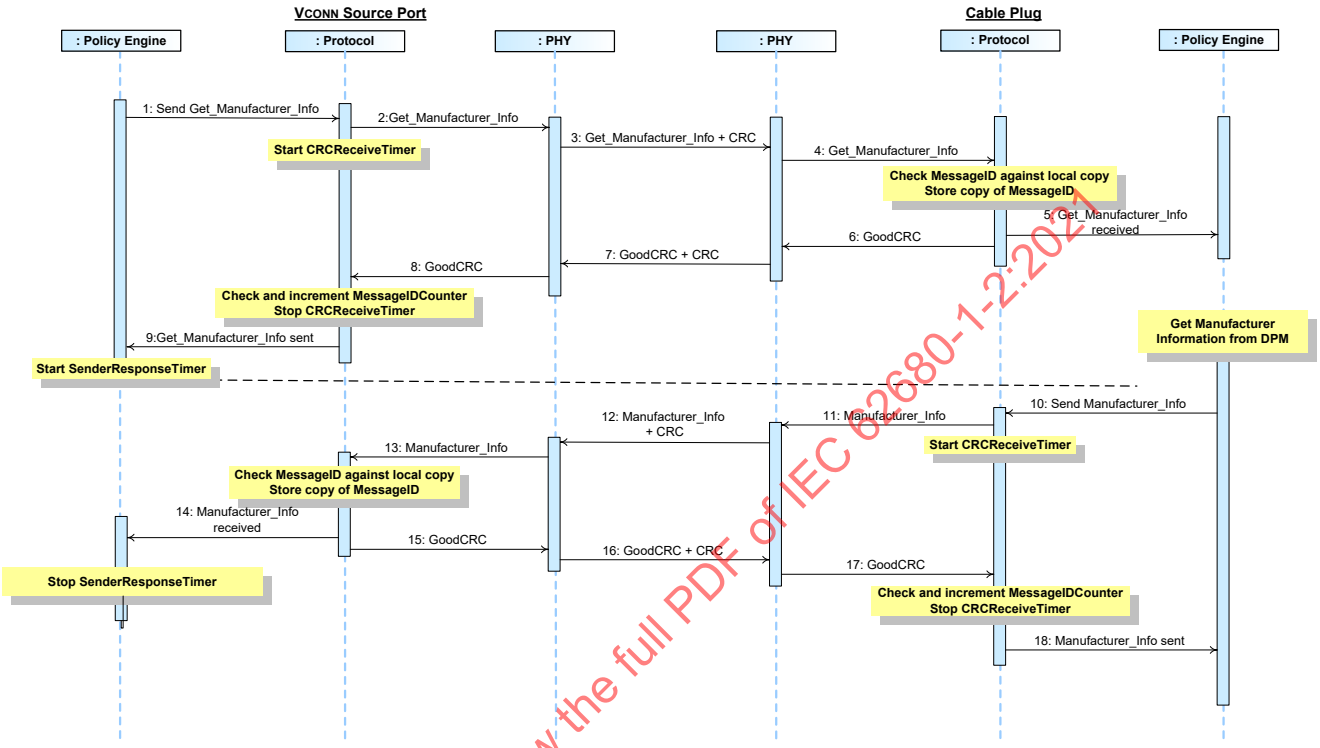


Table 8-44 below provides a detailed explanation of what happens at each labeled step in Figure 8-44 above.

Table 8-44 Steps for a VCONN Source getting Sink's Port Manufacturer Information Sequence

Step	VCONN Source	Cable Plug
1	The Port is currently acting as the VCONN Source. Policy Engine directs the Protocol Layer to send a ***Get_Manufacturer_Info*** Message with a request for Port information.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Get_Manufacturer_Info*** Message.	Physical Layer receives the ***Get_Manufacturer_Info*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Get_Manufacturer_Info*** 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 ***Get_Manufacturer_Info*** 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.

Step	VCONN Source	Cable Plug
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 ***Get_Manufacturer_Info*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the Cable Plug's manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a ***Manufacturer_Info*** 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 ***Manufacturer_Info*** Message.	Physical Layer appends a CRC and sends the ***Manufacturer_Info*** 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 ***Manufacturer_Info*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Manufacturer_Info*** Message was successfully sent.
	The Cable Plug has informed the Source of its manufacturer information.	

8.3.2.10.7 Country Codes

8.3.2.10.7.1 Source Gets Country Codes from a Sink

Figure 8-45 shows an example sequence between a Source and a Sink when the Source gets the Sink's Country Codes.

Figure 8-45 Source Gets Sink's Country Codes

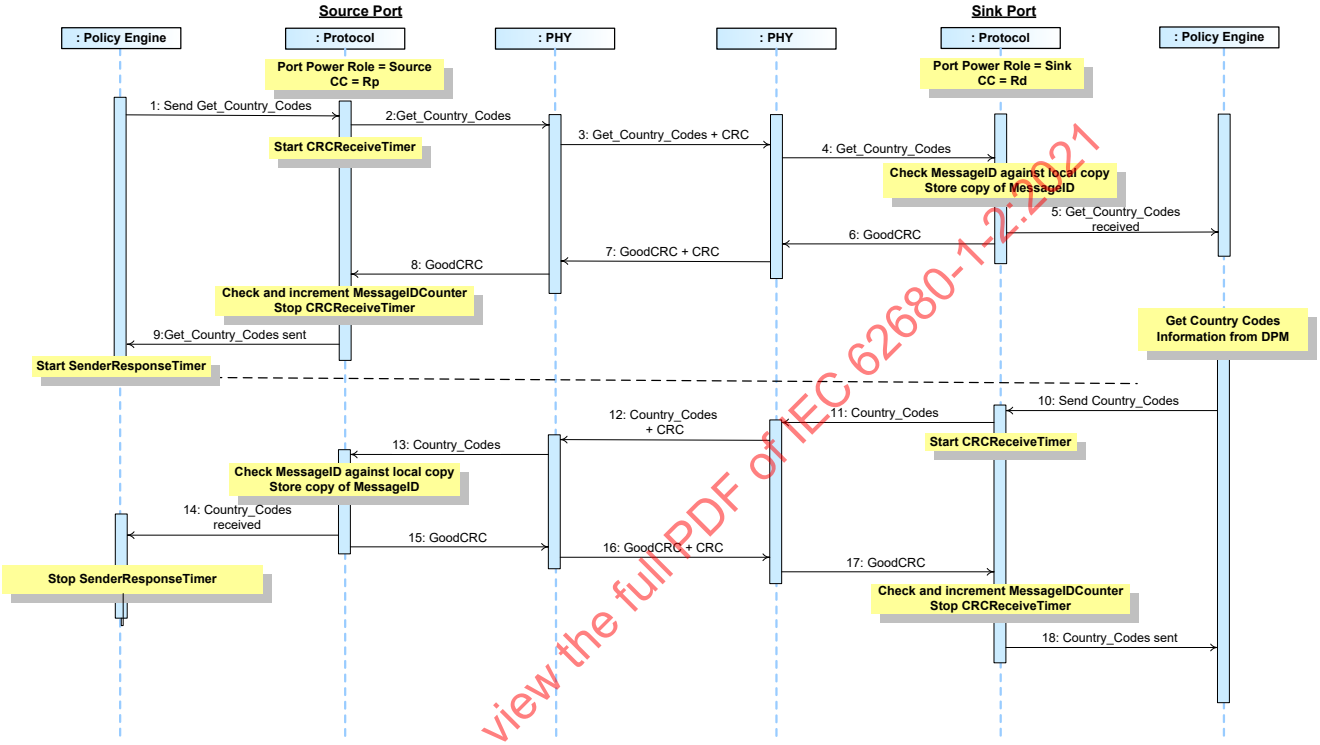


Table 8-45 below provides a detailed explanation of what happens at each labeled step in Figure 8-45 above.

Table 8-45 Steps for a Source getting Country Codes Sequence

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

Step	Source Port	Sink Port
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 *Get_Country_Codes* Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the Port's manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a ***Country_Codes*** 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 ***Country_Codes*** Message.	Physical Layer appends a CRC and sends the ***Country_Codes*** 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 ***Country_Codes*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Country_Codes*** Message was successfully sent.
	The Sink has informed the Source of the country codes.	

8.3.2.10.7.2 Sink Gets Country Codes from a Source

Figure 8-46 shows an example sequence between a Source and a Sink when the Source gets the Sink's country codes.

Figure 8-46 Sink Gets Source's Country Codes

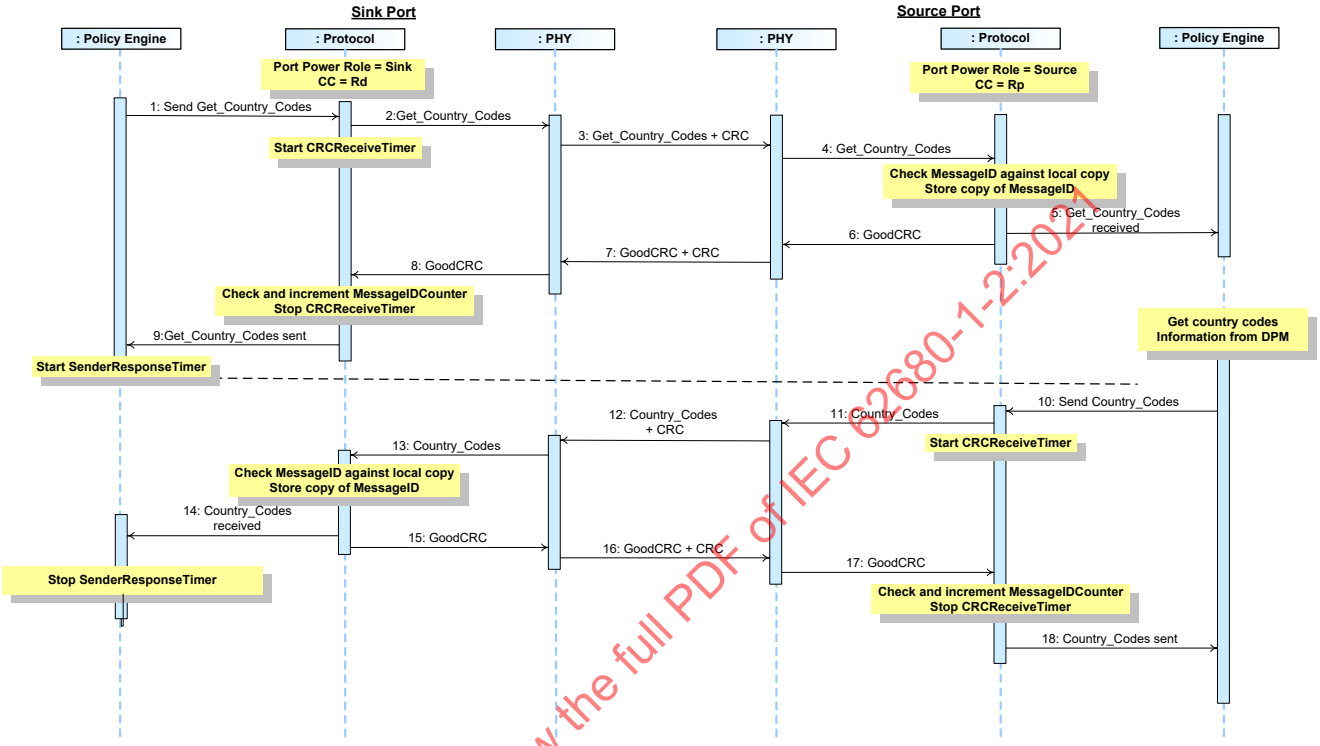


Table 8-46 below provides a detailed explanation of what happens at each labeled step in Figure 8-46 above.

Table 8-46 Steps for a Source getting Sink's Country Codes Sequence

Step	Sink Port	Source Port
1	The Port has ***Port Power Role*** set to Sink with the Rd pull down on its CC wire. Policy Engine directs the Protocol Layer to send a *Get_Country_Codes* Message with a request for Port information.	The Port has ***Port Power Role*** set to Source and the Rp 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 *Get_Country_Codes* Message.	Physical Layer receives the *Get_Country_Codes* Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the *Get_Country_Codes* 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 *Get_Country_Codes* 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.

Step	Sink Port	Source Port
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 *Get_Country_Codes* Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the Port's manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a ***Country_Codes*** 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 ***Country_Codes*** Message.	Physical Layer appends a CRC and sends the ***Country_Codes*** 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 ***Country_Codes*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Country_Codes*** Message was successfully sent.
	The Sink has informed the Source of the country codes.	

8.3.2.10.7.3 VCONN Source Gets Country Codes from a Cable Plug

Figure 8-47 shows an example sequence between a VCONN Source (Source or Sink) and a Cable Plug when the VCONN Source gets the Cable Plug's Country Codes.

Figure 8-47 VCONN Source Gets Cable Plug's Country Codes

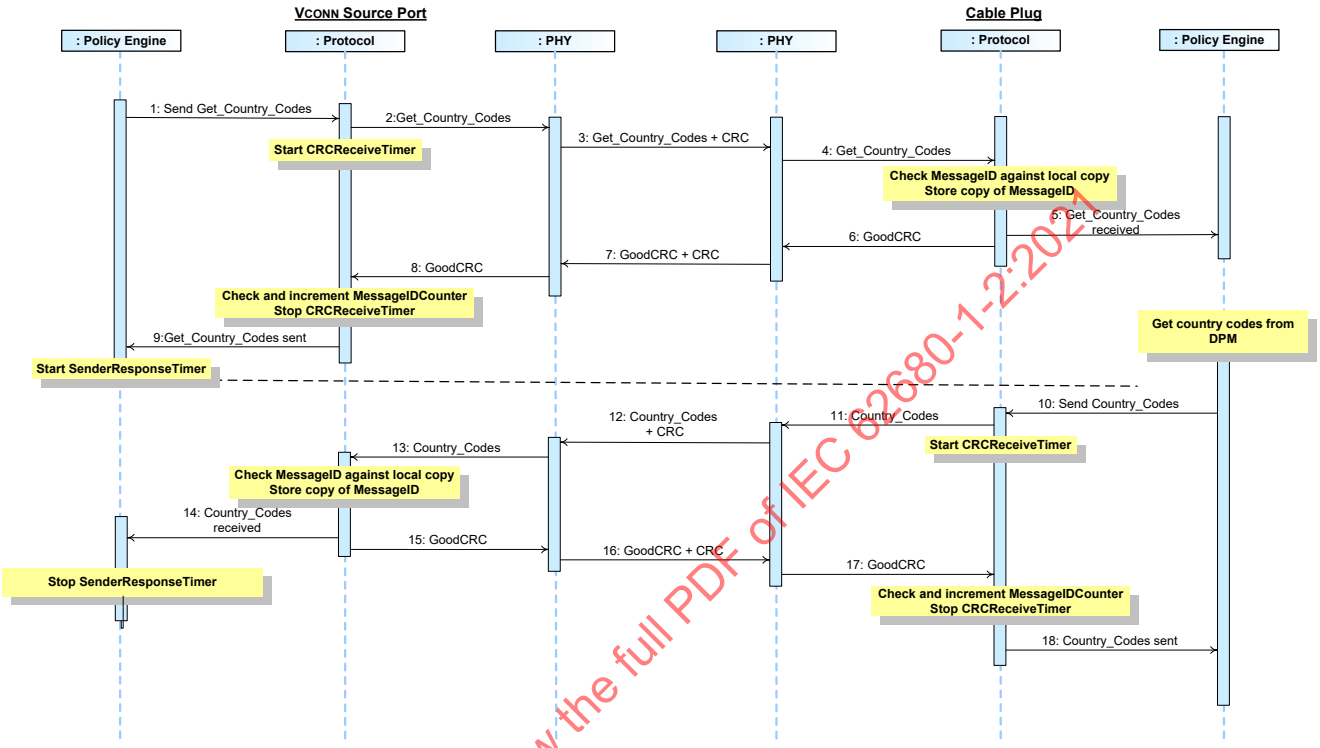


Table 8-47 below provides a detailed explanation of what happens at each labeled step in Figure 8-47 above.

Table 8-47 Steps for a VCONN Source getting Sink's Country Codes Sequence

Step	VCONN Source	Cable Plug
1	The Port is currently acting as the VCONN Source. Policy Engine directs the Protocol Layer to send a *Get_Country_Codes* Message with a request for Port information.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the *Get_Country_Codes* Message.	Physical Layer receives the *Get_Country_Codes* Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the *Get_Country_Codes* 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 *Get_Country_Codes* 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.

Step	VCONN Source	Cable Plug
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 *Get_Country_Codes* Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the Cable Plug's manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a ***Country_Codes*** 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 ***Country_Codes*** Message.	Physical Layer appends a CRC and sends the ***Country_Codes*** 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 ***Country_Codes*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Country_Codes*** Message was successfully sent.
	The Cable Plug has informed the Source of its country codes.	

8.3.2.10.8 Country Information

8.3.2.10.8.1 Source Gets Country Information from a Sink

Figure 8-48 shows an example sequence between a Source and a Sink when the Source gets the Sink's country information.

Figure 8-48 Source Gets Sink's Country Information

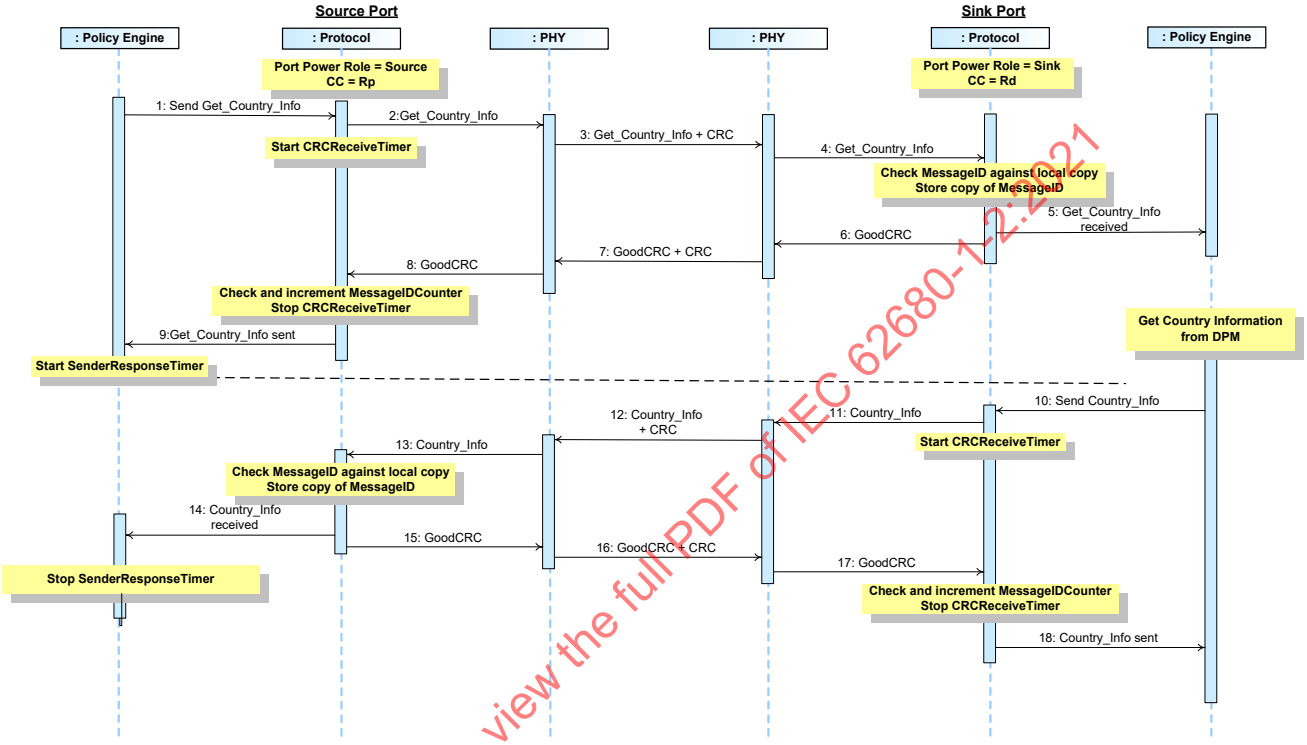


Table 8-48 below provides a detailed explanation of what happens at each labeled step in Figure 8-48 above.

Table 8-48 Steps for a Source getting Country Information Sequence

Step	Source Port	Sink Port
1	The Port has ***Port Power Role*** set to Source and the Rp pull up on its CC wire. Policy Engine directs the Protocol Layer to send a ***Get_Country_Info*** Message with a request for Port information for a specific Country Code.	The Port has ***Port Power Role*** set to Sink with the Rd 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 ***Get_Country_Info*** Message.	Physical Layer receives the ***Get_Country_Info*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Get_Country_Info*** 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 ***Get_Country_Info*** Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.

Step	Source Port	Sink Port
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 ***Get_Country_Info*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the Port's manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a ***Country_Info*** 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 ***Country_Info*** Message.	Physical Layer appends a CRC and sends the ***Country_Info*** 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 ***Country_Info*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Country_Info*** Message was successfully sent.
	The Sink has informed the Source of the country information.	

8.3.2.10.8.2 Sink Gets Country Information from a Source

Figure 8-49 shows an example sequence between a Source and a Sink when the Source gets the Sink's country codes.

Figure 8-49 Sink Gets Source's Country Information

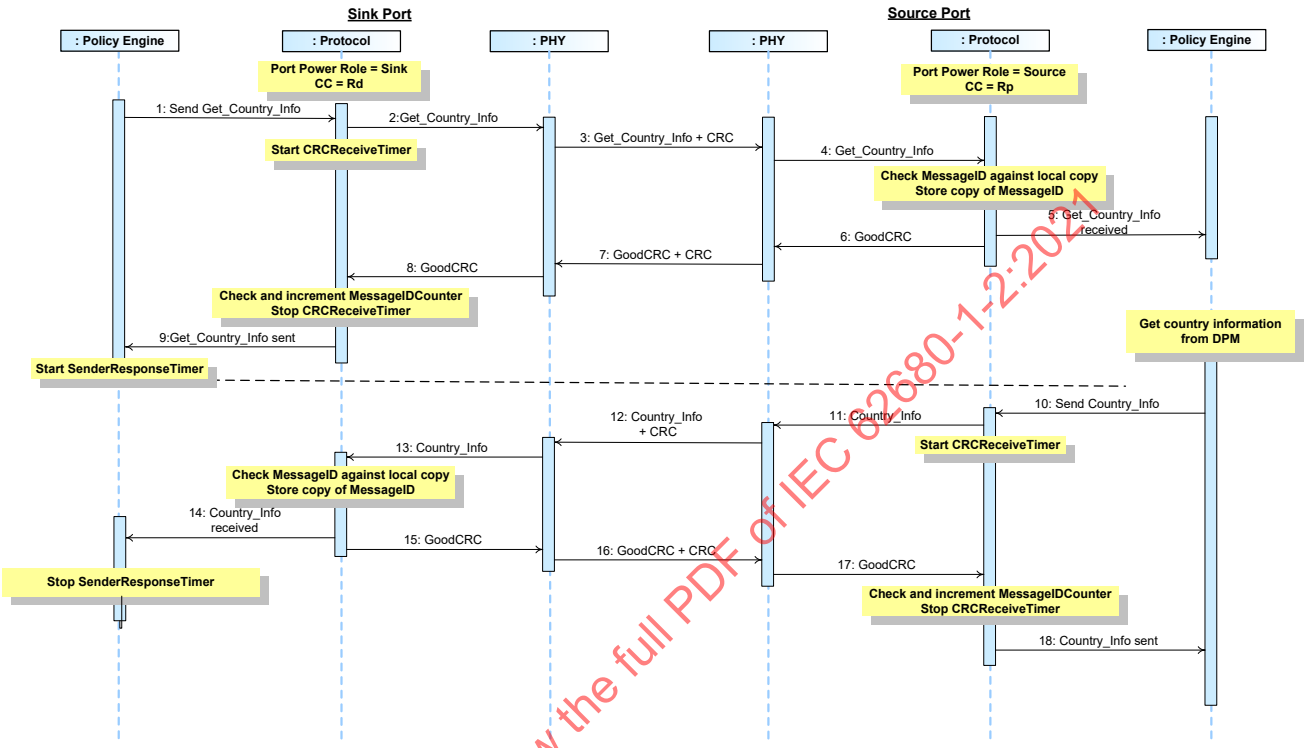


Table 8-49 below provides a detailed explanation of what happens at each labeled step in Figure 8-49 above.

Table 8-49 Steps for a Source getting Sink's Country Information Sequence

Step	Sink Port	Source Port
1	The Port has ***Port Power Role*** set to Sink with the Rd pull down on its CC wire. Policy Engine directs the Protocol Layer to send a ***Get_Country_Info*** Message with a request for Port information for a specific country code.	The Port has ***Port Power Role*** set to Source and the Rp 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 ***Get_Country_Info*** Message.	Physical Layer receives the ***Get_Country_Info*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Get_Country_Info*** 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 ***Get_Country_Info*** 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.

Step	Sink Port	Source Port
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 ***Get_Country_Info*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the Port's manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a ***Country_Info*** 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 ***Country_Info*** Message.	Physical Layer appends a CRC and sends the ***Country_Info*** 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 ***Country_Info*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Country_Info*** Message was successfully sent.
	The Sink has informed the Source of the country information.	

8.3.2.10.8.1 VCONN Source Gets Country Information from a Cable Plug

Figure 8-50 shows an example sequence between a VCONN Source (Source or Sink) and a Cable Plug when the VCONN Source gets the Cable Plug's country information.

Figure 8-50 VCONN Source Gets Cable Plug's Country Information

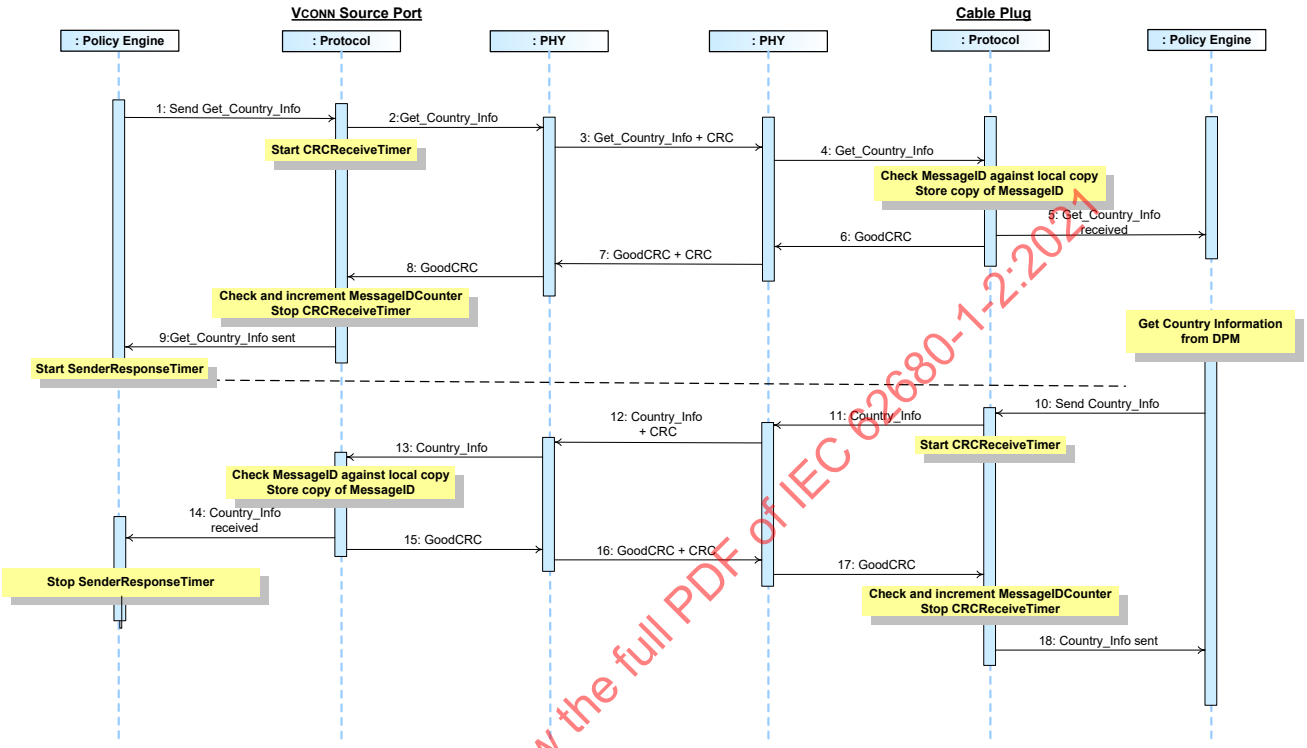


Table 8-50 below provides a detailed explanation of what happens at each labeled step in Figure 8-50 above.

Table 8-50 Steps for a VCONN Source getting Sink's Country Information Sequence

Step	VCONN Source	Cable Plug
1	The Port is currently acting as the VCONN Source. Policy Engine directs the Protocol Layer to send a ***Get_Country_Info*** Message with a request for Port information for a specific country code.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Get_Country_Info*** Message.	Physical Layer receives the ***Get_Country_Info*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Get_Country_Info*** 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 ***Get_Country_Info*** 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.

Step	VCONN Source	Cable Plug
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 ***Get_Country_Info*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine requests the DPM for the Cable Plug's manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a ***Country_Info*** 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 ***Country_Info*** Message.	Physical Layer appends a CRC and sends the ***Country_Info*** 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 ***Country_Info*** Message information to the Policy Engine that consumes it.	
14	The Policy Engine 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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Country_Info*** Message was successfully sent.
	The Cable Plug has informed the Source of its country information.	

8.3.2.11 Security

8.3.2.11.1 Source requests security exchange with Sink

Figure 8-51 shows an example sequence for a security exchange between a Source and a Sink.

Figure 8-51 Source requests security exchange with Sink

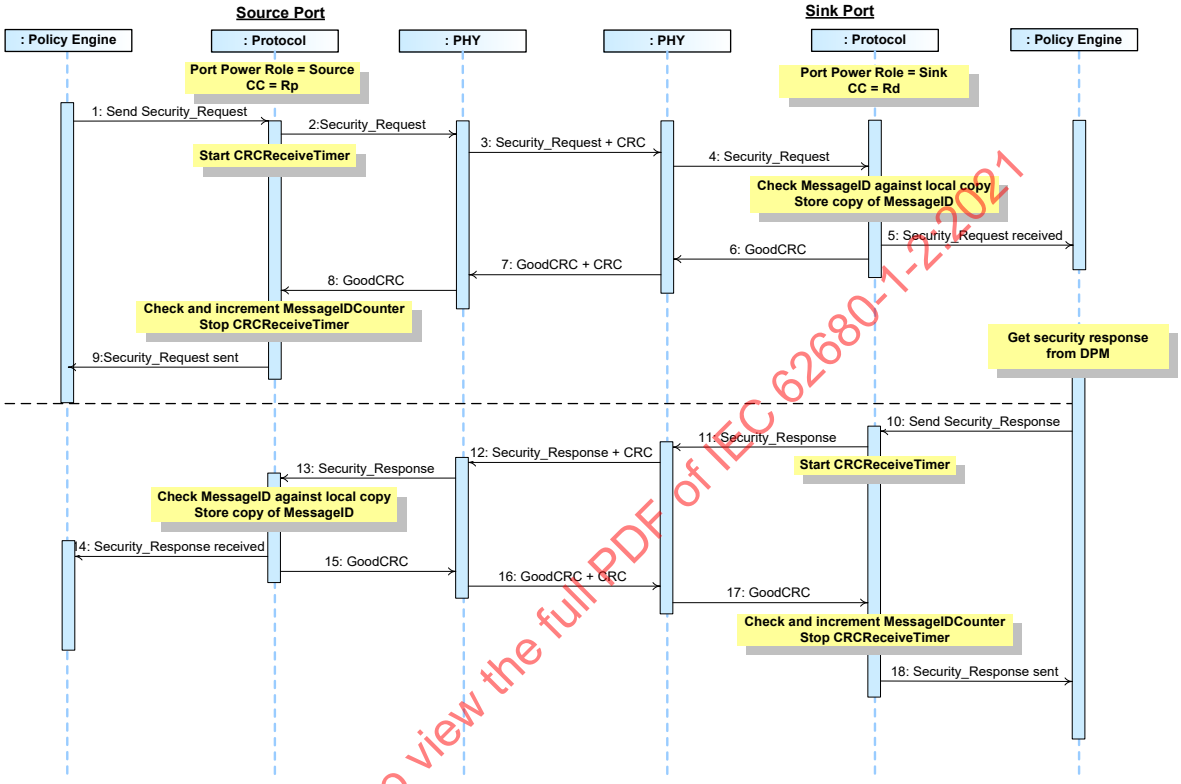


Table 8-51 below provides a detailed explanation of what happens at each labeled step in Figure 8-51 above.

Table 8-51 Steps for a Source requesting a security exchange with a Sink Sequence

Step	Source Port	Sink Port
1	The Port has ***Port Power Role*** set to Source and the Rp pull up on its CC wire. Policy Engine directs the Protocol Layer to send a ***Security_Request*** Message using a payload supplied by the DPM.	The Port has ***Port Power Role*** set to Sink with the Rd 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 ***Security_Request*** Message.	Physical Layer receives the ***Security_Request*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Security_Request*** 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 ***Security_Request*** Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.

Step	Source Port	Sink Port
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 ***Security_Request*** Message was successfully sent.	
10		Policy Engine requests the DPM for the response to the security request which is provided. The Policy Engine tells the Protocol Layer to form a ***Security_Response*** 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 ***Security_Response*** Message.	Physical Layer appends a CRC and sends the ***Security_Response*** 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 ***Security_Response*** Message information to the Policy Engine that consumes it.	
14	Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.	
15	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.
16		Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.
17		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Security_Response*** Message was successfully sent.
	The security exchange is complete.	

8.3.2.11.2 Sink requests security exchange with Source

Figure 8-52 shows an example sequence for a security exchange between a Sink and a Source.

Figure 8-52 Sink requests security exchange with Source

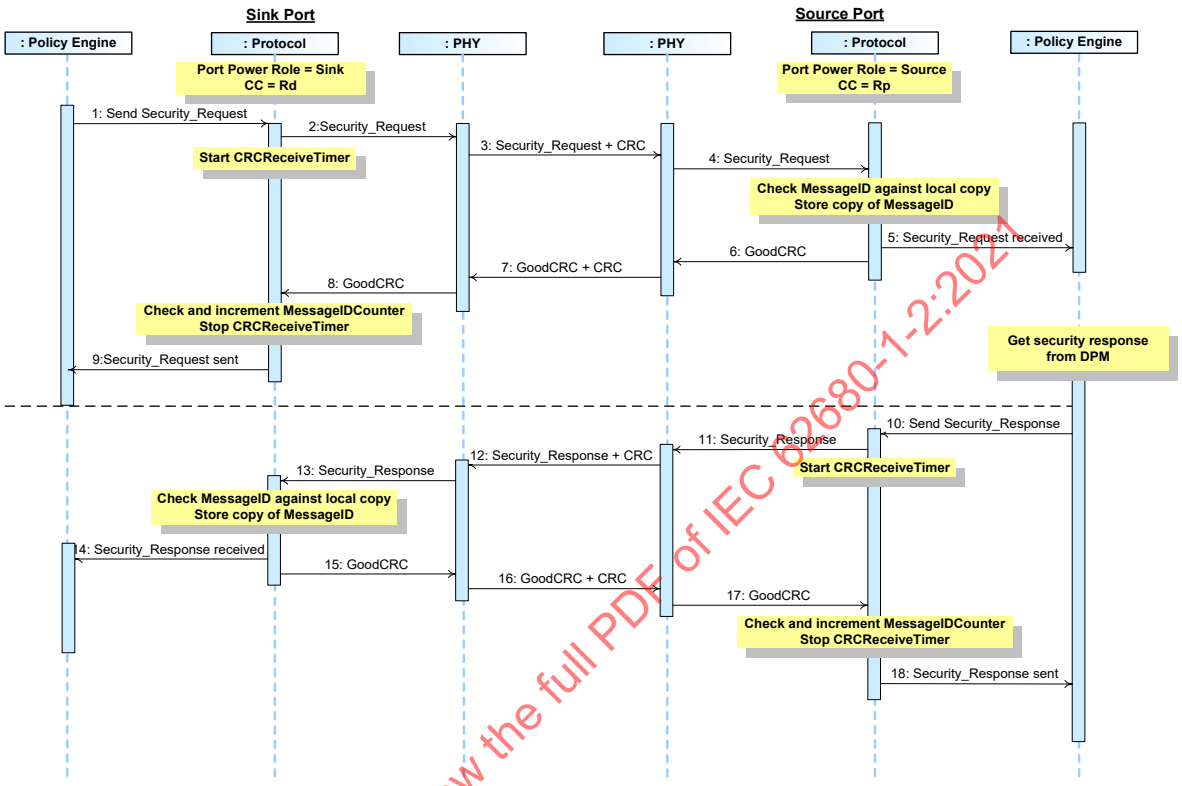


Table 8-52 below provides a detailed explanation of what happens at each labeled step in Figure 8-52 above.

Table 8-52 Steps for a Sink requesting a security exchange with a Source Sequence

Step	Sink Port	Source Port
1	The Port has ***Port Power Role*** set to Sink with the Rd pull down on its CC wire. Policy Engine directs the Protocol Layer to send a ***Security_Request*** Message using a payload supplied by the DPM.	The Port has ***Port Power Role*** set to Source and the Rp 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 ***Security_Request*** Message.	Physical Layer receives the ***Security_Request*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Security_Request*** 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 ***Security_Request*** 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.

Step	Sink Port	Source Port
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 ***Security_Request*** Message was successfully sent.	
10		Policy Engine requests the DPM for the response to the security request which is provided. The Policy Engine tells the Protocol Layer to form a ***Security_Response*** 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 ***Security_Response*** Message.	Physical Layer appends a CRC and sends the ***Security_Response*** 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 ***Security_Response*** Message information to the Policy Engine that consumes it.	
14	Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.	
15	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.
16		Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.
17		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Security_Response*** Message was successfully sent.
	The security exchange is complete.	

8.3.2.11.3 Vconn Source requests security exchange with Cable Plug

Figure 8-53 shows an example sequence for a security exchange between a Vconn Source and a Cable Plug.

Figure 8-53 Vconn Source requests security exchange with Cable Plug

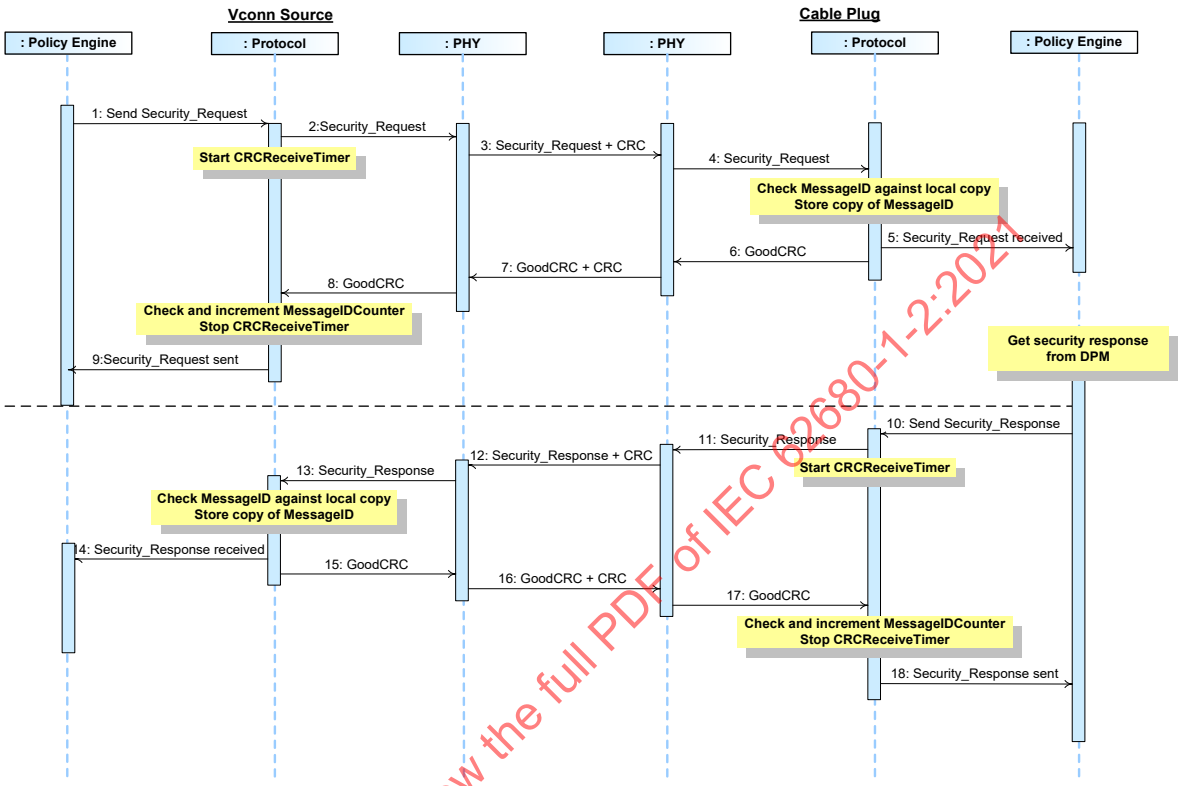


Table 8-53 below provides a detailed explanation of what happens at each labeled step in Figure 8-53 above.

Table 8-53 Steps for a Vconn Source requesting a security exchange with a Cable Plug Sequence

Step	Vconn Source	Cable Plug
1	Policy Engine directs the Protocol Layer to send a ***Security_Request*** Message using a payload supplied by the DPM.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Security_Request*** Message.	Physical Layer receives the ***Security_Request*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Security_Request*** 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 ***Security_Request*** 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.

Step	Vconn Source	Cable Plug
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 ***Security_Request*** Message was successfully sent.	
10		Policy Engine requests the DPM for the response to the security request which is provided. The Policy Engine tells the Protocol Layer to form a ***Security_Response*** 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 ***Security_Response*** Message.	Physical Layer appends a CRC and sends the ***Security_Response*** 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 ***Security_Response*** Message information to the Policy Engine that consumes it.	
14	Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.	
15	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.
16		Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.
17		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Security_Response*** Message was successfully sent.
	The security exchange is complete.	

8.3.2.12 Firmware Update

8.3.2.12.1 Source requests firmware update exchange with Sink

Figure 8-54 shows an example sequence for a firmware update exchange between a Source and a Sink.

Figure 8-54 Source requests firmware update exchange with Sink

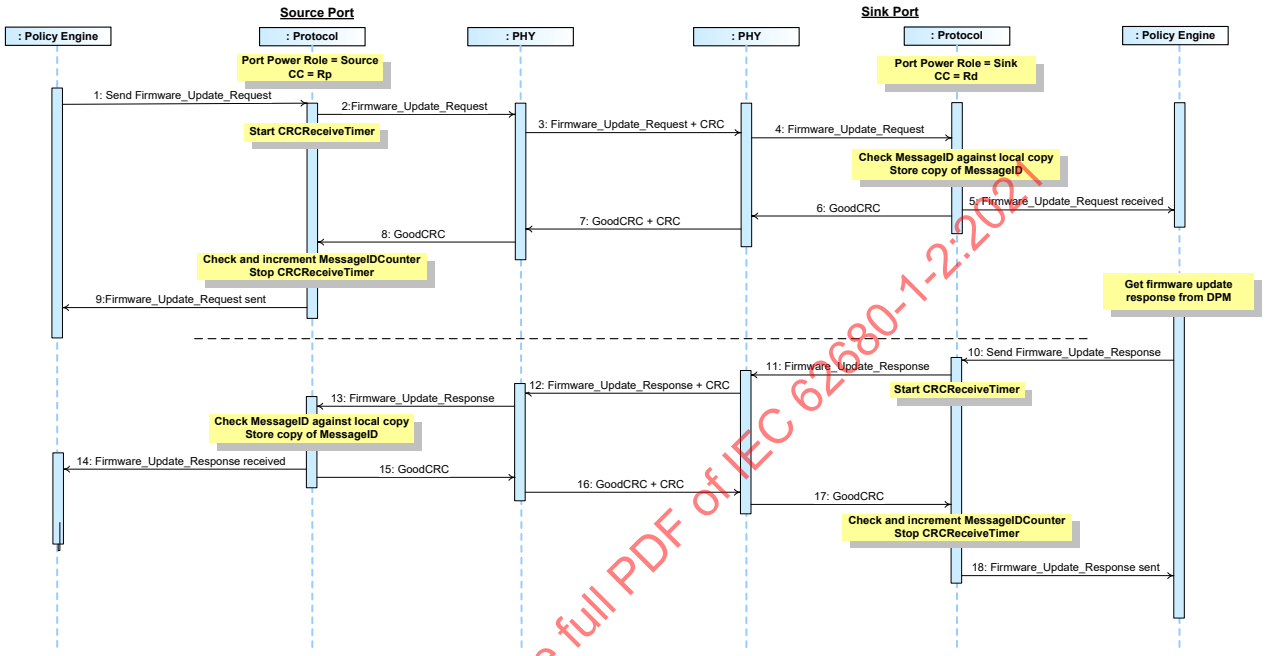


Table 8-54 below provides a detailed explanation of what happens at each labeled step in Figure 8-54 above.

Table 8-54 Steps for a Source requesting a firmware update exchange with a Sink Sequence

Step	Source Port	Sink Port
1	The Port has ***Port Power Role*** set to Source and the Rp pull up on its CC wire. Policy Engine directs the Protocol Layer to send a ***Firmware_Update_Request*** Message using a payload supplied by the DPM.	The Port has ***Port Power Role*** set to Sink with the Rd 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 ***Firmware_Update_Request*** Message.	Physical Layer receives the ***Firmware_Update_Request*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Firmware_Update_Request*** 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 ***Firmware_Update_Request*** 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.

Step	Source Port	Sink Port
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 ***Firmware_Update_Request*** Message was successfully sent.	
10		Policy Engine requests the DPM for the response to the firmware update request which is provided. The Policy Engine tells the Protocol Layer to form a ***Firmware_Update_Response*** 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 ***Firmware_Update_Response*** Message.	Physical Layer appends a CRC and sends the ***Firmware_Update_Response*** 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 ***Firmware_Update_Response*** Message information to the Policy Engine that consumes it.	
14	Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.	
15	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.
16		Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.
17		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Firmware_Update_Response*** Message was successfully sent.
	The firmware update exchange is complete.	

8.3.2.12.2 Sink requests firmware update exchange with Source

Figure 8-55 shows an example sequence for a firmware update exchange between a Sink and a Source.

Figure 8-55 Sink requests firmware update exchange with Source

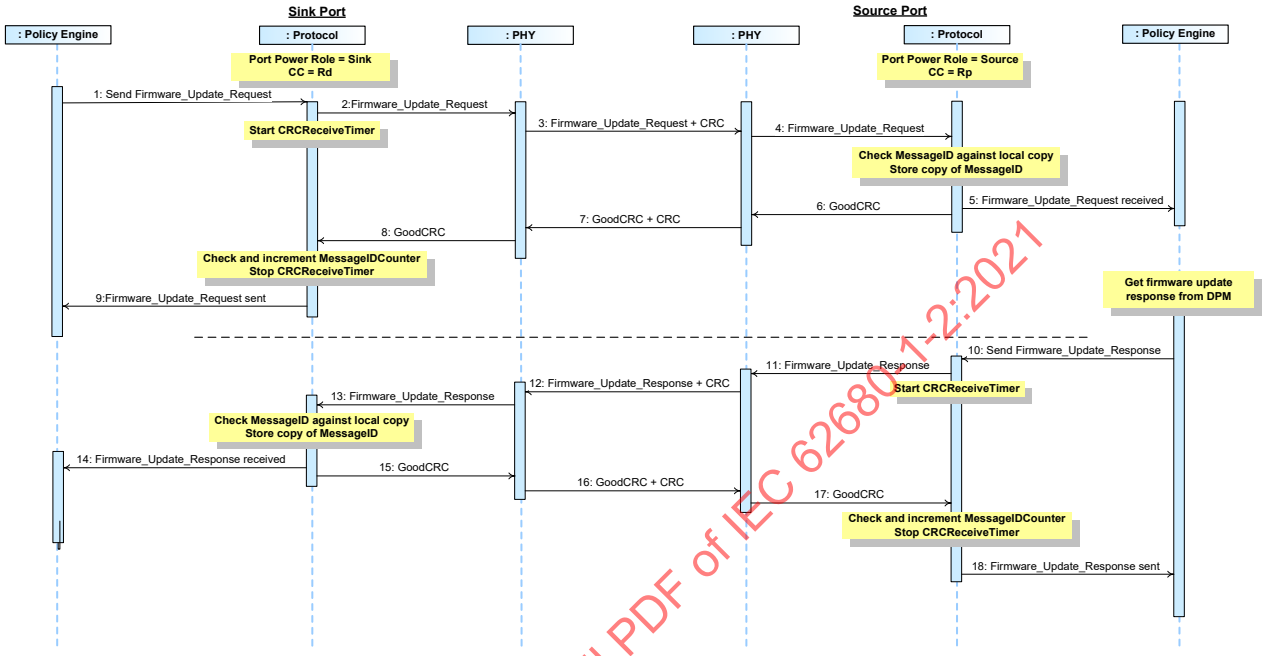


Table 8-55 below provides a detailed explanation of what happens at each labeled step in Figure 8-55 above.

Table 8-55 Steps for a Sink requesting a firmware update exchange with a Source Sequence

Step	Sink Port	Source Port
1	The Port has ***Port Power Role*** set to Sink with the Rd pull down on its CC wire. Policy Engine directs the Protocol Layer to send a ***Firmware_Update_Request*** Message using a payload supplied by the DPM.	The Port has ***Port Power Role*** set to Source and the Rp 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 ***Firmware_Update_Request*** Message.	Physical Layer receives the ***Firmware_Update_Request*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Firmware_Update_Request*** 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 ***Firmware_Update_Request*** 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.	

Step	Sink Port	Source Port
9	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Firmware_Update_Request*** Message was successfully sent.	
10		Policy Engine requests the DPM for the response to the firmware update request which is provided. The Policy Engine tells the Protocol Layer to form a ***Firmware_Update_Response*** 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 ***Firmware_Update_Response*** Message.	Physical Layer appends a CRC and sends the ***Firmware_Update_Response*** 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 ***Firmware_Update_Response*** Message information to the Policy Engine that consumes it.	
14	Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.	
15	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.
16		Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.
17		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Firmware_Update_Response*** Message was successfully sent.
	The firmware update exchange is complete.	

8.3.2.12.3 Vconn Source requests firmware update exchange with Cable Plug

Figure 8-56 shows an example sequence for a firmware update exchange between a Vconn Source and a Cable Plug.

Figure 8-56 Vconn Source requests firmware update exchange with Cable Plug

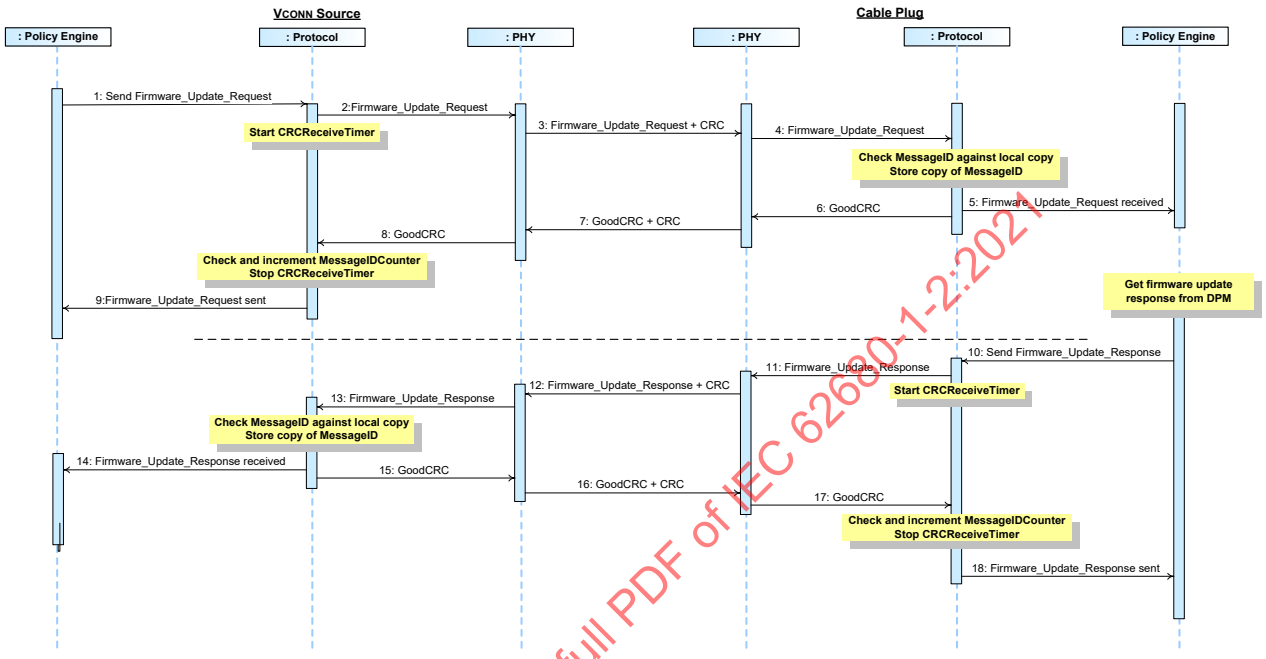


Table 8-56 below provides a detailed explanation of what happens at each labeled step in Figure 8-56 above.

Table 8-56 Steps for a Vconn Source requesting a firmware update exchange with a Cable Plug Sequence

Step	Vconn Source	Cable Plug
1	Policy Engine directs the Protocol Layer to send a ***Firmware_Update_Request*** Message using a payload supplied by the DPM.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Firmware_Update_Request*** Message.	Physical Layer receives the ***Firmware_Update_Request*** Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Firmware_Update_Request*** 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 ***Firmware_Update_Request*** 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.	

Step	Vconn Source	Cable Plug
9	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Firmware_Update_Request*** Message was successfully sent.	
10		Policy Engine requests the DPM for the response to the firmware update request which is provided. The Policy Engine tells the Protocol Layer to form a ***Firmware_Update_Response*** 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 ***Firmware_Update_Response*** Message.	Physical Layer appends a CRC and sends the ***Firmware_Update_Response*** 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 ***Firmware_Update_Response*** Message information to the Policy Engine that consumes it.	
14	Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.	
15	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.
16		Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.
17		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Firmware_Update_Response*** Message was successfully sent.
	The firmware update exchange is complete.	

8.3.2.13 Structured VDM

8.3.2.13.1 DFP to UFP Discover Identity

Figure 8-57 shows an example sequence between a DFP and UFP, where both Port Partners are in an Explicit Contract and the DFP attempts to discover identity information from the UFP.

Figure 8-57 DFP to UFP Discover Identity

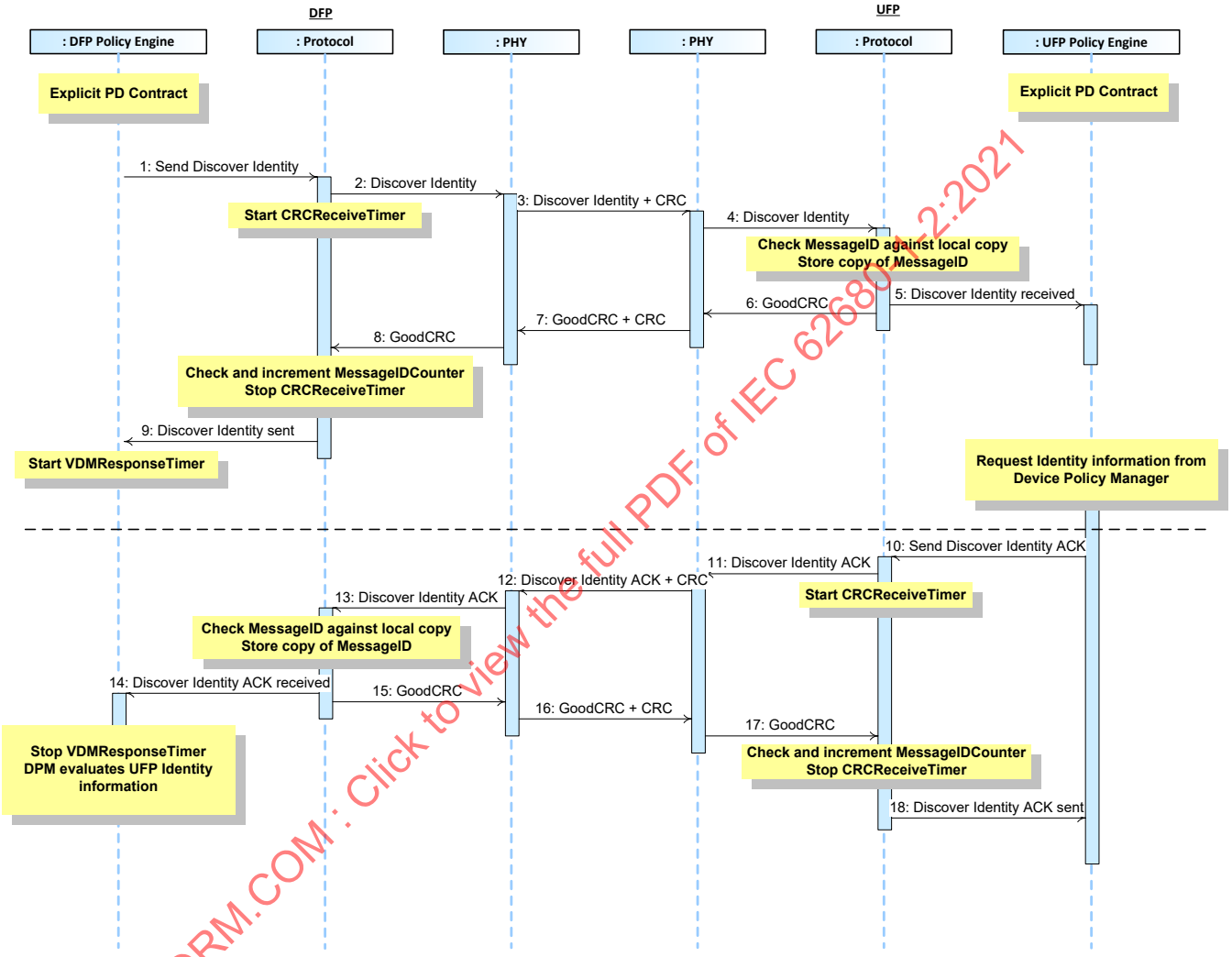


Table 8-57 below provides a detailed explanation of what happens at each labeled step in Figure 8-57 above.

Table 8-57 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 ***Discover Identity*** Command request.	The UFP has an Explicit Contract.
2	Protocol Layer creates the ***Discover Identity*** Command request and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Discover Identity*** Command request.	Physical Layer receives the ***Discover Identity*** Command request and checks the CRC to verify the Message.

Step	DFP	UFP
4		Physical Layer removes the CRC and forwards the ***Discover Identity*** Command request 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 ***Discover Identity*** Command request 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 ***Discover Identity*** Command request was successfully sent. Policy Engine starts the ***VDMResponseTimer***.	
10		Policy Engine requests the identity information from the Device Policy Manager. The Policy Engine tells the Protocol Layer to form a ***Discover Identity*** Command ACK response.
11		Protocol Layer creates the ***Discover Identity*** Command ACK response and passes to Physical Layer. Starts ***CRCReceiveTimer***.
12	Physical Layer receives the ***Discover Identity*** 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 ***Discover Identity*** Command ACK response.
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 ***Discover Identity*** Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the ***VDMResponseTimer*** and passed the Identity information to the Device Policy Manager for evaluation.	
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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Discover Identity*** Command ACK response was successfully sent.

8.3.2.13.2 Source Port to Cable Plug Discover Identity

Figure 8-58 shows an example sequence between 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 8-58 Source Port to Cable Plug Discover Identity

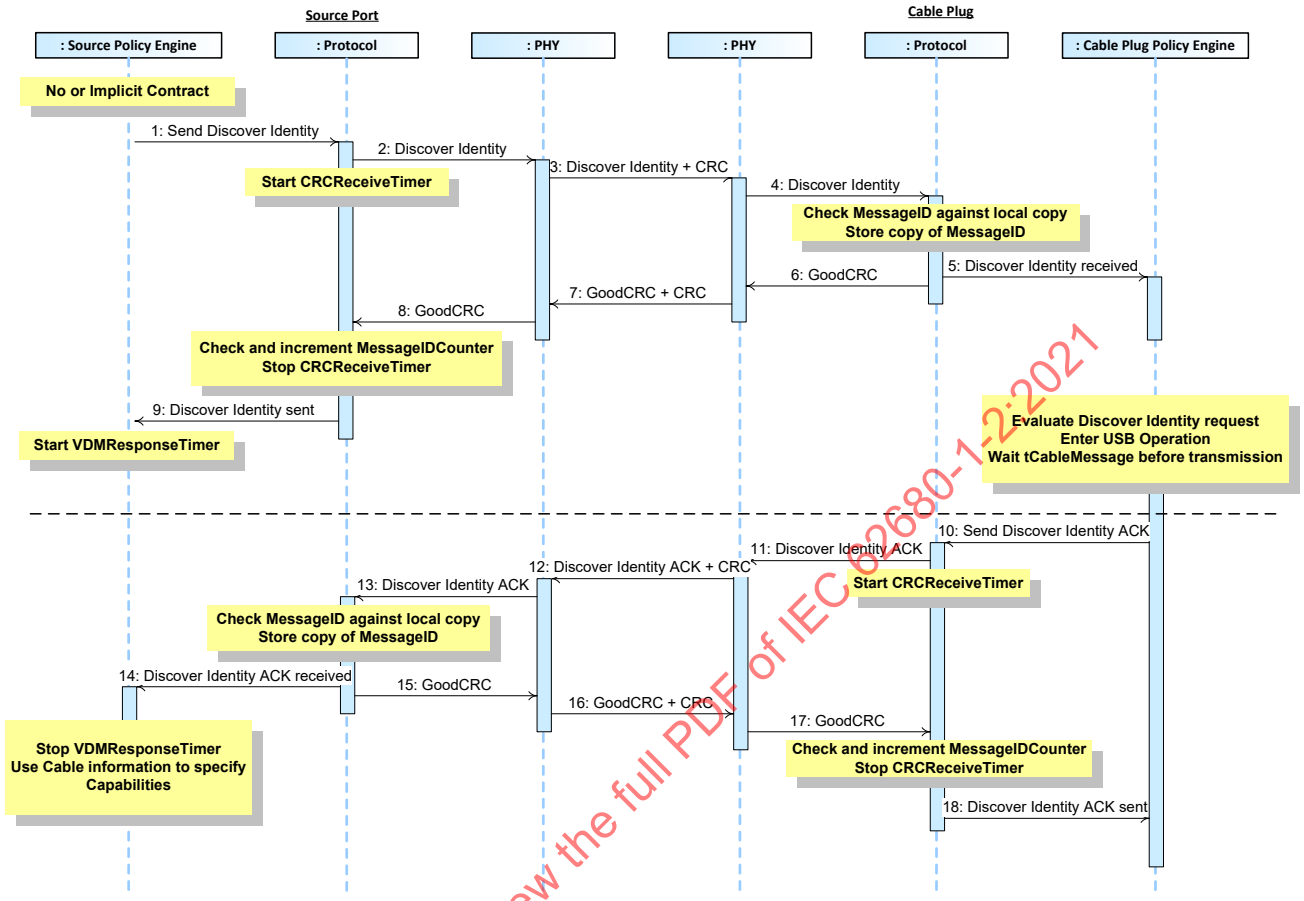


Table 8-58 below provides a detailed explanation of what happens at each labeled step in Figure 8-58 above.

Table 8-58 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 ***Discover Identity*** Command request.	
2	Protocol Layer creates the ***Discover Identity*** Command request and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Discover Identity*** Command request.	Physical Layer receives the ***Discover Identity*** Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Discover Identity*** Command request 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 ***Discover Identity*** Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.

Step	Source Port	Cable Plug
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 ***Discover Identity*** Command request was successfully sent. Policy Engine starts the ***VDMResponseTimer***.	
10		Policy Engine requests the identity information from the Device Policy Manager. ***tCableMessage*** after the ***GoodCRC*** Message was sent the Policy Engine tells the Protocol Layer to form a ***Discover Identity*** Command ACK response.
11		Protocol Layer creates the ***Discover Identity*** Command ACK response and passes to Physical Layer. Starts ***CRCReceiveTimer***.
12	Physical Layer receives the ***Discover Identity*** Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the ***Discover Identity*** Command ACK response.
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 ***Discover Identity*** Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the ***VDMResponseTimer*** and passes the identity information to the Device Policy Manager for evaluation.	
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.
18	The Source uses the Cable Plug information as input to its offered capabilities.	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Accept*** Message was successfully sent.

8.3.2.13.3 DFP to Cable Plug Discover Identity

Figure 8-59 shows an example sequence between a DFP and a Cable Plug, where the DFP attempts to discover identity information from the Cable Plug.

Figure 8-59 DFP to Cable Plug Discover Identity

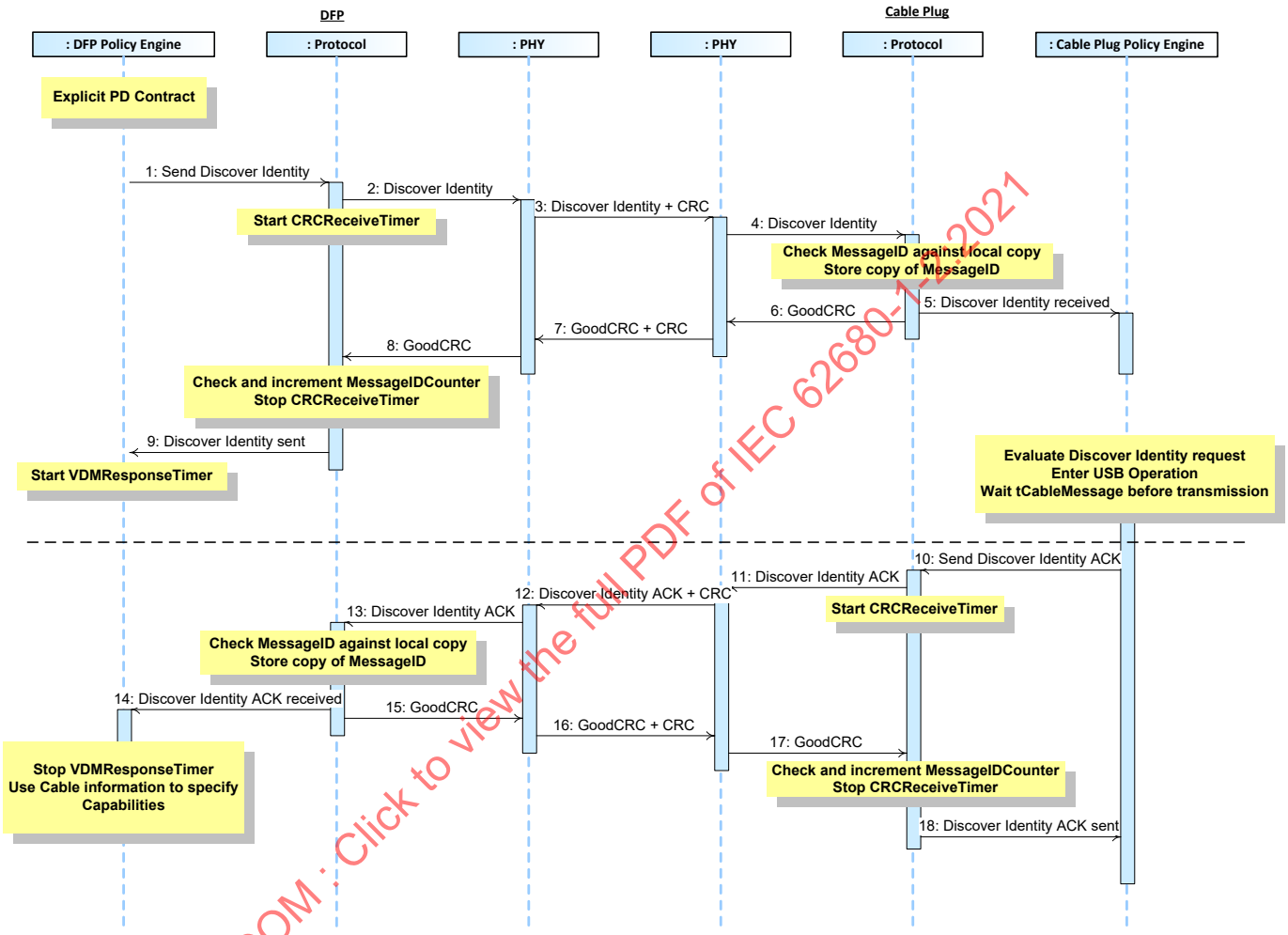


Table 8-59 below provides a detailed explanation of what happens at each labeled step in Figure 8-59 above.

Table 8-59 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 ***Discover Identity*** Command request.	
2	Protocol Layer creates the ***Discover Identity*** Command request and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Discover Identity*** Command request.	Physical Layer receives the ***Discover Identity*** Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Discover Identity*** Command request to the Protocol Layer.

Step	DFP	Cable Plug
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 ***Discover Identity*** Command request 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 ***Discover Identity*** Command request was successfully sent. Policy Engine starts the ***VDMResponseTimer***.	
10		Policy Engine requests the identity information from the Device Policy Manager. ***tCableMessage*** after the ***GoodCRC*** Message was sent the Policy Engine tells the Protocol Layer to form a ***Discover Identity*** Command ACK response.
11		Protocol Layer creates the ***Discover Identity*** Command ACK response and passes to Physical Layer. Starts ***CRCReceiveTimer***.
12	Physical Layer receives the ***Discover Identity*** 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 ***Discover Identity*** Command ACK response.
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 ***Discover Identity*** Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the ***Discover Identity*** Command ACK response and passes the identity information to the Device Policy Manager for evaluation.	
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.
18	The DFP when acting as a Source uses the Cable Plug information as input to its offered capabilities.	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Accept*** Message was successfully sent.

8.3.2.13.4 DFP to UFP Enter Mode

Figure 8-60 shows an example sequence between a 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 8-60 DFP to UFP Enter Mode

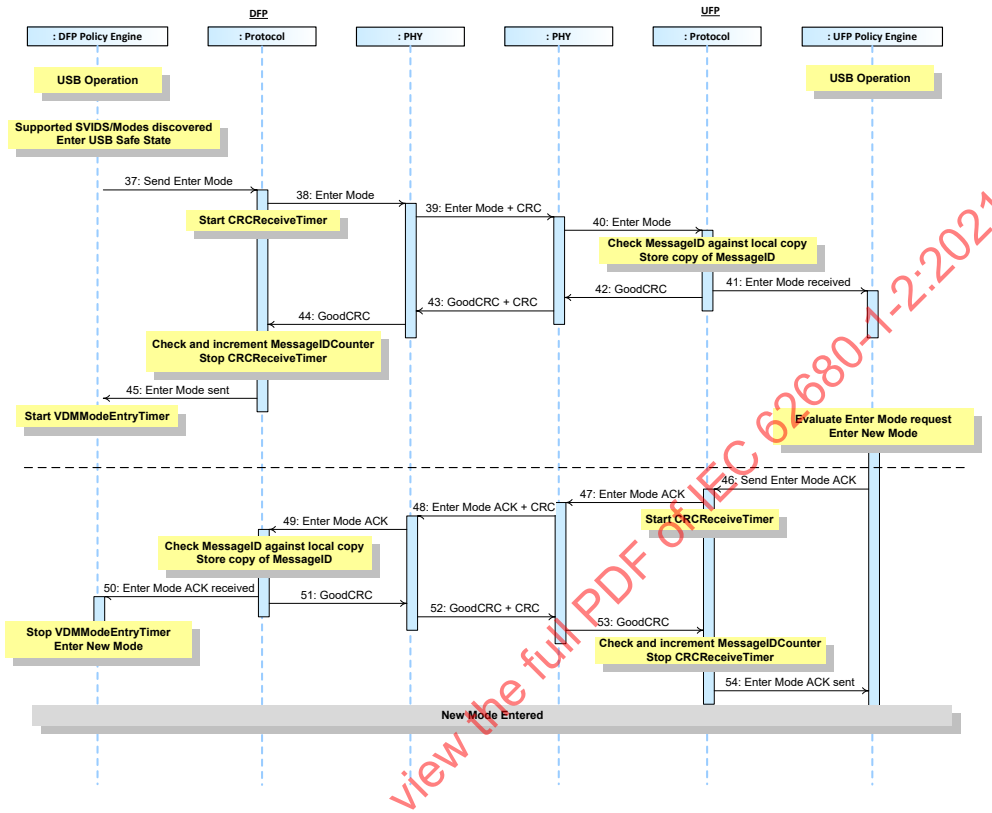


Table 8-60 below provides a detailed explanation of what happens at each labeled step in Figure 8-60 above.

Table 8-60 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 ***Discover SVIDs*** Command request and the supported Modes using the ***Discover Modes*** 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 ***Enter Mode*** Command request.	The UFP has an Explicit Contract.
2	Protocol Layer creates the ***Enter Mode*** Command request and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Enter Mode*** Command request.	Physical Layer receives the ***Enter Mode*** Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Enter Mode*** Command request to the Protocol Layer.

Step	DFP	UFP
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 ***Enter Mode*** Command request 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 ***Enter Mode*** Command request was successfully sent. Policy Engine starts the ***VDMModeEntryTimer***.	
10		Policy Engine requests the Device Policy Manager to enter the new Mode. The Policy Engine tells the Protocol Layer to form an ***Enter Mode*** Command ACK response.
11		Protocol Layer creates the ***Enter Mode*** Command ACK response and passes to Physical Layer. Starts ***CRCReceiveTimer***.
12	Physical Layer receives the ***Enter Mode*** 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 ***Enter Mode*** Command ACK response.
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 ***Enter Mode*** Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the ***VDMModeEntryTimer*** and requests the Device Policy Manager to enter the new Mode.	
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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Enter Mode*** Command ACK response was successfully sent.
	DFP and UFP are operating in the new Mode	

8.3.2.13.5 DFP to UFP Exit Mode

Figure 8-61 shows an example sequence between a DFP and a UFP, where the DFP commands the UFP to exit the only Active Mode.

Figure 8-61 DFP to UFP Exit Mode

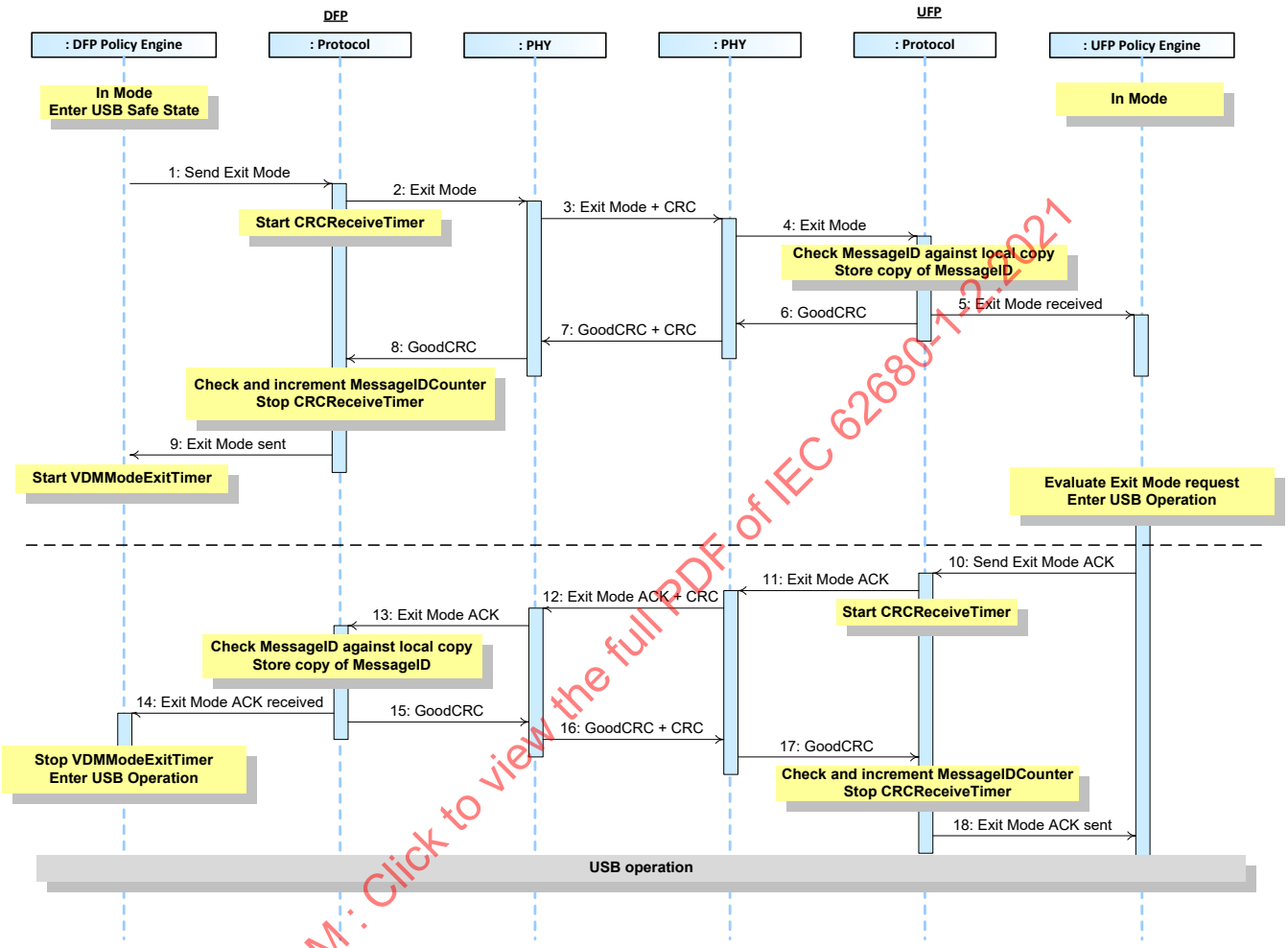


Table 8-61 below provides a detailed explanation of what happens at each labeled step in Figure 8-61 above.

Table 8-61 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 ***Exit Mode*** Command request.	The UFP is in a Mode.
2	Protocol Layer creates the ***Exit Mode*** Command request and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Exit Mode*** Command request.	Physical Layer receives the ***Exit Mode*** Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Exit Mode*** Command request to the Protocol Layer.

Step	DFP	UFP
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 ***Exit Mode*** Command request 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 ***Exit Mode*** Command request was successfully sent. Policy Engine starts the ***VDMModeExitTimer***.	
10		Policy Engine requests the Device Policy Manager to enter USB operation. The Policy Engine tells the Protocol Layer to form an ***Exit Mode*** Command ACK response.
11		Protocol Layer creates the ***Exit Mode*** Command ACK response and passes to Physical Layer. Starts ***CRCReceiveTimer***.
12	Physical Layer receives the ***Exit Mode*** 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 ***Exit Mode*** Command ACK response.
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 ***Exit Mode*** Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the ***VDMModeExitTimer*** and requests the Device Policy Manager to enter USB Operation.	
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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Exit Mode*** Command ACK response was successfully sent.
	Both DFP and UFP are in USB Operation	

8.3.2.13.6 DFP to Cable Plug Enter Mode

Figure 8-62 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 8-62 DFP to Cable Plug Enter Mode

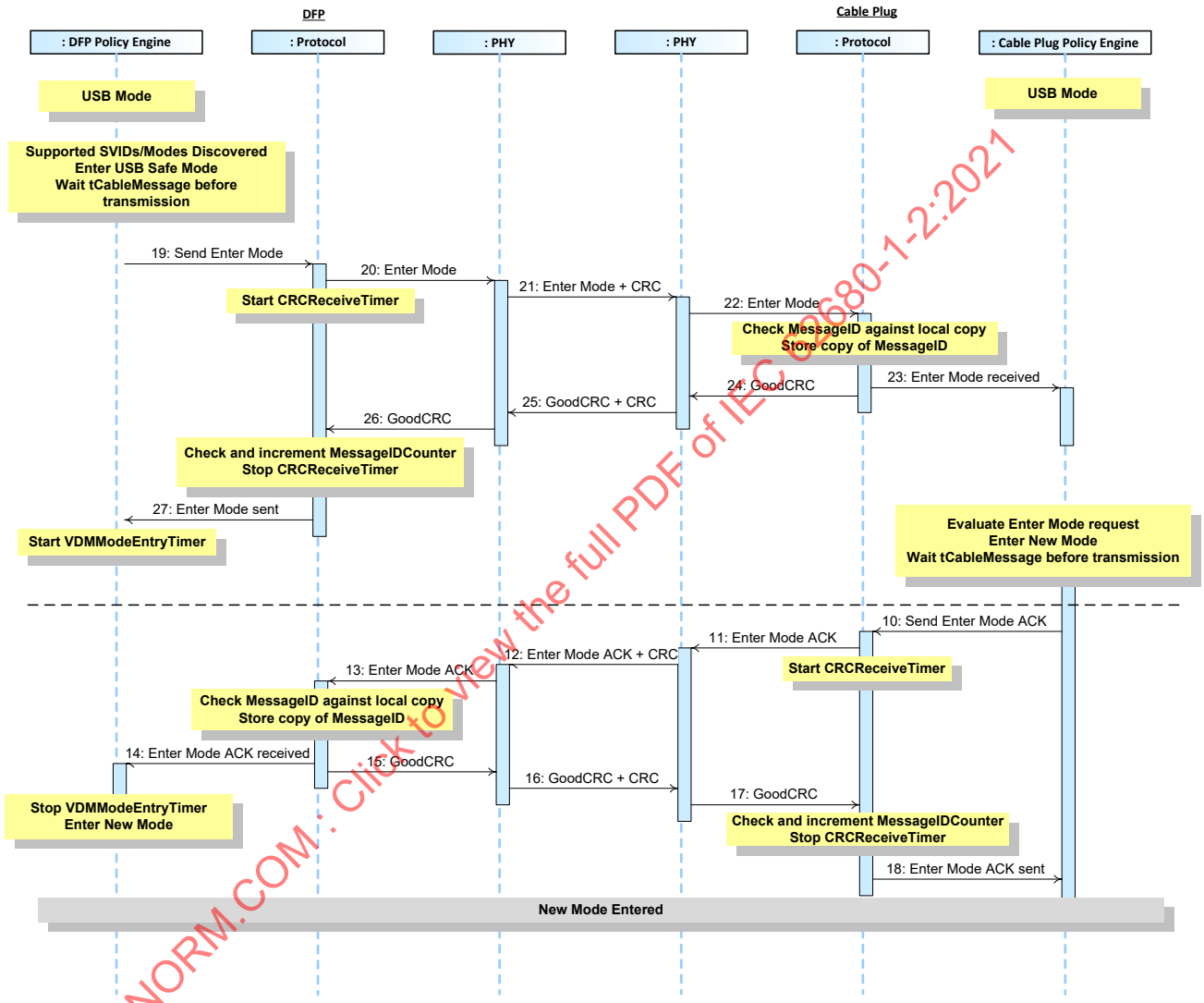


Table 8-62 below provides a detailed explanation of what happens at each labeled step in Figure 8-62 above.

Table 8-62 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 ***Discover SVIDs*** Command request and the supported Modes using the ***Discover Modes*** Command request The DFP goes to USB Safe State. The Device Policy Manager requests the Policy Engine to enter a Mode. ***tCableMessage*** after the last ***GoodCRC*** Message was sent the Policy Engine directs the Protocol Layer to send an ***Enter Mode*** Command request.	
2	Protocol Layer creates the ***Enter Mode*** Command request and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Enter Mode*** Command request.	Physical Layer receives the ***Enter Mode*** Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Enter Mode*** Command request 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 ***Enter Mode*** Command request 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 ***Enter Mode*** Command request was successfully sent. Policy Engine starts the ***VDMModeEntryTimer***.	
10		Policy Engine requests the Device Policy Manager to enter the new Mode. ***tCableMessage*** after the ***GoodCRC*** Message was sent the Policy Engine tells the Protocol Layer to form an ***Enter Mode*** Command ACK response.
11		Protocol Layer creates the ***Enter Mode*** Command ACK response and passes to Physical Layer. Starts ***CRCReceiveTimer***.
12	Physical Layer receives the ***Enter Mode*** 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 ***Enter Mode*** Command ACK response.
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 ***Enter Mode*** Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the ***VDMModeEntryTimer*** and requests the Device Policy Manager to enter the new Mode.	
15	Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.	

Step	DFP	Cable Plug
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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Enter Mode*** Command ACK response was successfully sent.
	DFP and Cable Plug are operating in the new Mode	

8.3.2.13.7 DFP to Cable Plug Exit Mode

Figure 8-63 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 8-63 DFP to Cable Plug Exit Mode

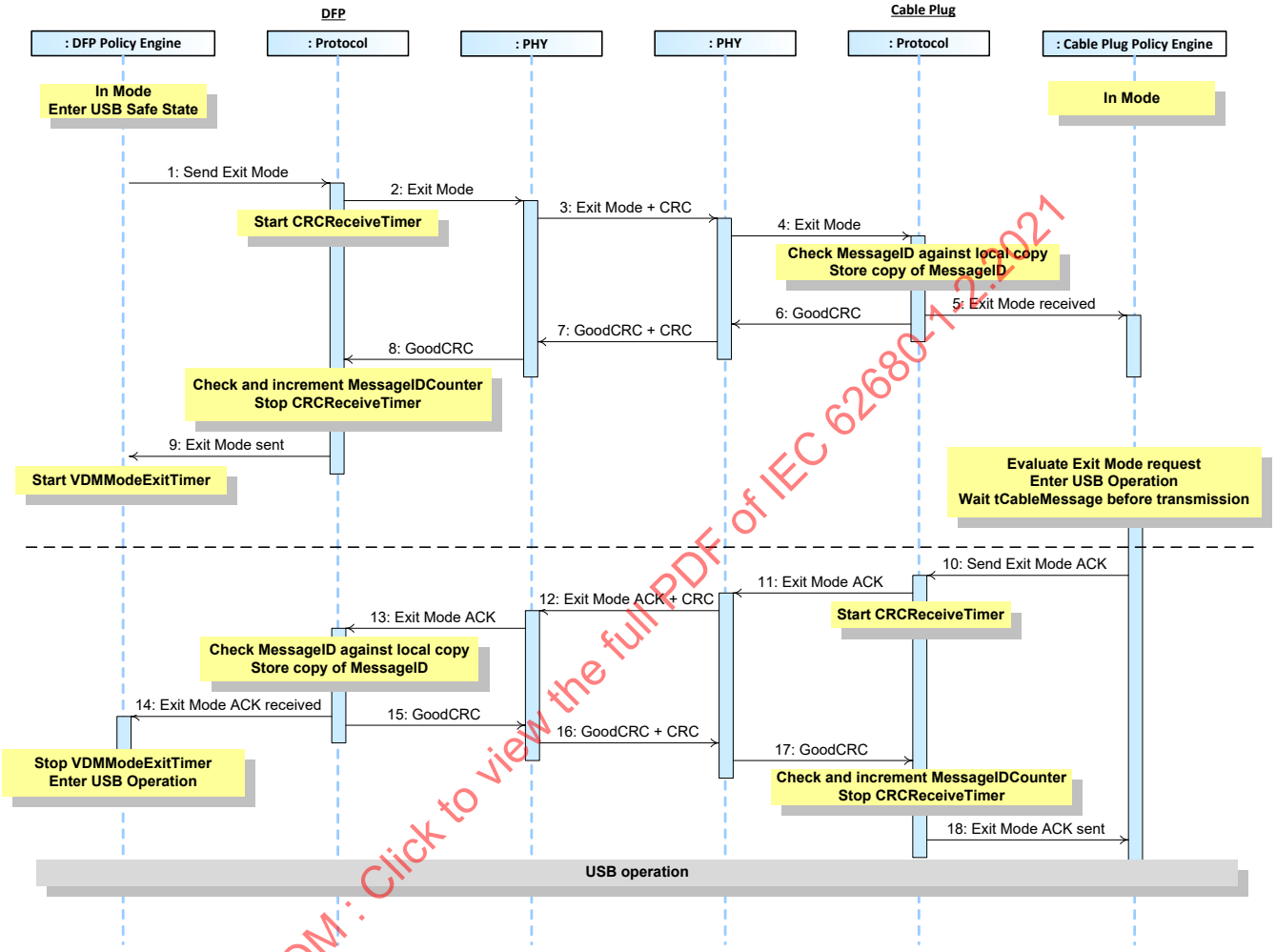


Table 8-63 below provides a detailed explanation of what happens at each labeled step in Figure 8-63 above.

Table 8-63 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 ***Exit Mode*** Command request.	The Cable Plug is in a Mode.
2	Protocol Layer creates the ***Exit Mode*** Command request and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Exit Mode*** Command request.	Physical Layer receives the ***Exit Mode*** Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Exit Mode*** Command request to the Protocol Layer.

Step	DFP	Cable Plug
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 ***Exit Mode*** Command request 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 ***Exit Mode*** Command request was successfully sent. Policy Engine starts the ***VDMModeExitTimer***.	
10		Policy Engine requests the Device Policy Manager to enter USB operation. ***cableMessage*** after the ***GoodCRC*** Message was sent the Policy Engine tells the Protocol Layer to form an ***Exit Mode*** Command ACK response.
11		Protocol Layer creates the ***Exit Mode*** Command ACK response and passes to Physical Layer. Starts ***CRCReceiveTimer***.
12	Physical Layer receives the ***Exit Mode*** 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 ***Exit Mode*** Command ACK response.
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 ***Exit Mode*** Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the ***VDMModeExitTimer*** and requests the Device Policy Manager to enter USB Operation.	
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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Exit Mode*** Command ACK response was successfully sent.
	Both DFP and Cable Plug are in USB Operation	

8.3.2.13.8 UFP to DFP Attention

Figure 8-64 shows an example sequence between a USB Type-C DFP and a UFP, where the UFP requests attention from the DFP.

Figure 8-64 UFP to DFP Attention

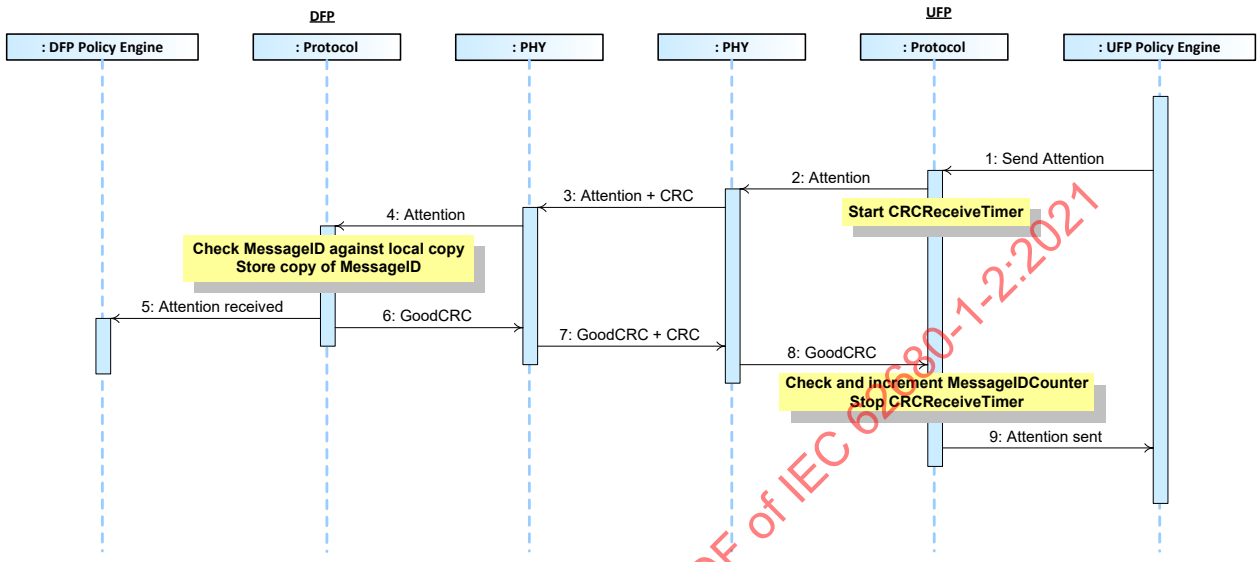


Table 8-64 below provides a detailed explanation of what happens at each labeled step in Figure 8-64 above.

Table 8-64 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 ***Attention*** Command request.
2		Protocol Layer creates the ***Attention*** Command request and passes to Physical Layer. Starts ***CRCReceiveTimer***.
3	Physical Layer receives the ***Attention*** Command request and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the ***Attention*** Command request.
4	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 ***Attention*** Command request information to the Policy Engine that consumes it.	
5	The Policy Engine informs the Device Policy Manager	
6	Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.	
7	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.
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 ***Attention*** Command request was successfully sent.

8.3.2.14 Built in Self-Test (BIST)

8.3.2.14.1 BIST Carrier Mode

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

Figure 8-65 shows the Messages as they flow across the bus and within the devices. This test enables the measurement of power supply noise and frequency drift.

1. Connection is established and stable.
2. Tester sends a ***BIST*** Message with a ***BIST Carrier Mode*** BIST Data Object.
3. UUT answers with a ***GoodCRC*** Message.
4. UUT starts sending the Test Pattern.
5. Operator does the measurements.
6. The test ends after ***tBISTContMode***.

See also Section 5.9.1 and Section 6.4.3.1.

Figure 8-65 BIST Carrier Mode Test

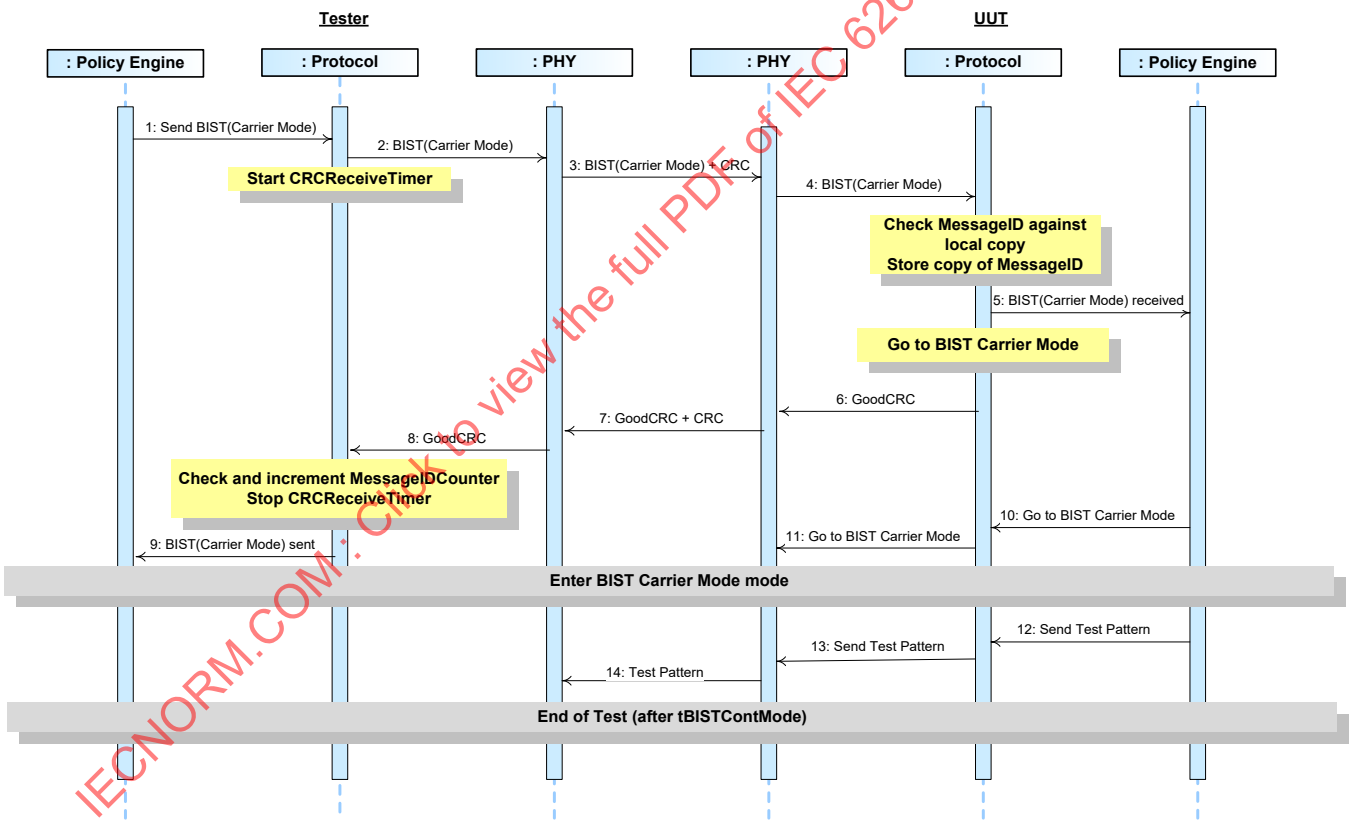


Table 8-65 Steps for BIST Carrier Mode Test

Step	Tester	UUT
1	The Policy Engine directs the Protocol Layer to generate a ***BIST*** Message, with a BIST Data Object of ***BIST Carrier Mode***, to put the UUT into BIST Carrier Mode test 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 Carrier Mode***. The Policy Engine goes to ***BIST Carrier Mode***.
11		Protocol Layer tells Physical Layer to go into ***BIST Carrier Mode***.
	UUT enters ***BIST Carrier 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.
	The UUT exits ***BIST Carrier Mode*** after ***tBISTContMode.***	

8.3.2.14.2 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 8-65 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 2and 3 are repeated any number of times.
5. The test ends after ***Hard Reset*** Signaling is issued.

See also Section 5.9.2 and Section 6.4.3.2.

Figure 8-66 BIST Test Data Test

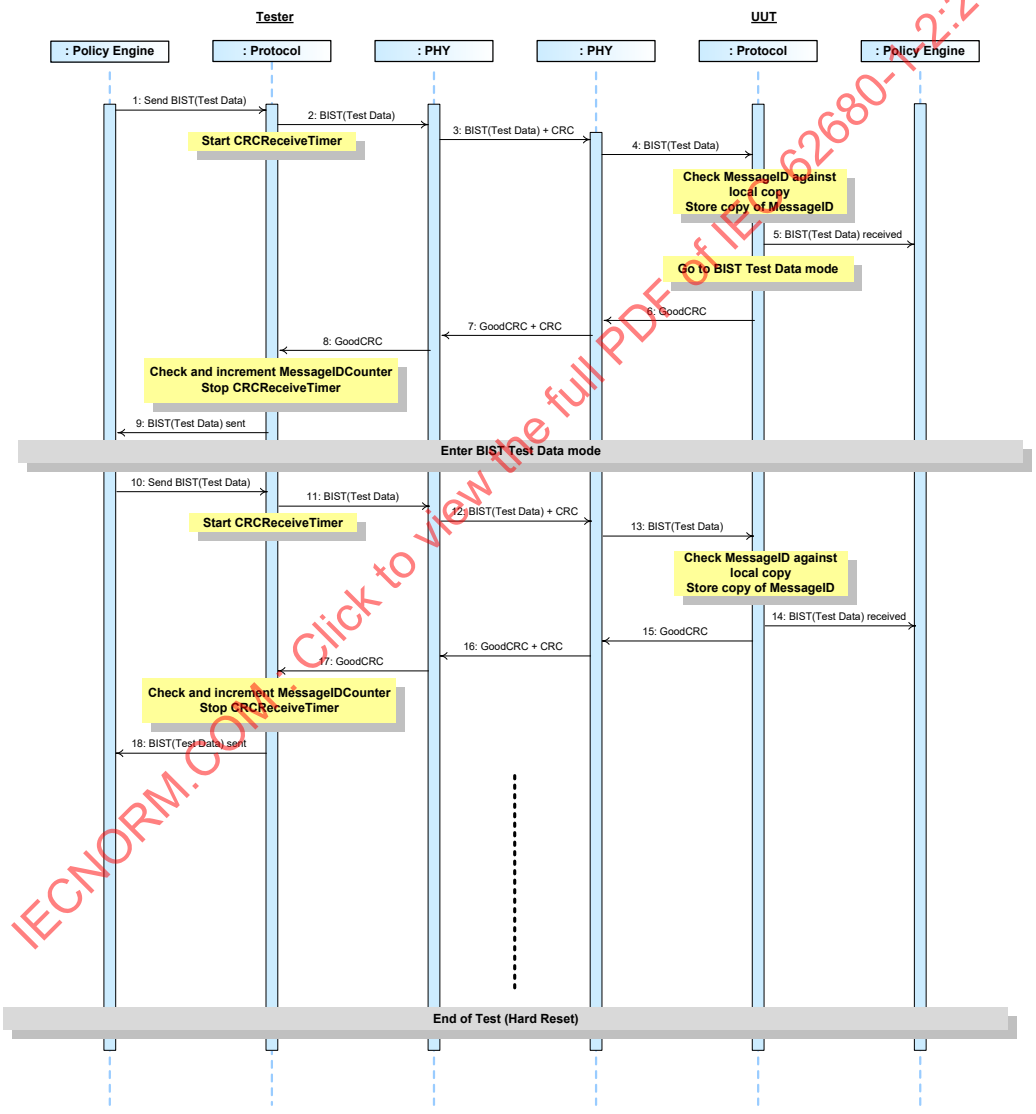


Table 8-66 Steps for BIST Test Data Test

Step	Tester	UUT
1	The Policy Engine directs the Protocol Layer to generate a ***BIST*** Message, with a BIST Data Object of ***BIST Test Data***, to put the UUT into BIST Test Data test 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. The Policy Engine goes into BIST Test Data Mode where it sends no further Messages except for ***GoodCRC*** Messages in response to received Messages (see Section 6.4.3.2).
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.	
	UUT enters BIST Test Data test mode	
10	The Policy Engine directs the Protocol Layer to generate a ***BIST*** Message, with a BIST Data Object of ***BIST Test Data***, to put the UUT into BIST Test Data test mode.	
11	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
12	Physical Layer appends CRC and sends the ***BIST*** Message.	Physical Layer receives the ***BIST*** Message and checks the CRC to verify the Message.
13		Physical Layer removes the CRC and forwards the ***BIST*** Message to the Protocol Layer.

Step	Tester	UUT
14		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. The Policy Engine goes into BIST Test Data Mode where it sends no further Messages except for ***GoodCRC*** Messages in response to received Messages (see Section 6.4.3.2).
15		Protocol Layer generates a ***GoodCRC*** Message and passes it Physical Layer.
16	Physical Layer receives the ***GoodCRC*** and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the ***GoodCRC*** Message.
17	Physical Layer removes the CRC and forwards the ***GoodCRC*** Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***BIST*** Message was successfully sent.	
	Repeat steps 10-18 any number of times	
	The UUT exits BIST Test Data test mode after a Hard Reset	

8.3.2.15 Enter USB

8.3.2.15.1 UFP Entering USB4TM Mode (Valid)

This is an example of an Enter USB operation where the DFP requests *[USB4]* mode when this is a ***Valid*** mode of operation for the UFP. Figure 8-67 shows the Messages as they flow across the bus and within the devices to accomplish the Enter USB process.

Figure 8-67 UFP Entering USB4 Mode (Valid)

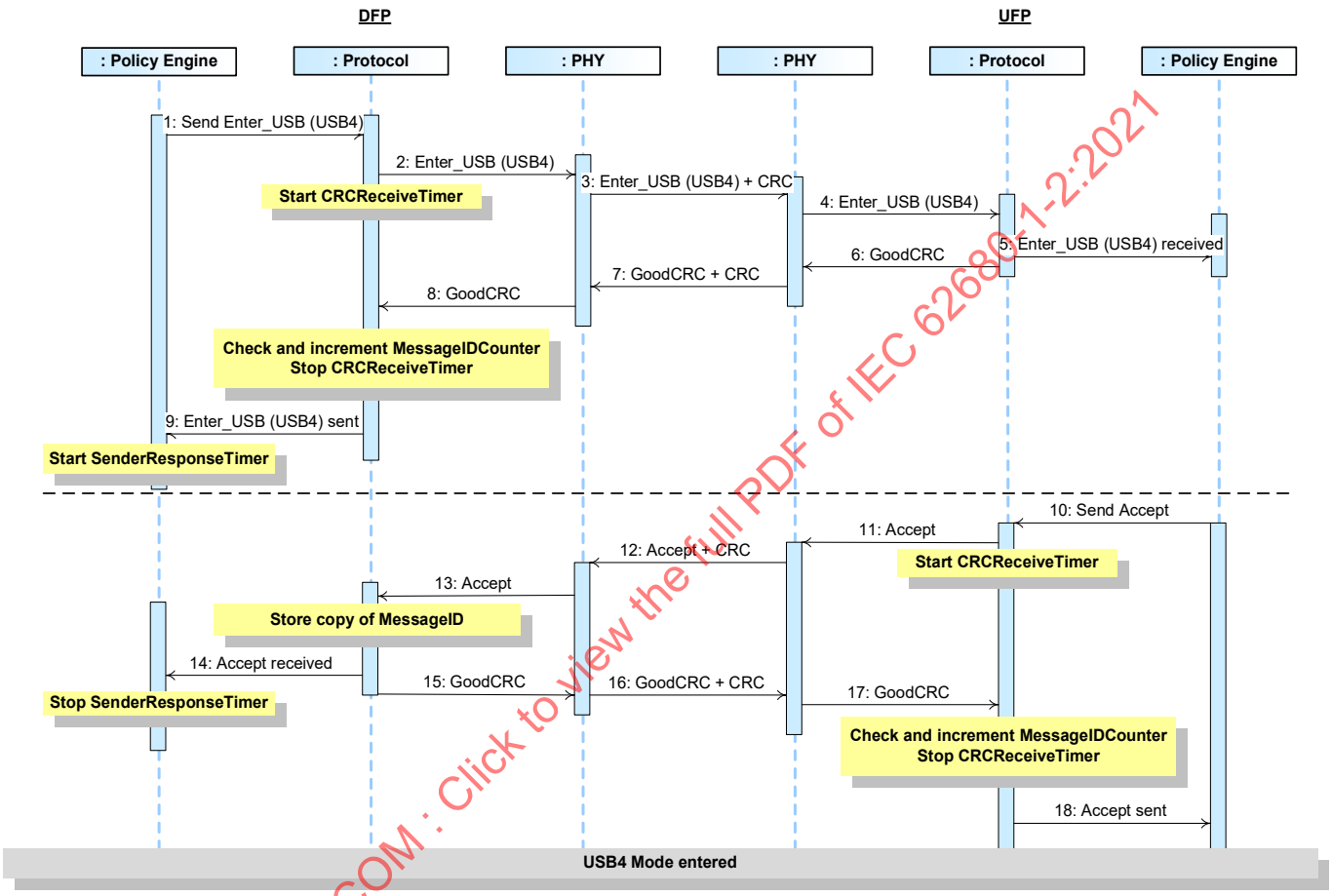


Table 8-67 below provides a detailed explanation of what happens at each labeled step in Figure 8-67 above.

Table 8-67 Steps for UFP USB4 Mode Entry (Valid)

Step	DFP	UFP
1	The Policy Engine directs the Protocol Layer to generate an ***Enter_USB*** Message to request entry to ***[USB4]*** mode.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Enter_USB*** Message.	Physical Layer receives the ***Enter_USB*** Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Enter_USB*** Message to the Protocol Layer.

Step	DFP	UFP
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 ***Enter_USB*** 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 ***Enter_USB*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine 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 Message.	Physical Layer appends a CRC and sends the Message.
13	Protocol Layer stores the ***MessageID*** of the incoming Message.	
14	The Protocol Layer forwards the received ***Accept*** Message information to the Policy Engine that consumes it.	
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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Accept*** Message was successfully sent.
	Both Port Partners enter ***[USB4]*** operation.	

8.3.2.15.2 Cable Plug Entering USB4 Mode (Valid)

This is an example of an Enter USB operation where the DFP requests ***[USB4]*** mode when this is a ***Valid*** mode of operation for the Cable Plug. Figure 8-68 shows the Messages as they flow across the bus and within the devices to accomplish the Enter USB process.

Figure 8-68 Cable Plug Entering USB4 Mode (Valid)

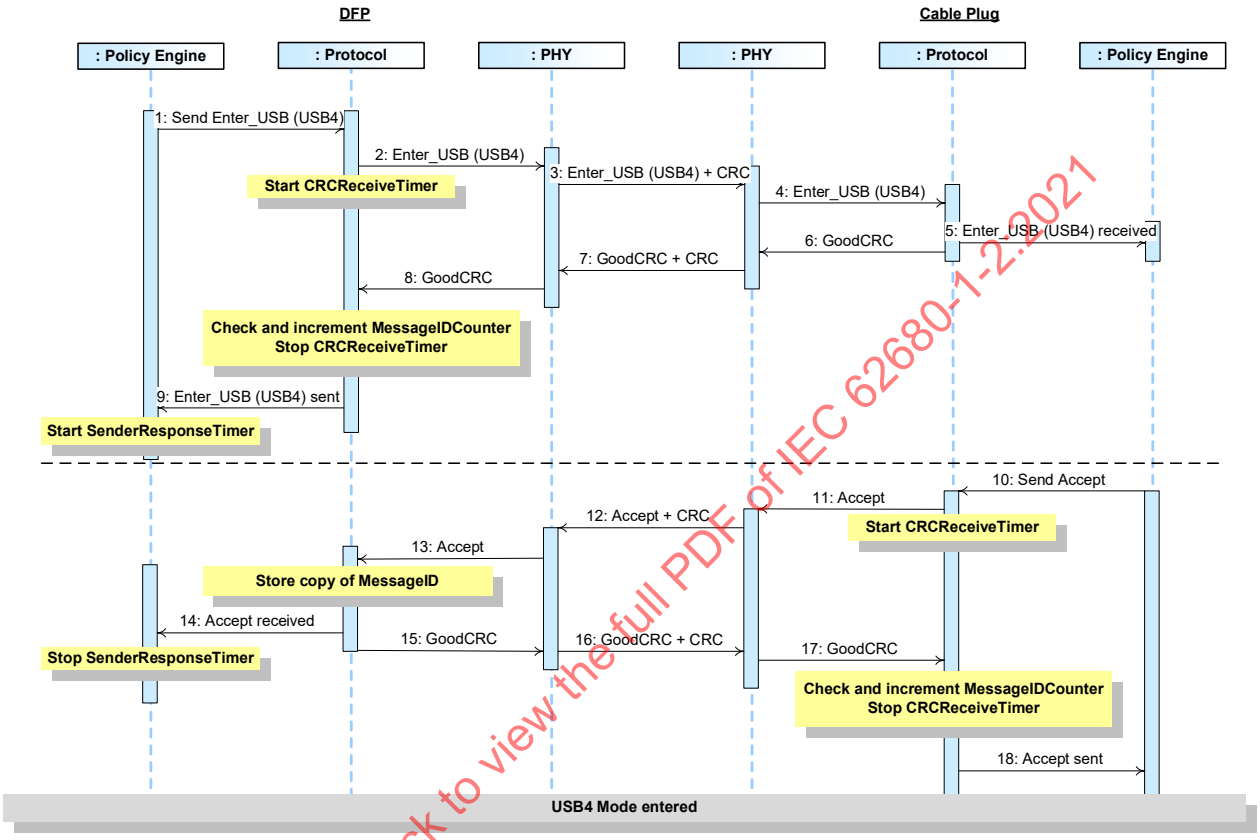


Table 8-68 below provides a detailed explanation of what happens at each labeled step in Figure 8-68 above.

Table 8-68 Steps for Cable Plug USB4 Mode Entry (Valid)

Step	DFP	Cable Plug
1	The Policy Engine directs the Protocol Layer to generate an ***Enter_USB*** Message to request entry to ***[USB4]*** mode.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Enter_USB*** Message.	Physical Layer receives the ***Enter_USB*** Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Enter_USB*** 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 ***Enter_USB*** Message information to the Policy Engine that consumes it.

Step	DFP	Cable Plug
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 ***Enter_USB*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine 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 Message.	Physical Layer appends a CRC and sends the Message.
13	Protocol Layer stores the ***MessageID*** of the incoming Message.	
14	The Protocol Layer forwards the received ***Accept*** Message information to the Policy Engine that consumes it.	
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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Accept*** Message was successfully sent.
	Cable Plug enters ***[USB4]*** operation.	

8.3.2.15.3 UFP Entering USB4 Mode (Invalid)

This is an example of an Enter USB operation where the DFP requests ***[USB4]*** mode when this is an ***Invalid*** mode of operation for the UFP. Figure 8-69 shows the Messages as they flow across the bus and within the devices to accomplish the Enter USB process.

Figure 8-69 UFP Entering USB4 Mode (Invalid)

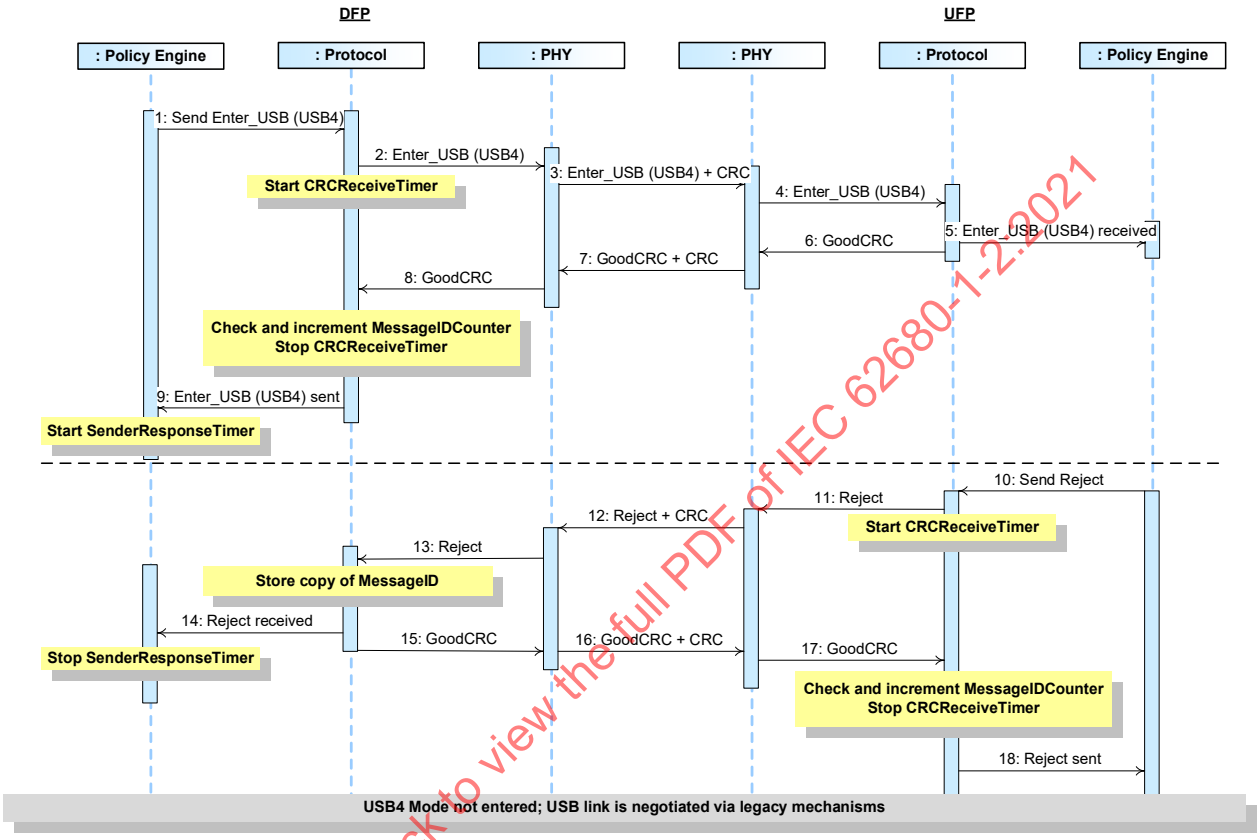


Table 8-69 below provides a detailed explanation of what happens at each labeled step in Figure 8-69 above.

Table 8-69 Steps for UFP USB4 Mode Entry (Invalid)

Step	DFP	UFP
1	The Policy Engine directs the Protocol Layer to generate an ***Enter_USB*** Message to request entry to ***[USB4]*** mode.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Enter_USB*** Message.	Physical Layer receives the ***Enter_USB*** Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Enter_USB*** 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 ***Enter_USB*** Message information to the Policy Engine that consumes it.

Step	DFP	UFP
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 ***Enter_USB*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine tells the Protocol Layer to form an ***Reject*** 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 Message.	Physical Layer appends a CRC and sends the Message.
13	Protocol Layer stores the ***MessageID*** of the incoming Message.	
14	The Protocol Layer forwards the received ***Reject*** Message information to the Policy Engine that consumes it.	
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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Reject*** Message was successfully sent.
	Port Partners do not enter ***[USB4]*** operation.	

8.3.2.15.4 Cable Plug Entering USB4 Mode (Invalid)

This is an example of an Enter USB operation where the DFP requests ***[USB4]*** mode when this is an ***Invalid*** mode of operation for the Cable Plug. Figure 8-70 Cable Plug Entering USB4 Mode (Invalid) shows the Messages as they flow across the bus and within the devices to accomplish the Enter USB process.

Figure 8-70 Cable Plug Entering USB4 Mode (Invalid)

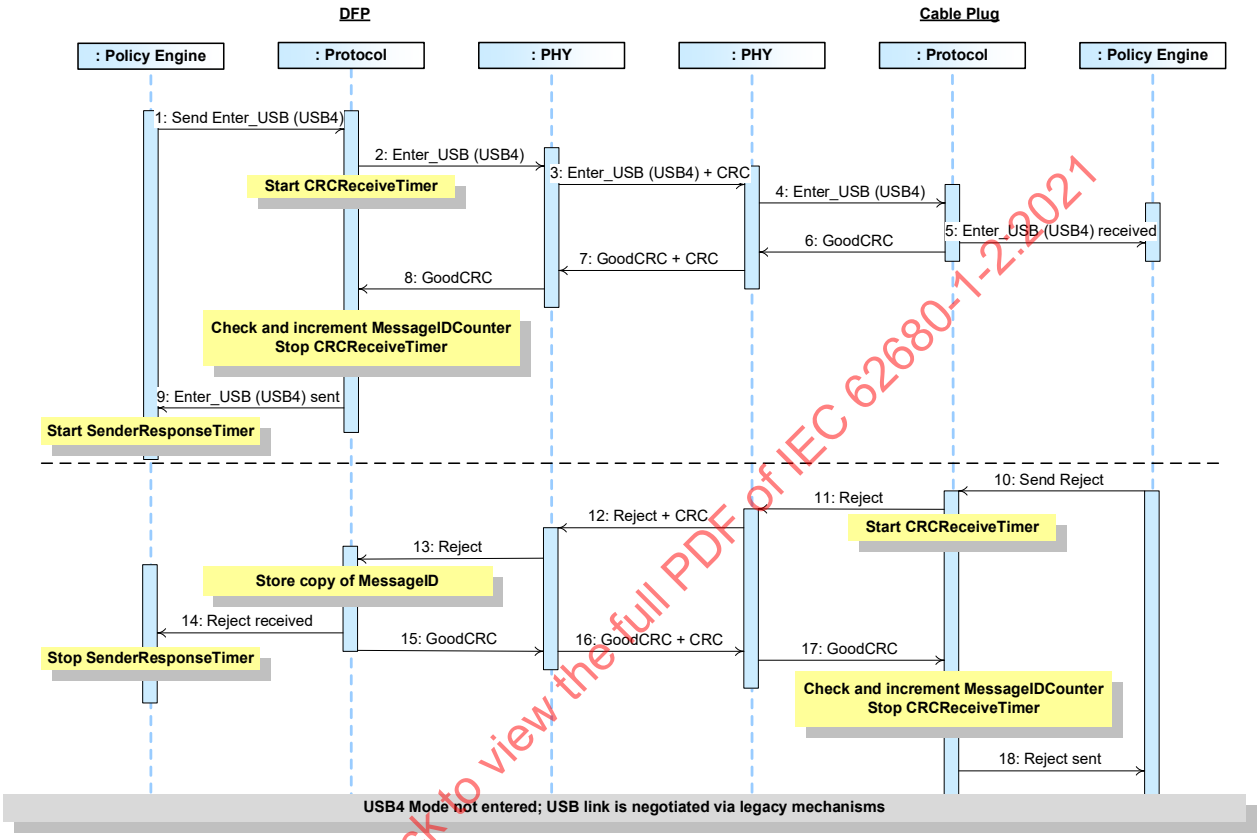


Table 8-70 below provides a detailed explanation of what happens at each labeled step in Figure 8-70 above.

Table 8-70 Steps for Cable Plug USB4 Mode Entry (Invalid)

Step	DFP	Cable Plug
1	The Policy Engine directs the Protocol Layer to generate an ***Enter_USB*** Message to request entry to ***[USB4]*** mode.	
2	Protocol Layer creates the Message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Enter_USB*** Message.	Physical Layer receives the ***Enter_USB*** Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the ***Enter_USB*** 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 ***Enter_USB*** Message information to the Policy Engine that consumes it.

Step	DFP	Cable Plug
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 ***Enter_USB*** Message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine tells the Protocol Layer to form an ***Reject*** 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 Message.	Physical Layer appends a CRC and sends the Message.
13	Protocol Layer stores the ***MessageID*** of the incoming Message.	
14	The Protocol Layer forwards the received ***Reject*** Message information to the Policy Engine that consumes it.	
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.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Reject*** Message was successfully sent.
	Cable Plug does not enter ***[USB4]*** operation.	

8.3.3 State Diagrams

8.3.3.1 Introduction to state diagrams used in Chapter 8

The state diagrams defined in Section 8.3.3 are ***Normative*** and ***Shall*** define the operation of the Power Delivery Policy Engine. Note that these state diagrams are not intended to replace a well written and robust design.

Figure 8-71 Outline of States

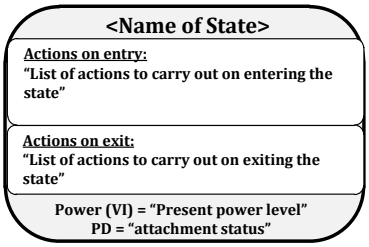


Figure 8-71 shows an outline of the states defined in the following sections. At the top there is the name of the state. This is followed by "Actions on entry" a list of actions carried out on entering the state. If

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”, “Detached”, 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 8-72 indicates how such references are made. The reference is indicated with a hatched box. The box contains the name of the state and whether the state is a DFP or UFP. It has also been necessary to indicate conditional entry to either Source Port or Sink Port state diagrams. This is achieved by the use of a bulleted list indicating the pre-conditions (see example in Figure 8-73). It is also possible that the entry and return states are the same. Figure 8-74 indicates a state reference where each referenced state corresponds to either the entry state or the exit state.

Figure 8-72 References to states

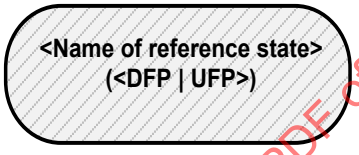
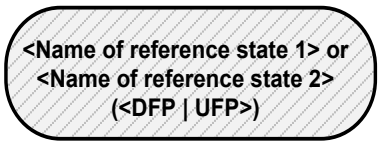


Figure 8-73 Example of state reference with conditions

Figure 8-74 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:

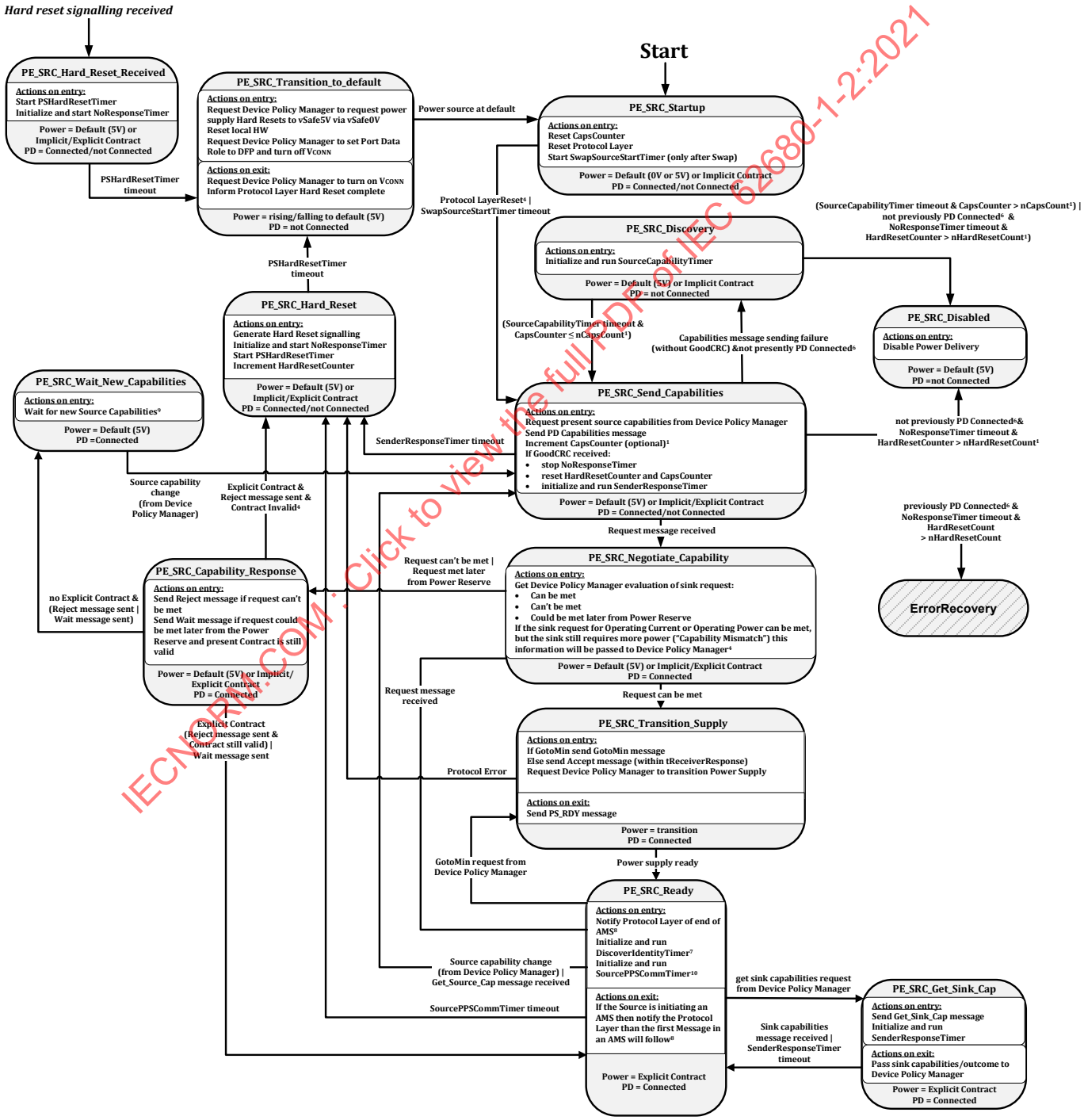
1. Message and related indications passed up to the Policy Engine from the Protocol Layer (Message sent; Message received etc.)
2. Events triggered within the Policy Engine e.g. timer timeouts.
3. Information and requests coming from the Device Policy manager relating either to Local Policy, or to other modules which the Device Policy Manager controls such as power supply and USB-C Port Control.

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 6.11.2.3) 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 8-75 below shows the state diagram for the Policy Engine in a Source Port. The following sections describe operation in each of the states.

Figure 8-75 Source Port Policy Engine State Diagram



[1] 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.

[2] 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.

[3] "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.

[4] 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.

[5] 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 Detach.

[6] Port Partners are no longer PD Connected after a Hard Reset, but 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.

[7] 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] If transition into the ***PE_SRC_Ready*** state will result in an immediate transition out of the ***PE_SRC_Ready*** state e.g. it is due to a Protocol Error that has not resulted in a Soft Reset then the notifications of the end of AMS and first Message in an AMS ***May Not*** be sent to avoid changing the Rp value unnecessarily.

[9] 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.

[10] The ***SourcePPSCommTimer*** is only initialized and run when the present Explicit Contract is for a PPS APDO. Source's that do not support PPS do not need to implement the ***SourcePPSCommTimer***.

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 6.11.2.3).

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: Sources ***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*** ≤ ***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 Partners have not been PD Connected (the Source 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 Detached until such time as a Detach/re-Attach is detected.

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 Partners have not been PD Connected (the Source 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 Detached until such time as a Detach/re-Attach is detected.

The Policy Engine ***Shall*** go to the ***ErrorRecovery*** state when:

- The Port Partners have previously been PD Connected (the Source 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*** 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.

On entry to the ***PE_SRC_Ready*** state the Source ***Shall*** notify the Protocol Layer of the end of the Atomic Message Sequence (AMS). If the transition into ***PE_SRC_Ready*** is the result of Protocol Error that has not caused a Soft Reset (see Section 8.3.3.4.1) then the notification to the Protocol Layer of the end of the AMS *Shall Not* be sent since there is a Message to be processed.

On entry to the ***PE_SRC_Ready*** state 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).

On entry to the ***PE_SRC_Ready*** state if the current Explicit Contract is for a PPS APDO, then the Policy Engine ***Shall*** do the following:

- Initialize and run the ***SourcePPSCommTimer***.

On exit from the ***PE_SRC_Ready***, if the Source is initiating an AMS then the Policy Engine ***Shall*** notify the Protocol Layer that the first Message in an AMS will follow.

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

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

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 ***NoResponseTimer***. Note that the ***NoResponseTimer*** ***Shall*** continue to run in every state until it is stopped or times out.
- initialize and run the ***PSHardResetTimer*** and increment the ***HardResetCounter***.

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***
- initialize and run the ***NoResponseTimer***. 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.

8.3.3.2.11 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 7.1.5)
- request a reset of the local hardware
- 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***:

- request the Device Policy Manager to turn on VCONN
- 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.

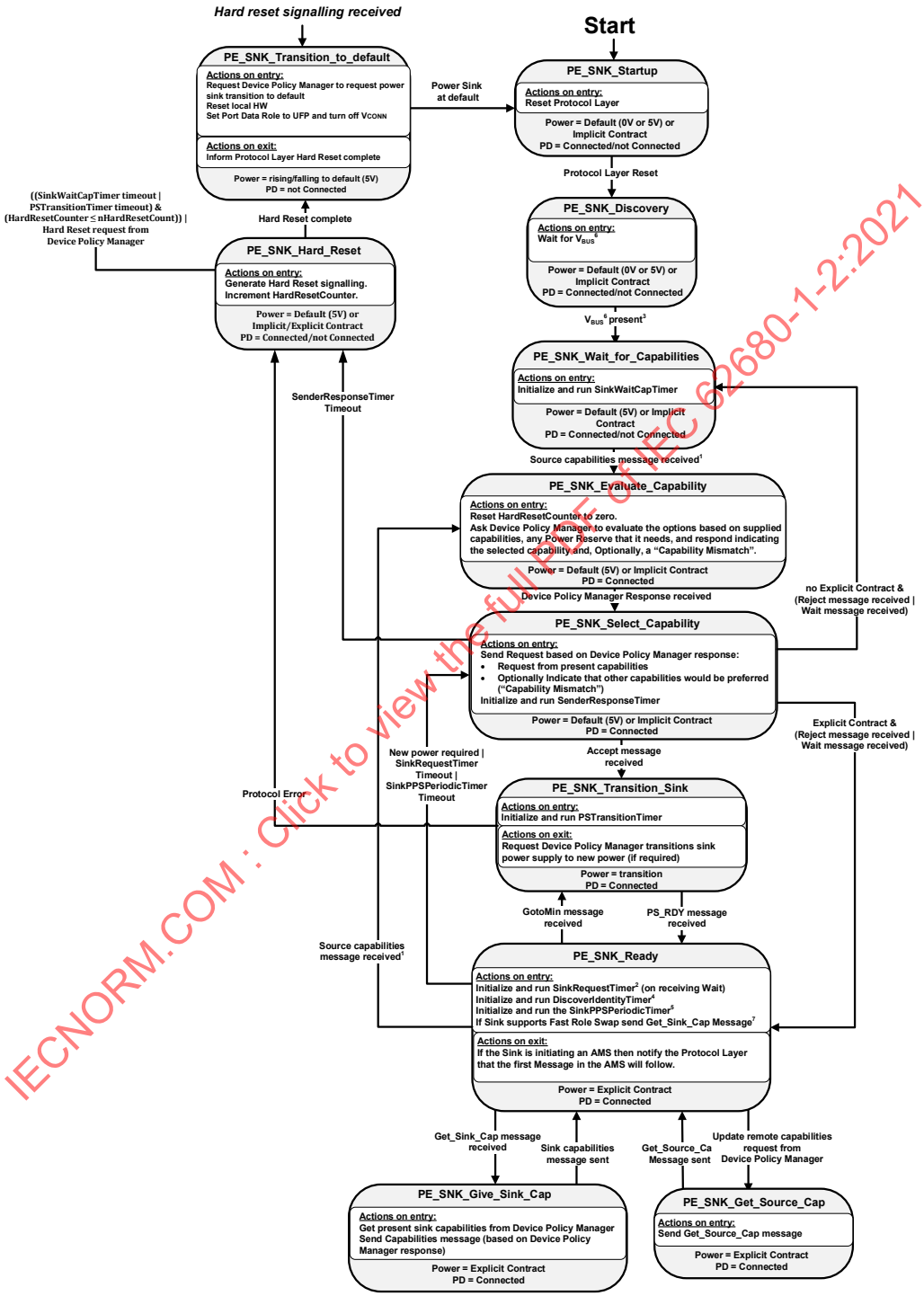
The Policy Engine ***Shall*** transition to the ***PE_SRC_Send_Capabilities*** state when:

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

8.3.3.3 Policy Engine Sink Port State Diagram

Figure 8-76 below shows the state diagram for the Policy Engine in a Sink Port. The following sections describe operation in each of the states.

Figure 8-76 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 ***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.

[3] 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.

[4] 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.

[5] The ***SinkPPSPeriodicTimer*** is only initialized and run when the present Explicit Contract is for a PPS APDO. Sink's that do not support PPS do not need to implement the ***SinkPPSPeriodicTimer***.

[6] A Sink that is a VPD ***May*** use VCONN as a proxy for V_{BUS}.

[7] To be sent once, and only required if Fast Role Swap is supported by the Sink.

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 6.11.2.3).

Once the reset process completes, the Policy Engine ***Shall*** transition to the ***PE_SNK_Discovery*** state.

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.

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.

8.3.3.3.6 PE_SNK_Transition_Sink State

On entry to the ***PE_SNK_Transition_Sink*** state the Policy Engine ***Shall*** initialize and run the ***PSTransitionTimer*** (timeout will lead to a Hard Reset see Section 8.3.3.3.8 and ***Shall*** then request the Device Policy Manager to transition the Sink's power supply to the new power level. Note that if there is no power level change the Device Policy Manager ***Should Not*** affect any change to the power supply.

On exit from the ***PE_SNK_Transition_Sink*** state the Policy Engine ***Shall*** request the Device Policy Manager to transition the Sink's power supply to the new power level.

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

- A ***PS_RDY*** Message is received from the Source.

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

- A Protocol Error occurs.

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 if this is a DFP which needs to establish communication with a Cable Plug, then the Policy Engine ***Shall*** do the following:

- Initialize and run the ***DiscoverIdentityTimer*** (no ***GoodCRC*** Message response yet received to ***Discover Identity*** Message).

On entry to the ***PE_SNK_Ready*** state if the current Explicit Contract is for a PPS APDO, then the Policy Engine ***Shall*** do the following:

- Initialize and run the ***SinkPPSPeriodicTimer***.

On entry to the ***PE_SNK_Ready*** state if the Sink supports Fast Role Swap, then the Policy Engine ***Shall*** do the following:

- Send a ***Get_Sink_Cap*** Message.

On exit from the ***PE_SNK_Ready*** state, if the transition is as a result of a DPM request to start a new Atomic Message Sequence (AMS) then the Policy Engine ***Shall*** notify the Protocol Layer that the first Message in an AMS will follow.

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 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 |
- ***PSTransitionTimer*** timeout) &
- (***HardResetCounter*** ≤ ***nHardResetCount***)) |
- Hard Reset request from Device Policy Manager.

Note: if the ***SinkWaitCapTimer*** times out and the ***HardResetCounter*** is greater than ***nHardResetCount*** the Sink ***Shall*** assume that the Source is non-responsive.

Note: The ***HardResetCounter*** 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
- request the Device Policy Manger 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:

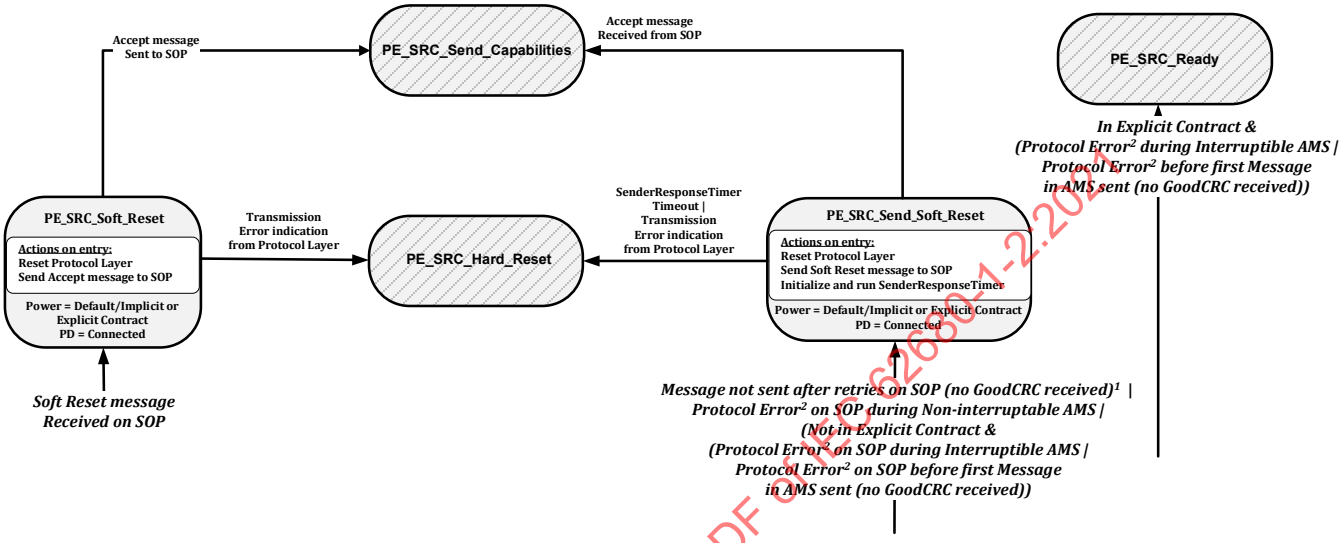
- The ***Get_Source_Cap*** Message is sent.

8.3.3.4 SOP Soft Reset and Protocol Error State Diagrams

8.3.3.4.1 Source Port Soft Reset and Protocol Error State Diagram

Figure 8-77 below shows the state diagram for the Policy Engine in a Source Port when performing a Soft Reset of its Port Partner i.e. using ***SOP***. The following sections describe operation in each of the states.

Figure 8-77 Source Port Soft Reset and Protocol Error State Diagram



[1] Excludes the ***Soft_Reset*** Message itself.

[2] An Unrecognized or Unsupported Message received on ***SOP*** will result in a ***Not_Supported*** Message response being generated on ***SOP*** (see Section 6.3.16).

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 on ***SOP*** is detected by the Protocol Layer during a Non-interruptible AMS (see Section 6.8.1) or
- A Message has not been sent after retries to the Sink or
- When not in an Explicit Contract and
    - Protocol Errors occurred on ***SOP*** during an Interruptible AMS or
    - Protocol Errors occurred on ***SOP*** 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.

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 on ***SOP*** (without GoodCRC) and the source is not presently Attached (as indicated in Figure 8-75). 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 Hard Reset Signaling will be generated.
- During a Power Role Swap when the power supply is in transition (see Section 8.3.3.18.3 and Section 8.3.3.18.4). In this case USB Type-C Error Recovery will be triggered directly.
- During a Data Role Swap when there is a mismatch in the Port Date Role field (see Section 6.2.1.1.6). 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:

- When in an Explicit Contract
  - Protocol Errors occurred on ***SOP*** during an Interruptible AMS.
  - Protocol Errors occurred on ***SOP*** 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 ***SOP*** Protocol Layer to perform a Soft Reset, then ***Shall*** send a ***Soft_Reset*** Message to the Sink on ***SOP***, 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 on ***SOP***.

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

- A ***SenderResponseTimer*** timeout occurs.
- Or the Protocol Layer indicates that a transmission error has occurred.

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 on ***SOP*** from the Protocol Layer.

On entry to the ***PE_SRC_Soft_Reset*** state the Policy Engine ***Shall*** reset the ***SOP*** Protocol Layer and ***Shall*** then request the Protocol Layer to send an ***Accept*** Message on ***SOP***.

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 on ***SOP***.

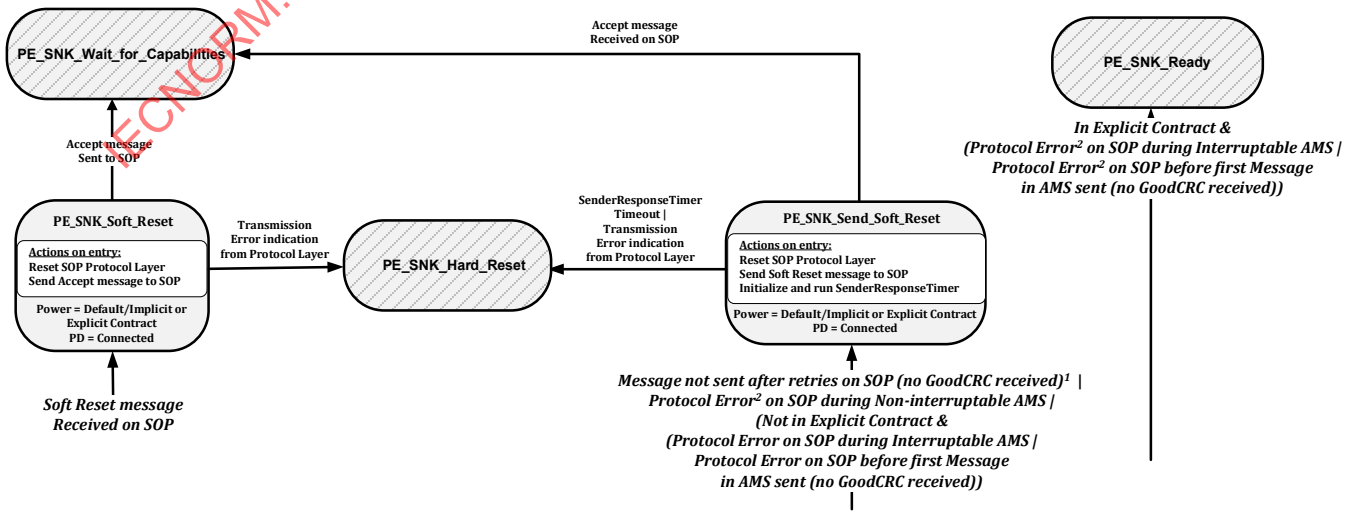
The Policy Engine ***Shall*** transition to the ***PE_SRC_Hard_Reset*** state when:

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

8.3.3.4.2 SOP Sink Port Soft Reset and Protocol Error State Diagram

Figure 8-78 below shows the state diagram for the Policy Engine in a Sink Port when performing a Soft Reset of its Port Partner i.e. using ***SOP***. The following sections describe operation in each of the states.

Figure 8-78 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 ***Not_Supported*** Message response being generated (see Section 6.3.16).

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 on ***SOP*** is detected by the Protocol Layer during a Non-interruptible AMS (see Section 6.8.1) or
- A Message has not been sent after retries to the Sink or
- When not in an Explicit Contract and
    - Protocol Errors occurred on ***SOP*** during an Interruptible AMS or
    - Protocol Errors occurred on ***SOP*** 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.

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 8.3.3.18.3 and Section 8.3.3.18.4). 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:

- When in an Explicit Contract
    - Protocol Errors occurred on ***SOP*** during an Interruptible AMS.
    - Protocol Errors occurred on ***SOP*** 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 ***SOP*** Protocol Layer to perform a Soft Reset, then ***Shall*** send a ***Soft_Reset*** Message on ***SOP*** 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 on ***SOP***.

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.

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 on ***SOP*** from the Protocol Layer.

On entry to the ***PE_SNK_Soft_Reset*** state the Policy Engine ***Shall*** reset the ***SOP*** Protocol Layer and ***Shall*** then request the Protocol Layer to send an ***Accept*** Message on ***SOP***.

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

- The ***Accept*** Message has been sent on ***SOP***.

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

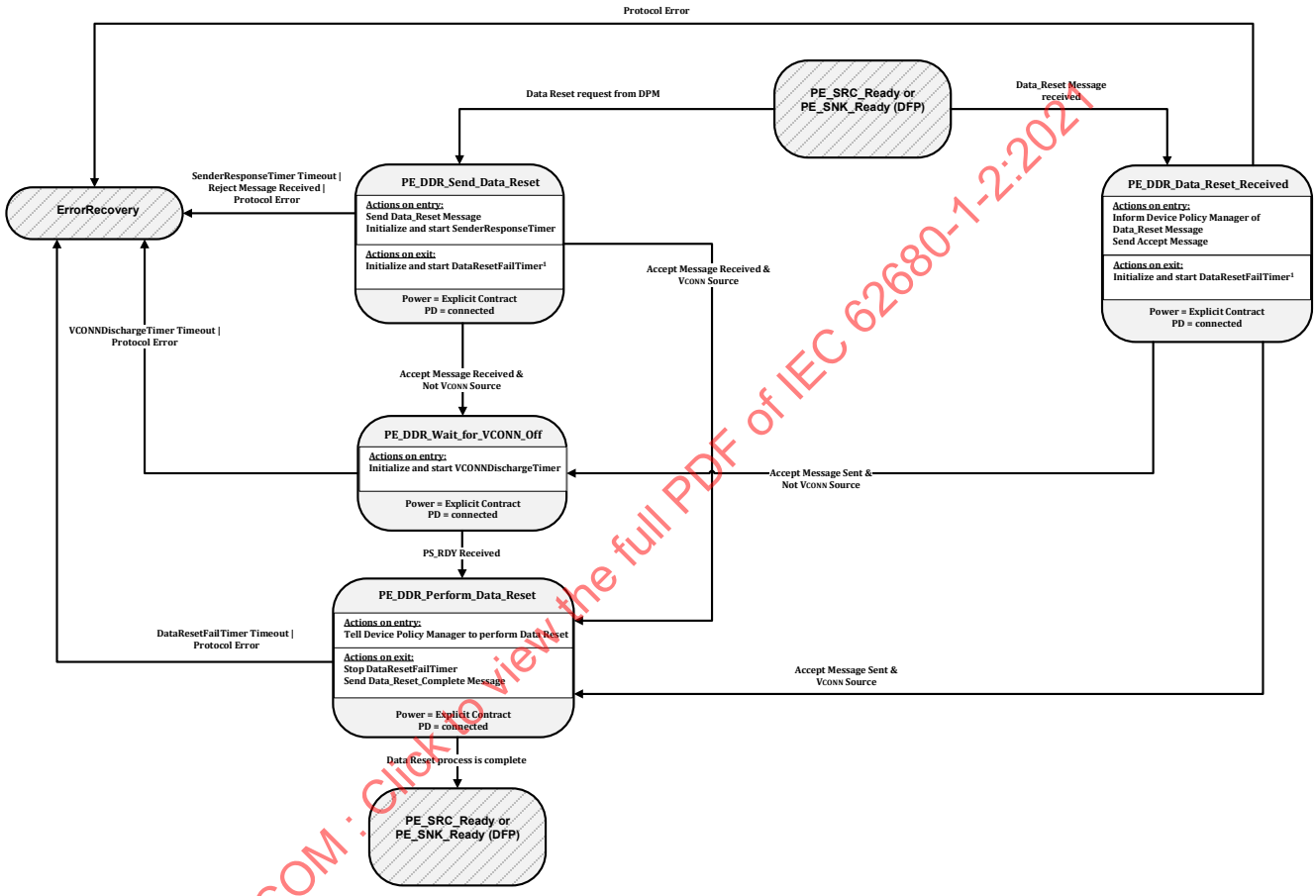
- The Protocol Layer indicates that a transmission error has occurred.

8.3.3.5 Data Reset State Diagrams

8.3.3.5.1 DFP Data_Reset Message State Diagrams

Figure 8-79 shows the state diagram for a ***Data_Reset*** Message sent or received by a DFP.

Figure 8-79 DFP Data_Reset Message State Diagram



[1] Note that the ***DataResetFailTimer*** ***Shall*** continue to run in every state until it is stopped or times out.

8.3.3.5.1.1 PE_DDR_Send_Data_Reset State

The ***PE_DDR_Send_Data_Reset*** State ***Shall*** be entered from the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** State when requested by the Device Policy Manager.

On entry to the ***PE_DDR_Send_Data_Reset*** State the Policy Engine ***Shall*** request the Protocol Layer to send a ***Data_Reset*** Message and then initialize and start the ***SenderResponseTimer***.

On exit from the ***PE_DDR_Send_Data_Reset*** State the Policy Engine ***Shall*** initialize and start the ***DataResetFailTimer***.

The Policy Engine ***Shall*** transition to the ***PE_DDR_Perform_Data_Reset*** State when:

- An ***Accept*** Message has been received and
- The DFP is presently the VCONN Source.

The Policy Engine ***Shall*** transition to the ***PE_DDR_Wait_For_VCONN_Off*** State when:

- An ***Accept*** Message has been received and

- The DFP is not presently the VCONN Source.

The Policy Engine ***Shall*** transition to ***ErrorRecovery*** when:

- A ***SenderResponseTimer*** timeout occurs or
- A ***Reject*** Message is received or
- A Protocol Error occurs.

8.3.3.5.1.2 PE_DDR_Data_Reset_Received State

The ***PE_DDR_Data_Reset_Received*** State ***Shall*** be entered from the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** State when a ***Data_Reset*** Message is received.

On entry to the ***PE_DDR_Data_Reset_Received*** State the Policy Engine ***Shall*** inform the Device Policy Manager and then ***Shall*** send an ***Accept*** Message.

On exit from the ***PE_DDR_Data_Reset_Received*** State the Policy Engine ***Shall*** initialize and start the ***DataResetFailTimer***.

The Policy Engine ***Shall*** transition to the ***PE_DDR_Perform_Data_Reset*** State when:

- An ***Accept*** Message has been sent and
- The DFP is presently the VCONN Source.

The Policy Engine ***Shall*** transition to the ***PE_DDR_Wait_For_VCONN_Off*** State when:

- An ***Accept*** Message has been sent and
- The DFP is not presently the VCONN Source.

The Policy Engine ***Shall*** transition to ***ErrorRecovery*** when:

- A Protocol Error occurs.

8.3.3.5.1.3 PE_DDR_Wait_For_VCONN_Off State

On entry to the ***PE_DDR_Wait_For_VCONN_Off*** State the Policy Engine ***Shall*** initialize and start the ***VCONNDischargeTimer***.

The Policy Engine ***Shall*** transition to the ***PE_DDR_Perform_Data_Reset*** State when:

- A ***PS_RDY*** Message is received.

The Policy Engine ***Shall*** transition to ***ErrorRecovery*** when:

- The ***VCONNDischargeTimer*** has timed out or
- A Protocol Error occurs.

8.3.3.5.1.4 PE_DDR_Perform_Data_Reset State

On entry to the ***PE_DDR_Perform_Data_Reset*** State the Policy Engine ***Shall*** request the Device Policy Manager to complete the Data Reset process as defined in Section 6.3.14.

On exit from the ***PE_DDR_Perform_Data_Reset*** State the Policy Engine ***Shall*** stop the ***DataResetFailTimer*** and send a ***Data_Reset_Complete*** Message.

The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** State depending on the DFP's Power Role when:

- The DPM indicates that Data Reset process is complete (see Section 6.3.14).

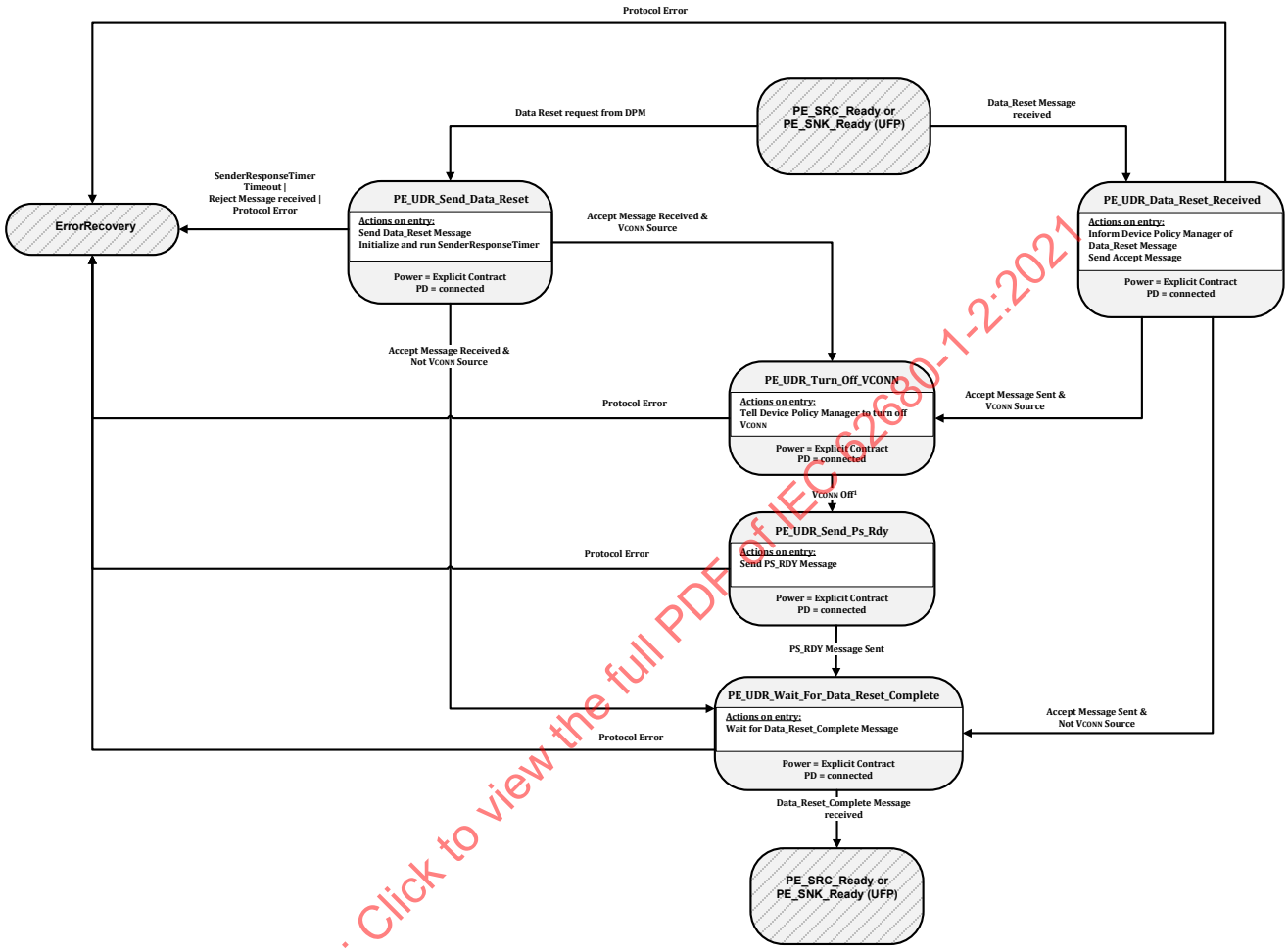
The Policy Engine ***Shall*** transition to ***ErrorRecovery*** when:

- The ***DataResetFailTimer*** times out
- A Protocol Error occurs.

8.3.3.5.2 UFP Data_Reset Message State Diagrams

Figure 8-80 shows the state diagram for a ***Data_Reset*** Message sent or received by a UFP.

Figure 8-80 UFP Data_Reset Message State Diagram



[1] VCONN ***Shall*** be fully discharged see Section 7.1.15.

8.3.3.5.2.1 PE_UDR_Send_Data_Reset State

The ***PE_UDR_Send_Data_Reset*** State ***Shall*** be entered from the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** State when requested by the Device Policy Manager.

On entry to the ***PE_UDR_Send_Data_Reset*** State the Policy Engine ***Shall*** request the Protocol Layer to send a ***Data_Reset*** Message and then initialize and start the ***SenderResponseTimer***.

The Policy Engine ***Shall*** transition to the ***PE_UDR_Turn_Off_VCONN*** State when:

- An ***Accept*** Message has been received and
- The UFP is presently the VCONN Source.

The Policy Engine ***Shall*** transition to the ***PE_UDR_Wait_For_Data_Reset_Complete*** State when:

- An ***Accept*** Message has been received and
- The UFP is not presently the VCONN Source.

The Policy Engine ***Shall*** transition to ***ErrorRecovery*** when:

- The ***SenderResponseTimer*** has timed out or

- A ***Reject*** Message has been received or
- A Protocol Error occurs.

8.3.3.5.2.2 PE_UDR_Data_Reset_Received State

The ***PE_UDR_Data_Reset_Received*** State ***Shall*** be entered from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** State when a ***Data_Reset*** Message is received.

On entry to the ***PE_UDR_Data_Reset_Received*** State the Policy Engine ***Shall*** inform the Device Policy Manager and then ***Shall*** send an ***Accept*** Message.

The Policy Engine ***Shall*** transition to the ***PE_UDR_Turn_Off_VCONN*** State when:

- An ***Accept*** Message has been sent and
- The UFP is presently the VCONN Source.

The Policy Engine ***Shall*** transition to the ***PE_UDR_Wait_For_Data_Reset_Complete*** State when:

- An ***Accept*** Message has been sent and
- The UFP is not presently the VCONN Source.

The Policy Engine ***Shall*** transition to ***ErrorRecovery*** when:

- A Protocol Error occurs.

8.3.3.5.2.3 PE_UDR_Turn_Off_VCONN State

On entry to the ***PE_UDR_Turn_Off_VCONN*** State the Policy Engine ***Shall*** request the Device Policy Manager to turn off VCONN.

The Policy Engine ***Shall*** transition to the ***PE_UDR_Send_Ps_Rdy*** State when:

- The DPM indicates that VCONN has been turned off (VCONN below vRaReconnect see ***[USB Type-C 2.0]***).

The Policy Engine ***Shall*** transition to ***ErrorRecovery*** when:

- A Protocol Error occurs.

8.3.3.5.2.4 PE_UDR_Send_Ps_Rdy State

On entry to the ***PE_UDR_Send_Ps_Rdy*** State the Policy Engine ***Shall*** send a ***PS_RDY*** Message.

The Policy Engine ***Shall*** transition to the ***PE_UDR_Wait_For_Data_Reset_Complete*** State when:

- The ***PS_RDY*** Message has been sent.

The Policy Engine ***Shall*** transition to ***ErrorRecovery*** when:

- A Protocol Error occurs.

8.3.3.5.2.5 PE_UDR_Wait_For_Data_Reset_Complete State

On entry to the ***PE_UDR_Wait_For_Data_Reset_Complete*** State the Policy Engine ***Shall*** wait for the ***Data_Reset_Complete*** Message.

The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** State depending on the UFP's Power Role when:

- The ***Data_Reset_Complete*** Message is received.

The Policy Engine ***Shall*** transition to ***ErrorRecovery*** when:

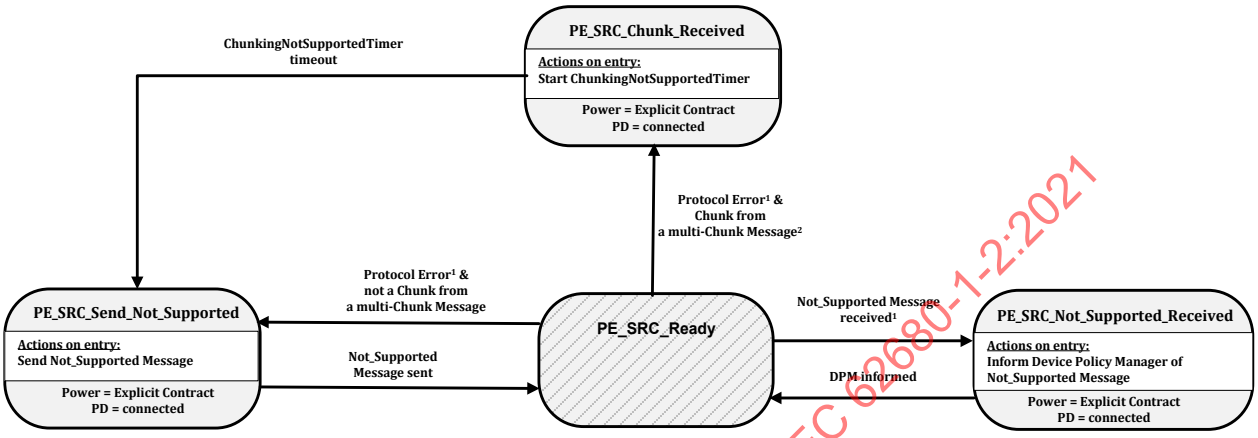
- A Protocol Error occurs.

8.3.3.6 Not Supported Message State Diagrams

8.3.3.6.1 Source Port Not Supported Message State Diagram

Figure 8-81 shows the state diagram for a ***Not_Supported*** Message sent or received by a Source Port.

Figure 8-81 Source Port Not Supported Message State Diagram



[1] Transition can either be the result of a Protocol Error during an interruptible AMS or as a result of an unsupported Message being received in the ***PE_SRC_Ready*** state directly (see also Section 8.3.3.4.1).

[2] Transition can only occur where a manufacturer has opted not to implement a Chunking state machine (see Section 6.11.2.1) and is communicating with a system which is attempting to send it Chunks.

8.3.3.6.1.1 PE_SRC_Send_Not_Supported State

The ***PE_SRC_Send_Not_Supported*** state ***Shall*** be entered from the ***PE_SRC_Ready*** state either as the result of a Protocol Error received during an interruptible AMS or as a result of an unsupported Message being received in the ***PE_SRC_Ready*** state directly except for the first Chunk in a multi-Chunk Message (see also Section 6.11.2.1 and Section 8.3.3.4.1).

On entry to the ***PE_SRC_Send_Not_Supported*** state (from the ***PE_SRC_Ready*** state) the Policy Engine ***Shall*** request the Protocol Layer to send a ***Not_Supported*** Message.

The Policy Engine ***Shall*** transition back to the previous state (***PE_SRC_Ready*** see Figure 8-81) when:

- The ***Not_Supported*** Message has been successfully sent.

8.3.3.6.1.2 PE_SRC_Not_Supported_Received State

The ***PE_SRC_Not_Supported_Received*** state ***Shall*** be entered from the ***PE_SRC_Ready*** state when a ***Not_Supported*** Message is received.

On entry to the ***PE_SRC_Not_Supported_Received*** state the Policy Engine ***Shall*** inform the Device Policy Manager.

The Policy Engine ***Shall*** transition back to the previous state (***PE_SRC_Ready*** see Figure 8-81) when:

- The Device Policy Manager has been informed.

8.3.3.6.1.3 PE_SRC_Chunk_Received State

The ***PE_SRC_Chunk_Received*** state ***Shall*** be entered from the ***PE_SRC_Ready*** state either as the result of a Protocol Error received during an interruptible AMS or as a result of an unsupported Message being received in the ***PE_SRC_Ready*** state directly where the Message is a Chunk in a multi-Chunk Message (see also Section 6.6.18.1 and Section 8.3.3.4.1).

On entry to the *PE_SRC_Chunk_Received* state (from the *PE_SRC_Ready* state) the Policy Engine ***Shall*** initialize and run the *ChunkingNotSupportedTimer*.

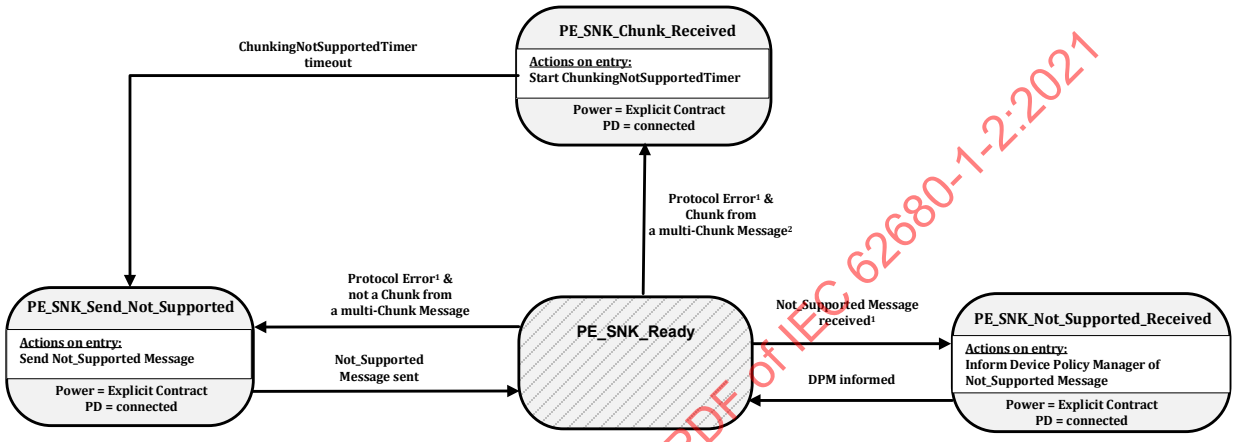
The Policy Engine ***Shall*** transition to *PE_SRC_Send_Not_Supported* when:

- The *ChunkingNotSupportedTimer* has timed out.

8.3.3.6.2 Sink Port Not Supported Message State Diagram

Figure 8-82 shows the state diagram for a *Not_Supported* Message sent or received by a Sink Port.

Figure 8-82 Sink Port Not Supported Message State Diagram



[1] Transition can either be the result of a Protocol Error during an interruptible AMS or as a result of an unsupported Message being received in the *PE_SNK_Ready* state directly (see also Section 8.3.3.4.2).

8.3.3.6.2.1 PE_SNK_Send_Not_Supported State

The *PE_SNK_Send_Not_Supported* state ***Shall*** be entered from the *PE_SNK_Ready* state either as the result of a Protocol Error received during an interruptible AMS or as a result of an unsupported Message being received in the *PE_SNK_Ready* state directly except for the first Chunk in a multi-Chunk Message (see also Section 6.11.2.1 and Section 8.3.3.4.1).

On entry to the *PE_SNK_Send_Not_Supported* state (from the *PE_SNK_Ready* state) the Policy Engine ***Shall*** request the Protocol Layer to send a *Not_Supported* Message.

The Policy Engine ***Shall*** transition back to the previous state (*PE_SNK_Ready* see Figure 8-82) when:

- The *Not_Supported* Message has been successfully sent.

8.3.3.6.2.2 PE_SNK_Not_Supported_Received State

The *PE_SNK_Not_Supported_Received* state ***Shall*** be entered from the *PE_SNK_Ready* state when a *Not_Supported* Message is received.

On entry to the *PE_SNK_Not_Supported_Received* state the Policy Engine ***Shall*** inform the Device Policy Manager.

The Policy Engine ***Shall*** transition back to the previous state (*PE_SNK_Ready* see Figure 8-82) when:

- The Device Policy Manager has been informed.

8.3.3.6.2.3 PE_SNK_Chunk_Received State

The *PE_SNK_Chunk_Received* state ***Shall*** be entered from the *PE_SNK_Ready* state either as the result of a Protocol Error received during an interruptible AMS or as a result of an unsupported Message being received in the *PE_SNK_Ready* state directly where the Message is a Chunk in a multi-Chunk Message (see also Section 6.6.18.1 and Section 8.3.3.4.1).

On entry to the *PE_SNK_Chunk_Received* state (from the *PE_SNK_Ready* state) the Policy Engine ***Shall*** initialize and run the *ChunkingNotSupportedTimer*.

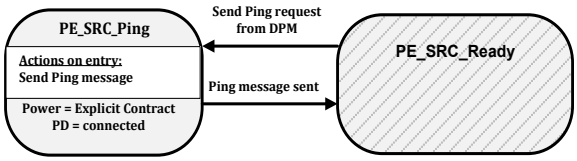
The Policy Engine ***Shall*** transition to *PE_SNK_Send_Not_Supported* when:

- The *ChunkingNotSupportedTimer* has timed out.

8.3.3.7 Source Port Ping State Diagram

Figure 8-81 shows the state diagram for a *Ping* Message from a Source Port.

Figure 8-83 Source Port Ping State Diagram



8.3.3.7.1 PE_SRC_Ping State

On entry to the *PE_SRC_Ping* state (from the *PE_SRC_Ready* state) 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_Ready*) (see Figure 8-75) when:

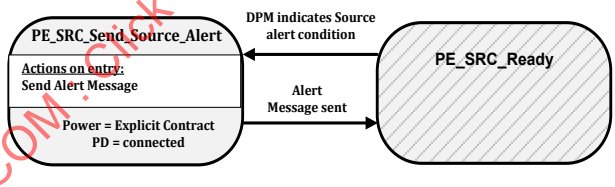
- The *Ping* Message has been successfully sent.

8.3.3.8 Source Alert State Diagrams

8.3.3.8.1 Source Port Source Alert State Diagram

Figure 8-84 shows the state diagram for an Alert Message sent by a Source Port.

Figure 8-84 Source Port Source Alert State Diagram



8.3.3.8.1.1 PE_SRC_Send_Source_Alert State

The *PE_SRC_Send_Source_Alert* state ***Shall*** be entered from the *PE_SRC_Ready* state when the Device Policy Manager indicates that there is a Source alert condition to be reported.

On entry to the *PE_SRC_Send_Source_Alert* state the Policy Engine ***Shall*** request the Protocol Layer to send an Alert Message.

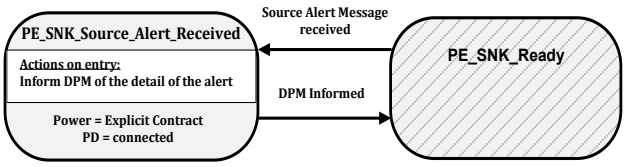
The Policy Engine ***Shall*** transition back to *PE_SRC_Ready* (see Figure 8-75) when:

- The *Alert* Message has been successfully sent.

8.3.3.8.2 Sink Port Source Alert State Diagram

Figure 8-85 shows the state diagram for an Alert Message received by a Sink Port.

Figure 8-85 Sink Port Source Alert State Diagram



8.3.3.8.2.1 PE_SNK_Source_Alert_Received State

The ***PE_SNK_Source_Alert_Received*** state ***Shall*** be entered from the ***PE_SNK_Ready*** state when an Alert Message is received.

On entry to the ***PE_SNK_Source_Alert_Received*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the details of the Source alert.

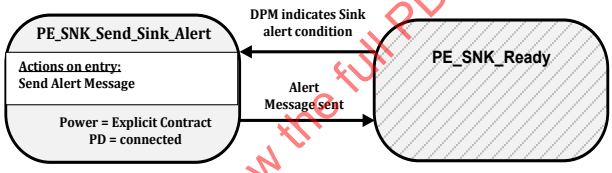
The Policy Engine ***Shall*** transition back to ***PE_SNK_Ready*** (see Figure 8-76) when:

- The DPM has been informed.

8.3.3.8.3 Sink Port Sink Alert State Diagram

Figure 8-86 shows the state diagram for an Alert Message sent by a Sink Port.

Figure 8-86 Sink Port Sink Alert State Diagram



8.3.3.8.3.1 PE_SNK_Send_Sink_Alert State

The ***PE_SNK_Send_Sink_Alert*** state ***Shall*** be entered from the ***PE_SNK_Ready*** state when the Device Policy Manager indicates that there is a Source alert condition to be reported.

On entry to the ***PE_SNK_Send_Sink_Alert*** state the Policy Engine ***Shall*** request the Protocol Layer to send an Alert Message.

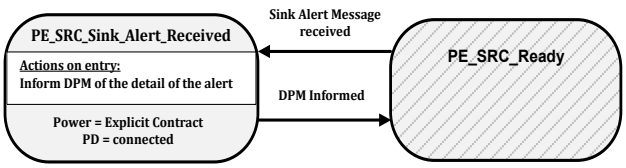
The Policy Engine ***Shall*** transition back to ***PE_SNK_Ready*** (see Figure 8-76) when:

- The ***Alert*** Message has been successfully sent.

8.3.3.8.4 Source Port Sink Alert State Diagram

Figure 8-87 shows the state diagram for an Alert Message received by a Source Port.

Figure 8-87 Source Port Sink Alert State Diagram



8.3.3.8.4.1 PE_SRC_Sink_Alert_Received State

The ***PE_SRC_Sink_Alert_Received*** state ***Shall*** be entered from the ***PE_SRC_Ready*** state when an Alert Message is received.

On entry to the ***PE_SRC_Sink_Alert_Received*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the details of the Source alert.

The Policy Engine ***Shall*** transition back to ***PE_SRC_Ready*** (see Figure 8-75) when:

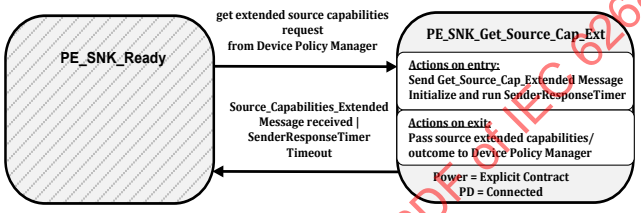
- The DPM has been informed.

8.3.3.9 Source Capabilities Extended State Diagrams

8.3.3.9.1 Sink Port Get Source Capabilities Extended State Diagram

Figure 8-88 shows the state diagram for a Sink on receiving a request from the Device Policy Manager to get the Port Partner's extended Source capabilities. See also Section 6.5.1.

Figure 8-88 Sink Port Get Source Capabilities Extended State Diagram



8.3.3.9.1.1 PE_SNK_Get_Source_Cap_Ext State

The Policy Engine ***Shall*** transition to the ***PE_SNK_Get_Source_Cap_Ext*** state, from the ***PE_SNK_Ready*** state, due to a request to get the remote extended source capabilities from the Device Policy Manager.

On entry to the ***PE_SNK_Get_Source_Cap_Ext*** state the Policy Engine ***Shall*** send a ***Get_Source_Cap_Extended*** Message and initialize and run the ***SenderResponseTimer***.

On exit from the ***PE_SNK_Get_Source_Cap_Ext*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the outcome (capabilities or response timeout).

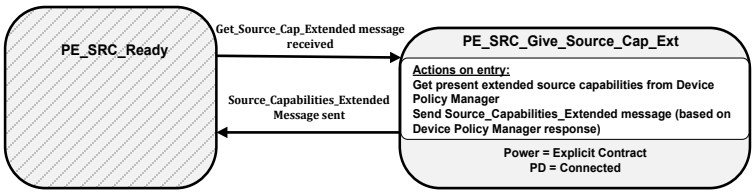
The Policy Engine ***Shall*** transition back to the ***PE_SNK_Ready*** state (see Figure 8-76) when:

- A ***Source_Capabilities_Extended*** Message is received
- Or ***SenderResponseTimer*** times out.

8.3.3.9.2 Source Give Source Capabilities Extended State Diagram

Figure 8-89 shows the state diagram for a Source on receiving a ***Get_Source_Cap_Extended*** Message. See also Section 6.5.1.

Figure 8-89 Source Give Source Capabilities Extended State Diagram



8.3.3.9.2.1 PE_SRC_Give_Source_Cap_Ext State

The Policy Engine ***Shall*** transition to the ***PE_SRC_Give_Source_Cap_Ext*** state, from the ***PE_SRC_Ready*** state, when a ***Get_Source_Cap_Extended*** Message is received.

On entry to the ***PE_SRC_Give_Source_Cap_Ext*** state the Policy Engine ***Shall*** request the present extended Source capabilities from the Device Policy Manager and then send a ***Source_Capabilities_Extended*** Message based on these capabilities.

The Policy Engine ***Shall*** transition back to the ***PE_SRC_Ready*** state (see Figure 8-75) when:

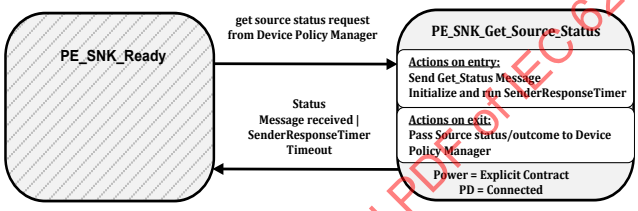
- The ***Source_Capabilities_Extended*** Message has been successfully sent.

8.3.3.10 Status State Diagrams

8.3.3.10.1 Sink Port Get Source Status State Diagram

Figure 8-90 shows the state diagram for a Sink on receiving a request from the Device Policy Manager to get the Port Partner's Source status. See also Section 6.5.2.

Figure 8-90 Sink Port Get Source Status State Diagram



8.3.3.10.1.1 PE_SNK_Get_Source_Status State

The Policy Engine ***Shall*** transition to the ***PE_SNK_Get_Source_Status*** state, from the ***PE_SNK_Ready*** state, due to a request to get the remote source status from the Device Policy Manager.

On entry to the ***PE_SNK_Get_Source_Status*** state the Policy Engine ***Shall*** send a ***Get_Status*** Message and initialize and run the ***SenderResponseTimer***.

On exit from the ***PE_SNK_Get_Source_Status*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the outcome (status or response timeout).

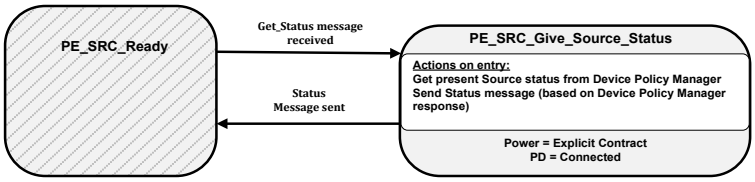
The Policy Engine ***Shall*** transition back to the ***PE_SNK_Ready*** state (see Figure 8-76) when:

- A ***Status*** Message is received
- Or ***SenderResponseTimer*** times out.

8.3.3.10.2 Source Give Source Status State Diagram

Figure 8-91 shows the state diagram for a Source on receiving a ***Get_Status*** Message. See also Section 6.5.1.

Figure 8-91 Source Give Source Status State Diagram



8.3.3.10.2.1 PE_SRC_Give_Source_Status State

The Policy Engine ***Shall*** transition to the ***PE_SRC_Give_Source_Status*** state, from the ***PE_SRC_Ready*** state, when a ***Get_Status*** Message is received.

On entry to the ***PE_SRC_Give_Source_Status*** state the Policy Engine ***Shall*** request the present Source status from the Device Policy Manager and then send a ***Status*** Message based on these capabilities.

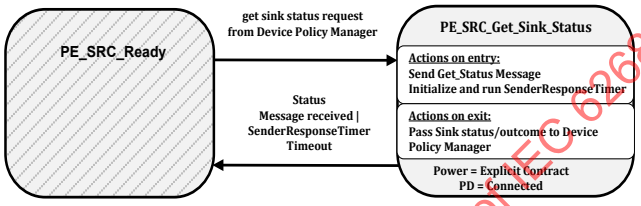
The Policy Engine ***Shall*** transition back to the ***PE_SRC_Ready*** state (see Figure 8-75) when:

- The ***Status*** Message has been successfully sent.

8.3.3.10.3 Source Port Get Sink Status State Diagram

Figure 8-92 shows the state diagram for a Source on receiving a request from the Device Policy Manager to get the Port Partner's Sink status. See also Section 6.5.2.

Figure 8-92 Source Port Get Sink Status State Diagram



8.3.3.10.3.1 PE_SRC_Get_Sink_Status State

The Policy Engine ***Shall*** transition to the ***PE_SRC_Get_Sink_Status*** state, from the ***PE_SRC_Ready*** state, due to a request to get the remote source status from the Device Policy Manager.

On entry to the ***PE_SRC_Get_Sink_Status*** state the Policy Engine ***Shall*** send a ***Status*** Message and initialize and run the ***SenderResponseTimer***.

On exit from the ***PE_SRC_Get_Sink_Status*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the outcome (status or response timeout).

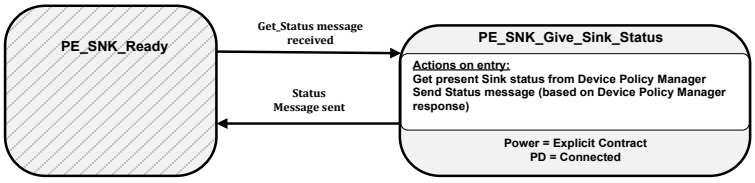
The Policy Engine ***Shall*** transition back to the ***PE_SRC_Ready*** state (see Figure 8-75) when:

- A ***Status*** Message is received
- Or ***SenderResponseTimer*** times out.

8.3.3.10.4 Sink Give Sink Status State Diagram

Figure 8-93 shows the state diagram for a Sink on receiving a ***Get_Status*** Message. See also Section 6.5.1.

Figure 8-93 Sink Give Sink Status State Diagram



8.3.3.10.4.1 PE_SNK_Give_Sink_Status State

The Policy Engine ***Shall*** transition to the ***PE_SNK_Give_Sink_Status*** state, from the ***PE_SNK_Ready*** state, when a ***Get_Status*** Message is received.

On entry to the ***PE_SNK_Give_Sink_Status*** state the Policy Engine ***Shall*** request the present extended Source capabilities from the Device Policy Manager and then send a ***Status*** Message based on these capabilities.

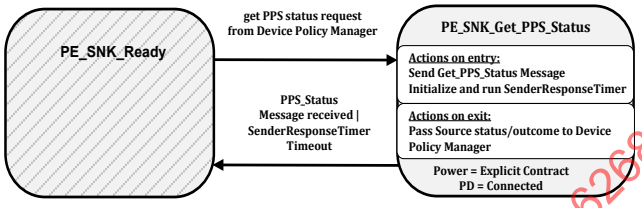
The Policy Engine ***Shall*** transition back to the ***PE_SNK_Ready*** state (see Figure 8-76) when:

- The ***Status*** Message has been successfully sent.

8.3.3.10.5 Sink Port Get Source PPS Status State Diagram

Figure 8-94 shows the state diagram for a Sink on receiving a request from the Device Policy Manager to get the Port Partner's Source status when operating as a PPS. See also Section 6.5.10.

Figure 8-94 Sink Port Get Source PPS Status State Diagram



8.3.3.10.5.1 PE_SNK_Get_PPS_Status State

The Policy Engine ***Shall*** transition to the ***PE_SNK_Get_PPS_Status*** state, from the ***PE_SNK_Ready*** state, due to a request to get the remote source PPS status from the Device Policy Manager.

On entry to the ***PE_SNK_Get_PPS_Status*** state the Policy Engine ***Shall*** send a ***Get_PPS_Status*** Message and initialize and run the ***SenderResponseTimer***.

On exit from the ***PE_SNK_Get_PPS_Status*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the outcome (status or response timeout).

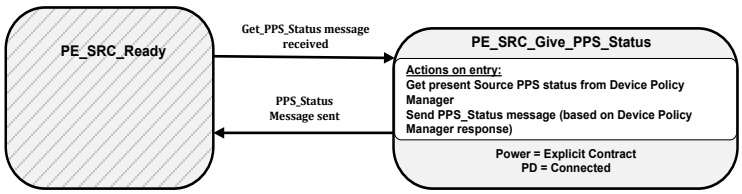
The Policy Engine ***Shall*** transition back to the ***PE_SNK_Ready*** state (see Figure 8-76) when:

- A ***PPS_Status*** Message is received
- Or ***SenderResponseTimer*** times out.

8.3.3.10.6 Source Give Source PPS Status State Diagram

Figure 8-95 shows the state diagram for a Source on receiving a ***Get_PPS_Status*** Message. See also Section 6.5.1.

Figure 8-95 Source Give Source PPS Status State Diagram



8.3.3.10.6.1 PE_SRC_Give_PPS_Status State

The Policy Engine ***Shall*** transition to the ***PE_SRC_Give_PPS_Status*** state, from the ***PE_SRC_Ready*** state, when a ***Get_PPS_Status*** Message is received.

On entry to the ***PE_SRC_Give_PPS_Status*** state the Policy Engine ***Shall*** request the present Source PPS status from the Device Policy Manager and then send a ***PPS_Status*** Message based on these capabilities.

The Policy Engine ***Shall*** transition back to the ***PE_SRC_Ready*** state (see Figure 8-75) when:

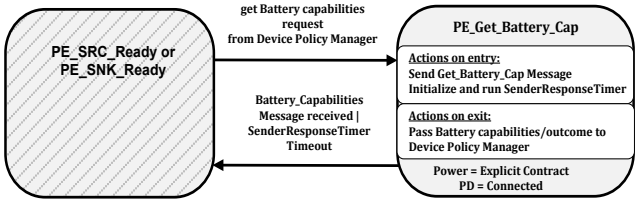
- The ***PPS_Status*** Message has been successfully sent.

8.3.3.11 Battery Capabilities State Diagrams

8.3.3.11.1 Get Battery Capabilities State Diagram

Figure 8-96 shows the state diagram for a Source or Sink on receiving a request from the Device Policy Manager to get the Port Partner's Battery capabilities for a specified Battery. See also Section 6.5.5.

Figure 8-96 Get Battery Capabilities State Diagram



8.3.3.11.1.1 PE_Get_Battery_Cap State

The Policy Engine ***Shall*** transition to the ***PE_Get_Battery_Cap*** state, from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state, due to a request to get the remote Battery capabilities, for a specified Battery, from the Device Policy Manager.

On entry to the ***PE_Get_Battery_Cap*** state the Policy Engine ***Shall*** send a ***Get_Battery_Cap*** Message and initialize and run the ***SenderResponseTimer***.

On exit from the ***PE_Get_Battery_Cap*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the outcome (capabilities or response timeout).

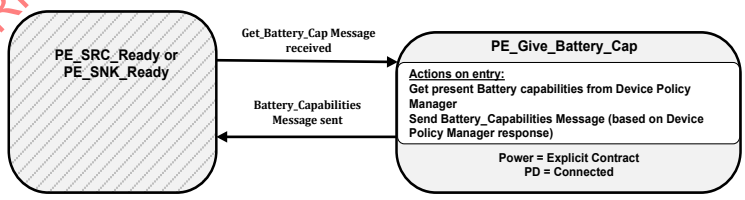
The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state as appropriate (see Figure 8-75 and Figure 8-76) when:

- A ***Battery_Capabilities*** Message is received
- Or ***SenderResponseTimer*** times out.

8.3.3.11.2 Give Battery Capabilities State Diagram

Figure 8-97 shows the state diagram for a Source or Sink on receiving a ***Get_Battery_Cap*** Message. See also Section 6.5.5.

Figure 8-97 Give Battery Capabilities State Diagram



8.3.3.11.2.1 PE_Give_Battery_Cap State

The Policy Engine ***Shall*** transition to the ***PE_Give_Battery_Cap*** state, from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state, when a ***Get_Battery_Cap*** Message is received.

On entry to the ***PE_Give_Battery_Cap*** state the Policy Engine ***Shall*** request the present Battery capabilities, for the requested Battery, from the Device Policy Manager and then send a ***Source_Capabilities_Extended*** Message based on these capabilities.

The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state as appropriate (see Figure 8-75 and Figure 8-76) when:

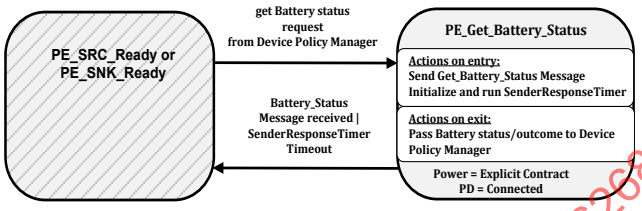
- The ***Battery_Capabilities*** Message has been successfully sent.

8.3.3.12 Battery Status State Diagrams

8.3.3.12.1 Get Battery Status State Diagram

Figure 8-98 shows the state diagram for a Source or Sink on receiving a request from the Device Policy Manager to get the Port Partner's Battery status for a specified Battery. See also Section 6.5.4.

Figure 8-98 Get Battery Status State Diagram



8.3.3.12.1.1 PE_Get_Battery_Status State

The Policy Engine ***Shall*** transition to the ***PE_Get_Battery_Status*** state, from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state, due to a request to get the remote Battery status, for a specified Battery, from the Device Policy Manager.

On entry to the ***PE_Get_Battery_Status*** state the Policy Engine ***Shall*** send a ***Get_Battery_Status*** Message and initialize and run the ***SenderResponseTimer***.

On exit from the ***PE_Get_Battery_Status*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the outcome (status or response timeout).

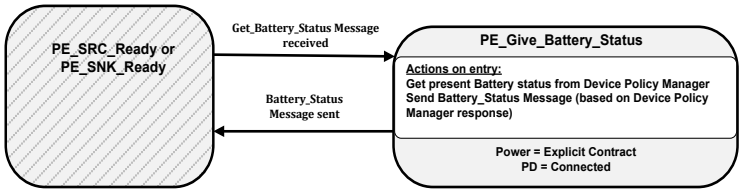
The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state as appropriate (see Figure 8-75 and Figure 8-76) when:

- A ***Battery_Status*** Message is received
- Or ***SenderResponseTimer*** times out.

8.3.3.12.2 Give Battery Status State Diagram

Figure 8-99 shows the state diagram for a Source or Sink on receiving a ***Get_Battery_Status*** Message. See also Section 6.5.4.

Figure 8-99 Give Battery Status State Diagram



8.3.3.12.2.1 PE_Give_Battery_Status State

The Policy Engine ***Shall*** transition to the ***PE_Give_Battery_Status*** state, from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state, when a ***Get_Battery_Status*** Message is received.

On entry to the ***PE_Give_Battery_Status*** state the Policy Engine ***Shall*** request the present Battery status, for the requested Battery, from the Device Policy Manager and then send a ***Battery_Status*** Message based on this status.

The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state as appropriate (see Figure 8-75 and Figure 8-76) when:

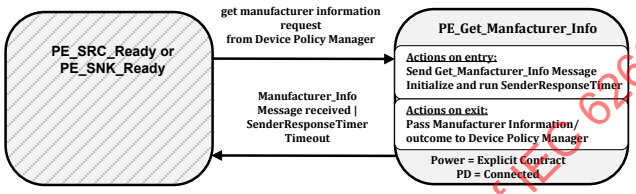
- The ***Battery_Status*** Message has been successfully sent.

8.3.3.13 Manufacturer Information State Diagrams

8.3.3.13.1 Get Manufacturer Information State Diagram

Figure 8-100 shows the state diagram for a Source or Sink on receiving a request from the Device Policy Manager to get the Port Partner's Manufacturer Information. See also Section 6.5.6.

Figure 8-100 Get Manufacturer Information State Diagram



8.3.3.13.1.1 PE_Get_Manufacturer_Info State

The Policy Engine ***Shall*** transition to the ***PE_Get_Manufacturer_Info*** state, from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state, due to a request to get the remote Manufacturer Information from the Device Policy Manager.

On entry to the ***PE_Get_Manufacturer_Info*** state the Policy Engine ***Shall*** send a ***Get_Manufacturer_Info*** Message and initialize and run the ***SenderResponseTimer***.

On exit from the ***PE_Get_Manufacturer_Info*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the outcome (status or response timeout).

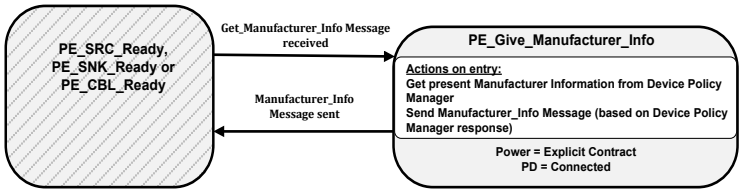
The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state as appropriate (see Figure 8-75 and Figure 8-76) when:

- A ***Manufacturer_Info*** Message is received
- Or ***SenderResponseTimer*** times out.

8.3.3.13.2 Give Manufacturer Information State Diagram

Figure 8-101 shows the state diagram for a Source, Sink or Cable Plug on receiving a ***Get_Manufacturer_Info*** Message. See also Section 6.5.6.

Figure 8-101 Give Manufacturer Information State Diagram



8.3.3.13.2.1 PE_Give_Manufacturer_Info State

The Policy Engine ***Shall*** transition to the ***PE_Give_Manufacturer_Info*** state, from either the ***PE_SRC_Ready, PE_SNK_Ready*** or ***PE_CBL_Ready*** state, when a ***Get_Manufacturer_Info*** Message is received.

On entry to the ***PE_Give_Manufacturer_Info*** state the Policy Engine ***Shall*** request the manufacturer information from the Device Policy Manager and then send a ***Manufacturer_Info*** Message based on this status.

The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready, PE_SNK_Ready*** or ***PE_CBL_Ready*** state as appropriate (see Figure 8-75, Figure 8-76and Figure 8-139) when:

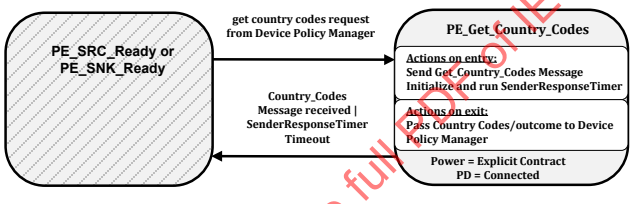
- The ***Manufacturer_Info*** Message has been successfully sent.

8.3.3.14 Country Codes and Information State Diagrams

8.3.3.14.1 Get Country Codes State Diagram

Figure 8-102 shows the state diagram for a Source or Sink on receiving a request from the Device Policy Manager to get the Port Partner's Country Codes. See also Section 6.5.11.

Figure 8-102 Get Country Codes State Diagram



8.3.3.14.1.1 PE_Get_Country_Codes State

The Policy Engine ***Shall*** transition to the ***PE_Get_Country_Codes*** state, from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state, due to a request to get the remote Country Codes from the Device Policy Manager.

On entry to the ***PE_Get_Country_Codes*** state the Policy Engine ***Shall*** send a *Get_Country_Codes* Message and initialize and run the ***SenderResponseTimer***.

On exit from the ***PE_Get_Country_Codes*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the outcome (status or response timeout).

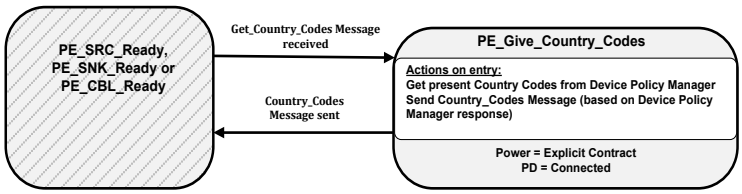
The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state as appropriate (see Figure 8-75 and Figure 8-76) when:

- A ***Country_Codes*** Message is received
- Or ***SenderResponseTimer*** times out.

8.3.3.14.2 Give Country Codes State Diagram

Figure 8-103 shows the state diagram for a Source, Sink or Cable Plug on receiving a *Get_Country_Codes* Message. See also Section 6.5.11.

Figure 8-103 Give Country Codes State Diagram



8.3.3.14.2.1 PE_Give_Country_Codes State

The Policy Engine ***Shall*** transition to the ***PE_Give_Country_Codes*** state, from either the ***PE_SRC_Ready, PE_SNK_Ready*** or ***PE_CBL_Ready*** state, when a *Get_Country_Codes* Message is received.

On entry to the ***PE_Give_Country_Codes*** state the Policy Engine ***Shall*** request the country codes from the Device Policy Manager and then send a ***Country_Codes*** Message containing these codes.

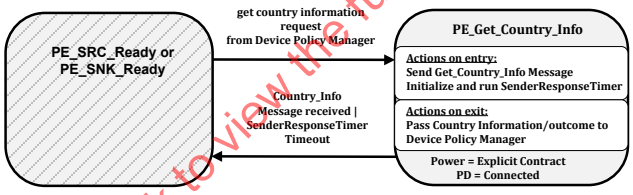
The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready, PE_SNK_Ready*** or ***PE_CBL_Ready*** state as appropriate (see Figure 8-75, Figure 8-76and Figure 8-139) when:

- The ***Country_Codes*** Message has been successfully sent.

8.3.3.14.3 Get Country Information State Diagram

Figure 8-104 shows the state diagram for a Source or Sink on receiving a request from the Device Policy Manager to get the Port Partner's Country Information. See also Section 6.5.12.

Figure 8-104 Get Country Information State Diagram



8.3.3.14.3.1 PE_Get_Country_Info State

The Policy Engine ***Shall*** transition to the ***PE_Get_Country_Info*** state, from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state, due to a request to get the remote Manufacturer Information from the Device Policy Manager.

On entry to the ***PE_Get_Country_Info*** state the Policy Engine ***Shall*** send a ***Get_Manufacturer_Info*** Message and initialize and run the ***SenderResponseTimer***.

On exit from the ***PE_Get_Country_Info*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the outcome (country information or response timeout).

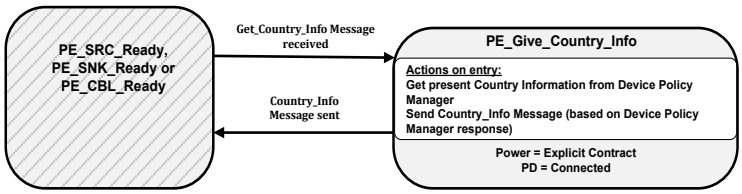
The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state as appropriate (see Figure 8-75 and Figure 8-76) when:

- A ***Country_Info*** Message is received
- Or ***SenderResponseTimer*** times out.

8.3.3.14.4 Give Country Information State Diagram

Figure 8-101 shows the state diagram for a Source, Sink or Cable Plug on receiving a ***Get_Country_Info*** Message. See also Section 6.5.12.

Figure 8-105 Give Country Information State Diagram



8.3.3.14.4.1 PE_Give_Country_Info State

The Policy Engine ***Shall*** transition to the ***PE_Give_Country_Info*** state, from either the ***PE_SRC_Ready, PE_SNK_Ready*** or ***PE_CBL_Ready*** state, when a ***Get_Country_Info*** Message is received.

On entry to the ***PE_Give_Country_Info*** state the Policy Engine ***Shall*** request the country information from the Device Policy Manager and then send a ***Country_Info*** Message containing this country information.

The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready, PE_SNK_Ready*** or ***PE_CBL_Ready*** state as appropriate (see Figure 8-75, Figure 8-76and Figure 8-139) when:

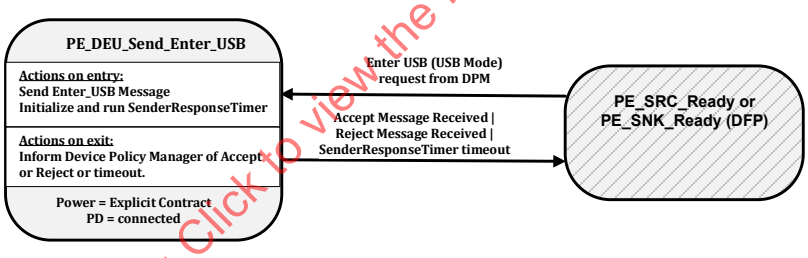
- The ***Country_Info*** Message has been successfully sent.

8.3.3.15 Enter_USB Message State Diagrams

8.3.3.15.1 DFP Enter_USB Message State Diagrams

Figure 8-106 shows the state diagram for an ***Enter_USB*** Message sent by a DFP.

Figure 8-106 DFP Enter_USB Message State Diagram



8.3.3.15.1.1 PE_DEU_Send_Enter_USB State

The ***PE_DEU_Send_Enter_USB*** State ***Shall*** be entered from the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** State when requested by the Device Policy Manager and the Port is operating as a DFP.

On entry to the ***PE_DEU_Send_Enter_USB*** State the Policy Engine ***Shall*** request the Protocol Layer to send an ***Enter_USB*** Message and then initialize and run the ***SenderResponseTimer***.

On exit from the ***PE_DEU_Send_Enter_USB*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the outcome: ***Accept*** Message received, ***Reject*** Message received, ***SenderResponseTimer*** timeout.

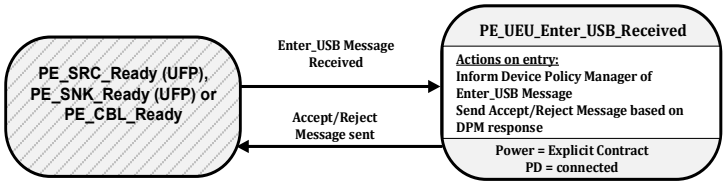
The Policy Engine ***Shall*** transition back to the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** State depending on the Ports power role when:

- An ***Accept*** Message has been received or
- A ***Reject*** Message has been received or
- There is a ***SenderResponseTimer*** timeout.

8.3.3.15.2 UFP or Cable Plug Enter_USB Message State Diagrams

Figure 8-107 shows the state diagram for an ***Enter_USB*** Message received by a UFP or Cable Plug.

Figure 8-107 UFP Enter_USB Message State Diagram



8.3.3.15.2.1 PE_UEU_Enter_USB_Received State

The ***PE_UEU_Enter_USB_Received*** state ***Shall*** be entered from the ***PE_SRC_Ready, PE_SNK_Ready*** or ***PE_CBL_Ready*** state as appropriate (see Figure 8-75, Figure 8-76and Figure 8-139) when an ***Enter_USB*** Message is received and the Port is operating as a UFP or is a Cable Plug.

On entry to the ***PE_UEU_Enter_USB_Received*** state the Policy Engine ***Shall*** inform the Device Policy Manager. The Device Policy Manager responds with an indication of whether the ***Enter_USB*** Message is to be accepted or rejected. The Policy Engine ***Shall*** send either an ***Accept*** Message or a ***Reject*** Message as appropriate.

The Policy Engine ***Shall*** transition back to the ***PE_SRC_Ready, PE_SNK_Ready*** or ***PE_CBL_Ready*** state as appropriate when:

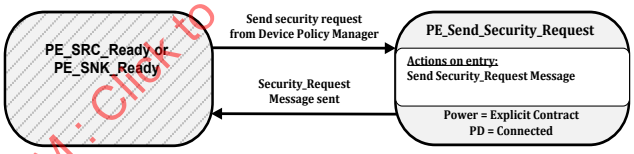
- Either an ***Accept*** Message or a ***Reject*** Message has been sent.

8.3.3.16 Security State Diagrams

8.3.3.16.1 Send Security Request State Diagram

Figure 8-108 shows the state diagram for a Source or Sink on receiving a request from the Device Policy Manager to send a security request. See also Section 6.5.8.

Figure 8-108 Send security request State Diagram



8.3.3.16.1.1 PE_Send_Security_Request State

The Policy Engine ***Shall*** transition to the ***PE_Send_Security_Request*** state, from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state, due to a request to send a security request from the Device Policy Manager.

On entry to the ***PE_Send_Security_Request*** state the Policy Engine ***Shall*** send a ***Security_Request*** Message.

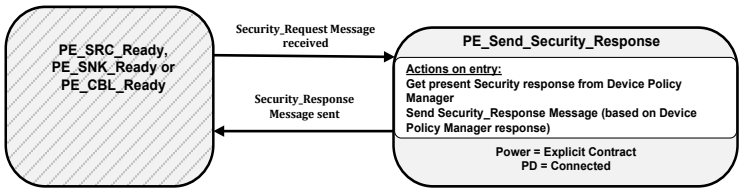
The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state as appropriate (see Figure 8-75 and Figure 8-76) when:

- The ***Security_Request*** Message has been sent.

8.3.3.16.2 Send Security Response State Diagram

Figure 8-109 shows the state diagram for a Source, Sink or Cable Plug on receiving a ***Security_Request*** Message. See also Section 6.5.8.

Figure 8-109 Send security response State Diagram



8.3.3.16.2.1 PE_Send_Security_Response State

The Policy Engine ***Shall*** transition to the ***PE_Send_Security_Response*** state, from either the ***PE_SRC_Ready, PE_SNK_Ready*** or ***PE_CBL_Ready*** state, when a ***Security_Request*** Message is received.

On entry to the ***PE_Send_Security_Response*** state the Policy Engine ***Shall*** request the appropriate response from the Device Policy Manager and then send a ***Security_Response*** Message based on this status.

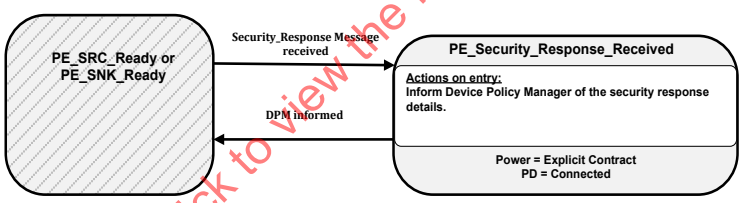
The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready, PE_SNK_Ready*** or ***PE_CBL_Ready*** state as appropriate (see Figure 8-75, Figure 8-76and Figure 8-139) when:

- The ***Security_Response*** Message has been successfully sent.

8.3.3.16.3 Security Response Received State Diagram

Figure 8-110 shows the state diagram for a Source or Sink on receiving a ***Security_Response*** Message. See also Section 6.5.8.

Figure 8-110 Security response received State Diagram



8.3.3.16.3.1 PE_Security_Response_Received State

The Policy Engine ***Shall*** transition to the ***PE_Security_Response_Received*** state, from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** when a ***Security_Response*** Message is received.

On entry to the ***PE_Security_Response_Received*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the details of the security response.

The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state as appropriate (see Figure 8-75, Figure 8-76 and Figure 8-139) when:

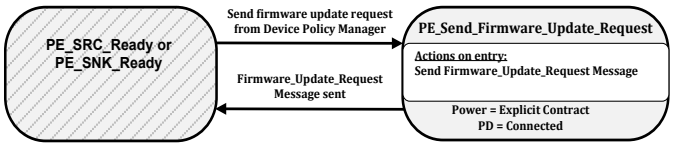
- The Device Policy Manager has been informed.

8.3.3.17 Firmware Update State Diagrams

8.3.3.17.1 Send Firmware Update Request State Diagram

Figure 8-111 shows the state diagram for a Source or Sink on receiving a request from the Device Policy Manager to send a firmware update request. See also Section 6.5.9.

Figure 8-111 Send firmware update request State Diagram



8.3.3.17.1.1 PE_Send_Firmware_Update_Request State

The Policy Engine ***Shall*** transition to the ***PE_Send_Firmware_Update_Request*** state, from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state, due to a request to send a firmware update request from the Device Policy Manager.

On entry to the ***PE_Send_Firmware_Update_Request*** state the Policy Engine ***Shall*** send a ***Firmware_Update_Request*** Message.

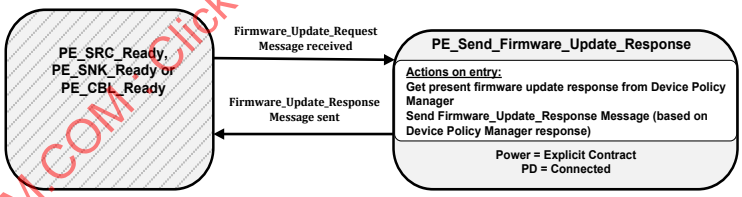
The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state as appropriate (see Figure 8-75 and Figure 8-76) when:

- The ***Firmware_Update_Request*** Message has been sent.

8.3.3.17.2 Send Firmware Update Response State Diagram

Figure 8-112 shows the state diagram for a Source, Sink or Cable Plug on receiving a ***Firmware_Update_Request*** Message. See also Section 6.5.9.

Figure 8-112 Send firmware update response State Diagram



8.3.3.17.2.1 PE_Send_Firmware_Update_Response State

The Policy Engine ***Shall*** transition to the ***PE_Send_Firmware_Update_Response*** state, from either the ***PE_SRC_Ready, PE_SNK_Ready*** or ***PE_CBL_Ready*** state, when a ***Firmware_Update_Request*** Message is received.

On entry to the ***PE_Send_Firmware_Update_Response*** state the Policy Engine ***Shall*** request the appropriate response from the Device Policy Manager and then send a ***Firmware_Update_Response*** Message based on this status.

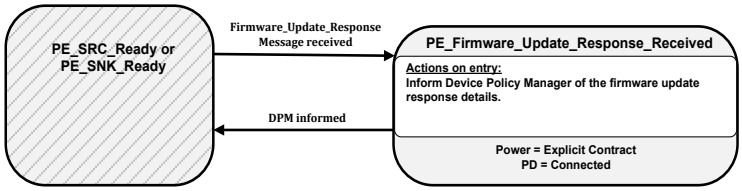
The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready, PE_SNK_Ready*** or ***PE_CBL_Ready*** state as appropriate (see Figure 8-75, Figure 8-76and Figure 8-139) when:

- The ***Firmware_Update_Response*** Message has been successfully sent.

8.3.3.17.3 Firmware Update Response Received State Diagram

Figure 8-113 shows the state diagram for a Source or Sink on receiving a ***Firmware_Update_Response*** Message. See also Section 6.5.9.

Figure 8-113 Firmware update response received State Diagram



8.3.3.17.3.1 PE_Firmware_Update_Response_Received State

The Policy Engine ***Shall*** transition to the ***PE_Firmware_Update_Response_Received*** state, from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** when a ***Firmware_Update_Response*** Message is received.

On entry to the ***PE_Firmware_Update_Response_Received*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the details of the firmware update response.

The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state as appropriate (see Figure 8-75, Figure 8-76 and Figure 8-139) when:

- The Device Policy Manager has been informed.

8.3.3.18 Dual-Role Port State Diagrams

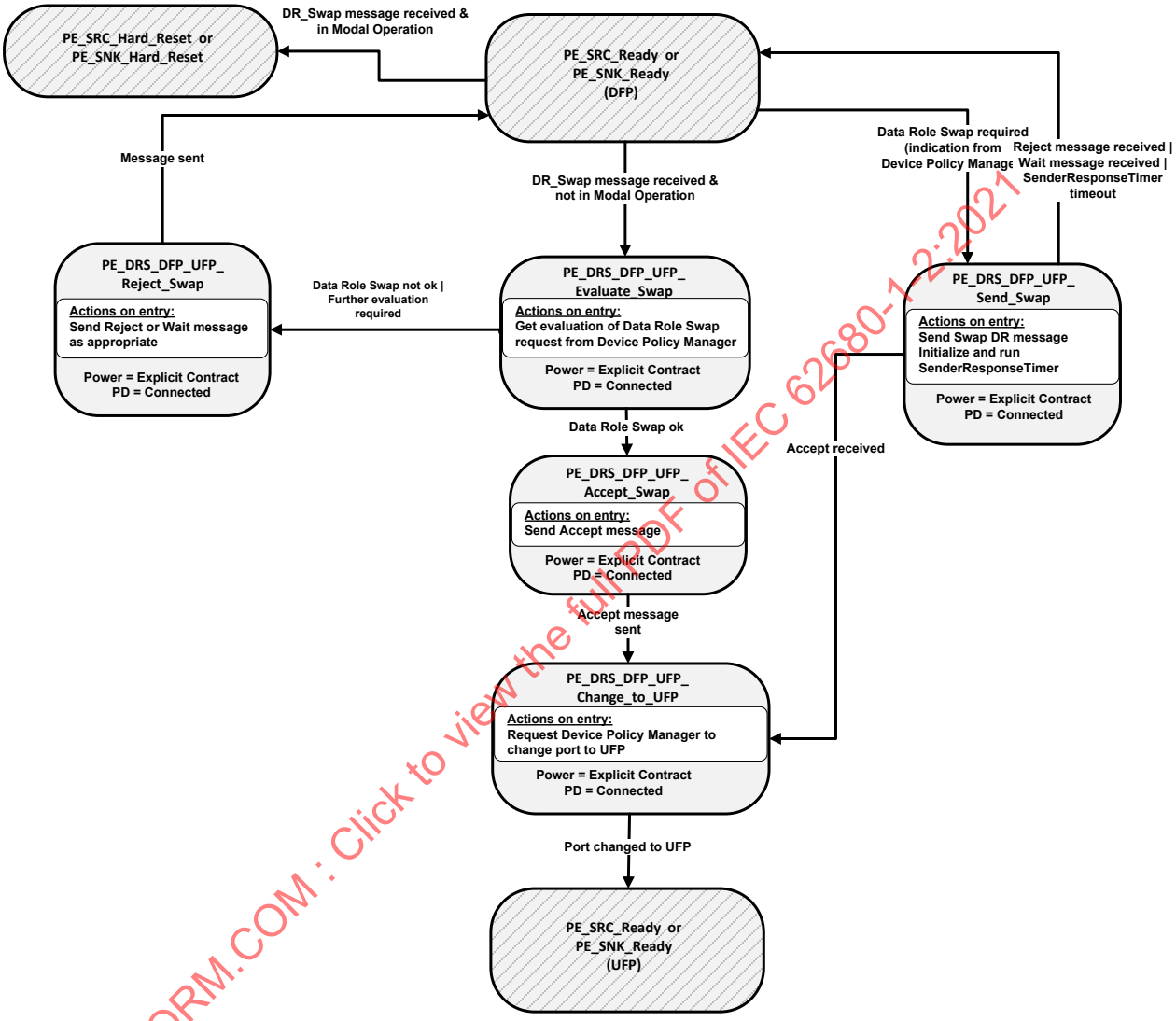
Dual-Role Ports that combine Source and Sink capabilities ***Shall*** comprise Source and Sink Policy Engine state machines. In addition they ***Shall*** have the capability to perform a Power Role Swap from the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** states and ***Shall*** return to USB Default Operation on a Hard Reset.

The State Diagrams in this section ***Shall*** apply to every ***[USB Type-C 2.0]*** DRP.

8.3.3.18.1 DFP to UFP Data Role Swap State Diagram

Figure 8-114 shows the additional state diagram required to perform a Data Role Swap from DFP to UFP operation and the changes that ***Shall*** be followed for error and Hard Reset handling.

Figure 8-114: DFP to UFP Data Role Swap State Diagram



8.3.3.18.1.1 PE_SRC_Ready or PE_SNK_Ready State

The Data Role Swap process ***Shall*** start only from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state where power is stable.

The Policy Engine ***Shall*** transition to the ***PE_DRS_DFP_UFP_Evaluate_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_Swap*** state when:

- The Device Policy Manager indicates that a Data Role Swap is required.

8.3.3.18.1.2 PE_DRS_DFP_UFP_Evaluate_Swap State

On entry to the ***PE_DRS_DFP_UFP_Evaluate_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_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_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.

8.3.3.18.1.3 PE_DRS_DFP_UFP_Accept_Swap State

On entry to the ***PE_DRS_DFP_UFP_Accept_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.

8.3.3.18.1.4 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 Port has been changed to a UFP.

8.3.3.18.1.5 PE_DRS_DFP_UFP_Send_Swap State

On entry to the ***PE_DRS_DFP_UFP_Send_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_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.

8.3.3.18.1.6 PE_DRS_DFP_UFP_Reject_Swap State

On entry to the ***PE_DRS_DFP_UFP_Reject_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).

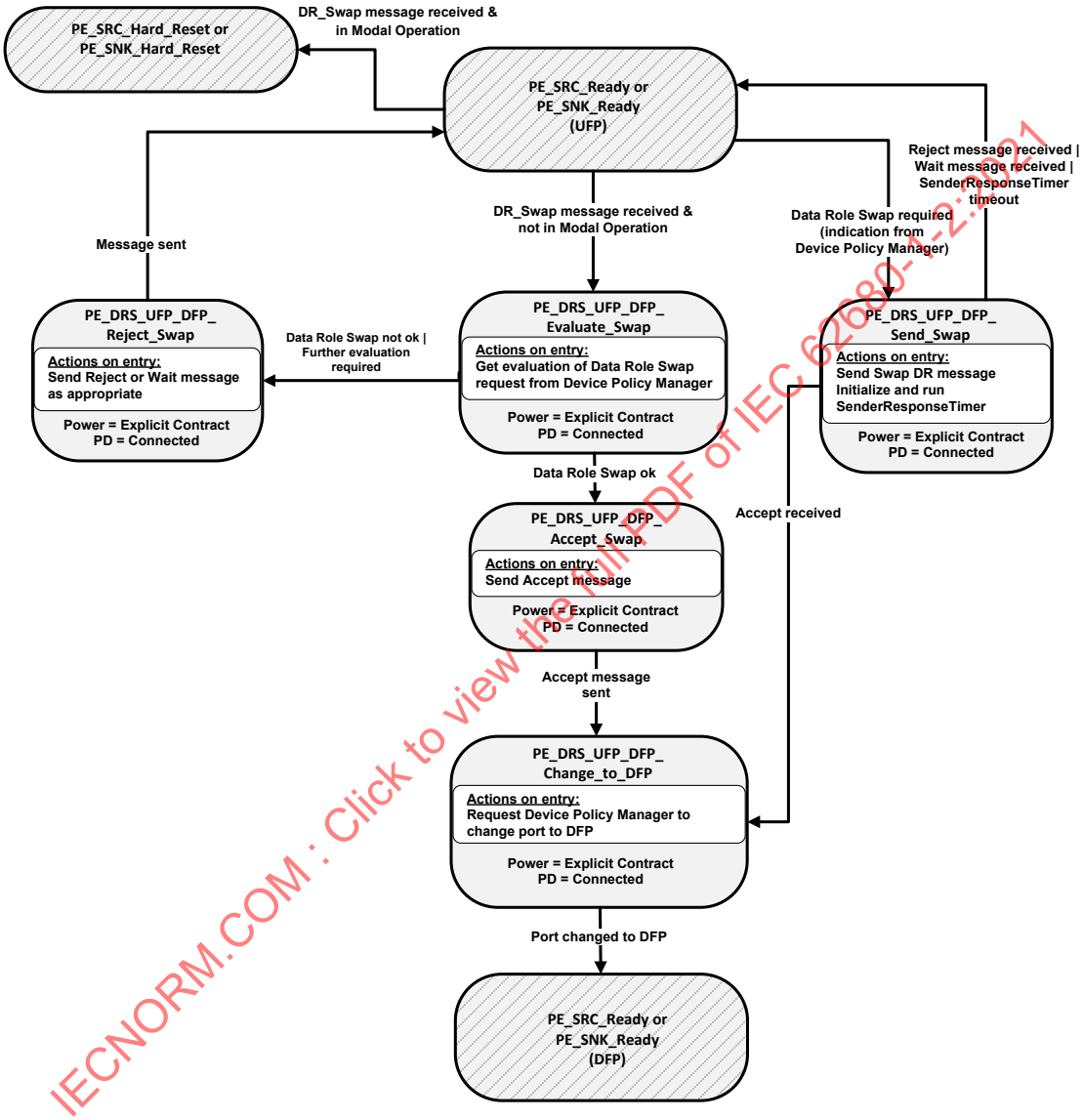
The Policy Engine ***Shall*** continue as a DFP and ***Shall*** transition to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state when:

- The Reject or Wait Message has been sent.

8.3.3.18.2 UFP to DFP Data Role Swap State Diagram

Figure 8-115 shows the additional state diagram required to perform a Data Role Swap from DRP UFP to DFP operation and the changes that ***Shall*** be followed for error and Hard Reset handling.

Figure 8-115: UFP to DFP Data Role Swap State Diagram



8.3.3.18.2.1 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_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_Swap*** state when:

- The Device Policy Manager indicates that a Data Role Swap is required.

8.3.3.18.2.2 PE_DRS_UFP_DFP_Evaluate_Swap State

On entry to the ***PE_DRS_UFP_DFP_Evaluate_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_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_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.

8.3.3.18.2.3 PE_DRS_UFP_DFP_Accept_Swap State

On entry to the ***PE_DRS_UFP_DFP_Accept_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.

8.3.3.18.2.4 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 Port has been changed to a DFP.

8.3.3.18.2.5 PE_DRS_UFP_DFP_Send_Swap State

On entry to the ***PE_DRS_UFP_DFP_Send_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_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.

8.3.3.18.2.6 PE_DRS_UFP_DFP_Reject_Swap State

On entry to the ***PE_DRS_UFP_DFP_Reject_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:

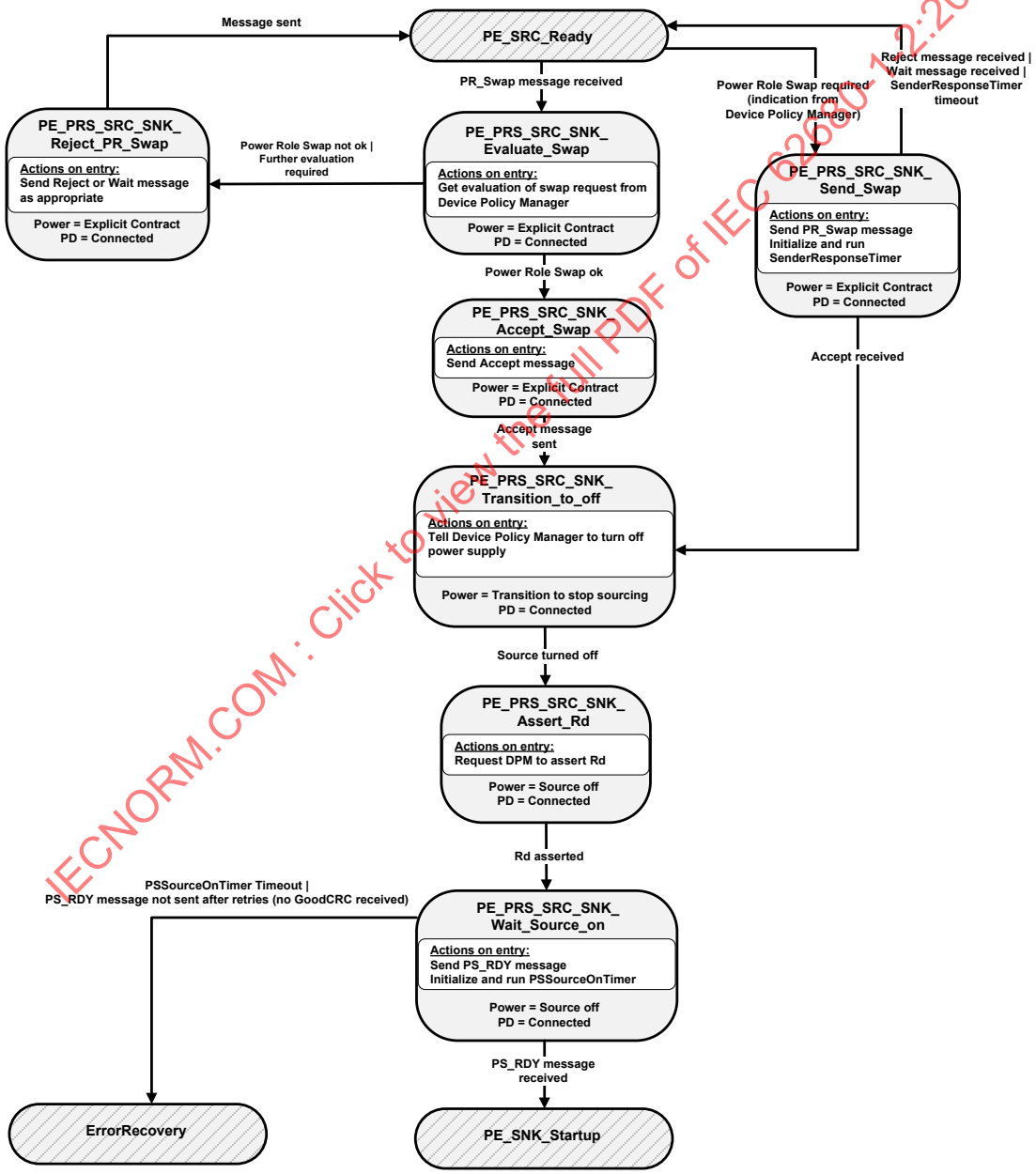
- The ***Reject*** or ***Wait*** Message has been sent.

8.3.3.18.3 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 8-116 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 handling.

Figure 8-116: Dual-Role Port in Source to Sink Power Role Swap State Diagram



8.3.3.18.3.1 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_Swap*** state when:

- A ***PR_Swap*** Message is received.

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

- The Device Policy Manager indicates that a Power Role Swap is required.

8.3.3.18.3.2 PE_PRS_SRC_SNK_Evaluate_Swap State

On entry to the ***PE_PRS_SRC_SNK_Evaluate_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_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_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.

8.3.3.18.3.3 PE_PRS_SRC_SNK_Accept_Swap State

On entry to the ***PE_PRS_SRC_SNK_Accept_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.

8.3.3.18.3.4 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.

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.

8.3.3.18.3.5 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 Rp to Rd.

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

- The Device Policy Manager indicates that Rd is asserted.

8.3.3.18.3.6 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 ***ErrorRecovery*** state when:

- 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.

8.3.3.18.3.7 PE_PRS_SRC_SNK_Send_Swap State

On entry to the ***PE_PRS_SRC_SNK_Send_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_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.

8.3.3.18.3.8 PE_PRS_SRC_SNK_Reject_Swap State

On entry to the ***PE_PRS_SRC_SNK_Reject_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:

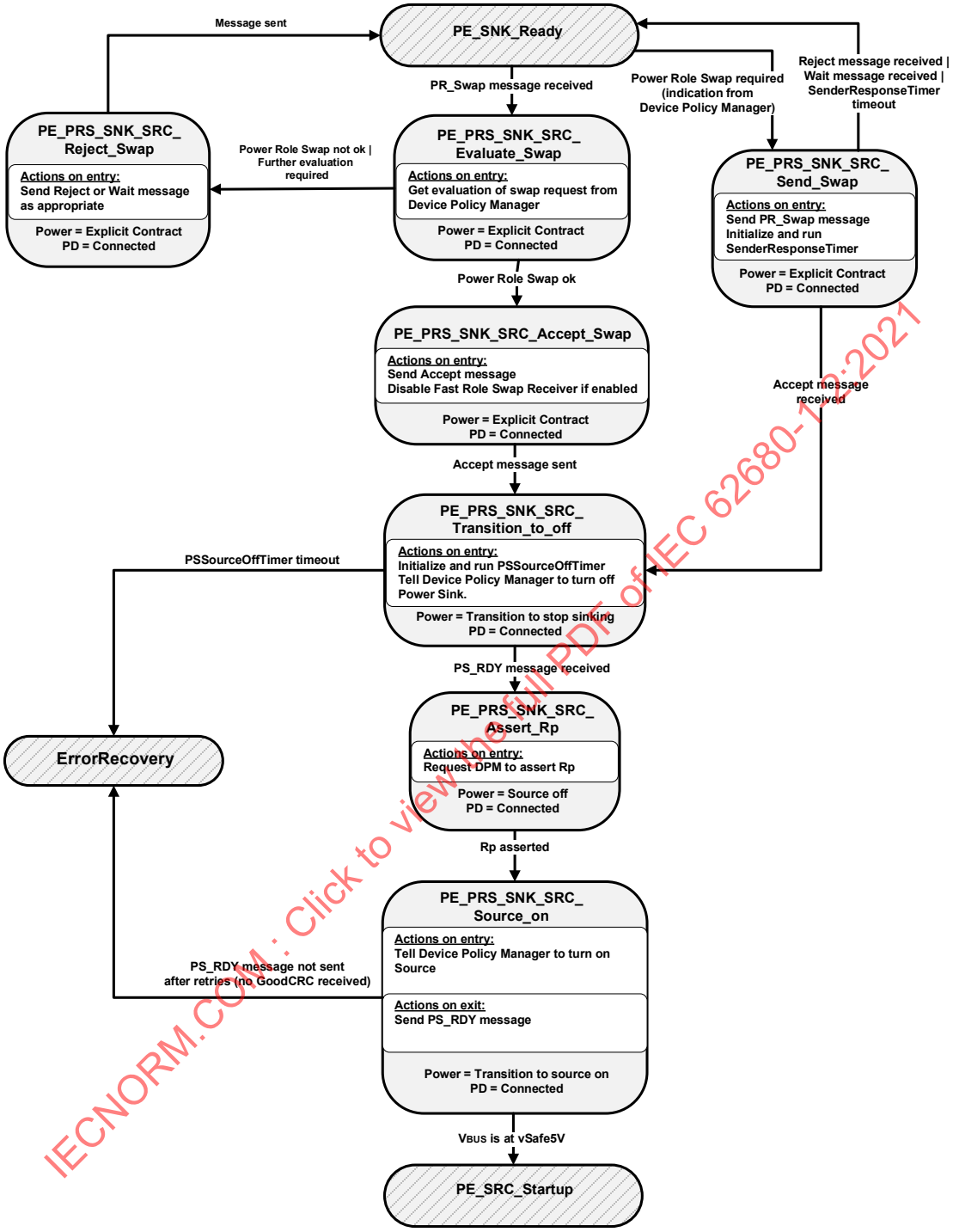
- The ***Reject*** or ***Wait*** Message has been sent.

8.3.3.18.4 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 8-117 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 handling.

Figure 8-117: Dual-role Port in Sink to Source Power Role Swap State Diagram



8.3.3.18.4.1 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_Swap*** state when:

- A ***PR_Swap*** Message is received.

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

- The Device Policy Manager indicates that a Power Role Swap is required.

8.3.3.18.4.2 PE_PRS_SNK_SRC_Evaluate_Swap State

On entry to the ***PE_PRS_SNK_SRC_Send_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_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_Swap*** state when:

- The Device Policy Manager indicates that a Power Role Swap is not ok.

8.3.3.18.4.3 PE_PRS_SNK_SRC_Accept_Swap State

On entry to the ***PE_PRS_SNK_SRC_Accept_Swap*** state the Policy Engine ***Shall*** request the Protocol Layer to send an ***Accept*** Message and ***Shall*** disable the Fast Role Swap receiver if this is enabled.

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

- The ***Accept*** Message has been sent.

8.3.3.18.4.4 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 ***ErrorRecovery*** state when:

- The ***PSSourceOffTimer*** times out.

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

- A ***PS_RDY*** Message is received.

8.3.3.18.4.5 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 Rd to Rp.

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

- The Device Policy Manager indicates that Rd is asserted.

8.3.3.18.4.6 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.

On exit from the ***PE_PRS_SNK_SRC_Source_on*** state the Policy Engine ***Shall*** send a ***PS_RDY*** Message.

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

- The Source Port VBUS is at ***vSafe5V***.

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

- 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.

8.3.3.18.4.7 PE_PRS_SNK_SRC_Send_Swap State

On entry to the ***PE_PRS_SNK_SRC_Send_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.

8.3.3.18.4.8 PE_PRS_SNK_SRC_Reject_Swap State

On entry to the ***PE_PRS_SNK_SRC_Reject_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:

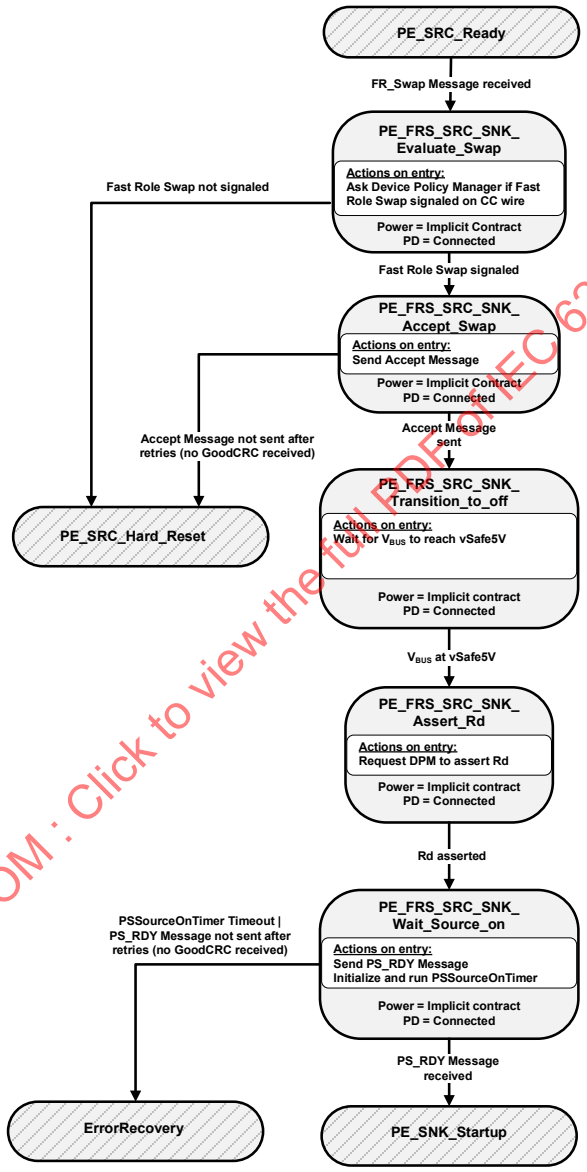
- The ***Reject*** or ***Wait*** Message has been sent.

8.3.3.18.5 Policy Engine in Source to Sink Fast 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 ***Should*** have the capability to do a Fast Role Swap from the ***PE_SRC_Ready*** state and ***Shall*** return to USB Default Operation on a Hard Reset.

Figure 8-118 shows the additional state diagram required to perform a Fast Role Swap from Source to Sink roles and the changes that ***Shall*** be followed for error handling.

Figure 8-118: Dual-Role Port in Source to Sink Fast Role Swap State Diagram



8.3.3.18.5.1 PE_SRC_Ready State

The Fast Role Swap process ***Shall*** start only from the ***PE_SRC_Ready*** state where power is stable.

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

- An ***FR_Swap*** Message is received.

8.3.3.18.5.2 PE_FRS_SRC_SNK_Evaluate_Swap State

On entry to the ***PE_FRS_SRC_SNK_Evaluate_Swap*** state the Policy Engine ***Shall*** ask the Device Policy Manager whether Fast Role Swap has been signaled on the CC wire.

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

- The Device Policy Manager indicates that a Fast Role Swap has been signaled.

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

- The Device Policy Manager indicates that a Fast Role Swap is not being signaled.

8.3.3.18.5.3 PE_FRS_SRC_SNK_Accept_Swap State

On entry to the ***PE_FRS_SRC_SNK_Accept_Swap*** state the Policy Engine ***Shall*** request the Protocol Layer to send an ***Accept*** Message.

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

- The ***Accept*** Message has been sent.

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

- The ***Accept*** Message is not sent after retries (a ***GoodCRC*** Message has not been received). Note: a soft reset ***Shall Not*** be initiated in this case.

8.3.3.18.5.4 PE_FRS_SRC_SNK_Transition_to_off State

On entry to the ***PE_FRS_SNK_SRC_Transition_to_off*** state the Policy Engine ***Shall*** wait until V_{BUS} has discharged to ***vSafe5V***.

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

- The Device Policy Manager indicates that V_{BUS} has discharged to ***vSafe5V***.

8.3.3.18.5.5 PE_FRS_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 Rp to Rd.

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

- The Device Policy Manager indicates that Rd is asserted.

8.3.3.18.5.6 PE_FRS_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 new Source is now applying Rp.

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

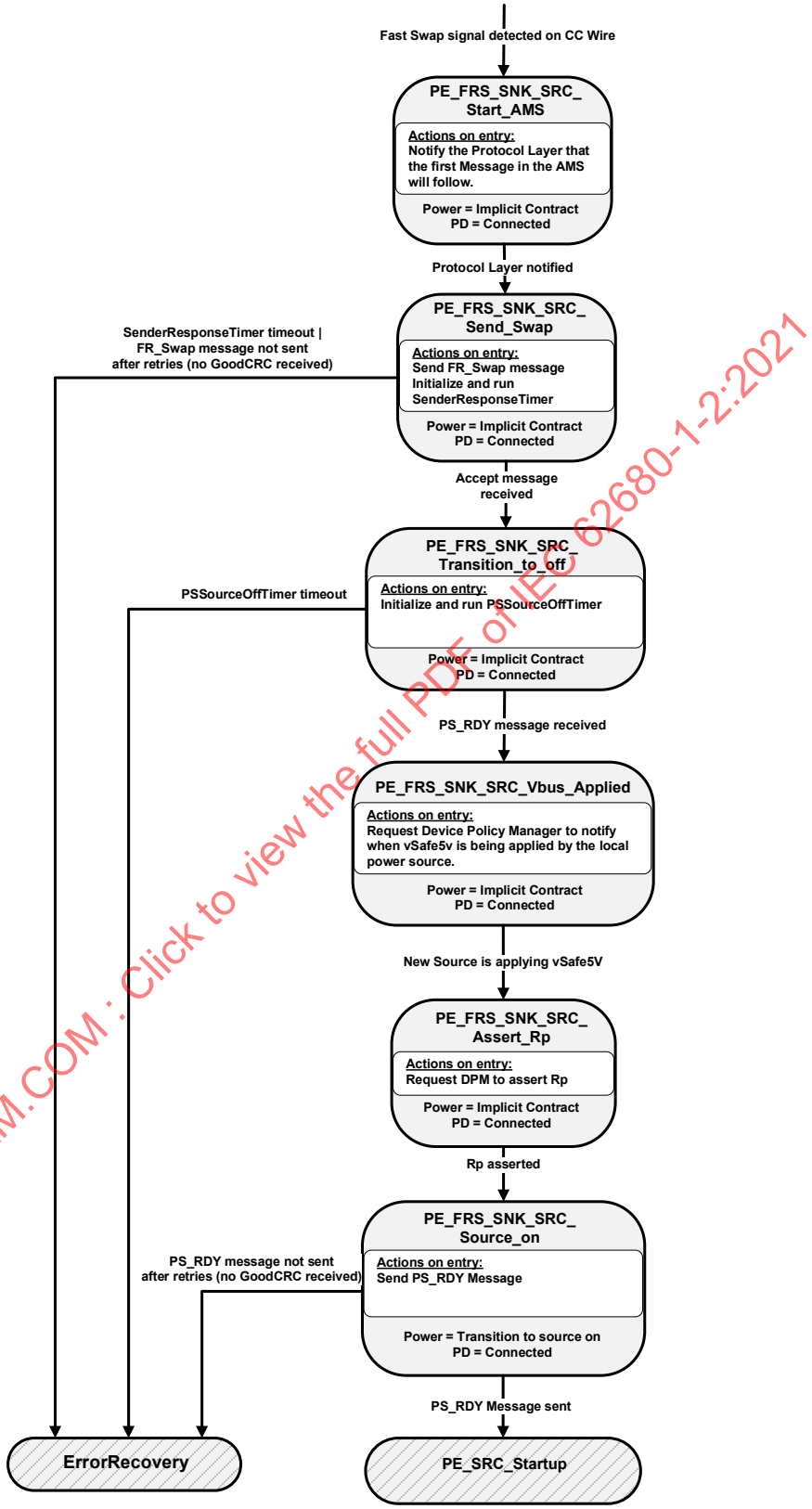
- The ***PSSourceOnTimer*** times out or
- The ***PS_RDY*** Message is not sent after retries (a ***GoodCRC*** Message has not been received). Note: a soft reset ***Shall Not*** be initiated in this case.

8.3.3.18.6 Policy Engine in Sink to Source Fast 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 ***Should*** have the capability to do a Fast Role Swap from the ***PE_SNK_Ready*** state and ***Shall*** return to USB Default Operation on a Hard Reset.

Figure 8-119 shows the additional state diagram required to perform a Fast Role Swap from Sink to Source roles and the changes that ***Shall*** be followed for error handling.

Figure 8-119: Dual-role Port in Sink to Source Fast Role Swap State Diagram



8.3.3.18.6.1 PE_FRS_SNK_SRC_Start_AMS State

The Policy Engine ***Shall*** transition to the ***PE_FRS_SNK_SRC_Send_Swap*** state from any other state provided there is an Explicit Contract in place when:

- The Sink Capabilities received from the initial Source by the Policy Engine has at least one of the Fast Role Swap bits set.
- The system has sufficient reserve power to provide the requested current to the initial Source, as requested in the Fast Role Swap bits in the Sink Capabilities, and is willing to dedicate it to the Port
- The Device Policy Manager indicates that a Fast Role Swap signal has been detected on the CC Wire.

On entry to the ***PE_FRS_SNK_SRC_Start_AMS*** state the Policy Engine ***Shall*** notify the Protocol Layer that the first Message in an AMS will follow.

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

- The Protocol Layer has been notified.

8.3.3.18.6.2 PE_FRS_SNK_SRC_Send_Swap State

On entry to the ***PE_FRS_SNK_SRC_Send_Swap*** state the Policy Engine ***Shall*** request the Protocol Layer to send an ***FR_Swap*** Message and ***Shall*** initialize and run the ***SenderResponseTimer***.

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

- An ***Accept*** Message is received.

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

- The ***SenderResponseTimer*** times out or
- The ***FR_Swap*** Message is not sent after retries (a ***GoodCRC*** Message has not been received). A soft reset ***Shall Not*** be initiated in this case.

8.3.3.18.6.3 PE_FRS_SNK_SRC_Transition_to_off State

On entry to the ***PE_FRS_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 ***ErrorRecovery*** state when:

- The ***PSSourceOffTimer*** times out.

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

- A ***PS_RDY*** Message is received.

8.3.3.18.6.4 PE_FRS_SNK_SRC_Vbus_Applied State

On entry to the ***PE_FRS_SNK_SRC_Vbus_Applied*** state the Policy Engine waits for a notification from the Device Policy Manager that the local power source has applied ***vSafe5V*** to V_{BUS} (see Section 5.8.6.3). Note this could have already been applied prior to entering this state or could be applied while waiting in this state.

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

- The Device Policy Manager indicates that ***vSafe5V*** is being applied.

8.3.3.18.6.5 PE_FRS_SNK_SRC_Assert_Rp State

On entry to the ***PE_FRS_SNK_SRC_Assert_Rp*** state the Policy Engine ***Shall*** request the Device Policy Manager to change the resistor asserted on the CC wire from Rd to Rp.

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

- The Device Policy Manager indicates that Rp is asserted.

8.3.3.18.6.6 PE_FRS_SNK_SRC_Source_on State

On entry to the ***PE_FRS_SNK_SRC_Source_on*** state the Policy Engine ***Shall*** request the Device Policy Manager to turn on the Source.

On exit from the ***PE_FRS_SNK_SRC_Source_on*** state (except if the exit is to send a ***Ping*** Message) the Policy Engine ***Shall*** send a ***PS_RDY*** Message.

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

- The ***PS_RDY*** Message has been sent.

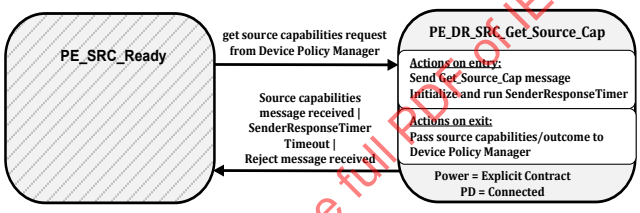
The Policy Engine ***Shall*** transition to the ***ErrorRecovery*** state when:

- 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.

8.3.3.18.7 Dual-Role (Source Port) Get Source Capabilities State Diagram

Figure 8-120 shows the state diagram for a Dual-Role device, presently operating as a Source, on receiving a request from the Device Policy Manager to get the Port Partner's Source capabilities. See also Section 6.4.1.1.3.

Figure 8-120 Dual-Role (Source) Get Source Capabilities diagram



8.3.3.18.7.1 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).

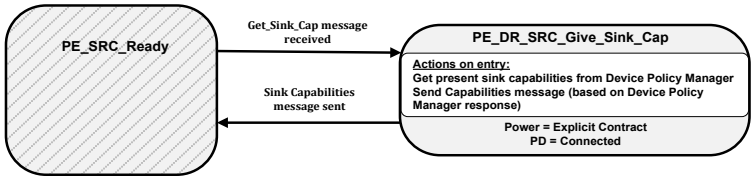
The Policy Engine ***Shall*** transition back to the ***PE_SRC_Ready*** state (see Figure 8-75) when:

- A ***Source_Capabilities*** Message is received
- Or ***SenderResponseTimer*** times out
- Or a ***Reject*** Message is received.

8.3.3.18.8 Dual-Role (Source Port) Give Sink Capabilities State Diagram

Figure 8-121 shows the state diagram for a Dual-Role device, presently operating as a Source, on receiving a ***Get_Sink_Cap*** Message. See also Section 6.4.1.1.3.

Figure 8-121 Dual-Role (Source) Give Sink Capabilities diagram



8.3.3.18.8.1 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.

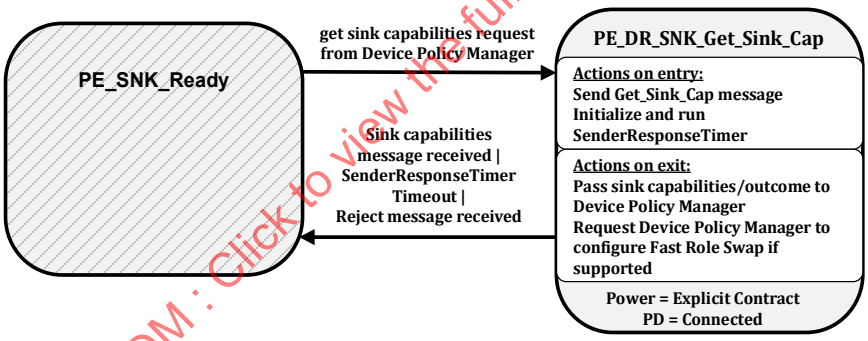
The Policy Engine ***Shall*** transition back to the ***PE_SRC_Ready*** state (see Figure 8-75) when:

- The ***Sink_Capabilities*** Message has been successfully sent.

8.3.3.18.9 Dual-Role (Sink Port) Get Sink Capabilities State Diagram

Figure 8-122 shows the state diagram for a Dual-Role device, presently operating as a Sink, on receiving a request from the Device Policy Manager to get the Port Partner's Sink capabilities. See also Section 6.4.1.1.3.

Figure 8-122 Dual-Role (Sink) Get Sink Capabilities State Diagram



8.3.3.18.9.1 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). If Fast Role Swap is supported, request Device Policy Manager prepare or disable 5V source and configure the Fast Role Swap receiver based on the Fast Role Swap bits in the received Sink Capabilities.

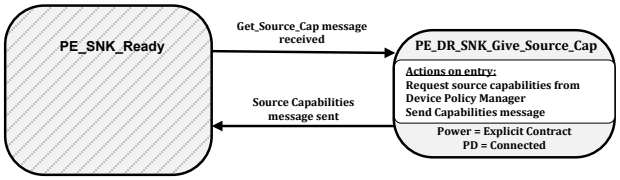
The Policy Engine ***Shall*** transition back to the ***PE_SNK_Ready*** state (see Figure 8-76) when:

- A ***Sink_Capabilities*** Message is received
- Or ***SenderResponseTimer*** times out
- Or a ***Reject*** Message is received.

8.3.3.18.10 Dual-Role (Sink Port) Give Source Capabilities State Diagram

Figure 8-123 shows the state diagram for a Dual-Role device, presently operating as a Sink, on receiving a ***Get_Source_Cap*** Message. See also Section 6.4.1.1.3.

Figure 8-123 Dual-Role (Sink) Give Source Capabilities State Diagram



8.3.3.18.10.1 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.

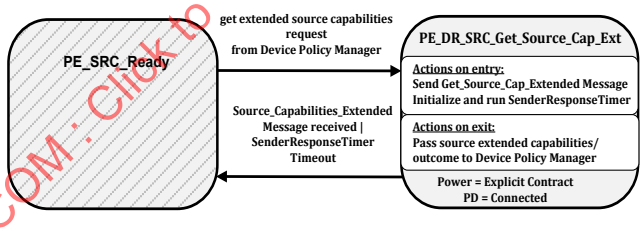
The Policy Engine ***Shall*** transition back to the ***PE_SNK_Ready*** state (see Figure 8-76) when:

- The ***Source_Capabilities*** Message has been successfully sent.

8.3.3.18.11 Dual-Role (Source Port) Get Source Capabilities Extended State Diagram

Figure 8-124 shows the state diagram for a Dual-Role device, presently operating as a Source, on receiving a request from the Device Policy Manager to get the Port Partner's extended Source capabilities. See also Section 6.5.1.

Figure 8-124 Dual-Role (Source) Get Source Capabilities Extended State Diagram



8.3.3.18.11.1 PE_DR_SRC_Get_Source_Cap_Ext State

The Policy Engine ***Shall*** transition to the ***PE_DR_SRC_Get_Source_Cap_Ext*** state, from the ***PE_SRC_Ready*** state, due to a request to get the remote extended source capabilities from the Device Policy Manager.

On entry to the ***PE_DR_SRC_Get_Source_Cap_Ext*** state the Policy Engine ***Shall*** send a ***Get_Source_Cap_Extended*** Message and initialize and run the ***SenderResponseTimer***.

On exit from the ***PE_DR_SRC_Get_Source_Cap_Ext*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the outcome (capabilities or response timeout).

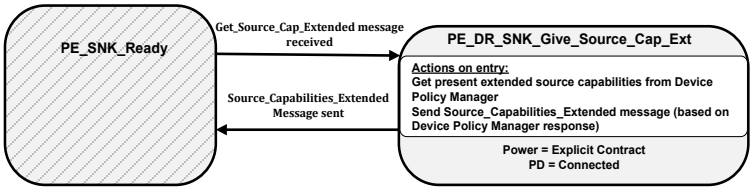
The Policy Engine ***Shall*** transition back to the ***PE_SRC_Ready*** state (see Figure 8-75) when:

- A ***Source_Capabilities_Extended*** Message is received
- Or ***SenderResponseTimer*** times out.

8.3.3.18.12 Dual-Role (Sink Port) Give Source Capabilities Extended State Diagram

Figure 8-125 shows the state diagram for a Dual-Role device, presently operating as a Sink, on receiving a ***Get_Source_Cap_Extended*** Message. See also Section 6.5.1.

Figure 8-125 Dual-Role (Source) Give Sink Capabilities diagram



8.3.3.18.12.1 PE_DR_SNK_Give_Source_Cap_Ext State

The Policy Engine ***Shall*** transition to the ***PE_DR_SNK_Give_Source_Cap_Ext*** state, from the ***PE_SNK_Ready*** state, when a ***Get_Source_Cap_Extended*** Message is received.

On entry to the ***PE_DR_SNK_Give_Source_Cap_Ext*** state the Policy Engine ***Shall*** request the present extended Source capabilities from the Device Policy Manager and then send a ***Source_Capabilities_Extended*** Message based on these capabilities.

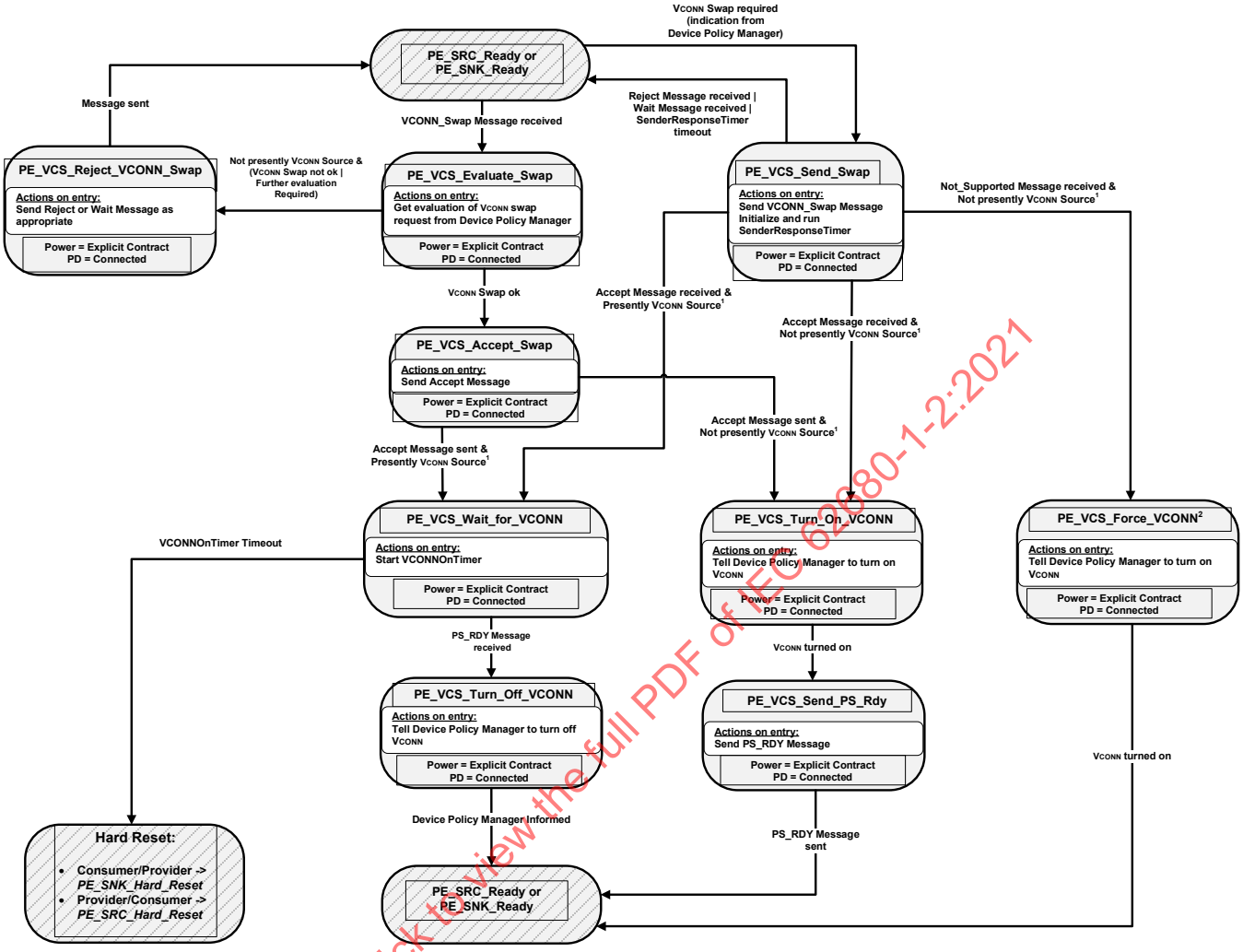
The Policy Engine ***Shall*** transition back to the ***PE_SNK_Ready*** state (see Figure 8-76) when:

- The ***Source_Capabilities_Extended*** Message has been successfully sent.

8.3.3.19 VCONN Swap State Diagram

The State Diagram in this section ***Shall*** apply to Ports that supply VCONN. Figure 8-126 shows the state operation for a Port on sending or receiving a VCONN Swap request.

Figure 8-126 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.

[2] The *PE_VCS_Force_VCONN* state is ***Optional***.

8.3.3.19.1 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
- The Port is presently the VCONN Source.

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

- An *Accept* Message is received and
- The Port is not presently the VCONN Source.

The Policy Engine ***Shall*** transition back 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 ***May*** transition to the ***PE_VCS_Force_VCONN*** state when:

- A ***Not_Supported*** Message is received and
- The Port is not presently the VCONN Source.

8.3.3.19.2 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. Note: Ports that are presently the VCONN Source must always accept a VCONN swap request (see Section 6.3.11).

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 Port is not presently the VCONN Source and
- 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.

8.3.3.19.3 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 Port'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 Port's VCONN is off.

8.3.3.19.4 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 Port 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.

8.3.3.19.5 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.

8.3.3.19.6 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 when:

- The Device Policy Manager has been informed.

8.3.3.19.7 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 Port's VCONN is on.

8.3.3.19.8 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 when:

- The ***PS_RDY*** Message has been sent.

8.3.3.19.9 PE_VCS_Force_VCONN State

On entry to the ***PE_VCS_Force_VCONN*** state the Policy Engine ***Shall*** tell the Device Policy Manager to turn on VCONN.

The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state when:

- The Port's VCONN is on.

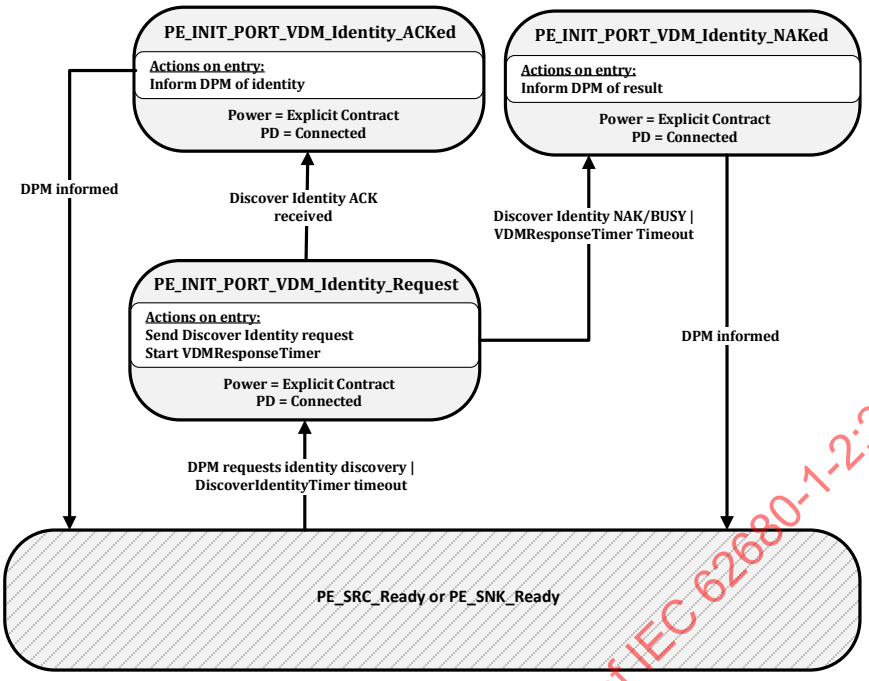
8.3.3.20 Initiator Structured VDM State Diagrams

The State Diagrams in this section ***Shall*** apply to all Initiators.

8.3.3.20.1 Initiator Structured VDM Discover Identity State Diagram

Figure 8-127 shows the state diagram for an Initiator when discovering the identity of its Port Partner or Cable Plug.

Figure 8-127 Initiator to Port VDM Discover Identity State Diagram



8.3.3.20.1.1 PE_INIT_PORT_VDM_Identity_Request State

The Policy Engine transitions to the ***PE_INIT_PORT_VDM_Identity_Request*** state from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state when:

- The Device Policy Manager requests the discovery of the identity of the Port Partner or
- The ***DiscoverIdentityTimer*** times out.

On entry to the ***PE_INIT_PORT_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_INIT_PORT_VDM_Identity_ACKed*** state when:

- A Structured VDM ***Discover Identity*** ACK Command response is received.

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

- A Structured VDM ***Discover Identity*** NAK or BUSY Command response is received or
- The ***VDMResponseTimer*** times out.

8.3.3.20.1.2 PE_INIT_PORT_VDM_Identity_ACKed State

On entry to the ***PE_INIT_PORT_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 when:

- The Device Policy Manager has been informed.

8.3.3.20.1.3 PE_INIT_PORT_VDM_Identity_NAKed State

On entry to the ***PE_INIT_PORT_VDM_Identity_NAKed*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the result (NAK, BUSY or timeout).

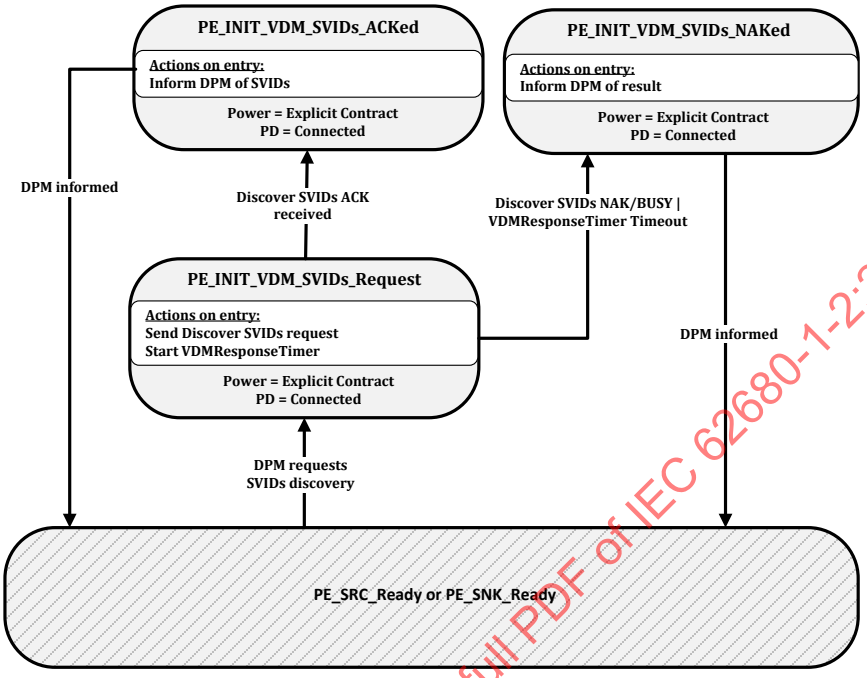
The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state when:

- The Device Policy Manager has been informed.

8.3.3.20.2 Initiator Structured VDM Discover SVIDs State Diagram

Figure 8-128 shows the state diagram for an Initiator when discovering SVIDs of its Port Partner or Cable Plug.

Figure 8-128 Initiator VDM Discover SVIDs State Diagram



8.3.3.20.2.1 PE_INIT_VDM_SVIDs_Request State

The Policy Engine transitions to the ***PE_INIT_VDM_SVIDs_Request*** state from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state when:

- The Device Policy Manager requests the discovery of the SVIDs of the Port Partner or a Cable Plug.

On entry to the ***PE_INIT_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_INIT_VDM_SVIDs_ACKed*** state when:

- A Structured VDM ***Discover SVIDs*** ACK Command response is received.

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

- A Structured VDM ***Discover SVIDs*** NAK or BUSY Command response is received or
- The ***VDMResponseTimer*** times out.

8.3.3.20.2.2 PE_INIT_VDM_SVIDs_ACKed State

On entry to the ***PE_INIT_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 when:

- The Device Policy Manager has been informed.

8.3.3.20.2.3 PE_INIT_VDM_SVIDs_NAKed State

On entry to the ***PE_INIT_VDM_SVIDs_NAKed*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the result (NAK, BUSY or timeout).

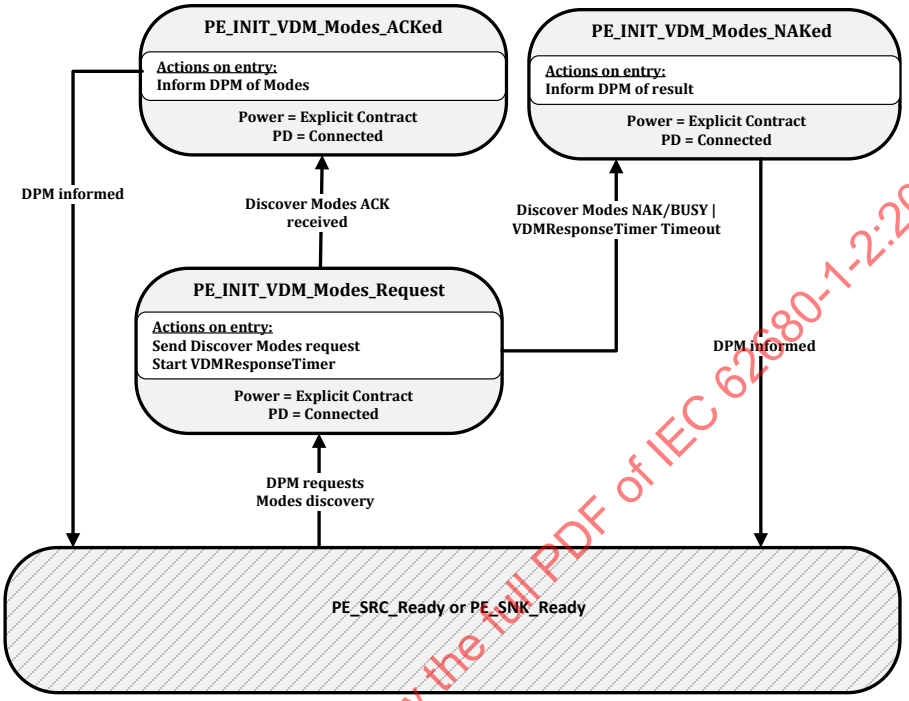
The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state when:

- The Device Policy Manager has been informed.

8.3.3.20.3 Initiator Structured VDM Discover Modes State Diagram

Figure 8-129 shows the state diagram for an Initiator when discovering Modes of its Port Partner or Cable Plug.

Figure 8-129 Initiator VDM Discover Modes State Diagram



8.3.3.20.3.1 PE_INIT_VDM_Modes_Request State

The Policy Engine transitions to the ***PE_INIT_VDM_Modes_Request*** state from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state when:

- The Device Policy Manager requests the discovery of the Modes of the Port Partner or a Cable Plug.

On entry to the ***PE_INIT_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_INIT_VDM_Modes_ACKed*** state when:

- A Structured VDM ***Discover Modes*** ACK Command response is received.

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

- A Structured VDM ***Discover Modes*** NAK or BUSY Command response is received or
- The ***VDMResponseTimer*** times out.

8.3.3.20.3.2 PE_INIT_VDM_Modes_ACKed State

On entry to the ***PE_INIT_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.

8.3.3.20.3.3 PE_INIT_VDM_Modes_NAKed State

On entry to the ***PE_INIT_VDM_Modes_NAKed*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the result (NAK, BUSY or timeout).

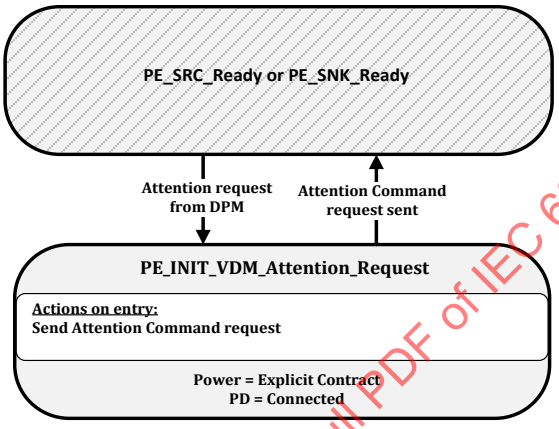
The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state for a DFP when:

- The Device Policy Manager has been informed.

8.3.3.20.4 Initiator Structured VDM Attention State Diagram

Figure 8-134 shows the state diagram for an Initiator when sending an ***Attention*** Command request.

Figure 8-130 Initiator VDM Attention State Diagram



8.3.3.20.4.1 PE_INIT_VDM_Attention_Request State

The Policy Engine transitions to the ***PE_INIT_VDM_Attention_Request*** state from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state when:

- When the Device Policy Manager requests attention from its Port Partner.

On entry to the ***PE_INIT_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 when:

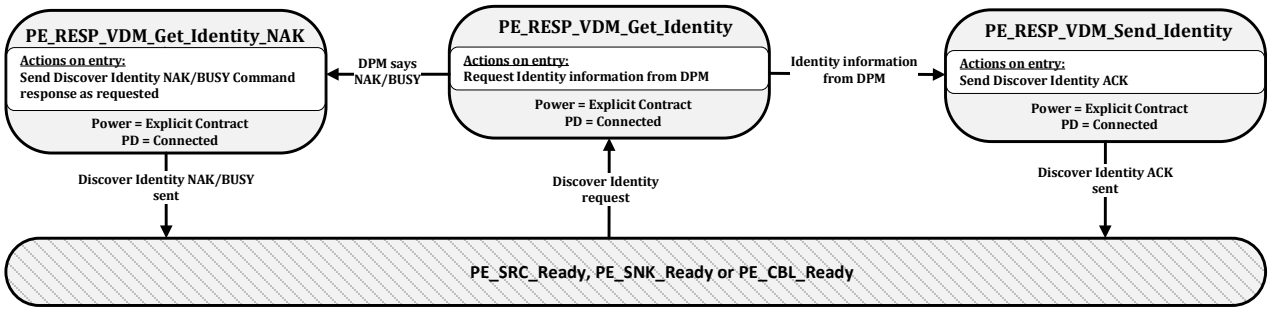
- The ***Attention*** Command request has been sent.

8.3.3.21 Responder Structured VDM State Diagrams

8.3.3.21.1 Responder Structured VDM Discover Identity State Diagram

Figure 8-131 shows the state diagram for a Responder receiving a ***Discover Identity*** Command request.

Figure 8-131 Responder Structured VDM Discover Identity State Diagram



8.3.3.21.1.1 PE_RESP_VDM_Get_Identity State

The Policy Engine transitions to the ***PE_RESP_VDM_Get_Identity*** state from either the ***PE_SRC_Ready***, ***PE_SNK_Ready*** or ***PE_CBL_Ready*** state when:

- A Structured VDM ***Discover Identity*** Command request is received.

On entry to the ***PE_RESP_VDM_Get_Identity*** state the Responder ***Shall*** request identity information from the Device Policy Manager.

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

- Identity information is received from the Device Policy Manager.

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

- The Device Policy Manager indicates that the response to the ***Discover Identity*** Command request is NAK or BUSY.

8.3.3.21.1.2 PE_RESP_VDM_Send_Identity State

On entry to the ***PE_RESP_VDM_Send_Identity*** state the Responder ***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.

8.3.3.21.1.3 PE_RESP_VDM_Get_Identity_NAK State

On entry to the ***PE_RESP_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.

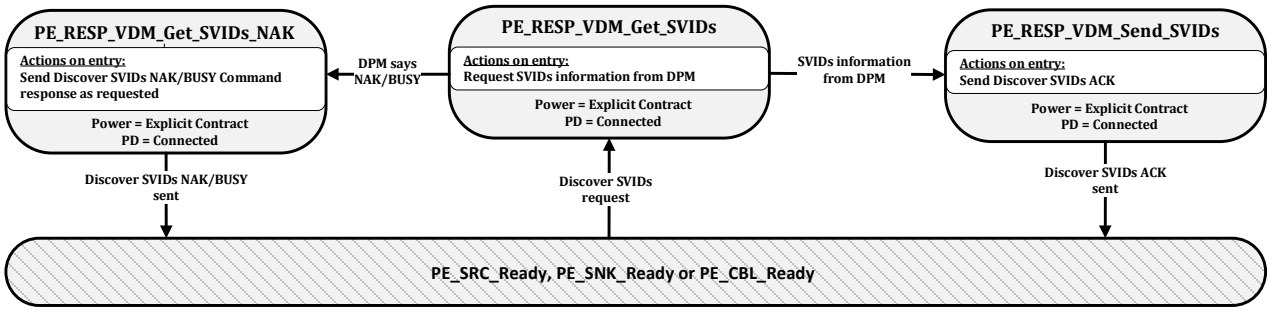
The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready, PE_SNK_Ready*** or ***PE_CBL_Ready*** state when:

- The Structured VDM ***Discover Identity*** NAK or BUSY Command response has been sent.

8.3.3.21.2 Responder Structured VDM Discover SVIDs State Diagram

Figure 8-132 shows the state diagram for a Responder when receiving a ***Discover SVIDs*** Command.

Figure 8-132 Responder Structured VDM Discover SVIDs State Diagram



8.3.3.21.2.1 PE_RESP_VDM_Get_SVIDs State

The Policy Engine transitions to the ***PE_RESP_VDM_Get_SVIDs*** state from either the ***PE_SRC_Ready, PE_SNK_Ready*** or ***PE_CBL_Ready*** state when:

- A Structured VDM ***Discover SVIDs*** Command request is received.

On entry to the ***PE_RESP_VDM_Get_SVIDs*** state the Responder ***Shall*** request SVIDs information from the Device Policy Manager.

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

- SVIDs information is received from the Device Policy Manager.

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

- The Device Policy Manager indicates that the response to the ***Discover SVIDs*** Command request is NAK or BUSY.

8.3.3.21.2.2 PE_UFP_VDM_Send_SVIDs State

On entry to the ***PE_RESP_VDM_Send_SVIDs*** state the Responder ***Shall*** send the Structured VDM ***Discover SVIDs*** ACK Command response.

The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready, PE_SNK_Ready*** or ***PE_CBL_Ready*** state when:

- The Structured VDM ***Discover SVIDs*** ACK Command response has been sent.

8.3.3.21.2.3 PE_UFP_VDM_Get_SVIDs_NAK State

On entry to the ***PE_RESP_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.

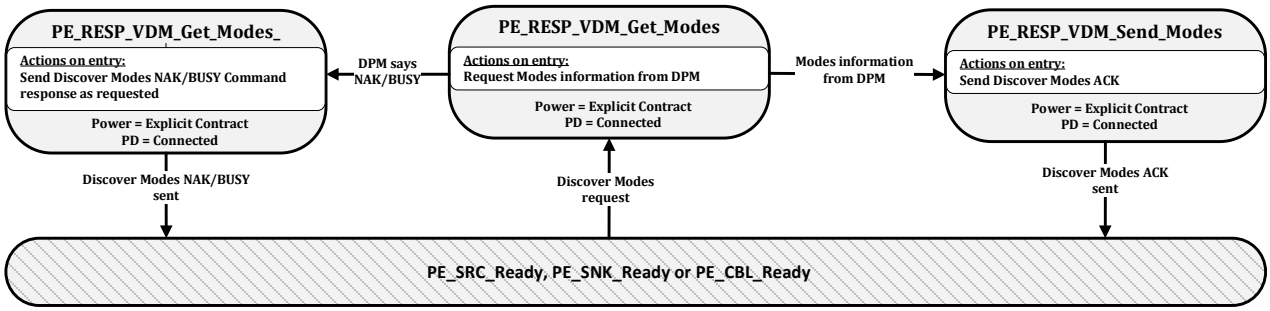
The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready***, ***PE_SNK_Ready*** or ***PE_CBL_Ready*** state when:

- The Structured VDM ***Discover SVIDs*** NAK or BUSY Command response has been sent.

8.3.3.21.3 Responder Structured VDM Discover Modes State Diagram

Figure 8-133 shows the state diagram for a Responder on receiving a ***Discover Modes*** Command.

Figure 8-133 Responder Structured VDM Discover Modes State Diagram



8.3.3.21.3.1 PE_RESP_VDM_Get_Modes State

The Policy Engine transitions to the ***PE_RESP_VDM_Get_Modes*** state from either the ***PE_SRC_Ready, PE_SNK_Ready*** or ***PE_CBL_Ready*** state when:

- A Structured VDM ***Discover Modes*** Command request is received.

On entry to the ***PE_RESP_VDM_Get_Modes*** state the Responder ***Shall*** request Modes information from the Device Policy Manager.

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

- Modes information is received from the Device Policy Manager.

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

- The Device Policy Manager indicates that the response to the ***Discover Modes*** Command request is NAK or BUSY.

8.3.3.21.3.2 PE_RESP_VDM_Send_Modes State

On entry to the ***PE_RESP_VDM_Send_Modes*** state the Responder ***Shall*** send the Structured VDM ***Discover Modes*** ACK Command response.

The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready, PE_SNK_Ready*** or ***PE_CBL_Ready*** state when:

- The Structured VDM ***Discover Modes*** ACK Command response has been sent.

8.3.3.21.3.3 PE_RESP_VDM_Get_Modes_NAK State

On entry to the ***PE_RESP_VDM_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.

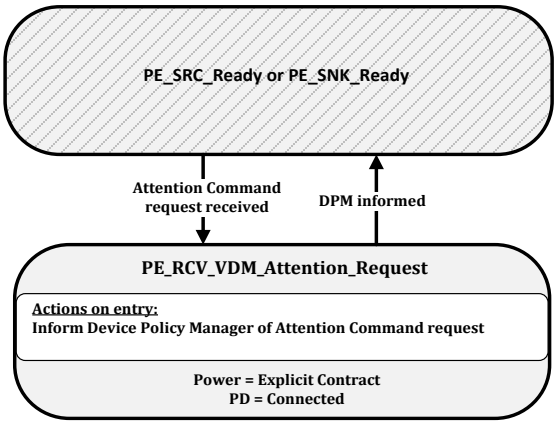
The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready*** , ***PE_SNK_Ready*** or ***PE_CBL_Ready*** state when:

- The Structured VDM ***Discover Modes*** NAK or BUSY Command response has been sent.

8.3.3.21.4 Receiving a Structured VDM Attention State Diagram

Figure 8-134 shows the state diagram when receiving an ***Attention*** Command request.

Figure 8-134 Receiving a Structured VDM Attention State Diagram



8.3.3.21.4.1 PE_RCV_VDM_Attention_Request State

The Policy Engine transitions to the ***PE_RCV_VDM_Attention_Request*** state from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state when:

- An ***Attention*** Command request is received.

On entry to the ***PE_RCV_VDM_Attention_Request*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the ***Attention*** Command request.

The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state when:

- The Device Policy Manager has been informed.

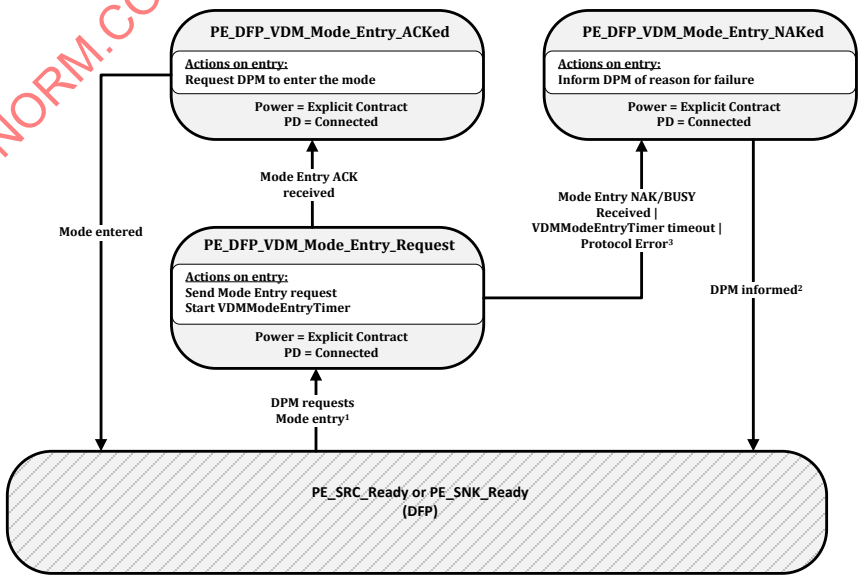
8.3.3.22 DFP Structured VDM State Diagrams

The State Diagrams in this section ***Shall*** apply to all DFPs that support Structured VDMs.

8.3.3.22.1 DFP Structured VDM Mode Entry State Diagram

Figure 8-135 shows the state operation for a DFP when entering a Mode.

Figure 8-135 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.

8.3.3.22.1.1 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.

8.3.3.22.1.2 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.

8.3.3.22.1.3 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 reason for failure (NAK, BUSY, timeout or Protocol Error).

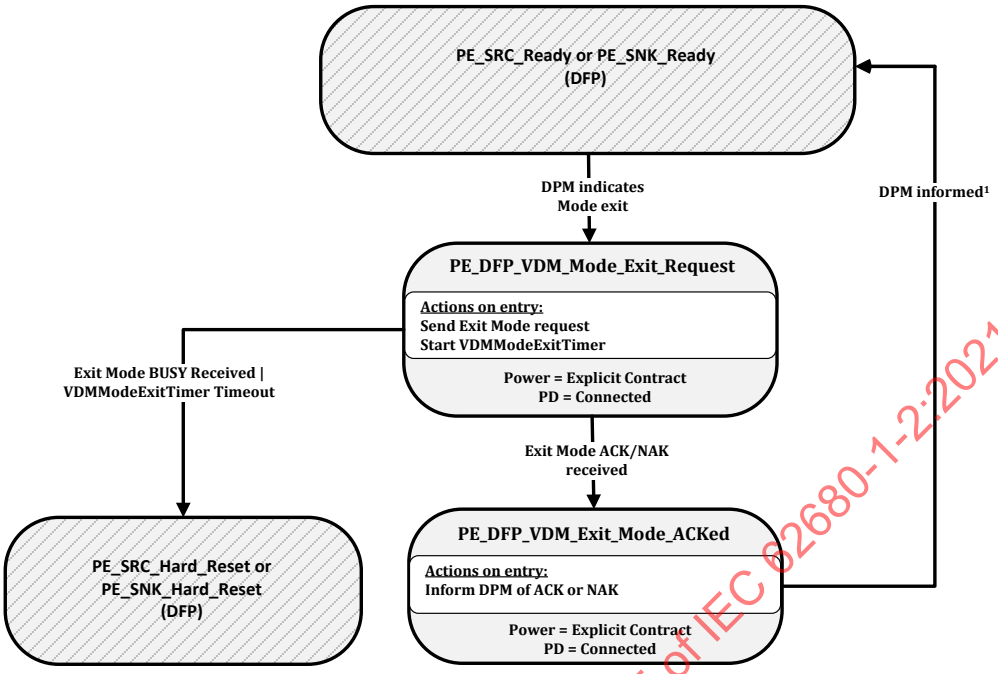
The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state for a DFP when:

- The Device Policy Manager has been informed.

8.3.3.22.2 DFP Structured VDM Mode Exit State Diagram

Figure 8-136 shows the state diagram for a DFP when exiting a Mode.

Figure 8-136 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.

8.3.3.22.2.1 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_SNK_Hard_Reset*** state depending on the present Power Role when:

- A Structured VDM ***Exit Mode*** BUSY Command response is received or
- The ***VDMModeExitTimer*** times out.

8.3.3.22.2.2 PE_DFP_VDM_DFP_Mode_Exit_ACKed State

On Exit to the ***PE_DFP_VDM_Mode_Exit_ACKed*** state the Policy Engine ***Shall*** inform the Device Policy Manager Of the result: ACK or NAK.

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.

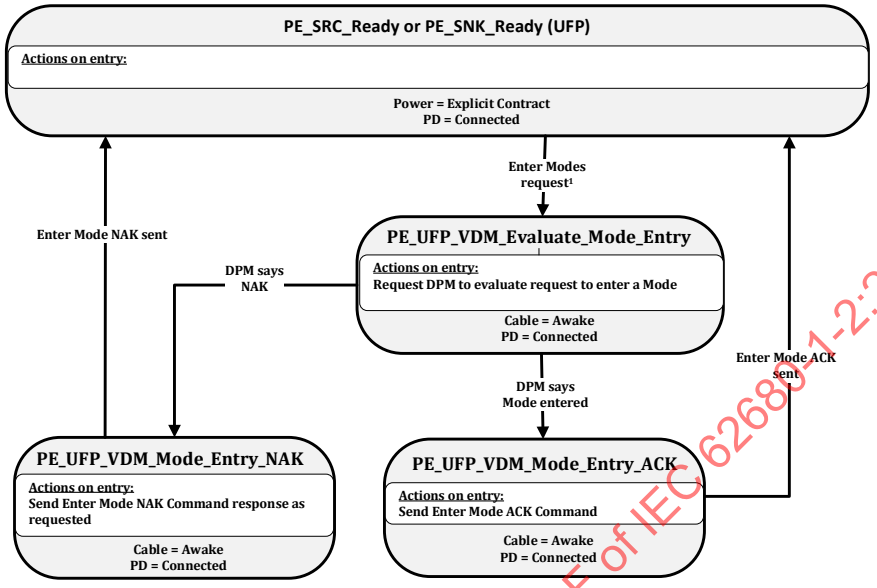
8.3.3.23 UFP Structured VDM State Diagrams

The State Diagrams in this section ***Shall*** apply to all UFPs that support Structured VDMs.

8.3.3.23.1 UFP Structured VDM Enter Mode State Diagram

Figure 8-137 shows the state diagram for a UFP in response to an ***Enter Mode*** Command.

Figure 8-137 UFP Structured VDM Enter Mode State Diagram



[1] The UFP is required to be in USB operation or USB Safe State at this point.

8.3.3.23.1.1 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.

8.3.3.23.1.2 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.

8.3.3.23.1.3 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.

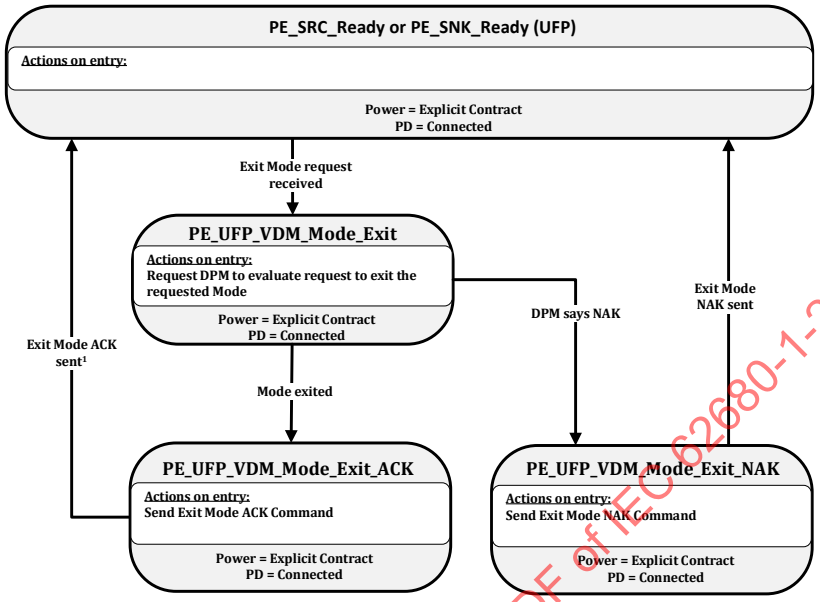
The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state for a UFP when:

- The Structured VDM ***Enter Mode*** NAK Command response has been sent.

8.3.3.23.2 UFP Structured VDM Exit Mode State Diagram

Figure 8-138 shows the state diagram for a UFP in response to an ***Exit Mode*** Command.

Figure 8-138 UFP Structured VDM Exit Mode State Diagram



[1] The UFP is required to be in USB operation or USB Safe State at this point.

8.3.3.23.2.1 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 Manager 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.

8.3.3.23.2.2 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.

8.3.3.23.2.3 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.

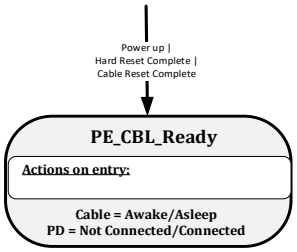
8.3.3.24 Cable Plug Specific State Diagrams

The State Diagrams in this section ***Shall*** apply to all Cable Plugs that support Structured VDMs.

8.3.3.24.1 Cable Plug Cable Ready State Diagram

Figure 8-139 shows the Cable Ready state diagram for a Cable Plug.

Figure 8-139 Cable Ready VDM State Diagram



8.3.3.24.1.1 PE_CBL_Ready State

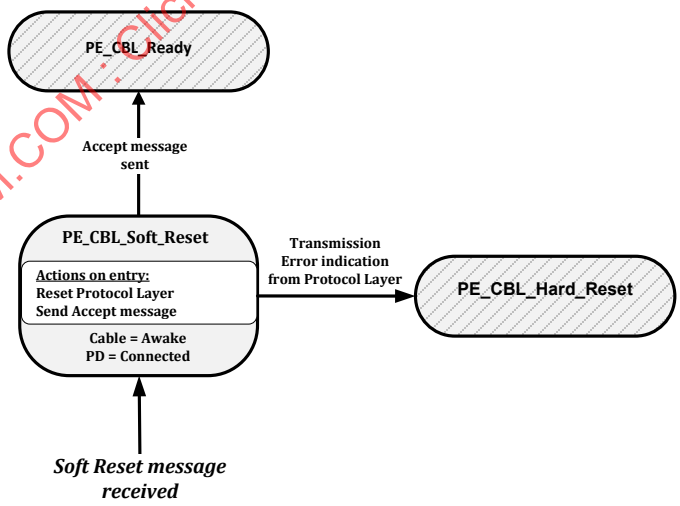
The ***PE_CBL_Ready*** state shown in the following sections is the normal operational state for a Cable Plug and where it starts after power up or a Hard/Cable Reset.

8.3.3.24.2 Soft/Hard/Cable Reset

8.3.3.24.2.1 Cable Plug Soft Reset State Diagram

Figure 8-140 shows the Cable Plug state diagram on reception of a ***Soft_Reset*** Message.

Figure 8-140 Cable Plug Soft Reset State Diagram



8.3.3.24.2.1.1 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.

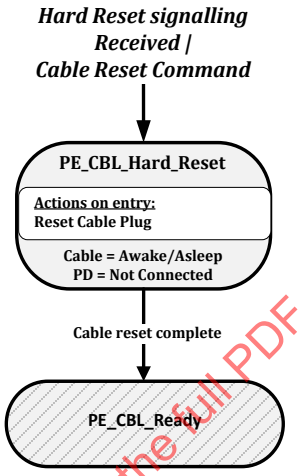
The Policy Engine ***Shall*** transition to the ***PE_CBL_Hard_Reset*** state when:

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

8.3.3.24.2.2 Cable Plug Hard Reset State Diagram

Figure 8-141 shows the Cable Plug state diagram for a Hard Reset or Cable Reset.

Figure 8-141 Cable Plug Hard Reset State Diagram



8.3.3.24.2.1 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).

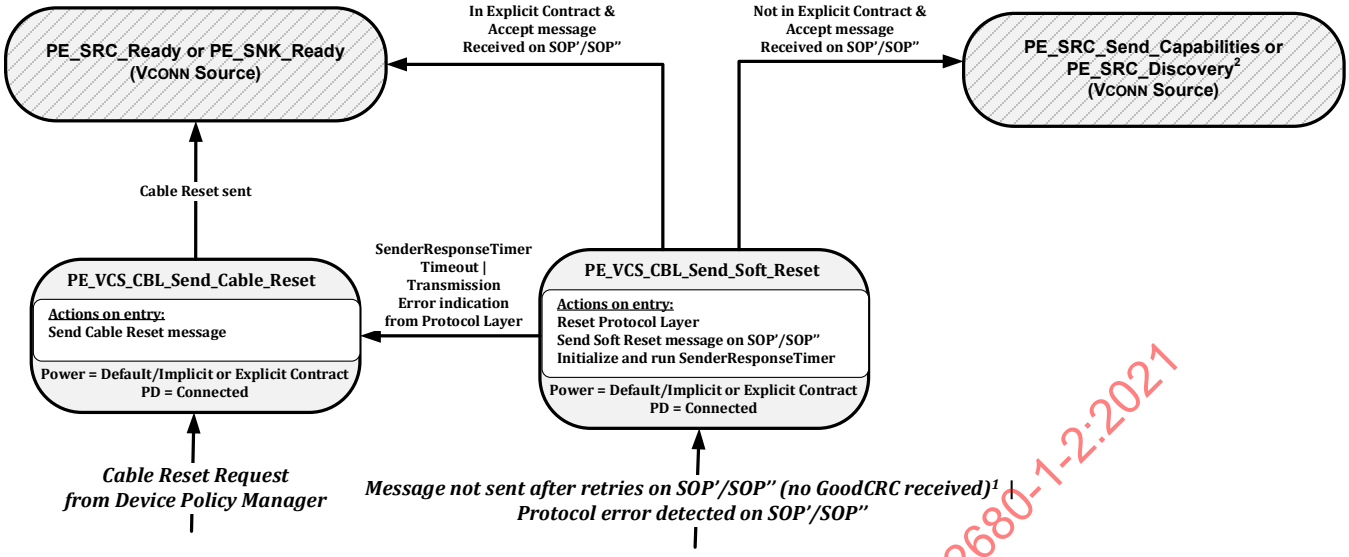
The Policy Engine ***Shall*** transition to the ***PE_CBL_Ready*** state when:

- The Cable Plug reset is complete.

8.3.3.24.2.3 VCONN Source SOP'/SOP'' Soft Reset or Cable Reset of a Cable Plug or VPD State Diagram

Figure 8-142 below shows the state diagram for the Policy Engine in a VCONN Source when performing a Soft Reset or Cable Reset of a Cable Plug or VPD on ***SOP'***/***SOP''***. The following sections describe operation in each of the states.

Figure 8-142 VCONN Source Soft Reset or Cable Reset of a Cable Plug or VPD State Diagram



[1] Excludes the *Soft_Reset* Message itself.

[2] Sink only communicates with the Cable Plug when in an Explicit Contract. If the *Discover Identity* 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.

8.3.3.24.2.3.1 PE_VCS_CBL_Send_Soft_Reset State

The *PE_VCS_CBL_Send_Soft_Reset* state ***Shall*** be entered from any state when a Protocol Error is detected on *SOP'*/*SOP"* by the Protocol Layer (see Section 6.8.1) or when a Message has not been sent after retries on *SOP'*/*SOP"* while communicating with a Cable Plug/VPD or whenever the Device Policy Manager directs a Soft Reset on *SOP'*/*SOP"*.

On entry to the *PE_VCS_CBL_Send_Soft_Reset* state the Policy Engine ***Shall*** request the *SOP'*/*SOP"* Protocol Layer to perform a Soft Reset, then ***Shall*** send a *Soft_Reset* Message on *SOP'*/*SOP"* 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 when:

- There is no Explicit Contract in place and
- An *Accept* Message has been received on *SOP'*/*SOP"*.

The Policy Engine ***Shall*** transition to the either the *PE_SRC_Send_Capabilities* state or *PE_SRC_Discovery* state, depending on the DFP's VCONN Source's Power Role, when:

- There is an Explicit Contract in place and
- An *Accept* Message has been received on *SOP'*/*SOP"*.

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

- A *SenderResponseTimer* timeout occurs
- Or the Protocol Layer indicates that a transmission error has occurred.

8.3.3.24.2.3.2 PE_VCS_CBL_Send_Cable_Reset State

The *PE_VCS_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_VCS_CBL_Send_Cable_Reset* state the Policy Engine ***Shall*** request the Protocol Layer to send *Cable Reset* Signaling.

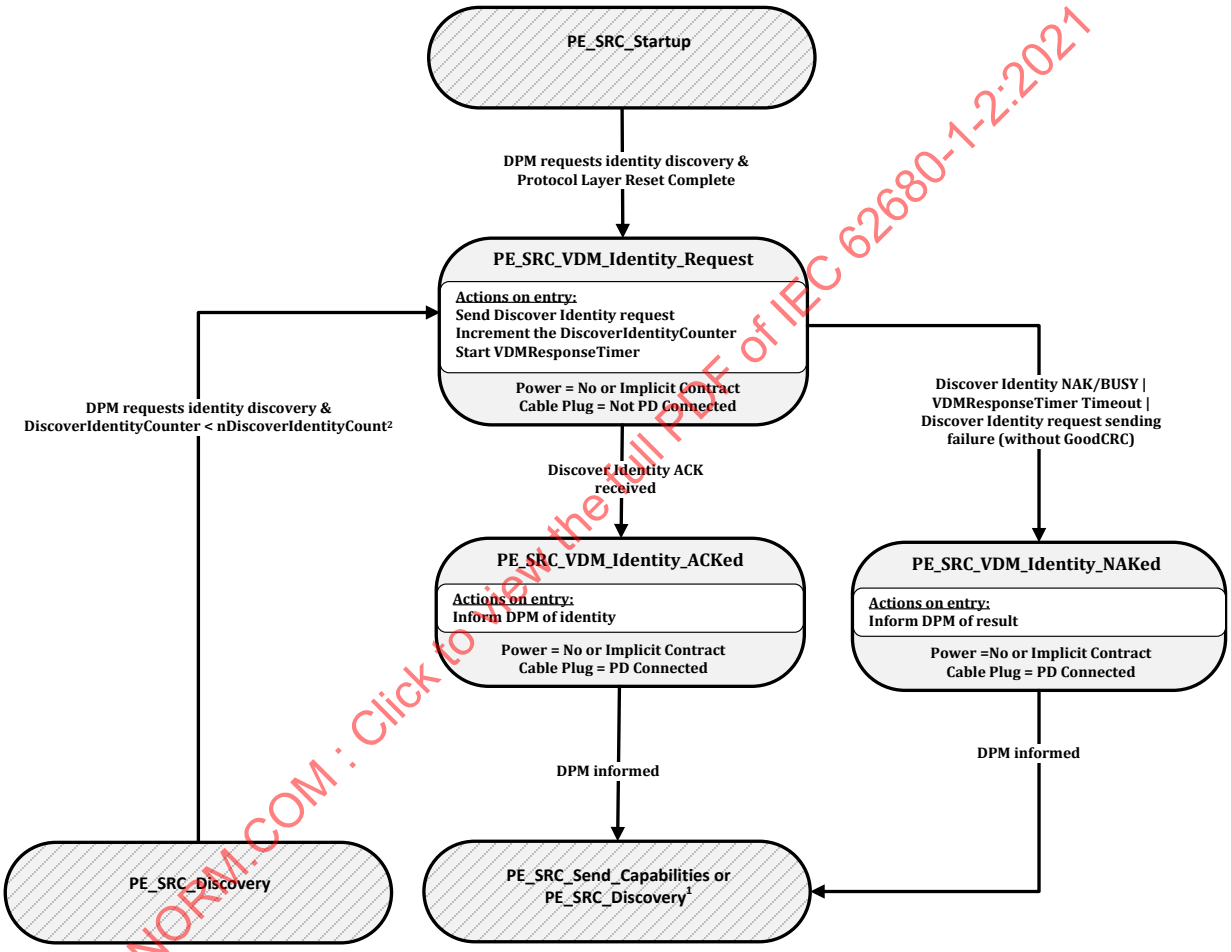
The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state, depending on the VCONN Source's Power Role, when:

- ***Cable Reset*** Signaling has been sent.

8.3.3.24.3 Source Startup Structured VDM Discover Identity of a Cable Plug State Diagram

Figure 8-143 shows the state diagram for Source discovery of identity information from a Cable Plug during the startup sequence.

Figure 8-143 Source Startup Structured VDM Discover Identity State Diagram



[1] If the ***Discover Identity*** 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*** Messages are sent out at a regular rate.

8.3.3.24.3.1 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 8-143 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).

8.3.3.24.3.2 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.

8.3.3.24.3.3 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:

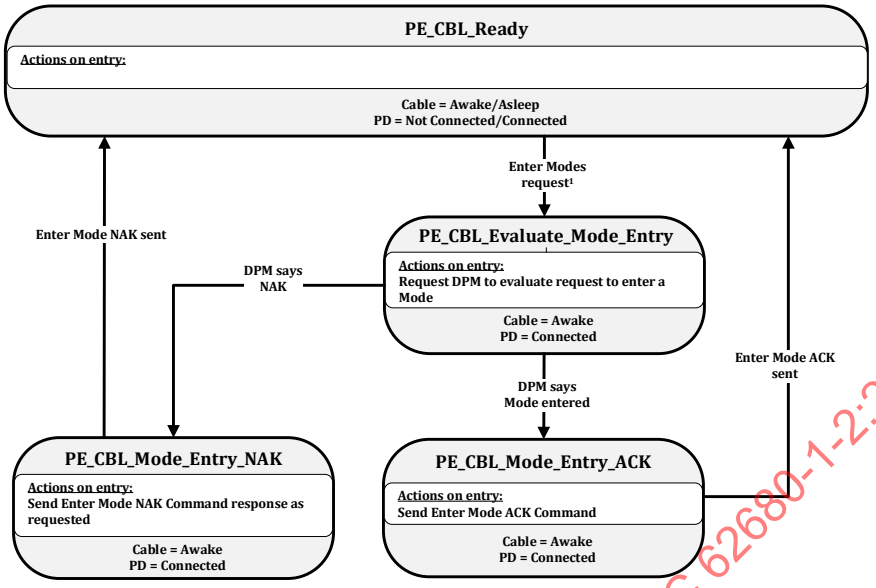
- The Device Policy Manager has been informed.

8.3.3.24.4 Cable Plug Mode Entry/Exit

8.3.3.24.4.1 Cable Plug Structured VDM Enter Mode State Diagram

Figure 8-144 shows the state diagram for a Cable Plug in response to an ***Enter Mode*** Command.

Figure 8-144 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.

8.3.3.24.4.1.1 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.

8.3.3.24.4.1.2 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.

8.3.3.24.4.1.3 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.

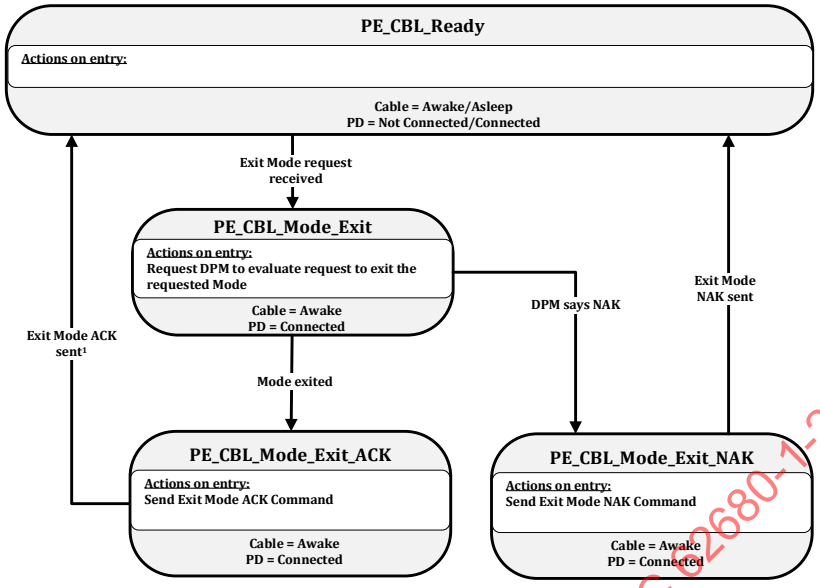
The Policy Engine ***Shall*** transition to the ***PE_CBL_Ready*** state when:

- The Structured VDM ***Enter Mode*** NAK Command response has been sent.

8.3.3.24.4.2 Cable Plug Structured VDM Exit Mode State Diagram

Figure 8-145 shows the state diagram for a Cable Plug in response to an ***Exit Mode*** Command.

Figure 8-145 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.

8.3.3.24.4.2.1 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.

8.3.3.24.4.2.2 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.

8.3.3.24.4.2.3 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:

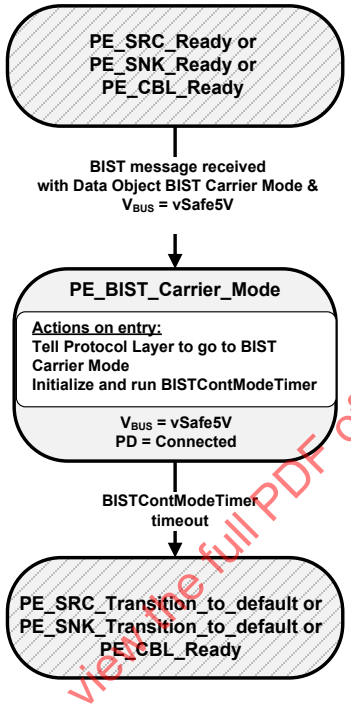
- The Structured VDM ***Exit Mode*** NAK Command response has been sent.

8.3.3.25 BIST State diagrams

8.3.3.25.1 BIST Carrier Mode State Diagram

Figure 8-146 shows the state diagram required by a UUT, which can be either a Source, Sink or Cable Plug, when operating in ***BIST Carrier Mode***. Transitions ***Shall*** be from either the ***PE_SRC_Ready***, ***PE_SNK_Ready*** or ***PE_CBL_Ready*** states.

Figure 8-146 BIST Carrier Mode State Diagram



8.3.3.25.1.1 PE_BIST_Carrier_Mode State

The Source, Sink or Cable Plug ***Shall*** enter the ***PE_BIST_Carrier_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 Carrier Mode*** BIST Data Object and
- V_{BUS} is at ***vSafe5V***.

On entry to the ***PE_BIST_Carrier_Mode*** state the Policy Engine ***Shall*** tell the Protocol Layer to go to BIST Carrier 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.

8.3.3.26 USB Type-C Referenced States

This section contains states cross-referenced from the ***[USB Type-C 2.0]*** specification.

8.3.3.26.1 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 2.0]*** 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.

8.3.3.27 Policy Engine States

Table 8-71 lists the states used by the various state machines.

Table 8-71 Policy Engine States

State name	Reference
Source Port	
PE_SRC_Startup	Section 8.3.3.2.1
PE_SRC_Discovery	Section 8.3.3.2.2
PE_SRC_Send_Capabilities	Section 8.3.3.2.3
PE_SRC_Negotiate_Capability	Section 8.3.3.2.4
PE_SRC_Transition_Supply	Section 8.3.3.2.5
PE_SRC_Ready	Section 8.3.3.2.6
PE_SRC_Disabled	Section 8.3.3.2.7
PE_SRC_Capability_Response	Section 8.3.3.2.8
PE_SRC_Hard_Reset	Section 8.3.3.2.9
PE_SRC_Hard_Reset_Received	Section 8.3.3.2.10
PE_SRC_Transition_to_default	Section 8.3.3.2.11
PE_SRC_Get_Sink_Cap	Section 8.3.3.2.12
PE_SRC_Wait_New_Capabilities	Section 8.3.3.2.13
Sink Port	
PE_SNK_Startup	Section 8.3.3.3.1
PE_SNK_Discovery	Section 8.3.3.3.2
PE_SNK_Wait_for_Capabilities	Section 8.3.3.3.3
PE_SNK_Evaluate_Capability	Section 8.3.3.3.4
PE_SNK_Select_Capability	Section 8.3.3.3.5
PE_SNK_Transition_Sink	Section 8.3.3.3.6
PE_SNK_Ready	Section 8.3.3.3.7
PE_SNK_Hard_Reset	Section 8.3.3.3.8
PE_SNK_Transition_to_default	Section 8.3.3.3.9
PE_SNK_Give_Sink_Cap	Section 8.3.3.3.10
PE_SNK_Get_Source_Cap	Section 8.3.3.3.11
Soft Reset and Protocol Error	
Source Port Soft Reset	
PE_SRC_Send_Soft_Reset	Section 8.3.3.4.1.1
PE_SRC_Soft_Reset	Section 8.3.3.4.1.2
Sink Port Soft Reset	
PE_SNK_Send_Soft_Reset	Section 8.3.3.4.2.1
PE_SNK_Soft_Reset	Section 8.3.3.4.2.2
Data Reset	
DFP Data Reset	
PE_DDR_Send_Data_Reset	Section 8.3.3.5.1.1
PE_DDR_Data_Reset_Received	Section 8.3.3.5.1.2
PE_DDR_Wait_For_VCONN_Off	Section 8.3.3.5.1.3
PE_DDR_Perform_Data_Reset	Section 8.3.3.5.1.4

State name	Reference
UFP Data Reset	
PE_UDR_Send_Data_Reset	Section 8.3.3.5.2.1
PE_UDR_Data_Reset_Received	Section 8.3.3.5.2.2
PE_UDR_Turn_Off_VCONN	Section 8.3.3.5.2.3
PE_UDR_Send_Ps_Rdy	Section 8.3.3.5.2.4
PE_UDR_Wait_For_Data_Reset_Complete	Section 8.3.3.5.2.5
Not Supported Message	
Source Port Not Supported	
PE_SRC_Send_Not_Supported	Section 8.3.3.6.1.1
PE_SRC_Not_Supported_Received	Section 8.3.3.6.1.2
PE_SRC_Chunk_Received	Section 8.3.3.6.1.3
Sink Port Not Supported	
PE_SNK_Send_Not_Supported	Section 8.3.3.6.2.1
PE_SNK_Not_Supported_Received	Section 8.3.3.6.2.2
PE_SNK_Chunk_Received	Section 8.3.3.6.2.3
Source Port Ping	
PE_SRC_Ping	Section 8.3.3.7.1
Source Alert	
Source Port Source Alert	
PE_SRC_Send_Source_Alert	Section 8.3.3.8.1.1
Sink Port Source Alert	
PE_SNK_Source_Alert_Received	Section 8.3.3.8.2.1
Sink Port Sink Alert	
PE_SNK_Send_Sink_Alert	Section 8.3.3.8.3.1
Source Port Sink Alert	
PE_SRC_Sink_Alert_Received	Section 8.3.3.8.4.1
Source Extended Capabilities	
Sink Port Get Source Capabilities Extended	
PE_SNK_Get_Source_Cap_Ext	Section 8.3.3.9.1.1
Source Port Give Source Capabilities Extended	
PE_SRC_Give_Source_Cap_Ext	Section 8.3.3.9.2.1
Source Status	
Sink Port Get Source Status	
PE_SNK_Get_Source_Status	Section 8.3.3.10.1.1
Source Port Give Source Status	
PE_SRC_Give_Source_Status	Section 8.3.3.10.2.1
Source Port Get Sink Status	
PE_SRC_Get_Sink_Status	Section 8.3.3.10.3.1
Sink Port Give Sink Status	
PE_SNK_Give_Sink_Status	Section 8.3.3.10.4.1
Sink Port Get PPS Status	
PE_SNK_Get_PPS_Status	Section 8.3.3.10.5.1
Source Port Give PPS Status	
PE_SRC_Give_PPS_Status	Section 8.3.3.10.6.1

State name	Reference
Battery Capabilities	
Get Battery Capabilities	
PE_Get_Battery_Cap	Section 8.3.3.11.1.1
Give Battery Capabilities	
PE_Give_Battery_Cap	Section 8.3.3.11.2.1
Battery Status	
Get Battery Status	
PE_Get_Battery_Status	Section 8.3.3.12.1.1
Give Battery Status	
PE_Give_Battery_Status	Section 8.3.3.12.2.1
Manufacturer Information	
Get Manufacturer Information	
PE_Get_Manufacturer_Info	Section 8.3.3.13.1
Give Manufacturer Information	
PE_Give_Manufacturer_Info	Section 8.3.3.13.2
Country Codes and Information	
Get Country Codes	
PE_Get_Country_Codes	Section 8.3.3.14.1.1
Give Country Codes	
PE_Give_Country_Codes	Section 8.3.3.14.2.1
Get Country Information	
PE_Get_Country_Info	Section 8.3.3.14.3.1
Give Country Information	
PE_Give_Country_Info	Section 8.3.3.14.4.1
Enter USB	
DFP Enter USB	
PE_DEU_Send_Enter_USB	Section 8.3.3.15.1.1
UFP Enter USB	
PE_UEU_Enter_USB_Received	Section 8.3.3.15.2.1
Security Request/Response	
Send Security Request	
PE_Send_Security_Request	Section 8.3.3.16.1
Send Security Response	
PE_Send_Security_Response	Section 8.3.3.16.2
Security Response Received	
PE_Security_Response_Received	Section 8.3.3.16.3
Firmware Update Request/Response	
Send Firmware Update Request	
PE_Send_Firmware_Update_Request	Section 8.3.3.17.1.1
Send Firmware Update Response	
PE_Send_Firmware_Update_Response	Section 8.3.3.17.2.1
Firmware Update Response Received	
PE_Firmware_Update_Response_Received	Section 8.3.3.17.3.1
Dual-Role Port	
DFP to UFP Data Role Swap	
PE_DRS_DFP_UFP_Evaluate_Swap	Section 8.3.3.18.1.2

State name	Reference
PE_DRS_DFP_UFP_Accept_Swap	Section 8.3.3.18.1.3
PE_DRS_DFP_UFP_Change_to_UFP	Section 8.3.3.18.1.4
PE_DRS_DFP_UFP_Send_Swap	Section 8.3.3.18.1.5
PE_DRS_DFP_UFP_Reject_Swap	Section 8.3.3.18.1.6
UFP to DFP Data Role Swap	
PE_DRS_UFP_DFP_Evaluate_Swap	Section 8.3.3.18.2.2
PE_DRS_UFP_DFP_Accept_Swap	Section 8.3.3.18.2.3
PE_DRS_UFP_DFP_Change_to_DFP	Section 8.3.3.18.2.4
PE_DRS_UFP_DFP_Send_Swap	Section 8.3.3.18.2.5
PE_DRS_UFP_DFP_Reject_Swap	Section 8.3.3.18.2.6
Source to Sink Power Role Swap	
PE_PRS_SRC_SNK_Evaluate_Swap	Section 8.3.3.18.3.2
PE_PRS_SRC_SNK_Accept_Swap	Section 8.3.3.18.3.3
PE_PRS_SRC_SNK_Transition_to_off	Section 8.3.3.18.3.4
PE_PRS_SRC_SNK_Assert_Rd	Section 8.3.3.18.3.5
PE_PRS_SRC_SNK_Wait_Source_on	Section 8.3.3.18.3.6
PE_PRS_SRC_SNK_Send_Swap	Section 8.3.3.18.3.7
PE_PRS_SRC_SNK_Reject_Swap	Section 8.3.3.18.3.8
Sink to Source Power Role Swap	
PE_PRS_SNK_SRC_Evaluate_Swap	Section 8.3.3.18.4.2
PE_PRS_SNK_SRC_Accept_Swap	Section 8.3.3.18.4.3
PE_PRS_SNK_SRC_Transition_to_off	Section 8.3.3.18.4.4
PE_PRS_SNK_SRC_Assert_Rp	
PE_PRS_SNK_SRC_Source_on	Section 8.3.3.18.4.5
PE_PRS_SNK_SRC_Send_Swap	Section 8.3.3.18.4.7
PE_PRS_SNK_SRC_Reject_Swap	Section 8.3.3.18.4.8
Source to Sink Fast Role Swap	
PE_FRS_SRC_SNK_Evaluate_Swap	Section 8.3.3.18.5.2
PE_FRS_SRC_SNK_Accept_Swap	Section 8.3.3.18.5.3
PE_FRS_SRC_SNK_Transition_to_off	Section 8.3.3.18.5.4
PE_FRS_SRC_SNK_Assert_Rd	Section 8.3.3.18.5.5
PE_FRS_SRC_SNK_Wait_Source_on	Section 8.3.3.18.5.6
Sink to Source Fast Role Swap	
PE_FRS_SNK_SRC_Start_AMS	Section 8.3.3.18.6.1
PE_FRS_SNK_SRC_Send_Swap	Section 8.3.3.18.6.2
PE_FRS_SNK_SRC_Transition_to_off	Section 8.3.3.18.6.3
PE_FRS_SNK_SRC_Vbus_Applied	Section 8.3.3.18.6.4
PE_FRS_SNK_SRC_Assert_Rp	Section 8.3.3.18.6.5
PE_FRS_SNK_SRC_Source_on	Section 8.3.3.18.6.6
Dual-Role Source Port Get Source Capabilities	
PE_DR_SRC_Get_Source_Cap	Section 8.3.3.18.7.1

State name	Reference
Dual-Role Source Port Give Sink Capabilities	
PE_DR_SRC_Give_Sink_Cap	Section 8.3.3.18.8.1
Dual-Role Sink Port Get Sink Capabilities	
PE_DR_SNK_Get_Sink_Cap	Section 8.3.3.18.9.1
Dual-Role Sink Port Give Source Capabilities	
PE_DR_SNK_Give_Source_Cap	Section 8.3.3.18.10.1
Dual-Role Source Port Get Source Capabilities Extended	
PE_DR_SRC_Get_Source_Cap_Ext	Section 8.3.3.18.11.1
Dual-Role Sink Port Give Source Capabilities Extended	
PE_DR_SNK_Give_Source_Cap_Ext	Section 8.3.3.18.12.1
USB Type-C VCONN Swap	
PE_VCS_Send_Swap	Section 8.3.3.19.1
PE_VCS_Evaluate_Swap	Section 8.3.3.19.2
PE_VCS_Accept_Swap	Section 8.3.3.19.3
PE_VCS_Reject_Swap	Section 8.3.3.19.4
PE_VCS_Wait_For_VCONN	Section 8.3.3.19.5
PE_VCS_Turn_Off_VCONN	Section 8.3.3.19.6
PE_VCS_Turn_On_VCONN	Section 8.3.3.19.7
PE_VCS_Send_Ps_Rdy	Section 8.3.3.19.8
PE_VCS_Force_VCONN	Section 8.3.3.19.9
Initiator Structured VDM	
Initiator to Port Structured VDM Discover Identity	
PE_INIT_PORT_VDM_Identity_Request	Section 8.3.3.20.1.1
PE_INIT_PORT_VDM_Identity_ACKed	Section 8.3.3.20.1.2
PE_INIT_PORT_VDM_Identity_NAKed	Section 8.3.3.20.1.3
Initiator Structured VDM Discover SVIDs	
PE_INIT_VDM_SVIDs_Request	Section 8.3.3.20.2.1
PE_INIT_VDM_SVIDs_ACKed	Section 8.3.3.20.2.2
PE_INIT_VDM_SVIDs_NAKed	Section 8.3.3.20.2.3
Initiator Structured VDM Discover Modes	
PE_INIT_VDM_Modes_Request	Section 8.3.3.20.3.1
PE_INIT_VDM_Modes_ACKed	Section 8.3.3.20.3.2
PE_INIT_VDM_Modes_NAKed	Section 8.3.3.20.3.3
Initiator Structured VDM Attention	
PE_INIT_VDM_Attention_Request	Section 8.3.3.20.4.1
Responder Structured VDM	
Responder Structured VDM Discovery Identity	
PE_RESP_VDM_Get_Identity	Section 8.3.3.21.1.1
PE_RESP_VDM_Send_Identity	Section 8.3.3.21.1.2
PE_RESP_VDM_Get_Identity_NAK	Section 8.3.3.21.1.3
Responder Structured VDM Discovery SVIDs	
PE_RESP_VDM_Get_SVIDs	Section 8.3.3.21.2.1
PE_RESP_VDM_Send_SVIDs	Section 8.3.3.21.2.2

State name	Reference
PE_RESP_VDM_Get_SVIDs_NAK	Section 8.3.3.21.2.3
Responder Structured VDM Discovery Modes	
PE_RESP_VDM_Get_Modes	Section 8.3.3.21.3.1
PE_RESP_VDM_Send_Modes	Section 8.3.3.21.3.2
PE_RESP_VDM_Get_Modes_NAK	Section 8.3.3.21.3.3
Receiving a Structured VDM Attention	
PE_RCV_VDM_Attention_Request	Section 8.3.3.21.4.1
DFP Structured VDM	
DFP Structured VDM Mode Entry	
PE_DFP_VDM_Mode_Entry_Request	Section 8.3.3.22.1.1
PE_DFP_VDM_Mode_Entry_ACKed	Section 8.3.3.22.1.2
PE_DFP_VDM_Mode_Entry_NAKed	Section 8.3.3.22.1.3
DFP Structured VDM Mode Exit	
PE_DFP_VDM_Mode_Exit_Request	Section 8.3.3.22.2.1
PE_DFP_VDM_Mode_Exit_ACKed	Section 8.3.3.22.2.2
UFP Structure VDM	
UFP Structured VDM Enter Mode	
PE_UFP_VDM_Evaluate_Mode_Entry	Section 8.3.3.23.1.1
PE_UFP_VDM_Mode_Entry_ACK	Section 8.3.3.23.1.2
PE_UFP_VDM_Mode_Entry_NAK	Section 8.3.3.23.1.3
UFP Structured VDM Exit Mode	
PE_UFP_VDM_Mode_Exit	Section 8.3.3.23.2.1
PE_UFP_VDM_Mode_Exit_ACK	Section 8.3.3.23.2.2
PE_UFP_VDM_Mode_Exit_NAK	Section 8.3.3.23.2.3
Cable Plug Specific	
Cable Ready	
PE_CBL_Ready	Section 8.3.3.24.1.1
Mode Entry	
PE_CBL_Evaluate_Mode_Entry	Section 8.3.3.24.4.1.1
PE_CBL_Mode_Entry_ACK	Section 8.3.3.24.4.1.2
PE_CBL_Mode_Entry_NAK	Section 8.3.3.24.4.1.3
Mode Exit	
PE_CBL_Mode_Exit	Section 8.3.3.24.4.2.1
PE_CBL_Mode_Exit_ACK	Section 8.3.3.24.4.2.2
PE_CBL_Mode_Exit_NAK	Section 8.3.3.24.4.2.3
Cable Soft Reset	
PE_CBL_Soft_Reset	Section 8.3.3.24.2.1.1
Cable Hard Reset	
PE_CBL_Hard_Reset	Section 8.3.3.24.2.2.1
VCONN Source Soft Reset or Cable Reset	
PE_VCS_CBL_Send_Soft_Reset	Section 8.3.3.24.2.3.1
PE_VCS_CBL_Send_Cable_Reset	Section 8.3.3.24.2.3.2
Source Startup Structured VDM Discover Identity	
PE_SRC_VDM_Identity_Request	Section 8.3.3.24.3.1

State name	Reference
PE_SRC_VDM_Identity_ACKed	Section 8.3.3.24.3.2
PE_SRC_VDM_Identity_NAKed	Section 8.3.3.24.3.3
BIST Carrier Mode	
PE_BIST_Carrier_Mode	Section 8.3.3.25.1.1
USB Type-C referenced states	
ErrorRecovery	Section 8.3.3.26.1

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 ***[USBTypeCBridge 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 3.2]***; 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 6.4.1.2.2.2), 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 3.2]***, ***[USB Type-C 2.0]*** (USB Type-C®) or ***[USBBC 1.2]*** When operating in ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** or ***[USBBC 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

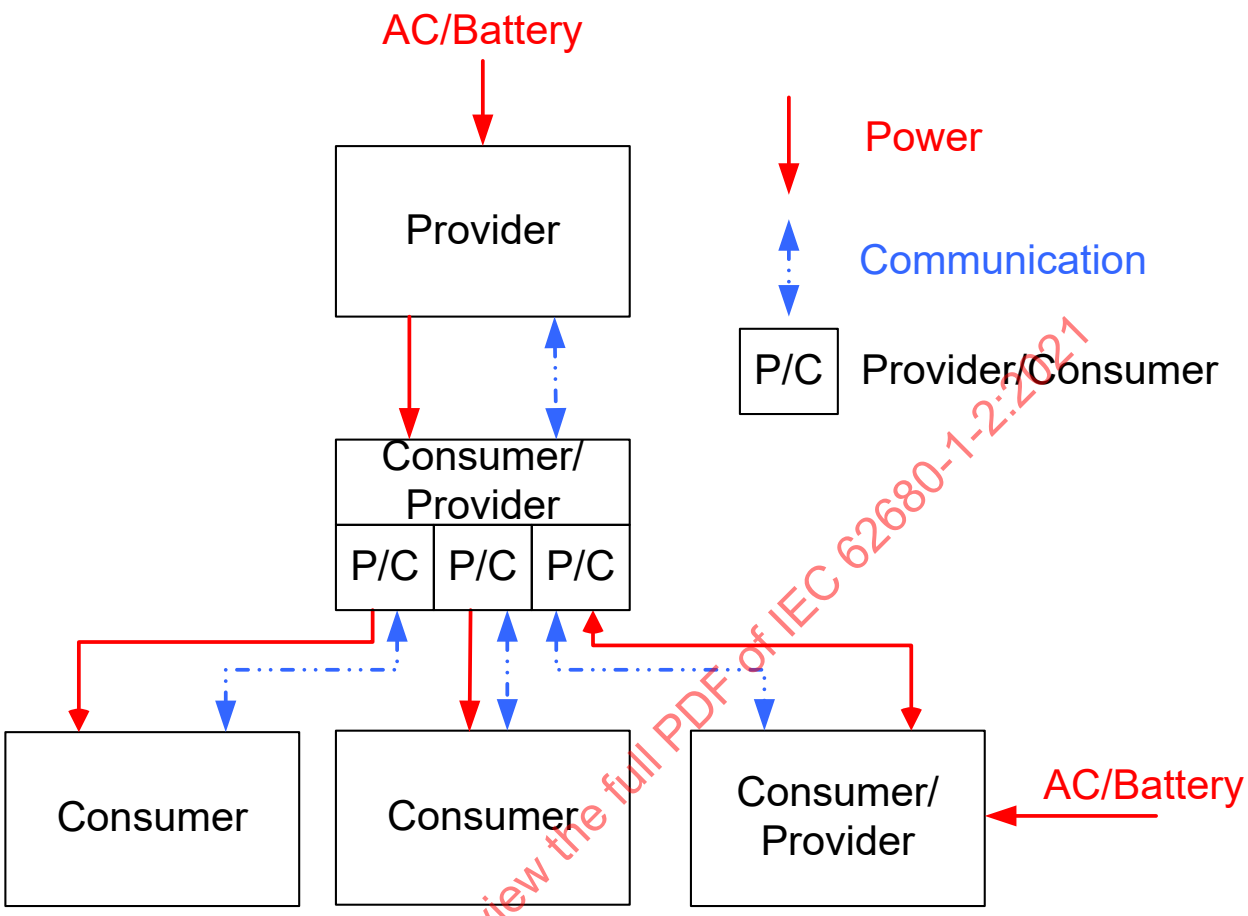
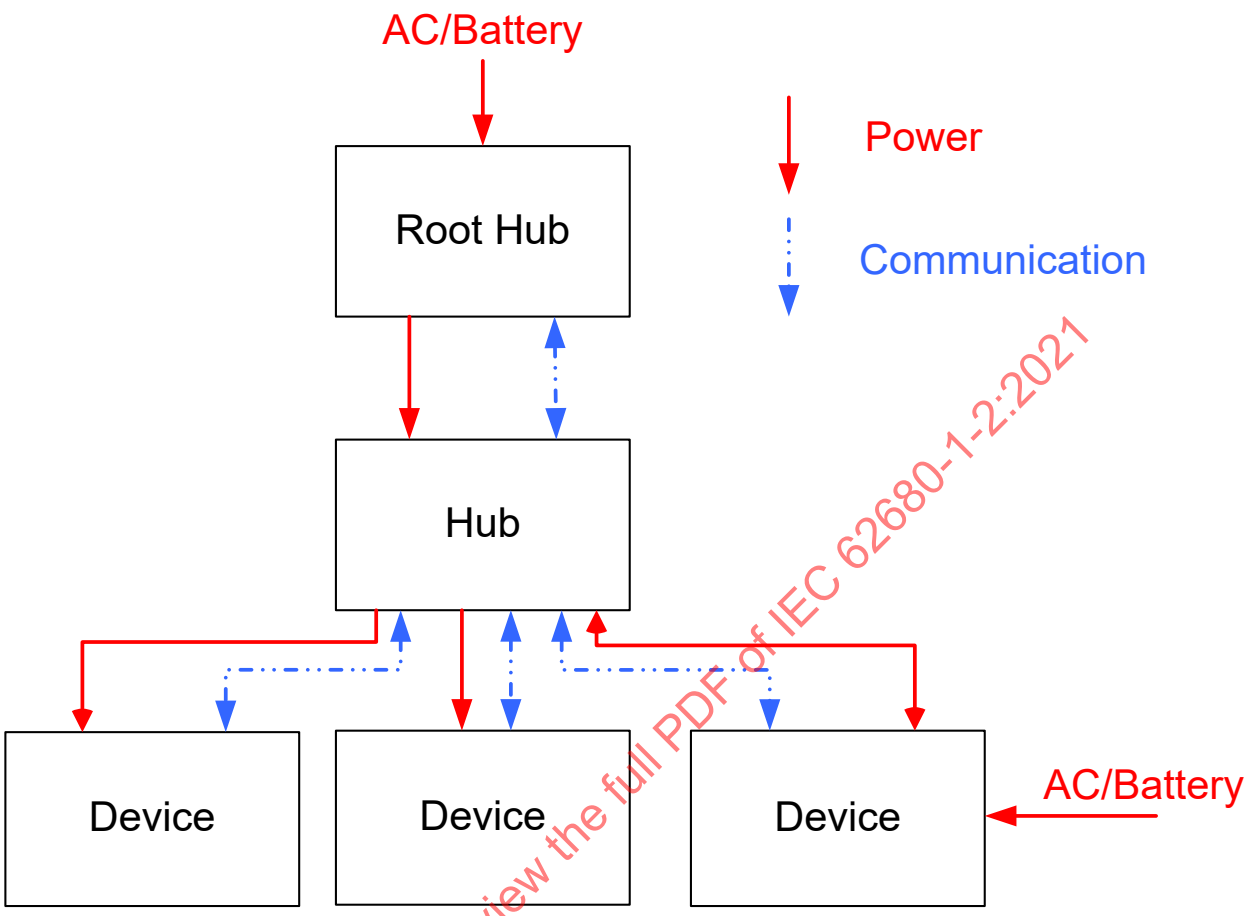


Figure 9-2 shows how this same topology can be mapped to USB. In a USB based system, policy is managed by the host and communication of system level policy information is via standard USB data line communication. This is a separate mechanism to the USB Power Delivery V_{BUS} protocol which is used to 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 3.2]*, *[USB Type-C 2.0]* or *[USBBC 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 ***[USBTypeCBridge 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 3.2]***.

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

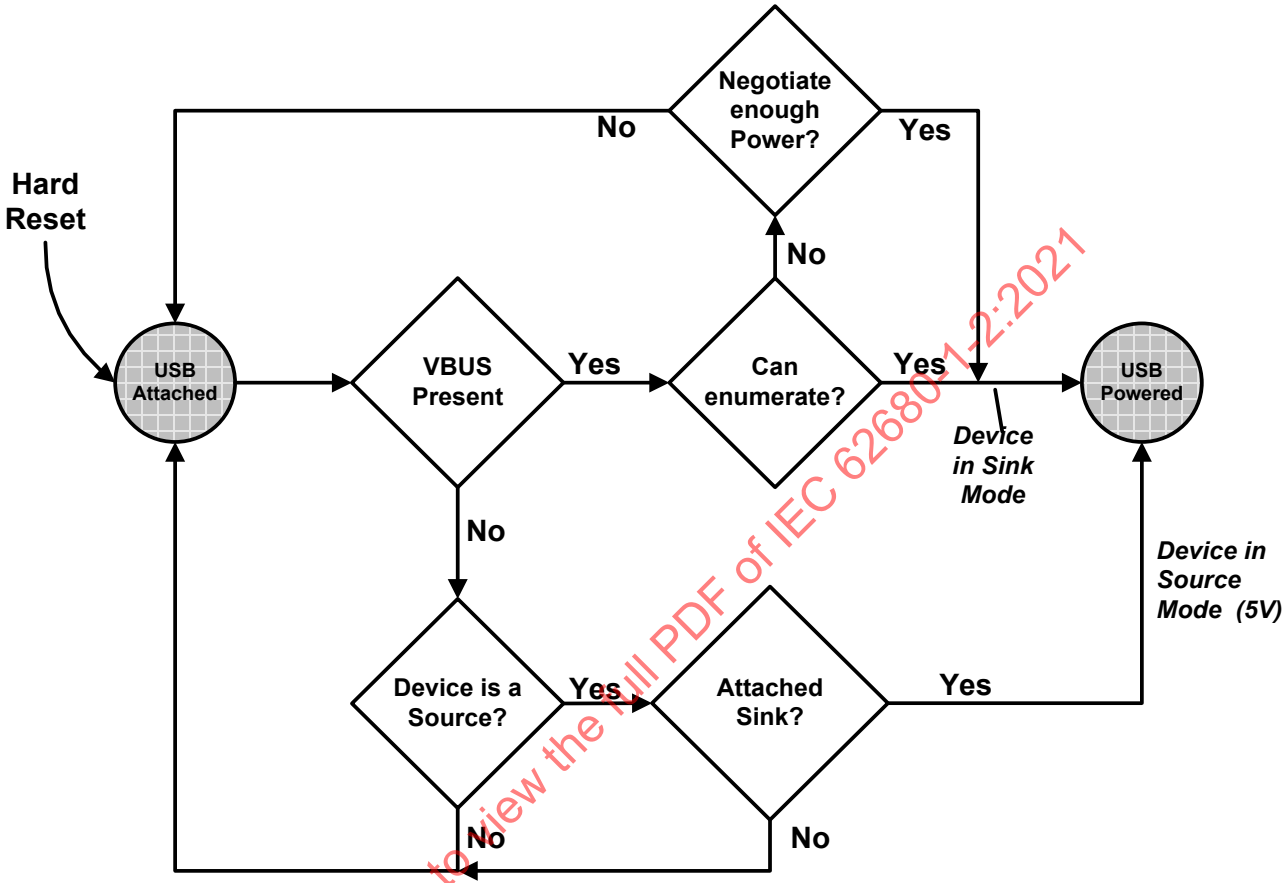


Figure 9-4 shows how a PDUSB Device determines when to transition from the USB Powered state to the USB Attached state when the device is a Consumer. A PDUSB Device determines that it is performing a Power Role Swap as described in Section 8.3.3.18.3 and Section 8.3.3.18.4. See Section 7.1.5 for additional information on device behavior during Hard Resets.

Figure 9-4 Any USB State to USB Attached State Transition (When operating as a Consumer)

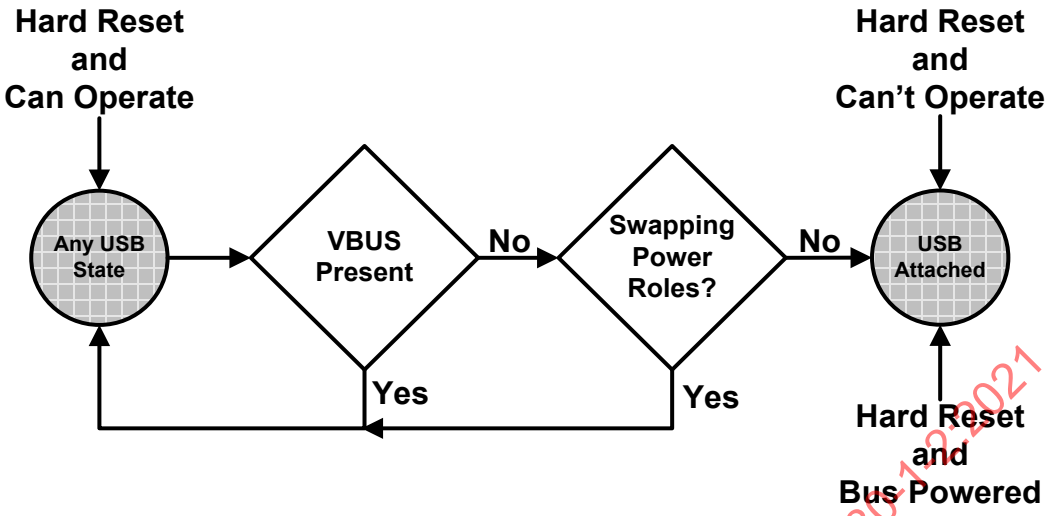


Figure 9-5 shows how a PDUSB Device determines when to transition from the USB Powered state to the USB Attached state when the device is a Provider.

Figure 9-5 Any USB State to USB Attached State Transition (When operating as a Provider)

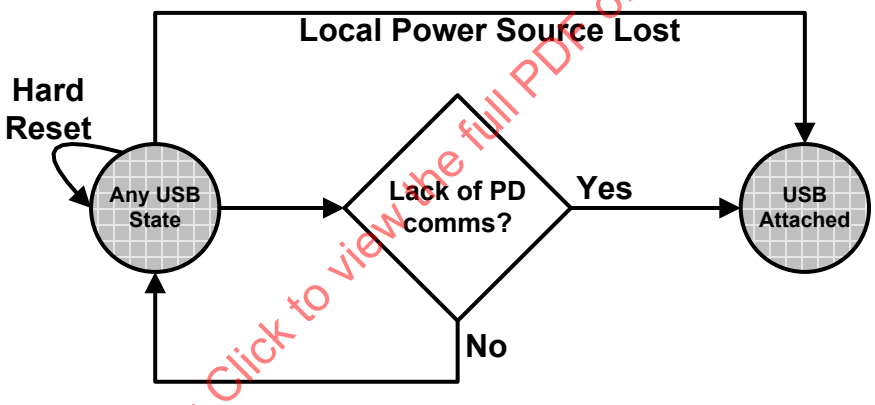
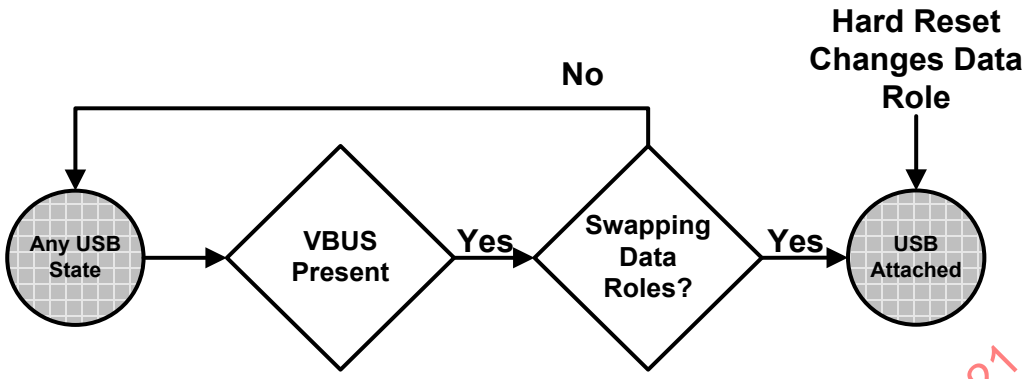


Figure 9-6 shows how a PDUSB Device using the USB Type-C connector determines when to transition from the USB Powered state to the USB Attached state after a Data Role Swap has been performed i.e. it has just changed from operation as a PDUSB Host to operation as a PDUSB Device. The Data Role Swap is 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 6.8.3. See Section 7.1.5 for additional information on device behavior during Hard Resets.

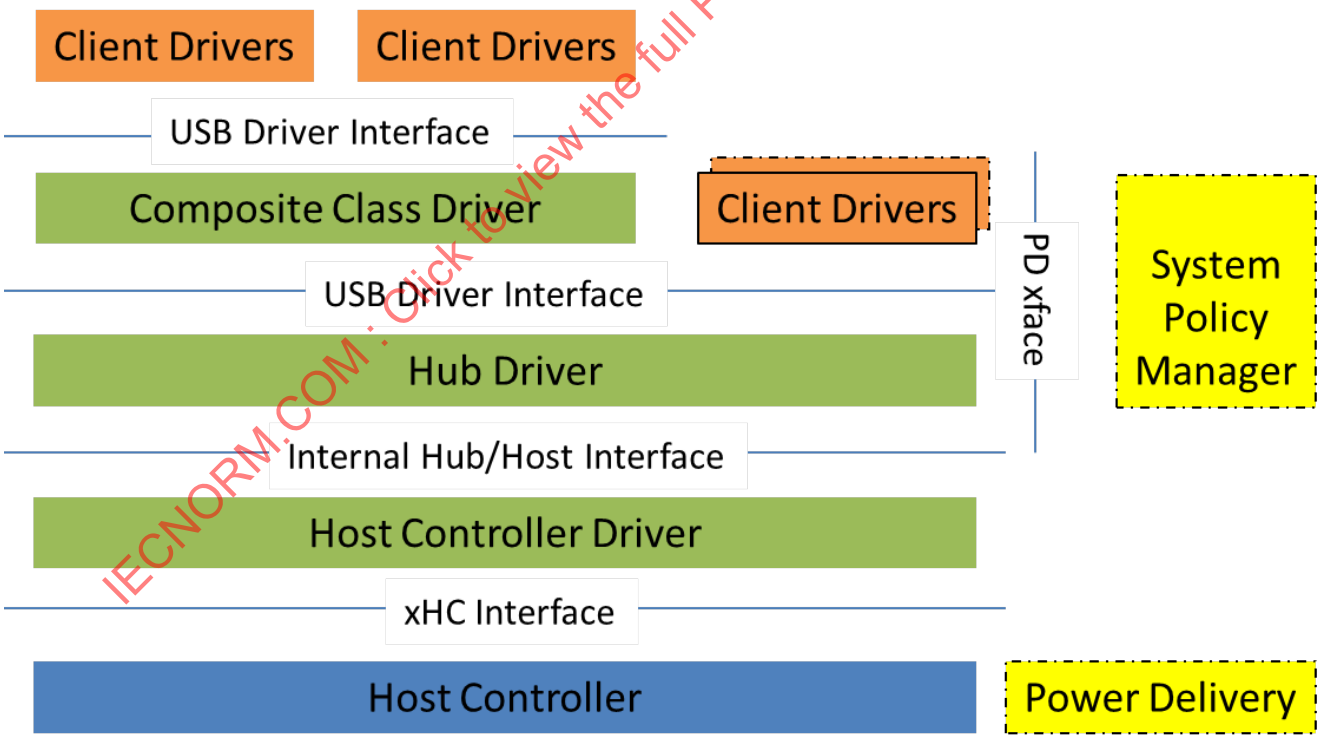
Figure 9-6 Any USB State to USB Attached State Transition (After a USB Type-C Data Role Swap)



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

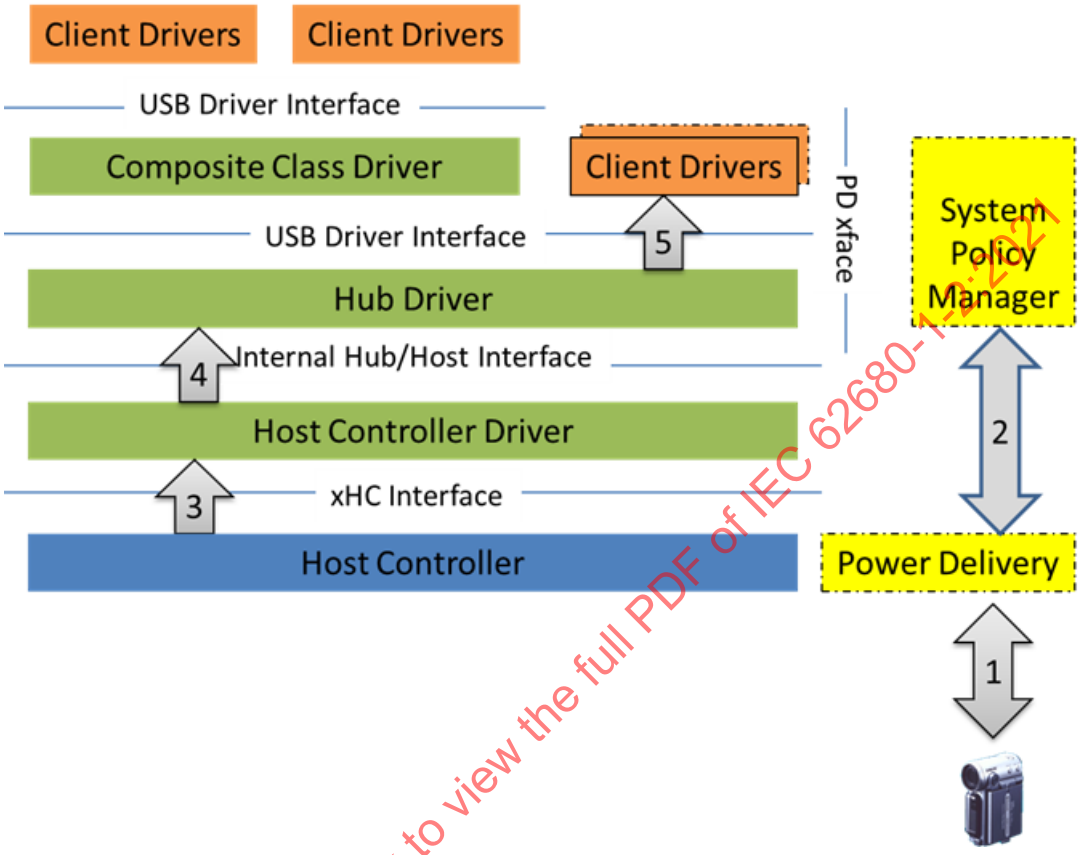


9.1.4 PDUSB Device Enumeration

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
POWER_DELIVERY_CAPABILITY	06H	Defines the various PD Capabilities of this device
BATTERY_INFO_CAPABILITY	07H	Provides information on each Battery supported by the device
PD_CONSUMER_PORT_CAPABILITY	08H	The Consumer characteristics of a Port on the device
PD_PROVIDER_PORT_CAPABILITY	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	*bLength*	1	Number	Size of descriptor
1	*bDescriptorType*	1	Constant	DEVICE CAPABILITY Descriptor type
2	*bDevCapabilityType*	1	Constant	Capability type: ***POWER_DELIVERY_CAPABILITY***
3	*bReserved*	1	Reserved	***Shall*** be set to zero.
4	*bmAttributes*	4	Bitmap	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: 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 *bcdBCVersion* 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 *bcdPDVersion* 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 bcdUSBTypeCVersion field 7 – ***Reserved***. ***Shall*** be set to zero.

Offset	Field	Size	Value	Description
				15:8 bmPowerSource. At least one of the following bits 8, 9 and 14 ***Shall*** be set to indicate which power sources are supported. **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. 31:16 ***Reserved*** and ***Shall*** be set to zero.
8	*bcdBCVersion*	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 *bmAttributes* field.
10	*bcdPDVersion*	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 *bmAttributes* field.
12	*bcdUSBTypeCVersion*	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 *bmAttributes* 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	*bLength*	1	Number	Size of descriptor
1	*bDescriptorType*	1	Constant	DEVICE CAPABILITY Descriptor type
2	*bDevCapabilityType*	1	Constant	Capability type: ***BATTERY_INFO_CAPABILITY***
3	*iBattery*	1	Index	Index of string descriptor ***Shall*** contain the user friendly name for this Battery.
4	*iSerial*	1	Index	Index of string descriptor ***Shall*** contain the Serial Number String for this Battery.
5	*iManufacturer*	1	Index	Index of string descriptor ***Shall*** contain the name of the Manufacturer for this Battery.
6	*bBatteryId*	1	Number	Value ***Shall*** be used to uniquely identify this Battery in status Messages.
7	*bReserved*	1	Number	***Reserved*** and ***Shall*** be set to zero.
8	*dwChargedThreshold*	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	*dwWeakThreshold*	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	*dwBatteryDesignCapacity*	4	mWh	***Shall*** contain the design capacity of the Battery.
20	*dwBatteryLastFullchargeCapacity*	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	*bLength*	1	Number	Size of descriptor
1	*bDescriptorType*	1	Constant	DEVICE CAPABILITY Descriptor type
2	*bDevCapabilityType*	1	Constant	Capability type: ***PD_CONSUMER_PORT_CAPABILITY***
3	*bReserved*	1	Number	***Reserved*** and ***Shall*** be set to zero.
4	*bmCapabilities*	2	Bitmap	Capability: This field ***Shall*** indicate the specification the Consumer Port will operate under. **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	*wMinVoltage*	2	Number	***Shall*** contain the minimum voltage in 50mV units that this Consumer is capable of operating at.
8	*wMaxVoltage*	2	Number	***Shall*** contain the maximum voltage in 50mV units that this Consumer is capable of operating at.
10	*wReserved*	2	Number	***Reserved*** and ***Shall*** be set to zero.
12	*dwMaxOperatingPower*	4	Number	***Shall*** contain the maximum power in 10mW units this Consumer can draw when it is in a steady state operating mode.
16	*dwMaxPeakPower*	4	Number	***Shall*** contain the maximum power in 10mW units this Consumer can draw for a short duration of time (*dwMaxPeakPowerTime*) before it falls back into a steady state.
20	*dwMaxPeakPowerTime*	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	*bLength*	1	Number	Size of descriptor
1	*bDescriptorType*	1	Constant	DEVICE CAPABILITY Descriptor type
2	*bDevCapabilityType*	1	Constant	Capability type: ***PD_PROVIDER_PORT_CAPABILITY***
3	*bReserved*	1	Number	***Reserved*** and ***Shall*** be set to zero.

<table>
<tr><th>Offset</th><th>Field</th><th>Size</th><th>Value</th><th>Description</th></tr>
<tr><td>4</td><td>bmCapabilities</td><td>2</td><td>Bitmap</td><td>This field Shall indicate the specification the Provider Port will operation under.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<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>
</table></td></tr>
<tr><td>6</td><td>bNumOfPDObjects</td><td>1</td><td>Number</td><td>Shall indicate the number of Power Data Objects.</td></tr>
<tr><td>7</td><td>bReserved</td><td>1</td><td>Number</td><td>Reserved and Shall be set to zero.</td></tr>
<tr><td>8</td><td>wPowerDataObject1</td><td>4</td><td>Bitmap</td><td>Shall contain the first Power Data Object supported by this Provider Port. See Section 6.4.1 for details of the Power Data Objects.</td></tr>
<tr><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td></tr>
<tr><td>4*(N+1)</td><td>wPowerDataObjectN</td><td>4</td><td>Bitmap</td><td>Shall contain the 2nd and subsequent Power Data Objects supported by this Provider Port. See Section 6.4.1 for details of the Power Data Objects.</td></tr>
</table>

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 ***[USBTypeCBridge 1.0]*** irrespective of whether the PDUSB Hub is a Provider, a Consumer, or both.

9.3.1 PD Specific Requests

PD 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	***Get_Battery_Status***	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 3.2]***.

Table 9-7 PD Request Codes

bRequest	Value
GET_BATTERY_STATUS	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
BATTERY_WAKE_MASK	Device	40
CHARGING_POLICY	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
10000000B	*Get_Battery_Status*	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	*bBatteryAttributes*	1	Number	***Shall*** indicate whether a Battery is installed and whether this is charging or discharging. **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	*bBatterySOC*	1	Number	***Shall*** indicate the Battery State of Charge given as percentage value from Battery Remaining Capacity.
2	*bBatteryStatus*	1	Number	If a Battery is present ***Shall*** indicate the present status of the Battery. **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	*bRemoteWakeCapStatus*	1	Bitmap	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. **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	*wRemainingOperatingTime*	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	*wRemainingChargeTime*	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
00000000B	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 2.0]*** 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.

The PDP Rating is a manufacturer declared value placed on packaging to help the user understand the capabilities of a charger or the size of charger required to power their device. For PDP values of 10W and above the PDP ***Shall*** be declared as an integer number of Watts. For PDP values less than 10W, the PDP ***Shall*** be declared in increments of 0.5W.

The Source Power rules define a PDP to provide a simple way to tell the user about the capabilities of their power adapter or device. PDP Rating is akin to the wattage rating of a light bulb – bigger numbers mean more capability.

The Sink Power rules define a PDP to provide a simple way to tell the user which Sources will provide adequate power for their Sink.

10.2 Source Power Rules

In order to meet the expectations of the user, the Maximum Current/Power in the Source Capabilities PDO or APDO for Sources with a PDP Rating of x Watts ***Shall*** be as follows:

- Maximum current for Normative and Optional Fixed/Variable supply PDOs ***Shall*** be either RoundUp(x/Voltage) or RoundDown(x/Voltage) to the nearest 10mA.
- Maximum current for Programmable Power Supply APDOs ***Shall*** be as defined in Table 10-7. Note that when the Constant Power bit is set in the APDO, the programmable power supply's output current is as defined in Table 10-7 however the programmable power supply will limit its output current so that the product of its actual output voltage times the output current does not exceed the PDP.
- Maximum current for Programmable Power Supply APDOs not defined in Table 10-7 ***Shall*** be RoundDown (x/Max Voltage) to the nearest 50mA.
- Maximum power for Optional Battery supply PDOs ***Shall*** be ≤ x.

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 Ratings are always capable of providing the power to devices designed for use with adapters with a smaller PDP Rating
- 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

Considerations	Rationale	Consequence
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 2.0]***	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 Rating of x Watts ***Shall*** support are as defined in Table 10-2.

Table 10-2 Normative Voltages and Minimum Currents

PDP Rating (W)	Current at 5V (A)	Current at 9V (A)	Current at 15V (A)	Current at 20V (A)
0.5 ≤ x ≤ 15	x ÷ 5			
15 < x ≤ 27	3	x ÷ 9		
27 < x ≤ 45	3	3	x ÷ 15	
45 < x ≤ 60	3	3	3	x ÷ 20
60 < x ≤ 100	3	3	3	x ÷ 20[1]
[1] Requires a 5A cable.				

Figure 10-1 illustrates the minimum current that a Source ***Shall*** support at each voltage for a given PDP Rating. Note: Not illustrated are that currents higher than 3A are allowed to be offered up to a limit of 5A

given that a 5A cable is detected by the Source and the voltage times current remains within the Source PDP Rating.

Figure 10-1 Source Power Rule Illustration

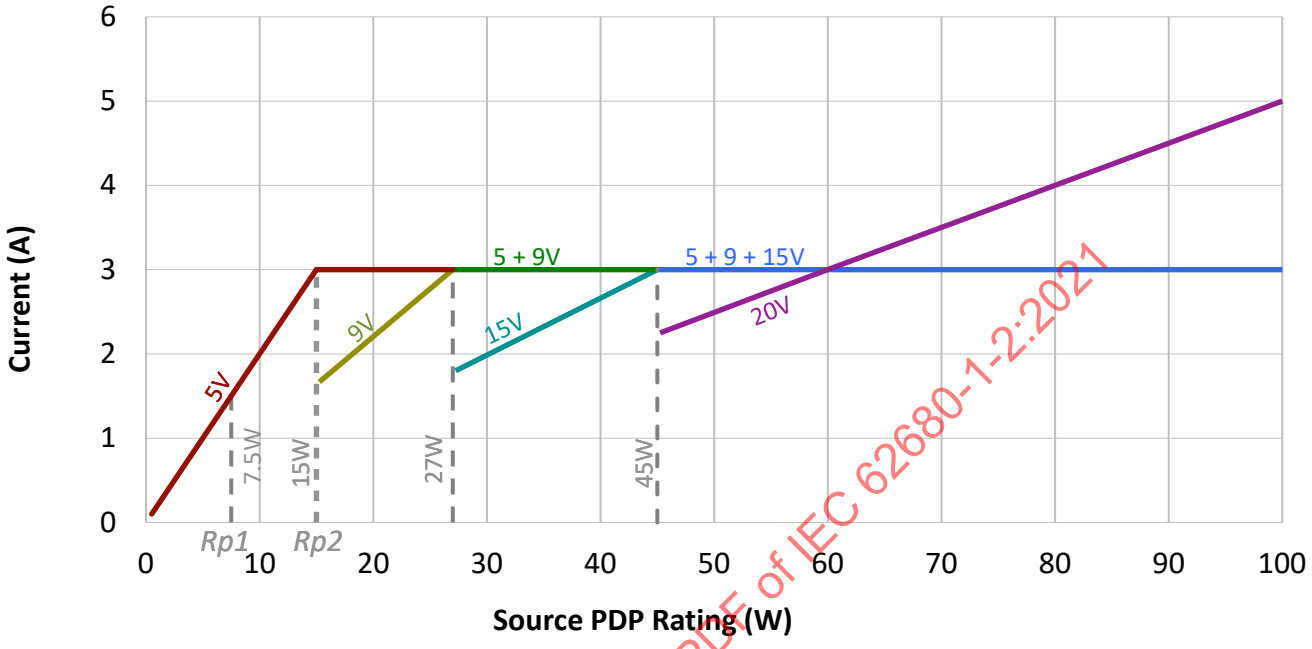


Figure 10-2 shows an example of an adapter with a rating at 50W. The adapter is required to support 20V at 2.5A, 15V at 3A, 9V at 3A and 5V at 3A.

Figure 10-2 Source Power Rule Example

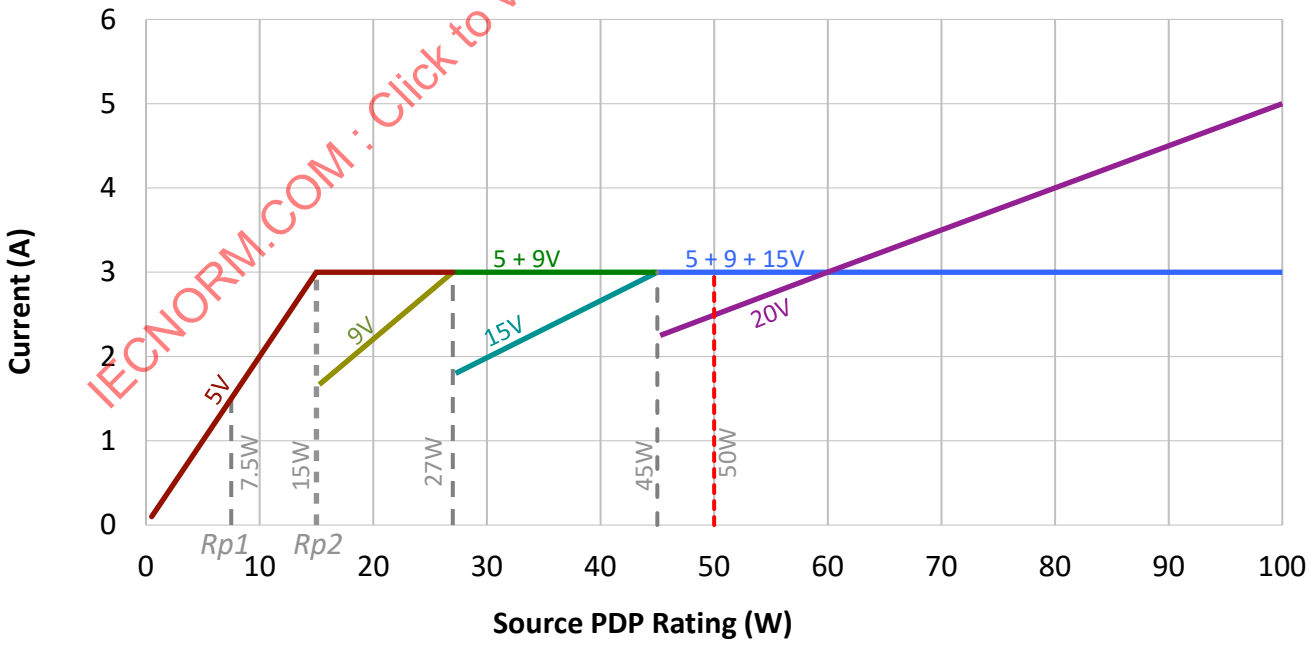


Table 10-3, Table 10-4, Table 10-5 and Table 10-6 show the Fixed Supply PDOs that ***Shall*** be supported for each of the ***Normative*** voltages defined in Table 10-2.

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	Current based on PDP **PDP Rating (x)** / **Current (A)** 0.5 ≤ x ≤ 15 / x ÷ 5 15 < x ≤ 25 / 3 ≤ A ≤ x ÷ 5 25 < x ≤ 100 / 3 ≤ A ≤ 5

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	Current based on PDP **PDP Rating (x)** / **Current (A)** 0.5 ≤ x ≤ 15 / PDO not required 15 < x ≤ 27 / x ÷ 9 27 < x ≤ 45 / 3 ≤ A ≤ x ÷ 9 45 < x ≤ 100 / 3 ≤ A ≤ 5

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	Current based on PDP **PDP Rating (x)** / **Current (A)** 0.5 ≤ x ≤ 27 / PDO not required 27 < x ≤ 45 / x ÷ 15 45 < x ≤ 75 / 3 ≤ A ≤ x ÷ 15 75 < x ≤ 100 / 3 ≤ A ≤ 5

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

Bit(s)	Description
B9…0	Current based on PDP **PDP Rating (x)** / **Current (A)** 0.5 ≤ x ≤ 45 / PDO not required 45 < x ≤100 / x ÷ 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

10.2.3.1 Optional Normative Fixed, Variable and Battery Supply

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. See Section 10.2 for the rules that Shall apply to Optional PDOs in order to be consistent with the declared PDP Rating and the Normative voltages and currents.

10.2.3.2 Optional Normative Programmable Power Supply

The voltages and currents a Programmable Power Supply with a PDP Rating of x Watts ***Shall*** support are as defined Table 10-7.

When ***Optional*** Programmable Power Supply APDOs are offered, the following requirements ***Shall*** apply:

- A Source that advertises ***Optional*** Programmable Power Supply APDOs ***Shall*** advertise the PDOs and APDOs shown in Table 10-7.
- A Source ***Shall*** advertise ***Optional*** Programmable Power Supply APDOs with Maximum Voltage and Minimum Voltages for nominal voltage as defined in Table 10-8.
- A Source that advertises Programmable Power Supply APDOs other than the ones listed in Table 10-8 ***Shall*** advertise additional APDO's with a maximum current of RoundDown (x/Max Voltage) to the nearest 50mA.
- In no case ***Shall*** a Source advertise a current that exceeds the attached cable's current rating.

Table 10-7 Programmable Power Supply PDOs and APDOs based on the PDP

PDP Rating (W)	5V fixed	9V fixed	15V fixed	20V fixed	5V Prog	9V Prog	15V Prog	20V Prog
x < 15W	PDP/5[4]	-	-	-	PDP/5[1]	-	-	-
15W	3A	-	-	-	3A	-	-	-
15 < x < 27W	3A[3]	PDP/9[4]	-	-	3A[2]	PDP/9[1]	-	-
27W	3A[3]	3A	-	-	-	3A	-	-
27 < x < 45W	3A[3]	3A[3]	PDP/15[4]	-	-	3A[2]	PDP/15[1]	-
45W	3A[3]	3A[3]	3A	-	-	-	3A	-
45 < x < 60W	3A[3]	3A[3]	3A[3]	PDP/20[4]	-	-	3A[2]	PDP/20[1]
60W	3A[3]	3A[3]	3A[3]	3A[3]	-	-	-	3A
60 < x < 100W	3A[3]	3A[3]	3A[3]	PDP/20[4]	-	-	-	PDP/20[2]
100W	3A[3]	3A[3]	3A[3]	5A	-	-	-	5A

Notes:
1. The PPS APDOs Maximum Current field ***Shall*** advertise RoundDown (PDP/Prog Voltage) to the nearest 50mA.
2. The PPS APDOs Maximum Current field ***Shall*** advertise at least 3A, but ***May*** advertise up to RoundDown(PDP/Prog voltage) to the nearest 50mA.
3. The Fixed PDOs Maximum Current field ***Shall*** advertise at least 3A, but ***May*** advertise up to RoundUp (PDP/voltage.) to the nearest 10mA.
4. The Fixed PDOs Maximum Current field ***Shall*** advertise either RoundDown (PDP/Voltage) or RoundUp (PDP/Voltage) to the nearest 10mA.

10.2.3.2.1 Programmable Power Supply Voltage Ranges

The Programmable Power Supply voltage ranges map to the Fixed Supply Voltages. For each Fixed Voltage there is a defined voltage range for the matching Programmable Power Supply APDO. Table 10-8 shows the Minimum and Maximum voltage for the Programmable Power Supply that corresponds to the Fixed nominal voltage.

Table 10-8 Programmable Power Supply Voltage Ranges

	Fixed Nominal Voltage			
	5V Prog	**9V Prog**	**15V Prog**	**20V Prog**
Maximum Voltage	5.9V	11V	16V	21V
Minimum Voltage	3.3V	3.3V	3.3V	3.3V

The voltage output at the Source's connector ***Shall*** be +/-5% for both the Maximum Voltage and the Minimum Voltage.

10.1.2.2 Examples of the use of the Programmable Power Supplies

The following examples illustrate what a power adapter that advertises a particular PDP Rating ***May*** offer:

1. PDP 15W
   - 5V @ 3A and 5V Prog @ 3A is the baseline
2. PDP 25W
   - 5V @ 3A, 9V @ 2.8A, 5V Prog @ 3A and 9V Prog @ 2.8A is the baseline
   - 5V @ 3A, 9V @ 2.8A, 5V Prog @ >3A up to 5A and 9V Prog @ 2.8A (with a 5A cable)
3. PDP 27W
   - 5V @ 3A, 9V @ 3A, 9V Prog @ 3A is the baseline
   - 5V @ 3A, 9V @ 3A, 5V Prog @ 3A and 9V Prog @ 3A can offer 5V Prog, but it is covered by the 9V Prog
   - 5V @ 3A, 9V @ 3A, 5V Prog @ >3A up to 5A and 9V Prog @ 3A (with a 5A cable)

4. PDP 36W
    - 5V @ 3A, 9V @ 3A, 15 @ 2.4A, 9V Prog @ 3 A and 15V Prog @ 2.4A is the baseline
    - 5V @ 3A, 9V @ 3A, 15 @ 2.4A, 5V Prog @ >3A up to 5A, 9V Prog @ >3A up to 4A and 15V Prog @ 2.4A (with a 5A cable)

The first example is a simple single output voltage supply. Both the Fixed and Programmable outputs supply 3A.

The second example illustrates that there are multiple ways to meet the requirements. The first sub-bullet is the power that the power rules require. The second sub-bullet illustrates that the power supply can offer more power at a particular voltage so long as it does not violate the power rules. In this case it offers 25W at both 5V and 9V.

The third example illustrates that there are multiple ways a 27W PDP Rated power adapter can be implemented and meet the power rules. The first sub-bullet shows that the 9V Prog @ 3A fully covers the 5V Prog @3A range so it is not necessary to advertise both. The second and third sub-bullets illustrate that the power adapter can advertise lower voltages at higher currents than required so long as the power does not exceed the PDP.

The fourth example illustrates as the PDP Rating goes higher there are more possible combinations that meet the power rules. Although there are multiple ways to meet the power rules, no more than a combination of seven PDO and APDOs can be offered.

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 higher PDP Rating 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 Sources with a PDP Rating of x W ***Shall***:

- Either operate or charge from Sources that have a PDP Rating ≥ x W.
- Either operate, charge or indicate a capability mismatch (see Section 6.4.2.3) from Sources that have a PDP Rating < x W and ≥ 0.5W.

A Sink optimized for a Source with ***Optional*** voltages and currents or power as described in Section 10.2.3 with a PDP Rating of x W ***Shall*** provide a similar user experience when powered from a Source with a PDP Rating of ≥ x W that supplies only the ***Normative*** voltages and currents as specified in Section 10.2.2.

The Operational Current/Power in the Sink Capabilities PDO for Sinks with an Operational PDP of x Watts ***Shall*** be as follows:

- Operational current for Fixed/Variable supply PDOs: RoundDown(x/Voltage) to the nearest 10mA.
- Operational power for Battery supply PDOs: ≤ x.

- Operational current for Programmable Power Supply APDOs as defined in Table 10-7: RoundDown (x/Prog Voltage) to the nearest 50mA.

Operational current for Programmable Power Supply APDOs not defined in Table 10-7 ***Shall*** be RoundDown (x/Max Voltage) to the nearest 50mA.

The Maximum Current/Power in the Sink RDO for Sinks with an Operational PDP of x Watts and Maximum PDP of y Watts ***Shall*** be as follows:

- Maximum current for Fixed/Variable Supply RDOs from Sinks without a Battery: RoundDown(x/Voltage) to the nearest 10mA.
- Maximum current for Fixed/Variable Supply RDOs from Sinks with a Battery: RoundDown(y/Voltage) to the nearest 10mA.
- Maximum power for Battery Supply RDOs from Sinks without a Battery: ≤ x.
- Maximum power for Battery Supply RDOs from Sinks with a Battery: ≤ y.
- Maximum current for PPS Supply RDOs from Source PDOs not defined in Table 10-7: RoundDown (x/Prog Voltage) to the nearest 50mA.
- Maximum current for PPS Supply RDOs from Source PDOs as defined in Table 10-7: RoundDown (y/Prog Voltage) to the nearest 50mA.

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 Rating they were designed to use.
- A Sink ***Shall Not*** advertise Variable Supply PDO maximum voltages and currents that exceed the PDP Rating they were designed to use.
- A Sink ***Shall Not*** advertise a Battery Supply PDO maximum allowable power that exceeds the PDP Rating they were designed to use.
- A Sink ***Shall Not*** advertise a PPS APDO maximum allowable power that exceeds the PDP Rating they were designed to use.

A. CRC calculation

A.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");

}
```

A.2 Table showing the full calculation over one Message

Function	Nibble	Symbol	Bits	CRC register transmitter	CRC register receiver	bit nr.
Preamble			0	FFFFFFFF	FFFFFFFF	1
			1	FFFFFFFF	FFFFFFFF	2
			0	FFFFFFFF	FFFFFFFF	3
			1	FFFFFFFF	FFFFFFFF	4
			0	FFFFFFFF	FFFFFFFF	5
			1	FFFFFFFF	FFFFFFFF	6
			0	FFFFFFFF	FFFFFFFF	7
			1	FFFFFFFF	FFFFFFFF	8
			0	FFFFFFFF	FFFFFFFF	9
			1	FFFFFFFF	FFFFFFFF	10
			0	FFFFFFFF	FFFFFFFF	11
			1	FFFFFFFF	FFFFFFFF	12
			0	FFFFFFFF	FFFFFFFF	13
			1	FFFFFFFF	FFFFFFFF	14
			0	FFFFFFFF	FFFFFFFF	15
			1	FFFFFFFF	FFFFFFFF	16
			0	FFFFFFFF	FFFFFFFF	17
			1	FFFFFFFF	FFFFFFFF	18
			0	FFFFFFFF	FFFFFFFF	19
			1	FFFFFFFF	FFFFFFFF	20
			0	FFFFFFFF	FFFFFFFF	21
			1	FFFFFFFF	FFFFFFFF	22
			0	FFFFFFFF	FFFFFFFF	23
			1	FFFFFFFF	FFFFFFFF	24
			0	FFFFFFFF	FFFFFFFF	25
			1	FFFFFFFF	FFFFFFFF	26
			0	FFFFFFFF	FFFFFFFF	27
			1	FFFFFFFF	FFFFFFFF	28
			0	FFFFFFFF	FFFFFFFF	29
			1	FFFFFFFF	FFFFFFFF	30
			0	FFFFFFFF	FFFFFFFF	31
			1	FFFFFFFF	FFFFFFFF	32
			0	FFFFFFFF	FFFFFFFF	33
			1	FFFFFFFF	FFFFFFFF	34
			0	FFFFFFFF	FFFFFFFF	35
			1	FFFFFFFF	FFFFFFFF	36
			0	FFFFFFFF	FFFFFFFF	37
			1	FFFFFFFF	FFFFFFFF	38
			0	FFFFFFFF	FFFFFFFF	39
			1	FFFFFFFF	FFFFFFFF	40
			0	FFFFFFFF	FFFFFFFF	41
			1	FFFFFFFF	FFFFFFFF	42
			0	FFFFFFFF	FFFFFFFF	43
			1	FFFFFFFF	FFFFFFFF	44
			0	FFFFFFFF	FFFFFFFF	45
			1	FFFFFFFF	FFFFFFFF	46
			0	FFFFFFFF	FFFFFFFF	47
			1	FFFFFFFF	FFFFFFFF	48
			0	FFFFFFFF	FFFFFFFF	49
			1	FFFFFFFF	FFFFFFFF	50
			0	FFFFFFFF	FFFFFFFF	51
			1	FFFFFFFF	FFFFFFFF	52
			0	FFFFFFFF	FFFFFFFF	53
			1	FFFFFFFF	FFFFFFFF	54
			0	FFFFFFFF	FFFFFFFF	55
			1	FFFFFFFF	FFFFFFFF	56
			0	FFFFFFFF	FFFFFFFF	57
			1	FFFFFFFF	FFFFFFFF	58
			0	FFFFFFFF	FFFFFFFF	59
			1	FFFFFFFF	FFFFFFFF	60
			0	FFFFFFFF	FFFFFFFF	61
			1	FFFFFFFF	FFFFFFFF	62
			0	FFFFFFFF	FFFFFFFF	63
			1	FFFFFFFF	FFFFFFFF	64
SOP			0	FFFFFFFF	FFFFFFFF	65
		Sync1 (#18)	0	FFFFFFFF	FFFFFFFF	66
			0	FFFFFFFF	FFFFFFFF	67
			1	FFFFFFFF	FFFFFFFF	68
			1	FFFFFFFF	FFFFFFFF	69
			0	FFFFFFFF	FFFFFFFF	70
		Sync1 (#18)	0	FFFFFFFF	FFFFFFFF	71
			0	FFFFFFFF	FFFFFFFF	72
			1	FFFFFFFF	FFFFFFFF	73
			1	FFFFFFFF	FFFFFFFF	74
			0	FFFFFFFF	FFFFFFFF	75
		Sync1 (#18)	0	FFFFFFFF	FFFFFFFF	76
			0	FFFFFFFF	FFFFFFFF	77
			1	FFFFFFFF	FFFFFFFF	78
			1	FFFFFFFF	FFFFFFFF	79
			1	FFFFFFFF	FFFFFFFF	80
		Sync2 (#11)	0	FFFFFFFF	FFFFFFFF	81
			0	FFFFFFFF	FFFFFFFF	82
			0	FFFFFFFF	FFFFFFFF	83
			1	FFFFFFFF	FFFFFFFF	84

Function	Nibble	Symbol	Bits	CRC register transmitter	CRC register receiver	bit nr.
GoodCRC Header #0101	#1	#09	1	FFFFFFFF	FFFFFFFF	85
			0	FFFFFFFE	FFFFFFFF	86
			0	FB3EE24B	FFFFFFFF	87
			1	F2BCD921	FFFFFFFF	88
	#0	#1E	0	E1B8AFF5	FFFFFFFF	89
			0	E1B8AFF5	FFFFFFFF	90
			1	C7B0425D	FFFFFFFE	91
			1	8BA1990D	FB3EE24B	92
			1	13822FAD	F2BCD921	93
	#1	#09	1	27045F5A	E1B8AFF5	94
			0	4AC9A303	C7B0425D	95
			0	95934606	8BA1990D	96
			1	2FE791BB	13822FAD	97
			0	5FCF2376	27045F5A	98
	#0	#1E	0	5FCF2376	27045F5A	99
			1	BF9E46EC	4AC9A303	100
			1	7BFD906F	95934606	101
			1	F7FB20DE	2FE791BB	102
			1	EB375C0B	5FCF2376	103
CRC-32 = swapped and inverted EB375C0B = 2FC51328	#8	#12	0	EB375C0B	5FCF2376	104
			1	EB375C0B	BF9E46EC	105
			0	EB375C0B	7BFD906F	106
			0	EB375C0B	F7FB20DE	107
			1	EB375C0B	EB375C0B	108
	#2	#14	0	EB375C0B	EB375C0B	109
			0	EB375C0B	D2AFA5A1	110
			1	EB375C0B	A19E56F5	111
			0	EB375C0B	47FDB05D	112
			1	EB375C0B	8B3A7D0D	113
	#3	#15	1	EB375C0B	8B3A7D0D	114
			0	EB375C0B	12B5E7AD	115
			1	EB375C0B	21AAD2ED	116
			0	EB375C0B	4355A5DA	117
			1	EB375C0B	86AB4BB4	118
	#1	#09	1	EB375C0B	86AB4BB4	119
			0	EB375C0B	0D569768	120
			0	EB375C0B	1E6C3367	121
			1	EB375C0B	3CD866CE	122
			0	EB375C0B	79B0CD9C	123
	#5	#0B	1	EB375C0B	79B0CD9C	124
			1	EB375C0B	F7A0868F	125
			0	EB375C0B	EB8010A9	126
			1	EB375C0B	D3C13CE5	127
			0	EB375C0B	A343647D	128
	#C	#1A	0	EB375C0B	A343647D	129
			1	EB375C0B	4686C8FA	130
			0	EB375C0B	8D0D91F4	131
			1	EB375C0B	1A1B23E8	132
			1	EB375C0B	343647D0	133
	#F	#1D	1	EB375C0B	343647D0	134
			0	EB375C0B	686C8FA0	135
			1	EB375C0B	D0D91F40	136
			1	EB375C0B	A1B23E80	137
			1	EB375C0B	43647D00	138
	#2	#14	0	EB375C0B	43647D00	139
			0	EB375C0B	8209E7B7	140
			1	EB375C0B	0413CF6E	141
			0	EB375C0B	0CE6836B	142
			1	EB375C0B	1D0C1B61	143
EOP		#0D	1	EB375C0B	1D0C1B61	144
			0	EB375C0B	3A1836C2	145
			1	EB375C0B	70F17033	146
			1	EB375C0B	E1E2E066	147
			0	EB375C0B	C704DD7B	148

Note: CRC transmitter is calculated over data bytes only, in casu marked nibbles, and calculation results are available one (bit-) clock later

Note: CRC receiver is calculated over data bytes and received CRC bytes, in casu marked nibbles, and calculation results are available five (bit-) clocks later

Fixed residual

B. 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.

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.

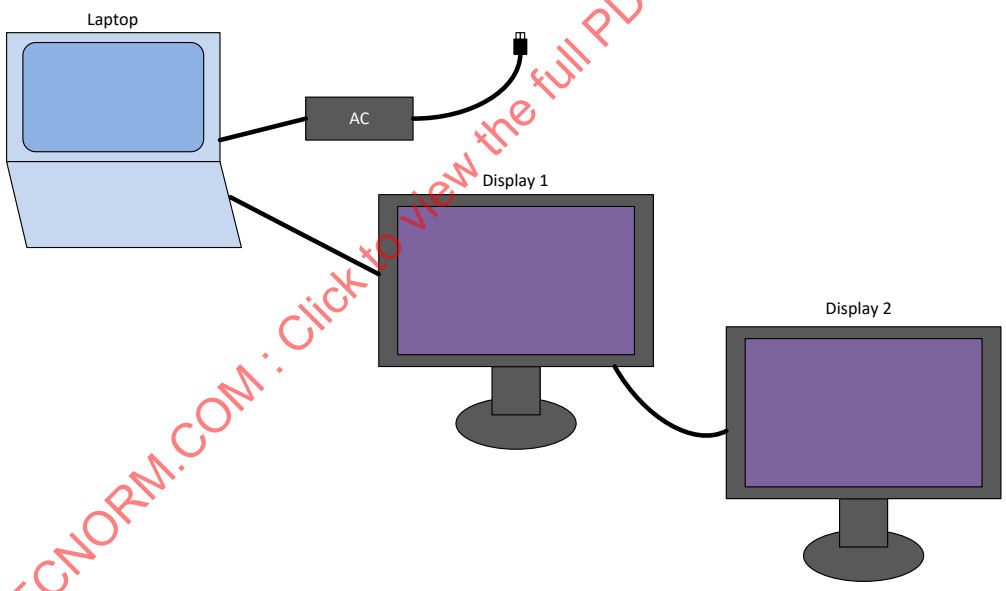
Although the Hubs in these illustrations support Power Delivery on both their UFPs and DFPs this is only one possible Hub implementation.

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.

B.1 External power is supplied downstream

Figure B-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 B-1 External power is supplied downstream

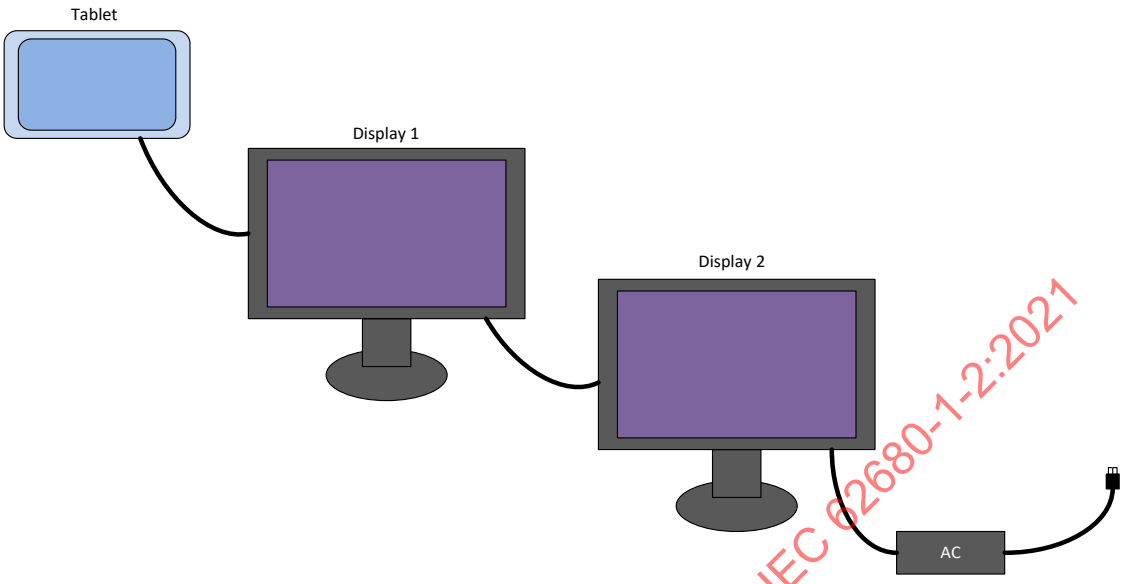
Step	Laptop	Display 1	Display 2	Device Policy Manager	Power (W)
Display 1					
1	Connected to wall supply	Detached	Detached		0
2	Display 1 Attached, V_{BUS} powered.	Attached, drawing 5V@150mA.	Detached		0.75
3	Set of Source Capabilities sent including: 5V@3A (15W), 9V@3A (27W), 15V@3A(45W) and 20V@3A (60W). The Unconstrained Power and USB suspend bits are set.	Source Capabilities received	Detached	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	Detached	Display 1 knows it needs 20v@1.5A (30W) for its own operation, evaluates the supplied capabilities and determines that this is available.	0.75
5	Sends Accept	Accept received	Detached	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.	Detached	Laptop evaluates the request, finds that it can meet this and so sends an accept.	30
Display 2					
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

Step	Laptop	Display 1	Display 2	Device Policy Manager	Power (W)
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
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

Step	Laptop	Display 1	Display 2	Device Policy Manager	Power (W)
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
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

B.2 External power is supplied upstream

Figure B-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 B-2 External power is supplied upstream

Step	Tablet	Display 1	Display 2	Device Policy Manager	Power (W)
Display 1 - Dead Battery					
1	Detached	Detached	Connected to the wall supply.		0
2		Attached to Display 2	Display 1 Attached		0
3		USB Type-C® Power drawn 5V@1.5A	USB Type-C Power advertised 5V@1.5A		0
4		Attached to Display 2, drawing 5V@1.5A (7.5W)	Providing 1-unit load to Display 1.		7.5

Step	Tablet	Display 1	Display 2	Device Policy Manager	Power (W)
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.	7.5
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.	7.5
7		Accept received	Sends Accept	Display 2 accepts the offer since it is within its capabilities.	7.5
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
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

Step	Tablet	Display 1	Display 2	Device Policy Manager	Power (W)
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
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

Step	Tablet	Display 1	Display 2	Device Policy Manager	Power (W)
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
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 ***Valid*** request has been made so Display 1 accepts the request.	31.5

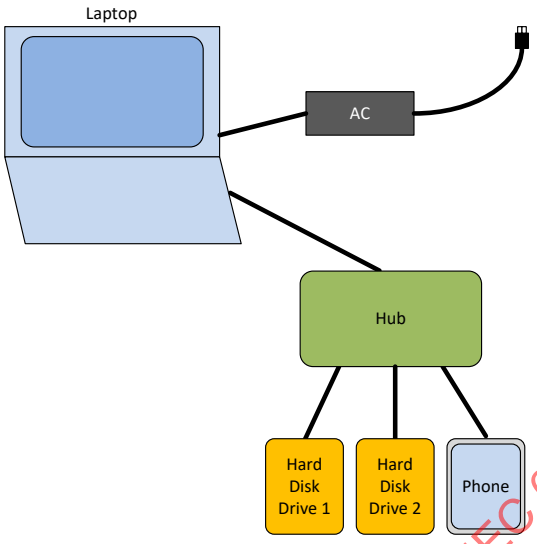
Step	Tablet	Display 1	Display 2	Device Policy Manager	Power (W)
29	PS_RDY received	PS_RDY received		Tablet indicates a capability mismatch to the user.	31.5
30	Get Sink Capabilities received.	Get Sink Capabilities sent		Due to the Capability Mismatch Flag Display 1 requests Sink Capabilities from the 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
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

Step	Tablet	Display 1	Display 2	Device Policy Manager	Power (W)
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
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

Step	Tablet	Display 1	Display 2	Device Policy Manager	Power (W)
42	PS_RDY received. Tablet starts drawing 5V@2.4A (12W) Display 1 and starts to charge.	Sends PS_RDY		Display 1 indicates its supply is ready to supply power.	42

B.3 Giving back power

Figure B-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 B-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 1-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
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

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
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
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

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
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
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

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
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
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

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
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
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

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
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
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

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
73		Request received from Hard Disk Drive 2	Once spun up Hard Disk Drive 2 requests 20V@0.25A operating current and 20V@0.5A maximum current.	Hard Disk Drive 2 no longer needs the additional power, so it gives back what it does not need.	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
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

Step	Laptop	Hub	Peripherals	Device Policy Manager	Hub Power (W)
79		PS_RDY sent to the Phone.	PS_RDY received by the Phone. The phone starts to draw 5V@2A but has to follow USB Suspend.	The Hub has allocated 0.75W for itself, 5W for Hard Disk Drive 2, 10W for the Phone (including 5W for the Power Reserve), and 5W for Hard Disk Drive 1 operation.	21.6

C. VDM Command Examples

C.1 Discover Identity Example

C.1.1 Discover Identity Command request

Table C-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 C-1 Discover Identity 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***	10b (Revision 3.0)
5…4	***Reserved***	0
3…0	***Message Type***	1111b (Vendor Defined Message)
VDM Header		
B31…16	Standard or Vendor ID (SVID)	0xFF00 (***PD SID***)
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	00b (Initiator)
B5	***Reserved***	0
B4…0	Command[1]	1 (***Discover Identity***)

C.1.2 Discover Identity Command response – Active Cable

Table C-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 C-2 Discover Identity Command response from Active Cable Responder Example

Bit(s)	Field	Value
Message Header		
15	***Reserved***	0
14…12	***Number of Data Objects***	5 (VDM Header + ID Header VDO + Cert Stat VDO + Product VDO + Cable VDO)
11…9	***MessageID***	0…7
8	***Cable Plug***	1
7…6	***Specification Revision***	10b (Revision 3.0)
5…4	***Reserved***	0
3…0	***Message Type***	1111b (Vendor Defined Message)
VDM Header		
B31…16	Standard or Vendor ID (SVID)	0xFF00 (***PD SID***)
B15	VDM Type	1 (Structured VDM)
B14…13	Structured VDM Version	01b (Version 2.0)
B12…11	***Reserved***	00b

Bit(s)	Field	Value
B10…8	Object Position	000b
B7…6	Command Type	01b (Responder ACK)
B5	***Reserved***	0
B4…0	Command	2 (***Discover Identity***)
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…0	32-bit unsigned integer	USB-IF assigned XID 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 VDO1 (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…21	VDO Version	010b (Version 1.2)
B20	***Reserved***	0
B19…18	Connector Type	10b (USB Type-C®)
B17	***Reserved***	0
B16…13	Cable Latency	0001b (<10ns (~1m))
B12…11	Cable Termination Type	11b (Both ends Active, V_{CONN} required)
B10…9	Maximum V_{BUS} Voltage	00b (20V)
B8	SBU Supported	0 (SBUs connections supported)
B7	SBU Type	0 (SBU is passive)
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	***Reserved***	0
Cable VDO2 (returned for Product Type "Active Cable")		
B31…24	Maximum Operating Temperature	70
B23…16	Shutdown Temperature	80
B15	***Reserved***	0
B14…12	U3 Power	010b (1-5mW)
B11	U3 to U0 transition mode	00b (U3 to U0 direct)
B10…8	***Reserved***	0
B7…6	USB 2.0 Hub Hops Consumed	2
B5	USB 2.0 Supported	0 (***[USB 2.0]*** supported)
B4	SuperSpeed Supported	0 (***[USB 3.2]*** SuperSpeed supported)
B3	SuperSpeed Lanes Supported	1b (Two lanes)
B2	***Reserved***	0
B1…0	SuperSpeed Signaling	01b (Gen 2)

C.1.3 Discover Identity Command response – Hub

Table C-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 C-3 Discover Identity Command response from Hub Responder Example

Bit(s)	Field	Value
Message Header		
15	***Reserved***	0
14…12	***Number of Data Objects***	4 (VDM Header + ID Header VDO + Cert Stat VDO + Product VDO)
11…9	***MessageID***	0…7
8	***Port Power Role***	0 or 1
7…6	***Specification Revision***	10b (Revision 3.0)
5…4	***Reserved***	0
3…0	***Message Type***	1111b (Vendor Defined Message)
VDM Header		
B31…16	Standard or Vendor ID (SVID)	0xFF00 (***PD SID***)
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 (***Discover Identity***)
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…0	32-bit unsigned integer	USB-IF assigned XID 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

C.2 Discover SVIDs Example

C.2.1 Discover SVIDs Command request

Table C-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 C-4 Discover SVIDs 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***	10b (Revision 3.0)
5…4	***Reserved***	0
3…0	***Message Type***	1111b (Vendor Defined Message)
VDM Header		
B31…16	Standard or Vendor ID (SVID)	0xFF00 (***PD SID***)
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	00b (Initiator)
B5	***Reserved***	0
B4…0	Command[1]	2 (***Discover SVIDs***)

C.2.1 Discover SVIDs Command response

Table C-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 C-5 Discover SVIDs Command response from Responder Example

Bit(s)	Field	Value
Message Header		
15	***Reserved***	0
14…12	***Number of Data Objects***	3 (VDM Header + 2*VDO)
11…9	***MessageID***	0…7
8	***Port Power Role***	0 or 1
7…6	***Specification Revision***	10b (Revision 3.0)
5…4	***Reserved***	0
3…0	***Message Type***	1111b (Vendor Defined Message)
VDM Header		
B31…16	Standard or Vendor ID (SVID)	0xFF00 (***PD SID***)
B15	VDM Type	1 (Structured VDM)
B14…13	Structured VDM Version	01b (Version 2.0)
B12…11	***Reserved***	00b (***Reserved***)
B10…8	Object Position	000b

Bit(s)	Field	Value
B7…6	Command Type	01b (Responder ACK)
B5	***Reserved***	0
B4…0	Command	2 (***Discover SVIDs***)
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

C.3 Discover Modes Example

C.3.1 Discover Modes Command request

Table C-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 C-6 Discover Modes 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***	10b (Revision 3.0)
5…4	***Reserved***	0
3…0	***Message Type***	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	***Reserved***	00b
B10…8	Object Position	000b
B7…6	Command Type	00b (Initiator)
B5	***Reserved***	0
B4…0	Command[1]	3 (***Discover Modes***)

C.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 C-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 C-7 Discover Modes Command response from Responder Example

Bit(s)	Field	Value
Message Header		
15	***Reserved***	0
14…12	***Number of Data Objects***	2 (VDM Header + 1 Mode VDO)
11…9	***MessageID***	0…7
8	***Port Power Role***	0 or 1
7…6	***Specification Revision***	10b (Revision 3.0)

Bit(s)	Field	Value
5…4	***Reserved***	0
3…0	Message Type	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	***Reserved***	00b
B10…8	Object Position	000b
B7…6	Command Type	01b (Responder ACK)
B5	***Reserved***	0
B4…0	Command	3 (***Discover Modes***)
Mode VDO		
B31…0	Mode 1	Standard or Vendor defined Mode value

C.4 Enter Mode Example

C.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 C-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 C-8 Enter Mode 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***	10b (Revision 3.0)
5…4	***Reserved***	0
3…0	***Message Type***	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	***Reserved***	00b
B10…8	Object Position	001b (a one in this field indicates a request to enter the first Mode in list returned by Discover Modes)
B7…6	Command Type	00b (Initiator)
B5	***Reserved***	0
B4…0	Command	4 (***Enter Mode***)

C.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 C-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 C-9 Enter Mode Command response from Responder 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***	10b (Revision 3.0)

Bit(s)	Field	Value
5…4	***Reserved***	0
3…0	***Message Type***	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	***Reserved***	00b
B10…8	Object Position	001b (offset of the Mode entered)
B7…6	Command Type	01b (Responder ACK)
B5	***Reserved***	0
B4…0	Command	4 (***Enter Mode***)

C.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 C-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 C-10 Enter Mode 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***	10b (Revision 3.0)
5…4	***Reserved***	0
3…0	***Message Type***	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	***Reserved***	00b
B10…8	Object Position	001b (a one in this field indicates a request to enter the first Mode in list returned by Discover Modes)
B7…6	Command Type	00b (Initiator)
B5	***Reserved***	0
B4…0	Command	4 (***Enter Mode***)
Including *Optional* **Mode specific VDO**		
B31…0	Mode specific	

C.5 Exit Mode Example

C.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 C-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 C-11 Exit Mode 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***	10b (Revision 3.0)
5…4	***Reserved***	0
3…0	***Message Type***	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	***Reserved***	00b
B10…8	Object Position	001b (identifies the previously entered Mode by its Object Position that is to be exited)
B7…6	Command Type	00b (Initiator)
B5	***Reserved***	0
B4…0	Command	5 (***Exit Mode***)

C.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 C-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 C-12 Exit Mode Command response from Responder 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***	10b (Revision 3.0)
5…4	***Reserved***	0

Bit(s)	Field	Value
3…0	***Message Type***	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 (***Exit Mode***)

C.6 Attention Example

C.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 C-11 shows the contents of the key fields in the Message Header and VDM header for an Initiator sending an ***Attention*** Command request.

Table C-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***	10b (Revision 3.0)
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***)

C.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 C-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 C-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***	10b (Revision 3.0)
5…4	***Reserved***	0
3…0	***Message Type***	1111b (Vendor Defined Message)

Bit(s)	Field	Value
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***)
Including *Optional* **Mode specific VDO**		
B31…0	Mode specific	

D. BMC Receiver Design Examples

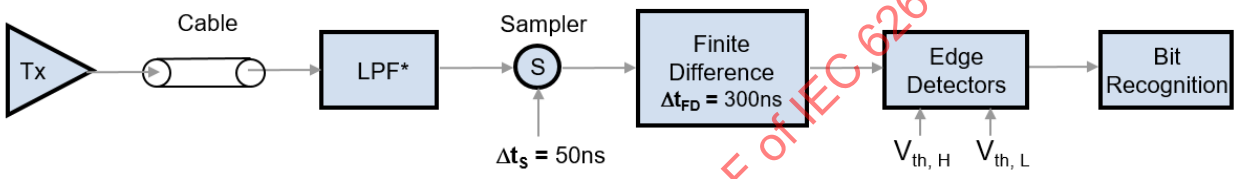
The BMC signal is DC-coupled so that the voltage level is affected by the ground IR drop. The DC offset of the BMC signal at Power Source and Power Sink are in the opposite directions. When the V_{BUS} current is increased from 0A, the BMC signal waveform shifts downward at Power Sink and shifts upward at Power Source. This section introduces two sample BMC receiver circuit implementations, which are immune from DC offset and high current load step. They can be used in Power Source, Power Sink and inside cables.

D.1 Finite Difference Scheme

D.1.1 Sample Circuitry

The sample Finite Difference BMC receiver shown in Figure D-1 consists of the Rx bandwidth limiting filter with the time constant tRxFilter, a sampler with the sampling step Δt_S, 50ns, a Finite Difference Calculator which calculates the voltage difference between the time interval of Δt_{FD}, 300ns, an edge detector controlled by two voltage thresholds, $V_{th,H}$ and $V_{th,L}$ and a logic block for bit recognition.

Figure D-1 Circuit Block of BMC Finite Difference Receiver



D.1.2 Theory

This section describes the fundamental theory of Finite Difference Scheme to recover the received BMC signal with the input and output signal waveforms of the circuit blocks shown in Figure D-1. To illustrate the robustness of the implementation, the V_{BUS} current load step rate is intentionally increased to 2A/µs at the sink load. In Figure D-2(a), the red curve represents the V_{BUS} current measured at the Power Sink when the current is increased at 9 µs from 0A to 5A and the blue dash curve represents the V_{BUS} current measured at the USB Type-C® connector of the power sink. In this example, the peak current overshoot with larger load step rate is increased to 518 mA which exceeds iOvershoot. Figure D-2(b) shows the total BMC noise at Power Sink, coupled from V_{BUS} and D+/D- through the worst ***[USB Type-C 2.0]*** compliant cable, after the Rx bandwidth limiting filter with the time constant tRxFilter is applied. The noise can be decomposed into 3 components. The first is the DC offset, $I_{VBUS}(t)*R_{GND}$, while I_{VBUS} is the V_{BUS} current and R_{GND} is the ground DC resistance of the cable. The offset is negative in Power Sink and positive at Power Source. The second noise component is the inductive V_{BUS} noise, M*d $I_{VBUS}(t)/dt$, while M is the mutual inductance between the V_{BUS} and CC wires in the cable and d $I_{VBUS}(t)/dt$ is the load step rate. The third component is ***[USB 2.0]*** Full Speed SE0 coupling noise which would normally occur randomly but was assumed to occur periodically in the simulation to account for the crosstalk in any phase between the BMC and ***[USB 2.0]*** signals. In Figure D-3, the blue dash curve represents the BMC signal when there is no V_{BUS} current and the red solid curve represents the BMC signal affected by the V_{BUS} coupling noise shown in Figure D-2(b). The green solid curve is the sample ***[USB 2.0]*** noise, after the Rx bandwidth limiting filter with the time constant tRxFilter is applied.

Figure D-2 BMC AC and DC noise from VBUS at Power Sink

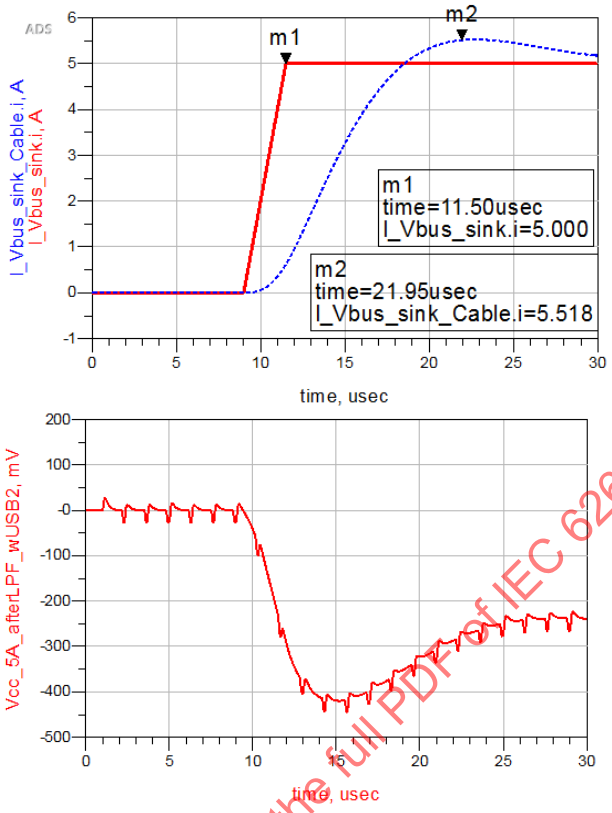
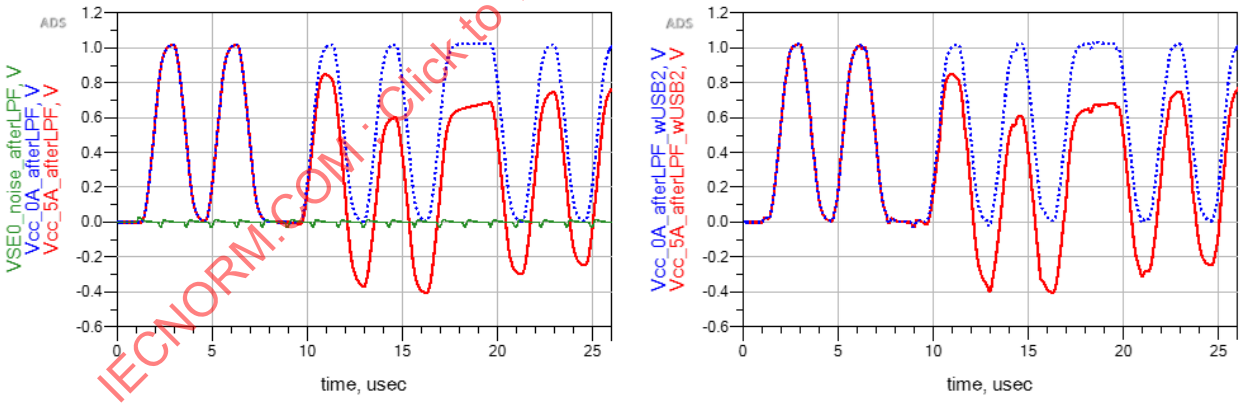


Figure D-3 Sample BMC Signals (a) without *[USB 2.0]* SE0 Noise (b) with *[USB 2.0]* SE0 Noise

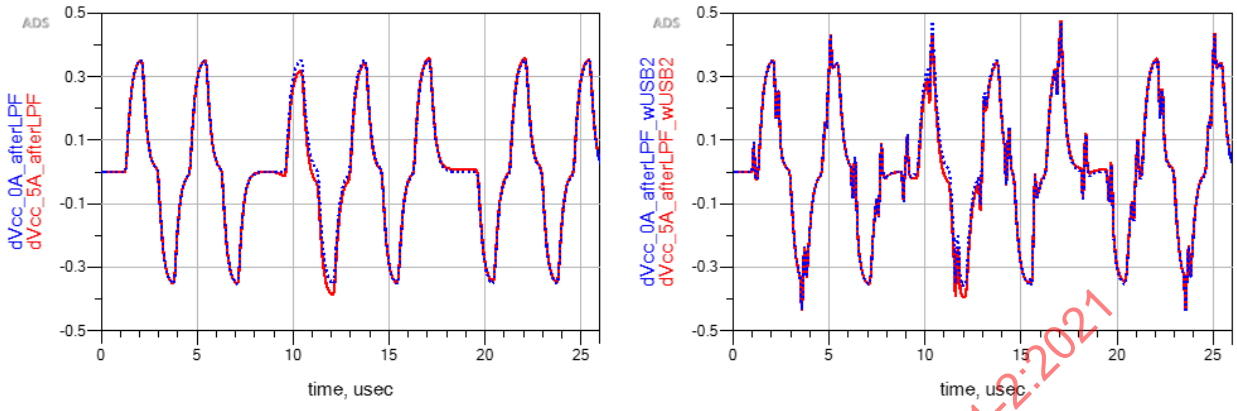


(a) (b)

The BMC signals shown in Figure D-3 are sampled every 50ns and the scaled derivative waveforms, Vcc(t) - Vcc (t - 50ns), without and with ***[USB 2.0]*** noise are shown in Figure D-4(a) and D-4(b), respectively. In Figure D-4(a), if there is no ***[USB 2.0]*** noise, the derivative waveform just changes slightly before and after the V_{BUS} current transition. That means, the slope of the BMC waveform is not sensitive to the DC offset and is very useful to be used to design a robust receiver against a large DC offset. However, the derivative waveforms with ***[USB 2.0]*** noise have large perturbation as shown in Figure D-4(b).

Figure D-4 Scaled BMC Signal Derivative with 50ns Sampling Rate

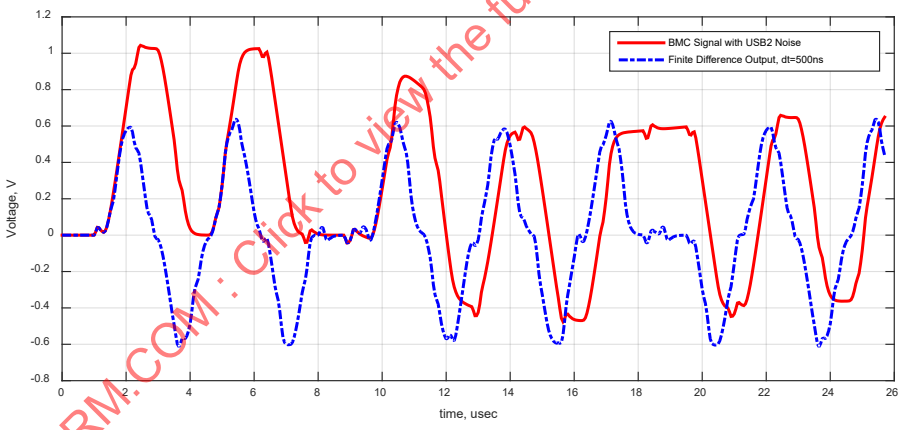
(a) without *[USB 2.0]* Noise (b) with *[USB 2.0]* Noise



(a) (b)

To remove the high frequency content of the ***[USB 2.0]*** noise, Finite Difference technique with the proper time interval is applied to the BMC waveform with ***[USB 2.0]*** noise in Figure D-3(b). Using Backward Finite Difference Calculator, ΔVcc = Vcc (t) - Vcc(t-Δt), Figure D-5 shows the Finite Difference Output while Δt = 500ns. The larger the time interval Δt is, the larger the peak-to-peak magnitude of the Finite Difference Output will be. However, the time interval is bounded by the rise time of the BMC signal so that 300ns to 500ns is a good range of the time interval.

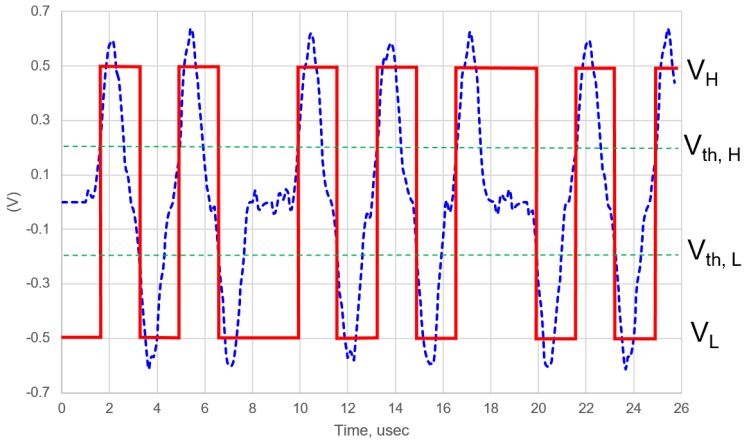
Figure D-5 BMC Signal and Finite Difference Output with Various Time Steps



D.1.3 Data Recovery

The edge detection is followed by the Finite Difference Calculation. At the input of the edge detector, if the voltage is larger than $V_{th,H}$ at the rising edge, the output will become high voltage level, V_H, if the voltage is smaller than $V_{th,L}$ at the falling edge, the output will become low voltage level, V_L. In this example, $V_{th,H}$ and $V_{th,L}$ are 0.2V and -0.2V, respectively. The solid curve in Figure D-6 represents the output of the edge detector, where V_H is 0.5V and V_L is -0.5V.

Figure D-6 Output of Finite Difference in dash line and Edge Detector in solid line



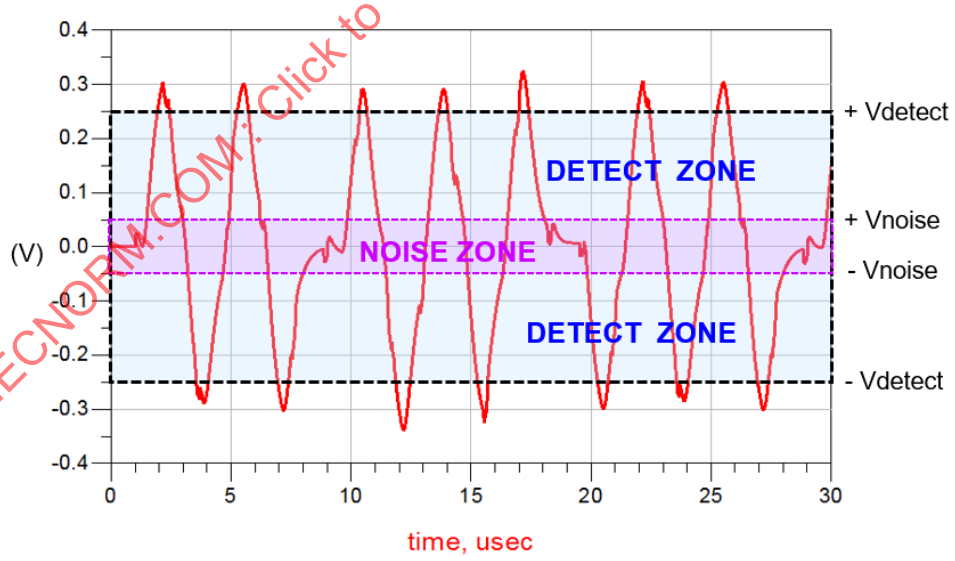
The duty cycle of the output signal from the edge detector varies depending on the thresholds, $V_{th, H}$ and $V_{th, L}$, as well as jitter and noise from silicon and channel. The techniques such as integrating receiver can be used to recover the BMC signal.

D.1.4 Noise Zone and Detection Zone

Figure D-7 shows the output of Finite Difference when the time interval of Finite Difference is set to 300ns. The noise Zone is defined in between +Vnoise and –Vnoise, in which the noise glitches occur. The detect zone is defined in between +Vdetect and –Vdetect, excluding the noise zone. The thresholds of the edge detectors, $V_{th, H}$ and $V_{th, L}$, must be properly set within the detect zone so that the data can be recovered successfully.

In this example, Vdetect is 250mV and Vnoise is 50mV. It is highly recommended that the product implemented with the similar techniques indicates the performance with the range of Vnoise and Vdetect in the electrical specification.

Figure D-7 Noise Zone and Detect Zone of BMC Receiver



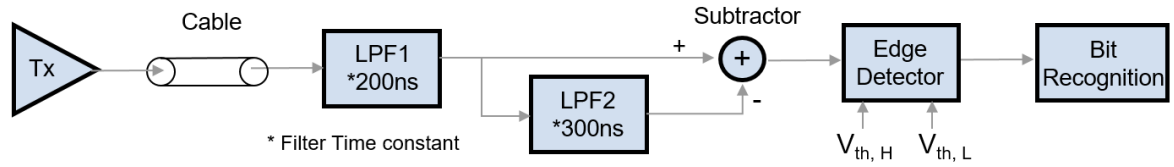
D.2 Subtraction Scheme

D.2.1 Sample Circuitry

The sample Subtraction BMC receiver shown in Figure D-8 consists of the two Low Pass Filters (LPF1 and LPF2), a Subtractor, an Edge Detector and a logic block for bit recognition. The time constant of the first

and second LPF are 200ns and 300ns, respectively. The Subtractor subtracts the LPF1 output from the LPF2 output. The Edge Detector controlled by two voltage thresholds, $V_{th,H}$ and $V_{th,L}$ to recover the data.

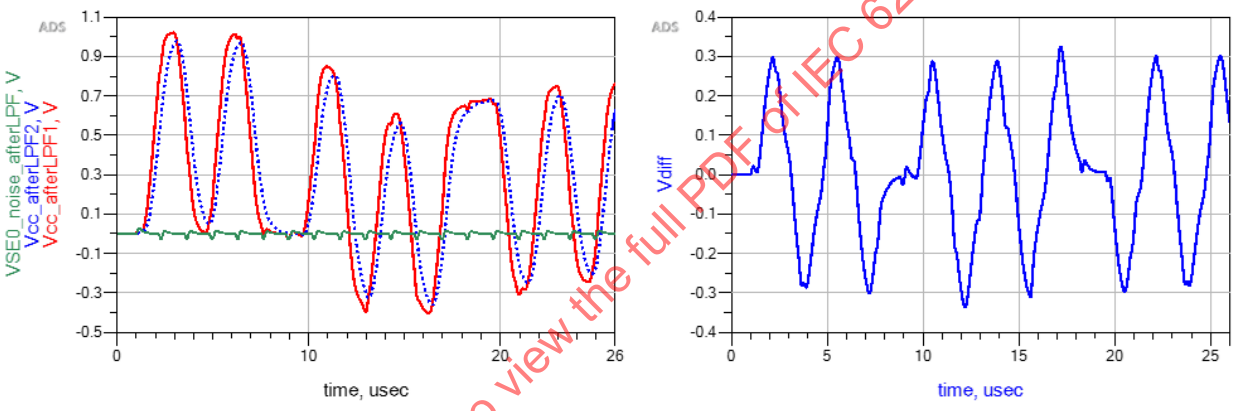
Figure D-8 Circuit Block of BMC Subtraction Receiver



D.2.2 Output of Each Circuit Block

Figure D-9(a) shows the output of LPF1 as the red solid line and LPF2 as the blue dash line as well as the ***[USB 2.0]*** noise in green solid line. Figure D-9(b) shows the voltage difference between the two output filters, Vdiff = Vcc_afterLPF1 – Vcc_afterLPF2. The Vdiff waveform looks very similar to the Finite Difference output waveform shown in Figure D-6 so that the data recovery method through the edge detector is the same as described in Section D.1.3.

Figure D-9 (a) Output of LPF1 and LPF2 (b) Subtraction of LPF1 and LPF2 Output



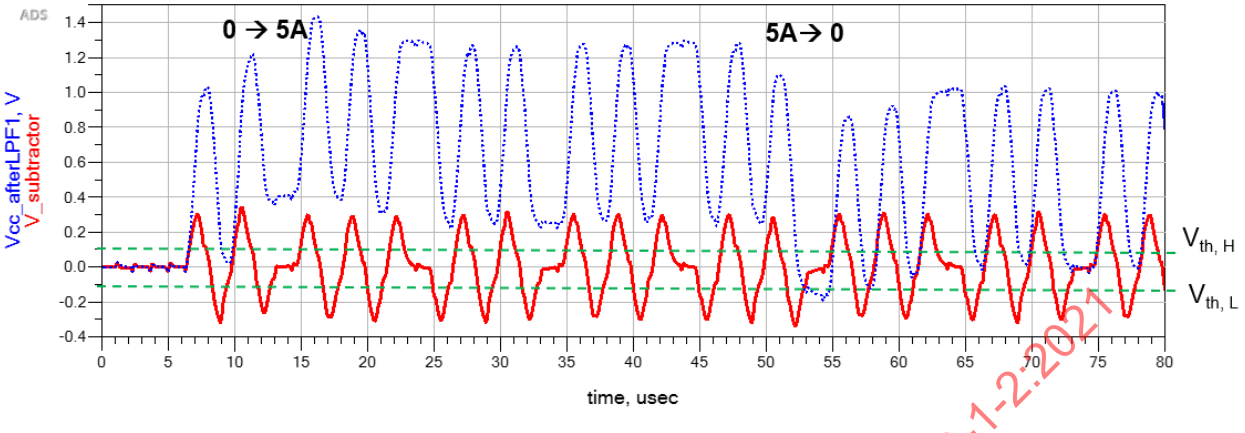
(a) **(b)**

D.2.3 Subtractor Output at Power Source and Power Sink

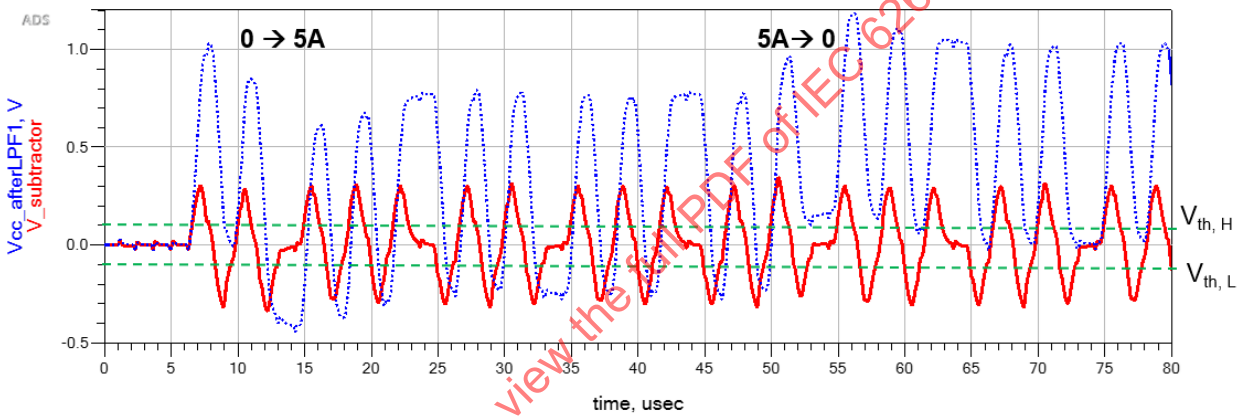
The following figures shows the example when the VBUS current increases from 0A to 5A and then decreases to 0A with high load step rate. The output of the LPF1 and the Subtractor at Power Source and Power Sink are shown in Figure D-10 (a) and (b), respectively. Although the BMC signals at Power Source and Power Sink shift toward the opposite direction, the Subtractor outputs at Power Source and Power Sink are almost identical disregard of the opposite direction of the DC offset.

Figure D-10 Output of the BMC LPF1 in blue dash curve and the Subtractor in red solid curve

(a) at Power Source (b) at Power Sink



(a)



(b)

D.2.4 Noise Zone and Detection Zone

The zone definition is the same as defined in Section D.1.7. The sizes of the noise zone and detection zone of the Subtraction Scheme are dependent on the filter time constant. When the time constant of the first and second LPF are 200ns and 300ns, respectively, Vdetect is 250mV and Vnoise is 50mV. It is highly recommended that the product implemented with the similar techniques indicates the performance with the range of Vnoise and Vdetect in the electrical specification.

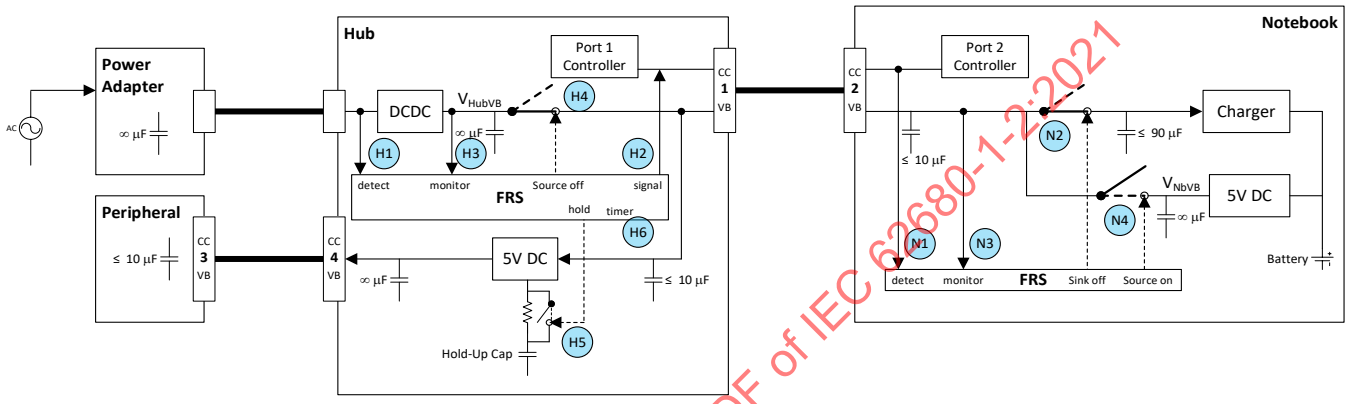
E. FRS System Level Example

E.1 Overview

Appendix E is intended to clarify Fast Role Swap (FRS) functionality at the system level through the use of an example implementation.

The following is an example of a Hub and Notebook implementation that supports Fast Role Swap (see Figure 7-14). It is not the only possible Hub or Notebook architecture. However, it is intended to provide an example system whose functionality is used here to illustrate how Fast Role Swap works.

Figure E-1 Example FRS Capable System



This appendix describes two cases that cover a variety of behaviors that may be seen in practice.

- *Slow V_{BUS} Discharge* where VBUS between the Hub and the Notebook takes more than 15ms (***tFRSwapInit***) to discharge below 5.5 V (***vSafe5V*** (max)). In this case the ***FR_Swap*** Message is sent by the Notebook while V_{BUS} is still greater than ***vSafe5V*** (max). See Figure E-2.
- *Fast V_{BUS} Discharge* where V_{BUS} between the Hub and the Notebook discharges very quickly, perhaps before the FRS Signaling is even complete. See Figure E-3.

However, neither the Hub nor the Notebook can anticipate how quickly V_{BUS} will discharge until the Power Adapter is disconnected from AC mains power or it is unplugged from the Hub.

The FRS signal is the momentary low driven by the Hub on the CC wire which is detected by the Notebook.

Figure E-2 and Figure E-3 show the voltage seen on V_{BUS} in relationship to the FRS Signaling. They also show the transition between when the Hub stops supplying V_{BUS} and when the Notebook starts supplying V_{BUS}.

Figure E-2 Slow V_{BUS} Discharge

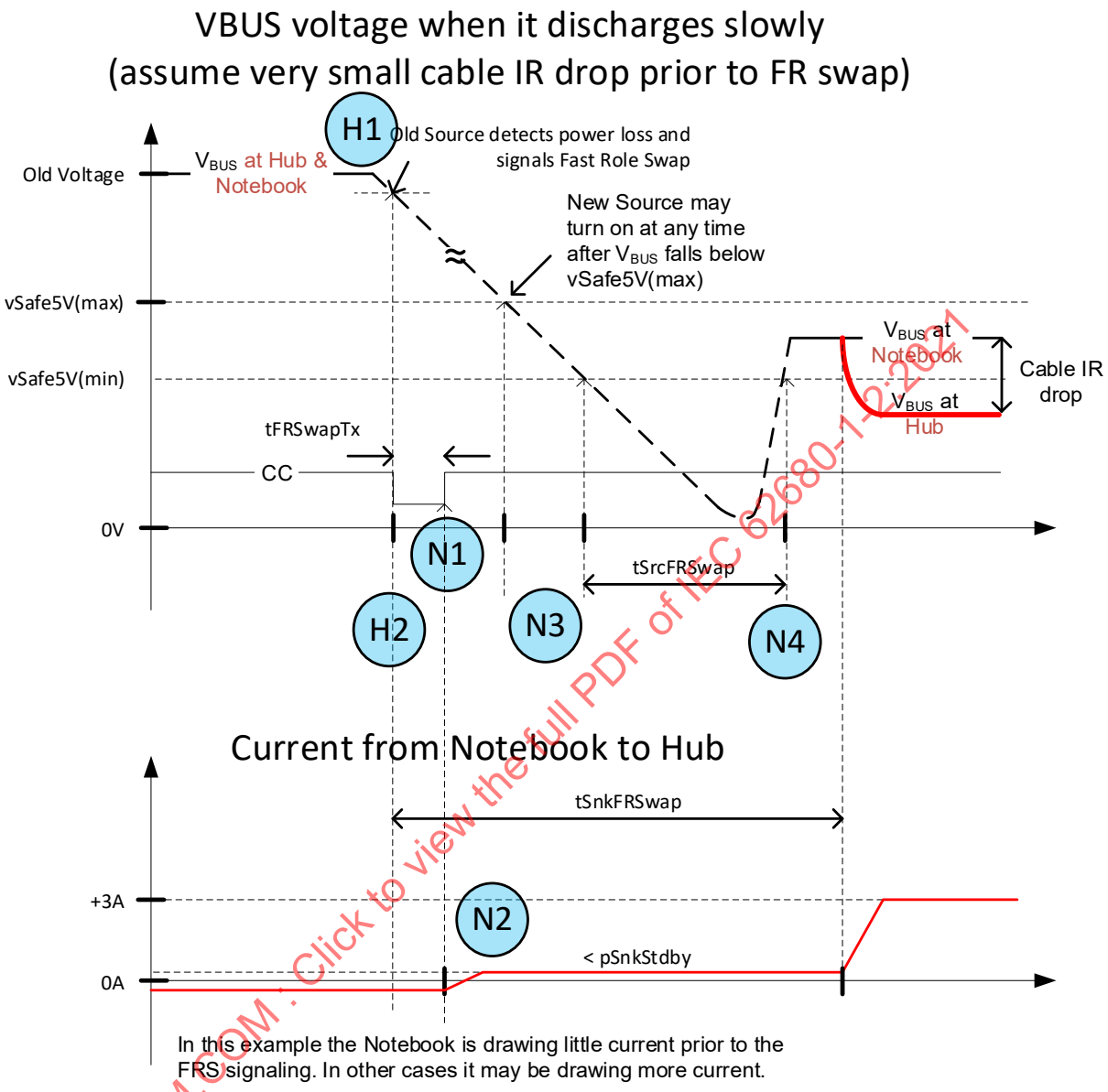
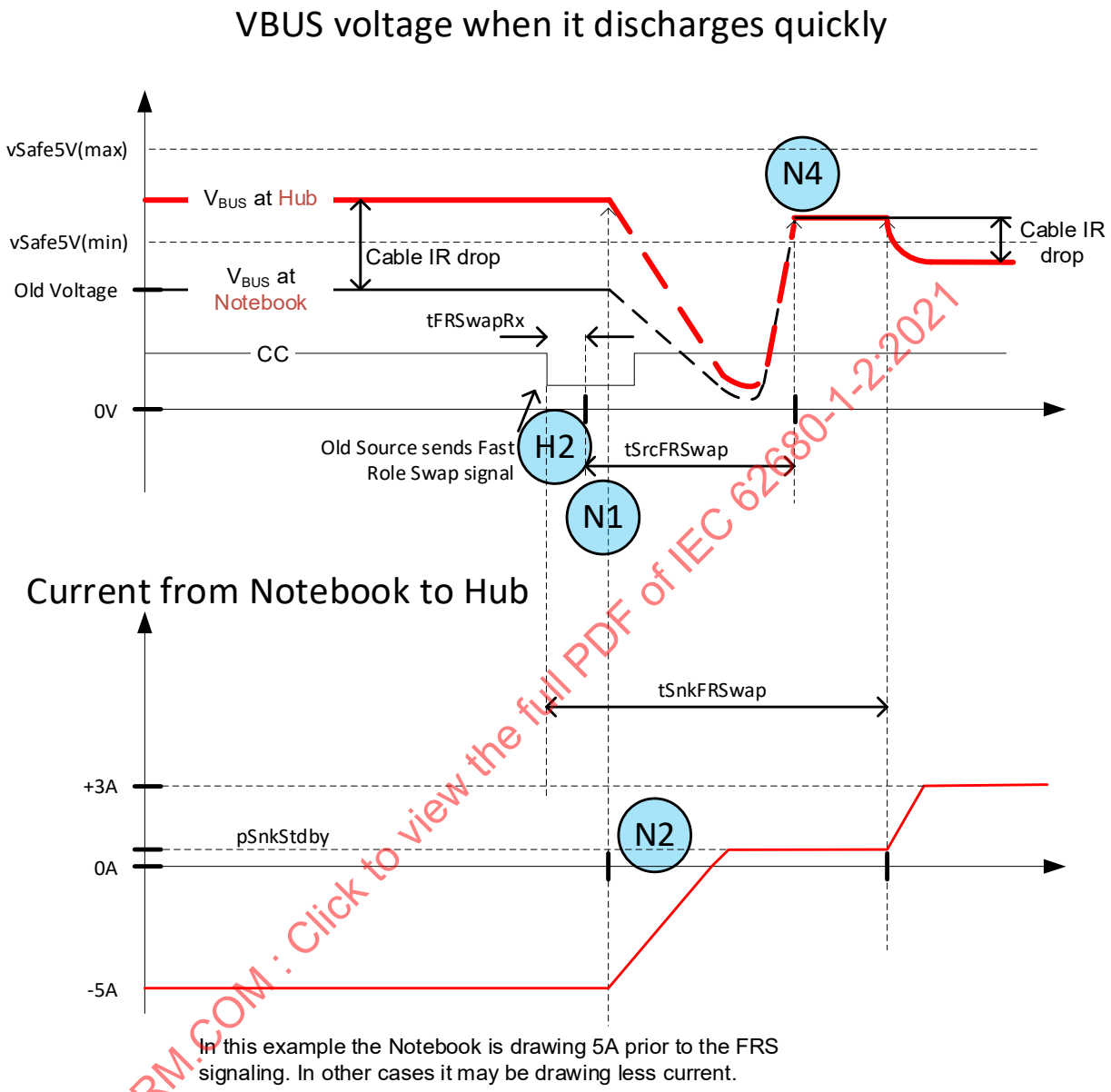


Figure E-3 Fast V_{BUS} Discharge



E.2 FRS Initial Setup

Before a Fast Role Swap can occur, some initial setup steps are required. They require the Notebook to discover whether Fast Role Swap is supported by the Hub, the amount of current the Hub requires after a Fast Role Swap, and whether the Notebook is able and willing to provide that amount. They also ensure that the Notebook supplies VCONN before, during and after an FRS. Table E-1 and Table E-2 below show two typical sequences that may be used to prepare a Notebook to support Fast Role Swap.

Table E-1: Sequence Table for setup of a Fast Role Swap (Hub connected to Power Adapter first)

Step #	Hub	Notebook
1	Hub connected to Power Adapter	
2	Hub is connected to Notebook.	
3		Notebook sources 5 V to VBUS (*vSafe5V*). Notebook sources 5 V to VCONN

4		Notebook reads the cable to check its current carrying capability and/or if it is an active cable requiring VCONN.
5		Notebook sends a Capabilities message
6	Hub sends a Request message	
7	Hub and Notebook establish an Explicit contract with Hub as sink.	
8		Notebook sends a ***Get_Source_Cap*** Message to determine how much power the Hub can provide.
9	Hub sends a ***Source_Capabilities*** Message with the Dual-Role Power bit set, and Unconstrained Power bit set, and Maximum Current > 0.	
10		Since the Hub can supply power the Notebook sends a ***PR_Swap*** Message
11	Hub sends an ***Accept*** message and starts supplying VBUS	
12		Notebook sends a ***Get_Sink_Cap*** Message to determine the current required by the Hub to support an FRS. If the Hub does not support FRS or the Notebook cannot supply the required current, the Notebook ignores any FRS signals it may see.
13	If the Hub can supply more than 3A, it initiates a VCONN swap to make to make itself the VCONN source and reads the cable to check its current carrying capability.	
14	Hub sends a ***Sink_Capabilities*** message	
15		Notebook sends a ***Request*** message
16	Hub and Notebook establish an Explicit contract with Hub as source.	
17		If the Notebook has detected that it is connected via an active cable (or one that supports alternate modes) and/or that it can support an FRS, it initiates a VCONN swap to make itself the VCONN source. This removes a requirement that the Hub to hold up VCONN during the FRS.
18	Normal PD Power traffic flow	
19	The Hub and Notebook are now ready to do a Fast Role Swap in case the Power Adapter gets removed.	

Table E-2 Sequence Table for setup of a Fast Role Swap (Hub connected to Notebook before Power Adapter)

Step #	Hub	Notebook
0	Hub is connected to Notebook.	
1		Notebook sources 5 V to VBUS (***vSafe5V***). Notebook sources 5 V to VCONN
2		Notebook reads the cable to check its current carrying capability and/or if it is an active cable requiring VCONN.
3		Notebook sends ***Source_Capabilities*** message
4	Hub sends ***Request*** Message	
5	Hub and Notebook establish an Explicit contract with Hub as sink.	
6		Notebook sends a ***Get_Source_Cap*** Message to determine how much power the Hub can provide
7	Hub sends a ***Source_Capabilities*** Message with the Dual-Role Power bit set, and Unconstrained Power bit cleared, and Maximum Current = 0.	

8		Since the Hub cannot supply power the Notebook does not send a ***PR_Swap*** Message
9	The Power Adapter is connected to the Hub	
10	If the Hub can source more than 3A, it initiates a VCONN swap to become the VCONN source.	
11	Hub reads the e-marker to determine the cable's current carrying capability.	
12	Hub initiates a Power Role Swap to become the Source	
13	Hub sends a ***Source_Capabilities*** Message with the Unconstrained Power bit set and Maximum Current > 0.	
14	Hub and Notebook establish an Explicit contract with Hub as source.	
15		Notebook sends a ***Get_Sink_Cap*** Message to determine the current required by the Hub to support an FRS. If the Hub does not support FRS or the Notebook cannot supply the required current, the Notebook ignores any FRS signals it may see.
16		If the Notebook has detected that it is connected via an active cable (or one that supports alternate modes) and/or that it can support an FRS, it initiates a VCONN swap to make itself the VCONN source. This removes a requirement that the Hub also hold up VCONN during the FRS.
17	The Hub and Notebook are now ready to do a Fast Role Swap in case the Power Adapter gets removed.	

E.3 FRS Process

After the initial setup is completed and the Notebook has determined both that the Hub may request FRS and that the Notebook is able and willing to supply the requested current, the system is ready to support FRS. This section describes the sequence of events that take place during a Fast Role Swap. The following figures and tables do not cover the actions of the Device Policy Manager or the Policy Engine. Those actions occur orthogonally to the electrical events shown in this appendix. However, the diagrams do indicate the inputs/outputs where the DPM and Policy Engine interact with the electrical events:

- The Notebook sends the ***FR_Swap*** Message to initiate the FRS message sequence (see Figure 7-34) within 15ms after the Notebook detects the FRS signal on CC.
- The Notebook sends the final ***PS_RDY*** Message in the FRS message sequence only after it is sourcing V_{BUS}.

Figure E-4 Sequence Diagram for slow VBUS discharge (it discharges after FR_Swap message is sent)

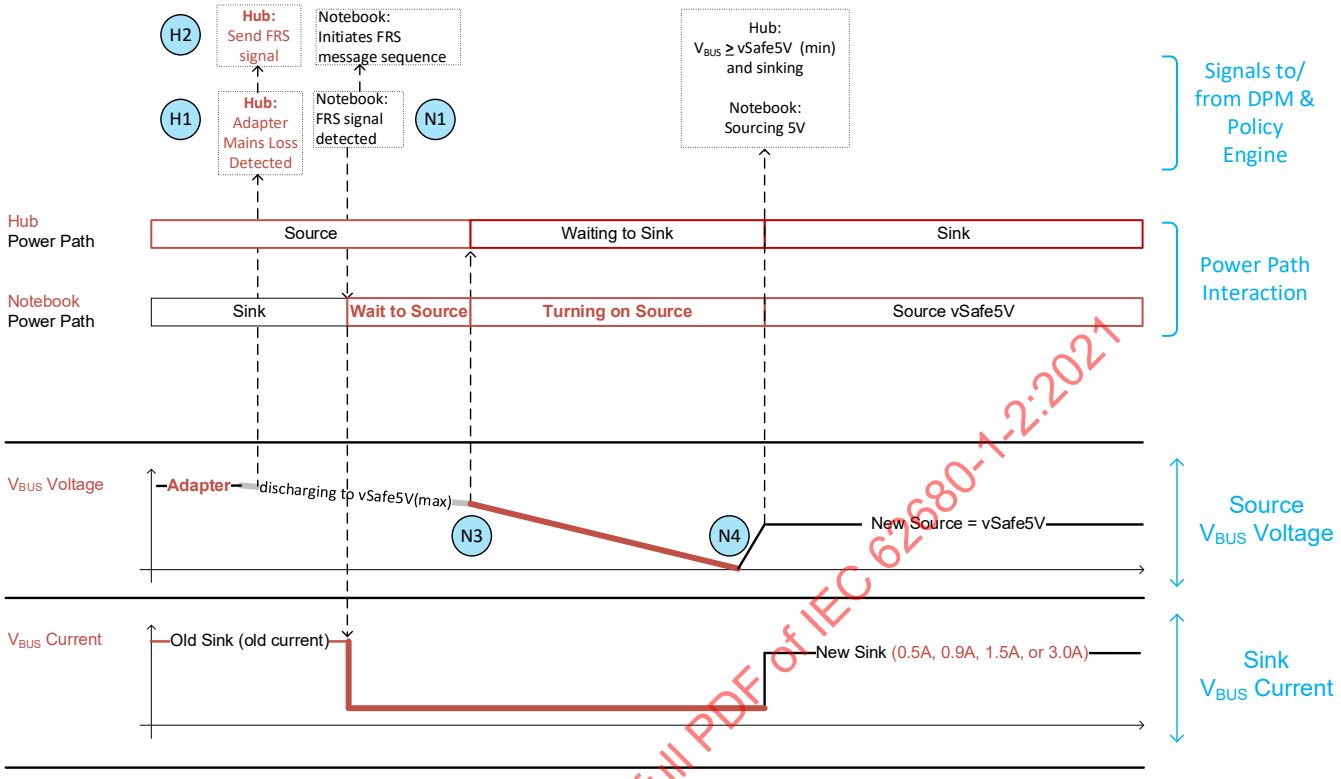


Table E-3 Sequence Table for slow Vbus discharge (it discharges after FR_Swap message is sent)

Step #	Hub	Notebook
1	The Power Adapter's AC mains power is lost.	
2	Hub detects the Power Adapter disconnect (H1) as quickly as possible.	
3	Hub sends FRS signal on CC (H2) and starts monitoring V_{HubVB} (H3). Hub also starts a ***tSnkFRSwap timer***.	
4		Notebook detects FRS signal on CC (N1) that triggers sending of the FR_Swap message. This may happen at any point in the following steps so long as it is within 15 ms (***tFRSwapInit***).
5		Notebook opens the sinking switch (N2), as quickly as possible to minimize power drained from hub after FRS signal.
6		Notebook begins monitoring V_{BUS} (N3) to know when to turn the Notebook into a Source.
7	Hub opens the sourcing switch (H4) while V_{HubVB} > 5.5V (after the FRS signal is sent). However the sourcing switch (H4) should be kept closed until V_{HubVB} is as close to 5.5V as possible. It is important for the Hub to open its sourcing switch (H4) before the Notebook's sourcing switch (N4) gets closed to minimize inrush current.	
8	Hub closes the switch (H5) to use the hold-up capacitor to supply V_{BUS} to the peripheral(s). Systems with a holding cap permanently in place do not need the switch (H5). Hub does not draw more than pSnkStby from V_{BUS}, until the ***tSnkFRSwap*** timer expires.	

9		Notebook detects V_{BUS} < V_{NbVB} (N1) before closing the sourcing switch (N4) when V_{NbVB} is as close as possible to 5.5V. This minimizes the time when V_{BUS} is not sourced.
10		Notebook closes sourcing switch (N4). When this occurs the Hub's input capacitance on V_{BUS} will be less than 10µF (***cSnkBulk***).
11	Hub's ***tSnkFRSwap*** timer expires (H6).	
12	Hub draws up to the current it advertised in the Fast Role Swap field of its ***Sink_Capabilities*** Message.	
13	Hubs with (H5) will open (H5) and remove the Hold-Up capacitor.	

Figure E-5: Sequence for V_{BUS} discharges quickly (before FR_Swap message is sent) after adapter disconnected.

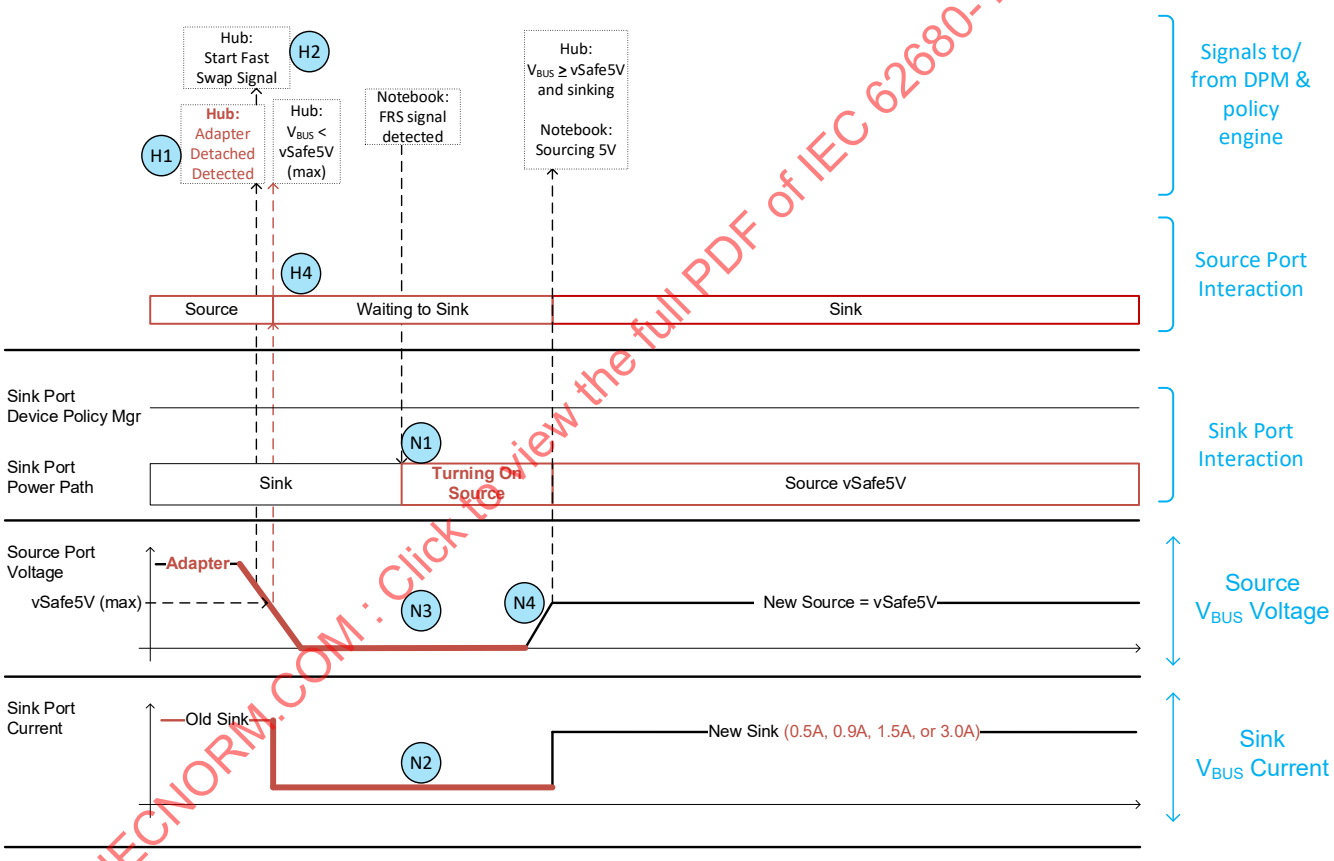


Table E-4 Vbus discharges quickly after adapter disconnected.

Step #	Hub	Notebook
1	The Power Adapter is detached from the Hub.	
2	Hub detects Power Adapter disconnect (H1) causing V_{HubVB} to drop below 5.5V very rapidly.	
3	Hub sends FRS signal on CC (H2) and starts monitoring V_{HubVB} (H3). Hub opens sourcing switch (H4). Hub also starts a ***tSnkFRSwap*** timer.	
4	Hub closes the switch (H5) to use the hold-up capacitor to supply V_{BUS} to the peripheral(s).	

	Systems with a holding cap permanently in place do not need the switch (H5). Hub does not draw more than ***pSnkStdby*** from V_{BUS}, until the ***tSnkFRSwap*** timer expires.	
5		Notebook detects FRS signal on CC (N1) that triggers sending of the ***FR_Swap*** Message. This may happen at any point in the following steps so long as it is within 15 ms (***tFRSwapInit***).
6		Notebook opens the sinking switch (N2), as quickly as possible to minimize power drained from hub after FRS signal.
7		Notebook begin monitoring V_{BUS} (N3) to know when to turn the Notebook into a Source.
8		Notebook detects $V_{BUS} < V_{NbVB}$ (N3).
9		Notebook closes sourcing switch (N4). When this occurs the Hub's input capacitance on V_{BUS} will be less than 10 µF (***cSnkBulk***).
10	Hub's ***tSnkFRSwap*** timer expires (H6).	
11	Hub draws up to the current it advertised in the Fast Role Swap field of its ***Sink_Capabilities*** Message.	
12	Hubs with (H5) will open (H5) and remove the Hold-Up capacitor.	

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

INTERFACES DE BUS UNIVERSEL EN SÉRIE POUR LES DONNÉES ET L'ALIMENTATION ÉLECTRIQUE

Partie 1-2: Composants communs – Spécification de l'alimentation électrique par port USB

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, l'IEC – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de l'IEC"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'IEC, participent également aux travaux. L'IEC collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) L'IEC elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de l'IEC. L'IEC n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à l'IEC, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de l'IEC, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de l'IEC ou de toute autre Publication de l'IEC, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de l'IEC peuvent faire l'objet de droits de brevet. L'IEC ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

La Norme internationale IEC 62680-1-2 a été établie par le domaine technique 18: Systèmes et applications domestiques multimédia pour les réseaux d'utilisateurs finaux, du comité d'études 100 de l'IEC: Systèmes et équipements audio, vidéo et services de données.

Le texte de la présente norme a été établi par l'USB Implementers Forum (USB-IF). Les règles structurelles et éditoriales utilisées dans la présente publication reflètent les pratiques en vigueur au sein de l'organisme responsable de sa soumission.

Le texte de cette Norme internationale est issu des documents suivants:

CDV	Rapport de vote
100/3440/CDV	100/3505/RVC

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette Norme internationale.

Ce document a été rédigé selon les Directives ISO/IEC, Partie 2.

Le comité a décidé que le contenu de ce document ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous "http://webstore.iec.ch" dans les données relatives au document recherché. A cette date, le document sera

- reconduit,
- supprimé,
- remplacé par une édition révisée, ou
- amendé.

IMPORTANT – Le logo *"colour inside"* qui se trouve sur la page de couverture de ce document indique qu'il contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer ce document en utilisant une imprimante couleur.

INTRODUCTION

La série IEC 62680 repose sur une série de spécifications qui ont été développées à l'origine par l'USB Implementers Forum (USB-IF). Ces spécifications ont été soumises à l'IEC dans le cadre d'un accord particulier conclu entre l'IEC et l'USB-IF.

La présente norme est la publication de l'USB-IF relative à l'alimentation électrique par port USB, révision 3.0, version 2.0.

L'USB Implementers Forum, Inc. (USB-IF) est un organisme à but non lucratif fondé par le groupe de sociétés qui a développé la spécification du bus universel en série. L'USB-IF a été créé pour fournir une plateforme de soutien et de forum pour le progrès et l'adoption de la technologie du bus universel en série. Le forum facilite le développement de périphériques (appareils) USB compatibles et de haute qualité et promeut les avantages de la technologie USB et la qualité des produits qui ont été validés par des essais de conformité.

TOUTES LES SPÉCIFICATIONS USB VOUS SONT FOURNIES "EN L'ÉTAT", SANS GARANTIE D'AUCUNE SORTE, Y COMPRIS TOUTE GARANTIE DE QUALITÉ MARCHANDE, DE NON-VIOLATION OU D'ADÉQUATION À UN USAGE PARTICULIER. L'USB IMPLEMENTERS FORUM ET LES AUTEURS DE L'ENSEMBLE DES SPÉCIFICATIONS USB CI-APRÈS DÉCLINENT TOUTE RESPONSABILITÉ, Y COMPRIS TOUTE RESPONSABILITÉ RELATIVE À LA VIOLATION DE DROITS DE PROPRIÉTÉ, EN CE QUI CONCERNE L'UTILISATION OU LA MISE EN ŒUVRE DES INFORMATIONS CONTENUES DANS LA PRÉSENTE SPÉCIFICATION.

LA MISE À DISPOSITION D'UNE SPÉCIFICATION USB, QUELLE QU'ELLE SOIT, N'IMPLIQUE L'OCTROI D'AUCUNE LICENCE, EXPRESSE OU IMPLICITE, PAR PERCLUSION OU AUTRE, SUR AUCUN DROIT DE PROPRIÉTÉ INTELLECTUELLE.

La conclusion des accords des adoptants de l'USB peut toutefois permettre à une société signataire de participer à un accord de licence réciproque RAND-Z pour les produits conformes. Pour plus d'informations, se rendre sur:

https://www.usb.org/documents

L'IEC NE PREND PAS POSITION SUR LA QUESTION DE SAVOIR S'IL VAUT LA PEINE QUE VOUS CONCLUIEZ UN QUELCONQUE ACCORD USB ADOPTERS AGREEMENT OU QUE VOUS PARTICIPIEZ À L'USB IMPLEMENTERS FORUM.

Bus universel en série
Spécification de l'alimentation électrique

Révision: ***3.0***

Version: ***2.0***

Date de publication: ***29 août 2019***

LICENCE LIMITÉE DE DROITS D'AUTEUR

LES PROMOTEURS DE L'USB 3.0 DÉLIVRENT UNE LICENCE CONDITIONNELLE DE DROITS D'AUTEUR SOUS LES DROITS INCLUS DANS LA SPÉCIFICATION DE L'ALIMENTATION ÉLECTRIQUE PAR PORT USB AFIN D'UTILISER ET DE REPRODUIRE LA SPÉCIFICATION DANS LE SEUL BUT, ET UNIQUEMENT SI NÉCESSAIRE, D'ÉVALUER LA PERTINENCE DE LA MISE EN ŒUVRE DE LA SPÉCIFICATION AVEC DES PRODUITS CONFORMES À LA SPÉCIFICATION. NONOBSTANT CE QUI PRÉCÈDE, L'UTILISATION DE LA SPÉCIFICATION EN VUE DE DÉPOSER OU DE MODIFIER UNE DEMANDE DE BREVET RELATIVE À LA SPÉCIFICATION OU À DES PRODUITS CONFORMES USB N'EST PAS AUTORISÉE. HORMIS CETTE LICENCE EXPLICITE DE DROITS D'AUTEUR, AUCUN AUTRE DROIT OU LICENCE N'EST ACCORDÉ, CE SANS LIMITATION DES LICENCES DE BREVETS. POUR OBTENIR D'AUTRES LICENCES DE PROPRIÉTÉ INTELLECTUELLE OU DES ENGAGEMENTS CONCERNANT LES DROITS ASSOCIÉS À LA SPÉCIFICATION, UNE PARTIE DOIT EXÉCUTER L'ACCORD DES ADOPTANTS DE L'USB 3.0. NOTE: EN UTILISANT LA SPÉCIFICATION, VOUS ACCEPTEZ LES TERMES DE CETTE LICENCE EN VOTRE PROPRE NOM ET, SI VOUS LE FAITES EN QUALITÉ D'EMPLOYÉ, AU NOM DE VOTRE EMPLOYEUR.

DÉNI DE RESPONSABILITÉ SUR LA PROPRIÉTÉ INTELLECTUELLE

LA PRÉSENTE SPÉCIFICATION VOUS EST FOURNIE "EN L'ÉTAT", SANS GARANTIE D'AUCUNE SORTE, EN CE COMPRIS TOUTE GARANTIE DE QUALITÉ MARCHANDE, DE NON-VIOLATION OU D'ADAPTATION À UN USAGE PARTICULIER. LES AUTEURS DE LA PRÉSENTE SPÉCIFICATION DÉCLINENT TOUTE RESPONSABILITÉ, Y COMPRIS TOUTE RESPONSABILITÉ RELATIVE À LA VIOLATION DE DROITS DE PROPRIÉTÉ, EN CE QUI CONCERNE L'UTILISATION OU LA MISE EN ŒUVRE DES INFORMATIONS CONTENUES DANS LA PRÉSENTE SPÉCIFICATION. LA DISPOSITION DE LA PRÉSENTE SPÉCIFICATION N'IMPLIQUE L'OCTROI D'AUCUNE LICENCE, EXPRESSE OU IMPLICITE, PAR PERCLUSION OU AUTRE, SUR AUCUN DROIT DE PROPRIÉTÉ INTELLECTUELLE.

Envoyer les commentaires par courrier électronique à techsup@usb.org

Pour plus d'informations, se rendre sur le site web de l'USB Implementers Forum à l'adresse http://www.usb.org

USB Type-C® et USB4™ sont des marques de l'Universal Serial Bus Implementers Forum (USB-IF). Thunderbolt™ est une marque commerciale d'Intel Corporation.

La marque ou le logo Thunderbolt™ ne peut être utilisé qu'avec des produits conçus selon cette spécification, qui ont reçu la certification appropriée et sont utilisés dans le cadre d'une licence de la marque Thunderbolt™ – voir http://usb.org/compliance pour plus d'informations.

Tous les noms de produits sont des marques, des marques déposées ou des marques de service de leurs propriétaires respectifs.

Présidence

Alvin Cox	Cabling Sub-Chair
Bob Dunstan	Specification Chair/Protocol Subgroup Chair
Deric Waters	PHY Chair
Ed Berrios	Power Supply Chair
Rahman Ismail	System Policy Chair
Richard Petrie	Specification Chair/Device Policy Chair

Editeurs

Bob Dunstan

Richard Petrie

Participants

Charles Wang	ACON, Advanced-Connectek, Inc.	Jie min	Cadence Design Systems, Inc.
Conrad Choy	ACON, Advanced-Connectek, Inc.	Mark Summers	Cadence Design Systems, Inc.
Dennis Chuang	ACON, Advanced-Connectek, Inc.	Michal Staworko	Cadence Design Systems, Inc.
Steve Sedio	ACON, Advanced-Connectek, Inc.	Sathish Kumar Ganesan	Cadence Design Systems, Inc.
Sunney Yang	ACON, Advanced-Connectek, Inc.	Alessandro Ingrassia	Canova Tech
Vicky Chuang	ACON, Advanced-Connectek, Inc.	Andrea Colognese	Canova Tech
Joseph Scanlon	Advanced Micro Devices	Antonio Orzelli	Canova Tech
Caspar Lin	Allion Labs, Inc.	Davide Ghedin	Canova Tech
Casper Lee	Allion Labs, Inc.	Matteo Casalin	Canova Tech
Danny Shih	Allion Labs, Inc.	Michael Marioli	Canova Tech
Howard Chang	Allion Labs, Inc.	Nicola Scantamburlo	Canova Tech
Greg Stewart	Analogix Semiconductor, Inc.	Paolo Pilla	Canova Tech
Mehran Badii	Analogix Semiconductor, Inc.	Yi-Feng Lin	Canyon Semiconductor
Alexei Kosut	Apple	YuHung Lin	Canyon Semiconductor
Bill Cornelius	Apple	David Tsai	Chrontel, Inc.
Carlos Colderon	Apple	Anshul Gulati	Cypress Semiconductor
Chris Uiterwijk	Apple	Anup Nayak	Cypress Semiconductor
Colin Whitby-Strevens	Apple	Benjamin Kropf	Cypress Semiconductor
Corey Axelowitz	Apple	Dhanraj Rajput	Cypress Semiconductor
Corey Lange	Apple	Ganesh Subramaniam	Cypress Semiconductor
Dave Conroy	Apple	Jagadeesan Raj	Cypress Semiconductor
David Sekowski	Apple	Junjie cui	Cypress Semiconductor
Girault Jones	Apple	Manu Kumar	Cypress Semiconductor
James Orr	Apple	Muthu M	Cypress Semiconductor
Jason Chung	Apple	Nicholas Bodnaruk	Cypress Semiconductor
Jay Kim	Apple	Pradeep Bajpai	Cypress Semiconductor
Jeff Wilcox	Apple	Rajaram R	Cypress Semiconductor
Jennifer Tsai	Apple	Rama Vakkantula	Cypress Semiconductor
Karl Bowers	Apple	Rushil Kadakia	Cypress Semiconductor
Keith Porthouse	Apple	Simon Nguyen	Cypress Semiconductor
Kevin Hsiue	Apple	Steven Wong	Cypress Semiconductor
Matt Mora	Apple	Subu Sankaran	Cypress Semiconductor
Paul Baker	Apple	Sumeet Gupta	Cypress Semiconductor
Reese Schreiber	Apple	Tejender Sheoran	Cypress Semiconductor
Ruchi Chaturvedi	Apple	Venkat Mandagulathar	Cypress Semiconductor
Sameer Kelkar	Apple	Xiaofeng Shen	Cypress Semiconductor
Sasha Tietz	Apple	Zeng Wei	Cypress Semiconductor
Scott Jackson	Apple	Adie Tan	Dell Inc.
Sree Raman	Apple	Adolfo Montero	Dell Inc.
William Ferry	Apple	Bruce Montag	Dell Inc.
Zaki Moussaoui	Apple	Gary Verdun	Dell Inc.
Jeff Liu	ASMedia Technology Inc.	Marcin Nowak	Dell Inc.
Kuo Lung Li	ASMedia Technology Inc.	Merle Wood	Dell Inc.
Ming-Wei Hsu	ASMedia Technology Inc.	Mohammed Hijazi	Dell Inc.
PS Tseng	ASMedia Technology Inc.	Siddhartha Reddy	Dell Inc.
Sam Tzeng	ASMedia Technology Inc.	Bindhu Vasu	Dialog Semiconductor (UK) Ltd
Thomas Hsu	ASMedia Technology Inc.	Chanchal Gupta	Dialog Semiconductor (UK) Ltd
Weikao Chang	ASMedia Technology Inc.	Dipti Baheti	Dialog Semiconductor (UK) Ltd
Yang Cheng	ASMedia Technology Inc.	Duc Doan	Dialog Semiconductor (UK) Ltd
Shawn Meng	Bizlink Technology Inc.	Holger Petersen	Dialog Semiconductor (UK) Ltd
Bernard Shyu	Bizlink Technology, Inc.	Jianming Yao	Dialog Semiconductor (UK) Ltd
Eric Wu	Bizlink Technology, Inc.	John Shi	Dialog Semiconductor (UK) Ltd
Morphy Hsieh	Bizlink Technology, Inc.	KE Hong	Dialog Semiconductor (UK) Ltd
Sean O'Neal	Bizlink Technology, Inc.	Kevin Mori	Dialog Semiconductor (UK) Ltd
Tiffany Hsiao	Bizlink Technology, Inc.	Larry Ping	Dialog Semiconductor (UK) Ltd
Weichung Ooi	Bizlink Technology, Inc.	Mengfei Liu	Dialog Semiconductor (UK) Ltd
Rahul Bhushan	Broadcom Corp.	Scott Brown	Dialog Semiconductor (UK) Ltd
Asila nahas	Cadence Design Systems, Inc.	Yimin Chen	Dialog Semiconductor (UK) Ltd
Claire Ying	Cadence Design Systems, Inc.	Yong Li	Dialog Semiconductor (UK) Ltd

Dan Ellis	DisplayLink (UK) Ltd.	Alan Berkema	Hewlett Packard
Jason Young	DisplayLink (UK) Ltd.	Lee Atkinson	Hewlett Packard
Kevin Jacobs	DisplayLink (UK) Ltd.	Rahul Lakdawala	Hewlett Packard
Paulo Alcobia	DisplayLink (UK) Ltd.	Robin Castell	Hewlett Packard
Peter Burgers	DisplayLink (UK) Ltd.	Roger Benson	Hewlett Packard
Richard Petrie	DisplayLink (UK) Ltd.	Ron Schooley	Hewlett Packard
Abel Astley	Ellisys	Hideyuki HAYAFUJI	Hosiden Corporation
Chuck Trefts	Ellisys	Keiji Mine	Hosiden Corporation
Emmanuel Durin	Ellisys	Masaki YAMAOKA	Hosiden Corporation
Mario Pasquali	Ellisys	Takashi MUTO	Hosiden Corporation
Tim Wei	Ellisys	Yasunori NISHIKAWA	Hosiden Corporation
Chien-Cheng Kuo	Etron Technology, Inc.	Kenneth Chan	HP Inc.
Jack Yang	Etron Technology, Inc.	Lee Atkinson	HP Inc.
Richard Crisp	Etron Technology, Inc.	Steve Chen	HP Inc.
Shyanjia Chen	Etron Technology, Inc.	Suketu Partiwala	HP Inc.
TsungTa Lu	Etron Technology, Inc.	Suketu Partiwala	HP Inc.
Christian Klein	Fairchild Semiconductor	Vaibhav Malik	HP Inc.
Oscar Freitas	Fairchild Semiconductor	Walter Fry	HP Inc.
Souhib Harb	Fairchild Semiconductor	Bai Sean	Huawei Technologies Co., Ltd.
Amanda Ying	Feature Integration Technology Inc.	Chunjiang Zhao	Huawei Technologies Co., Ltd.
Jacky Chan	Feature Integration Technology Inc.	JianQuan Wu	Huawei Technologies Co., Ltd.
Kenny Hsieh	Feature Integration Technology Inc.	Li Zongjian	Huawei Technologies Co., Ltd.
KungAn Lin	Feature Integration Technology Inc.	Lihua Duan	Huawei Technologies Co., Ltd.
Paul Yang	Feature Integration Technology Inc.	Min Chen	Huawei Technologies Co., Ltd.
su Jaden	Feature Integration Technology Inc.	Wang Feng	Huawei Technologies Co., Ltd.
Yu-Lin Chu	Feature Integration Technology Inc.	Wei Haihong	Huawei Technologies Co., Ltd.
Yulin Lan	Feature Integration Technology Inc.	Robert Heaton	Indie Semiconductor
AJ Yang	Foxconn/Hon Hai	Vincent Wang	Indie Semiconductor
Bob Hall	Foxconn/Hon Hai	Sie Boo Chiang	Infineon Technologies
Fred Fons	Foxconn/Hon Hai	Tue Fatt David Wee	Infineon Technologies
Jie zheng	Foxconn/Hon Hai	Wee Tar Richard Ng	Infineon Technologies
Patrick Casher	Foxconn/Hon Hai	Wolfgang Furtner	Infineon Technologies
Steve Sedio	Foxconn/Hon Hai	Bob Dunstan	Intel Corporation
Terry Little	Foxconn/Hon Hai	Brad Saunders	Intel Corporation
Bob McVay	Fresco Logic Inc.	Chee Lim Nge	Intel Corporation
Christopher Meyers	Fresco Logic Inc.	Christine Krause	Intel Corporation
Dian Kurniawan	Fresco Logic Inc.	Dan Froelich	Intel Corporation
Tom Burton	Fresco Logic Inc.	David Harriman	Intel Corporation
Adam Rodriguez	Google Inc.	David Hines	Intel Corporation
Alec Berg	Google Inc.	David Thompson	Intel Corporation
Dave Bernard	Google Inc.	Guobin Liu	Intel Corporation
David Schneider	Google Inc.	Harry Skinner	Intel Corporation
Jim Guerin	Google Inc.	Henrik Leegaard	Intel Corporation
Juan Fantin	Google Inc.	Jenn Chuan Cheng	Intel Corporation
Ken Wu	Google Inc.	Jervis Lin	Intel Corporation
Mark Hayter	Google Inc.	John Howard	Intel Corporation
Nithya Jagannathan	Google Inc.	Karthi Vadivelu	Intel Corporation
Srikanth Lakshmikanthan	Google Inc.	Leo Heiland	Intel Corporation
Todd Broch	Google Inc.	Maarit Harkonen	Intel Corporation
Toshak Singhal	Google Inc.	Nge Chee Lim	Intel Corporation
Vincent Palatin	Google Inc.	Paul Durley	Intel Corporation
Xuelin Wu	Google Inc.	Rahman Ismail	Intel Corporation
Alan Kinningham	Granite River Labs	Rajaram Regupathy	Intel Corporation
Balamurugan Manialagan	Granite River Labs	Ronald Swartz	Intel Corporation
Mike Engbretson	Granite River Labs	Sarah Sharp	Intel Corporation
Mike Wu	Granite River Labs	Scott Brenden	Intel Corporation
Mukesh Tatiya	Granite River Labs	Sridharan Ranganathan	Intel Corporation
Rajaraman V	Granite River Labs	Steve McGowan	Intel Corporation
Tim Lin	Granite River Labs	Tim McKee	Intel Corporation

Toby Opferman	Intel Corporation
Ziv Kabiry	Intel Corporation
Jia Wei	Intersil Corporation
Al Hsiao	ITE Tech. Inc.
Greg Song	ITE Tech. Inc.
Richard Guo	ITE Tech. Inc.
Victor Lin	ITE Tech. Inc.
Y.C. Chou	ITE Tech. Inc.
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.
Jit Lim	Keysight Technologies Inc.
Babu Mailachalam	Lattice Semiconductor Corp
Gianluca Mariani	Lattice Semiconductor Corp
Joel Coplen	Lattice Semiconductor Corp
Thomas Watza	Lattice Semiconductor Corp
Vesa Lauri	Lattice Semiconductor Corp
Keneth Kim	LG electronics
Bruce Chuang	Leadtrend
Eilian Liu	Leadtrend
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
Aaron Melgar	Lion Semiconductor
Chris Zhou	Lion Semiconductor
Sehyung Jeon	Lion Semiconductor
Wonyoung Kim	Lion Semiconductor
Yongho Kim	Lion Semiconductor
Dave Thompson	LSI Corporation
Alan Kinningham	Luxshare-ICT
Daniel Chen	Luxshare-ICT
Eric Wen	Luxshare-ICT
James Stevens	Luxshare-ICT
Josue Castillo	Luxshare-ICT
Pat Young	Luxshare-ICT
Scott Shuey	Luxshare-ICT
Chikara Kakizawa	Maxim Integrated Products
Jacob Scott	Maxim Integrated Products
Ken Helfrich	Maxim Integrated Products
Michael Miskho	Maxim Integrated Products
Chris Yokum	MCCI Corporation
Geert Knapen	MCCI Corporation
Terry Moore	MCCI Corporation
Velmurugan Selvaraj	MCCI Corporation
Satoru Kumashiro	MegaChips Corporation
Brian Marley	Microchip Technology Inc.
Dave Perchlik	Microchip Technology Inc.
Don Perkins	Microchip Technology Inc.
Fernando Gonzalez	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.
Prasanna Vengateshan	Microchip Technology Inc.
Rich Wahler	Microchip Technology Inc.
Richard Petrie	Microchip Technology Inc.
Ronald Kunin	Microchip Technology Inc.
Shannon Cash	Microchip Technology Inc.
Thomas Farkas	Microchip Technology Inc.
Andrew Yang	Microsoft Corporation
Anthony Chen	Microsoft Corporation
Arvind Murching	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
Adib Al Abaji	Molex LLC
Aaron Xu	Monolithic Power Systems Inc.
Bo Zhou	Monolithic Power Systems Inc.
Christian Sporck	Monolithic Power Systems Inc.
Di Han	Monolithic Power Systems Inc.
Zhihong Yu	Monolithic Power Systems Inc.
Dan Wagner	Motorola Mobility Inc.
Ben Crowe	MQP Electronics Ltd.
Pat Crowe	MQP Electronics Ltd.
Sten Carlsen	MQP Electronics Ltd.
Kenji Oguma	NEC Corporation
Frank Borngräber	Nokia Corporation
Kai Inha	Nokia Corporation
Pekka Leinonen	Nokia Corporation
Richard Petrie	Nokia Corporation
Sten Carlsen	Nokia Corporation
Abhijeet Kulkarni	NXP Semiconductors
Ahmad Yazdi	NXP Semiconductors
Bart Vertenten	NXP Semiconductors
Dennis Ha	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
Winston Langeslag	NXP Semiconductors
Robert Heaton	Obsidian Technology
Andrew Yoo	ON Semiconductor
Brady Maasen	ON Semiconductor
Bryan McCoy	ON Semiconductor

Christian Klein	ON Semiconductor	Matti Kulmala	Salcomp Plc
Cor Voorwinden	ON Semiconductor	Toni Lehimo	Salcomp Plc
Edward Berrios	ON Semiconductor	Tong Kim	Samsung Electronics Co. Ltd.
Michael Smith	ON Semiconductor	Alvin Cox	Seagate Technology LLC
Oscar Freitas	ON Semiconductor	Emmanuel Lemay	Seagate Technology LLC
Tom Duffy	ON Semiconductor	John Hein	Seagate Technology LLC
Craig Wiley	Parade Technologies Inc.	Marc Noblitt	Seagate Technology LLC
Aditya Kulkarni	Power Integrations	Michael Morgan	Seagate Technology LLC
Amruta Patra	Power Integrations	Ronald Rueckert	Seagate Technology LLC
Rahul Joshi	Power Integrations	Tony Priborsky	Seagate Technology LLC
Ricardo Pregiteer	Power Integrations	Chin Chang	Semtech Corporation
Shruti Anand	Power Integrations	Tom Farkas	Semtech Corporation
Amit gupta	Qualcomm, Inc	Ning Dai	Silergy Corp.
George Paparrizos	Qualcomm, Inc	Wanfeng Zhang	Silergy Corp.
Giovanni Garcea	Qualcomm, Inc	Kafai Leung	Silicon Laboratories, Inc.
Jack Pham	Qualcomm, Inc	Kok Hong Soh	Silicon Laboratories, Inc.
James Goel	Qualcomm, Inc	Sorin Badiu	Silicon Laboratories, Inc.
Joshua Warner	Qualcomm, Inc	Steven Ghang	Silicon Laboratories, Inc.
Karyn Vuong	Qualcomm, Inc	Abhishek Sardeshpande	SiliConch Systems Private Limited
Lalan Mishra	Qualcomm, Inc	Aniket Mathad	SiliConch Systems Private Limited
Vamsi Samavedam	Qualcomm, Inc	Chandana N	SiliConch Systems Private Limited
Vatsal Patel	Qualcomm, Inc	Jaswanth Ammineni	SiliConch Systems Private Limited
Chris Sporck	Qualcomm, Inc.	Jinisha Patel	SiliConch Systems Private Limited
Craig Aiken	Qualcomm, Inc.	Kaustubh Kumar	SiliConch Systems Private Limited
Narendra Mehta	Qualcomm, Inc.	Nitish Nitish	SiliConch Systems Private Limited
Terry Remple	Qualcomm, Inc.	Pavitra Balasubramanian	SiliConch Systems Private Limited
Will Kun	Qualcomm, Inc.	Rakesh Polasa	SiliConch Systems Private Limited
Yoram Rimoni	Qualcomm, Inc.	Satish Anand Verkila	SiliConch Systems Private Limited
Fan-Hau Hsu	Realtek Semiconductor Corp.	Shubham Paliwal	SiliConch Systems Private Limited
Tsung-Peng Chuang	Realtek Semiconductor Corp.	Vishnu Pusuluri	SiliConch Systems Private Limited
Atsushi Mitamura	Renesas Electronics Corp.	John Sisto	SMSC
Bob Dunstan	Renesas Electronics Corp.	Ken Gay	SMSC
Brian Allen	Renesas Electronics Corp.	Mark Bohm	SMSC
Dan Aoki	Renesas Electronics Corp.	Richard Wahler	SMSC
Hajime Nozaki	Renesas Electronics Corp.	Shannon Cash	SMSC
John Carpenter	Renesas Electronics Corp.	Tim Knowlton	SMSC
Kiichi Muto	Renesas Electronics Corp.	William Chiechi	SMSC
Masami Katagiri	Renesas Electronics Corp.	Shigenori Tagami	Sony Corporation
Nobuo Furuya	Renesas Electronics Corp.	Shinichi Hirata	Sony Corporation
Patrick Yu	Renesas Electronics Corp.	Amanda Hosler	Specwerkz
Peter Teng	Renesas Electronics Corp.	Bob Dunstan	Specwerkz
Philip Leung	Renesas Electronics Corp.	Diane Lenox	Specwerkz
Steve Roux	Renesas Electronics Corp.	Michael Munn	StarTech.com Ltd.
Tetsu Sato	Renesas Electronics Corp.	Fabien Friess	ST-Ericsson
Toshifumi Yamaoka	Renesas Electronics Corp.	Giuseppe Platania	ST-Ericsson
Chunan Kuo	Richtek Technology Corporation	Jean-Francois Gatto	ST-Ericsson
Heinz Wei	Richtek Technology Corporation	Milan Stamenkovic	ST-Ericsson
TZUHSIEN CHUANG	Richtek Technology Corporation	Nicolas Florenchie	ST-Ericsson
Tatsuya Irisawa	Ricoh Company Ltd.	Patrizia Milazzo	ST-Ericsson
Akihiro Ono	Rohm Co. Ltd.	Christophe Cochard	STMicroelectronics
Chris Lin	Rohm Co. Ltd.	Christophe Lorin	STMicroelectronics
Hidenori Nishimoto	Rohm Co. Ltd.	Filippo Bonaccorso	STMicroelectronics
Kris Bahar	Rohm Co. Ltd.	Jessy Guilbot	STMicroelectronics
Manabu Miyata	Rohm Co. Ltd.	Joel Huloux	STMicroelectronics
Ruben Balbuena	Rohm Co. Ltd.	John Bloomfield	STMicroelectronics
Takashi Sato	Rohm Co. Ltd.	Massimo Panzica	STMicroelectronics
Vijendra Kuroodi	Rohm Co. Ltd.	Meriem Mersel	STMicroelectronics
Yusuke Kondo	Rohm Co. Ltd.	Nathalie Ballot	STMicroelectronics
Kazuomi Nagai	ROHM Co., Ltd.	Pascal Legrand	STMicroelectronics

Patrizia Milazzo	STMicroelectronics
Richard O'Connor	STMicroelectronics
Morten Christiansen	Synopsys, Inc.
Nivin George	Synopsys, Inc.
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
Grant Ley	Texas Instruments
Gregory Watkins	Texas Instruments
Ingolf Frank	Texas Instruments
Ivo Huber	Texas Instruments
Javed Ahmad	Texas Instruments
Jean Picard	Texas Instruments
John Perry	Texas Instruments
Martin Patoka	Texas Instruments
Mike Campbell	Texas Instruments
Scott Jackson	Texas Instruments
Shafiuddin Mohammed	Texas Instruments
Srinath Hosur	Texas Instruments
Steven Tom	Texas Instruments
Yoon Lee	Texas Instruments
Tim Wilhelm	The Silanna Group Pty. Ltd.
Tod Wolf	The Silanna Group Pty. Ltd.
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.
Ho Wen Tsai	Weltrend Semiconductor
Hung Chiang	Weltrend Semiconductor
Jeng Cheng Liu	Weltrend Semiconductor
Priscilla Lee	Weltrend Semiconductor
Wayne Lo	Weltrend Semiconductor
Charles Neumann	Western Digital Technologies, Inc.
Curtis Stevens	Western Digital Technologies, Inc.
John Maroney	Western Digital Technologies, Inc.
Joe O'Brien	Wilder Technologies
Will Miller	Wilder Technologies
Juejia Zhou	Xiaomi Communications Co., Ltd.
Xiaoxing Yang	Xiaomi Communications Co., Ltd.

Historique des révisions

Révision	Version	Commentaires	Date de publication
1.0	1.0	Version initiale Révision 1.0	5 juillet 2012
1.0	1.1	Y compris errata jusqu'au 31 octobre 2012	31 octobre 2012
1.0	1.2	Y compris errata jusqu'au 26 juin 2013	26 juin 2013
1.0	1.3	Y compris errata jusqu'au 11 mars 2014	11 mars 2014
2.0	1.0	Version initiale Révision 2.0	11 août 2014
2.0	1.1	Y compris errata jusqu'au 7 mai 2015	7 mai 2015
2.0	1.2	Y compris errata jusqu'au 25 mars 2016	25 mars 2016
2.0	1.3	Y compris errata jusqu'au 11 janvier 2017	11 janvier 2017
3.0	1.0	Version initiale Révision 3.0	11 décembre 2015
3.0	1.0a	Y compris errata jusqu'au 25 mars 2016	25 mars 2016
3.0	1.1	Y compris errata jusqu'au 12 janvier 2016	12 janvier 2017
3.0	1.2	Y compris errata jusqu'au 21 juin 2018	21 juin 2018
3.0	2.0	Y compris errata jusqu'au 29 août 2019	29 août 2019

Sommaire

AVANT-PROPOS 644
INTRODUCTION 646
DÉNI DE RESPONSABILITÉ SUR LA PROPRIÉTÉ INTELLECTUELLE 648
Présidence 650
Editeurs 650
Participants 650
Historique des révisions 656
Sommaire 657
Tableaux 666
Figures 672
1. Introduction 682
1.1 Vue d'ensemble 682
1.2 Objet 683
1.3 Domaine d'application 684
1.4 Conventions 684
1.4.1 Ordre de priorité 684
1.4.2 Mots-clés 684
1.4.3 Numérotation 686
1.5 Documents connexes 686
1.6 Termes et abréviations 686
1.7 Valeurs de paramètres 695
1.8 Modifications par rapport à la révision 2.0 696
1.9 Compatibilité avec la révision 2.0 696
2. Vue d'ensemble 697
2.1 Introduction 697
2.2 Vue d'ensemble des sections 698
2.3 Compatibilité avec la révision 2.0 700
2.4 Dispositifs aptes à l'alimentation électrique par port USB 700
2.5 Communication SOP* 701

2.5.1 Introduction 701
2.5.2 SOP* anticollision 701
2.5.3 Communication SOP 702
2.5.4 Communication SOP'/SOP'' avec des fiches de câbles 702
2.6 Vue d'ensemble du fonctionnement 704
2.6.1 Fonctionnement de la source 704
2.6.2 Fonctionnement du destinataire 707
2.6.3 Fiches de câbles 709
2.7 Vue d'ensemble de l'architecture 710
2.7.1 Politique 713
2.7.2 Formation et transmission des messages 714
2.7.3 Anticollision 714
2.7.4 Alimentation 715
2.7.5 DFP/UFP 715
2.7.6 Câble et connecteurs 716
2.7.7 Interactions entre dispositifs non PD, BC et PD 716
2.7.8 Règles d'alimentation 716
3. Assemblages de câbles et connecteurs USB Type-A et USB Type-B 717
4. Exigences électriques 718
4.1 Interopérabilité avec les autres spécifications USB 718
4.2 Détection de batterie déchargée/Détection de port non alimenté 718
4.3 Chute de tension ohmique d'un câble par rapport à la masse (chute de tension ohmique) 718
4.4 Détection du type de câble 718
5. Couche physique 720
5.1 Vue d'ensemble de la couche physique 720
5.2 Fonctions de la couche physique 720
5.3 Codage de symboles 721
5.4 Ensembles ordonnés 723
5.5 Ordonnancement des bits transmis 724
5.6 Format de paquet 725
5.6.1 Mise en trames des paquets 726
5.6.2 CRC 728
5.6.3 Erreurs de détection de paquet 730
5.6.4 Réinitialisation matérielle 730

5.6.5 Réinitialisation de câble 731
5.7 Anticollision 732
5.8 Schéma de signalisation Biphase Mark Coding (BMC) 733
5.8.1 Encodage et signalisation 733
5.8.2 Masques d'émission et de réception 737
5.8.3 Modèle de charge d'émetteur 744
5.8.4 Spécifications BMC communes 745
5.8.5 Spécifications pour l'émetteur BMC 746
5.8.6 Spécifications BMC pour le récepteur 749
5.9 Autotest intégré (BIST) 753
5.9.1 Mode porteur BIST 753
5.9.2 BIST Test Data 753
6. Couche protocole 754
6.1 Vue d'ensemble 754
6.2 Messages 754
6.2.1 Construction des messages 754
6.3 Message de contrôle 767
6.3.1 Message GoodCRC 768
6.3.2 Message GotoMin 768
6.3.3 Message Accept 769
6.3.4 Message Reject 769
6.3.5 Message Ping 769
6.3.6 Message PS_RDY 770
6.3.7 Message Get_Source_Cap 770
6.3.8 Message Get_Sink_Cap 770
6.3.9 Message DR_Swap 770
6.3.10 Message PR_Swap 771
6.3.11 Message VCONN_Swap 771
6.3.12 Message Wait 772
6.3.13 Message Soft Reset 773
6.3.14 Message Data_Reset 774
6.3.15 Message Data_Reset_Complete 775
6.3.16 Message Not_Supported 775
6.3.17 Message Get_Source_Cap_Extended 775
6.3.18 Message Get_Status 775

6.3.19 Message FR_Swap 775
6.3.20 Get_PPS_Status 776
6.3.21 Get_Country_Codes 776
6.3.22 Message Get_Sink_Cap_Extended 776
6.4 Message de données 776
6.4.1 Message de capacités 777
6.4.2 Message de demande 788
6.4.3 Message BIST 793
6.4.4 Message Vendor Defined 796
6.4.5 Message Battery_Status 829
6.4.6 Message d'alerte 831
6.4.7 Message Get_Country_Info 833
6.4.8 Message Enter_USB 833
6.5 Message étendu 836
6.5.1 Message Source_Capabilities_Extended 837
6.5.2 Message Status 842
6.5.3 Message Get_Battery_Cap 846
6.5.4 Message Get_Battery_Status 846
6.5.5 Message Battery_Capabilities 847
6.5.6 Message Get_Manufacturer_Info 848
6.5.7 Message Manufacturer_Info 849
6.5.8 Messages de sécurité 850
6.5.9 Messages de mise à jour du micrologiciel 851
6.5.10 Message PPS_Status 852
6.5.11 Message Country_Codes 854
6.5.12 Message Country_Info 854
6.5.13 Message Sink_Capabilities_Extended 855
6.6 Temporisateurs 859
6.6.1 CRCReceiveTimer 859
6.6.2 SenderResponseTimer 860
6.6.3 Temporisateurs de capacité 860
6.6.4 Temporisateurs d'attente et temps d'attente 861
6.6.5 Temporisateurs d'alimentation électrique 861
6.6.6 NoResponseTimer 863
6.6.7 Temporisateurs BIST 863
6.6.8 Temporisateurs de permutation des rôles d'alimentation 864

6.6.9 Temporisateurs de réinitialisation logicielle 864
6.6.10 Temporisateurs de réinitialisation des données 864
6.6.11 Temporisateurs de réinitialisation matérielle 865
6.6.12 Temporisateurs de VDM structuré 866
6.6.13 Temporisateurs de VCONN 867
6.6.14 tCableMessage 867
6.6.15 DiscoverIdentityTimer 867
6.6.16 Temporisateurs anticollisions 868
6.6.17 Temporisateurs de permutation rapide des rôles 868
6.6.18 Temporisateurs de fragmentation 868
6.6.19 Temporisateurs d'alimentation électrique programmable 869
6.6.20 tEnterUSB 870
6.6.21 Valeurs de temporisation et temporisateurs 870
6.7 Compteurs 874
6.7.1 Compteur MessageID 874
6.7.2 Compteur de relances 874
6.7.3 Compteur de réinitialisations matérielles 875
6.7.4 Compteur de capacités 875
6.7.5 Compteur de découvertes d'identités 875
6.7.6 VDMBusyCounter 875
6.7.7 Valeurs de compteurs et compteurs 875
6.8 Réinitialisation 876
6.8.1 Réinitialisation logicielle et erreur de protocole 876
6.8.2 Réinitialisation des données 878
6.8.3 Réinitialisation matérielle 879
6.8.4 Réinitialisation de câble 879
6.9 Anticollision 880
6.10 Rejet de message 880
6.11 Comportement d'état 881
6.11.1 Présentation des diagrammes d'état utilisés au Chapitre 6 881
6.11.2 Fonctionnement d'état 882
6.11.3 Liste des états de la couche protocole 910
6.12 Applicabilité des messages 912
6.12.1 Applicabilité des messages de contrôle 913
6.12.2 Applicabilité des messages de données 914
6.12.3 Applicabilité des messages étendus 915

6.12.4 Applicabilité des commandes de VDM structuré 916
6.12.5 Applicabilité des signaux de réinitialisation 917
6.12.6 Applicabilité des signaux de permutation rapide des rôles 918
6.13 Paramètres de valeur 918
7. Alimentation 919
7.1 Exigences relatives à la source 919
7.1.1 Aspects comportementaux 919
7.1.2 Capacité de masse de la source 919
7.1.3 Types de sources 920
7.1.4 Transitions de source 920
7.1.5 Réponse aux réinitialisations matérielles 929
7.1.6 Modification de la capacité de puissance de sortie 930
7.1.7 Fonctionnement robuste de la source 930
7.1.8 Tolérance et plage de tensions de sortie 931
7.1.9 Charge et décharge de la capacité de masse sur V_{BUS} 933
7.1.10 Veille de permutation pour les sources 933
7.1.11 Fonctionnement en courant de crête de la source 933
7.1.12 Paramètres étendus des capacités de source 934
7.1.13 Permutation rapide des rôles 937
7.1.14 Non application des limites de la vitesse de balayage de V_{BUS} 939
7.1.15 Cycle d'alimentation V_{CONN} 940
7.2 Exigences relatives au destinataire 941
7.2.1 Aspects comportementaux 941
7.2.2 Capacité de masse du destinataire 941
7.2.3 Veille du destinataire 942
7.2.4 Consommation électrique pendant la veille 943
7.2.5 Courant négocié à zéro 943
7.2.6 Comportement de charge transitoire 943
7.2.7 Veille de permutation pour les destinataires 943
7.2.8 Fonctionnement en courant de crête du destinataire 943
7.2.9 Fonctionnement robuste du destinataire 944
7.2.10 Permutation rapide des rôles 945
7.3 Transitions 947
7.3.1 Augmentation du courant 948
7.3.2 Augmentation de la tension 951

7.3.3 Augmentation de la tension et du courant 954
7.3.4 Augmentation de la tension et diminution du courant 957
7.3.5 Diminution de la tension et augmentation du courant 960
7.3.6 Diminution du courant 963
7.3.7 Diminution de la tension 966
7.3.8 Diminution de la tension et du courant 969
7.3.9 Permutation des rôles d'alimentation demandée par le destinataire 972
7.3.10 Permutation des rôles d'alimentation demandée par la source 976
7.3.11 Diminution de courant GotoMin 980
7.3.12 Réinitialisation matérielle déclenchée par la source 983
7.3.13 Réinitialisation matérielle déclenchée par le destinataire 986
7.3.14 Pas de variation de courant ou de tension 989
7.3.15 Permutation rapide des rôles 992
7.3.16 Augmentation de la tension d'alimentation électrique programmable 995
7.3.17 Diminution de la tension d'alimentation électrique programmable 997
7.3.18 Modification du PDO ou de l'APDO source 999
7.3.19 Augmentation du courant d'alimentation électrique programmable 1002
7.3.20 Diminution du courant d'alimentation électrique programmable 1003
7.3.21 Demande identique d'alimentation électrique programmable 1006
7.4 Paramètres électriques 1007
7.4.1 Paramètres électriques de la source 1007
7.4.2 Paramètres électriques du destinataire 1013
7.4.3 Paramètres électriques communs 1015
8. Politique d'utilisation des dispositifs 1016
8.1 Vue d'ensemble 1016
8.2 Gestionnaire de politique d'utilisation des dispositifs 1016
8.2.1 Capacités 1018
8.2.2 Politique système 1018
8.2.3 Contrôle de la source/du destinataire 1018
8.2.4 Détection de câble 1019
8.2.5 Gestion des exigences d'alimentation 1019
8.2.6 Utilisation du bit "Unconstrained Power" avec les batteries et les alimentations en courant alternatif 1021
8.2.7 Interface avec le moteur de politique 1024
8.3 Moteur de politique 1026
8.3.1 Introduction 1026

8.3.2 Schémas de séquence atomique de messages 1026
8.3.3 Diagrammes d'états 1248
9. Etats et rapport de statut 1369
9.1 Vue d'ensemble 1369
9.1.1 Exigences relatives au dispositif et au hub PDUSB 1372
9.1.2 Mise en correspondance vers les états du dispositif USB 1372
9.1.3 Pile de logiciels PD 1375
9.1.4 Enumération du dispositif PDUSB 1376
9.2 Descripteurs spécifiques à l'alimentation USB 1378
9.2.1 Descripteur de capacité USB Power Delivery 1378
9.2.2 Descripteur de capacité Battery Info 1380
9.2.3 Descripteur de capacité PD Consumer Port 1380
9.2.4 Descripteur de capacité PD Provider Port 1381
9.3 Demandes et événements spécifiques à l'alimentation USB 1383
9.3.1 Demandes spécifiques à l'alimentation USB 1383
9.4 Demandes de hub PDUSB et de dispositifs périphériques PDUSB 1384
9.4.1 GetBatteryStatus 1384
9.4.2 SetPDFeature 1385
10. Règles d'alimentation 1388
10.1 Introduction 1388
10.2 Règles d'alimentation de la source 1388
10.2.1 Considérations relatives aux règles d'alimentation de la source 1388
10.2.2 Tensions et courants normalisés 1389
10.2.3 Tensions/courants facultatifs 1392
10.2.4 Partage de puissance entre les ports 1394
10.3 Règles d'alimentation du destinataire 1394
10.3.1 Considérations relatives aux règles d'alimentation du destinataire 1394
10.3.2 Règles normatives du destinataire 1394
A. Calcul de CRC 1396
A.1 Exemple de code C 1396
A.2 Tableau présentant le calcul complet sur un message 1398
B. Exemples de séquence de messages PD 1399
B.1 Alimentation externe fournie en aval 1399
B.2 Alimentation externe fournie en amont 1404
B.3 Rendu de puissance 1413

C. Exemples de commande VDM 1428
C.1 Exemple de découverte d'identité 1428
C.1.1 Demande de commande de découverte d'identité 1428
C.1.2 Réponse de commande de découverte d'identité – Câble actif 1428
C.1.3 Réponse de commande de découverte d'identité – Hub 1430
C.2 Exemple de découverte de SVID 1431
C.2.1 Demande de commande de découverte de SVID 1431
C.2.1 Réponse de commande de découverte de SVID 1431
C.3 Exemple de découverte de modes 1433
C.3.1 Demande de commande de découverte de modes 1433
C.3.2 Réponse de commande de découverte de modes 1433
C.4 Exemple d'entrée dans un mode 1435
C.4.1 Demande de commande d'entrée dans un mode 1435
C.4.2 Réponse de commande d'entrée dans un mode 1435
C.4.1 Demande de commande d'entrée dans un mode avec VDO supplémentaire 1436
C.5 Exemple de sortie d'un mode 1437
C.5.1 Demande de commande de sortie d'un mode 1437
C.5.2 Réponse de commande de sortie d'un mode 1437
C.6 Exemple d'attention 1439
C.6.1 Demande de commande Attention 1439
C.6.2 Demande de commande Attention avec VDO supplémentaire 1439
D. Exemple de conception de récepteur BMC 1441
D.1 Schéma des différences finies 1441
D.1.1 Exemple de circuits 1441
D.1.2 Théorie 1441
D.1.3 Récupération des données 1444
D.1.4 Zone de bruit et zone de détection 1445
D.2 Schéma de soustraction 1445
D.2.1 Exemple de circuits 1445
D.2.2 Sortie de chaque bloc de circuits 1446
D.2.3 Sortie du soustracteur au niveau de la source d'alimentation et du destinataire d'alimentation 1447
D.2.4 Zone de bruit et zone de détection 1447
E. Exemple de FRS au niveau du système 1448
E.1 Vue d'ensemble 1448

E.2 Configuration initiale de FRS 1451
E.3 Processus de permutation rapide des rôles 1453

Tableaux

Tableau 1-1 Termes et abréviations 687
Tableau 5-1 Tableau de codage de symboles 4b5b 721
Tableau 5-2 Ensembles ordonnés 723
Tableau 5-3 Validation d'ensembles ordonnés 724
Tableau 5-4 Taille des données 724
Tableau 5-5 Ensemble ordonné SOP 726
Tableau 5-6 Ensemble ordonné SOP' 727
Tableau 5-7 Ensemble ordonné SOP'' 727
Tableau 5-8 Ensemble ordonné SOP'_Debug 728
Tableau 5-9 Ensemble ordonné SOP''_Debug 728
Tableau 5-10 Mise en correspondance du CRC-32 729
Tableau 5-11 Ensemble ordonné Hard Reset 730
Tableau 5-12 Ensemble ordonné Cable Reset 731
Tableau 5-13 Valeurs de Rp utilisées pour éviter les collisions 733
Tableau 5-14 Définition de masque BMC Tx, valeurs de X 738
Tableau 5-15 Définition de masque BMC Tx, valeurs de Y 739
Tableau 5-16 Définition de masque BMC Rx 743
Tableau 5-17Exigences normatives BMC communes 746
Tableau 5-18 Exigences normatives pour l'émetteur BMC 746
Tableau 5-19 Exigences normatives BMC pour le récepteur 749
Tableau 6-1 En-tête de message 756
Tableau 6-2 Interopérabilité des révisions pendant un contrat explicite 759
Tableau 6-3 En-tête de message étendu 760
Tableau 6-4 Utilisation du bit Unchunked Message Supported 762
Tableau 6-5 Types de messages de contrôle 767
Tableau 6-6 Types de messages de données 777
Tableau 6-7 Objet de données d'alimentation 779
Tableau 6-8 Objet de données d'alimentation augmentée 779
Tableau 6-9 PDO d'alimentation électrique fixe - Source 781
Tableau 6-10 Capacité de courant de crête d'une source d'alimentation fixe 783
Tableau 6-11 PDO d'alimentation électrique variable (batterie exceptée) - Source 784
Tableau 6-12 PDO d'alimentation électrique par batterie - Source 784
Tableau 6-13 APDO d'alimentation électrique programmable - Source 784

Tableau 6-14 PDO d'alimentation électrique fixe - Destinataire 786
Tableau 6-15 PDO d'alimentation électrique variable (batterie exceptée) - Destinataire 787
Tableau 6-16 PDO d'alimentation par batterie - Destinataire 788
Tableau 6-17 APDO d'alimentation électrique programmable - Destinataire 788
Tableau 6-18 Objet de données de demande fixe et variable 789
Tableau 6-19 Objets de données de demande fixe et variable avec prise en charge de GiveBack 789
Tableau 6-20 Objets de données de demande de batterie 789
Tableau 6-21 Objet de données de demande de batterie avec prise en charge de GiveBack 790
Tableau 6-22 Objet de données de demande programmable 790
Tableau 6-23 Objet de données BIST 794
Tableau 6-24 En-tête de VDM non structuré 797
Tableau 6-25 En-tête de VDM structuré 798
Tableau 6-26 Commandes relatives au VDM structuré 799
Tableau 6-27 Valeurs de SVID 800
Tableau 6-28 Commandes et réponses 802
Tableau 6-29 VDO d'en-tête d'ID 804
Tableau 6-30 Types de produits (UFP) 805
Tableau 6-31 Types de produits (fiche de câble) 806
Tableau 6-32 Types de produits (DFP) 806
Tableau 6-33 VDO Cert Stat 807
Tableau 6-34 VDO de produit 807
Tableau 6-35 VDO d'UFP 1 807
Tableau 6-36 VDO d'UFP 2 808
Tableau 6-37 VDO de DFP 809
Tableau 6-38 VDO de câble passif 810
Tableau 6-39 VDO Active Cable 1 812
Tableau 6-40 VDO Active Cable 2 815
Tableau 6-41 VDO d'AMA 817
Tableau 6-42 VDO d'VPD 818
Tableau 6-43 VDO de répondeur de découverte de SVID 820
Tableau 6-44 Objet de données de statut de batterie (BSDO) 830
Tableau 6-45 Objet de données d'alerte 831
Tableau 6-46 Objet de données de code pays 833
Tableau 6-47 Objet de données Enter_USB 834
Tableau 6-48 Types de messages étendus 836
Tableau 6-49 Bloc de données étendues de capacités de source (SCEDB) 837
Tableau 6-50 Bloc de données de statut SOP (SDB) 842
Tableau 6-51 Bloc de données de statut SOP'/SOP" (SDB) 845

Tableau 6-52 Bloc de données d'obtention de capacités de batterie (GBCDB) 846
Tableau 6-53 Bloc de données d'obtention de statut de batterie (GBSDB) 847
Tableau 6-54 Bloc de données de capacité de batterie (BCDB) 847
Tableau 6-55 Bloc de données d'obtention d'informations de fabricant (GMIDB) 848
Tableau 6-56 Bloc de données d'informations de fabricant (MIDB) 849
Tableau 6-57 Bloc de données de statut PPS (PPSSDB) 853
Tableau 6-58 Bloc de données de codes pays (CCDB) 854
Tableau 6-59 Bloc de données d'informations de pays (CIDB) 855
Tableau 6-60 Bloc de données étendues de capacités de destinataire (SCEDB) 856
Tableau 6-61 Valeurs de temporisation 871
Tableau 6-62 Temporisateurs 872
Tableau 6-63 Paramètres des compteurs 876
Tableau 6-64 Compteurs 876
Tableau 6-65 Réponse à un message entrant (hors VDM) 877
Tableau 6-66 Réponse à un VDM entrant 878
Tableau 6-67 Rejet de message 880
Tableau 6-68 Etats de la couche protocole 910
Tableau 6-69 Applicabilité des messages de contrôle 913
Tableau 6-70 Applicabilité des messages de données 914
Tableau 6-71 Applicabilité des messages étendus 915
Tableau 6-72 Applicabilité des commandes de VDM structuré 917
Tableau 6-73 Applicabilité des signaux de réinitialisation 917
Tableau 6-74 Applicabilité des signaux de permutation rapide des rôles 918
Tableau 6-75 Paramètres de valeur 918
Tableau 7-1 Description de séquence d'augmentation du courant 950
Tableau 7-2 Description de séquence d'augmentation de la tension 953
Tableau 7-3 Description de séquence d'augmentation de la tension et du courant 956
Tableau 7-4 Description de séquence d'augmentation de la tension et de diminution du courant 959
Tableau 7-5 Description de séquence de diminution de la tension et d'augmentation du courant 962
Tableau 7-6 Description de séquence de diminution du courant 965
Tableau 7-7 Description de séquence de diminution de la tension 968
Tableau 7-8 Description de séquence de diminution de la tension et du courant 971
Tableau 7-9 Description de séquence pour une permutation des rôles d'alimentation demandée par le destinataire 974
Tableau 7-10 Description de séquence pour une permutation des rôles d'alimentation demandée par la source 978
Tableau 7-11 Description de séquence de diminution de courant GotoMin 982
Tableau 7-12 Description de séquence pour une réinitialisation matérielle déclenchée par la source 985

Tableau 7-13 Description de séquence pour une réinitialisation matérielle déclenchée par le destinataire 988
Tableau 7-14 Description de séquence en cas d'absence de variation de courant ou de tension 991
Tableau 7-15 Description de séquence pour la permutation rapides de rôles 993
Tableau 7-16 Description de séquence d'augmentation de la tension d'alimentation électrique programmable 996
Tableau 7-17 Description de séquence de diminution de la tension d'alimentation électrique programmable 998
Tableau 7-18 Description de séquence de modification du PDO ou de l'APDO source 1000
Tableau 7-19 Description de séquence d'augmentation du courant en mode PPS 1003
Tableau 7-20 Description de séquence de diminution du courant en mode PPS 1005
Tableau 7-21 Description de séquence d'augmentation du courant en mode PPS 1007
Tableau 7-22 Paramètres électriques de la source 1007
Tableau 7-23 Paramètres électriques du destinataire 1013
Tableau 7-24 Paramètres électriques communs source/destinataire 1015
Tableau 8-1 Flux de messages de base 1027
Tableau 8-2 Problèmes potentiels dans le flux de messages de base 1029
Tableau 8-3 Flux de messages de base avec défaillance CRC 1031
Tableau 8-4 AMS interruptible et non interruptible 1032
Tableau 8-5 Etapes d'une négociation de puissance réussie 1035
Tableau 8-6 Etapes d'une négociation GotoMin 1040
Tableau 8-7 Etapes d'une négociation de puissance réussie 1043
Tableau 8-8 Etapes d'une réinitialisation logicielle 1046
Tableau 8-9 Etapes d'une réinitialisation des données initiée par le DFP, où le DFP est la source Vconn 1049
Tableau 8-10 Etapes de la réception d'une réinitialisation des données par le DFP, où le DFP est la source Vconn 1053
Tableau 8-11 Etapes d'une réinitialisation des données initiée par le DFP, où l'UFP est la source Vconn 1057
Tableau 8-12 Etapes de la réception d'une réinitialisation des données par le DFP, où l'UFP est la source Vconn 1062
Tableau 8-13 Etapes de la réinitialisation matérielle déclenchée par la source 1068
Tableau 8-14 Etapes de la réinitialisation matérielle déclenchée par le destinataire 1072
Tableau 8-15 Etapes de la réinitialisation matérielle déclenchée par la source – Réinitialisation longue du destinataire 1075
Tableau 8-16 Etapes d'une séquence réussie de permutation des rôles d'alimentation initiée par la source 1081
Tableau 8-17 Etapes d'une séquence réussie de permutation des rôles d'alimentation initiée par le destinataire 1087
Tableau 8-18 Etapes d'une séquence réussie de permutation rapide des rôles 1093
Tableau 8-19 Etapes de la permutation des rôles de transmission de données, déclenchée par un UFP fonctionnant en tant que destinataire 1097

Tableau 8-20 Etapes de la permutation des rôles de transmission de données, déclenchée par un UFP fonctionnant en tant que source 1100
Tableau 8-21 Etapes de la permutation des rôles de transmission de données, déclenchée par un DFP fonctionnant en tant que source 1103
Tableau 8-22 Etapes de la permutation des rôles de transmission de données, déclenchée par un DFP fonctionnant en tant que destinataire 1106
Tableau 8-23 Etapes de la permutation de la source VCONN de source à destinataire 1109
Tableau 8-24 Etapes de la permutation de la source VCONN de destinataire à source 1113
Tableau 8-25 Etapes de l'alerte de la source au destinataire 1117
Tableau 8-26 Etapes de l'alerte du destinataire à la source 1119
Tableau 8-27 Etapes d'une séquence où le destinataire obtient le statut de la source 1121
Tableau 8-28 Etapes d'une séquence où la source obtient le statut du destinataire 1124
Tableau 8-29 Etapes d'une séquence où le destinataire obtient le statut PPS de la source 1126
Tableau 8-30 Etapes d'une séquence où le destinataire obtient les capacités de la source 1129
Tableau 8-31 Etapes d'une séquence où la source double fonction obtient les capacités du destinataire double fonction en tant que source 1132
Tableau 8-32 Etapes d'une séquence où la source obtient les capacités du destinataire 1135
Tableau 8-33 Etapes d'une séquence où le destinataire double fonction obtient les capacités de la source double fonction en tant que destinataire 1138
Tableau 8-34 Etapes d'une séquence où le destinataire obtient les capacités étendues de la source 1141
Tableau 8-35 Etapes d'une séquence où la source double fonction obtient les capacités étendues du destinataire double fonction 1144
Tableau 8-36 Etapes d'une séquence où le destinataire obtient les capacités de la batterie de la source 1147
Tableau 8-37 Etapes d'une séquence où la source obtient les capacités de la batterie du destinataire 1150
Tableau 8-38 Etapes d'une séquence où le destinataire obtient le statut de la batterie de la source 1153
Tableau 8-39 Etapes d'une séquence où la source obtient le statut de la batterie du destinataire 1156
Tableau 8-40 Etapes d'une séquence où la source obtient les informations du fabricant du port du destinataire 1159
Tableau 8-41 Etapes d'une séquence où la source obtient les informations du fabricant du port du destinataire 1162
Tableau 8-42 Etapes d'une séquence où la source obtient les informations du fabricant de la batterie du destinataire 1165
Tableau 8-43 Etapes d'une séquence où la source obtient les informations du fabricant de la batterie du destinataire 1168
Tableau 8-44 Etapes d'une séquence où la source VCONN obtient les informations du fabricant du port du destinataire 1171
Tableau 8-45 Etapes d'une séquence où la source obtient les codes pays 1174
Tableau 8-46 Etapes d'une séquence où la source obtient les codes pays du destinataire 1177
Tableau 8-47 Etapes d'une séquence où une source VCONN obtient les codes pays du destinataire 1180
Tableau 8-48 Etapes d'une séquence où la source obtient les informations relatives au pays 1183

Tableau 8-49 Etapes d'une séquence où la source obtient les informations relatives au pays du destinataire....1186
Tableau 8-50 Etapes d'une séquence où une source VCONN obtient les informations relatives au pays du destinataire....1189
Tableau 8-51 Etapes d'une séquence où une source demande un échange de sécurité avec un destinataire....1192
Tableau 8-52 Etapes d'une séquence où un destinataire demande un échange de sécurité avec une source....1195
Tableau 8-53 Etapes d'une séquence où une source Vconn demande un échange de sécurité avec une fiche de câble....1198
Tableau 8-54 Etapes d'une séquence où une source demande un échange de mise à jour du micrologiciel avec un destinataire....1201
Tableau 8-55 Etapes d'une séquence où un destinataire demande un échange de mise à jour du micrologiciel avec une source....1204
Tableau 8-56 Etapes d'une séquence où une source Vconn demande un échange de mise à jour du micrologiciel avec une fiche de câble....1207
Tableau 8-57 Etapes de la découverte d'identité DFP vers UFP....1210
Tableau 8-58 Etapes de la découverte d'identité port source vers fiche de câble....1213
Tableau 8-59 Etapes de la découverte d'identité DFP vers fiche de câble....1216
Tableau 8-60 Etapes du mode d'entrée DFP vers UFP....1219
Tableau 8-61 Etapes du mode de sortie DFP vers UFP....1222
Tableau 8-62 Etapes du mode d'entrée DFP vers fiche de câble....1225
Tableau 8-63 Etapes du mode de sortie DFP vers fiche de câble....1228
Tableau 8-64 Etapes de la demande d'attention d'UFP à DFP....1230
Tableau 8-65 Etapes de l'essai du mode porteur BIST....1234
Tableau 8-66 Etapes de l'essai de données d'essai BIST....1237
Tableau 8-67 Etapes pour l'entrée en mode USB4 (valide) de l'UFP....1240
Tableau 8-68 Etapes pour l'entrée en mode USB4 (valide) de la fiche de câble....1243
Tableau 8-69 Etapes pour l'entrée en mode USB4 (non valide) de l'UFP....1245
Tableau 8-70 Etapes pour l'entrée en mode USB4 (non valide) de la fiche de câble....1247
Tableau 8-71 Etats du moteur de politique....1362
Tableau 9-1 Codes de type d'alimentation USB....1378
Tableau 9-2 Descripteur de capacité USB Power Delivery....1378
Tableau 9-3 Descripteur de capacité Battery Info....1380
Tableau 9-4 Descripteur de capacité PD Consumer Port....1380
Tableau 9-5 Descripteur de capacité PD Provider Port....1381
Tableau 9-6 Demandes d'alimentation USB....1383
Tableau 9-7 Codes de demande d'alimentation USB....1383
Tableau 9-8 Sélecteurs de caractéristique PD....1383
Tableau 9-9 Structure du statut de la batterie....1384
Tableau 9-10 Masque de réveil de la batterie....1386

Tableau 9-11 Codage de la politique de charge 1386
Tableau 10-1 Considérations pour les sources 1389
Tableau 10-2 Tensions normatives et courants minimaux 1389
Tableau 10-3 PDO d'alimentation fixe – Source 5 V 1391
Tableau 10-4 PDO d'alimentation fixe – Source 9V 1391
Tableau 10-5 PDO d'alimentation fixe – Source 15V 1391
Tableau 10-6 PDO d'alimentation fixe – Source 20V 1392
Tableau 10-7 PDO et APDO d'alimentation programmable en fonction de la PDP 1393
Tableau 10-8 Plages de tensions d'alimentation électrique programmable 1393
Tableau B-1 Alimentation externe fournie en aval 1400
Tableau B-2 Alimentation externe fournie en amont 1405
Tableau B-3 Rendu de puissance 1414
Tableau C-1 Exemple de demande de commande de découverte d'identité d'un initiateur 1428
Tableau C-2 Exemple de réponse de commande de découverte d'identité d'un répondeur qui est un câble actif 1428
Tableau C-3 Exemple de réponse de commande de découverte d'identité d'un répondeur qui est un hub 1430
Tableau C-4 Exemple de demande de commande de découverte de SVID d'un initiateur 1431
Tableau C-5 Exemple de réponse de commande de découverte de SVID d'un répondeur 1431
Tableau C-6 Exemple de demande de commande de découverte de modes de l'initiateur 1433
Tableau C-7 Exemple de réponse de commande de découverte de modes d'un répondeur 1433
Tableau C-8 Exemple de demande de commande d'entrée dans un mode d'un initiateur 1435
Tableau C-9 Exemple de réponse de commande d'entrée dans un mode d'un répondeur 1435
Tableau C-10 Exemple de demande de commande d'entrée dans un mode d'un initiateur 1436
Tableau C-11 Exemple de demande de commande de sortie d'un mode d'un initiateur 1437
Tableau C-12 Exemple de réponse de commande de sortie d'un mode d'un répondeur 1437
Tableau C-13 Exemple de demande de commande Attention d'un initiateur 1439
Tableau C-14 Exemple de demande de commande Attention d'un initiateur avec un VDO supplémentaire 1439
Tableau E-1: Tableau de séquence de configuration d'une permutation rapide des rôles (hub connecté à un adaptateur de puissance en premier lieu) 1451
Tableau E-2 Tableau de séquence de configuration d'une permutation rapide des rôles (hub connecté à l'ordinateur portable avant l'adaptateur de puissance) 1452
Tableau E-3 Tableau de séquence de déchargement lent de V Vbus (déchargement après l'envoi du message FR_Swap) 1455
Tableau E-4 Déchargement rapide de Vbus après déconnexion de l'adaptateur 1457

Figures

Figure 2-1 Structure logique des dispositifs aptes à l'alimentation électrique par port USB 700
Figure 2-2 Exemple de communication SOP' entre source VCONN et fiche(s) de câble 703

Figure 2-3 Pile de communications de l'alimentation électrique par port USB 710
Figure 2-4 Communication de l'alimentation électrique par port USB sur USB 711
Figure 2-5 Vue d'une architecture à haut niveau 712
Figure 5-1 Interprétation d'ensembles ordonnés 723
Figure 5-2 Ordre de transmission pour diverses tailles de données 725
Figure 5-3 Format de paquet pour l'alimentation électrique par port USB 725
Figure 5-4 Génération de code CRC 32 729
Figure 5-5 Format en ligne de Hard Reset 731
Figure 5-6 Format en ligne de Cable Reset 732
Figure 5-7 Exemple de BMC 733
Figure 5-8 Schéma fonctionnel d'émetteur BMC 734
Figure 5-9 Schéma fonctionnel de récepteur BMC 734
Figure 5-10 Début de préambule codé BMC 735
Figure 5-11 Emission ou réception d'une trame codée BMC et terminée par un 0 avec une dernière transition du niveau haut vers le niveau bas 735
Figure 5-12 Emission ou réception d'une trame codée BMC et terminée par un 1 avec une dernière transition du niveau haut vers le niveau bas 736
Figure 5-13 Emission ou réception d'une trame codée BMC et terminée par un 0 avec une dernière transition du niveau bas vers le niveau haut 736
Figure 5-14 Emission ou réception d'une trame codée BMC et terminée par un 1 avec une dernière transition du niveau bas vers le niveau haut 737
Figure 5-15 Masque BMC Tx "1" 738
Figure 5-16 Masque BMC Tx "0" 738
Figure 5-17 Masque BMC Rx "1" lorsque l'alimentation électrique est fournie 741
Figure 5-18 Masque BMC Rx "0" lorsque l'alimentation électrique est fournie 741
Figure 5-19 Masque BMC Rx "1" lorsque l'alimentation électrique est au point neutre 742
Figure 5-20 Masque BMC Rx "0" lorsque l'alimentation électrique est au point neutre 742
Figure 5-21 Masque BMC Rx "1" lorsque l'alimentation électrique est utilisée 743
Figure 5-22 Masque BMC Rx "0" lorsque l'alimentation électrique est utilisée 743
Figure 5-23 Modèle de charge d'émetteur pour BMC Tx à partir d'une source 744
Figure 5-24 Modèle de charge d'émetteur pour BMC Tx à partir d'un destinataire 745
Figure 5-25 Schéma de l'émetteur représentant zDriver 747
Figure 5-26 Chronogramme des écarts inter-trames 748
Figure 5-27 Exemple de configuration multipoint avec deux DRP 750
Figure 5-28 Exemple de configuration multipoint montrant un DFP et un UFP 751
Figure 5-29 Trame de données d'essai 753
Figure 6-1 Format de paquet d'alimentation électrique par port USB, avec charge utile de message de contrôle 754
Figure 6-2 Format de paquet d'alimentation électrique par port USB, avec charge utile de message de données 755

Figure 6-3 Format de paquet d'alimentation électrique par port USB, avec en-tête et charge utile de message étendu 755
Figure 6-4 Exemple de séquence Security_Request non fragmentée (bit Chunked = 0) 762
Figure 6-5 Exemple de transmission d'octets pour le message Security_Request de taille de données égale à 7 (le bit Chunked est à 0) 763
Figure 6-6 Exemple de transmission d'octets pour le message Security_Response de taille de données égale à 7 (le bit Chunked est à 0) 763
Figure 6-7 Exemple de séquence Security_Request fragmentée (bit Chunked = 1) 764
Figure 6-8 Exemple de message Security_Request de taille de données égale à 7 (le bit Chunked est à 1) 765
Figure 6-9 Exemple de fragment 0 de message Security_Request de taille de données égale à 30 (le bit Chunked est à 1) 765
Figure 6-10 Exemple de transmission d'octets pour une demande de fragment de message Security_Response (le bit Chunked est à 1) 766
Figure 6-11 Exemple de fragment 1 de message Security_Request de taille de données égale à 30 (le bit Chunked est à 1) 766
Figure 6-12 Exemple de message de capacités avec 2 objets de données d'alimentation 778
Figure 6-13 Message BIST 793
Figure 6-14 Message Vendor Defined 796
Figure 6-15 Réponse de commande Discover Identity 803
Figure 6-16 Réponse de commande Discover Identity pour un DRD 803
Figure 6-17 Exemple de réponse de découverte de SVID avec 3 SVID 820
Figure 6-18 Exemple de réponse de découverte de SVID avec 4 SVID 820
Figure 6-19 Exemple de réponse de découverte de SVID avec 12 SVID, suivi par une réponse vide 820
Figure 6-20 Exemple de réponse de découverte de modes pour un SVID à 3 modes 821
Figure 6-21 Séquence d'entrée dans un mode réussie 822
Figure 6-22 Séquence d'entrée dans un mode interrompue par un message Source Capabilities, puis réexécutée 823
Figure 6-23 Séquence d'entrée dans un mode échouée en raison d'un message NAK 824
Figure 6-24 Séquence de sortie de mode 825
Figure 6-25 Séquence de demande/réponse de commande Attention 826
Figure 6-26 Séquence de demande/réponse de commande 826
Figure 6-27 Processus d'entrée dans un mode/de sortie de mode 828
Figure 6-28 Message Battery_Status 830
Figure 6-29 Message d'alerte 831
Figure 6-30 Message Get_Country_Info 833
Figure 6-31 Message Enter_USB 834
Figure 6-32 Message Source_Capabilities_Extended 837
Figure 6-33 Message Status 842
Figure 6-34 Message de statut SOP'/SOP'' 845
Figure 6-35 Message Get_Battery_Cap 846

Figure 6-36 Message Get_Battery_Status 846
Figure 6-37 Message Battery_Capabilities 847
Figure 6-38 Message Get_Manufacturer_Info 848
Figure 6-39 Message Manufacturer_Info 849
Figure 6-40 Message Security_Request 850
Figure 6-41 Message Security_Response 851
Figure 6-42 Message Firmware_Update_Request 851
Figure 6-43 Message Firmware_Update_Response 852
Figure 6-44 Message PPS_Status 852
Figure 6-45 Message Country_Codes 854
Figure 6-46 Message Country_Info 855
Figure 6-47 Message Sink_Capabilities_Extended 855
Figure 6-48 Présentation des états 881
Figure 6-49 Références aux états 881
Figure 6-50 Architecture de fragmentation représentant le message et le flux de contrôle 883
Figure 6-51 Diagramme d'état de Rx fragmenté 885
Figure 6-52 Diagramme d'état de Tx fragmenté 889
Figure 6-53 Diagramme d'état du routeur de messages fragmentés 894
Figure 6-54 Diagramme d'états commun de transmission d'un message de la couche protocole 896
Figure 6-55 Diagramme d'états de transmission d'un message de la couche protocole Source 900
Figure 6-56 Diagramme d'états de transmission d'un message de la couche protocole destinataire 902
Figure 6-57 Réception d'un message de la couche protocole 903
Figure 6-58 Réinitialisation matérielle/de câble 906
Figure 7-1 Placement de la capacité de masse de la source 919
Figure 7-2 Enveloppe des transitions de tension positives 921
Figure 7-3 Enveloppe des transitions de tension négatives 922
Figure 7-4 Transitions de tension positives de la PPS 923
Figure 7-5 Transitions de tension négatives de la PPS 924
Figure 7-6 Ondulation prévue de la PPS relative à un octet de poids faible 924
Figure 7-7 Tension et limite de courant programmables PPS 926
Figure 7-8 iPpsCLOperatingDetail 927
Figure 7-9 Tension et limite de courant programmables PPS 928
Figure 7-10 Réponse à une réinitialisation matérielle de V_{BUS} et V_{CONN} Source 930
Figure 7-11 Application des limites vSrcNew et vSrcValid après tSrcReady 932
Figure 7-12 Surcharge du courant de crête de la source 934
Figure 7-13 Mesure du temps de maintien 936
Figure 7-14 Puissance V_{BUS} au cours d'une permutation rapide des rôles 937

Figure 7-15 Détection et temporisation de V_{BUS} lors d'une permutation rapide des rôles, avec V_{BUS} initiale (à la nouvelle source) > vSafe5V (min). 938
Figure 7-16 Détection et temporisation de VBUS lors d'une permutation rapide des rôles, avec VBUS initiale (à la nouvelle source) < vSafe5V (min). 939
Figure 7-17 Cycle d'alimentation V_{CONN} UFP dans une réinitialisation des données 940
Figure 7-18 Cycle d'alimentation V_{CONN} DFP dans une réinitialisation des données 941
Figure 7-19 Placement de la capacité de masse du destinataire 942
Figure 7-20 Diagramme de transition pour l'augmentation du courant 948
Figure 7-21 Diagramme de transition pour l'augmentation de la tension 951
Figure 7-22 Diagramme de transition pour l'augmentation de la tension et du courant 954
Figure 7-23 Diagramme de transition pour l'augmentation de la tension et la diminution du courant 957
Figure 7-24 Diagramme de transition pour la diminution de la tension et l'augmentation du courant 960
Figure 7-25 Diagramme de transition pour la diminution du courant 963
Figure 7-26 Diagramme de transition pour la diminution de la tension 966
Figure 7-27 Diagramme de transition pour la diminution de la tension et du courant 969
Figure 7-28 Diagramme de transition pour une permutation des rôles d'alimentation demandée par le destinataire 972
Figure 7-29 Diagramme de transition pour une permutation des rôles d'alimentation demandée par la source 976
Figure 7-30 Diagramme de transition pour la diminution de courant GotoMin 980
Figure 7-31 Diagramme de transition pour une réinitialisation matérielle déclenchée par la source 983
Figure 7-32 Diagramme de transition pour une réinitialisation matérielle déclenchée par le destinataire 986
Figure 7-33 Diagramme de transition en cas d'absence de variation de courant ou de tension 989
Figure 7-34 Diagramme de transition pour la permutation rapide des rôles 992
Figure 7-35 Diagramme de transition pour l'augmentation de la tension d'alimentation électrique programmable 995
Figure 7-36 Diagramme de transition pour la diminution de la tension d'alimentation électrique programmable 997
Figure 7-37 Diagramme de transition pour la modification du PDO ou de l'APDO source 999
Figure 7-38 Diagramme de transition pour l'augmentation du courant en mode PPS 1002
Figure 7-39 Diagramme de transition pour la diminution du courant en mode PPS 1004
Figure 7-40 Diagramme de transition en cas d'absence de variation de courant ou de tension en mode PPS 1006
Figure 8-1 Exemple d'écrans en guirlande 1023
Figure 8-2 Echange de messages de base (réussi) 1027
Figure 8-3 Flux de messages de base indiquant de possibles erreurs 1028
Figure 8-4 Flux de messages de base avec CRC erroné suivi d'une relance 1030
Figure 8-5 Négociation réussie d'alimentation fixe, variable ou par batterie 1034
Figure 8-6 Opération GotoMin réussie 1039
Figure 8-7 Maintien de PPS 1042

Figure 8-8 Réinitialisation logicielle 1045
Figure 8-9 Réinitialisation des données initiée par le DFP, où le DFP est la source Vconn 1048
Figure 8-10 Réception d'une réinitialisation par le DFP, où le DFP est la source Vconn 1052
Figure 8-11 Réinitialisation des données initiée par le DFP, où l'UFP est la source Vconn 1056
Figure 8-12 Réception d'une réinitialisation par le DFP, où l'UFP est la source Vconn 1061
Figure 8-13 Réinitialisation matérielle déclenchée par la source 1066
Figure 8-14 Réinitialisation matérielle déclenchée par le destinataire 1070
Figure 8-15 Réinitialisation déclenchée par la source - Réinitialisation longue du destinataire 1074
Figure 8-16 Séquence réussie de permutation des rôles d'alimentation initiée par la source 1079
Figure 8-17 Séquence réussie de permutation des rôles d'alimentation initiée par le destinataire 1085
Figure 8-18 Séquence réussie de permutation rapide des rôles 1091
Figure 8-19 Permutation des rôles de transmission de données, déclenchée par un UFP fonctionnant en tant que destinataire 1096
Figure 8-20 Permutation des rôles de transmission de données, déclenchée par un UFP fonctionnant en tant que source 1099
Figure 8-21 Permutation des rôles de transmission de données, déclenchée par un DFP fonctionnant en tant que source 1102
Figure 8-22 Permutation des rôles de transmission de données, déclenchée par un DFP fonctionnant en tant que destinataire 1105
Figure 8-23 Permutation de la source VCONN de source à destinataire 1108
Figure 8-24 Permutation de la source VCONN de destinataire à source 1112
Figure 8-25 Alerte de la source au destinataire 1116
Figure 8-26 Alerte du destinataire à la source 1118
Figure 8-27 Destinataire obtenant le statut de la source 1120
Figure 8-28 Source obtenant le statut du destinataire 1123
Figure 8-29 Destinataire obtenant le statut PPS de la source 1125
Figure 8-30 Destinataire obtenant les capacités de la source 1128
Figure 8-31 Source double fonction obtenant les capacités du destinataire double fonction en tant que source 1131
Figure 8-32 Source obtenant les capacités du destinataire 1134
Figure 8-33 Destinataire double fonction obtenant les capacités de la source double fonction en tant que destinataire 1137
Figure 8-34 Destinataire obtenant les capacités étendues de la source 1140
Figure 8-35 Source double fonction obtenant les capacités étendues du destinataire double fonction 1143
Figure 8-36 Destinataire obtenant les capacités de la batterie de la source 1146
Figure 8-37 Source obtenant les capacités de la batterie du destinataire 1149
Figure 8-38 Destinataire obtenant le statut de la batterie de la source 1152
Figure 8-39 Source obtenant le statut de la batterie du destinataire 1155
Figure 8-40 Source obtenant les informations du fabricant du port du destinataire 1158
Figure 8-41 Destinataire obtenant les informations du fabricant du port de la source 1161

Figure 8-42 Source obtenant les informations du fabricant de la batterie du destinataire 1164
Figure 8-43 Destinataire obtenant les informations du fabricant de la batterie de la source 1167
Figure 8-44 Source VCONN obtenant les informations du fabricant d'une fiche de câble 1170
Figure 8-45 Source obtenant les codes pays du destinataire 1173
Figure 8-46 Destinataire obtenant les codes pays de la source 1176
Figure 8-47 Source VCONN obtenant les codes pays d'une fiche de câble 1179
Figure 8-48 Source obtenant les informations relatives au pays du destinataire 1182
Figure 8-49 Destinataire obtenant les informations relatives au pays de la source 1185
Figure 8-50 Source VCONN obtenant les informations relatives au pays d'une fiche de câble 1188
Figure 8-51 Source demandant un échange de sécurité avec le destinataire 1191
Figure 8-52 Destinataire demandant un échange de sécurité avec la source 1194
Figure 8-53 Source Vconn demandant un échange de sécurité avec une fiche de câble 1197
Figure 8-54 Source demandant un échange de mise à jour du micrologiciel avec le destinataire 1200
Figure 8-55 Destinataire demandant un échange de mise à jour du micrologiciel avec la source 1203
Figure 8-56 Source Vconn demandant un échange de mise à jour du micrologiciel avec une fiche de câble 1206
Figure 8-57 Découverte d'identité DFP vers UFP 1209
Figure 8-58 Découverte d'identité port source vers fiche de câble 1212
Figure 8-59 Découverte d'identité DFP vers fiche de câble 1215
Figure 8-60 Mode d'entrée DFP vers UFP 1218
Figure 8-61 Mode de sortie DFP vers UFP 1221
Figure 8-62 Mode d'entrée DFP vers fiche de câble 1224
Figure 8-63 Mode de sortie DFP vers fiche de câble 1227
Figure 8-64 Demande d'attention d'UFP à DFP 1230
Figure 8-65 Essai du mode porteur BIST 1232
Figure 8-66 Essai de données d'essai BIST 1235
Figure 8-67 UFP qui entre en mode USB4 (valide) 1239
Figure 8-68 Fiche de câble qui entre en mode USB4 (valide) 1242
Figure 8-69 UFP qui entre en mode USB4 (non valide) 1244
Figure 8-70 Fiche de câble qui entre en mode USB4 (non valide) 1246
Figure 8-71 Présentation des états 1248
Figure 8-72 Références aux états 1249
Figure 8-73 Exemple de référence d'état avec conditions 1249
Figure 8-74 Exemple de référence d'état avec la même entrée et la même sortie 1249
Figure 8-75 Diagramme d'états du port source du moteur de politique 1251
Figure 8-76 Diagramme d'état d'un port destinataire 1260
Figure 8-77 Diagramme d'état de réinitialisation logicielle et d'erreur de protocole du port source 1267
Figure 8-78 Diagramme de réinitialisation logicielle et d'erreur de protocole du port destinataire 1269

Figure 8-79 Diagramme d'états d'un message Data_Reset du DFP 1271
Figure 8-80 Diagramme d'états d'un message Data_Reset de l'UFP 1274
Figure 8-81 Diagramme d'état de messages non pris en charge d'un port source 1277
Figure 8-82 Diagramme d'état de messages non pris en charge d'un port destinataire 1278
Figure 8-83 Diagramme d'état PING du port source 1280
Figure 8-84 Diagramme d'état d'alerte de la source d'un port source 1280
Figure 8-85 Diagramme d'état d'alerte de la source d'un port destinataire 1281
Figure 8-86 Diagramme d'état d'alerte du destinataire d'un port destinataire 1282
Figure 8-87 Diagramme d'état d'alerte du destinataire d'un port source 1282
Figure 8-88 Diagramme d'état d'un port destinataire obtenant les capacités étendues de la source 1283
Figure 8-89 Diagramme d'état d'une source donnant les capacités étendues de la source 1284
Figure 8-90 Diagramme d'état du port destinataire obtenant le statut de la source 1285
Figure 8-91 Diagramme d'état de la source donnant le statut de la source 1285
Figure 8-92 Diagramme d'état du port source obtenant le statut du destinataire 1286
Figure 8-93 Diagramme d'état du destinataire donnant le statut du destinataire 1287
Figure 8-94 Diagramme d'état du port destinataire obtenant le statut PPS de la source 1288
Figure 8-95 Diagramme d'état de la source donnant le statut PPS de la source 1289
Figure 8-96 Diagramme d'état d'obtention des capacités de la batterie 1289
Figure 8-97 Diagramme d'état d'indication des capacités de la batterie 1290
Figure 8-98 Diagramme d'état d'obtention du statut de la batterie 1291
Figure 8-99 Diagramme d'état d'indication du statut de la batterie 1292
Figure 8-100 Diagramme d'état d'obtention des informations du fabricant 1293
Figure 8-101 Diagramme d'état d'indication des informations du fabricant 1293
Figure 8-102 Diagramme d'état d'obtention des codes pays 1294
Figure 8-103 Diagramme d'état d'indication des codes pays 1295
Figure 8-104 Diagramme d'état d'obtention des informations relatives aux pays 1296
Figure 8-105 Diagramme d'état d'indication des informations relatives aux pays 1297
Figure 8-106 Diagramme d'états du message Enter_USB d'un DFP 1297
Figure 8-107 Diagramme d'états du message Enter_USB d'un UFP 1298
Figure 8-108 Diagramme d'état d'envoi d'une demande de sécurité 1299
Figure 8-109 Diagramme d'état d'envoi d'une réponse de sécurité 1300
Figure 8-110 Diagramme d'état de réponse de sécurité reçue 1301
Figure 8-111 Diagramme d'état d'envoi d'une demande de mise à jour du micrologiciel 1302
Figure 8-112 Diagramme d'état d'envoi d'une réponse de mise à jour du micrologiciel 1302
Figure 8-113 Diagramme d'état de réponse reçue de mise à jour du micrologiciel 1303
Figure 8-114 Diagramme d'état de permutation des rôles de transmission de données de DFP à UFP 1305
Figure 8-115 Diagramme d'état de permutation des rôles de transmission de données d'UFP à DFP 1308

Figure 8-116 Diagramme d'état du port double fonction lors de la permutation des rôles d'alimentation de source à destinataire 1311
Figure 8-117 Diagramme d'état du port double fonction lors de la permutation des rôles d'alimentation de destinataire à source 1315
Figure 8-118 Diagramme d'état du port double fonction lors de la permutation rapide des rôles de source à destinataire 1319
Figure 8-119 Diagramme d'état du port double fonction lors de la permutation rapide des rôles de destinataire à source 1322
Figure 8-120 Diagramme (d'une source) double fonction obtenant les capacités de la source 1325
Figure 8-121 Diagramme (d'une source) double fonction indiquant les capacités du destinataire 1326
Figure 8-122 Diagramme d'état (d'un destinataire) double fonction obtenant les capacités du destinataire 1326
Figure 8-123 Diagramme d'état (d'un destinataire) double fonction indiquant les capacités de la source 1327
Figure 8-124 Diagramme d'état (d'une source) double obtenant les capacités étendues de la source 1328
Figure 8-125 Diagramme (d'une source) double fonction indiquant les capacités du destinataire 1329
Figure 8-126 Diagramme d'état de permutation de VCONN 1330
Figure 8-127 Diagrammes d'états de découverte d'identité de VDM structuré de l'initiateur au port 1334
Figure 8-128 Diagramme d'état de découverte des SVID de VDM structuré d'un initiateur 1335
Figure 8-129 Diagramme d'état de découverte des modes de VDM d'un initiateur 1337
Figure 8-130 Diagramme d'état de demande d'attention de VDM d'un initiateur 1338
Figure 8-131 Diagramme d'état de découverte d'identité de VDM structuré d'un répondeur 1339
Figure 8-132 Diagramme d'état de découverte de SVID de VDM structuré d'un répondeur 1340
Figure 8-133 Diagramme d'état de découverte des modes de VDM structuré d'un répondeur 1342
Figure 8-134 Diagramme d'état de réception d'une demande d'attention de VDM structuré 1343
Figure 8-135 Diagramme d'état d'entrée dans un mode de VDM d'un DFP 1344
Figure 8-136 Diagramme d'état de sortie d'un mode de VDM d'un DFP 1346
Figure 8-137 Diagramme d'état d'entrée dans un mode de VDM structuré d'un UFP 1347
Figure 8-138 Diagramme d'état de sortie d'un mode de VDM structuré d'un UFP 1349
Figure 8-139 Diagramme d'état de VDM de câble prêt 1350
Figure 8-140 Diagramme d'état d'une réinitialisation logicielle de la fiche de câble 1351
Figure 8-141 Diagramme d'état d'une réinitialisation matérielle de la fiche de câble 1352
Figure 8-142 Diagramme d'états d'une réinitialisation logicielle de la source VCONN ou d'une réinitialisation de câble d'une fiche de câble ou d'un VPD 1353
Figure 8-143 Diagramme d'état de découverte d'identité de VDM structuré au démarrage de la source 1355
Figure 8-144 Diagramme d'état d'entrée dans un mode de VDM structuré d'une fiche de câble 1357
Figure 8-145 Diagramme d'état de sortie d'un mode de VDM structuré d'une fiche de câble 1359
Figure 8-146 Diagramme d'état de mode porteur BIST 1360
Figure 9-1 Exemple de topologie d'alimentation USB 1370
Figure 9-2 Mise en correspondance de la topologie PD vers USB 1371

Figure 9-3 Passage de l'état USB Attached à l'état USB Powered 1372
Figure 9-4 Passage de tout état USB à l'état USB Attached (fonctionnement en tant que consommateur) 1373
Figure 9-5 Passage de tout état USB à l'état USB Attached (fonctionnement en tant que fournisseur) 1374
Figure 9-6 Passage de tout état USB à l'état USB Attached (après une permutation des rôles de données USB Type-C) 1374
Figure 9-7 Pile de logiciels sur un système d'exploitation compatible avec l'alimentation USB 1375
Figure 9-8 Enumération d'un dispositif PDUSB 1376
Figure 10-1 Présentation de la règle d'alimentation de la source 1390
Figure 10-2 Exemple de règle d'alimentation de la source 1390
Figure B-1 Alimentation externe fournie en aval 1399
Figure B-2 Alimentation externe fournie en amont 1404
Figure B-3 Rendu de puissance 1413
Figure D-1 Bloc de circuits du récepteur de différences finies BMC 1441
Figure D-2 Bruit BMC en courant alternatif et en courant continu de VBUS au niveau du destinataire de l'alimentation 1442
Figure D-3 Modèle de signaux BMC (a) sans bruit *[USB 2.0]* SE0 (b) avec bruit *[USB 2.0]* SE0 1442
Figure D-4 Dérivée du signal BMC mise à l'échelle avec vitesse d'échantillonnage de 50 ns 1443
Figure D-5 Signal BMC et résultat des différences finies avec différents intervalles de temps 1444
Figure D-6 Résultat des différences finies (ligne en pointillés) et du détecteur de front (ligne continue) 1444
Figure D-7 Zone de bruit et zone de détection du récepteur BMC 1445
Figure D-8 Bloc de circuits du récepteur de soustraction BMC 1445
Figure D-9 (a) Sortie de LPF1 et de LPF2 (b) Soustraction des sorties LPF1 et LPF2 1446
Figure D-10 Sortie de BMC LPF1 (courbe bleue en pointillés) et du soustracteur (courbe rouge) 1447
Figure E-1 Exemple de système apte à la FRS 1448
Figure E-2 Déchargement lent de V_{BUS} 1449
Figure E-3 Déchargement rapide de V_{BUS} 1450
Figure E-4 Diagramme de séquence de déchargement lent de V_{BUS} (déchargement après l'envoi du message FR_Swap) 1454

1. Introduction

USB a évolué d'une interface de données capable de fournir une alimentation électrique limitée à une source d'alimentation primaire dotée d'une interface de données. Aujourd'hui, de nombreux dispositifs se rechargent ou reçoivent leur alimentation électrique par les ports USB que contiennent les ordinateurs portables, les voitures, les avions, et même les prises murales. USB est devenu une prise électrique omniprésente pour nombre de petits dispositifs tels que les téléphones portables, les lecteurs MP3 et autres dispositifs portatifs. Les utilisateurs ont besoin d'USB pour répondre à leurs exigences non seulement en matière de données, mais aussi pour alimenter ou recharger leurs dispositifs simplement, souvent sans avoir besoin de charger un pilote, pour assurer des fonctions USB "traditionnelles".

Cependant, il existe encore de nombreux dispositifs qui nécessitent un branchement mural supplémentaire pour leur alimentation, ou dont le courant de fonctionnement dépasse le courant assigné d'un port USB. Les réglementations internationales exigent une gestion toujours meilleure de l'énergie, en raison des problèmes écologiques et pratiques liés à la disponibilité de l'énergie électrique. Les réglementations limitent la quantité d'électricité disponible sur une prise murale, ce qui a conduit à un besoin éminent d'optimiser l'utilisation de l'énergie. La spécification de l'alimentation électrique par port USB dispose du potentiel pour réduire le plus possible le gaspillage en devenant une norme pour le chargement des dispositifs non couverts par ***[USBBC 1.2]***.

L'utilisation accrue des solutions sans fil tend à faire disparaître le câblage pour la transmission de données, mais les chargeurs filaires restent nécessaires. De plus, les exigences de conception industrielle amènent la connectivité filaire à offrir beaucoup plus à travers un même connecteur.

L'alimentation électrique par port USB est conçue pour permettre une fonctionnalité maximale du port USB, en assurant une alimentation électrique plus souple coexistant avec les données sur un même câble. Elle a pour but d'interopérer avec l'écosystème USB existant et de le compléter; en augmentant les niveaux de puissance des normes USB existantes, par exemple pour le chargement des batteries, en permettant de nouveaux cas d'utilisation avec des niveaux de puissance supérieurs, notamment pour les disques durs (DD) et les imprimantes à alimentation USB.

Avec l'alimentation électrique par port USB, le sens de l'alimentation n'est plus fixe. Cela permet au produit disposant de l'alimentation électrique (hôte ou périphérique) de la fournir. Par exemple, un écran alimenté par une prise murale peut alimenter ou charger un ordinateur portable. En variante, les transformateurs ou chargeurs USB sont capables d'alimenter les ordinateurs portables et autres dispositifs alimentés par batteries par le biais de leurs ports USB, qui fournissent habituellement l'alimentation électrique.

L'alimentation électrique par port USB permet aux hubs de devenir les moyens d'optimisation de la gestion de l'énergie à travers de multiples périphériques en permettant à chaque dispositif de ne prélever que la puissance dont il a besoin et d'en prélever davantage lorsqu'une application donnée l'exige. Par exemple, les appareils alimentés par batterie peuvent obtenir un courant de charge supérieur, puis le retransmettre temporairement, lorsque le DD de l'utilisateur exige une accélération. ***En option***, les hubs peuvent communiquer avec le PC pour permettre une gestion encore plus intelligente et souple de la puissance, automatiquement ou avec un certain degré d'intervention de l'utilisateur.

L'alimentation électrique par port USB permet à des cas d'utilisation à faible puissance, comme pour les casques, de négocier uniquement la puissance dont ils ont besoin. Ceci offre une solution simple permettant aux dispositifs USB de fonctionner à leurs niveaux de puissance optimaux.

La spécification de l'alimentation électrique par port USB, en plus d'offrir des mécanismes de négociation de la puissance, peut également servir de canal à bande latérale pour des échanges de messages normalisés ou définis par le fournisseur. L'alimentation électrique permet des modes de fonctionnement alternatifs, en assurant les mécanismes permettant de découvrir des modes alternatifs, d'y accéder et d'en sortir. La spécification permet également de découvrir les capacités des câbles, notamment les vitesses et les intensités de courant prises en charge.

1.1 Vue d'ensemble

La présente spécification définit la façon dont les dispositifs USB peuvent négocier davantage de courant et/ou des tensions plus élevées ou plus basses sur le câble USB (en se servant du fil CC USB Type-C®

comme canal de communications) que ce qui est défini dans les spécifications *[USB 2.0]*, *[USB 3.2]*, *[USB Type-C 2.0]* ou *[USBBC 1.2]*. Elle permet à des dispositifs dont les exigences de puissance sont supérieures à celles auxquelles la spécification actuelle peut satisfaire d'obtenir la puissance dont ils ont besoin pour fonctionner à partir de la tension V_{BUS} et de négocier avec des sources d'alimentation externes (par exemple, des adaptateurs). De plus, elle permet à une source et à un destinataire de permuter les rôles d'alimentation de sorte qu'un dispositif puisse alimenter l'hôte en électricité. Par exemple, un écran peut alimenter un ordinateur portable pour charger sa batterie.

La spécification de l'alimentation électrique par port USB est guidée par les principes suivants:

- elle s'applique parfaitement aux anciens dispositifs USB;
- elle est compatible avec les câbles USB existants conformes à la spécification;
- elle réduit le plus possible les risques de dommages dus aux câbles non conformes (par exemple, les câbles en "Y", etc.);
- elle est optimisée pour des mises en œuvre à faible coût.

La présente spécification définit des mécanismes permettant de découvrir des modes, d'y entrer et d'en sortir, que ces modes soient définis par une norme ou par un fournisseur particulier. Ces modes peuvent être pris en charge par le port partenaire ou par un câble reliant les deux utilisateurs du port.

La spécification définit des mécanismes permettant de découvrir les capacités de câbles pouvant communiquer en utilisant l'alimentation électrique.

Cette spécification ajoute un mécanisme de permutation des rôles de transmission de données de sorte que le port amont devienne le port aval, et inversement. Elle permet aussi de permuter l'extrémité d'alimentation V_{CONN} d'un câble alimenté.

Pour permettre une charge optimale, la spécification définit deux mécanismes qu'un chargeur USB peut annoncer pour que le dispositif les utilise:

1. une liste de tensions fixes, chacune ayant un courant maximal. Le dispositif choisit une tension et un courant dans la liste. Il s'agit du modèle traditionnel employé par les dispositifs qui utilisent l'électronique interne pour gérer la charge de leurs batteries, y compris modifier la tension et le courant réellement fournis à la batterie. L'effet non souhaitable de ce modèle est que le circuit de charge génère de la chaleur, ce qui peut être problématique pour les dispositifs de faible encombrement ;
2. une liste de plages de tensions programmables, chacune ayant un courant maximal (PPS). Le dispositif demande une tension (par incréments de 20 mV) située dans la plage annoncée et un courant maximal. Le chargeur USB fournit la tension demandée jusqu'à ce que le courant maximal soit atteint; le chargeur USB réduit alors sa tension de sortie, de manière à ne pas fournir davantage que le courant maximal demandé. Au cours de la partie du cycle de charge à courant élevé, le chargeur USB peut être directement connecté (à l'aide d'un dispositif de sécurité approprié) à la batterie. Ce modèle est utilisé par les dispositifs qui souhaitent réduire le plus possible l'impact thermique de leurs circuits de charge internes.

1.2 Objet

La spécification de l'alimentation électrique par port USB définit un système d'alimentation électrique qui couvre tous les éléments d'un système USB, y compris: hôtes, dispositifs, hubs, chargeurs et assemblages de câbles. La présente spécification décrit l'architecture, les protocoles, le comportement de l'alimentation électrique, les connecteurs et le câblage nécessaires à la gestion de l'alimentation électrique sur USB jusqu'à 100 W. La présente spécification est destinée à être entièrement compatible avec les infrastructures USB existantes et à en assurer une extension. Il est prévu que cette spécification offre aux OEM de systèmes, ainsi qu'aux développeurs d'alimentations électriques et de périphériques, une souplesse appropriée pour une polyvalence des produits et leur différenciation sur le marché, sans perdre en compatibilité ascendante.

L'alimentation électrique par port USB est conçue pour fonctionner indépendamment des mécanismes définis par le bus USB existant et servant à négocier la puissance, qui sont les suivants:

[USB 2.0], *[USB 3.2]* demandes de capacité "dans la bande" pour les interfaces de forte puissance;
[USBBC 1.2] mécanismes d'alimentation de puissance plus élevée (non rendus obligatoires par la présente spécification);

[USB Type-C 2.0] mécanismes d'alimentation de puissance plus élevée.

Les conditions de fonctionnement initiales restent le fonctionnement USB par défaut défini dans *[USB 2.0]*, *[USB 3.2]*, *[USB Type-C 2.0]* ou *[USBBC 1.2]*.

Le port DFP fournit ***vSafe5V*** sur V_{BUS}.
Le port UFP consomme la puissance fournie par V_{BUS}.

1.3 Domaine d'application

La présente spécification se veut une extension des spécifications *[USB 2.0]*, *[USB 3.2]*, *[USB Type-C 2.0]* et *[USBBC 1.2]* existantes. Elle traite uniquement des éléments nécessaires à la mise en œuvre de l'alimentation électrique par port USB. Elle vise les fournisseurs d'alimentation électrique, les fabricants de plateformes, dispositifs et assemblages de câbles *[USB 2.0]*, *[USB 3.2]*, *[USB Type-C 2.0]* et *[USBBC 1.2]*.

Des informations à caractère ***Normatif(s/ve/ves)*** sont fournies pour permettre l'interopérabilité des composants conçus selon cette spécification. Les informations à caractère purement informatif, le cas échéant, donnent un exemple de mise en œuvre possible de la conception proposée.

1.4 Conventions

1.4.1 Ordre de priorité

En cas de contradiction entre le texte, les figures et les tableaux, la priorité ***Devoir/Doit/Doivent*** être donnée aux tableaux, aux figures, et enfin au texte.

1.4.2 Mots-clés

Les mots-clés suivants permettent de différencier les niveaux d'exigences et d'options.

1.4.2.1 Normatif conditionnel

Le mot-clé ***Normatif(s/ve/ves) conditionnel(s/le/les)*** est utilisé pour indiquer une fonctionnalité qui est obligatoire lorsqu'une autre fonctionnalité associée a été mise en œuvre. Les concepteurs sont tenus de mettre en œuvre l'ensemble de ces exigences lorsque les fonctionnalités dépendantes ont été mises en œuvre afin d'assurer l'interopérabilité avec les autres dispositifs conformes.

1.4.2.2 Déconseillé

Le mot-clé ***Déconseillé(s/ée/ées)*** est utilisé pour indiquer une fonctionnalité prise en charge dans les précédentes versions de la spécification, mais qui n'est désormais plus prise en charge.

1.4.2.3 Rejeté

Rejeter, ***Rejette*** et ***Rejeté(s/ée/ées)*** sont des mots-clés équivalents qui indiquent qu'un paquet ***Devoir/Doit/Doivent***, à réception, être éliminé par la couche PHY et non transmis à la couche protocole pour traitement. Aucun message ***GoodCRC*** ne ***Devoir/Doit/Doivent*** être envoyé en réponse au paquet.

1.4.2.4 Ignoré

Ignorer, ***Ignore*** et ***Ignoré(s/ée/ées)*** sont des mots-clés équivalents qui indiquent des messages ou des champs de message qui, à réception, ne ***Devoir/Doit/Doivent*** entraîner aucune action particulière de la part du destinataire. Un message ***Ignoré(s/ée/ées)*** ne ***Devoir/Doit/Doivent*** pas avoir d'autre résultat que de retourner un message ***GoodCRC*** pour acquitter la réception du message. Un message comportant un champ ***Ignoré(s/ée/ées)*** ***Devoir/Doit/Doivent*** être traité normalement, sauf pour les actions concernant le champ ***Ignoré(s/ée/ées)***.

1.4.2.5 Invalide

Le terme ***Invalide(s)*** est un mot-clé qui, lorsqu'il est relatif à un paquet, indique que l'usage ou les champs du paquet ne rentrent pas dans l'usage de la spécification définie. Lorsque le terme ***Invalide(s)*** est relatif à

un contrat explicite, il indique qu'un contrat explicite précédemment établi ne peut plus être mis à jour par la source. Lorsque le terme ***Invalide(s)*** est relatif à des codes K individuels ou à des séquences de codes K, il indique que la signalisation reçue ne rentre pas dans la spécification définie.

1.4.2.6 Pouvoir/Peut/Peuvent

Les mots-clés ***Pouvoir/Peut/Peuvent*** sont utilisés pour indiquer un choix sans préférence implicite.

1.4.2.7 Peut/Peuvent ne pas

Les mots-clés ***Peut ne pas/Peuvent ne pas*** ont une signification inverse à ***Pouvoir/Peut/Peuvent***. Ils indiquent la possibilité de ne pas mettre en œuvre une fonction donnée, sans préférence implicite.

1.4.2.8 N/A

Le mot-clé ***N/A*** est utilisé pour indiquer qu'un champ (ou une valeur) n'est pas applicable, qu'il n'a pas de valeur définie et qu'il ***Ne doit/doivent pas*** être vérifié ou utilisé par le récepteur.

1.4.2.9 Facultatif/En option/Normatif facultatif

Les mots-clés ***Facultatif(s/ve/ves), En option*** et ***Normatif(s/ve/ves) facultatif(s/ve/ves)*** sont utilisés pour décrire les fonctionnalités non obligatoires de la présente spécification. Toutefois, si une fonctionnalité ***Facultative*** est mise en œuvre, la fonctionnalité ***Devoir/Doit/Doivent*** être mise en œuvre comme défini par la présente spécification.

1.4.2.10 Réservé

Le mot-clé ***Réservé(s/ée/ées)*** est utilisé pour indiquer les valeurs réservées d'un bit, d'un octet, d'un mot, d'un champ ou d'un code qui sont mises de côté pour une normalisation future. Leur utilisation et leur interprétation ***Pouvoir/Peut/Peuvent*** être spécifiées par de futures extensions à la présente spécification; sauf indication contraire, elles ***Ne doit/doivent pas*** être utilisées ou modifiées par la mise en œuvre d'un fournisseur. Un bit, un octet, un mot ou un champ ***Réservé(s/ée/ées) Devoir/Doit/Doivent*** être défini sur 0 par l'expéditeur et ***Devoir/Doit/Doivent*** être ***Ignoré(s/ée/ées)*** par le récepteur. Les valeurs de champs ***Réservé(s/ée/ées) Ne doit/doivent pas*** être envoyées par l'expéditeur et ***Devoir/Doit/Doivent*** être ***Ignoré(s/ée/ées)*** par le récepteur.

1.4.2.11 Devoir/Doit/Doivent/Normatif(s/ve/ves)

Les mots-clés ***Devoir/Doit/Doivent*** et ***Normatif(s/ve/ves)*** sont utilisés de manière équivalente pour indiquer une exigence obligatoire. Les concepteurs sont tenus de mettre en œuvre l'ensemble de ces exigences afin d'assurer l'interopérabilité avec les autres dispositifs conformes.

1.4.2.12 Ne doit/doivent pas

Les mots-clés ***Ne doit/doivent pas*** ont une signification inverse à ***Devoir/Doit/Doivent***, indiquant une opération non conforme.

1.4.2.13 Il convient de/que

Le mot-clé ***Il convient d'/de/qu'/que*** est utilisé pour indiquer la flexibilité d'un choix avec alternative préférentielle. Il équivaut à l'expression "Il est recommandé de".

1.4.2.14 Il convient de ne pas

Le mot-clé ***Il convient de ne pas*** a une signification inverse à ***Il convient d'/de/qu'/que***. Il équivaut à l'expression "Il est recommandé de ne pas...".

1.4.2.15 Valide

Le mot-clé ***Valide(s)*** a une signification inverse à ***Invalide(s)***, indiquant soit un paquet ou une signalisation qui relève de la spécification définie, soit un contrat explicite pouvant être mis à jour par la source.

1.4.3 Numérotation

Les nombres immédiatement suivis de la lettre "b" en minuscule (par exemple 01b) sont des valeurs binaires. Les nombres immédiatement suivis de la lettre "B" en majuscule sont des valeurs exprimées en octets. Les nombres immédiatement suivis de la lettre "h" en minuscule (par exemple 3Ah) ou précédés par "0x" (par exemple 0xFF00) sont des valeurs hexadécimales. Les nombres qui ne sont pas immédiatement suivis des lettres "b", "B" ou "h" sont des valeurs décimales.

1.5 Documents connexes

[USB 2.0] – Spécification Universal Serial Bus, révision 2.0, plus ECN et errata http://www.usb.org/developers/docs/usb20_docs/.
[USB 3.2] – Spécification Universal Serial Bus 3.2, révision 1.0, 22 septembre 2017. www.usb.org/developers/docs.
[USBTypeCAuthentication 1.0] – Spécification d'authentification pour Universal Serial Bus Type-C, révision 1.0, 25 mars 2016. www.usb.org/developers/docs.
[USBPDFirmwareUpdate 1.0] – Spécification de mise à jour du micrologiciel d'alimentation électrique par port Universal Serial Bus, révision 1.0, 15 septembre 2016. http://www.usb.org/developers/powerdelivery/
[USBBC 1.2] – Spécification de charge de batterie Universal Serial Bus, révision 1.2, plus errata (citée dans le présent document en tant que Spécification de charge de batterie). www.usb.org/developers/devclass_docs#approved.
[USBBridge 1.1] – Spécification de passerelle Universal Serial Bus Type-C, révision 1.1, 10 octobre 2017. www.usb.org/developers/docs.
[USBTypeCBridge 1.0] – Spécification de passerelle Universal Serial Bus Type-C, révision 1.0, 25 mars 2016. www.usb.org/developers/docs.
[USBPD 2.0] – Spécification d'alimentation électrique par port Universal Serial Bus, révision 2, version 1.2, 25 mars 2016. www.usb.org/developers/docs.
[USBPDCompliance] – Plan de conformité de l'alimentation électrique par port USB, révision 1.02, version 2.0, 8 mars 2017 http://www.usb.org/developers/docs/devclass_docs/.
[USB Type-C 2.0] – Spécification de câble et de connecteur Universal Serial Bus Type-C, révision 2.0, à déterminer, août 2019. www.usb.org/developers/docs.
[IEC 60958-1] IEC 60958-1 Interface audionumérique – Partie 1: Généralités, Edition 3.0 2008-09 www.iec.ch
[IEC 60950-1] IEC 60950-1:2005 Matériels de traitement de l'information – Sécurité – Partie 1: Exigences générales: Amendement 1:2009, Amendement 2:2013
[IEC 62368-1] IEC 62368-1 Equipements des technologies de l'audio/vidéo, de l'information et de la communication – Partie 1: Exigences de sécurité
[IEC 63002] Projet de CD pour l'IEC 63002 Méthode d'identification et d'interopérabilité des communications des alimentations externes utilisées avec les dispositifs informatiques portatifs.
[ISO 3166] ISO 3166 Norme internationale des codes des noms de pays et de leurs subdivisions. http://www.iso.org/iso/home/standards/country_codes.htm.
[USB4] – Spécification Universal Serial Bus 4 (USB4™), version 1.0, août 2019. www.usb.org/developers/docs.
[DPTC1.0] – Norme DisplayPort™ Alt Mode sur USB Type-C®, version 1.0b, 3 novembre 2017. www.vesa.org.
[TBT3] voir ***[USB4]***, Chapitre 13, pour le fonctionnement d'un dispositif Thunderbolt™ 3.

1.6 Termes et abréviations

Cette section définit les termes utilisés dans le présent document. Pour les termes supplémentaires concernant le Bus universel en série, voir le Chapitre 2, "Termes et abréviations", dans ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** et ***[USBBC 1.2]***.

Tableau 1-1 Termes et abréviations

Terme	Description
Câble actif	Câble avec une fiche USB à chaque extrémité, au moins une d'entre elles étant une fiche de câble compatible SOP', qui comprennent également des circuits de conditionnement des signaux du bus de données. Le câble prend en charge la commande de VDM structuré ***Discover Identity*** pour déterminer ses caractéristiques, en plus des autres commandes VDM structurées (câble marqué électroniquement, voir ***[USB Type-C 2.0]***).
Mode actif	Mode entré qui n'a pas été quitté.
Mode alternatif	Défini dans ***[USB Type-C 2.0]***. Equivalent au mode de la spécification de l'alimentation électrique par port USB.
Adaptateur de mode alternatif (AMA, *Alternate Mode Adapter*)	Dispositif PDUSB qui prend en charge les modes alternatifs définis dans ***[USB Type-C 2.0]***. Noter que, du fait qu'un AMA est un dispositif PDUSB, il ne dispose que d'un seul port UFP, uniquement adressable par des paquets SOP.
Contrôleur de mode alternatif (AMC, *Alternate Mode Controller*)	Port DFP prenant en charge la connexion vers des AMA définie dans ***[USB Type-C 2.0]***. Un port DFP qui est un AMC peut également être un hôte PDUSB.
Objet de données d'alimentation augmentée (APDO, *Augmented Power Data Object*)	Objet de données utilisé pour exposer les capacités d'alimentation d'un port source ou les exigences d'alimentation d'un destinataire, en tant que partie d'un message ***Source_Capabilities*** ou ***Sink_Capabilities***, respectivement. L'objet de données d'alimentation électrique programmable est défini.
Séquence atomique de messages (AMS, *Atomic Message Sequence*)	Séquence de messages fixe définie en 8.3.2 commençant et finissant généralement dans l'un des états suivants: ***PE_SRC_Ready***, ***PE_SNK_Ready*** ou ***PE_CBL_Ready***. Une AMS peut être interruptible ou non.
Branchement	Union mécanique de la paire de ports par un câble.
Branchés	Ports d'alimentation électrique USB reliés mécaniquement par un câble USB.
Batterie	Dispositif de stockage d'énergie situé à l'arrière d'un port pouvant être une source ou un destinataire d'alimentation électrique.
Emplacement de batterie	Emplacement physique dans lequel une batterie remplaçable à chaud peut être installée. A tout moment, un emplacement de batterie peut ou peut ne pas contenir de batterie remplaçable à chaud.
Alimentation par batterie	Alimentation électrique appliquant directement la sortie d'une batterie sur V_{BUS}. Elle est signalée par le PDO Battery Supply (voir 6.4.1.2.4)
Modulation de fréquence binaire (BFSK)	Schéma de signalisation désormais ***Déconseillé(s/ée/ées)*** par la présente spécification. La modulation BFSK utilisait une paire de fréquences discrètes pour transmettre des informations binaires (des 0 et des 1) sur V_{BUS}. Voir ***[USBPD 2.0]*** pour plus d'informations.
Biphase Mark Coding (BMC)	Modification du codage Manchester où chaque 0 comporte une transition et un 1 comporte deux transitions (voir ***[IEC 60958-1]***).
BIST	Built-In Self-Test (autotest intégré) – Mécanisme d'essai d'alimentation électrique pour la couche PHY.
Objet de données BIST (BDO, *BIST Data Object*)	Objet de données utilisé par les messages ***BIST***.
Mode BIST	Mode d'essai de récepteur ou d'émetteur BIST activé par un message ***BIST***.
Fiche de câble	Terme utilisé pour décrire un élément apte à l'alimentation électrique au sein d'un système multipoint adressé par des paquets SOP'/SOP''. D'un point de vue logique, la fiche de câble est associée à une fiche USB à l'une des extrémités du câble. Dans une mise en œuvre pratique, l'électronique pourra se situer n'importe où dans le câble.
Réinitialisation de câble	Elle est initiée par un signal ***Cable Reset*** venant du port DFP. Elle rétablit l'état d'alimentation par défaut des fiches de câbles et réinitialise le moteur de communications PD à son état par défaut. Elle ne réinitialise pas les ports partenaires mais rétablit V_{CONN} à son état de branchement.
Charge simultanée	Mécanisme permettant à un appareil USB (VPD) alimenté par V_{CONN} de transmettre l'alimentation électrique et la communication CC d'un port à l'autre sans aucune interférence ni aucune reprise de régulation.

Terme	Description
Port en charge simultanée	Embase USB Type-C sur un dispositif USB, conçue pour permettre à une source d'être connectée par l'intermédiaire du dispositif USB de manière à charger un système auquel il est branché. L'utilisation la plus courante consiste à permettre à un hôte à un seul port de prendre en charge un dispositif USB pendant sa charge.
Fragment	Partie de bloc de données de longueur inférieure ou égale à ***MaxExtendedMsgChunkLen*** (26 octets). Les blocs de données peuvent être envoyés sous forme de message unique ou de série de fragments.
Fragmentation	Processus consistant à découper un bloc de données de longueur supérieure à***MaxExtendedMsgLegacyLen*** (26 octets) en deux ou plusieurs fragments.
Prise commutée	Port n'appliquant pas ***vSafe5V*** sur V_{BUS} tant qu'un destinataire n'est pas branché.
Commande	Paire de demande et réponse définie comme partie d'un message structuré défini par le fournisseur (voir 6.4.4.2)
Canal de configuration (CC)	Fil simple utilisé par le schéma de signalisation BMC de la couche PHY (voir ***[USB Type-C 2.0]***).
Connecté	Ports d'alimentation électrique USB ayant échangé un message et un message de réponse ***GoodCRC*** utilisant le protocole d'alimentation électrique par port USB, de manière à ce que les deux ports partenaires sachent que chacun est apte à l'alimentation électrique par port USB.
Consommateur	Capacité d'un port PD (généralement un port UFP d'un dispositif) à recevoir l'alimentation électrique du conducteur d'alimentation (par exemple V_{BUS}). Correspond à un port USB Type-C avec Rd affirmée sur son fil CC.
Consommateur/fournisseur	Consommateur ayant également la capacité de jouer le rôle de fournisseur. Correspond à un port double fonction avec Rd affirmée sur son fil CC.
Mode BIST continu	Mode BIST dans lequel le port ou la fiche de câble à l'essai émet un flux continu de données d'essai.
Tension constante (CV)	Mode dans lequel la tension de sortie de la source reste constante lorsque la charge varie.
Contrat	Accord conclu entre une paire de ports sur le niveau et le sens de l'alimentation. Un contrat peut être négocié de manière explicite entre la paire de ports ou peut être un niveau d'alimentation implicite défini par l'état en cours. Lors d'un fonctionnement en mode alimentation électrique, il y a toujours un contrat explicite ou implicite en place. Le contrat peut uniquement être modifié en cas de (re)négociation, de permutation des rôles d'alimentation, de permutation des rôles de transmission de données, de réinitialisation matérielle ou de défaillance de la source.
Message de contrôle	Un message est défini comme un message de contrôle lorsque le champ ***Number of Data Objects*** de l'en-tête de message est défini sur 0. Le message de contrôle est uniquement composé d'un en-tête de message et d'un CRC.
Limite de courant (CL, *Current Limit*)	Fonction de limitation de courant pour une source. Lorsque le destinataire tente d'appeler plus de courant provenant de la source que la valeur de la limite de courant demandée, la source réduit sa tension de sortie, de sorte que le courant qu'elle fournit reste inférieur ou égal à la valeur demandée.
Bloc de données	Unité de données de charge utile d'un message étendu. La taille de chaque type de bloc de données est spécifiée sous forme d'une série d'octets d'une longueur maximale de ***MaxExtendedMsgLen*** octets. Il s'agit d'une notion distincte de celle d'objet de données utilisé par un message de données et qui est toujours un objet de 32 bits.
Message de données	Un message de données est constitué d'un en-tête de message suivi d'un ou plusieurs objets de données. Les messages de données sont facilement identifiables dans la mesure où le champ ***Number of Data Objects*** est une valeur non nulle.
Objet de données	Unité de données de charge utile d'un message de données. Cet objet de 32 bits contient des informations spécifiques à différents types de messages de données. Les objets de données Power, Request, BIST et Vendor sont définis.
Permutation des rôles de transmission de données	Processus d'échange des rôles de port DFP (hôte) et de port UFP (dispositif) entre ports partenaires, utilisant le connecteur ***[USB Type-C 2.0]***.

Terme	Description
Batterie déchargée	Dispositif avec une batterie déchargée lorsque la batterie d'un dispositif est inapte à alimenter ses fonctions.
Débranchement	Débranchement mécanique de la paire de ports, en enlevant le câble.
Débranché	Ports d'alimentation électrique USB qui ne sont plus reliés mécaniquement par un câble USB.
Dispositif	Lorsqu'il est indiqué en lettres minuscules (dispositif), il se réfère à tout produit USB, qu'il s'agisse d'un dispositif USB ou d'un hôte USB. Lorsqu'il est indiqué en lettres majuscules, il se réfère à un dispositif USB (périphérique ou hub).
Gestionnaire de politique d'utilisation des dispositifs (DPM, *Device Policy Manager*)	Module fonctionnant dans une source ou un destinataire, qui applique une politique locale à chaque port du dispositif, par l'intermédiaire du moteur de politique.
Processus de découverte	Séquence de commande utilisant des messages structurés définis par le fournisseur, ayant pour résultat l'identification du port partenaire, de ses SVID et modes pris en charge.
Port orienté en aval (DFP, *Downstream Facing Port*)	Indique la position du port dans la topologie USB qui correspond généralement à un port racine de l'hôte USB ou à un port en aval du hub, comme défini dans ***[USB Type-C 2.0]***. A la connexion, le port fonctionne par défaut en tant qu'hôte USB (lorsque la communication USB est prise en charge) et en tant que source.
Données double fonction (DRD, *Dual-Role Data*)	Aptitude à fonctionner en tant que port DFP ou UFP.
Port de données double fonction	Port capable de fonctionner en tant que DRD.
Alimentation double fonction (DRP, *Dual-Role Power*)	Aptitude à fonctionner en tant que source ou que destinataire.
Dispositif d'alimentation double fonction	Produit contenant un ou plusieurs ports d'alimentation double fonction, capables de fonctionner en tant que source ou que destinataire.
Port d'alimentation double fonction	Port capable de fonctionner en tant que DRP.
Fin de paquet (EOP, *End of Packet*)	Marqueur de code K servant à identifier la fin d'un paquet.
Processus d'entrée dans un mode	Séquence de commande utilisant des messages structurés définis par le fournisseur, ayant pour résultat l'entrée des ports partenaires dans un mode.
Rétablissement sur erreur	Processus de rétablissement sur erreur défini dans ***[USB Type-C 2.0]***.
Processus de sortie d'un mode	Séquence de commande utilisant des messages structurés définis par le fournisseur, ayant pour résultat la sortie des ports partenaires d'un mode.
Contrat explicite	Accord conclu entre une paire de ports, résultant d'un processus de négociation d'alimentation électrique. Un contrat explicite est établi (ou reconduit) lorsqu'une source envoie un message ***Accept*** en réponse à un message ***Request*** envoyé par un destinataire et suivi par un message ***PS_RDY*** indiquant que l'alimentation électrique est prête. Cela correspond à l'état ***PE_SRC_Ready*** pour un moteur de politique de source et à l'état ***PE_SNK_Ready*** pour un moteur de politique de destinataire. Le contrat explicite peut être modifié par le processus de renégociation. Il est exigé que toutes les paires de ports établissent un contrat explicite.
Message étendu (EM, *Extended Message*)	Message contenant des blocs de données. Le message étendu est défini par le champ ***Extended*** mis à 1 dans l'en-tête de message, et il contient un en-tête de message étendu immédiatement à la suite de l'en-tête de message.
En-tête de message étendu	Chaque message étendu contient un en-tête de message étendu de 16 bits immédiatement à la suite de l'en-tête de message contenant des informations sur le bloc de données et sur toute fragmentation appliquée.
Permutation rapide des rôles	Processus d'échange rapide des rôles de source et de destinataire entre ports partenaires en raison de la déconnexion d'une alimentation externe.

Terme	Description
Demande de permutation rapide des rôles	Indication d'une source initiale au destinataire initial qu'une permutation rapide des rôles est nécessaire. La demande de permutation rapide des rôles est indiquée par le pilotage de la ligne CC à la terre; il ne s'agit pas d'un message ou d'un signal.
Batterie fixe	Batterie qu'il n'est pas facile de déplacer ou de remplacer pour un utilisateur final, nécessitant par exemple un outil spécial pour y accéder ou étant soudée.
Alimentation fixe	Alimentation électrique à tension fixe bien régulée. Elle est signalée par le PDO Fixed Supply (voir 6.4.1.2.2)
Trame	Terme générique faisant référence à une communication atomique transmise par une alimentation USB, comme un paquet, une trame d'essai ou un signal.
Réinitialisation matérielle	Elle est assurée par un signal ***Hard Reset*** venant de l'un ou l'autre des ports partenaires. Elle rétablit V_{BUS} au fonctionnement USB par défaut et réinitialise le moteur de communications PD à son état par défaut pour les deux ports partenaires, ainsi que pour toute fiche de câble branchée. Elle renvoie les ports partenaires à leurs rôles de transmission de données par défaut et retourne la source V_{CONN} vers le port source.
DD	Disque dur.
Batterie remplaçable à chaud	Batterie facilement accessible par un utilisateur pour l'enlever ou la remplacer par une autre batterie.
VDO d'en-tête d'ID	VDO d'une commande ***Discover Identity*** suivant immédiatement l'en-tête de VDM. Le VDO d'en-tête d'identification contient des informations correspondant au produit d'alimentation électrique.
Contrat implicite	Accord sur les niveaux d'alimentation entre une paire de ports qui se conclut non pas à la suite d'un processus de négociation d'alimentation électrique, mais à la suite d'une permutation des rôles d'alimentation ou d'une permutation rapide des rôles. Les contrats implicites sont transitoires, dans la mesure où il est exigé que la paire de ports négocie immédiatement un contrat explicite après la permutation des rôles d'alimentation. Un contrat implicite ***Devoir/Doit/Doivent*** être limité à un courant USB Type-C (voir ***[USB Type-C 2.0]***).
Initiateur	Expéditeur initial d'une demande de commande sous la forme d'une requête.
Interruptible	Une AMS qui, à réception d'une erreur de protocole, retourne à l'état disponible approprié afin de traiter le message entrant, est dite interruptible. Toute AMS est interruptible jusqu'à l'envoi du premier message de l'AMS (un message ***GoodCRC*** a été reçu). Une AMS de messages de fournisseur est interruptible durant la totalité de la séquence.
IoC	Valeur de l'intensité de courant négociée définie dans l'***[IEC 63002]***.
Chute de tension ohmique	Chute de tension dans le câble et les connecteurs entre la source et le destinataire. Il s'agit d'une fonction de la résistance de la masse et du conducteur d'alimentation dans le câble, plus la résistance de contact dans les connecteurs, multipliées par l'intensité du courant qui s'écoule dans le circuit.
Code K	Symboles spéciaux fournis par le schéma de codage 4b5b. Les codes K sont utilisés pour signaler une réinitialisation matérielle et une réinitialisation de câble, et pour identifier les limites d'un paquet.
Politique locale	Tout appareil apte à l'alimentation électrique par port USB a sa propre politique, appelée politique locale, exécutée par son moteur de politique afin de contrôler le comportement de son alimentation électrique. A tout moment, la politique locale peut être la politique par défaut, codée en dur ou modifiée par des changements de paramètres opérationnels, ou encore une politique fournie par le système hôte, ou une combinaison de ces politiques. ***En option***, la politique locale peut être modifiée par un gestionnaire de politique système.
LPS	Alimentation électrique limitée (Limited Power Supply) définie dans l'***[IEC 62368-1]***.
Message	Charge utile d'un paquet constituée d'un en-tête de message pour les messages de contrôle, et d'un en-tête de message et de données pour les messages de données et les messages étendus, comme défini en 6.
En-tête de message	Chaque message commence par un en-tête de message de 16 bits, contenant des informations de base sur le message et sur les capacités du port PD.

Terme	Description
Messagerie	Communication sous forme de messages, comme défini au Chapitre 6.
Fonctionnement modal	Etat comportant un ou plusieurs modes actifs. Un fonctionnement modal prend fin lorsqu'il n'y a plus aucun mode actif.
Mode	Fonctionnement défini par un fournisseur ou un organisme de normalisation, associé à un SVID dont la définition ne relève pas du domaine d'application des spécifications USB-IF. L'entrée dans le mode et la sortie de celui-ci font appel aux processus d'entrée dans un mode et de sortie d'un mode. Les modes sont équivalents aux "modes alternatifs" décrits dans ***[USB Type-C 2.0]***.
Multipoint	Se réfère à un système d'alimentation électrique avec une ou plusieurs fiches de câbles, dans lequel la communication se fait avec les fiches de câbles plutôt qu'avec le port partenaire. Les systèmes multipoints partagent le canal de communication d'alimentation électrique avec les ports partenaires.
Négociation	Il s'agit du processus PD dans lequel: 1. la source annonce ses capacités; 2. le destinataire demande l'une des capacités annoncées; 3. la source accuse réception de la demande et modifie sa sortie pour satisfaire à cette demande. Le résultat de la négociation est un contrat d'alimentation électrique/de consommation entre la paire de ports.
Non interruptible	AMS qui, à réception d'une erreur de protocole, génère une réinitialisation logicielle ou une réinitialisation matérielle. Toute AMS liée à l'alimentation est non interruptible dès que le premier message de l'AMS a été envoyé (un message ***GoodCRC*** a été reçu).
OCP	Protection contre les surintensités (Over-Current Protection)
OTP	Protection contre la surchauffe (Over-Temperature Protection)
OVP	Protection contre les surtensions (Over-Voltage Protection)
Paquet	Une unité entière de communication PD comprenant un préambule, un ***SOP****, une charge utile, un CRC et un ***EOP***, comme défini en 5.6.
Câble passif	Câble avec une fiche USB à chaque extrémité, l'une d'entre elles au moins étant une fiche de câble compatible SOP', qui ne comprend aucun circuit de conditionnement des signaux du bus de données. Prend en charge le VDM structuré ***Discover Identity*** pour déterminer ses caractéristiques (câble marqué électroniquement, voir ***[USB Type-C 2.0]***). Noter que cette spécification ne traite pas des câbles passifs qui ne sont pas des câbles marqués électroniquement.
PD	Alimentation électrique par port USB
Apte à l'alimentation électrique par port USB	Port prenant en charge l'alimentation électrique par port USB.
Connexion PD	Voir "Connecté".
Alimentation PD (PDP, *PD Power*)	Sortie alimentation d'une source, spécifiée par le fabricant et exprimée dans des PDO d'alimentation fixe, comme défini en 10.
PDP assignée	PDP déclarée par le fabricant pour une source.
PDUSB	Port de dispositif USB ou port d'hôte USB apte à la fois à l'alimentation électrique par port USB et à la communication USB. Voir également "hôte PDUSB", "dispositif PDUSB" et "hub PDUSB".
Dispositif PDUSB	Dispositif USB avec un port UFP apte à l'alimentation électrique par port USB. Un dispositif PDUSB n'est adressé que par des paquets SOP.
Hôte PDUSB	Hôte USB apte à l'alimentation électrique par port USB sur au moins l'un de ses ports DFP. Un hôte PDUSB n'est adressé que par des paquets SOP.
Hub PDUSB	Dispositif USB d'extension de port avec un port UFP et un ou plusieurs ports DFP, apte à l'alimentation électrique par port USB sur au moins l'un de ses ports. Un hub PDUSB n'est adressé que par des paquets SOP.
Périphérique PDUSB	Dispositif USB avec un port UFP apte à l'alimentation électrique par port USB qui n'est pas un hub PDUSB. Un périphérique PDUSB n'est adressé que par des paquets SOP.
Couche PHY	Couche physique responsable de l'envoi et de la réception de messages sur le fil CC USB Type-C entre une paire de ports.

Terme	Description
Politique	La politique définit le comportement des parties du système aptes à l'alimentation électrique par port USB et définit les capacités qu'elle annonce, les demandes émises pour (re)négocier la puissance et les réponses données aux demandes reçues.
Moteur de politique (PE, *Policy Engine*)	Le moteur de politique interprète l'entrée du gestionnaire de politique d'utilisation des dispositifs afin de mettre en œuvre la politique pour un port donné. Il ordonne à la couche protocole d'envoyer les messages appropriés.
Port	Interface généralement accessible au moyen d'une embase, ou par l'intermédiaire d'une fiche à l'extrémité d'un câble captif raccordé sur un circuit. L'alimentation électrique par port USB définit l'interaction entre une paire de ports.
Paire de ports	Deux ports aptes à l'alimentation électrique par port USB branchés.
Port partenaire	Un contrat est négocié entre une paire de ports connectée par un câble USB. Ces ports sont connus en tant que ports partenaires.
Conducteur d'alimentation	Fil fournissant l'alimentation électrique de la source au destinataire. Par exemple, V_{BUS}de l'interface USB.
Consommateur de puissance	Voir "Consommateur".
Objet de données d'alimentation (PDO, *Power Data Object*)	Objet de données utilisé pour exposer les capacités d'alimentation d'un port source ou les exigences d'alimentation d'un destinataire, en tant que partie d'un message ***Source_Capabilities*** ou ***Sink_Capabilities***, respectivement. Des objets de données d'alimentation fixe, variable et par batterie sont définis.
Mode alimentation électrique	Fonctionnement après établissement initial d'un contrat entre une paire de ports. Ce mode persiste pendant le fonctionnement normal de l'alimentation électrique, y compris après une permutation des rôles d'alimentation. Le mode alimentation électrique peut être quitté uniquement en débranchant les ports, en procédant à une réinitialisation matérielle ou lorsque la source coupe l'alimentation (sauf si l'alimentation est coupée lors de la procédure d'échange des rôles d'alimentation).
Fournisseur d'alimentation électrique	Voir "Fournisseur".
Réserve de puissance	Puissance conservée par une source afin de s'assurer qu'elle peut satisfaire à toutes les exigences en matière de puissance des destinataires branchés sur au moins un port.
Permutation des rôles d'alimentation	Processus d'échange des rôles de source et de destinataire entre ports partenaires.
Préambule	Début d'une transmission permettant au récepteur de se verrouiller sur la porteuse. Le préambule est constitué d'une séquence de 64 bits d'alternance de 0 et de 1 commençant par un "0" et finissant par un "1", qui n'est pas encodée en 4b5b.
Type de produit	Catégorisation de produit renvoyée comme partie de la commande ***Discover Identity***.
VDO de type de produit	VDO identifiant un certain type de produit dans le VDO d'en-tête d'identification d'une commande ***Discover Identity***.
Alimentation électrique programmable (PPS, *Programmable Power Supply*)	Alimentation électrique dont la tension de sortie peut être réglée par programme par faibles incréments sur sa plage annoncée. La PPS a également un repli de courant de sortie programmable. Les capacités de la PPS sont signalées par l'APDO Programmable Power Supply (voir 6.4.1.2.5).
Erreur de protocole	Message inattendu lors d'une séquence atomique de messages Une erreur de protocole pendant une AMS non interruptible a pour résultat une réinitialisation logicielle ou une réinitialisation matérielle. Une erreur de protocole pendant une AMS interruptible a pour résultat un retour à l'état disponible approprié lorsque le message est traité.
Couche protocole	Entité qui forme les messages servant à la communication d'informations entre ports partenaires.
Fournisseur	Capacité d'un port PD (généralement un hôte, un hub ou un port DFP de bloc d'alimentation) à alimenter le conducteur d'alimentation (par exemple V_{BUS}). Correspond à un port USB Type-C avec Rp affirmée sur son fil CC.

Terme	Description
Fournisseur/Consommateur	Fournisseur ayant également la capacité d'agir en tant que consommateur. Correspond à un port d'alimentation double fonction avec Rp affirmée sur son fil CC.
PS1, PS2	Classification d'alimentation électrique définie dans l'***[IEC 62368-1]***.
PSD	Destinataire qui appelle du courant mais n'a pas d'autre fonction de communication USB ou de mode alternatif, par exemple une réserve de puissance.
Rd	Résistance de dépolarisation sur le fil CC du port USB Type-C servant à indiquer que le port est un destinataire (voir ***[USB Type-C 2.0]***).
Rebranchement	Branchement de la paire de ports par un câble après un précédent débranchement.
Renégociation	Processus par lequel un des ports partenaires souhaite modifier le contrat négocié.
Objet de données de demande (RDO, *Request Data Object*)	Objet de données utilisé par un port destinataire pour négocier un contrat, au sein d'un message ***Request***.
Relance	Redémarrage d'une AMS interruptible à partir du début après une erreur de protocole.
Répondeur	Récepteur d'une demande de commande envoyée par un initiateur qui répond par une réponse de commande.
Rp	Résistance de polarisation sur le fil CC du port USB type-C servant à indiquer que le port est une source (voir ***[USB Type-C 2.0]***).
Fonctionnement de sécurité	Les sources doivent avoir la capacité à tolérer ***vSafe5V*** appliquée par les deux ports partenaires.
Signalisation	Préambule suivi par un ensemble ordonné de quatre codes K servant à indiquer un symbole de ligne particulier, par exemple ***Hard Reset*** comme défini en 5.4.
Schéma de signalisation	Mécanisme physique servant à la transmission de bits. Seul le schéma de signalisation BMC est défini dans la présente spécification. Note: Le schéma de signalisation BFSK pris en charge dans les précédentes révisions de la présente spécification est désormais ***Déconseillé(s/ée/ées)***.
Port à fonction unique	Port qui est seulement capable de fonctionner en tant que source ou en tant que destinataire, mais pas les deux.
Destinataire	Port qui consomme la puissance provenant de V_{BUS}, appelé plus communément dispositif.
Charge commandée par le destinataire	Principe de charge dans lequel le destinataire connecte la source à sa batterie par l'intermédiaire du circuit de sécurité et d'autres circuits. Lorsque la fonction de limite de courant est activée, la source contrôle automatiquement son courant de sortie en réglant sa tension de sortie.
Réinitialisation logicielle	Processus qui réinitialise le moteur de communications PD à son état par défaut.
Communication SOP	Communication faisant appel à des paquets SOP, ce qui implique aussi qu'une séquence de messages soit suivie.
Paquet SOP	Tout paquet d'alimentation électrique commençant par un ***SOP***.
Communication SOP*	Communication avec une fiche de câble faisant appel à des paquets SOP*, ce qui implique aussi qu'une séquence de messages soit suivie.
Paquet SOP*	Terme qui fait référence à tout paquet d'alimentation électrique qui commence par ***SOP***, ***SOP'*** ou ***SOP''***.
Communication SOP'	Communication avec une fiche de câble faisant appel à des paquets SOP', ce qui implique aussi qu'une séquence de messages soit suivie.
Paquet SOP'	Tout paquet d'alimentation électrique commençant par un ***SOP'*** et servant à communiquer avec une fiche de câble.
Communication SOP''	Communication avec une fiche de câble faisant appel à des paquets SOP'', ce qui implique aussi qu'une séquence de messages soit suivie.

Terme	Description
Paquet SOP''	Tout paquet d'alimentation électrique commençant par un ***SOP''*** et servant à communiquer avec une fiche de câble lorsque des paquets SOP' sont utilisés pour communiquer avec l'autre fiche de câble.
Source	Rôle joué par un port fournissant actuellement l'alimentation électrique sur V_{BUS}, le plus fréquemment un port en aval de l'hôte ou du hub.
ID normalisé (SID, *Standard ID*)	Valeur non signée de 16 bits, attribuée par l'USB-IF à une norme industrielle donnée.
ID normalisé ou ID de fournisseur (SVID, *Standard or Vendor ID*)	Terme générique faisant référence à un VID ou à un SID. SVID est utilisé à la place de l'expression "ID normalisé ou ID de fournisseur".
Début de paquet (SOP, *Start of Packet*)	Marqueur de code K servant à identifier le début d'un paquet. Trois séquences de début de paquet sont définies: ***SOP, SOP'*** et ***SOP''***, ***SOP**** étant utilisé pour les désigner tous les trois, au lieu de ***SOP***/***SOP'***/***SOP''.***
Politique système	Politique de l'ensemble du système, générée par le système, décomposée en politiques exigées par chaque paire de ports pour affecter la politique système. Elle est transmise par programme à chacun des dispositifs pour être traitée par leurs moteurs de politique.
Gestionnaire de politique système (SPM, *System Policy Manager*)	Module fonctionnant sur l'hôte USB. Il applique la politique système en communiquant avec des consommateurs et des fournisseurs aptes à l'alimentation électrique par port USB qui sont également connectés à l'hôte par USB.
Trame d'essai	Trame constituée d'un préambule, d'un ***SOP****, suivis de données d'essai (voir 5.9).
Modèle d'essai	Flux continu de données d'essai dans une séquence donnée (voir 5.9)
Dispositif d'essai	Le dispositif d'essai est admis comme étant un élément d'équipement d'essai gérant le processus d'essai BIST d'une UUT PD.
Message inopiné	Message pris en charge par un port mais qui a été reçu dans un état incorrect.
Intervalle unitaire (UI, *Unit Interval*)	Temps mis pour transmettre un bit de données unique sur le fil.
Unité à l'essai (UUT, *Unit Under Test*)	Dispositif PD soumis à l'essai par le dispositif d'essai et répondant à la mise en place d'une séquence d'essai BIST particulière.
Message non reconnu	Message qu'un port ne comprend pas, par exemple un message utilisant un type ReservedMessage, un message défini par une révision de la spécification postérieure à la révision prise en charge par ce port, ou encore un message non structuré pour lequel le VID n'est pas reconnu.
Message non pris en charge	Message qu'un port reconnaît mais qu'il ne prend pas en charge. Il s'agit d'un message défini par la spécification mais qui n'est pas pris en charge par ce port.
Port orienté en amont (UFP, *Upstream Facing Port*)	Indique la position du port dans la topologie USB, correspondant généralement à un port sur un dispositif, comme défini dans ***[USB Type-C 2.0]***. A la connexion, le port fonctionne par défaut en tant que dispositif USB (lorsque la communication USB est prise en charge) et en tant que destinataire.
Etat USB branché	Synonyme de la définition ***[USB 2.0]*** et ***[USB 3.2]*** de l'état branché
Fonctionnement USB par défaut	Fonctionnement d'un port au branchement ou après une réinitialisation matérielle, lorsque la source DFP applique ***vSafe0V*** ou ***vSafe5V*** sur V_{BUS}et que le destinataire UFP fonctionne à ***vSafe5V***, comme défini dans ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** ou ***[USBBC 1.2]***.
Dispositif USB	Hub ou dispositif périphérique défini dans ***[USB 2.0]*** et ***[USB 3.2]***.
Hôte USB	Système informatique hôte sur lequel le contrôleur hôte USB est installé, comme défini dans ***[USB 2.0]*** et ***[USB 3.2]***.
Etat USB alimenté	Synonyme de la définition ***[USB 2.0]*** et ***[USB 3.2]*** de l'état alimenté.
Etat de sécurité USB	Etat du connecteur USB Type-C lorsque des broches doivent être réaffectées à d'autres usages (voir ***[USB Type-C 2.0]***) pour qu'elles ne soient pas endommagées par leur port partenaire et qu'elles n'endommagent pas ce dernier.

Terme	Description
USB Type-A	Terme utilisé pour faire référence à toute fiche ou embase A, y compris les fiches micro A et les fiches et embases normalisées A. Par hypothèse, les embases micro AB sont vues comme une combinaison d'USB Type-A et d'USB Type-B.
USB Type-B	Terme utilisé pour faire référence à toute fiche ou embase B, y compris les fiches micro B et les fiches et embases normalisées B, y compris les versions PD et non PD. Par hypothèse, les embases micro AB sont vues comme une combinaison d'USB Type-A et d'USB Type-B.
USB Type-C	Terme utilisé pour faire référence à la fiche ou l'embase d'un connecteur USB Type-C, comme défini dans ***[USB Type-C 2.0]***.
USB-IF PD SID (PD SID)	ID normalisé attribué à la présente spécification par l'USB Implementer's Forum.
Alimentation variable	Alimentation électrique très mal régulée qui n'est pas une batterie. Elle est signalée par le PDO Variable Supply (voir 6.4.1.2.3)
Accessoire alimenté par VCONN	Accessoire alimenté par VCONN pour fonctionner dans un mode (voir ***[USB Type-C 2.0]***).
Dispositif USB alimenté par VCONN (VPD, *Vconn Powered USB Device*)	Dispositif USB à câble captif pouvant être alimenté par VCONN ou par VBUS, comme défini dans ***[USB Type-C 2.0]***. Un VPD est un dispositif USB à câble captif qui peut être alimenté par VCONN ou VBUS et qui ne répond qu'aux messages SOP' définis dans les tableaux présentés en 6.12 (Applicabilité des messages). Il ne répond qu'aux messages envoyés avec une spécification de révision 3.0 ou ultérieure. Avec un VPD, la prise en charge des modes alternatifs n'est pas admise. Le terme VPD désigne soit un VPD, soit un CT-VPD sans chargeur connecté.
Dispositif de charge USB alimenté par VCONN (CT-VPD, *Charge Through Vconn Powered USB Device*)	Un CT-VPD est un VPD équipé d'un port supplémentaire pour le raccordement d'une source (par exemple, un chargeur), comme défini dans ***[USB Type-C 2.0]***. Lorsqu'aucun chargeur n'est connecté, un CT-VPD se comporte de la même manière qu'un VPD. Lorsqu'un chargeur est connecté, aucune communication PD avec le CT-VPD proprement dit n'est possible, car le fil CC est connecté au port du chargeur. Par conséquent, toute la communication PD se situe au niveau du chargeur et du câble à l'aide duquel il est connecté.
Source VCONN	Port USB Type-C responsable de l'alimentation VCONN.
Permutation de VCONN	Processus d'échange de la source VCONN entre ports partenaires.
En-tête de VDM	Premier objet de données suivant l'en-tête du message dans un message défini par le fournisseur. L'en-tête de VDM contient le SVID relatif au VDM envoyé et fournit des informations relatives à la commande dans le cas d'un VDM structuré (voir 6.4.4).
Objet de données du fournisseur (VDO, *Vendor Data Object*)	Objet de données utilisé pour envoyer des informations spécifiques au fournisseur au sein d'un message ***Vendor_Defined***.
Message défini par le fournisseur (VDM, *Vendor Defined Message*)	Message de données PD défini pour une utilisation par le fournisseur/normalisée. Ces messages sont en outre composés de messages VDM structurés, dont les commandes sont définies dans la présente spécification, et de messages VDM non structurés, qui sont entièrement définis par le fournisseur (voir 6.4.4).
Vendor ID (VID)	Valeur non signée de 16 bits, attribuée par l'USB-IF à un fournisseur donné.
VI	Désigne la puissance (tension * intensité de courant = puissance)
Bloc d'alimentation	Alimentation électrique ou "transformateur" branché(e) sur une prise de courant alternatif. Elle fournit une alimentation en courant continu pour alimenter un dispositif ou pour charger une batterie.

1.7 Valeurs de paramètres

Dans la présente spécification, les paramètres sont exprimés en valeurs absolues. Pour plus d'informations sur la façon dont chacun des paramètres est mesuré, voir ***[USBPDCompliance]***.

1.8 Modifications par rapport à la révision 2.0

La présente spécification comprend les mises à jour suivantes:

règles d'alimentation PD (également appliquées à ***[USBPD 2.0]***);
mécanismes anticollisions et de simplification de la communication;
prise en charge de l'***[IEC 63002]***, y compris des capacités d'alimentation électrique étendues et du statut;
capacités et statut de batterie;
aptitude à effectuer une permutation rapide des rôles d'alimentation;
prise en charge de ***[USBTypeCAuthentication 1.0]*** et de ***[USBPDFirmwareUpdate 1.0]***;
alimentation électrique programmable (PPS, *Programmable Power Supply*) pour la prise en charge de la charge commandée par le destinataire.

Ce qui suit est désormais ***Déconseillé(s/ée/ées)*** par la présente spécification:

le schéma de signalisation BFSK;
les définitions des câbles et connecteurs normalisés/micro A/B;
le fonctionnement sur batterie déchargée pour les ports A/B;
les profils (remplacés par les règles d'alimentation PD).

Ce qui suit a été déplacé vers d'autres spécifications:

la politique système, désormais définie dans ***[USBTypeCBridge 1.0]***.

Pour plus d'informations, voir Section 9.

1.9 Compatibilité avec la révision 2.0

La révision 3.0 de la spécification de l'alimentation électrique par port USB est conçue pour être entièrement interopérable avec les systèmes ***[USBPD 2.0]*** utilisant une signalisation BMC sur le connecteur ***[USB Type-C 2.0]*** et pour être compatible avec le matériel de la révision 2.0.

Voir 2.3 pour plus d'informations sur les mécanismes définis pour permettre la compatibilité.

2. Vue d'ensemble

Cette section ne contient aucune exigence ***Normatif(s/ve/ves)***.

2.1 Introduction

Dans une alimentation électrique par port USB, des paires de ports branchés directement négocient la tension, le courant et/ou le sens du débit de puissance sur le câble USB, en utilisant le fil CC du connecteur USB Type-C® comme canal de communication. Les mécanismes utilisés fonctionnent indépendamment d'autres méthodes USB utilisées pour négocier la puissance.

L'alimentation électrique par port USB se comporte également comme un canal à bande latérale permettant les communications avec l'ensemble câble-connecteur reliant les ports. Les modes sont associés à un ID normalisé ou de fournisseur (SVID). Des messages d'alimentation électrique VDM structurés peuvent être utilisés pour découvrir des SVID et des modes, puis pour accéder aux modes et les quitter en fonction des besoins. Plusieurs modes actifs peuvent également fonctionner simultanément.

Tout contrat négocié sur la base de la présente spécification remplace tous les contrats d'alimentation précédents établis à partir des mécanismes normalisés ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** ou ***[USBBC 1.2]***. Pendant la durée du mode alimentation électrique, il y a un contrat (explicite ou implicite) en place qui détermine le niveau de puissance disponible et le sens de l'alimentation. La paire de ports reste en mode alimentation électrique jusqu'à son débranchement, jusqu'à une réinitialisation matérielle ou jusqu'à ce que la source coupe l'alimentation (sauf pendant une permutation des rôles d'alimentation ou une permutation rapide des rôles lorsque la source initiale coupe l'alimentation afin que la nouvelle source établisse l'alimentation).

Un contrat explicite est négocié par le processus par lequel la source envoie un ensemble de capacités, à partir duquel il est exigé que le destinataire demande une capacité particulière, à la suite de quoi la source accepte cette demande.

Un contrat implicite est le niveau de puissance spécifié admis dans des états particuliers (pendant et après une permutation des rôles d'alimentation ou une permutation rapide des rôles). Les contrats implicites sont temporaires. Il est demandé aux paires de ports de négocier immédiatement un contrat explicite.

Chaque fournisseur dispose d'une politique locale régissant l'affectation de la puissance à ses ports. Les destinataires ont également leur propre politique locale régissant leur façon d'appeler du courant. Une politique système peut être mise en œuvre sur USB pour permettre de modifier ces politiques locales et ainsi de gérer l'affectation globale de la puissance dans le système.

Lorsque les dispositifs aptes à l'alimentation électrique par port USB sont branchés les uns aux autres, les ports DFP et UFP sont initialement, par défaut, en fonctionnement normal USB par défaut. Le port DFP fournit ***vSafe5V*** et le port UFP appelle du courant conformément aux règles définies par les spécifications ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** ou ***[USBBC 1.2]***. Après la négociation d'alimentation électrique, l'alimentation peut être fournie à des tensions plus élevées ou plus faibles, et avec des intensités de courant supérieures à celles définies dans ces spécifications. Il est également possible d'effectuer une permutation des rôles d'alimentation ou une permutation rapide des rôles pour échanger les rôles d'alimentation de sorte que le port DFP reçoive l'alimentation et que le port UFP la fournisse, d'effectuer une permutation des rôles de transmission de données de sorte que le port DFP devienne le port UFP, et inversement, et d'effectuer une permutation de Vconn pour modifier l'extrémité fournissant Vconn au câble.

Avant un contrat explicite, seul le port source, qui est également la source VCONN, peut communiquer avec l'ensemble câble-connecteur branché. Cela est important pour ***[USB Type-C 2.0]***, où le câblage 5 A est marqué, ainsi que d'autres détails de l'ensemble câble-connecteur, tels que la vitesse prise en charge.

La découverte de câble, qui détermine si le câble peut communiquer, peut se produire au branchement initial d'une paire de ports, avant l'établissement d'un contrat explicite. Il est également possible d'effectuer une découverte du câble après une permutation des rôles d'alimentation ou une permutation rapide des rôles, avant de rétablir un contrat explicite, lorsque le port UFP est la source et qu'un contrat implicite est en place. La découverte de câble peut être effectuée après l'établissement d'un contrat explicite, si le câble n'a pas encore été découvert.

Lorsqu'un contrat explicite est en place, le port source ou le port destinataire, sous réserve qu'il soit également la source VCONN, est autorisé à communiquer avec l'ensemble câble-connecteur branché. Cette communication peut comprendre:

la découverte de câble (lorsque le câble n'a pas déjà été découvert);
la découverte de capacités;
la découverte de SVID;
la découverte de modes;
l'entrée de modes pris en charge par l'ensemble câble-connecteur;
la sortie de modes pris en charge par l'ensemble câble-connecteur.

2.2 Vue d'ensemble des sections

La présente spécification comprend les sections suivantes:

Section 1	Introduction, conventions utilisées dans le document, liste des termes et abréviations, références et détails concernant l'utilisation des paramètres.
Section 2	Vue d'ensemble du document, y compris une description du fonctionnement de l'alimentation électrique par port USB et de l'architecture.
Section 3	Caractéristiques mécaniques et électriques des câbles et connecteurs utilisés par l'alimentation électrique par port USB. Section ***Déconseillé(s/ée/ées)*** Voir ***[USBPD 2.0]*** pour l'ancienne spécification des connecteurs PD.
Section 4	Exigences électriques pour le fonctionnement sur batterie déchargée et la détection des câbles.
Section 5	Détails des exigences applicables à la couche PHY de l'alimentation électrique par port USB
Section 6	Exigences applicables à la couche protocole, y compris les messages, temporisateurs, compteurs et le fonctionnement d'état.
Section 7	Exigences applicables à l'alimentation électrique pour les fournisseurs et les consommateurs.
Section 8	Exigences applicables au gestionnaire de politique d'utilisation des dispositifs. Diagrammes de séquence et diagrammes d'état des messages de moteur de politique.
Section 9	Exigences applicables aux dispositifs USBPD, y compris la mise en correspondance de VBUS vers les états USB. Exigences applicables au gestionnaire de politique système, y compris les descripteurs, les événements et les demandes.
Section 10	Définitions de la puissance de sortie assignée pour l'alimentation électrique par port USB.

Annexe A	Exemple de calculs de CRC.
Annexe B	Scénarios présentant le fonctionnement du gestionnaire de politique d'utilisation des dispositifs.
Annexe C	Exemples d'utilisation de VDM structuré.

2.3 Compatibilité avec la révision 2.0

La révision 3.0 de la spécification de l'alimentation électrique par port USB est conçue pour être entièrement interopérable avec les systèmes ***[USBPD 2.0]*** utilisant une signalisation BMC sur le connecteur ***[USB Type-C 2.0]*** et pour être compatible avec le matériel de la révision 2.0.

La présente spécification impose à tous les systèmes de la révision 3.0 de prendre entièrement en charge le fonctionnement de la révision 2.0. Ils doivent découvrir la révision prise en charge utilisée par leur port partenaire et par toutes les fiches de câbles connectées et revenir à un fonctionnement utilisant le plus faible indice de révision commun (voir 6.2.1.1.5).

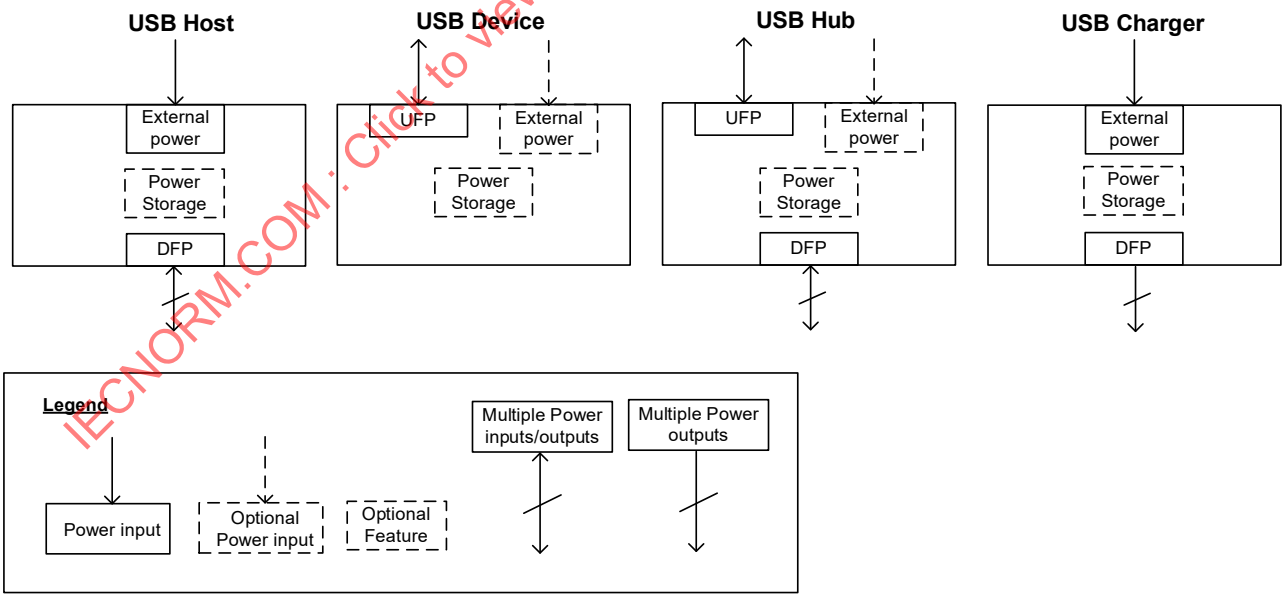
La présente spécification définit des messages étendus contenant jusqu'à 260 octets de données (voir 6.2.1.2). Ces messages sont plus longs que prévu par le matériel PHY existant. Pour prendre en compte les systèmes fondés sur la révision 2.0, un mécanisme de fragmentation est rendu obligatoire, de sorte que les messages sont limités aux dimensions de la révision 2.0 jusqu'à ce qu'il ait été découvert que les deux systèmes prennent en charge les longueurs de message les plus importantes.

La présente spécification comprend les modifications apportées aux objets définis par le fournisseur (VDO) utilisés lors de la découverte des câbles marqués passifs/actifs et des adaptateurs de mode alternatif (voir 6.4.4.2). Pour permettre aux systèmes de déterminer le format VDO utilisé, le numéro de version du message structuré défini par le fournisseur (SVDM) a été incrémenté à 2.0. Les numéros de version ont également été incorporés dans les VDO eux-mêmes pour faciliter les modifications futures, si cela s'avère nécessaire.

2.4 Dispositifs aptes à l'alimentation électrique par port USB

Quelques exemples de dispositifs aptes à l'alimentation électrique par port USB peuvent être observés à la Figure 2-1 (un hôte, un dispositif, un hub et un chargeur). Ils ne sont donnés qu'à titre de référence et ne limitent pas les possibilités de configuration des produits qui peuvent être conçus à partir de la présente spécification.

Figure 2-1 Structure logique des dispositifs aptes à l'alimentation électrique par port USB



Anglais	Français
USB Host	Hôte USB
USB Device	Dispositif USB
USB Hub	Hub USB
USB Charger	Chargeur USB
Legend	Légende

External Power	Alimentation externe
Power Storage	Stock de puissance
Power Input	Entrée d'alimentation
Optional Power Input	Entrée d'alimentation facultative
Optional Feature	Fonctionnalité facultative
Multiple Power inputs/outputs	Entrées/sorties d'alimentation multiples
Multiple Power outputs	Sorties d'alimentation multiples

Par hypothèse, chaque dispositif apte à l'alimentation électrique par port USB est constitué d'au moins un port. Par hypothèse, les fournisseurs ont une source et les consommateurs ont un destinataire. Chaque dispositif contient un ou plusieurs des composants suivants:

ports UFP qui:
- reçoivent l'alimentation électrique;
- ***En option***, fournissent l'alimentation électrique (dispositif d'alimentation double fonction);
- ***En option***, communiquent par USB;
- communiquent en utilisant des paquets SOP;
- ***En option***, communiquent en utilisant des paquets SOP*.

ports DFP qui:
- fournissent l'alimentation électrique;
- ***En option***, reçoivent l'alimentation électrique (dispositif d'alimentation double fonction);
- ***En option***, communiquent par USB;
- communiquent en utilisant des paquets SOP;
- ***En option***, communiquent en utilisant des paquets SOP*.

une source pouvant être:
- une source d'alimentation externe, par exemple en courant alternatif;
- un dispositif de stockage d'énergie (par exemple une batterie);
- dérivée d'un autre port (par exemple un hub alimenté par le bus).

une destinataire pouvant être:
- un dispositif de stockage d'énergie (par exemple une batterie);
- utilisé pour alimenter des fonctions internes;
- utilisé pour alimenter des dispositifs branchés sur d'autres dispositifs (par exemple un hub alimenté par bus).

une source Vconn qui:
- peut être un port partenaire, le DFP/UFP ou une source/un destinataire;
- alimente la ou les fiches de câbles;
- est le seul port autorisé à dialoguer avec la ou les fiches de câbles à tout moment donné.

2.5 Communication SOP*

2.5.1 Introduction

Le début de paquet (ou SOP) sert de schéma d'adressage pour identifier si la communication est destinée à l'un des ports partenaires (communication SOP) ou à l'une des fiches de câbles (communication SOP'/SOP''). SOP/SOP' et SOP'' sont désignés collectivement par SOP*. Le terme de fiche de câble, dans le cas des communications SOP'/SOP'', sert à représenter une entité logique dans le câble qui est apte à la communication PD et qui est susceptible ou non d'être physiquement logée dans la fiche.

Les sections qui suivent décrivent comment ce schéma d'adressage fonctionne pour des communications de type port à port et port à fiche de câble.

2.5.2 SOP* anticollision

Pour tous les SOP*, la source coordonne la communication afin d'éviter les collisions sur le bus en permettant au destinataire d'initier un échange de messages lorsqu'elle n'a pas elle-même besoin de

communiquer. Dès qu'un contrat explicite est en place, la source indique au destinataire qu'il peut initier une séquence de messages. Cette séquence peut être une communication avec la source ou avec l'une des fiches de câbles. Dès qu'il est nécessaire que la source elle-même initie une séquence de message, cela est indiqué au destinataire. La source attend alors la fin de toute communication SOP* du destinataire en cours avant d'initier elle-même une séquence de messages.

2.5.3 Communication SOP

La communication SOP est utilisée pour les communications port à port entre la source et le destinataire. La communication SOP est reconnue par les deux ports partenaires, mais pas par les fiches de câbles intervenant. La communication SOP a la priorité sur les autres communications SOP* car il est primordial d'achever les opérations relatives à l'alimentation le plus vite possible. Les séquences de messages relatives à l'alimentation sont également autorisées à interrompre d'autres séquences pour s'assurer que la négociation et le contrôle de la puissance ont la priorité sur le bus.

2.5.4 Communication SOP'/SOP'' avec des fiches de câbles

La communication SOP' est reconnue par l'électronique d'une fiche de câble (voir ***[USB Type-C 2.0]***). La communication SOP'' peut également être prise en charge lorsque la communication SOP' l'est aussi. L'affectation SOP' et SOP'' est fixe et n'est pas modifiée de façon dynamique.

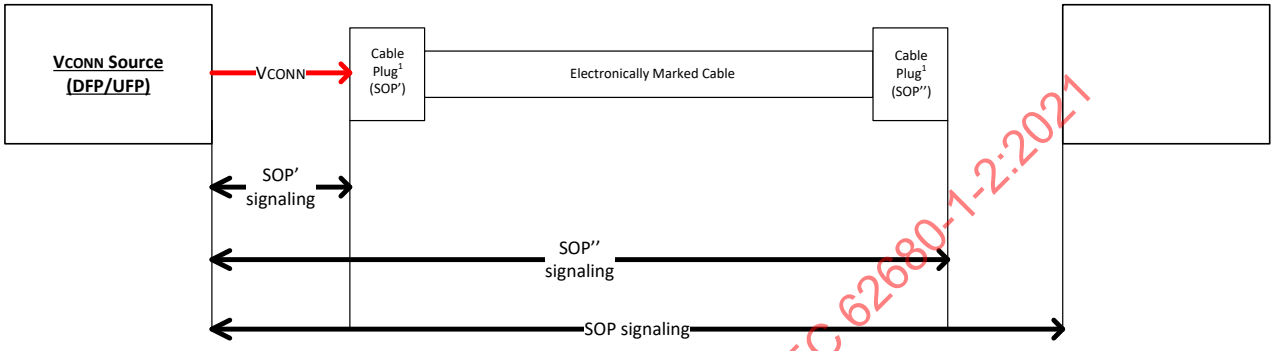
La communication SOP entre les ports partenaires n'est pas reconnue par la fiche de câble. La Figure 2-2 représente l'utilisation de communications SOP* entre une source VCONN (DFP/UFP) et les fiches de câbles.

Toutes les communications SOP* ont lieu sur un fil unique (CC). Cela signifie que les périodes de communication SOP* doivent être coordonnées afin d'empêcher qu'une communication importante ne soit bloquée. Pour un produit qui ne reconnaît pas les paquets SOP/SOP' ou SOP'', cela ressemble à un canal non inactif, conduisant à des paquets manquants et des relances. Les communications entre les ports partenaires ont la priorité, ce qui signifie que les communications avec la fiche de câble peuvent être interrompues, ce qui ne conduit cependant pas à une réinitialisation logicielle ou matérielle.

Lorsqu'aucun contrat ou contrat implicite n'est en place (par exemple après une permutation des rôles d'alimentation ou une permutation rapide des rôles), seul le port source fournissant VCONN est autorisé à envoyer des paquets à une fiche de câble (SOP'/SOP'') et est autorisé à répondre à des paquets provenant de la fiche de câble (SOP'/SOP'') avec un GoodCRC afin de découvrir les caractéristiques de la fiche de câble (voir Figure 2-2). Pendant cette phase, toutes les communications avec la fiche de câble sont initiées et contrôlées par la source qui agit pour prévenir les conflits entre paquets SOP*. Le destinataire ne communique pas avec la fiche de câble, même s'il s'agit du port DFP, et il ***Rejette*** tous les paquets SOP' reçus.

Lorsqu'un contrat explicite est en place, la source VCONN (le port DFP ou le port UFP) peut communiquer avec la ou les fiches de câbles en utilisant les paquets SOP'/SOP'' (voir Figure 2-2). Pendant cette phase, toutes les communications avec la fiche de câble sont initiées et contrôlées par la source VCONN qui agit pour prévenir les conflits entre paquets SOP*. Le port qui n'est pas la source VCONN ne communique pas avec la fiche de câble et ne reconnaît aucun paquet SOP'/SOP'' reçu. Seul le port DFP, lorsqu'il agit en tant que source VCONN, est autorisé à envoyer des SOP* afin de contrôler l'entrée et la sortie des modes et de gérer le fonctionnement modal.

Figure 2-2 Exemple de communication SOP' entre source VCONN et fiche(s) de câble



Anglais	Français
VCONN Source (DFP/UFP)	Source VCONN (DFP/UFP)
Cable Plug[1]	Fiche de câble[1]
Electronically Marked Cable	Câble marqué électroniquement
SOP signaling	Signal SOP

[1] Une fiche de câble peut être branchée physiquement sur le port DFP ou sur le port UFP.

2.6 Vue d'ensemble du fonctionnement

Un port d'alimentation électrique USB fournissant de l'électricité est connu en tant que source et un port consommant de l'électricité est connu en tant que destinataire. Il n'y a qu'un port source et qu'un port destinataire dans chaque connexion PD entre ports partenaires. Au branchement, le port source (le port avec Rp affirmée, voir ***[USB Type-C 2.0]***) est également le port DFP et la source VCONN. Au branchement, le port destinataire (le port avec Rd affirmée) est également le port UFP, mais il n'est pas la source VCONN.

Les rôles de source/destinataire, les rôles de DFP/UFP et le rôle de source VCONN peuvent tous par la suite être permutés deux à deux. Un port qui prend en charge à la fois le rôle de source et le rôle de destinataire est appelé port d'alimentation double fonction (DRP). Un port qui prend en charge à la fois le rôle de DFP et le rôle d'UFP est appelé port de données double fonction (DRD).

Lorsque la capacité de communication USB est prise en charge dans le rôle de DFP, le port est alors également en mesure d'agir en tant qu'hôte USB. De la même manière, lorsque la capacité de communication USB est prise en charge dans le rôle d'UFP, le port est alors également en mesure d'agir en tant que dispositif USB.

Les sections qui suivent décrivent le fonctionnement de haut niveau des ports qui assument les rôles de DFP, d'UFP, de source et de destinataire. Ces sections ne décrivent pas de fonctionnement non admis. Cependant, si un certain comportement n'est pas décrit, alors il n'est probablement pas pris en charge par la présente spécification.

Pour plus d'informations sur la façon dont l'alimentation électrique par port USB est mise en correspondance avec les états USB d'un dispositif PDUSB, voir 9.1.2.

2.6.1 Fonctionnement de la source

La source fonctionne différemment en fonction du statut de branchement:

Au branchement (aucune connexion PD ni aucun contrat):
- dans le cas d'un port qui n'est que source, la source détecte le branchement du destinataire;
- dans le cas d'un port DRP qui bascule, le port devient un port source lors du branchement d'un destinataire;
- la source met alors V_{BUS} à ***vSafe5V***;

Avant la connexion PD (aucune connexion PD ni aucun contrat PD):
- avant d'envoyer les messages ***Source_Capabilities***, la source peut détecter le type de câblage branché et peut modifier ses capacités annoncées en fonction du type de câble détecté:
  - la source essaie de communiquer avec l'une des fiches de câbles en utilisant les paquets SOP'. Si la fiche de câble répond, la communication prend place;
  - la capacité par défaut d'un câble USB Type-C est de 3 A, mais la communication SOP' est utilisée pour découvrir les autres capacités du câble;
- la source annonce périodiquement ses capacités en envoyant des messages ***Source_Capabilities*** à intervalles ***tTypeCSendSourceCap***;

A l'établissement de la connexion PD (aucune connexion PD ni aucun contrat):
- la présence d'un port partenaire apte à l'alimentation électrique par port USB est détectée:
  - soit en recevant un message ***GoodCRC*** en réponse à un message ***Source_Capabilities***;
  - soit en recevant un signal ***Hard Reset***;

A l'établissement d'un contrat explicite (connexion PD mais aucun contrat explicite ou contrat implicite après une permutation des rôles d'alimentation ou une permutation rapide des rôles):
- la source reçoit un message ***Request*** du destinataire et, s'il s'agit d'une demande ***Valide(s)***, répond par un message ***Accept*** suivi par un message ***PS_RDY*** lorsque son alimentation électrique est prête à fournir le niveau de puissance fixé par accord. A ce stade, un contrat explicite a été conclu;
- un port DFP qui ne génère pas de paquets SOP' ou SOP'' n'est pas tenu de détecter les paquets SOP' ou SOP'' et les ***Rejette***;

Pendant la connexion PD (contrat explicite - état ***PE_SRC_Ready***):

- la source traite et répond (si une réponse est exigée) à tous les messages reçus et elle envoie des messages appropriés chaque fois que sa politique locale l'exige:
  - la source informe le destinataire chaque fois que ses capacités varient, en envoyant un message ***Source_Capabilities***;
  - la source a toujours Rp affirmée sur son fil CC;
  - lorsque ce port est un port DRP, la source peut initier ou recevoir une demande de permutation des rôles d'alimentation. Après la permutation des rôles d'alimentation, ce port devient un destinataire et un contrat implicite est en place jusqu'à ce qu'un contrat explicite soit négocié immédiatement après;
  - lorsque ce port est un port DRD, la source peut initier ou recevoir une demande de permutation des rôles de transmission de données. Après permutation des rôles de transmission de données, le port DFP (hôte) devient un port UFP (dispositif). Le port reste une source et le rôle de source VCONN (ou non) reste inchangé;
  - la source peut initier ou recevoir une demande d'échange de source VCONN. Pendant une permutation de VCONN, VCONN est appliquée par les deux extrémités (établissement de la nouvelle connexion avant la coupure de l'autre). Le port reste une source et les rôles de DFP/UFP restent inchangés;
- lorsqu'il s'agit de la source VCONN, la source peut communiquer avec une fiche de câble en utilisant la communication SOP' ou SOP'' à tout moment où elle n'est pas engagée dans d'autres communications SOP:
  - si des paquets SOP sont reçus par la source pendant une communication SOP' ou SOP'', la communication SOP' ou SOP'' est immédiatement arrêtée (la temporisation de la fiche de câble expire et il n'y a pas de relance);
  - s'il est nécessaire que la source initie une communication SOP pendant une communication SOP' ou SOP'' en cours (par exemple pour une modification de capacités), alors les communications SOP' ou SOP'' sont interrompues;
- lorsque le port source est également un port DFP:
  - la source peut contrôler l'entrée et la sortie des modes dans la ou les fiches de câbles et commander le fonctionnement modal;
  - la source peut initier des VDM non structurés ou structurés;
  - la source peut contrôler l'entrée et la sortie de modes du destinataire et commander un fonctionnement modal en utilisant des VDM structurés;
- lorsque le port source fait partie d'un système à ports multiples:
  - il émet des demandes GotoMin si la réserve de puissance est nécessaire;

Echec de débranchement ou de communications:

- une source détecte le débranchement d'une fiche et abaisse VBUS à ***vSafe5V*** dans un délai ***tSafe5V*** et à ***vSafe0V*** dans un délai ***tSafe0V*** (en utilisant la détection de débranchement USB Type-C par CC);
- lorsque la source détecte l'échec de réception d'un message ***GoodCRC*** en réponse à un message dans un délai ***tReceive:***
  - cela conduit à une réinitialisation logicielle, dans un délai ***tSoftReset*** d'expiration du ***CRCReceiveTimer***;
  - si le processus de réinitialisation logicielle ne peut être achevé, une réinitialisation matérielle est déclenchée dans un délai ***tHardReset*** d'expiration du ***CRCReceiveTimer*** afin de rétablir VBUS à la valeur de fonctionnement USB par défaut dans un délai de ~1-1,5 s;
    - lorsque la source est aussi la source VCONN, VCONN subit également un cycle d'alimentation pendant la réinitialisation matérielle;
- lorsque la source fonctionnant en mode PPS ne parvient pas à recevoir de communication périodique (par exemple un message ***Request***) en provenance du destinataire dans un délai ***tPPSTimeout***:
  - la source procède à une réinitialisation matérielle et conduit VBUS à ***vSafe5V***;

- le fait de ne recevoir aucune réponse aux autres tentatives de communication est interprété comme une erreur par la source (voir traitement des erreurs);
- les erreurs pendant les transitions d'alimentation conduisent automatiquement à une réinitialisation matérielle afin de rétablir la puissance à ses niveaux par défaut.

Traitement des erreurs:

- les erreurs de protocole sont traitées par un message ***Soft_Reset***, émis par l'un ou l'autre des ports partenaires, qui réinitialise les compteurs, les temporisateurs et les états, mais qui ne modifie pas la tension et le courant négociés, ni le rôle du port (par exemple source, DFP/UFP, source VCONN), et n'entraîne pas la sortie du fonctionnement modal;
- les erreurs graves sont traitées par un signal ***Hard Reset*** émis par l'un ou l'autre des ports partenaires. Une réinitialisation matérielle:
  - réinitialise le protocole comme dans le cas d'une réinitialisation logicielle, mais fait également revenir l'alimentation électrique au fonctionnement USB par défaut (sortie ***vSafe0V*** ou ***vSafe5V***) afin de protéger le destinataire;
  - rétablit le rôle de transmission de données du port à celui de DFP;
  - a pour effet de couper VCONN lorsque le destinataire est la source VCONN. Le port source est alors rétabli en tant que source VCONN;
  - fait quitter tous les modes actifs de sorte que la source n'est plus en fonctionnement modal.
- Après une réinitialisation matérielle, le port partenaire est censé répondre au cours de ***tNoResponse***. Si ce n'est pas le cas,***nHardResetCount*** des réinitialisations matérielles supplémentaires sont effectuées avant que la source ne procède aux autres étapes de rétablissement sur erreur, comme défini dans ***[USB Type-C 2.0]***, en entrant dans l'état ***ErrorRecovery***.

2.6.2 Fonctionnement du destinataire

Au branchement (aucune connexion PD ni aucun contrat):

- le destinataire détecte le branchement de la source par la présence de ***vSafe5V***;
- dans le cas d'un port DRP qui bascule, le port devient un port destinataire lors du branchement d'une source;
- dès que le destinataire détecte la présence de la tension ***vSafe5V*** sur V_{BUS}, il attend un message ***Source_Capabilities*** indiquant la présence d'une source apte à l'alimentation électrique par port USB;
- si le destinataire ne reçoit pas de message ***Source_Capabilities*** dans un délai ***tTypeCSinkWaitCap***, il émet alors un signal ***Hard Reset*** afin d'entraîner l'envoi par le port source d'un message ***Source_Capabilities***, si le port source est apte à l'alimentation électrique par port USB;
- le destinataire ne génère pas de paquets SOP' ou SOP''. Il n'est pas tenu de détecter les paquets SOP' ou SOP'' et il ne les reconnaît pas.

A l'établissement de la connexion PD (aucune connexion PD ni aucun contrat):

- le destinataire reçoit un message ***Source_Capabilities*** et répond par un message ***GoodCRC***;
- le destinataire ne génère pas de paquets SOP' ou SOP''. Il n'est pas tenu de détecter les paquets SOP' ou SOP'' et les ***Rejette***.

A l'établissement d'un contrat explicite (connexion PD mais aucun contrat explicite ou contrat implicite après une permutation des rôles d'alimentation ou une permutation rapide des rôles):

- le destinataire reçoit de la source un message ***Source_Capabilities*** et répond par un message ***Request***. S'il s'agit d'une demande ***Valide(s)***, le destinataire reçoit un message ***Accept*** suivi d'un message ***PS_RDY*** lorsque l'alimentation de la source est prête à fournir la puissance au niveau fixé par accord. A ce stade, la source et le destinataire ont conclu un contrat explicite:
  - le port destinataire peut demander l'une des capacités offertes par la source, même s'il s'agit de la sortie ***vSafe5V*** offerte par ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** ou ***[USBBC 1.2]***, afin de permettre une future négociation de puissance:
    - un destinataire qui ne demande aucune capacité au moyen d'un message ***Request*** donne lieu à une erreur;
  - un destinataire incapable de fonctionner entièrement avec les capacités offertes demande la capacité par défaut, mais indique qu'il préférerait un autre niveau de puissance et fournit à l'utilisateur final une indication physique de la défaillance (par exemple au moyen d'une LED);
  - un destinataire ne génère pas de paquets SOP' ou SOP''. Il n'est pas tenu de détecter les paquets SOP' ou SOP'' et les ***Rejette***.

Pendant la connexion PD (contrat explicite - état ***PE_SNK_Ready***):

- le destinataire traite tous les messages reçus et y répond (si une réponse est exigée). Il envoie des messages appropriés chaque fois que sa politique locale l'exige;
- un destinataire dont les besoins en puissance ont changé en informe la source au moyen d'un nouveau message ***Request***. Le port destinataire peut demander l'une des capacités précédemment offertes par la source, même s'il s'agit de la sortie ***vSafe5V*** offerte par ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** ou ***[USBBC 1.2]***, afin de permettre une future négociation de puissance:
  - le fait de ne demander aucune capacité au moyen d'un message ***Request*** donne lieu à une erreur;
  - un destinataire incapable de fonctionner entièrement avec les capacités offertes demande l'une des capacités offertes, mais indique une incohérence de capacité, c'est-à-dire qu'il préférerait un autre niveau de puissance. Il fournit également à l'utilisateur final une indication physique de la défaillance (par exemple au moyen d'une LED);
- un destinataire fonctionnant en mode PPS envoie périodiquement un message ***Request*** au cours de ***tPPSRequest***, même si la demande n'a pas changé;

- le destinataire a toujours Rd affirmée sur son fil CC;
- lorsque ce port est un port DRP, le destinataire peut initier ou recevoir une demande de permutation des rôles d'alimentation. Après la permutation des rôles d'alimentation, ce port devient une source et un contrat implicite est en place jusqu'à ce qu'un contrat explicite soit négocié immédiatement après;
- lorsque ce port est un port DRD, le destinataire peut initier ou recevoir une demande de permutation des rôles de transmission de données. Après permutation des rôles de transmission de données, le port DFP (hôte) devient un port UFP (dispositif). Le port reste un destinataire et le rôle de source VCONN (ou non) reste inchangé;
- le destinataire peut initier ou recevoir une demande d'échange de source VCONN. Pendant une permutation de VCONN, VCONN est appliquée par les deux extrémités (établissement de la nouvelle connexion avant la coupure de l'autre). Le port reste un destinataire et les rôles de DFP/UFP restent inchangés;
- lorsqu'il s'agit de la source VCONN, le destinataire peut communiquer avec une fiche de câble en utilisant la communication SOP' ou SOP'' à tout moment où il n'est pas engagé dans d'autres communications SOP:
  - si des paquets SOP sont reçus par le destinataire pendant une communication SOP' ou SOP'', la communication SOP' ou SOP'' est immédiatement arrêtée (la temporisation de la fiche de câble expire et il n'y a pas de relance);
  - s'il est nécessaire que le destinataire initie une communication SOP pendant une communication SOP' ou SOP'' en cours (par exemple, pour une modification de capacités), alors les communications SOP' ou SOP'' sont interrompues;
  - lorsque le port destinataire est aussi un port DFP, le destinataire peut contrôler l'entrée et la sortie des modes dans la ou les fiches de câbles et commander le fonctionnement modal.
- lorsque le port destinataire est également un port DFP:
  - le destinataire peut initier des VDM non structurés ou structurés;
  - le destinataire peut contrôler l'entrée et la sortie de modes de la source et commander un fonctionnement modal en utilisant des VDM structurés.

Echec de débranchement ou de communications:

- Un destinataire détecte la disparition de V_{BUS} et interprète cela comme la fin de la connexion PD,
  - à moins que la tension ***vSafe0V*** ne soit due à une réinitialisation matérielle, à une permutation des rôles d'alimentation ou à une permutation rapide des rôles.
- Un destinataire détecte le débranchement d'une fiche et décharge V_{BUS}.
- lorsque le destinataire détecte l'échec de réception d'un message ***GoodCRC*** en réponse à un message dans un délai ***tReceive***:
  - cela conduit à une réinitialisation logicielle, dans un délai ***tSoftReset*** d'expiration du ***CRCReceiveTimer***;
  - Si le processus de réinitialisation logicielle ne peut être achevé, une réinitialisation matérielle est déclenchée dans un délai ***tHardReset*** d'expiration du ***CRCReceiveTimer*** afin de rétablir V_{BUS} à la valeur de fonctionnement USB par défaut dans un délai de ~1-1,5 s;
  - le fait de ne recevoir aucune réponse aux autres tentatives de communication est interprété comme une erreur par le destinataire (voir traitement des erreurs);
- les erreurs pendant les transitions d'alimentation conduisent automatiquement à une réinitialisation matérielle afin de rétablir la puissance à ses niveaux par défaut.

Traitement des erreurs:

- les erreurs de protocole sont traitées par un message ***Soft_Reset***, émis par l'un ou l'autre des ports partenaires, qui réinitialise les compteurs, les temporisateurs et les états, mais qui ne modifie pas la tension et le courant négociés, ni le rôle du port (par exemple destinataire, DFP/UFP, source VCONN), et n'entraîne pas la sortie du fonctionnement modal;
- les erreurs graves sont traitées par un signal ***Hard Reset*** émis par l'un ou l'autre des ports partenaires. Une réinitialisation matérielle:

  - réinitialise le protocole comme dans le cas d'une réinitialisation logicielle, mais fait également revenir l'alimentation électrique au fonctionnement USB par défaut (sortie ***vSafe0V*** ou ***vSafe5V***) afin de protéger le destinataire;
  - rétablit le rôle de transmission de données du port à celui d'UFP;
  - a pour effet de couper VCONN lorsque le destinataire est la source VCONN. Le port source est alors rétabli en tant que source VCONN;
  - fait quitter tous les modes actifs de sorte que la source n'est plus en fonctionnement modal.
- Après une réinitialisation matérielle, le port partenaire est censé répondre au cours de ***tTypeCSinkWaitCap***. Si ce n'est pas le cas, 2 réinitialisations matérielles supplémentaires sont effectuées avant que le port UFP ne se mette dans l'état ***PE_SNK_Wait_for_Capabilities***.

2.6.3 Fiches de câbles

- Les fiches de câbles sont alimentées en présence de VCONN, mais elles ne connaissent pas le statut du contrat.
- Les fiches de câbles n'initient aucune séquence de messages et ne font que répondre aux messages qui leur sont envoyés.
- Echec de débranchement ou de communications:
  - les communications peuvent être interrompues à tout moment;
  - il n'existe pas de schéma de temporisation applicable aux communications entre port DFP/UFP et fiche de câble;
  - la fiche de câble est prête à répondre à des demandes potentiellement répétées.
- Traitement des erreurs:
  - la fiche de câble détecte le signal Hard Reset pour déterminer que la source et le destinataire ont été réinitialisés et il est nécessaire qu'elle se réinitialise elle-même (équivalent à un cycle d'alimentation):
    - la fiche de câble ne peut pas générer elle-même de signal Hard Reset;
    - le processus de réinitialisation matérielle entraîne un cycle d'alimentation sur VBUS et Vconn, de sorte qu'il est prévu qu'il réinitialise de lui-même les fiches de câbles;
  - une fiche de câble détecte le signal Cable Reset pour déterminer qu'il est nécessaire qu'elle se réinitialise elle-même (équivalent à un cycle d'alimentation).

2.7 Vue d'ensemble de l'architecture

Cette architecture logique n'est pas destinée à être utilisée comme architecture de mise en œuvre. Par définition, une architecture de mise en œuvre fait partie de la définition d'un produit et, par conséquent, ne relève pas du domaine d'application de la présente spécification.

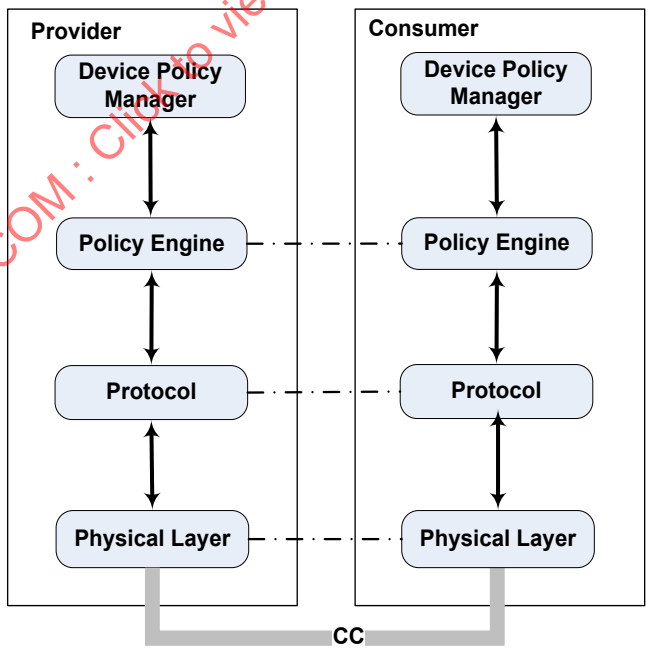
Cette section décrit l'architecture logique de haut niveau de l'alimentation électrique par port USB à laquelle il est fait référence dans l'ensemble de la présente spécification. Dans la pratique, diverses options de mise en œuvre sont possibles, sur la base de nombreuses possibilités de types différents de dispositifs PD. Les dispositifs PD peuvent avoir de nombreuses configurations différentes, par exemple avec communication USB ou non USB, port unique ou ports multiples, alimentations électriques dédiées ou alimentations électriques partagées sur des ports multiples, mises en œuvre matérielles ou sur logiciel, etc. L'architecture décrite dans la présente section est donc donnée à titre de référence uniquement, afin d'indiquer le modèle logique de haut niveau utilisé par la spécification PD. Cette architecture sert à identifier les concepts clés, ainsi qu'à indiquer les blocs logiques et les liaisons possibles entre eux.

L'architecture de l'alimentation électrique par port USB de chaque dispositif apte à l'alimentation électrique par port USB est constituée d'un certain nombre de composants essentiels.

La pile de communications donnée à la Figure 2-3 comprend:

- un **gestionnaire de politique d'utilisation des dispositifs** (voir 8.2) qui existe dans tous les dispositifs et qui gère les ressources de l'alimentation électrique par port USB au sein ceux-ci, sur un ou plusieurs ports et sur la base de la politique locale d'utilisation du dispositif;
- un **moteur de politique** (voir 8.3) qui existe dans chaque port d'alimentation électrique USB assurant la politique locale pour ce port;
- une **couche protocole** (voir 6) qui permet à des messages d'être échangés entre un port source et un port destinataire;
- une **couche physique** (voir 5) qui traite la transmission et la réception des bits sur le fil ainsi que la transmission des données.

Figure 2-3 Pile de communications de l'alimentation électrique par port USB

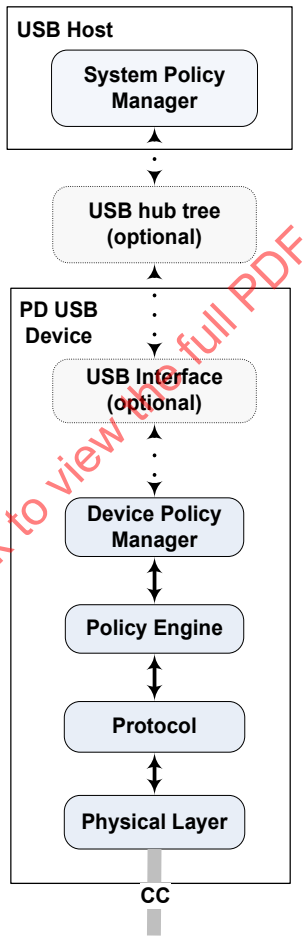


Anglais	Français
Device Policy Manager	Gestionnaire de politique d'utilisation des dispositifs
Policy Engine	Moteur de politique

Protocol	Protocole
Physical Layer	Couche physique
Consumer	Consommateur
Provider	Fournisseur

De plus, les appareils aptes à l'alimentation électrique USB qui peuvent fonctionner en tant que dispositifs USB peuvent communiquer sur USB (voir Figure 2-4). Un **gestionnaire de politique système *Facultatif(s/ve/ves),*** (voir Chapitre 9) qui réside dans l'hôte USB communique avec le dispositif PD sur USB par l'intermédiaire du port racine et potentiellement sur une arborescence de hubs USB. Le **gestionnaire de politique d'utilisation des dispositifs** interagit avec l'interface USB dans chaque dispositif afin de fournir et de mettre à jour les informations relatives à l'alimentation électrique dans le domaine USB. Noter qu'un dispositif PD n'est pas tenu d'avoir une interface de dispositif USB.

Figure 2-4 Communication de l'alimentation électrique par port USB sur USB



Anglais	Français
USB Host	Hôte USB
System Policy Manager	Gestionnaire de politique système
USB hub tree (optional)	Arbre de hub USB (facultatif)
PD USB Device	Dispositif d'alimentation USB
USB Interface (optional)	Interface USB (facultative)

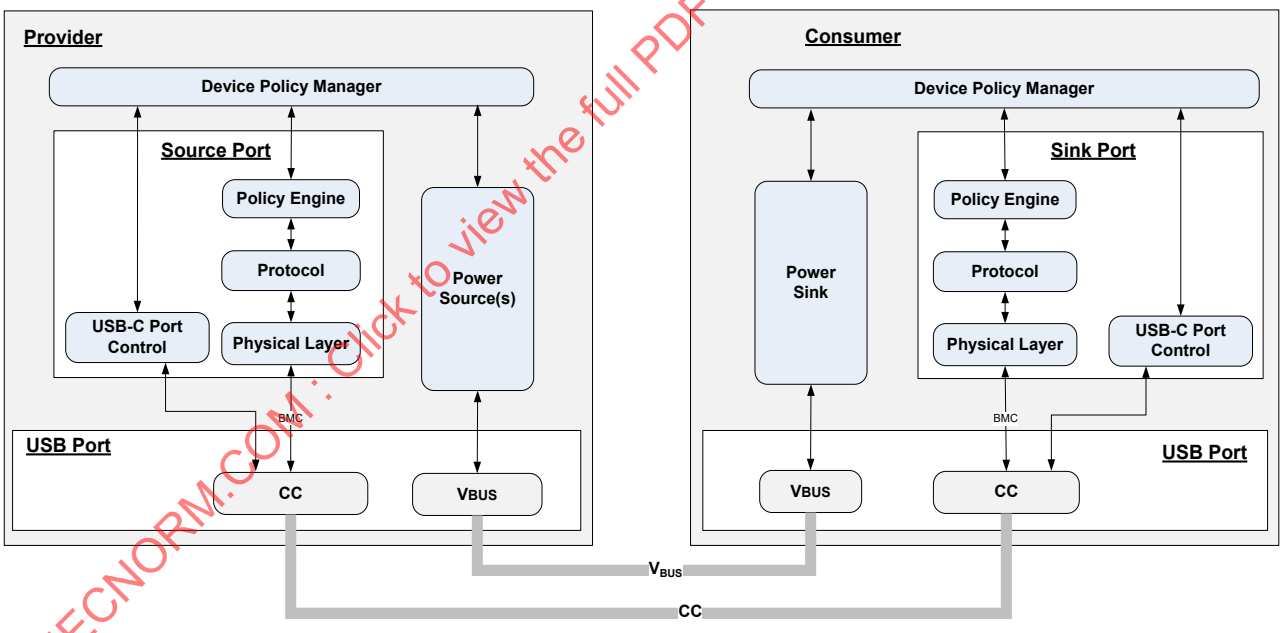
Device Policy Manager	Gestionnaire de politique d'utilisation des dispositifs
Policy Engine	Moteur de politique
Protocol	Protocole
Physical Layer	Couche Physique

La Figure 2-5 représente les blocs logiques entre deux ports PD branchés. En plus de la pile de communication décrite ci-dessus, il existe également:

pour un fournisseur ou un dispositif d'alimentation double fonction: une ou plusieurs **sources** alimentant un ou plusieurs ports;
pour un consommateur ou un dispositif d'alimentation double fonction: un **destinataire** utilisant l'alimentation;
un module de **contrôle de port USB-C** (voir 4.4) qui détecte le branchement/débranchement des câbles, comme défini dans *[USB Type-C 2.0]*;
l'alimentation électrique par port USB utilise le câblage normalisé défini dans *[USB Type-C 2.0]*.

Le **gestionnaire de politique d'utilisation des dispositifs** dialogue avec la pile de communications, la source/le destinataire et le bloc de contrôle de port USB-C afin de gérer les ressources du fournisseur ou du consommateur.

Figure 2-5 Vue d'une architecture à haut niveau



Anglais	Français
Provider	Fournisseur
Device Policy Manager	Gestionnaire de politique d'utilisation des dispositifs
Source Port	Port source
Policy Engine	Moteur de politique
Protocol	Protocole
Physical Layer	Couche physique
USB-C Port Control	Contrôle de port USB-C

Power Source(s)	Source(s) d'alimentation
USB Port	Port USB
Consumer	Consommateur
Power Sink	Destinataire d'alimentation
Sink Port	Port destinataire

2.7.1 Politique

Il existe deux niveaux de politique possibles:

1) la politique système appliquée à l'ensemble du système par le gestionnaire de politique système, sur de multiples fournisseurs ou consommateurs;
2) la politique locale imposée à un fournisseur ou à un consommateur par le gestionnaire de politique d'utilisation des dispositifs.

Une politique comprend plusieurs blocs logiques:

- le gestionnaire de politique système (sur l'ensemble du système);
- le gestionnaire de politique d'utilisation des dispositifs (un par fournisseur ou par consommateur);
- le moteur de politique (un par port source ou destinataire).

2.7.1.1 Gestionnaire de politique système

Le protocole d'alimentation électrique par port USB étant essentiellement de type point à point, la mise en œuvre d'une politique système exige une communication par un mécanisme de communication de données supplémentaire, c'est-à-dire par USB. Le gestionnaire de politique système surveille et contrôle la politique système entre les divers fournisseurs et consommateurs connectés par USB. Le gestionnaire de politique système réside dans l'hôte USB et communique par USB avec le gestionnaire de politique d'utilisation des dispositifs dans chacun des dispositifs connectés. Les dispositifs sans capacités de communication de données USB ou qui n'ont pas de données connectées ne sont pas en mesure de participer à la politique système.

Le gestionnaire de politique système est ***Facultatif(s/ve/ves),***dans un système donné. Par conséquent, les fournisseurs et les consommateurs d'alimentation électrique par port USB peuvent fonctionner sans sa présence. Cela comprend les systèmes dans lesquels l'hôte USB ne fournit pas de gestionnaire de politique système et peut également comprendre les systèmes "sans chef" qui ne comportent pas d'hôte USB. Dans les cas où il n'y a pas d'hôte, l'alimentation électrique par port USB est utile pour assurer la charge ou pour l'alimentation des dispositifs, dans la mesure où la fonctionnalité USB utile n'est pas possible. Lorsqu'il y a un hôte USB, mais qu'il n'y a pas de gestionnaire de politique système, les fournisseurs et les consommateurs peuvent négocier la puissance entre eux, indépendamment des règles d'alimentation USB, mais ils sont davantage limités en ce qui concerne les options disponibles pour gérer l'alimentation.

2.7.1.2 Gestionnaire de politique d'utilisation des dispositifs

Le gestionnaire de politique d'utilisation des dispositifs fournit des mécanismes de surveillance et de contrôle du système d'alimentation électrique par port USB au sein d'un consommateur ou d'un fournisseur particulier. Le gestionnaire de politique d'utilisation des dispositifs permet d'imposer des politiques locales dans le système, au moyen d'une communication avec le gestionnaire de politique système. Les politiques locales sont mises en œuvre port par port par le contrôle qu'exerce le gestionnaire de politique d'utilisation des dispositifs sur les ports sources/destinataires et par les communications avec le moteur de politique et le contrôle de port USB-C pour le port concerné.

2.7.1.3 Moteur de politique

Les fournisseurs et les consommateurs sont libres de mettre en œuvre leurs propres politiques locales sur leurs ports sources ou destinataires connectés directement. Celles-ci sont prises en charge par négociation et par des mécanismes de statut mis en œuvre par le moteur de politique pour ce port. Le moteur de politique interagit avec le gestionnaire de politique d'utilisation des dispositifs afin de déterminer la politique locale actuelle à imposer. Le moteur de politique est également informé par le gestionnaire de politique d'utilisation des dispositifs chaque fois qu'il y a une modification de la politique locale (par exemple une modification de capacités).

2.7.2 Formation et transmission des messages

2.7.2.1 Couche protocole

La couche protocole forme les messages servant à communiquer des informations entre une paire de ports. Elle est responsable de la formation des messages de capacités, des demandes et des acquittements. De plus, elle forme les messages utilisés pour permuter les rôles et maintenir la présence. Elle reçoit des entrées provenant du moteur de politique pour indiquer quels messages sont à envoyer, et elle retourne les réponses au moteur de politique.

Le protocole de base fait appel à un modèle de type "push" dans lequel le fournisseur envoie ses capacités au consommateur qui, à son tour, répond par une demande sur la base de l'offre. Cependant, le consommateur peut, de façon asynchrone, demander quelles sont les capacités actuelles du fournisseur. Il peut alors choisir une autre tension/intensité de courant.

Des messages étendus allant jusqu'à une taille de données de ***MaxExtendedMsgLen*** peuvent être envoyés et reçus à condition que la couche protocole détermine que les deux ports partenaires prennent en charge cette capacité. Lorsque l'un des deux ports partenaires ne prend pas en charge les messages étendus ayant une taille de données supérieure à ***MaxExtendedMsgLegacyLen***, la couche protocole prend alors en charge un mécanisme de fragmentation afin de découper les messages plus longs en plus petits fragments d'une taille de ***MaxExtendedMsgChunkLen***.

2.7.2.2 Couche PHY

La couche PHY est responsable de l'émission et de la réception des messages sur le fil CC USB Type-C, ainsi que de la gestion des données. Elle tente d'éviter les collisions sur le fil, en se rétablissant de la situation s'il s'en produit. Elle détecte également les erreurs dans les messages qui utilisent un CRC.

2.7.3 Anticollision

2.7.3.1 Moteur de politique

Le moteur de politique d'une source indique à la couche protocole le début et la fin de chaque séquence atomique de messages (AMS) initiée par la source. Le moteur de politique d'un destinataire indique à la couche protocole le début de chaque AMS initiée par le destinataire. Cela permet de coordonner l'initiation d'AMS entre les ports partenaires.

2.7.3.2 Couche protocole

La couche protocole de la source demande à la couche PHY de définir la valeur de Rp sur ***SinkTxOk*** pour indiquer que le destinataire peut initier une AMS en envoyant le premier message de la séquence. La couche protocole de la source demande à la couche PHY de définir la valeur de Rp sur ***SinkTxNG*** pour indiquer que le destinataire ne peut pas initier d'AMS, du fait que la source est sur le point d'en initier une.

La couche protocole du destinataire, lorsque le moteur de politique indique qu'une AMS est en train d'être initiée, attend que la valeur de Rp soit définie sur ***SinkTxOk*** avant d'initier l'AMS en envoyant le premier message de la séquence.

2.7.3.3 Couche PHY

La couche PHY de la source définit la valeur de Rp sur ***SinkTxOk*** ou ***SinkTxNG***, comme prescrit par la couche protocole. La couche PHY du destinataire détecte la présence de la valeur Rp actuelle et en informe la couche protocole.

2.7.4 Alimentation

2.7.4.1 Source

Chaque fournisseur comporte une ou plusieurs sources partagées entre un ou plusieurs ports. Ces sources sont contrôlées par la politique locale. Les sources démarrent en fonctionnement USB par défaut, dans lequel le port applique ***vSafe0V*** ou ***vSafe5V*** sur V_{BUS} et retourne à cet état au débranchement ou après un signal Hard Reset. Si la source applique ***vSafe0V*** par défaut, elle détecte les événements de branchement et fait passer sa sortie à ***vSafe5V*** dès la détection d'un branchement.

2.7.4.2 Destinataire

Par hypothèse, les consommateurs ont un destinataire connecté à un port. Ce destinataire est contrôlé par la politique locale. Les destinataires démarrent en fonctionnement USB par défaut, dans lequel le port peut fonctionner à ***vSafe5V***, avec les intensités de courant USB spécifiées par défaut, et retournent à cet état au débranchement ou après un signal Hard Reset.

2.7.4.3 Ports d'alimentation double fonction

Les ports d'alimentation double fonction sont aptes à fonctionner en tant que source ou que destinataire et à permuter entre les deux rôles en utilisant la permutation des rôles d'alimentation ou la permutation rapide des rôles.

2.7.4.4 Détection de batterie déchargée ou de pertes d'alimentation

[USB Type-C 2.0] définit les mécanismes destinés à communiquer avec un destinataire ou un port DRP et à le charger en cas de batterie déchargée.

2.7.4.5 Source VCONN

L'un des ports, initialement le port source, est la source VCONN. Les fiches de câbles utilisent cette alimentation pour déterminer quelle fiche de câble est SOP'. La responsabilité de l'alimentation VCONN peut être permutée entre les ports source et destinataire en établissant la nouvelle connexion avant la coupure de l'autre, afin de s'assurer que les fiches de câbles continuent d'être alimentées. Pour assurer une communication fiable avec les fiches de câbles, seule la source VCONN est autorisée à communiquer avec les fiches de câbles. Ayant une permutation des rôles d'alimentation, une permutation des rôles de transmission de données ou une permutation rapide des rôles, il est nécessaire que chacun des ports s'assure qu'il est la source VCONN s'il est nécessaire qu'il communique avec les fiches de câbles après la permutation.

2.7.5 DFP/UFP

2.7.5.1 Port orienté en aval (DFP)

Le port orienté en aval ou DFP est équivalent au port USB A dans la topologie USB. Le port DFP correspond également à l'hôte USB, mais uniquement si la communication USB est prise en charge alors que le port agit en tant que DFP. Les produits tels que les blocs d'alimentation peuvent être un port DFP alors qu'ils n'ont pas de capacité de communication USB. Le port DFP agit également comme le maître du bus lorsqu'il contrôle un fonctionnement modal alternatif.

2.7.5.2 Port orienté en amont (UFP)

Le port orienté en amont ou UFP est équivalent au port USB B dans la topologie USB. Le port UFP correspond également à l'appareil USB, mais uniquement si la communication USB est prise en charge

alors que le port agit en tant qu'UFP. Les produits qui assurent la charge peuvent être un port UFP alors qu'ils n'ont pas de capacité de communication USB.

2.7.5.3 Ports de données double fonction

Les ports de données double fonction sont capables de fonctionner en tant que DFP ou UFP et de procéder à une permutation entre les deux rôles par le biais de la permutation des rôles de transmission de données. Noter que les produits peuvent être des ports de données double fonction sans être des ports d'alimentation double fonction, c'est-à-dire qu'ils peuvent commuter logiquement entre les rôles DFP et UFP, même si ce sont des ports uniquement sources ou uniquement destinataires.

2.7.6 Câble et connecteurs

2.7.6.1 Contrôle de port USB-C

Le bloc de contrôle de port USB-C fournit des mécanismes pour informer le gestionnaire de politique d'utilisation des dispositifs des événements de branchement/débranchement de câble.

La spécification de l'alimentation électrique par port USB admet les câbles USB certifiés et les mécanismes de détection associés définis dans la spécification ***[USB Type-C 2.0]***.

2.7.7 Interactions entre dispositifs non PD, BC et PD

L'alimentation électrique par port USB ne fonctionne que lorsque deux dispositifs d'alimentation électrique par port USB sont connectés directement. Lorsqu'un dispositif se trouve dans un environnement mixte, alors que l'autre dispositif ne prend pas en charge la spécification de l'alimentation électrique par port USB, les règles existantes sur l'alimentation de ***vSafe5V*** définies dans les spécifications ***[USB 2.0]***, ***[USB 3.2]***, ***[USBBC 1.2]*** ou ***[USB Type-C 2.0]*** s'appliquent.

Deux cas à prendre en compte en premier lieu:

- l'hôte (DFP/source) n'est pas compatible avec l'alimentation par port USB et, de fait, n'envoie pas d'annonces. Un dispositif apte à l'alimentation électrique par port USB branché ne voit aucune annonce et fonctionne en utilisant les règles définies dans les spécifications ***[USB 2.0]***, ***[USB 3.2]***, ***[USBBC 1.2]*** ou ***[USB Type-C 2.0]***;
- le dispositif (UFP/destinataire) n'est pas compatible avec l'alimentation par port USB et, de fait, ne voit aucune annonce. L'hôte (DFP/source) continue de fournir ***vSafe5V*** à V_{BUS}, conformément aux spécifications ***[USB 2.0]***, ***[USB 3.2]***, ***[USBBC 1.2]*** ou ***[USB Type-C 2.0]***.

2.7.8 Règles d'alimentation

Les règles d'alimentation définissent les plages de tension et de courant offertes par les sources d'alimentation électrique par port USB et utilisées par un destinataire d'alimentation électrique par port USB, pour une valeur donnée de puissance PD. Voir 10 pour plus d'informations.

3. Assemblages de câbles et connecteurs USB Type-A et USB Type-B

Cette section est désormais ***Déconseillé(s/ée/ées)***. Voir *[USBPD 2.0]* pour plus d'informations sur les câbles et les connecteurs utilisés dans les scénarios faisant appel au schéma de signalisation BFSK, avec les connecteurs USB Type-A ou USB Type-B.

4. Exigences électriques

Ce chapitre traite des exigences électriques applicables à la plateforme pour la mise en œuvre de l'alimentation électrique par port USB.

4.1 Interopérabilité avec les autres spécifications USB

L'alimentation électrique par port USB ***Pouvoir/Peut/Peuvent*** coexister avec les spécifications ***[USB 2.0]***, ***[USB 3.2]***, ***[USBBC 1.2]*** et ***[USB Type-C 2.0]*** (USB Type-C®). Si un dispositif demande une alimentation par l'intermédiaire de la spécification de charge de batterie, puis de la spécification d'alimentation électrique par port USB, il ***Devoir/Doit/Doivent*** suivre la spécification d'alimentation électrique par port USB jusqu'à ce que la paire de ports soit débranchée ou jusqu'à une réinitialisation matérielle. Si la connexion d'alimentation électrique par port USB est perdue, le port ***Devoir/Doit/Doivent*** revenir à son état par défaut (voir 6.8.3).

4.2 Détection de batterie déchargée/Détection de port non alimenté

Le fonctionnement sur batterie déchargée/sans alimentation a lieu lorsqu'il est nécessaire qu'un dispositif USB fournisse une alimentation à un hôte USB dans des circonstances où l'hôte USB:

- a une batterie déchargée qu'il est nécessaire de charger; ou
- a perdu sa source d'alimentation; ou
- n'a pas de source d'alimentation; ou
- ne souhaite pas fournir d'alimentation.

L'opération de charge d'une batterie déchargée pour des connexions entre connecteurs USB Type-C est définie dans ***[USB Type-C 2.0]***.

4.3 Chute de tension ohmique d'un câble par rapport à la masse (chute de tension ohmique)

Chaque port PD destinataire apte aux communications USB peut être susceptible de générer une communication USB non fiable si la chute de tension à la terre descend en deçà de la plage de mode commun acceptable pour les circuits de données des émetteurs/récepteurs USB à grande vitesse en raison d'une consommation de courant excessive. Le câblage USB certifié est spécifié de manière que de telles erreurs ne se produisent généralement pas (voir ***[USB Type-C 2.0]***).

4.4 Détection du type de câble

Les assemblages de câbles normalisés USB Type-C sont prévus pour des tensions d'alimentation électrique supérieures à ***vSafe5V*** et des intensités de courant d'au moins 3 A (voir ***[USB Type-C 2.0]***). La source ***Devoir/Doit/Doivent*** limiter les capacités maximales qu'elle offre de manière à ne pas dépasser les capacités du type de câblage détecté.

Les sources capables de délivrer plus de 3 A ***Devoir/Doit/Doivent*** détecter le type de câble branché et limiter les capacités qu'elles offrent sur la base de la capacité de transport de courant du câble, établie d'après les capacités du câble déterminées au moyen de la commande ***Discover Identity*** (voir 6.4.4.2) envoyée à l'aide de la communication SOP' (voir 2.5) adressée à la fiche de câble. Le VDO de câble renvoyé comme partie de la commande ***Discover Identity*** précise les valeurs maximales d'intensité de courant et de tension qui ***Devoir/Doit/Doivent*** être négociées pour un câble donné au sein d'un contrat explicite.

Le processus de détection de câble est généralement mis en œuvre lorsque la source est alimentée, après une permutation des rôles d'alimentation ou une permutation rapide des rôles, ou encore lorsque l'alimentation électrique est appliquée à un destinataire. La méthode exacte utilisée pour détecter ces événements dépend du fabricant et ***Devoir/Doit/Doivent*** satisfaire aux exigences suivantes:

- les sources ***Devoir/Doit/Doivent*** suivre le processus de détection de câble avant que la source n'envoie de messages ***Source_Capabilities*** offrant des intensités de courant supérieures à 3 A et/ou des tensions supérieures à 20 V;
- les destinataires qui comportent des connecteurs USB Type-C ***Devoir/Doit/Doivent*** choisir des capacités parmi les capacités de source offertes, en prenant pour hypothèse que la source a déjà déterminé les capacités du câble;
- les destinataires dont le bit DRP est défini ***Devoir/Doit/Doivent*** répondre à un message ***Get_Source_Cap*** en déclarant leurs capacités de source complètes, sans les limiter en fonction des capacités du câble.

5. Couche physique

5.1 Vue d'ensemble de la couche physique

La couche physique (couche PHY) définit la technologie de signalisation pour l'alimentation électrique par port USB. Le présent chapitre définit les exigences et paramètres électriques de la couche physique PD, nécessaires à l'interopérabilité entre dispositifs aptes à l'alimentation électrique par port USB.

5.2 Fonctions de la couche physique

La couche physique apte à l'alimentation électrique par port USB est constituée d'une paire d'émetteurs et de récepteurs qui communiquent sur un fil de signal unique (CC). Toutes les communications se font en semi-duplex. La couche PHY permet d'éviter les collisions, de manière à réduire le plus possible les erreurs de communication sur le canal.

L'émetteur assure les fonctions suivantes:

réception de données de paquet en provenance de la couche protocole;
calcul et ajout d'un CRC;
encodage des données de paquet, y compris le CRC (c'est-à-dire la charge utile);
transmission du paquet (préambule, ***SOP****, charge utile, CRC et ***EOP***) sur le canal par le codage Biphase Mark Coding (BMC) sur CC.

Le récepteur assure les fonctions suivantes:

récupération de l'horloge et verrouillage sur le paquet à partir du préambule;
détection du SOP*;
décodage des données reçues, y compris le CRC;
détection de l'EOP et validation du CRC:

- si le CRC est ***Valide(s)***, remise des données de paquet à la couche protocole;
- si le CRC est ***Invalide(s)***, suppression des données reçues.

5.3 Codage de symboles

A l'exception du préambule, toutes les communications sur la ligne ***Devoir/Doit/Doivent*** être codées par codage en ligne pour assurer un niveau raisonnable d'équilibrage du courant continu et un nombre convenable de transitions. Ce codage simplifie la conception du récepteur et permet davantage de variations dans la conception du récepteur.

Le codage en ligne 4b5b ***Devoir/Doit/Doivent*** être utilisé. Il permet de coder des données de 4 bits en symboles de 5 bits pour la transmission et de décoder des symboles de 5 bits en données de 4 bits pour la réception.

Le code 4b5b assure le codage des données en utilisant des symboles spéciaux. Les symboles spéciaux sont utilisés pour le signal ***Hard Reset*** et pour identifier les limites des paquets.

Tableau 5-1 Tableau de codage de symboles 4b5b

Nom	4b	Symbole 5b	Description
0	0000	11110	donnée hexa 0
1	0001	01001	donnée hexa 1
2	0010	10100	donnée hexa 2
3	0011	10101	donnée hexa 3
4	0100	01010	donnée hexa 4
5	0101	01011	donnée hexa 5
6	0110	01110	donnée hexa 6
7	0111	01111	donnée hexa 7
8	1000	10010	donnée hexa 8
9	1001	10011	donnée hexa 9
A	1010	10110	donnée hexa A
B	1011	10111	donnée hexa B
C	1100	11010	donnée hexa C
D	1101	11011	donnée hexa D
E	1110	11100	donnée hexa E
F	1111	11101	donnée hexa F
Sync-1	Code K	11000	Startsynch #1
Sync-2	Code K	10001	Startsynch #2
RST-1	Code K	00111	Hard Reset #1
RST-2	Code K	11001	Hard Reset #2
EOP	Code K	01101	EOP Fin de paquet
Réservé(s/ée/ées)	Erreur	00000	***Ne doit/doivent pas*** être utilisé
Réservé(s/ée/ées)	Erreur	00001	***Ne doit/doivent pas*** être utilisé
Réservé(s/ée/ées)	Erreur	00010	***Ne doit/doivent pas*** être utilisé
Réservé(s/ée/ées)	Erreur	00011	***Ne doit/doivent pas*** être utilisé
Réservé(s/ée/ées)	Erreur	00100	***Ne doit/doivent pas*** être utilisé
Réservé(s/ée/ées)	Erreur	00101	***Ne doit/doivent pas*** être utilisé
Sync-3	Code K	00110	Startsynch #3
Réservé(s/ée/ées)	Erreur	01000	***Ne doit/doivent pas*** être utilisé

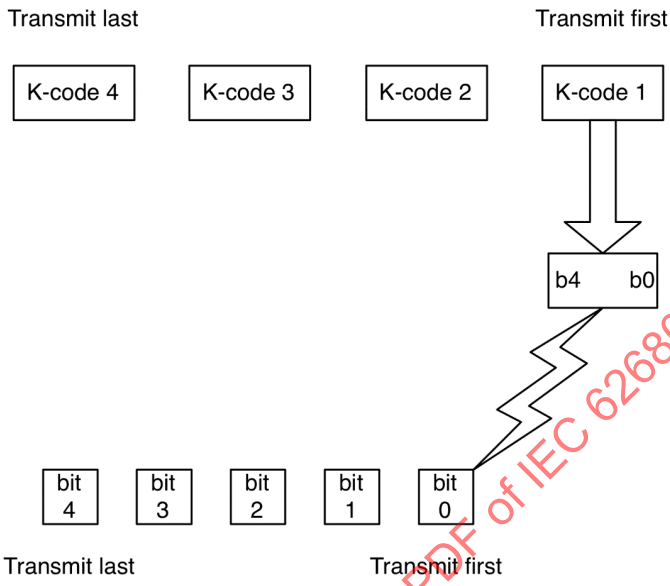
Nom	4b	Symbole 5b	Description
Réservé(s/ée/ées)	Erreur	01100	***Ne doit/doivent pas*** être utilisé
Réservé(s/ée/ées)	Erreur	10000	***Ne doit/doivent pas*** être utilisé
Réservé(s/ée/ées)	Erreur	11111	***Ne doit/doivent pas*** être utilisé

5.4 Ensembles ordonnés

Les ensembles ordonnés ***Devoir/Doit/Doivent*** être interprétés conformément à la Figure 5-1.

Un ensemble ordonné est constitué de codes 4K envoyés comme le montre la Figure 5-1.

Figure 5-1 Interprétation d'ensembles ordonnés



Anglais	Français
Transmit last	Transmettre en dernier
Transmit first	Transmettre en premier
K-code	Code K

Une liste des ensembles ordonnés utilisés par l'alimentation électrique par port USB peut être consultée dans le Tableau 5-2. ***SOP**** est un terme générique utilisé à la place de ***SOP***/***SOP'***/***SOP''***.

Tableau 5-2 Ensembles ordonnés

Ensemble ordonné	Référence
Cable Reset	Section 5.6.5
Hard Reset	Section 5.6.4
SOP	Section 5.6.1.2.1
SOP'	Section 5.6.1.2.2
SOP'_Debug	Section 5.6.1.2.4
SOP''	Section 5.6.1.2.3
SOP''_Debug	Section 5.6.1.2.5

Le récepteur ***Devoir/Doit/Doivent*** rechercher les quatre codes K. Lorsque le récepteur trouve les quatre codes K au bon endroit, il ***Devoir/Doit/Doivent*** les interpréter comme un ensemble ordonné ***Valide(s)***. Lorsque le récepteur trouve trois des quatre codes K au bon endroit, il ***Pouvoir/Peut/Peuvent*** les interpréter comme un ensemble ordonné ***Valide(s)***. ***Il convient d'/de/qu'/que*** le récepteur s'assure que les quatre codes K sont ***Valide(s)*** afin d'éviter toute ambiguïté lors de la détection (voir Tableau 5-3).

Tableau 5-3 Validation d'ensembles ordonnés

	1er code	2e code	3e code	4e code
Valide(s)[1]	Corrompu	Code K	Code K	Code K
Valide(s)[1]	Code K	Corrompu	Code K	Code K
Valide(s)[1]	Code K	Code K	Corrompu	Code K
Valide(s)[1]	Code K	Code K	Code K	Corrompu
Valide(s)[2] (parfait)	Code K	Code K	Code K	Code K
Invalide(s) (exemple)	Code K	Corrompu	Code K	Corrompu

1. ***Pouvoir/Peut/Peuvent*** être interprété comme un ensemble ordonné ***Valide(s)***.
2. ***Devoir/Doit/Doivent*** être interprété comme un ensemble ordonné ***Valide(s)***.

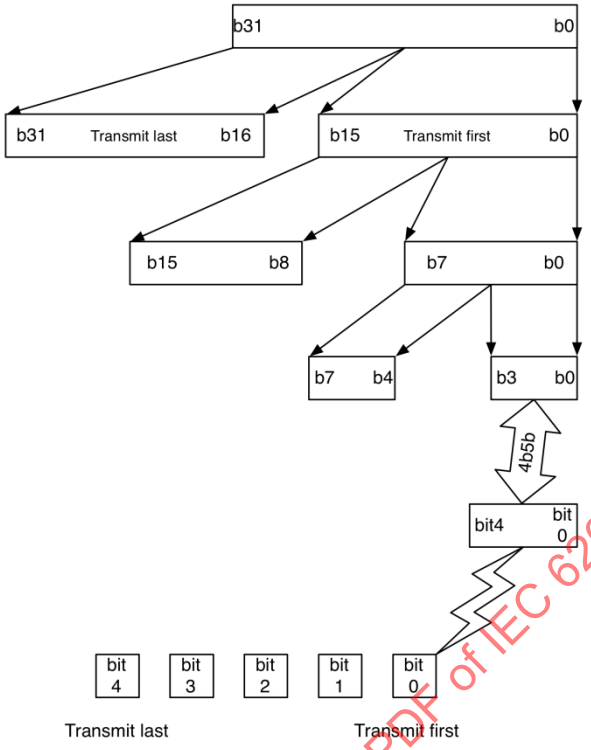
5.5 Ordonnancement des bits transmis

Cette section décrit l'ordre des bits sur le fil qui ***Devoir/Doit/Doivent*** être utilisé lors de la transmission de données de différentes tailles. Le Tableau 5-4 montre les différentes tailles de données possibles. La Figure 5-2 indique l'ordre de transmission qui ***Devoir/Doit/Doivent*** être suivi.

Tableau 5-4 Taille des données

	Sans encodage	Avec encodage
Octet	8 bits	10 bits
Mot	16 bits	20 bits
DWord	32 bits	40 bits

Figure 5-2 Ordre de transmission pour diverses tailles de données

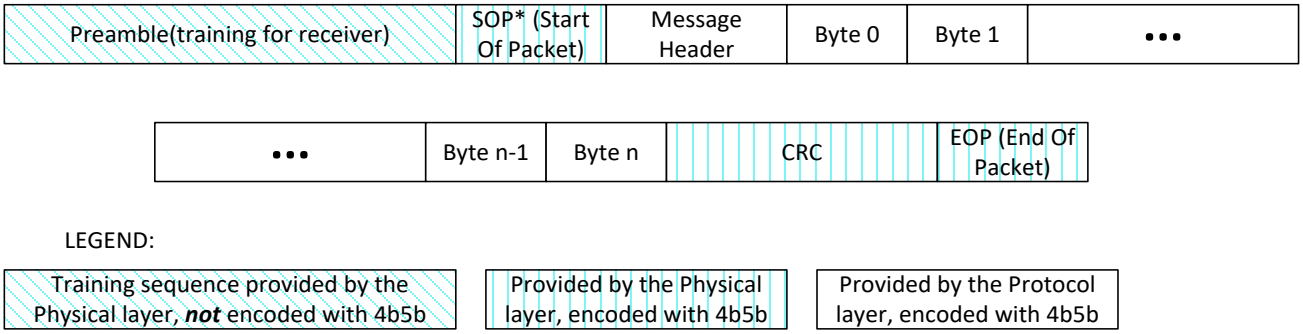


Anglais	Français
Transmit last	Transmettre en dernier
Transmit first	Transmettre en premier

5.6 Format de paquet

Le format de paquet ***Devoir/Doit/Doivent*** être constitué d'un préambule, d'un ***SOP**** (voir 5.6.1.2), de données de paquet, y compris l'en-tête du message, d'un CRC et d'un ***EOP*** (voir 5.6.1.5). Le format de paquet est représenté à la Figure 5-3. Il indique les parties du paquet qui ***Devoir/Doit/Doivent*** être encodées 4b/5b. Lorsque l'encodage 4b/5b est effectué, la totalité du paquet ***Devoir/Doit/Doivent*** être transmise en utilisant le codage BMC sur CC. Noter que tous les bits du paquet, y compris le préambule, sont encodés avec le code BMC. Voir 6.2.1 pour plus d'informations sur la construction des paquets pour les messages de contrôle, les messages de données et les messages étendus.

Figure 5-3 Format de paquet pour l'alimentation électrique par port USB



Anglais	Français
Preamble (training for receiver)	Préambule (entraînement pour le récepteur)
SOP* (Stat of Packet)	SOP* (début de paquet)
Message Header	En-tête de message
Byte	Octet
EOP (End of Packet)	EOP (fin de paquet)
LEGEND:	LEGENDE:
Training sequence provided by the Physical layer, ***not*** encoded with 4b5b	Séquence d'entraînement fournie par la couche physique, ***non*** codée en 4b5b
Provided by the Physical layer, encoded with 4b5b	Fourni par la couche physique, codé en 4b5b
Provided by the Protocol layer, encoded with 4b5b	Fourni par la couche protocole, codé en 4b5b

5.6.1 Mise en trames des paquets

La transmission commence par un préambule permettant au récepteur de se verrouiller sur la porteuse. Ce préambule est suivi d'un ***SOP****** (début de paquet). Le paquet se termine par un code K ***EOP*** (fin de paquet).

5.6.1.1 Préambule

Le préambule sert à effectuer un verrouillage dans le récepteur par la présentation d'une série alternée de "0" et de "1". Ainsi, la fréquence moyenne est la fréquence porteuse. A la différence du reste du paquet, le préambule ***Ne doit/doivent pas*** être codé 4b/5b.

Le préambule ***Devoir/Doit/Doivent*** consister en une séquence de 64 bits alternant les 0 et les 1. Le préambule ***Devoir/Doit/Doivent*** commencer par un "0" et ***Devoir/Doit/Doivent*** se terminer par un "1".

5.6.1.2 Séquences de début de paquet

5.6.1.2.1 Séquence de début de paquet (SOP)

SOP est en ensemble ordonné. L'ensemble ordonné ***SOP*** est défini par: trois codes K ***Sync-1*** suivis d'un code K ***Sync-2*** (voir Tableau 5-5).

Tableau 5-5 Ensemble ordonné SOP

Numéro de code K	Code K dans le tableau de codage
1	***Sync-1***
2	***Sync-1***
3	***Sync-1***
4	***Sync-2***

Une source ou un destinataire apte à l'alimentation électrique ***Devoir/Doit/Doivent*** être capable de détecter des paquets avec ***SOP*** et de communiquer avec eux. Si un ***SOPValide(s)*** n'est pas détecté (voir Tableau 5-3), la totalité de la transmission ***Devoir/Doit/Doivent*** être ***Rejeté(s/ée/ées)***.

L'émission et la réception de paquets SOP ***Devoir/Doit/Doivent*** être limitées aux ports aptes à l'alimentation électrique USB sur les hôtes PDUSB et les dispositifs PDUSB. Les fiches de câbles et les VPD ne ***Devoir/Doit/Doivent*** ni émettre ni recevoir de paquets SOP. Noter que les appareils PDUSB, même s'ils ont la forme physique d'un câble (comme les AMA), sont cependant tenus de répondre aux paquets SOP.

5.6.1.2.2 Séquence de début de paquet "prime" (SOP')

L'ensemble ordonné ***SOP'*** est défini par: deux codes K ***Sync-1*** suivis de deux codes K ***Sync-3*** (voir Tableau 5-6).

Tableau 5-6 Ensemble ordonné SOP'

Numéro de code K	Code K dans le tableau de codage
1	***Sync-1***
2	***Sync-1***
3	***Sync-3***
4	***Sync-3***

Un VPD ***Devoir/Doit/Doivent*** avoir une capacité de communication SOP'. Un VPD et une fiche de câble aptes aux communications SOP' ***Devoir/Doit/Doivent*** uniquement détecter et communiquer avec des paquets commençant par ***SOP'***.

Un port qui a besoin de communiquer avec une fiche de câble apte aux communications SOP', branchée entre une paire de ports, est capable de communiquer en utilisant les deux paquets commençant par ***SOP'*** pour communiquer avec la fiche de câble et commençant par ***SOP*** pour communiquer avec son port partenaire.

Dans le cas d'un VPD ou d'une fiche de câble qui prend en charge les communications SOP', si un ***SOP' Valide(s)*** n'est pas détecté (voir Tableau 5-3) alors la totalité de la transmission ***Devoir/Doit/Doivent*** être ***Rejeté(s/ée/ées)***. Dans le cas d'un port prenant en charge les communications SOP', si un ***SOP*** ou un ***SOP' Valide(s)*** n'est pas détecté (voir Tableau 5-3) la totalité de la transmission ***Devoir/Doit/Doivent*** être ***Rejeté(s/ée/ées)***. En l'absence de contrat explicite ou de contrat implicite en place, un destinataire ***Ne doit/doivent pas*** envoyer de paquets SOP' et ***Devoir/Doit/Doivent Rejeter*** tous les paquets commençant par ***SOP'***.

5.6.1.2.3 Séquence de début de paquet "seconde" (SOP'')

L'ensemble ordonné ***SOP''*** est défini par la séquence de codes K suivante: ***Sync-1***, ***Sync-3***, ***Sync-1***, ***Sync-3*** (voir Tableau 5-7).

Tableau 5-7 Ensemble ordonné SOP''

Numéro de code K	Code K dans le tableau de codage
1	***Sync-1***
2	***Sync-3***
3	***Sync-1***
4	***Sync-3***

Un VPD ***Ne doit/doivent pas*** avoir de capacité de communication SOP''. Une fiche de câble apte aux communications SOP'' ***Devoir/Doit/Doivent*** avoir une capacité de communications SOP' dans l'autre fiche de câble. Aucun câble ne ***Devoir/Doit/Doivent*** prendre en charge que les communications SOP''. Une fiche de câble à laquelle une communication SOP'' est attribuée ***Devoir/Doit/Doivent*** uniquement détecter et communiquer avec des paquets commençant par ***SOP''*** et ***Devoir/Doit/Doivent Rejeter*** tous les autres paquets.

Un port qui a besoin de communiquer avec une fiche de câble branchée entre une paire de ports est capable de communiquer en utilisant des paquets qui commencent par ***SOP'*** et ***SOP''*** pour communiquer avec les fiches de câbles et des paquets qui commencent par ***SOP*** pour communiquer avec son port partenaire. Un port qui prend en charge la communication SOP'' ***Devoir/Doit/Doivent*** également prendre en charge la communication SOP', et il ***Devoir/Doit/Doivent*** coordonner la communication SOP* de manière à éviter les collisions.

Dans le cas d'une fiche de câble prenant en charge les communications SOP'', si un ***SOP''*** ***Valide(s)*** n'est pas détecté (voir Tableau 5-3) alors la totalité de la transmission ***Devoir/Doit/Doivent*** être ***Rejeté(s/ée/ées)***. Pour le port, si un ***SOP**** ***Valide(s)*** n'est pas détecté (voir Tableau 5-3), alors la totalité de la transmission ***Devoir/Doit/Doivent*** être ***Rejeté(s/ée/ées)***.

5.6.1.2.4 Séquence de début de paquet "prime debug" (SOP'_Debug)

L'ensemble ordonné ***SOP'_Debug*** est défini par la séquence de codes K suivante: ***Sync-1***, ***RST-2***, ***RST-2***, ***Sync-3*** (voir Tableau 5-8). L'utilisation de cet ensemble ordonné n'est actuellement pas définie.

Tableau 5-8 Ensemble ordonné SOP'_Debug

Numéro de code K	Code K dans le tableau de codage
1	***Sync-1***
2	***RST-2***
3	***RST-2***
4	***Sync-3***

5.6.1.2.5 Séquence de début de paquet "seconde debug" (SOP''_Debug)

L'ensemble ordonné ***SOP''_Debug*** est défini par la séquence de codes K suivante: ***Sync-1***, ***RST-2***, ***Sync-3,*** ***Sync-2*** (voir Tableau 5-9). L'utilisation de cet ensemble ordonné n'est actuellement pas définie.

Tableau 5-9 Ensemble ordonné SOP''_Debug

Numéro de code K	Code K dans le tableau de codage
1	***Sync-1***
2	***RST-2***
3	***Sync-3***
4	***Sync-2***

5.6.1.3 Charge utile d'un paquet

La charge utile d'un paquet est fournie par la couche protocole (6.2) et ***Devoir/Doit/Doivent*** être encodée avec les codes de données hexadécimales du Tableau 5-1.

5.6.1.4 CRC

Le CRC ***Devoir/Doit/Doivent*** être inséré juste après la charge utile. Il est décrit en 5.6.2.

5.6.1.5 Fin de paquet (EOP)

Le marqueur de fin de paquet ***Devoir/Doit/Doivent*** être un code K ***EOP*** unique, défini dans le Tableau 5-1. Ce code ***Devoir/Doit/Doivent*** marquer la fin du CRC. Après l'***EOP***, le CRC résiduel ***Devoir/Doit/Doivent*** être vérifié. Si le CRC est incorrect, toute la transmission ***Devoir/Doit/Doivent*** être ***Rejeté(s/ée/ées)***. S'il est correct, le paquet ***Devoir/Doit/Doivent*** être transmis à la couche protocole. Noter qu'un ***EOP*** ***Pouvoir/Peut/Peuvent*** être utilisé pour mettre fin à un paquet prématurément, par exemple avant l'envoi d'un signal ***Hard Reset***.

5.6.2 CRC

L'en-tête de message et les données ***Devoir/Doit/Doivent*** être protégés par un code CRC de 32 bits.

Le CRC-32 protège l'intégrité de la charge utile de données. Le CRC-32 est défini comme suit:

- le CRC-32 polynômial ***Devoir/Doit/Doivent*** être = 04C1 1DB7h;
- la valeur initiale du CRC-32 ***Devoir/Doit/Doivent*** être = FFFF FFFFh;

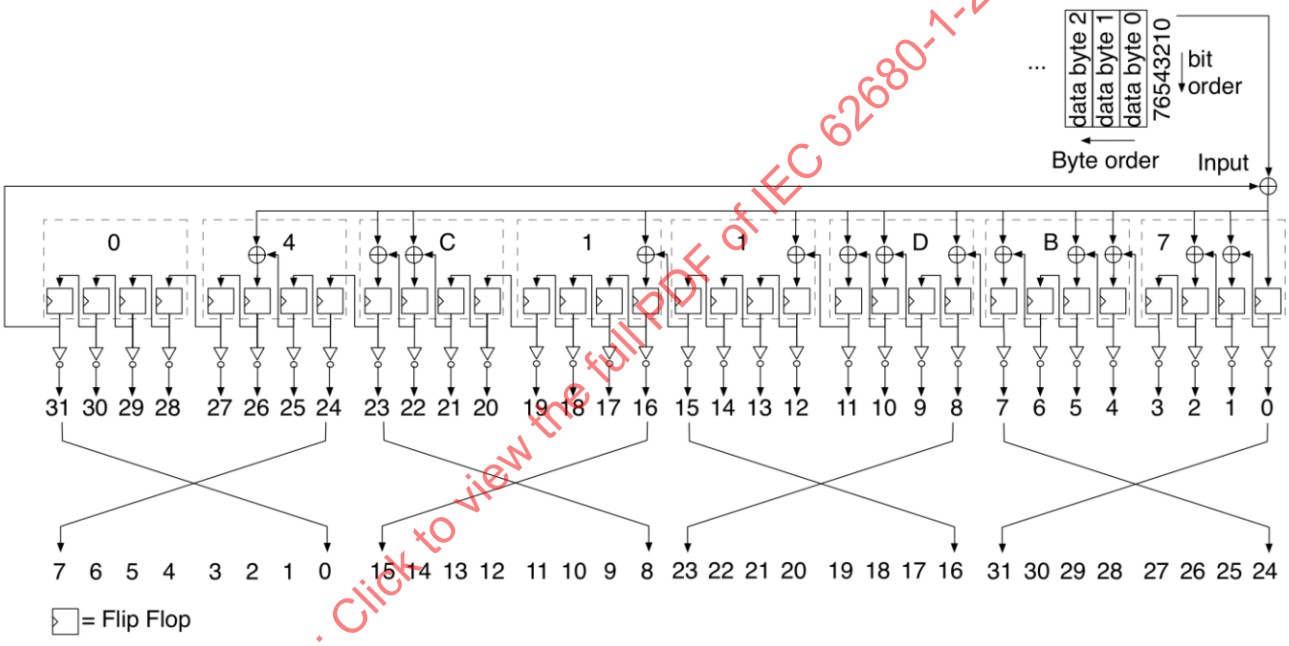
- le CRC-32 ***Devoir/Doit/Doivent*** être calculé pour tous les octets de la charge utile, à l'exclusion des symboles de mise en trame des paquets (c'est-à-dire que sont exclus le préambule, le ***SOP**** et l'***EOP***);
- le calcul du CRC-32 ***Devoir/Doit/Doivent*** commencer par le bit 0 de l'octet 0 et se poursuivre jusqu'au bit 7 de chacun des octets du paquet;
- le reste du code CRC-32 ***Devoir/Doit/Doivent*** être complémenté;
- la valeur résiduelle du code CRC-32 ***Devoir/Doit/Doivent*** être C704 DD7Bh.

Note: Cette inversion du reste du CRC-32 ajoute un décalage de FFFF FFFFh qui génère une valeur résiduelle CRC-32 constante de C704 DD7Bh côté récepteur.

Note: La mise en œuvre du CRC est identique à celle utilisée dans ***[USB 3.2]***.

La Figure 5-4 donne un exemple de génération de code CRC-32. L'ordre des bits de sortie ***Devoir/Doit/Doivent*** être celui spécifié dans le Tableau 5-10.

Figure 5-4 Génération de code CRC 32



Anglais	Français
Byte order	Ordre d'octets
Bit order	Ordre de bits
Input	Entrée
Data byte	Octet de données
Flip Flop	Bascule

Tableau 5-10 Mise en correspondance du CRC-32

CRC-32	Position du bit de résultat dans le champ CRC-32
0	31
1	30
2	29
3	28
4	27

CRC-32	Position du bit de résultat dans le champ CRC-32
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
25	6
26	5
27	4
28	3
29	2
30	1
31	0

Le CRC-32 ***Devoir/Doit/Doivent*** être codé avant la transmission.

5.6.3 Erreurs de détection de paquet

Les erreurs de CRC ou les erreurs détectées lors du décodage des symboles codés en utilisant le tableau de codage ***Devoir/Doit/Doivent*** être traitées de la même manière. Le message ***Devoir/Doit/Doivent*** être ***Rejeté(s/ée/ées)*** et le message ***GoodCRC*** ***Ne doit/doivent pas*** être renvoyé.

Tandis que le récepteur est en train de traiter un paquet, à tout moment où la ligne CC devient inactive, le récepteur ***Devoir/Doit/Doivent*** arrêter de traiter le paquet et le ***Rejeter*** (aucun message ***GoodCRC*** n'est renvoyé). Voir 5.8.6.1 pour la définition de BMC inactif.

5.6.4 Réinitialisation matérielle

Le signal ***Hard Reset*** est un ensemble ordonné d'octets envoyé dans le but d'être reconnu par la couche PHY. L'ensemble ordonné du signal ***Hard Reset*** est défini par: trois codes K ***RST-1*** suivis d'un code K ***RST-2*** (voir Tableau 5-11).

Tableau 5-11 Ensemble ordonné Hard Reset

Numéro de code K	Code K dans le tableau de codage
1	***RST-1***
2	***RST-1***
3	***RST-1***

Numéro de code K	Code K dans le tableau de codage
4	*RST-2*

Un dispositif ***Devoir/Doit/Doivent*** effectuer une réinitialisation matérielle lorsqu'il reçoit un signal ***Hard Reset***. Après avoir reçu le signal ***Hard Reset***, le dispositif ***Devoir/Doit/Doivent*** se réinitialiser, comme décrit en 6.8.3. Si un ***Hard Reset*****Valide(s)** n'est pas détecté (voir Tableau 5-3), la totalité de la transmission ***Devoir/Doit/Doivent*** être ***Rejeté(s/ée/ées)***.

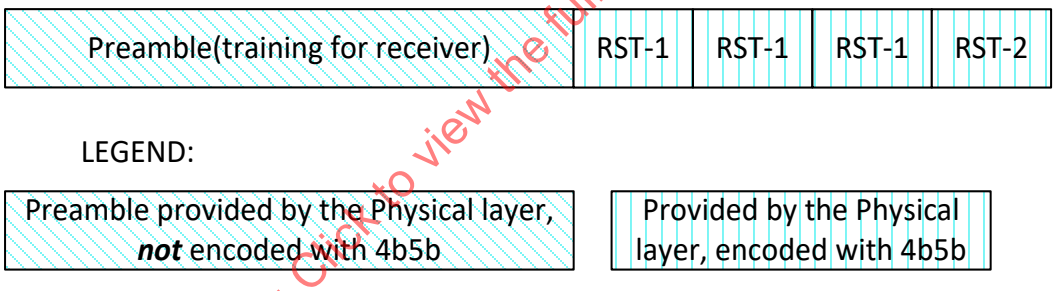
Une fiche de câble ***Devoir/Doit/Doivent*** effectuer une réinitialisation matérielle lorsqu'elle détecte l'envoi d'un signal ***Hard Reset*** entre les ports partenaires. Après avoir reçu le signal ***Hard Reset***, le dispositif ***Devoir/Doit/Doivent*** se réinitialiser, comme décrit en 6.8.3.

La procédure d'envoi du signal ***Hard Reset*** ***Devoir/Doit/Doivent*** être la suivante:

1. si la couche PHY est en train d'envoyer un message, celui-ci ***Devoir/Doit/Doivent*** être interrompu par l'envoi d'un code K ***EOP*** et le reste du message ***Rejeté(s/ée/ées)***;
2. si le fil CC n'est pas inactif, attendre qu'il le devienne (voir 5.8.6.1);
3. attendre ***tInterFrameGap***;
4. si le fil CC est toujours inactif, envoyer le préambule suivi par les 4 codes K pour le signal ***Hard Reset***;
5. désactiver le canal (c'est-à-dire arrêter d'envoyer et de recevoir), réinitialiser la couche PHY et informer la couche protocole que la couche PHY a été réinitialisée;
6. réactiver le canal lorsque la couche protocole le demande.

La Figure 5-5 donne le format en ligne du signal ***Hard Reset***, qui est un préambule suivi de l'ensemble ordonné ***Hard Reset***.

Figure 5-5 Format en ligne de Hard Reset



Anglais	Français
Preamble (training for receiver)	Préambule (entraînement pour le récepteur)
LEGEND:	LEGENDE:
Preamble provided by the Physical layer, ***not*** encoded with 4b5b	Préambule fourni par la couche physique, ***non*** codé en 4b5b
Provided by the Physical layer, encoded with 4b5b	Fourni par la couche physique, codé en 4b5b

5.6.5 Réinitialisation de câble

Le signal ***Cable Reset*** est un ensemble ordonné d'octets envoyé dans le but d'être reconnu par la couche PHY. L'ensemble ordonné ***Cable Reset*** est défini par la séquence de codes K suivante: ***RST-1***, ***Sync-1***, ***RST-1***, ***Sync-3*** (voir Tableau 5-12).

Tableau 5-12 Ensemble ordonné Cable Reset

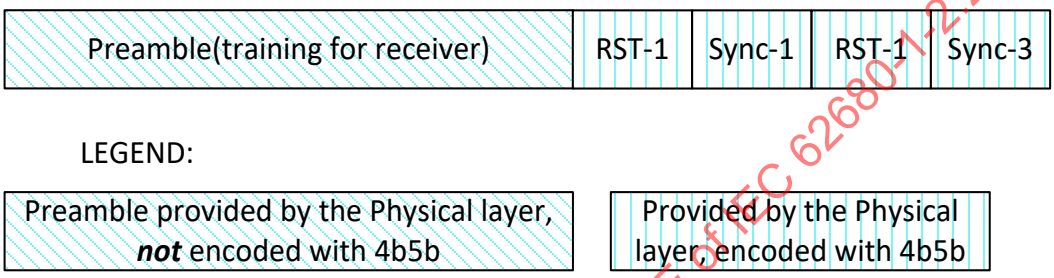
Numéro de code K	Code K dans le tableau de codage
1	*RST-1*

Numéro de code K	Code K dans le tableau de codage
2	*Sync-1*
3	*RST-1*
4	*Sync-3*

Le signal *Cable Reset* ne ***Devoir/Doit/Doivent*** être envoyé que par le port DFP. L'ensemble ordonné *Cable Reset* sert à réinitialiser les fiches de câbles sans qu'il soit nécessaire de réinitialiser matériellement les ports partenaires. L'état de la fiche de câble après le signal *Cable Reset* ***Devoir/Doit/Doivent*** être équivalent à un cycle d'alimentation de la fiche de câble.

La Figure 5-6 donne le format en ligne du signal *Cable Reset*, qui est un préambule suivi de l'ensemble ordonné *Cable Reset*.

Figure 5-6 Format en ligne de Cable Reset



Anglais	Français
Preamble (training for receiver)	Préambule (entraînement pour le récepteur)
LEGEND:	LEGENDE:
Preamble provided by the Physical layer, ***not*** encoded with 4b5b	Préambule fourni par la couche physique, ***non*** codé en 4b5b
Provided by the Physical layer, encoded with 4b5b	Fourni par la couche physique, codé en 4b5b

5.7 Anticollision

La couche PHY ***Devoir/Doit/Doivent*** surveiller la transmission des données sur le canal et ne lancer de transmissions que si le fil CC est inactif. Si la condition de bus inactif est présente, il ***Devoir/Doit/Doivent*** être jugé que commencer une transmission est sûr, sous réserve que les conditions décrites en 5.8.5.4 soient remplies. La condition de bus inactif ***Devoir/Doit/Doivent*** être vérifiée immédiatement avant la transmission. Si la transmission ne peut pas être initiée, le paquet ***Devoir/Doit/Doivent*** être ***Rejeté(s/ée/ées)***. Si le paquet est ***Rejeté(s/ée/ées)*** parce que le fil CC n'est pas inactif, la couche PHY ***Devoir/Doit/Doivent*** signaler à la couche protocole le fait qu'elle a ***Rejeté(s/ée/ées)*** le message dès que le fil CC devient inactif. Voir 5.8.6.1 pour la définition de fil CC inactif.

De plus, la couche PHY ***Devoir/Doit/Doivent*** contrôler la valeur de la résistance Rp afin d'éviter les collisions entre les transmissions par la source et par le destinataire. La source ***Devoir/Doit/Doivent*** définir une valeur de Rp correspondant à un courant de 3 A pour indiquer au destinataire qu'il ***Pouvoir/Peut/Peuvent*** initier une AMS. La source ***Devoir/Doit/Doivent*** définir une valeur de Rp correspondant à une intensité de courant de 1,5 A, ce qui ***Devoir/Doit/Doivent*** indiquer au destinataire qu'il ***Ne doit/doivent pas*** initier une AMS et qu'il ne ***Devoir/Doit/Doivent*** répondre qu'à des messages faisant partie d'une AMS. Voir *[USB Type-C 2.0]* (USB Type-C®) pour plus d'informations sur les valeurs de Rp correspondantes.

Le Tableau 5-13 précise les valeurs de Rp qui ***Devoir/Doit/Doivent*** être utilisées par la source pour contrôler l'initiation d'une AMS par le destinataire.

Tableau 5-13 Valeurs de Rp utilisées pour éviter les collisions

Rp source	Paramètre	Description	Fonctionnement du destinataire	Fonctionnement de la source
1,5 A @ 5 V	*SinkTxNG*	Le destinataire transmet “No Go”,	Le destinataire ne peut pas initier d'AMS. Le destinataire peut uniquement répondre à des messages faisant partie d'une AMS.	La source peut initier une AMS *tSinkTx* après avoir défini Rp sur cette valeur.
3A @ 5 V	*SinkTxOk*	Le destinataire transmet “Ok”,	le destinataire peut initier une AMS.	La source ne peut pas initier d'AMS tant que cette valeur est définie.

Voir également 6.6.16 et 6.11.2.1.

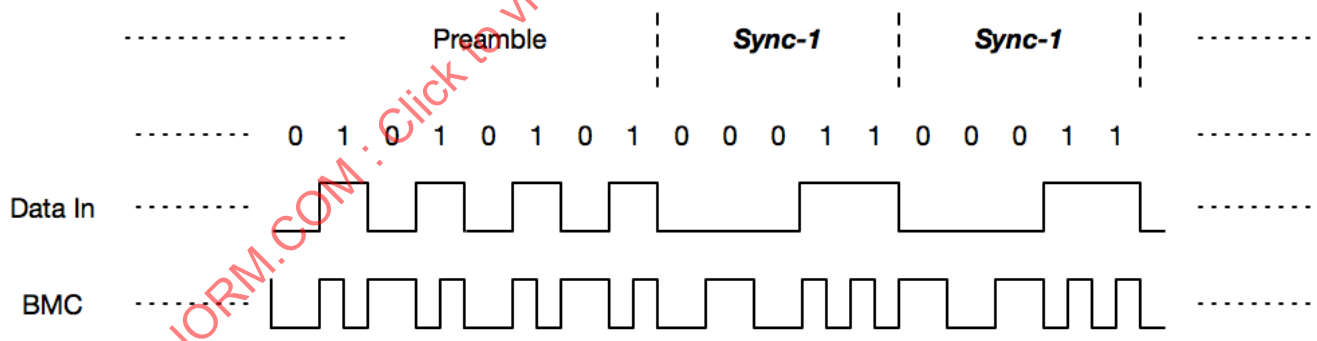
5.8 Schéma de signalisation Biphase Mark Coding (BMC)

Le codage Biphase Mark Coding (BMC) est le schéma de signalisation de la couche physique pour le transport des messages d'alimentation électrique par port USB. Cet encodage admet par hypothèse une connexion dédiée en courant continu, identifiée en tant que fil CC, servant à l'émission de messages PD.

Le codage Biphase Mark Coding est une version du codage Manchester (voir *[IEC 60958-1]*). En BMC, il y a une transition au début de chaque temps binaire (UI) et une deuxième transition au milieu de UI lorsqu'un 1 est transmis. Le codage BMC est équilibré de manière effective en courant continu (chaque 1 est équilibré en courant continu et deux 0 successifs sont équilibrés en courant continu, quel que soit le nombre de 1 qui interviennent). Il a une disparité bornée (limitée à 1 bit sur un paquet arbitraire, c'est-à-dire un niveau de courant continu très faible).

La Figure 5-7 donne un exemple de codage Biphase Mark Coding. Cet exemple montre la transition d'un préambule aux codes K *Sync-1* de l'ensemble ordonné *SOP* au début d'un message. Noter que d'autres codes K peuvent apparaître après le préambule pour un signal tel que *Hard Reset* et *Cable Reset*.

Figure 5-7 Exemple de BMC

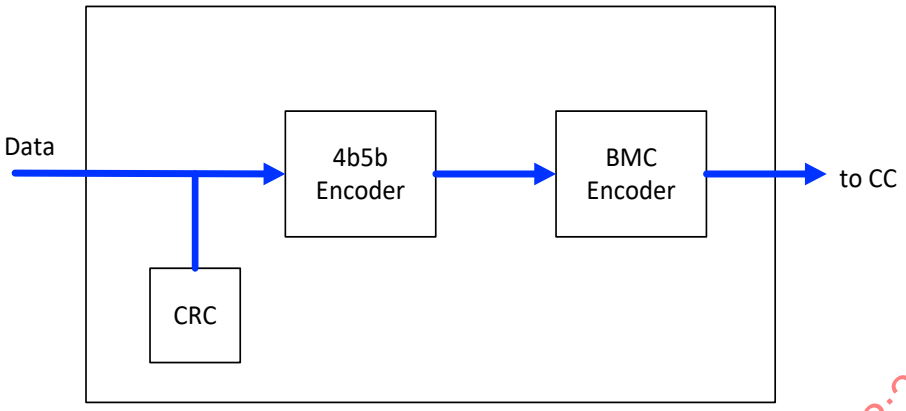


Anglais	Français
Preamble	Préambule
Data In	Données entrantes

5.8.1 Encodage et signalisation

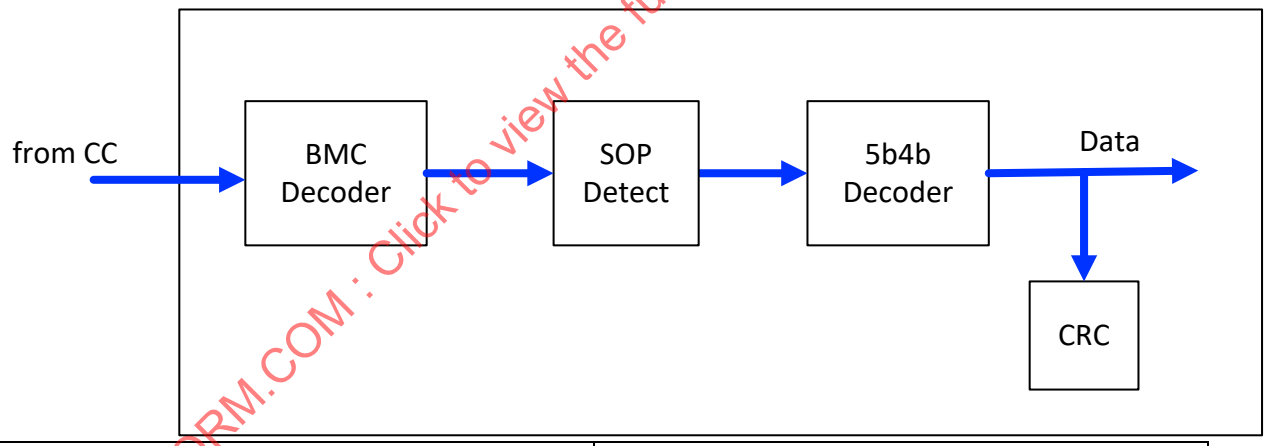
Le codage BMC utilise une signalisation en bande de base couplée à un courant continu sur fil CC. La Figure 5-8 montre un schéma fonctionnel pour un émetteur et la Figure 5-9 montre un schéma fonctionnel pour le récepteur correspondant.

Figure 5-8 Schéma fonctionnel d'émetteur BMC



Anglais	Français
Data	Données
4b5b Encoder	Encodeur 4b5b
BMC Encoder	Encodeur BMC
to CC	vers CC

Figure 5-9 Schéma fonctionnel de récepteur BMC

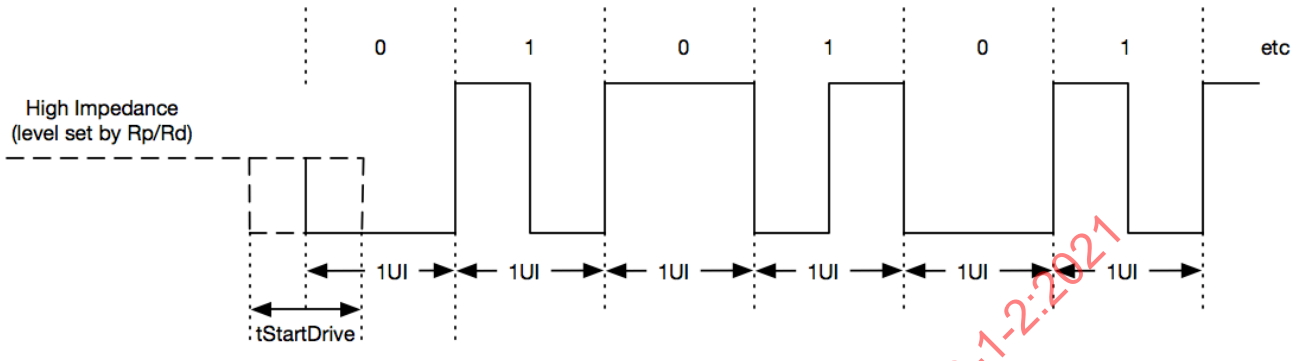


Anglais	Français
From CC	Depuis CC
BMC Decoder	Décodeur BMC
SOP Detect	Détection de SOP
5b4b Decoder	Décodeur 5b4b
Data	Données

Le signal de bande de base d'alimentation USB ***Devoir/Doit/Doivent*** être piloté sur le fil CC, avec un pilote à trois états qui ***Devoir/Doit/Doivent*** provoquer une oscillation ***vSwing*** sur le fil CC. Le pilote à trois états est limité en vitesse de balayage (voir le temps minimal de montée/descente en 5.8.5) afin de limiter le couplage à D+/D- et à d'autres circuits de signalisation dans les câbles qui ont toutes les fonctions USB Type-C® (voir ***[USB Type-C 2.0]***). Cette limitation de vitesse de balayage peut être obtenue par la conception du pilote ou par un filtre RC sur la sortie du pilote.

Lors de l'envoi du préambule, l'émetteur ***Devoir/Doit/Doivent*** commencer par l'émission d'un niveau faible. Le récepteur ***Devoir/Doit/Doivent*** tolérer la perte du premier front. L'émetteur ***Pouvoir/Peut/Peuvent*** faire varier le début du préambule de ***tStartDrive*** minutes (voir Figure 5-10).

Figure 5-10 Début de préambule codé BMC



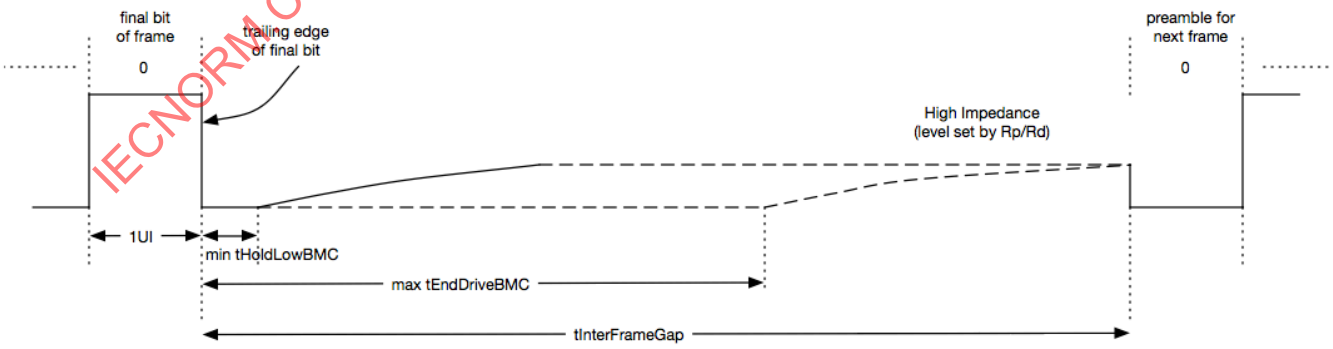
Anglais	Français
High Impedance (level set by Rp/Rd)	Haute impédance (niveau défini par Rp/Rd)

L'émetteur ***Devoir/Doit/Doivent*** terminer le bit final de la trame par un front (le "front arrière") pour s'assurer que le récepteur cale bien son horloge sur le bit final. Si le front arrière a pour conséquence un pilotage du fil CC par l'émetteur au niveau bas (c'est-à-dire que le demi-UI final de la trame est haut), l'émetteur:

1. ***Devoir/Doit/Doivent*** continuer de piloter le fil CC au niveau bas pendant une durée ***tHoldLowBMC***;
2. ***Devoir/Doit/Doivent*** ensuite continuer de piloter le fil CC au niveau bas pendant une durée ***tEndDriveBMC*** mesurée à partir du front arrière du bit final de la trame;
3. ***Devoir/Doit/Doivent*** ensuite laisser le fil CC en haute impédance.

La Figure 5-11 montre la fin d'une trame codée BMC avec encodage d'un 0 pour lequel le bit final de la trame se termine par une transition du niveau haut vers le niveau bas. La Figure 5-12 montre la fin d'une trame codée BMC avec encodage d'un 1 pour lequel le bit final de la trame se termine par une transition du niveau haut vers le niveau bas. Les deux figures montrent également l'exigence de temporisation ***tInterFrameGap*** avant le début de la trame suivante lorsque le port a émis ou reçu la trame précédente (voir 5.8.5.4).

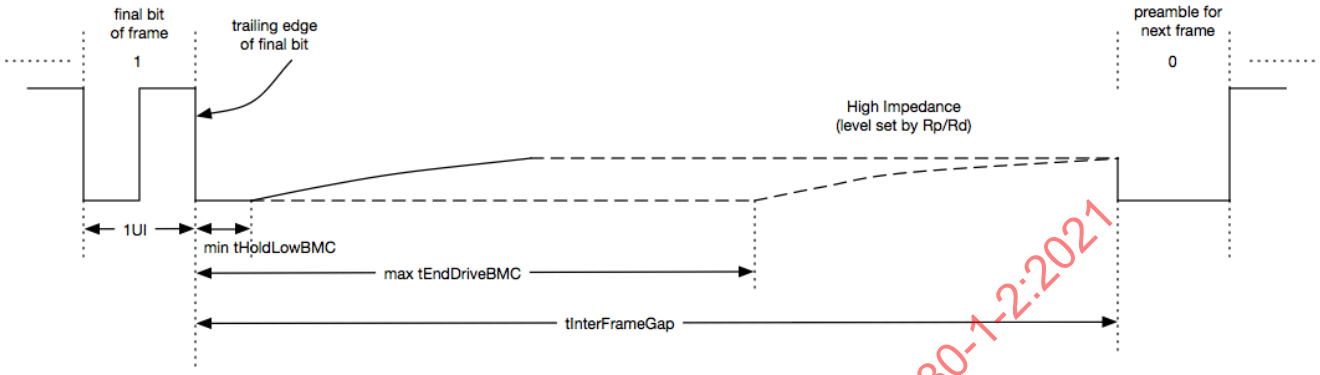
Figure 5-11 Emission ou réception d'une trame codée BMC et terminée par un 0 avec une dernière transition du niveau haut vers le niveau bas



Anglais	Français
Final bit of frame	Bit final de la trame
Trailing edge of final bit	Front arrière du bit final
High Impedance (level set by Rp/Rd)	Haute impédance (niveau défini par Rp/Rd)

Preamble for next frame	Préambule à la trame suivante

Figure 5-12 Emission ou réception d'une trame codée BMC et terminée par un 1 avec une dernière transition du niveau haut vers le niveau bas



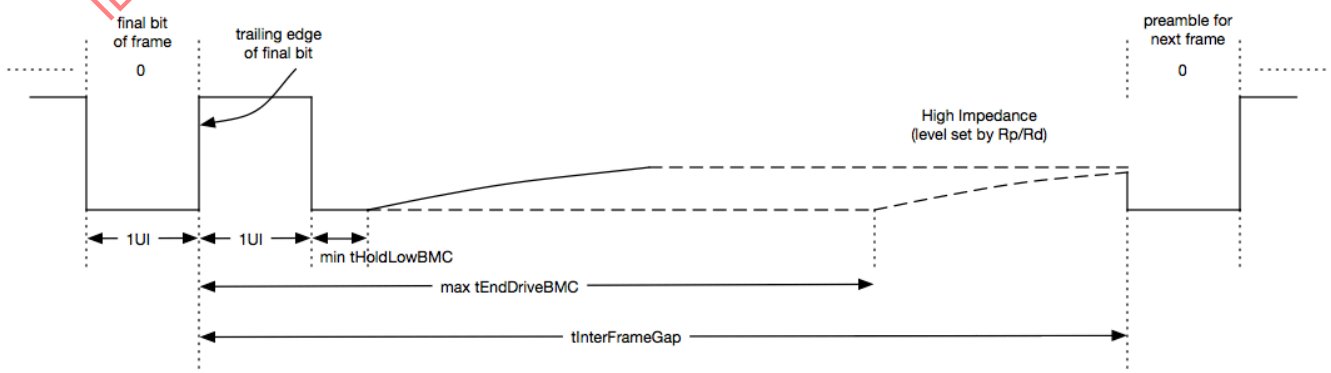
Anglais	Français
Final bit of frame	Bit final de la trame
Trailing edge of final bit	Front arrière du bit final
High Impedance (level set by Rp/Rd)	Haute impédance (niveau défini par Rp/Rd)
Preamble for next frame	Préambule à la trame suivante

Si le front arrière a pour conséquence un pilotage du fil CC par l'émetteur au niveau haut (c'est-à-dire que le demi-UI final de la trame est bas), l'émetteur:

1. ***Devoir/Doit/Doivent*** continuer de piloter le fil CC au niveau haut pendant 1 UI;
2. ***Devoir/Doit/Doivent*** ensuite continuer de piloter le fil CC au niveau bas pendant une durée ***tHoldLowBMC***;
3. ***Devoir/Doit/Doivent*** ensuite continuer de piloter le fil CC au niveau bas pendant une durée ***tEndDriveBMC*** mesurée à partir du front final du bit final de la trame;
4. ***Devoir/Doit/Doivent*** ensuite laisser le fil CC en haute impédance.

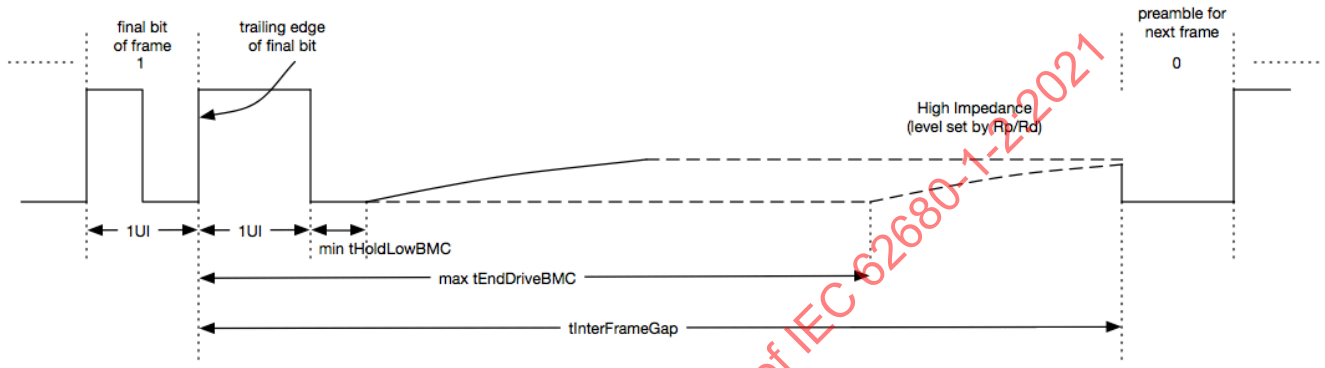
La Figure 5-13 montre la fin d'une trame codée BMC se terminant par un 0 encodé pour lequel le bit final de la trame se termine par une transition du niveau bas vers le niveau haut. La Figure 5-14 montre la fin d'une trame codée BMC se terminant par un 1 encodé pour lequel le bit final de la trame se termine par une transition du niveau bas vers le niveau haut. Les deux figures montrent également l'exigence de temporisation ***tInterFrameGap*** avant le début de la trame suivante lorsque le port a émis ou reçu la trame précédente (voir 5.8.5.4).

Figure 5-13 Emission ou réception d'une trame codée BMC et terminée par un 0 avec une dernière transition du niveau bas vers le niveau haut



Anglais	Français
Final bit of frame	Bit final de la trame
Trailing edge of final bit	Front arrière du bit final
High Impedance (level set by Rp/Rd)	Haute impédance (niveau défini par Rp/Rd)
Preamble for next frame	Préambule à la trame suivante

Figure 5-14 Emission ou réception d'une trame codée BMC et terminée par un 1 avec une dernière transition du niveau bas vers le niveau haut



Anglais	Français
Final bit of frame	Bit final de la trame
Trailing edge of final bit	Front arrière du bit final
High Impedance (level set by Rp/Rd)	Haute impédance (niveau défini par Rp/Rd)
Preamble for next frame	Préambule à la trame suivante

Note: Il n'y a aucune exigence relative au maintien d'une relation de phase dans le temps entre des paquets dos à dos.

5.8.2 Masques d'émission et de réception

5.8.2.1 Masques d'émission

Le signal de transmission ***Ne doit/doivent pas*** violer les masques définis à la Figure 5-15, à la Figure 5-16, au Tableau 5-14 et au Tableau 5-15, à la sortie d'une charge équivalente au modèle de câble et au modèle de charge du récepteur décrits en 5.8.3. Les masques s'appliquent à la plage complète de valeurs Rp/Rd définies dans *[USB Type-C 2.0]*. Note: il n'est pas nécessaire que la mesure de l'émetteur prenne en compte un changement de décalage de signal dû à la différence de masse lorsqu'un courant circule dans le câble.

Le signal transmis ***Devoir/Doit/Doivent*** avoir un temps de montée inférieur ou égal à ***tRise***. Le signal transmis ***Devoir/Doit/Doivent*** avoir un temps de descente inférieur ou égal à ***tFall***. Les limites maximales des temps de montée et de descente sont imposées par les masques Tx internes.

Figure 5-15 Masque BMC Tx "1"

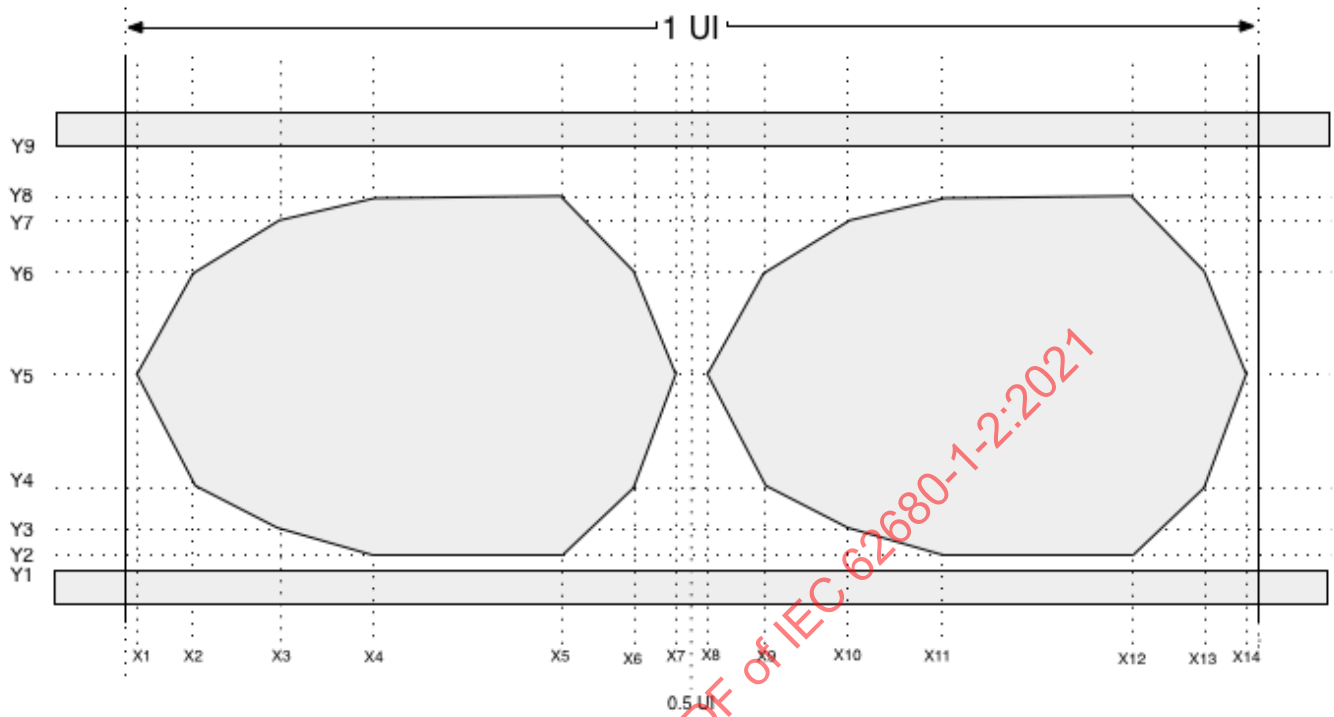


Figure 5-16 Masque BMC Tx "0"

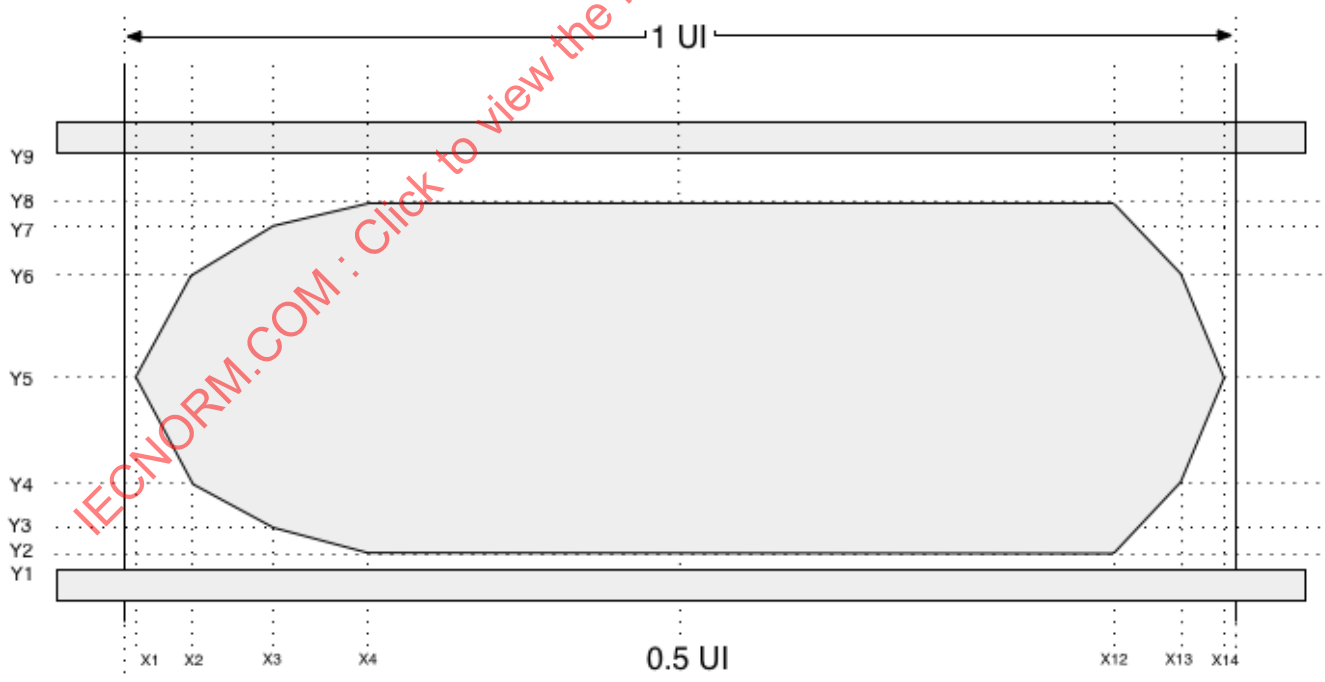


Tableau 5-14 Définition de masque BMC Tx, valeurs de X

Nom	Description	Valeur	Unités
X1Tx	Front gauche du masque	0,015	UI
X2Tx	voir figure	0,07	UI

Nom	Description	Valeur	Unités
X3Tx	voir figure	0,15	UI
X4Tx	voir figure	0,25	UI
X5Tx	voir figure	0,35	UI
X6Tx	voir figure	0,43	UI
X7Tx	voir figure	0,485	UI
X8Tx	voir figure	0,515	UI
X9Tx	voir figure	0,57	UI
X10Tx	voir figure	0,65	UI
X11Tx	voir figure	0,75	UI
X12Tx	voir figure	0,85	UI
X13Tx	voir figure	0,93	UI
X14Tx	Front droit du masque	0,985	UI

Tableau 5-15 Définition de masque BMC Tx, valeurs de Y

Nom	Description	Valeur	Unités
Y1Tx	Borne inférieure du masque externe	-0,075	V
Y2Tx	Borne inférieure du masque interne	0,075	V
Y3Tx	voir figure	0,15	V
Y4Tx	voir figure	0,325	V
Y5Tx	Point milieu vertical du masque interne	0,5625	V
Y6Tx	voir figure	0,8	V
Y7Tx	voir figure	0,975	V
Y8Tx	voir figure	1,04	V
Y9Tx	Borne supérieure du masque externe	1,2	V

5.8.2.2 Masques de réception

Une source qui utilise le schéma de signalisation BMC ***Devoir/Doit/Doivent*** être capable de recevoir un signal conforme au masque lorsqu'elle fournit une alimentation électrique, comme défini à la Figure 5-17, à la Figure 5-18 et au Tableau 5-16. Le masque Rx de la source est borné par le balayage d'un signal conforme au masque Tx, en ajoutant ***vNoiseActive*** entre le passage de l'alimentation au point neutre et les décalages de la source.

Un consommateur qui utilise le schéma de signalisation BMC ***Devoir/Doit/Doivent*** être capable de recevoir un signal conforme au masque lorsqu'il reçoit une alimentation électrique, comme défini à la Figure 5-21, à la Figure 5-22 et au Tableau 5-16. Le masque Rx du consommateur est borné par le balayage d'un signal conforme au masque Tx, en ajoutant ***vNoiseActive*** entre le passage de l'alimentation au point neutre et les décalages du consommateur.

Tout produit qui utilise le schéma de signalisation BMC ***Devoir/Doit/Doivent*** être capable de recevoir un signal conforme au masque pendant le passage de l'alimentation au point neutre, comme défini à la Figure 5-19, à la Figure 5-20 et au Tableau 5-16.

Les dispositifs d'alimentation double fonction ***Devoir/Doit/Doivent*** satisfaire aux exigences du récepteur pour une source lorsqu'ils fournissent une alimentation électrique pendant toute transmission faisant appel au schéma de signalisation BMC ou pour un destinataire lorsqu'ils utilisent une alimentation électrique pendant toute transmission faisant appel au schéma de signalisation BMC.

Les fiches de câbles ***Devoir/Doit/Doivent*** satisfaire aux exigences du récepteur à la fois pour une source et pour un destinataire pendant toute transmission faisant appel au schéma de signalisation BMC.

Les paramètres utilisés dans les masques sont spécifiés pour convenir aux mises en œuvre de récepteur déclenché sur un front ou assurant un suréchantillonnage.

Les masques sont définis séparément pour "1" et pour "0" car le code BMC impose une transition au point milieu de l'intervalle unitaire lorsqu'un "1" est transmis.

Les masques Rx sont définis pour borner le bruit Rx après application du filtre de limitation de la largeur de bande Rx avec la constante de temps ***tRxFilter***.

Les limites du masque extérieur Rx, ***Y1Rx*** et ***Y5Rx***, sont spécifiées conformément à ***vSwing*** max et prennent en compte la moitié de ***vNoiseActive*** provenant du couplage de bruit du câble et du décalage de signal ***vIRDropGNDC*** dû à la différence de masse lorsqu'un courant circule dans le câble.

La dimension verticale du masque interne de Rx, ***Y4Rx*** - ***Y2Rx***, est obtenue pour le passage au point neutre de l'alimentation en réduisant la dimension verticale du masque Tx interne, ***Y7Tx*** - ***Y3Tx***, au temps ***X3Tx***, de ***vNoiseActive*** pour tenir compte du couplage de bruit du câble. Le signal reçu se compose d'une forme d'onde conforme au masque Tx, plus ***vNoiseActive***.

La dimension verticale du masque interne de Rx pour fournir l'alimentation électrique est obtenue en réduisant la dimension verticale du masque Tx interne de ***vNoiseActive*** et de ***vIRDropGNDC*** pour tenir compte du couplage de bruit du câble et du décalage en courant continu du signal. Le signal reçu se compose d'une forme d'onde conforme au masque Tx, plus la valeur maximale de ***vNoiseActive***, plus ***vIRDropGNDC***, la valeur ***vIRDropGNDC*** oscillant entre les valeurs minimale et maximale, comme le permet la présente spécification.

La dimension verticale du masque interne de Rx pour utiliser l'alimentation électrique est obtenue en réduisant la dimension verticale du masque Tx interne de ***vNoiseActive*** max et de ***vIRDropGNDC*** max pour tenir compte du couplage de bruit du câble et du décalage en courant continu du signal. Le signal reçu se compose d'une forme d'onde conforme au masque Tx, plus la valeur maximale de ***vNoiseActive***, plus ***vIRDropGNDC***, la valeur ***vIRDropGNDC*** oscillant entre les valeurs minimale et maximale, comme le permet la présente spécification.

L'axe central du masque Rx interne, ***Y3Rx***, est à la moitié de la tension nominale ***vSwing*** pour le passage au point neutre de l'alimentation. Il est décalé vers le haut de la moitié de ***vIRDropGNDC*** max pour fournir l'alimentation électrique, et il est décalé vers le bas de la moitié de ***vIRDropGNDC*** max pour utiliser l'alimentation électrique.

La sensibilité du récepteur ***Devoir/Doit/Doivent*** être définie de manière que le récepteur ne traite pas le bruit sur une trajectoire de signal non piloté comme un signal entrant. Les amplitudes de signal inférieures à ***vNoiseIdle*** max ***Devoir/Doit/Doivent*** être traitées comme du bruit lorsque le codage BMC est inactif.

Figure 5-17 Masque BMC Rx "1" lorsque l'alimentation électrique est fournie

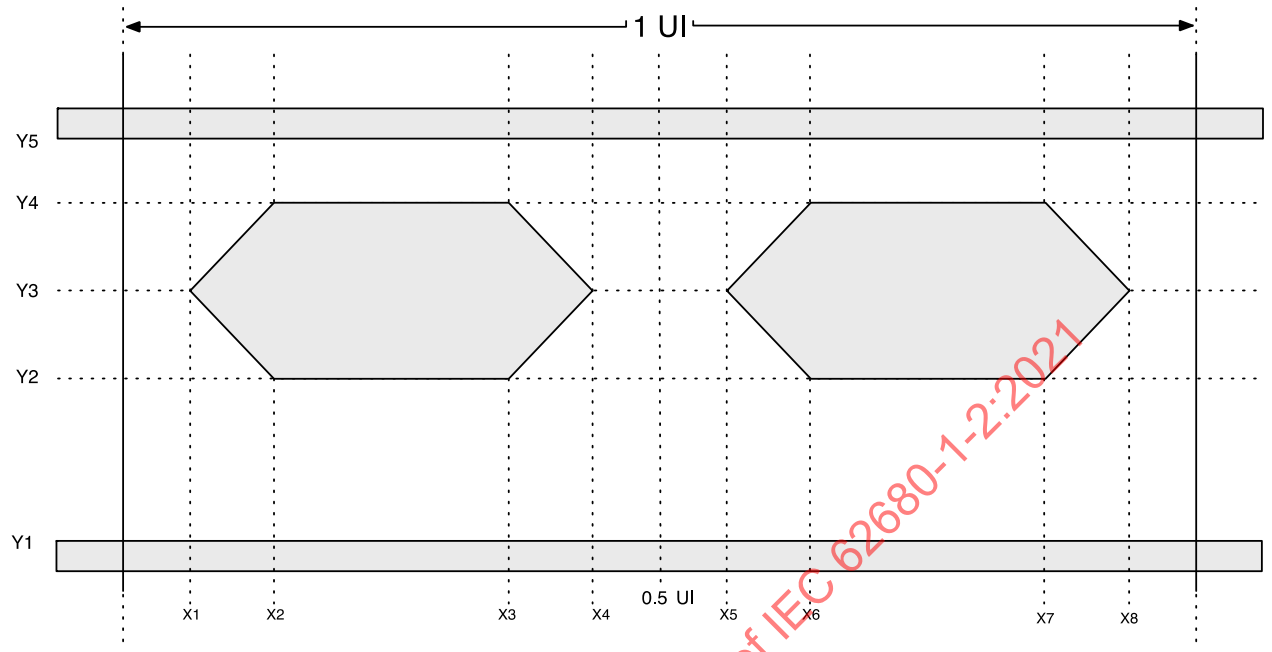


Figure 5-18 Masque BMC Rx "0" lorsque l'alimentation électrique est fournie

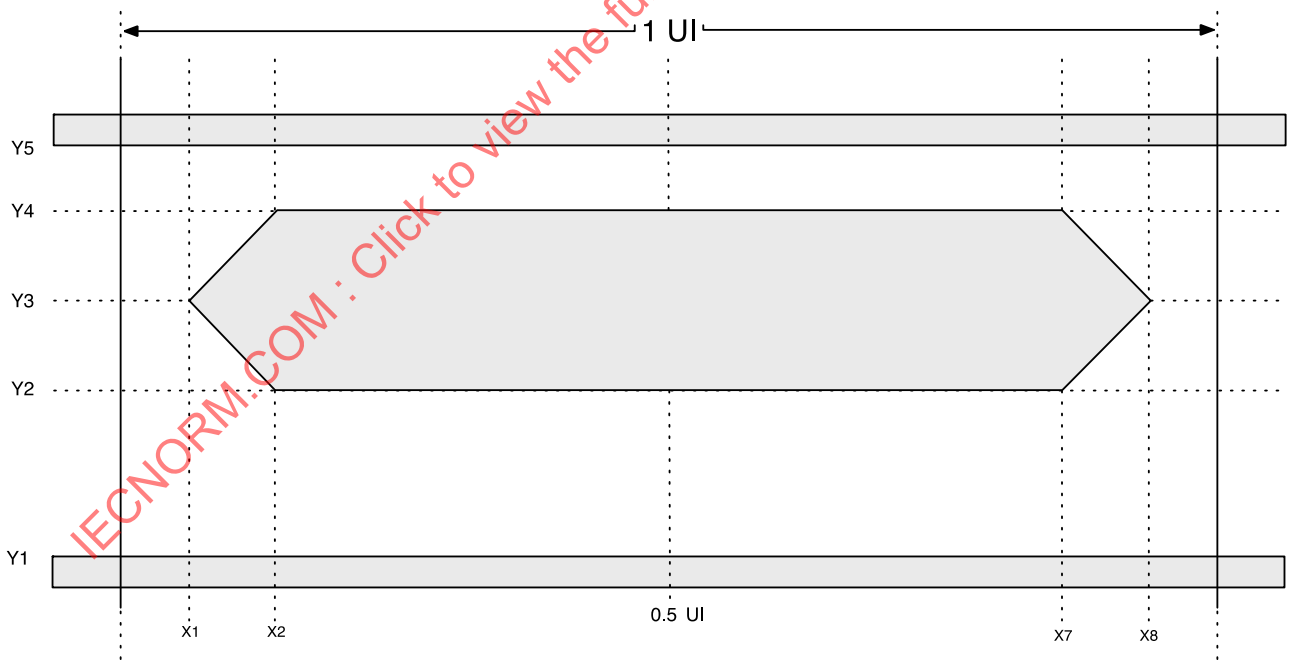


Figure 5-19 Masque BMC Rx "1" lorsque l'alimentation électrique est au point neutre

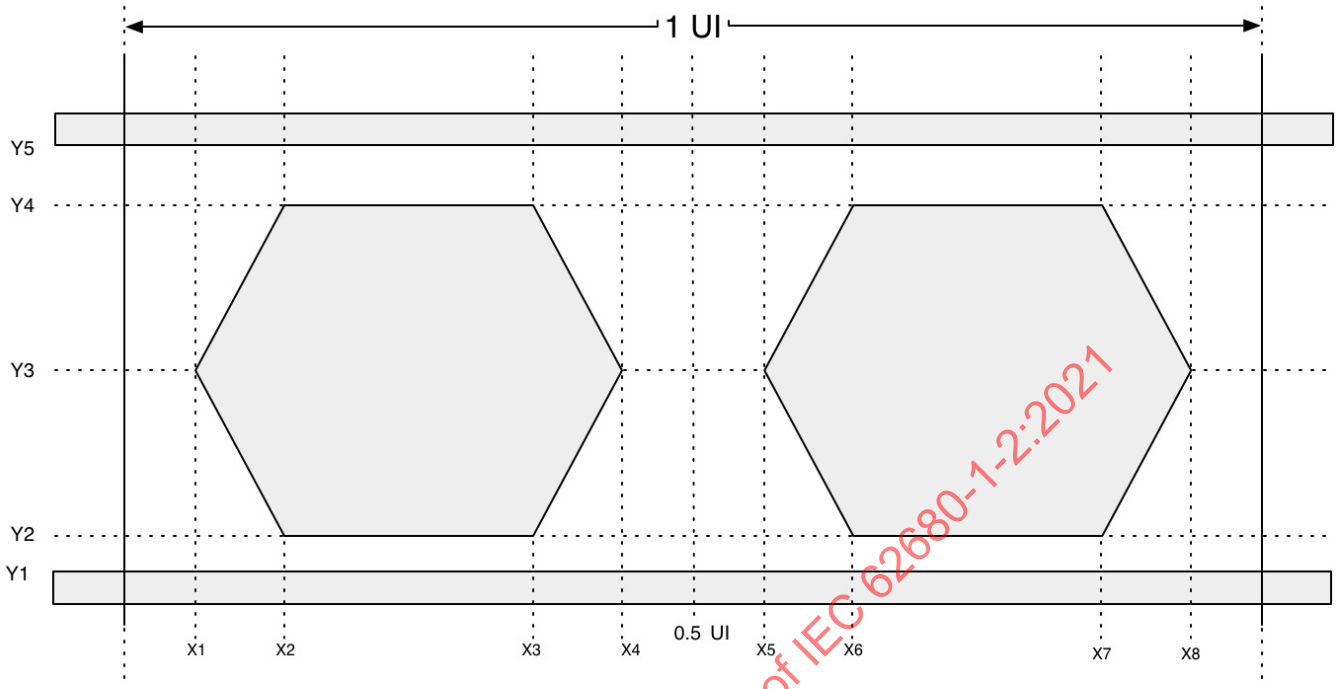


Figure 5-20 Masque BMC Rx "0" lorsque l'alimentation électrique est au point neutre

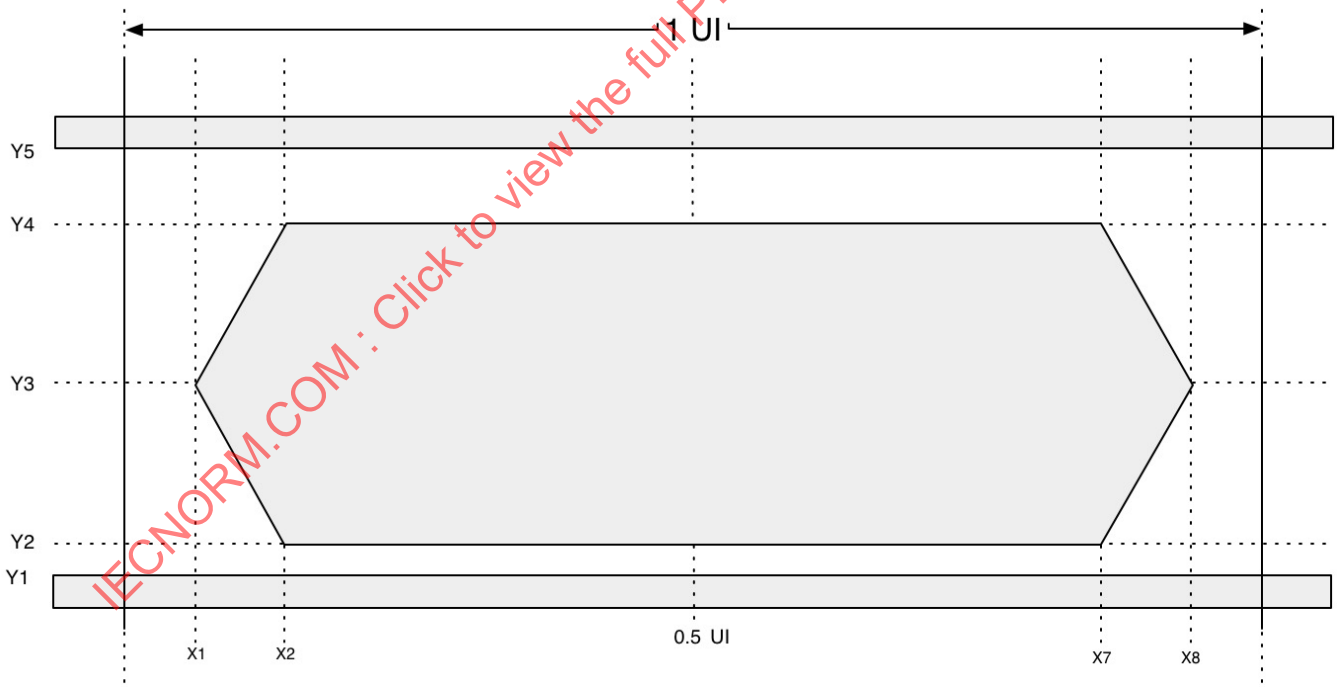


Figure 5-21 Masque BMC Rx "1" lorsque l'alimentation électrique est utilisée

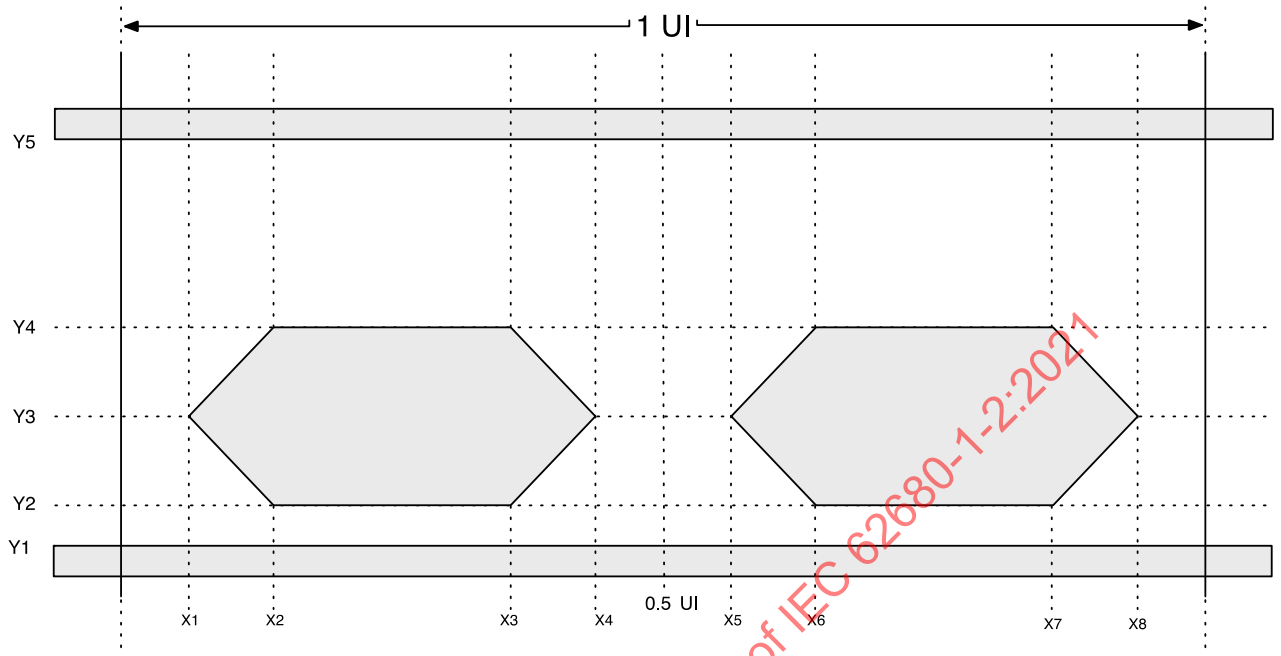


Figure 5-22 Masque BMC Rx "0" lorsque l'alimentation électrique est utilisée

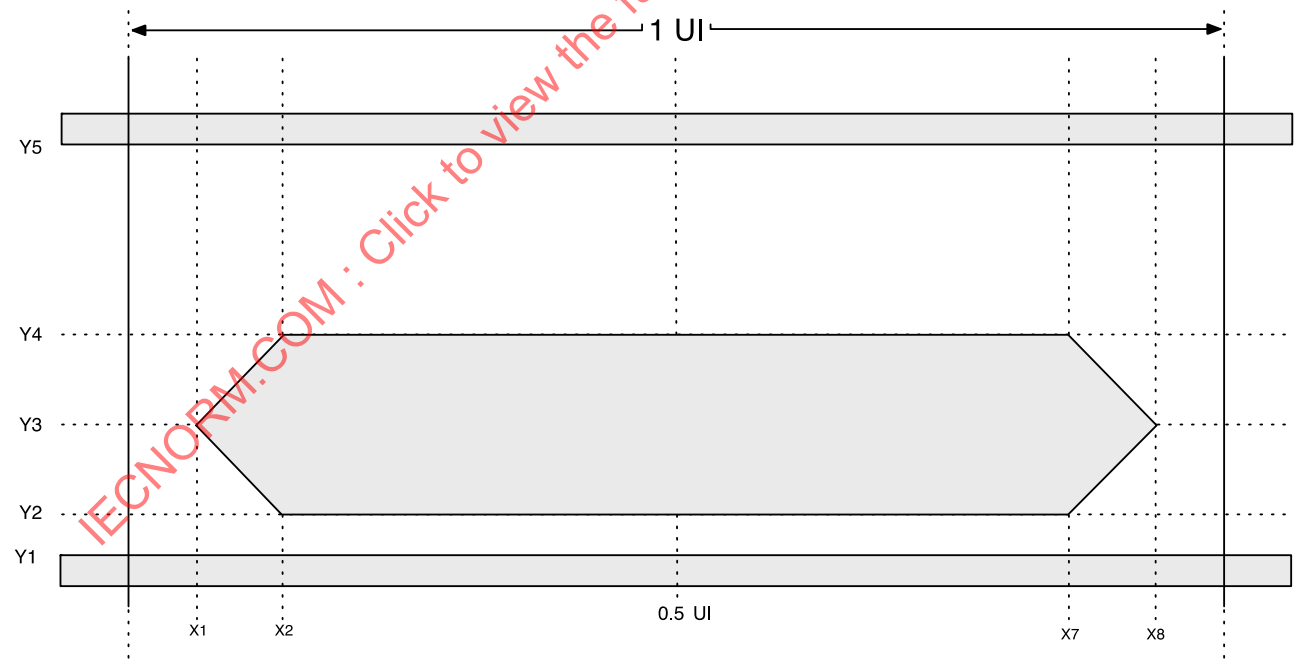


Tableau 5-16 Définition de masque BMC Rx

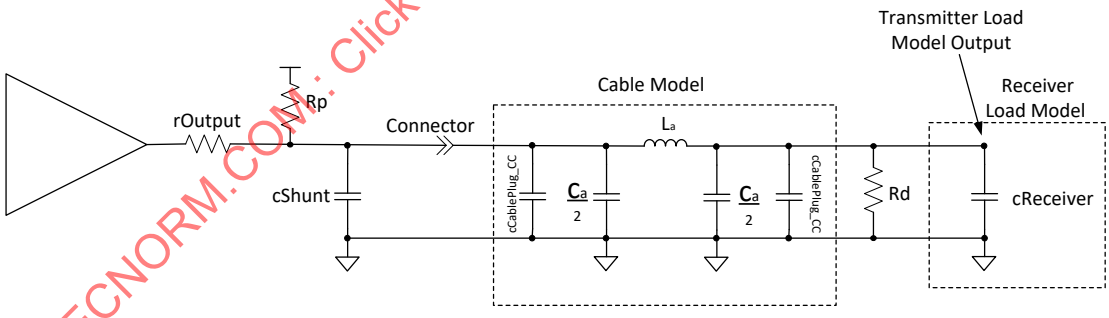
Nom	Description	Valeur	Unités
X1Rx	Front gauche du masque	0,07	UI
X2Rx	Front supérieur du masque	0,15	UI
X3Rx	Voir figure	0,35	UI

Nom	Description	Valeur	Unités
X4Rx	Voir figure	0,43	UI
X5Rx	Voir figure	0,57	UI
X6Rx	Voir figure	0,65	UI
X7Rx	Voir figure	0,85	UI
X8Rx	Voir figure	0,93	UI
Y1Rx	Borne inférieure du masque externe	-0,3325	V
Y2Rx	Borne inférieure du masque interne	***Y3Rx*** – 0,205 lorsque l'alimentation électrique est fournie[1] ou lorsque l'alimentation électrique est utilisée[1] ***Y3Rx*** – 0,33 lorsque l'alimentation électrique est au point neutre[1]	V
Y3Rx	Axe central du masque interne	0,6875 alimentation électrique fournie[1] 0,5625 alimentation électrique au point neutre[1] 0,4375 alimentation électrique utilisée[1]	V
Y4Rx	Borne supérieure du masque interne	***Y3Rx*** + 0,205 lorsque l'alimentation électrique est fournie[1] ou lorsque l'alimentation électrique est utilisée[1] ***Y3Rx*** + 0,33 lorsque l'alimentation électrique est au point neutre[1]	V
Y5Rx	Borne supérieure du masque externe	1,5325	V
Note 1: La position de l'axe central du masque interne dépend du fait que le récepteur est en train de fournir ou d'utiliser l'alimentation électrique ou que l'alimentation électrique est au point neutre (voir plus haut dans la présente section).			

5.8.3 Modèle de charge d'émetteur

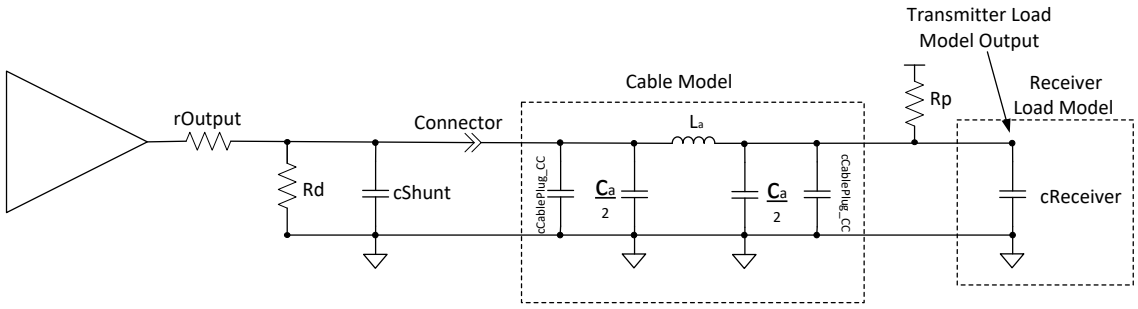
Le modèle de charge d'émetteur ***Devoir/Doit/Doivent*** être équivalent au circuit décrit à la Figure 5-23 pour une source et à la Figure 5-24 pour un destinataire. Il est formé de la concaténation d'un modèle de charge de câble et d'un modèle de charge de récepteur. Voir ***[USB Type-C 2.0]*** pour plus d'informations sur les résistances Rp et Rd. Noter que les paramètres *zCable_CC*, *tCableDelay_CC* et *cCablePlug_CC* sont définis dans ***[USB Type-C 2.0]***.

Figure 5-23 Modèle de charge d'émetteur pour BMC Tx à partir d'une source



Anglais	Français
Connector	Connecteur
Cable Model	Modèle de câble
Transmitter Load Model Output	Sortie du modèle de charge d'émetteur
Receiver Load Model	Modèle de charge de récepteur

Figure 5-24 Modèle de charge d'émetteur pour BMC Tx à partir d'un destinataire



Anglais	Français
Connector	Connecteur
Cable Model	Modèle de câble
Transmitter Load Model Output	Sortie du modèle de charge d'émetteur
Receiver Load Model	Modèle de charge de récepteur

Les composants du système d'émetteur *rOutput* et *cShunt* sont représentés à titre d'information. Ils ne font pas partie du modèle de charge d'émetteur. Pour une description de la conception du système d'émetteur, voir 5.8.5.

La valeur de l'inductance modélisée du câble, La, (en nH) ***Devoir/Doit/Doivent*** être calculée à partir de la formule suivante:

$$La = tCableDelay\_CC_{max} * zCable\_CC_{min}$$

tCableDelay_CC étant le retard de propagation dans le câble du signal modélisé et *zCable_CC* l'impédance modélisée du câble.

L'inductance modélisée du câble est de 640 nH pour un câble avec $zCable\_CC_{min}$ = 32 Ω et $tCableDelay\_CC_{max}$ = 20 ns.

La valeur de la capacité électrique modélisée du câble, Ca, (en pF) ***Devoir/Doit/Doivent*** être calculée à partir de la formule suivante:

$$Ca = \frac{tCableDelay\_CC_{max}}{zCable\_CC_{min}}$$

La capacité électrique modélisée du câble est Ca = 625 pF pour un câble avec $zCable\_CC_{min}$ = 32 Ω et $tCableDelay\_CC_{max}$ = 20 ns. Par conséquent, Ca/2 = 312,5 pF.

cCablePlug_CC modélise la capacité électrique de la fiche à chaque extrémité du câble. ***cReceiver*** modélise la capacité électrique du récepteur. Les valeurs maximales ***Devoir/Doit/Doivent*** être utilisées dans chacun des cas.

Note: Le modèle de charge d'émetteur admet par hypothèse qu'il n'y a aucun autre courant de retour sur la liaison à la masse.

5.8.4 Spécifications BMC communes

La présente section définit les exigences communes au récepteur et à l'émetteur.

5.8.4.1 Paramètres BMC communs

Les exigences électriques spécifiées dans le Tableau 5-17 ***Devoir/Doit/Doivent*** être appliquées à la fois à l'émetteur et au récepteur.

Tableau 5-17Exigences normatives BMC communes

Nom	Description	Min	Nom	Max	Unités	Commentaire
fBitRate	Débit binaire	270	300	330	Kbit/s	
tUnitInterval[1]	Intervalle unitaire	3,03		3,70	µs	1/***fBitRate***
Note 1: ***tUnitInterval*** définit le temps d'émission d'un bit de données non codées, et non les durées les plus courtes de niveau haut ou bas sur le fil après encodage par BMC. Une cellule unique de bits de données a une durée de 1 UI, alors qu'une cellule de bits de données de valeur 1 contient une transition 01 ou 10 placée au centre, en plus de la transition en début de cellule.						

5.8.5 Spécifications pour l'émetteur BMC

L'émetteur ***Devoir/Doit/Doivent*** satisfaire aux spécifications définies dans le Tableau 5-18.

Tableau 5-18 Exigences normatives pour l'émetteur BMC

Nom	Description	Min	Nom	Max	Unités	Commentaire
pBitRate	Différence maximale entre le débit binaire pendant la partie du paquet qui suit le préambule et le débit binaire de référence.			0,25	%	Le débit binaire de référence est le débit binaire moyen des 32 derniers bits du préambule.
rFRSwapTx	Permutation rapide des rôles Transmission de demande Résistance du pilote (à l'exclusion de la résistance du câble)			5	Ω	Résistance de pilote maximale pour une demande de permutation rapide des rôles de l'émetteur. L'hypothèse retenue est celle du cas le plus défavorable de résistance de câble de 15 Ω, comme défini dans ***[USB Type-C 2.0]***. Note: Sur la base de cette valeur, la résistance combinée du pilote et du câble d'un émetteur de demande de permutation rapide des rôles maximale est de 20 Ω.
tEndDriveBMC	Temps nécessaire pour cesser de piloter la ligne après la fin du dernier bit de la trame.			23	µs	La valeur minimale est limitée par ***tHoldLowBMC***.
tFall	Temps de descente	300			ns	Points d'amplitude à 10 % et 90 %, la valeur minimale étant en condition non chargée.
tHoldLowBMC	Temps nécessaire pour cesser de piloter la ligne après la transition finale de niveau haut à niveau bas.	1			µs	La valeur maximale est limitée par ***tEndDriveBMC***.
tInterFrameGap	Temps décompté à partir de la fin du dernier bit d'une trame jusqu'au début du premier bit du préambule suivant.	25			µs	
tFRSwapTx	Durée de transmission d'une demande de permutation rapide des rôles	60		120	µs	Une demande de permutation rapide des rôles est indiquée de la source initiale vers le destinataire initial en pilotant le fil CC au niveau bas pendant cette durée.
tRise	Temps de montée	300			ns	Points d'amplitude à 10 % et 90 %, la valeur minimale

Nom	Description	Min	Nom	Max	Unités	Commentaire
						étant en condition non chargée.
tStartDrive	Temps avant le début du premier bit du préambule pendant lequel l'émetteur ***Devoir/Doit/Doivent*** commencer à piloter la ligne.	-1		1	µs	
vSwing	Oscillation de tension	1,05	1,125	1,2	V	S'applique dans les conditions non chargée et chargée spécifiées en 5.8.3.
zDriver	Impédance de sortie de l'émetteur	33		75	Ω	Impédance de sortie de la source à la fréquence de Nyquist de la faible vitesse *[USB 2.0]* (750 kHz) pendant que la source pilote la ligne CC.

5.8.5.1 Capacité électrique en l'absence d'émission

cReceiver est la capacité électrique qu'un port DFP ou UFP ***Devoir/Doit/Doivent*** présenter sur la ligne CC lorsque le récepteur du port DFP ou UFP n'émet pas sur la ligne. L'émetteur ***Pouvoir/Peut/Peuvent*** avoir une capacité électrique supérieure à *cReceiver* pendant qu'il pilote la ligne CC, mais il ***Devoir/Doit/Doivent*** satisfaire aux exigences relatives au masque de forme d'onde. Lorsque l'émission est terminée, l'émetteur ***Devoir/Doit/Doivent*** déconnecter la capacité électrique dépassant *cReceiver* du fil CC dans un délai *tInterFrameGap*.

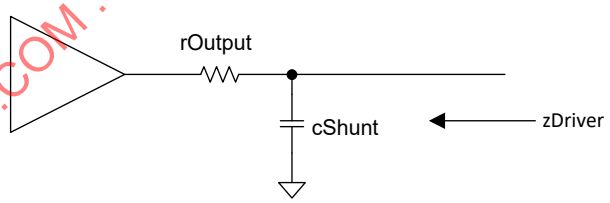
5.8.5.2 Impédance de sortie de la source

L'impédance de sortie de la source *zDriver* est déterminée par la résistance du pilote et la capacité électrique parallèle de la source. Il s'agit donc d'un terme dépendant de la fréquence. *zDriver* a un impact sur la pénétration du bruit dans le câble. Elle est spécifiée de sorte que le bruit au récepteur soit limité.

zDriver est définie par l'équation suivante:

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

Figure 5-25 Schéma de l'émetteur représentant zDriver



cShunt ***Ne doit/doivent pas*** entraîner de violation de *cReceiver* en l'absence d'émission.

5.8.5.3 Dérive du débit binaire

Les limites de dérive de *fBitRate* sont définies afin d'aider les mises en œuvre de récepteur de faible complexité.

fBitRate est la réciproque de la durée de bit moyenne des 32 bits précédents pour une portion de paquet donnée. La variation de *fBitRate* au cours d'un paquet ***Devoir/Doit/Doivent*** être inférieure à *pBitRate*. Le débit binaire de référence (refBitRate) est le *fBitRate* moyen des 32 derniers bits du préambule. *fBitRate* sur l'ensemble du paquet, y compris l'*EOP*, ***Devoir/Doit/Doivent*** être compris dans *pBitRate* de refBitRate. *pBitRate* est exprimé sous forme de pourcentage:

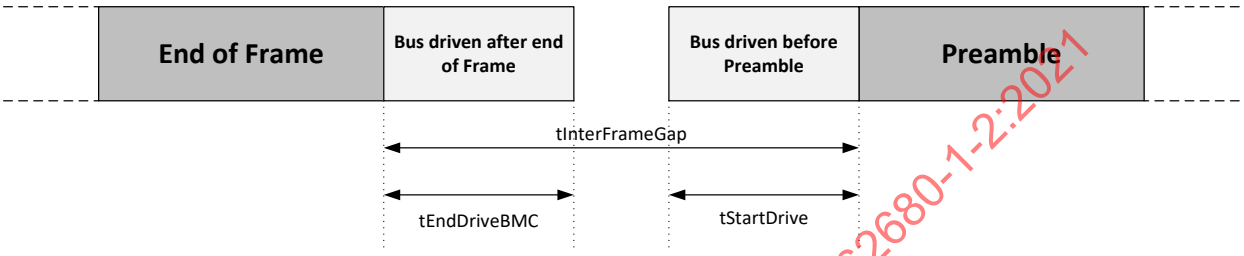
pBitRate = | *fBitRate* – refBitRate | / refBitRate x 100 %

L'émetteur ***Devoir/Doit/Doivent*** avoir le même ***pBitRate*** pour tous les types de paquets. Les signaux ***Mode porteur BIST*** et Bit Stream sont des signaux continus sans charge utile. Pendant la vérification de ***pBitRate***, tout ensemble de 1044 bits (***SOP*** de 20 bits suivi de 1024 bits PRBS) au sein d'un signal continu ***Pouvoir/Peut/Peuvent*** être vu comme faisant partie du paquet suivant le préambule, et les 32 bits précédents sont vus comme les 32 derniers bits du préambule servant à calculer refBitRate.

5.8.5.4 Ecart inter-trame

La Figure 5-26 représente le chronogramme des écarts inter-trames.

Figure 5-26 Chronogramme des écarts inter-trames



Anglais	Français
End of Frame	Fin de trame
Bus driven after end of Frame	Pilotage du bus après la fin de trame
Bus driven before Preamble	Pilotage du bus avant le préambule
Preamble	Préambule

L'émetteur ***Devoir/Doit/Doivent*** piloter le bus pendant au plus ***tEndDriveBMC*** après la transmission du dernier bit de la trame.

Avant de commencer à émettre le préambule de la trame suivante, l'émetteur de la trame suivante ***Devoir/Doit/Doivent*** s'assurer qu'il attend bien ***tInterFrameGap*** après:

1. avoir transmis la trame précédente, par exemple en envoyant le message suivant dans une AMS immédiatement après avoir envoyé un message ***GoodCRC***; ou
2. avoir reçu la trame précédente, par exemple pour répondre à un message reçu par un message ***GoodCRC***; ou
3. avoir observé une condition d'inactivité sur le fil CC (voir 5.7). Dans ce cas, le port qui attend pour initier une AMS observe l'état d'inactivité (voir 5.8.6.1), puis attend ***tInterFrameGap*** avant d'émettre la trame. Voir aussi 5.7 pour plus d'informations sur le moment où une AMS peut être initiée.

Note: L'émetteur est également tenu de vérifier une condition de bus inactif immédiatement avant de commencer à émettre la trame suivante (voir 5.8.6.1).

L'émetteur de la trame suivante ***Pouvoir/Peut/Peuvent*** faire varier le début du préambule de ***tStartDrive*** (voir 5.8.1).

Voir aussi 5.8.1 pour les figures représentant les chronogrammes relatifs à la transmission, à la réception et à l'observation de l'inactivité en ce qui concerne les trames.

5.8.5.5 Mise en court-circuit de la sortie d'émetteur

Un émetteur dans un port ou une fiche de câble ***Devoir/Doit/Doivent*** tolérer que sa sortie soit mise en court-circuit à la masse pendant au maximum ***tFRSwapTx***. Cela est dû à la possibilité qu'une permutation rapide des rôles soit signalée alors que l'émetteur est en train de transmettre (voir 5.8.5.6).

5.8.5.6 Transmission de permutation rapide des rôles

Le processus de permutation rapide des rôles est prévu pour être utilisé par un HUB PDUSB qui dispose actuellement d'une alimentation secteur externe et qui fournit une alimentation à la fois par ses ports en

aval vers des dispositifs USB et par ses ports en amont vers un hôte USB tel qu'un ordinateur portable. Au débranchement de l'alimentation secteur externe, la permutation rapide des rôles permet de maintenir une alimentation V_{BUS} en permettant à l'hôte USB d'appliquer une tension ***vSafe5V*** lorsqu'il constate que V_{BUS} descend au-dessous de ***vSafe5V***, après avoir détecté un signal de permutation rapide des rôles. L'AMS de permutation rapide des rôles sert ensuite à affecter correctement les rôles de source/destinataire et à configurer les résistances Rp/Rd (voir 8.3.2.7).

La source initiale ***Devoir/Doit/Doivent*** signaler une demande de permutation rapide des rôles en pilotant le fil CC à la masse avec une résistance inférieure à ***rFRSwapTx*** pendant ***tFRSwapTx***. La source initiale ***Devoir/Doit/Doivent*** uniquement signaler une permutation rapide des rôles lorsqu'elle a un contrat explicite. La source initiale ***Pouvoir/Peut/Peuvent*** signaler une permutation rapide des rôles même si ses capacités de destinataire n'ont pas encore été interrogées par le destinataire initial. Lors de la transmission du signal de permutation rapide des rôles, tous les messages en suspens ***Devoir/Doit/Doivent*** être ***Rejeté(s/ée/ées)*** (voir 6.11.2.2.1).

Le signal de permutation rapide des rôles ***Pouvoir/Peut/Peuvent*** être prioritaire par rapport à toute transmission active.

La réponse du destinataire initial au signal de permutation rapide des rôles consistant à envoyer un message ***FR_Swap***, la source initiale ***Devoir/Doit/Doivent*** veiller à ce que Rp soit définie sur ***SinkTxOk*** dès que le signal de permutation rapide des rôles est complet.

5.8.6 Spécifications BMC pour le récepteur

Le récepteur ***Devoir/Doit/Doivent*** satisfaire aux spécifications définies dans le Tableau 5-19.

Tableau 5-19 Exigences normatives BMC pour le récepteur

Nom	Description	Min	Nom	Max	Unités	Commentaire
cReceiver	Capacité électrique du récepteur CC	200		600	pF	Le système DFP ou UFP ***Devoir/Doit/Doivent*** avoir une capacité électrique dans cette plage en l'absence d'émission sur la ligne.
nBER	Taux d'erreur de bit, S/N = 25 dB			10^{-6}		
nTransitionCount	Transitions pour détection de signal	3				Nombre de transitions à détecter pour déclarer un bus non inactif.
tFRSwapRx	Durée de détection d'une demande de permutation rapide des rôles	30		50	µs	Une demande de permutation rapide des rôles entraîne la détection par le récepteur d'un signal bas pendant au moins cette durée.
tRxFilter	Filtre de limitation de largeur de bande Rx (numérique ou analogique)	100			ns	Constante de temps d'un filtre unipolaire pour limiter la pénétration des bruits à large bande[1].
tTransitionWindow	Fenêtre temporelle pour la détection de non inactivité.	12		20	µs	
vFRSwapCableTx	Seuil de détection de la tension de demande de permutation rapide des rôles	490	520	550	mV	La demande de permutation rapide des rôles doit être inférieure à ce seuil de tension pour être détectée.
vIRDropGNDC	Chute de tension ohmique d'un câble par rapport à la masse			250	mV	Comme spécifié dans ***[USB Type-C 2.0]***

Nom	Description	Min	Nom	Max	Unités	Commentaire
vNoiseActive	Amplitude du bruit lorsque le codage BMC est actif.			165	mV	Bruit crête-crête des lignes V_{BUS}, USB 2.0 et SBU après application du filtre de limitation de la largeur de bande Rx avec la constante de temps ***tRxFilter***.
vNoiseIdle	Amplitude du bruit lorsque le codage BMC est inactif.			300	mV	Bruit crête-crête des lignes V_{BUS}, USB 2.0 et SBU après application du filtre de limitation de la largeur de bande Rx avec la constante de temps ***tRxFilter***.
zBmcRx	Impédance d'entrée du récepteur	1			MΩ	
Note 1: La pénétration de bruit à large bande est due au couplage dans l'interconnexion du câble.						

5.8.6.1 Définition de l'état inactif

La collision des paquets BMC est évitée par la détection des transitions de signal au récepteur. C'est l'équivalent du squelch en modulation FSK. La détection est active lorsque des transitions ***nTransitionCount*** se produisent au récepteur à l'intérieur d'une fenêtre temporelle de ***tTransitionWindow***. Après avoir attendu ***tTransitionWindow*** sans avoir détecté ***nTransitionCount*** transitions, le bus ***Devoir/Doit/Doivent*** être déclaré inactif.

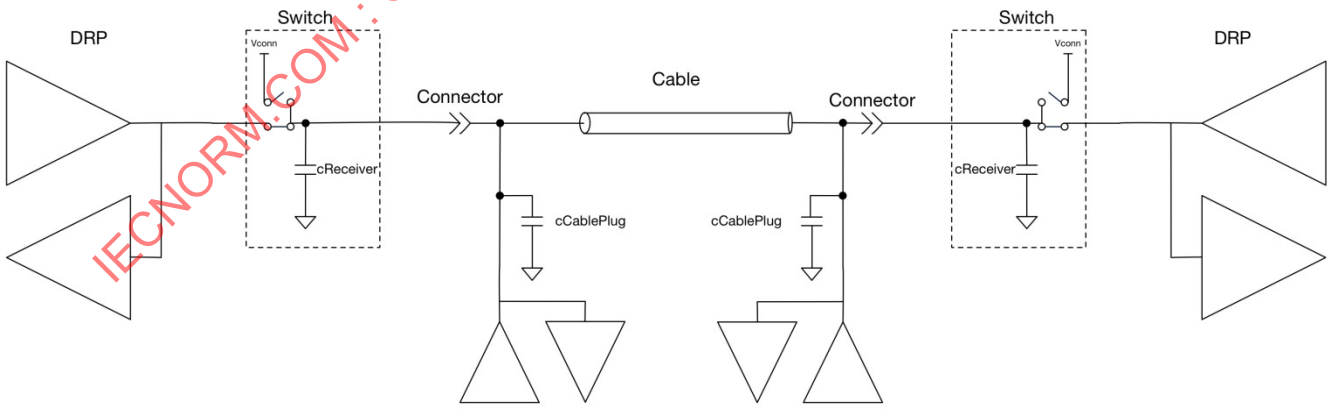
Pour plus d'informations sur le moment où des transmissions ***Pouvoir/Peut/Peuvent*** commencer, voir 5.8.5.4.

5.8.6.2 Multipoint

Le schéma de signalisation BMC convient à une utilisation dans des configurations multipoints comportant un ou plusieurs émetteurs/récepteurs BMC connectés au fil CC, par exemple lorsqu'une ou les deux extrémités d'un câble contiennent un émetteur/récepteur multipoint. Dans la présente spécification, l'emplacement de l'émetteur/récepteur multipoint est désigné comme étant la fiche de câble.

La Figure 5-27 ci-dessous montre une configuration multipoint type, avec deux DRP.

Figure 5-27 Exemple de configuration multipoint avec deux DRP

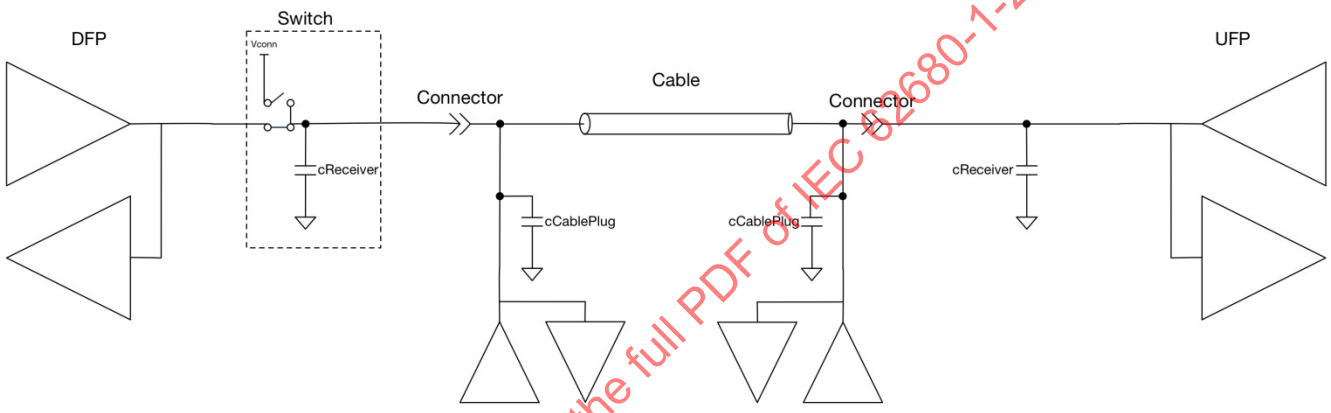


Anglais	Français
Switch	Commutateur
Connector	Connecteur
Cable	Câble

L'émetteur/récepteur multipoint ***Devoir/Doit/Doivent*** obéir à toutes les caractéristiques électriques spécifiées dans la présente section, à l'exception de celles qui concernent la capacité électrique. La capacité électrique maximale admise pour le nœud multipoint lorsqu'il ne pilote pas la ligne est *cCablePlug_CC*, définie dans ***[USB Type-C 2.0]***. Il n'y a aucune contrainte quant à la distance de l'émetteur/récepteur multipoint par rapport à l'extrémité de la fiche. Le ou les émetteurs/récepteurs multipoints ***Pouvoir/Peut/Peuvent*** être situés n'importe où le long du câble, fiches comprises. L'émetteur/récepteur multipoint rencontre moins de différence de masse comparé aux émetteurs/récepteurs dans l'hôte ou le dispositif, et il n'apporte aucune réflexion significative.

Il est possible d'avoir une configuration au branchement où un port est apte à être une source Vconn et pas l'autre port, ce dernier ne comportant pas de commutateur. Un exemple de DFP comportant un commutateur branché sur un port UFP sans commutateur est décrit sur la Figure 5-28. La capacité électrique sur la ligne CC pour un port inapte à jouer le rôle de source VCONN ***Devoir/Doit/Doivent*** tout de même se situer dans ***cReceiver***, sauf pendant la transmission.

Figure 5-28 Exemple de configuration multipoint montrant un DFP et un UFP



Anglais	Français
Switch	Commutateur
Connector	Connecteur
Cable	Câble

5.8.6.3 Détection de permutation rapide des rôles

Un destinataire initial se prépare à une permutation rapide des rôles en veillant à ce que, après qu'il a détecté le signal de permutation rapide des rôles, son alimentation électrique soit prête à répondre en appliquant ***vSafe5V*** selon le chronogramme décrit en 7.1.13. Le destinataire initial ***Devoir/Doit/Doivent*** uniquement répondre au signal de permutation rapide des rôles lorsque toutes les conditions suivantes sont respectées:

- un contrat explicite a été établi et les capacités de destinataire de la source initiale ont été reçues par le destinataire initial, à sa demande;
- dans le message ***Sink_Capabilities*** reçu de la source initiale, au moins un des bits de permutation rapide des rôles est défini sur son PDO fixe de 5 V;
- le récepteur initial est capable de fournir, et souhaite fournir, le courant demandé par la source initiale dans les bits de permutation rapide des rôles de son message ***Sink_Capabilities***.

Lors de la détection du signal de permutation rapide des rôles, tous les messages en suspens ***Devoir/Doit/Doivent*** être ***Rejeté(s/ée/ées)*** (voir 6.11.2.2.1).

Lorsque le destinataire initial est préparé pour une permutation rapide des rôles et que le bus est inactif, la tension CC moyennée sur ***tFRSwapRx*** min reste supérieure à 0,7 V (voir ***[USB Type-C 2.0]***) puisque la résistance Rp de la source correspond à une intensité de courant de 1,5 A ou de 3,0 A. Cependant, un bruit

de ***vNoiseIdle*** ***Pouvoir/Peut/Peuvent*** faire en sorte que la tension de la ligne CC atteigne 0,7 V-***vNoiseIdle***/2 pendant de courts intervalles de temps. Lorsque le destinataire initial est préparé pour une permutation rapide des rôles alors qu'il est en cours de transmission et que la source initiale est en train de signaler une demande de permutation rapide des rôles, la transmission est atténuée de manière que la tension CC de crête ne dépasse pas ***vFRSwapCableTx*** min. Par conséquent, lorsque le destinataire initial est préparé pour une permutation rapide des rôles, il ***Ne doit/doivent pas*** détecter un signal de permutation rapide lorsque la tension CC, moyennée sur ***tFRSwapRx*** min, est supérieure à 0,7 V. Lorsque le destinataire initial est préparé pour une permutation rapide des rôles, il ***Devoir/Doit/Doivent*** détecter une tension CC inférieure à ***vFRSwapCableTx*** min pendant ***tFRSwapRx*** comme étant une demande de permutation rapide des rôles. Note: Le destinataire initial n'est pas tenu de moyenner la tension CC pour satisfaire à ces exigences.

Le destinataire initial ***Devoir/Doit/Doivent*** initier l'AMS de permutation rapide des rôles dans un délai ***tFRSwapInit*** de détection de la demande de permutation rapide des rôles, afin d'attribuer les résistances Rp/Rd aux ports adéquats et de resynchroniser les diagrammes d'état (voir 6.3.19).

Le destinataire initial ***Devoir/Doit/Doivent*** devenir la nouvelle source et il ***Devoir/Doit/Doivent*** commencer à fournir ***vSafe5V*** avec l'intensité de courant USB Type-C (voir ***[USB Type-C 2.0]***) au plus tard dans un délai ***tSrcFRSwap*** après la baisse de V_{BUS} au-dessous de ***vSafe5V***. Un destinataire initial ***Devoir/Doit/Doivent*** désactiver ses circuits de détection du seuil de déconnexion de V_{BUS} tant que la détection d'une permutation rapide des rôles est active.

Note: Pendant la transition d'alimentation, la source V_{CONN} vers la ou les fiches de câbles ne peut pas être assurée.

5.9 Autotest intégré (BIST)

Les sections qui suivent définissent la fonctionnalité BIST qui ***Devoir/Doit/Doivent*** être prise en charge.

5.9.1 Mode porteur BIST

En ***Mode porteur BIST***, la couche physique ***Devoir/Doit/Doivent*** envoyer une chaîne continue de "1" et de "0" encodée BMC, ce qui permet de mesurer le bruit dans l'alimentation électrique et la dérive de fréquence.

Noter que cette transmission est purement une séquence de bits alternés et qu'elle ***Ne doit/doivent pas*** être formatée en tant que paquet.

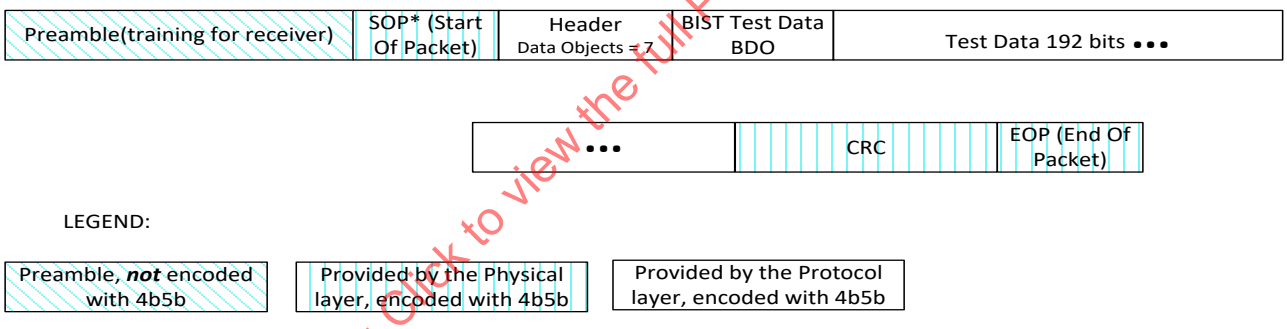
Voir aussi 6.4.3.

5.9.2 BIST Test Data

Un message ***BIST Test Data*** est utilisé par le dispositif d'essai pour envoyer divers modèles d'essai générés par ce dispositif d'essai à l'intention de l'UUT afin de soumettre le récepteur de l'UUT à l'essai. Voir aussi 6.4.3.

La Figure 5-29 montre la trame de données d'essai qui ***Devoir/Doit/Doivent*** être envoyée à l'UUT par le dispositif d'essai. Le message ***BIST***, avec un objet de données BIST ***BIST Test Data***, est constitué d'un préambule, suivi de ***SOP****, puis de l'en-tête du message avec une longueur de données de 7 objets de données, puis d'un objet de données BIST ***BIST Test Data***, puis de 6 objets de données contenant des données d'essai, puis du CRC et enfin d'un ***EOP***.

Figure 5-29 Trame de données d'essai



Anglais	Français
Preamble (training for receiver)	Préambule (entraînement pour le récepteur)
SOP* (Start of Packet)	SOP* (début de paquet)
Header	En-tête
BIST Test Data BDO	BDO de données d'essai BIST
Test Data 192 bits…	Données d'essai 192 bits…
EOP (End of Packet)	EOP (fin de paquet)
LEGEND:	LEGENDE:
Preamble: ***not*** encoded with 4b5b	Préambule: ***non*** codé en 4b5b
Provided by the Physical layer, encoded with 4b5b	Fourni par la couche physique, codé en 4b5b
Provided by the Protocol layer, encoded with 4b5b	Fourni par la couche protocole, codé en 4b5b

6. Couche protocole

6.1 Vue d'ensemble

Le présent chapitre décrit les exigences de la couche protocole de la spécification d'alimentation électrique par port USB, comprenant:

des précisions sur la façon dont les messages sont construits et utilisés;
l'utilisation de temporisateurs et de valeurs de temporisation;
l'utilisation des messages et des compteurs de répétition;
les opérations de réinitialisation;
le traitement des erreurs;
le comportement des états.

Pour avoir une vue d'ensemble sur la théorie de fonctionnement de l'alimentation électrique par port USB, se référer à la Section 2.6.

6.2 Messages

La présente spécification définit trois types de messages:

les messages de contrôle courts et qui servent à gérer le flux de messages entre les ports partenaires ou à échanger des messages qui ne nécessitent pas de données supplémentaires. Les messages de contrôle ont une longueur de 16 bits;
les messages de données qui servent à échanger des informations entre une paire de ports partenaires. Les messages de données ont une longueur allant de 48 à 240 bits;

- Il existe trois types de messages de données:
  - ceux utilisés pour exposer les capacités et négocier la puissance;
  - ceux utilisés pour le BIST;
  - ceux qui sont définis par le fournisseur;

les messages étendus servant à échanger des informations entre une paire de ports partenaires. Les messages étendus comportent jusqu'à ***MaxExtendedMsgLen*** octets;

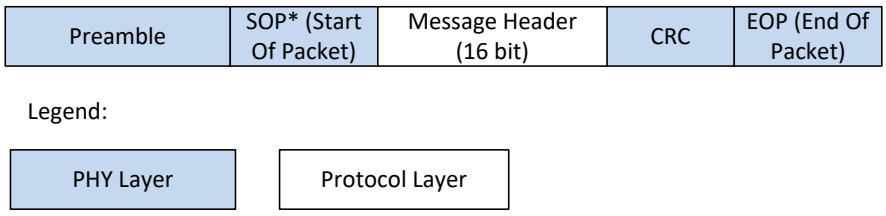
- Il existe plusieurs types de messages étendus:
  - ceux utilisés pour les informations de source et de batterie;
  - ceux utilisés pour la sécurité;
  - ceux utilisés pour la mise à jour du micrologiciel;
  - ceux qui sont définis par le fournisseur.

6.2.1 Construction des messages

Tous les messages ***Devoir/Doit/Doivent*** se composer d'un en-tête de message et d'une portion de données de longueur variable (y compris nulle). Soit un message provient de la couche protocole et est transmis à la couche physique, soit il est reçu par la couche physique et transmis à la couche protocole.

La Figure 6-1 représente un message de contrôle faisant partie d'un paquet et montre les parties fournies par la couche protocole et par la couche PHY.

Figure 6-1 Format de paquet d'alimentation électrique par port USB, avec charge utile de message de contrôle

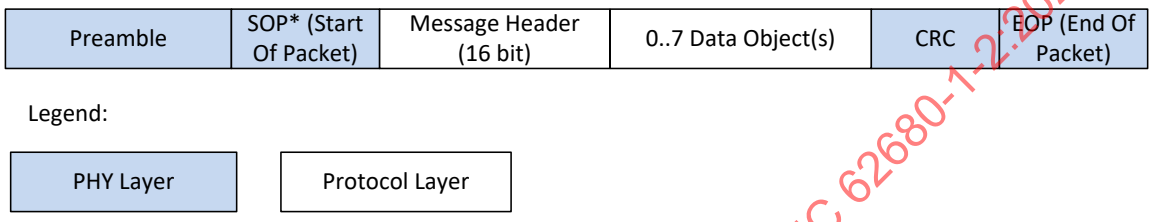


Anglais	Français

Preamble	Préambule
SOP* (Start of Packet)	SOP* (début de paquet)
Message Header (16 bit)	En-tête de message (16 bits)
EOP (End of Packet)	EOP (fin de paquet)
Legend:	Légende:
PHY Layer	Couche PHY
Protocol Layer	Couche protocole

La Figure 6-2 représente un message de données faisant partie d'un paquet et montre les parties fournies par la couche protocole et par la couche PHY.

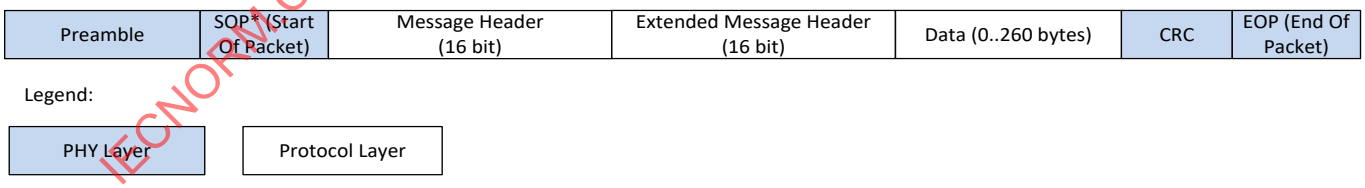
Figure 6-2 Format de paquet d'alimentation électrique par port USB, avec charge utile de message de données



Anglais	Français
Preamble	Préambule
SOP* (Start of Packet)	SOP* (début de paquet)
Message Header (16 bit)	En-tête de message (16 bits)
0..7 Data Oject(s)	0..7 objet(s) de données
EOP (End of Packet)	EOP (fin de paquet)
Legend:	Légende:
PHY Layer	Couche PHY
Protocol Layer	Couche protocole

La Figure 6-3 représente un message étendu faisant partie d'un paquet et montre les parties fournies par la couche protocole et par la couche PHY.

Figure 6-3 Format de paquet d'alimentation électrique par port USB, avec en-tête et charge utile de message étendu



Anglais	Français
Preamble	Préambule
SOP* (Start of Packet)	SOP* (début de paquet)
Message Header (16 bit)	En-tête de message (16 bits)
Extended Message Header (16 bit)	En-tête de message étendu (16 bits)
Data (0..260 bytes)	Données (0..260 octets)
EOP (End of Packet)	EOP (fin de paquet)
Legend:	Légende:
PHY Layer	Couche PHY

Protocol Layer	Couche protocole

6.2.1.1 En-tête de message

Chaque message ***Devoir/Doit/Doivent*** commencer par un en-tête de message, comme le montrent la Figure 6-1, la Figure 6-2 et la Figure 6-3, et comme défini dans le Tableau 6-1. L'en-tête de message contient des informations de base sur le message et sur les capacités du port PD.

L'en-tête de message ***Pouvoir/Peut/Peuvent*** être utilisé de façon autonome comme message de contrôle lorsque le nombre de champs d'objets de données est égal à 0 ou comme la première partie d'un message de données lorsque le champ ***Number of Data Objects*** est non nul.

Tableau 6-1 En-tête de message

Bit(s)	Début de paquet	Nom de champ	Référence
15	SOP*	***Extended***	Section 6.2.1.1.1
14…12	SOP*	***Number of Data Objects***	Section 6.2.1.1.2
11…9	SOP*	***MessageID***	Section 6.2.1.1.3
8	SOP uniquement	***Port Power Role***	Section 6.2.1.1.4
	SOP'/SOP''	***Cable Plug***	Section 6.2.1.1.7
7…6	SOP*	***Specification Revision***	Section 6.2.1.1.5
5	SOP uniquement	***Port Data Role***	Section 6.2.1.1.6
	SOP'/SOP''	***Réservé(s/ée/ées)***	Section 1.4.2.10
4…0	SOP*	***Message Type***	Section 6.2.1.1.8

6.2.1.1.1 Extended

Le champ ***Extended*** de 1 bit ***Devoir/Doit/Doivent*** être défini sur 0 pour indiquer un message de contrôle ou un message de données, et défini sur 1 pour indiquer un message étendu.

Le champ ***Extended*** ***Devoir/Doit/Doivent*** s'appliquer à tous les types de paquets SOP*.

6.2.1.1.2 Number of Data Objects

Lorsque le champ ***Extended*** est défini sur 0, le champ ***Number of Data Objects*** de 3 bits ***Devoir/Doit/Doivent*** indiquer le nombre d'objets de données de 32 bits qui suivent l'en-tête de message. Si ce champ est égal à 0, le message est un message de contrôle, et s'il est non nul, le message est un message de données.

Le champ ***Number of Data Objects*** ***Devoir/Doit/Doivent*** s'appliquer à tous les types de paquets SOP*.

Si le bit ***Extended*** et le bit ***Chunked*** sont tous deux définis sur 1, le champ ***Number of Data Objects*** ***Devoir/Doit/Doivent*** indiquer le nombre d'objets de données dans le message, complété pour atteindre la limite de 4 octets, y compris l'en-tête étendu, qui fait partie du premier objet de données.

Si le bit ***Extended*** est défini sur 1 et que le bit ***Chunked*** est défini sur 0, le champ ***Number of Data Objects*** ***Devoir/Doit/Doivent*** être ***Réservé(s/ée/ées)***. Noter que, dans ce cas, la longueur du message est déterminée uniquement par le champ ***Data Size*** dans l'en-tête de message étendu.

6.2.1.1.3 MessageID

Le champ ***MessageID*** de 3 bits est la valeur générée par un compteur déroulant, tenu à jour par l'auteur du message. Le ***MessageIDCounter*** ***Devoir/Doit/Doivent*** être initialisé à 0 à la mise sous tension, à la suite d'une réinitialisation logicielle ou d'une réinitialisation matérielle. Le ***MessageIDCounter*** ***Devoir/Doit/Doivent*** être incrémenté lorsqu'un message est reçu avec succès, comme l'indique la réception d'un message ***GoodCRC***. Note: L'utilisation de ***MessageID*** lors d'un essai avec des messages BIST est définie dans ***[USBPDCompliance]***.

Le champ ***MessageID*** ***Devoir/Doit/Doivent*** s'appliquer à tous les types de paquets SOP*.

6.2.1.1.4 Port Power Role

Le champ ***Port Power Role*** de 1 bit ***Devoir/Doit/Doivent*** indiquer le rôle d'alimentation actuel du port:

0b Destinataire
1b Source

Les messages tels que ***Ping*** et ***GotoMin***, qui ne sont toujours envoyés que par une source, ***Devoir/Doit/Doivent*** toujours avoir le champ ***Port Power Role*** défini sur "source". De même, les messages tels que le message ***Request***, qui ne sont toujours envoyés que par un destinataire, ***Devoir/Doit/Doivent*** toujours avoir le champ ***Port Power Role*** défini sur "destinataire".

Pendant la séquence de permutation des rôles d'alimentation, pour le port source initial, le champ ***Port Power Role*** ***Devoir/Doit/Doivent*** être défini sur "destinataire" dans le message ***PS_RDY***, indiquant que l'alimentation électrique de la source initiale est coupée (voir Figure 8-6 et Figure 8-7).

Pendant la séquence de permutation des rôles d'alimentation, pour le port destinataire initial, le champ ***Port Power Role*** ***Devoir/Doit/Doivent*** être défini sur "source" pour les messages initiés par le moteur de politique, après avoir reçu le message ***PS_RDY*** de la source initiale (voir Figure 8-6 et Figure 8-7).

Pendant la séquence de permutation rapide des rôles, pour le port source initial, le champ ***Port Power Role*** ***Devoir/Doit/Doivent*** être défini sur "destinataire" dans le message ***PS_RDY***, indiquant que V_{BUS} n'est pas pilotée par la source initiale et qu'elle se situe dans la plage de ***vSafe5V*** (voir Figure 8-18).

Pendant la séquence de permutation rapide des rôles, pour le port destinataire initial, le champ ***Port Power Role*** ***Devoir/Doit/Doivent*** être défini sur "source" pour les messages initiés par le moteur de politique, après avoir reçu le message ***PS_RDY*** de la source initiale (voir Figure 8-18).

Noter que le message ***GoodCRC*** envoyé par le destinataire initial en réponse au message ***PS_RDY*** de la source initiale a son champ ***Port Power Role*** défini sur "destinataire" dans la mesure où il est défini par la couche protocole. Les messages ultérieurs initiés par le moteur de politique, tels que le message ***PS_RDY*** envoyé pour indiquer que V_{BUS} est prête, ont le champ ***Port Power Role*** défini sur "source".

Le champ ***Port Power Role*** d'un message reçu ***Ne doit/doivent pas*** être vérifié par le récepteur et ***Ne doit/doivent pas*** conduire à une réinitialisation logicielle, une réinitialisation matérielle ou un rétablissement sur erreur s'il est incorrect.

Le champ ***Port Power Role*** ***Devoir/Doit/Doivent*** uniquement être défini pour des paquets SOP.

6.2.1.1.5 Specification Revision

Le champ ***Specification Revision*** ***Devoir/Doit/Doivent*** avoir l'une des valeurs suivantes (à l'exception de 11b):

00b – Révision 1.0
01b – Révision 2.0
10b – Révision 3.0
11b – ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé

Pour assurer l'interopérabilité avec les produits USBPD existants, les produits USBPD ***Devoir/Doit/Doivent*** prendre en charge chaque révision de la spécification PD à partir de ***[USBPD 2.0]*** pour ***SOP****; la seule exception à cette règle est un VPD qui ***Devoir/Doit/Doivent Ignorer*** les messages envoyés avec une révision de spécification PD 2.0 ou antérieure.

Après un branchement physique ou logique (rétablissement sur erreur USB Type-C®), un port découvre le niveau de révision de spécification commun entre lui-même et son port partenaire et/ou la ou les fiches de câbles. Il utilise ce niveau de révision de spécification jusqu'à ce qu'il se produise un débranchement, une réinitialisation matérielle ou un rétablissement sur erreur.

Après détection de la révision de spécification à utiliser, toutes les communications PD ***Devoir/Doit/Doivent*** être entièrement compatibles avec la révision correspondante de la spécification PD.

Le champ ***Specification Revision*** de 2 bits d'un message ***GoodCRC*** n'a pas de signification et il ***Devoir/Doit/Doivent*** être ignoré par le récepteur du message. L'expéditeur d'un message ***GoodCRC*** ***Devoir/Doit/Doivent*** définir le champ Specification Revision sur 01b lorsqu'il répond à un message qui contient 01b dans le champ Specification Revision de l'en-tête du message. L'expéditeur d'un message ***GoodCRC*** ***Pouvoir/Peut/Peuvent*** définir le champ Specification Revision sur 00b, 01b ou 10b lorsqu'il répond à un message qui contient 10b dans le champ Specification Revision de l'en-tête du message.

Le champ ***Specification Revision*** ***Devoir/Doit/Doivent*** s'appliquer à tous les types de paquets SOP*.

Un événement de branchement ou une réinitialisation matérielle ***Devoir/Doit/Doivent*** entraîner la nécessité de détecter la révision de spécification applicable pour les deux ports et les fiches de câbles, conformément aux règles prescrites ci-dessous.

Lorsque le port source communique en premier avec le port destinataire, le champ ***Specification Revision*** ***Devoir/Doit/Doivent*** être utilisé comme décrit dans les étapes suivantes:

1. le port source envoie un message ***Source_Capabilities*** au port destinataire en définissant le champ ***Specification Revision*** sur l'indice de la révision de la spécification d'alimentation électrique la plus élevée prise en charge par le port source;
2. le port destinataire répond par un message ***Request*** en définissant le champ ***Specification Revision*** sur l'indice de la révision de la spécification d'alimentation électrique la plus élevée prise en charge par le port destinataire, supérieur ou égal au contenu du champ ***Specification Revision*** reçu du port source;
3. les ports source et destinataire ***Devoir/Doit/Doivent*** utiliser le champ ***Specification Revision*** dans le message ***Request*** provenant du destinataire à l'étape 2 dans toutes les communications ultérieures jusqu'à ce qu'un débranchement, une réinitialisation matérielle ou un rétablissement sur erreur se produise.

Avant de conclure un contrat explicite, la source VCONN ***Devoir/Doit/Doivent*** appliquer les étapes suivantes pour établir un niveau de révision de spécification:

1. la source VCONN envoie une REQ ***Discover Identity*** à la fiche de câble (SOP') en définissant le champ ***Specification Revision*** du message sur l'indice de la révision de la spécification d'alimentation électrique la plus élevée prise en charge par la source VCONN. Après une permutation de VCONN, l'échange de messages ***Soft_Reset***/***Accept*** exigé est utilisé dans le même but (voir 6.3.13);
2. la fiche de câble répond par un ACK ***Discover Identity*** en définissant le champ ***Specification Revision*** du message sur l'indice de la révision de la spécification d'alimentation électrique la plus élevée prise en charge par la source VCONN, supérieur ou égal au contenu du champ ***Specification Revision*** reçu du port source;
3. la fiche de câble et la source VCONN ***Devoir/Doit/Doivent*** communiquer en utilisant la révision la plus ancienne des deux jusqu'à ce qu'un contrat explicite ait été établi;
4. le Tableau 6-2 montre la ***Specification Revision*** qui ***Devoir/Doit/Doivent*** être utilisée entre les ports partenaires et les fiches de câbles lorsque l'information ***Specification Revision*** a été découverte et qu'un contrat explicite est en place.

Notes:

a) une source VCONN qui ne communique pas avec la ou les fiches de câbles ***Pouvoir/Peut/Peuvent*** sauter la procédure ci-dessus;
b) si une fiche de câble ne répond pas à une REQ ***Discover Identity*** de révision 3.0 par un ACK ou BUSY ***Discover Identity***, la source VCONN ***Pouvoir/Peut/Peuvent*** répéter les étapes 1-4 en utilisant une REQ ***Discover Identity*** de révision 2.0 en étape 1 avant d'établir l'absence de fiche de câble avec laquelle communiquer.

Une source VCONN prenant en charge la révision 3.0 de la spécification d'alimentation électrique ***Pouvoir/Peut/Peuvent*** communiquer avec une fiche de câble prenant également en charge la révision 3.0 en utilisant une communication compatible avec la révision 3.0, quelle que soit l'information ***Specification Revision*** du port partenaire, tant qu'il n'existe pas de contrat explicite. Après établissement d'un contrat explicite, les ports partenaires et la ou les fiches de câbles ***Devoir/Doit/Doivent*** utiliser le Tableau 6-2 pour déterminer la révision à utiliser.

Toutes les données de tous les messages ***Devoir/Doit/Doivent*** être cohérentes avec le champ ***Specification Revision*** de l'en-tête de message pour ce message particulier.

Une fiche de câble ***Ne doit/doivent pas*** sauvegarder l'état de la ***Specification Revision*** fixée. Une fiche de câble ***Devoir/Doit/Doivent*** répondre avec la ***Specification Revision*** la plus élevée qu'elle prend en

charge, qui est supérieure ou égale à la ***Specification Revision*** contenue dans le message reçu de la source VCONN.

Les fiches de câbles ***Devoir/Doit/Doivent*** fonctionner en utilisant la même révision de spécification à la fois pour SOP' et pour SOP''. Les assemblages de câbles comportant deux fiches de câbles ***Devoir/Doit/Doivent*** fonctionner en utilisant la même révision de spécification pour les deux fiches de câbles.

Pour plus d'informations sur la manière dont les différentes révisions ***Devoir/Doit/Doivent*** interopérer, voir Tableau 6-2.

Tableau 6-2 Interopérabilité des révisions pendant un contrat explicite

Révision du port 1	Révision de la fiche de câble	Révision du port 2	Révision d'un fonctionnement port à port	Révision d'un fonctionnement port à fiche de câble
2	2	2	2	2
2	2	3	2	2
2	3	2	2	2
2	3	3	2	2
3	2	2	2	2
3	2	3	3	2
3	3	2	2	2
3	3	3	3	3

6.2.1.1.6 Port Data Role

Le champ ***Port Data Role*** de 1 bit ***Devoir/Doit/Doivent*** indiquer le rôle de transmission de données actuel du port:

- 0b UFP
- 1b DFP

Le champ ***Port Data Role*** ***Devoir/Doit/Doivent*** uniquement être défini pour des paquets SOP. Pour tous les autres paquets SOP*, le champ ***Port Data Role*** est ***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0.

S'il convient qu'un port USB Type-C® reçoive un message avec le champ ***Port Data Role*** défini sur le même rôle de transmission de données que son rôle de transmission de données actuel, à l'exception du message ***GoodCRC***, les actions de rétablissement sur erreur USB Type-C définies dans ***[USB Type-C 2.0]*** ***Devoir/Doit/Doivent*** être exécutées.

Pour un port USB Type-C, le champ ***Port Data Role*** ***Devoir/Doit/Doivent*** être défini sur la valeur par défaut lors du branchement après une réinitialisation matérielle: 0b pour un port avec Rd affirmée et 1b pour un port avec Rp affirmée.

Si un port n'a pas la capacité de communication USB, au branchement un port source ***Devoir/Doit/Doivent*** par défaut être un port DFP et un port destinataire ***Devoir/Doit/Doivent*** par défaut être un port UFP.

6.2.1.1.7 Cable Plug

Le champ ***Cable Plug*** de 1 bit ***Devoir/Doit/Doivent*** indiquer si ce message provient d'une fiche de câble ou d'un VDP:

- 0b Message provenant d'un port DFP ou UFP
- 1b Message provenant d'une fiche de câble

Le champ ***Cable Plug*** ***Devoir/Doit/Doivent*** s'appliquer uniquement aux types de paquets SOP' et SOP''.

6.2.1.1.8 Message Type

Le champ ***Message Type*** de 5 bits ***Devoir/Doit/Doivent*** indiquer le type de message en cours d'envoi. Pour décoder complètement le ***Message Type***, le champ ***Number of Data Objects*** est tout d'abord examiné pour déterminer si le message est un message de contrôle ou un message de données. Le ***Message Type*** spécifique peut ensuite être trouvé dans le Tableau 6-5 (message de contrôle) ou le Tableau 6-6 (message de données).

Le champ ***Message Type*** ***Devoir/Doit/Doivent*** s'appliquer à tous les types de paquets SOP*.

6.2.1.2 En-tête de message étendu

Chaque message étendu (indiqué par le champ ***Extended*** défini dans l'en-tête de message) ***Devoir/Doit/Doivent*** comporter un en-tête de message étendu suivant l'en-tête de message, comme représenté à la Figure 6-3 et défini dans le Tableau 6-3.

L'en-tête de message étendu sert à prendre en charge les messages étendus comportant des blocs de données de ***Data Size*** envoyés soit dans un message unique, soit en une série de fragments. Lorsque le bloc de données est envoyé en une série de fragments, chacun des fragments de la série, à l'exception du dernier, ***Devoir/Doit/Doivent*** comporter ***MaxExtendedMsgChunkLen*** octets. Le dernier fragment de la série ***Devoir/Doit/Doivent*** comporter le reste du bloc de données et, par conséquent, il peut avoir une taille inférieure à ***MaxExtendedMsgChunkLen*** octets et ***Devoir/Doit/Doivent*** être complété jusqu'à la limite d'objet de données de 4 octets suivante.

Tableau 6-3 En-tête de message étendu

Bit(s)	Début de paquet	Nom de champ	Référence
15	SOP*	***Chunked***	Section 6.2.1.2.1
14…11	SOP*	***Chunk Number***	Section 6.2.1.2.2
10	SOP*	***Request Chunk***	Section 6.2.1.2.3
9	SOP*	***Réservé(s/ée/ées)***	Section 1.4.2.10
8…0	SOP*	***Data Size***	Section 6.2.1.2.4

6.2.1.2.1 Chunked

Les ports partenaires ***Devoir/Doit/Doivent*** utiliser les champs Unchunked Extended Messages Supported du message ***Source_Capabilities*** et du message ***Request*** pour déterminer s'il y a lieu d'envoyer des messages de Data Size > ***MaxExtendedMsgLegacyLen*** octets dans un message étendu non fragmenté unique (voir 6.4.1.2.2.6 et 6.4.2.6).

Lorsque l'un ou l'autre des ports partenaires ne prend en charge que les messages étendus fragmentés:

1. le bit ***Chunked*** de chaque message étendu ***Devoir/Doit/Doivent*** être mis à 1;
2. chaque message étendu de taille de données > ***MaxExtendedMsgLegacyLen*** ***Devoir/Doit/Doivent*** être transmis sous forme de fragments entre les ports partenaires;
3. le champ ***Number of Data Objects*** de l'en-tête de message ***Devoir/Doit/Doivent*** indiquer le nombre d'objets de données dans le message, complété pour atteindre la limite de 4 octets, y compris l'en-tête étendu, qui fait partie du premier objet de données;
4. les points 1, 2 et 3 ci-dessus ***Devoir/Doit/Doivent*** s'appliquer jusqu'au débranchement de la paire de ports, jusqu'à une réinitialisation matérielle ou jusqu'à ce que la source coupe l'alimentation (sauf pendant une permutation des rôles d'alimentation ou une permutation rapide des rôles, lorsque la source initiale coupe l'alimentation afin que la nouvelle source puisse établir l'alimentation).

Lorsque les deux ports partenaires prennent en charge les messages étendus non fragmentés:

1. le bit ***Chunked*** de chaque message étendu ***Devoir/Doit/Doivent*** être mis à 0;
2. chaque message étendu ***Devoir/Doit/Doivent*** être transmis entre les ports partenaires sous forme non fragmentée;
3. le ***Number of Data Objects*** de l'en-tête de message est ***Réservé(s/ée/ées)***;

4. les points 1, 2 et 3 ci-dessus ***Devoir/Doit/Doivent*** s'appliquer jusqu'au débranchement de la paire de ports, jusqu'à une réinitialisation matérielle ou jusqu'à ce que la source coupe l'alimentation (sauf pendant une permutation des rôles d'alimentation ou une permutation rapide des rôles, lorsque la source initiale coupe l'alimentation afin que la nouvelle source puisse établir l'alimentation).

Lorsqu'elle envoie des messages étendus à la fiche de câble, la source VCONN ***Devoir/Doit/Doivent*** uniquement envoyer des messages fragmentés. Les fiches de câbles ***Devoir/Doit/Doivent*** toujours envoyer des messages étendus ayant une taille de données > ***MaxExtendedMsgLegacyLen*** fragmentée et ***Devoir/Doit/Doivent*** définir le bit ***Chunked*** sur 1 dans chaque message étendu.

Lorsque les messages étendus sont pris en charge, la fragmentation ***Devoir/Doit/Doivent*** être prise en charge.

6.2.1.2.2 Chunk Number

Le champ ***Chunk Number*** ***Devoir/Doit/Doivent*** uniquement être ***Valide(s)*** dans un message si le fanion ***Chunked*** est défini sur 1. Si le bit ***Chunked*** est défini sur 0, le champ ***Chunk Number*** ***Devoir/Doit/Doivent*** également être défini sur 0.

Le champ ***Chunk Number*** est utilisé différemment selon que le message est une demande de données ou qu'un bloc de données demandé est renvoyé:

Dans une demande de données, le champ ***Chunk Number*** indique le numéro du fragment demandé. L'auteur de la demande ***Devoir/Doit/Doivent*** seulement définir ce champ sur le numéro du fragment suivant de la série (le fragment suivant le dernier fragment reçu).

Dans le bloc de données demandé, le champ ***Chunk Number*** indique le numéro du fragment renvoyé. Le numéro de fragment de chaque fragment de la série ***Devoir/Doit/Doivent*** commencer à 0 et ***Devoir/Doit/Doivent*** s'incrémenter d'une unité pour chaque fragment jusqu'à une valeur maximale de 9, ce qui correspond à 10 fragments en tout.

6.2.1.2.3 Request Chunk

Le bit ***Request Chunk*** ***Devoir/Doit/Doivent*** uniquement servir au transfert fragmenté d'un message étendu lorsque le bit ***Chunked*** est défini sur 1 (voir Figure 6-7). Pour les transferts de messages étendus non fragmentés, les messages ***Devoir/Doit/Doivent*** être envoyés et reçus sans le mécanisme de demande/réponse (voir Figure 6-4).

Le bit ***Request Chunk*** ***Devoir/Doit/Doivent*** être défini sur 1 pour indiquer la présence d'une demande de fragment d'un bloc de données. Il ***Devoir/Doit/Doivent*** être défini sur 0 pour indiquer qu'il s'agit d'une réponse de fragment, contenant un fragment. A l'exception du fragment 0, un fragment de bloc de données demandé ***Devoir/Doit/Doivent*** uniquement être renvoyé sous forme de réponse de fragment à une demande correspondant à ce fragment. La demande de fragment et la réponse de fragment ***Devoir/Doit/Doivent*** toutes deux comporter la même valeur dans le champ ***Message Type***. Si le bit ***Request Chunk*** est défini sur 1, le champ ***Data Size*** ***Devoir/Doit/Doivent*** être défini sur 0.

6.2.1.2.4 Data Size

Le champ ***Data Size*** ***Devoir/Doit/Doivent*** indiquer combien d'octets de données au total comprend le bloc de données renvoyé. Le nombre total d'octets de données dans le message ***Ne doit/doivent pas*** dépasser ***MaxExtendedMsgLen***.

Si le champ ***Data Size*** est inférieur à ***MaxExtendedMsgLegacyLen*** et si le bit ***Chunked*** est défini, la charge utile du paquet ***Devoir/Doit/Doivent*** être complétée avec des zéros (0x00) jusqu'à la limite d'objet de données de 4 octets suivante.

Si le champ ***Data Size*** est plus grand que prévu pour un message étendu donné, mais inférieur ou égal à ***MaxExtendedMsgLen***, alors les champs prévus dans le message ***Devoir/Doit/Doivent*** être traités de manière appropriée et les champs supplémentaires ***Devoir/Doit/Doivent*** être ***Ignoré(s/ée/ées)***.

6.2.1.2.5 Exemples de messages étendus

Les exemples qui suivent montrent la transmission de messages étendus à la fois fragmentés (bit ***Chunked*** à 1) et non fragmentés (bit ***Chunked*** à 0). Les exemples utilisent un message ***Security_Request*** de ***Data Size*** de 7 octets, auquel il est répondu par un message ***Security_Response*** de ***Data Size*** de 30 octets. Les tailles de ces messages sont arbitraires et servent à représenter la transmission des messages.Elles ne sont pas censées correspondre à de véritables messages relatifs à la sécurité.

Pendant la négociation du contrat explicite après connexion, les utilisateurs du port utilisent les champs Unchunked Extended Messages Supported du message ***Source_Capabilities*** et du message ***Request*** pour déterminer la valeur du bit ***Chunked*** (voir Tableau 6-4). Lorsque les deux ports partenaires prennent en charge les messages non fragmentés, le bit ***Chunked*** est à 0; sinon, le bit ***Chunked*** est à 1.

Le bit ***Chunked*** sert à déterminer:

- si le mécanisme de demande/réponse de fragment est utilisé;
- si les messages étendus sont fragmentés;
- si le principe de remplissage est appliqué;
- si le champ ***Number of Data Objects*** est utilisé.

Les exemples qui suivent montrent l'utilisation prévue dans chacun des cas.

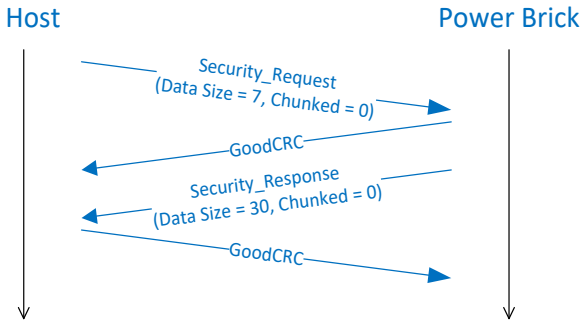
Tableau 6-4 Utilisation du bit Unchunked Message Supported

		Source: Message ***Source_Capabilities***	
		Bit Unchunked Message Supported = 0	Bit Unchunked Message Supported = 1
Destinataire: Message ***Request***	Bit Unchunked Message Supported = 0	Bit ***Chunked*** = 1	Bit ***Chunked*** = 1
	Bit Unchunked Message Supported = 1	Bit ***Chunked*** = 1	Bit ***Chunked*** = 0

6.2.1.2.5.1 Exemple non fragmenté de Security_Request/Security_Response

La Figure 6-4 représente une séquence type pour un message ***Security_Request*** auquel il est répondu par un message ***Security_Response*** faisant appel aux messages étendus non fragmentés (bit ***Chunked*** à 0) entre un hôte USB et un transformateur. Le bloc de données entier est renvoyé en un seul message. Le mécanisme de demande/réponse de fragment n'est pas utilisé.

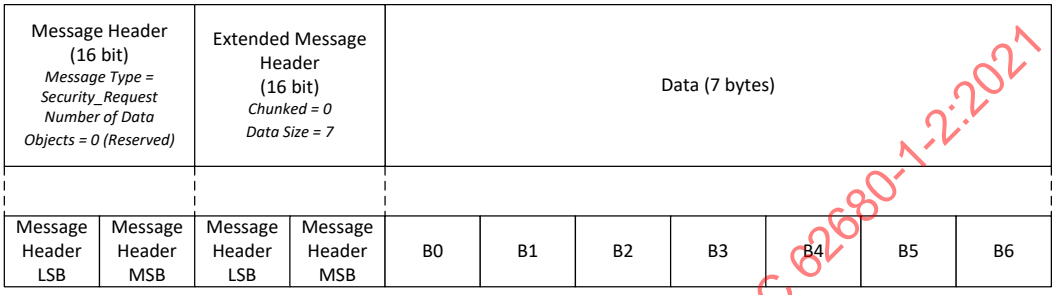
Figure 6-4 Exemple de séquence Security_Request non fragmentée (bit Chunked = 0)



Anglais	Français
Host	Hôte
Power Brick	Bloc d'alimentation

La Figure 6-5 décrit le message ***Security_Request*** représenté à la Figure 6-4. La figure représente l'ordre des octets sur le bus ainsi que l'absence de remplissage dans ce cas. Le champ ***Number of Data Objects*** a une valeur de 0, puisqu'il est ***Réservé(s/ée/ées)*** lorsque le bit ***Chunked*** est à 0. Le champ ***Data Size*** indique la longueur du message étendu lorsque le bit ***Chunked*** est à 0. Dans ce cas, elle est de 7 octets.

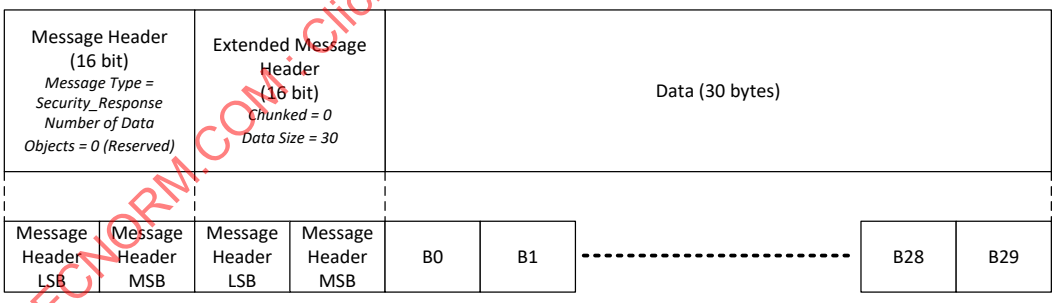
Figure 6-5 Exemple de transmission d'octets pour le message Security_Request de taille de données égale à 7 (le bit Chunked est à 0)



Anglais	Français
Message Header (16 bits)	En-tête de message (16 bits)
Reserved	Réservé
Extended Message Header (16 bit)	En-tête de message étendu (16 bits)

La Figure 6-6 décrit le message ***Security_Response*** représenté à la Figure 6-4. La figure représente l'ordre des octets sur le bus ainsi que l'absence de remplissage dans ce cas. Le champ ***Number of Data Objects*** a une valeur de 0, puisqu'il est ***Réservé(s/ée/ées)*** lorsque le bit ***Chunked*** est à 0. Le champ ***Data Size*** indique la longueur du message étendu lorsque le bit ***Chunked*** est à 0. Dans ce cas, elle est de 30 octets.

Figure 6-6 Exemple de transmission d'octets pour le message Security_Response de taille de données égale à 7 (le bit Chunked est à 0)



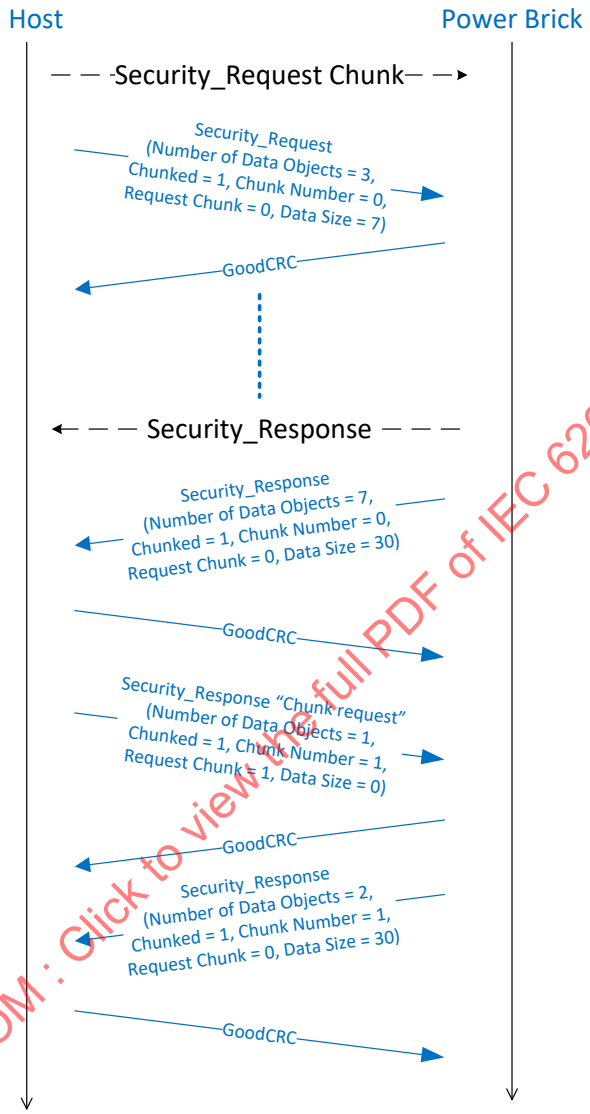
Anglais	Français
Message Header (16 bits)	En-tête de message (16 bits)
Reserved	Réservé
Extended Message Header (16 bit)	En-tête de message étendu (16 bits)

6.2.1.2.5.2 Exemple fragmenté de Security_Request/Security_Response

La Figure 6-7 représente une séquence type pour un message ***Security_Request*** auquel il est répondu par un message ***Security_Response*** faisant appel aux messages étendus fragmentés (bit ***Chunked*** à 1) entre un hôte USB et un transformateur. Noter que le ***Chunk Number*** 0 de chaque message étendu est envoyé

sans avoir besoin d'une demande de fragment, mais les ***Chunk Number*** 1 et suivants nécessitent une demande de fragment.

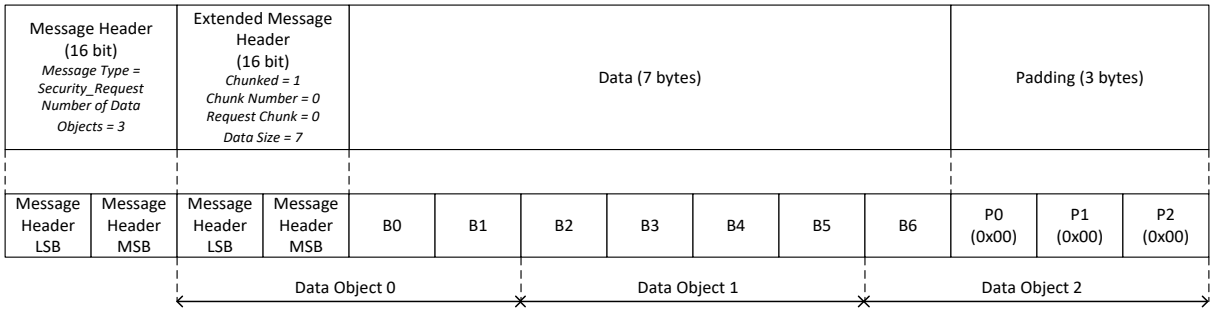
Figure 6-7 Exemple de séquence Security_Request fragmentée (bit Chunked = 1)



Anglais	Français
Host	Hôte
Power Brick	Bloc d'alimentation

La Figure 6-8 représente le message ***Security_Request*** de la Figure 6-7 avec davantage de détails, y compris l'ordre des octets sur le bus et le remplissage. Trois octets de remplissage ont été ajoutés au message, de sorte que le nombre total d'octets est un multiple de 32 bits, ce qui correspond à 3 objets de données. Le champ ***Number of Data Objects*** est défini sur 3 pour indiquer la longueur de ce fragment. Le ***Chunk Number*** est défini sur 0 et le champ***Data Size*** est défini sur 7 pour indiquer la longueur de l'ensemble du message étendu.

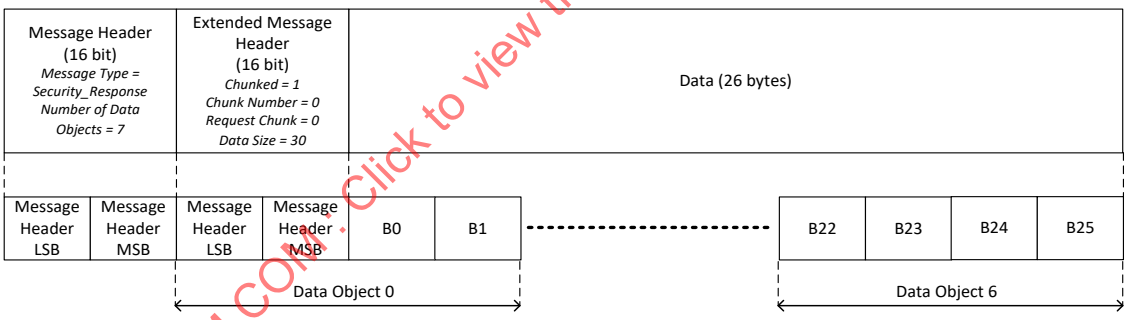
Figure 6-8 Exemple de message Security_Request de taille de données égale à 7 (le bit Chunked est à 1)



Anglais	Français
Message Header (16 bits)	En-tête de message (16 bits)
Extended Message Header (16 bit)	En-tête de message étendu (16 bits)
Data (7 bytes)	Données (7 octets)
Padding (3 bytes)	Remplissage (3 octets)
Data Object	Objet de données

La Figure 6-9 représente le ***Chunk Number*** 0 du message ***Security_Response*** de la Figure 6-7 avec davantage de détails, y compris l'ordre des octets sur le bus et le remplissage. Aucun remplissage n'est nécessaire pour ce fragment, puisque la charge utile complète de 26 octets plus l'en-tête de message étendu de 2 octets donne un multiple de 32 bits, correspondant à 7 objets de données. Le champ ***Number of Data Objects*** est défini sur 7 pour indiquer la longueur de ce fragment et le champ ***Data Size*** est défini sur 30 pour indiquer la longueur de l'ensemble du message étendu.

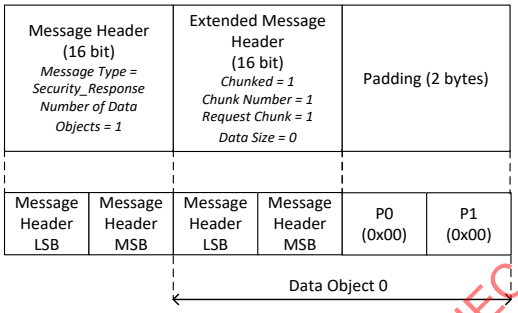
Figure 6-9 Exemple de fragment 0 de message Security_Request de taille de données égale à 30 (le bit Chunked est à 1)



Anglais	Français
Message Header (16 bits)	En-tête de message (16 bits)
Extended Message Header (16 bit)	En-tête de message étendu (16 bits)
Data (26 bytes)	Données (26 octets)
Data Object	Objet de données

La Figure 6-10 montre un exemple de format de message, d'ordre des octets et de remplissage pour la demande de fragment du message ***Security_Response*** pour le ***Chunk Number*** 1 représenté à la Figure 6-7. Dans la demande de fragment, le champ ***Number of Data Objects*** du message est défini sur 1 pour indiquer que la charge utile est de 32 bits, ce qui équivaut à 1 objet de données. Le bit ***Chunked*** étant défini sur 1, le mécanisme de demande/réponse de fragment est utilisé. Le message est une demande de fragment, ainsi le bit ***Request Chunk*** est défini sur 1. Dans ce cas, le fragment 1 fait l'objet d'une demande, ainsi ***Chunk Number*** est défini sur 0. ***Data Size*** est défini sur 0 pour indiquer la longueur du bloc de données en cours de transfert. Deux octets de remplissage sont ajoutés pour que la charge utile soit un multiple de 32 bits.

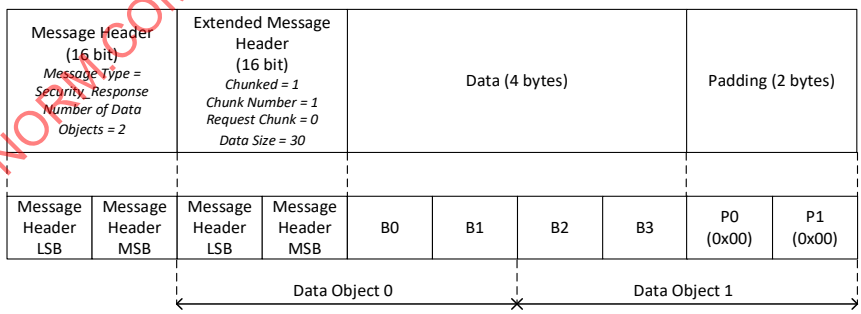
Figure 6-10 Exemple de transmission d'octets pour une demande de fragment de message Security_Response (le bit Chunked est à 1)



Anglais	Français
Message Header (16 bits)	En-tête de message (16 bits)
Extended Message Header (16 bit)	En-tête de message étendu (16 bits)
Padding (2 bytes)	Remplissage (2 octets)
Data Object	Objet de données

La Figure 6-11 représente le ***Chunk Number*** 1 du message ***Security_Response*** de la Figure 6-7 avec davantage de détails, y compris l'ordre des octets sur le bus et le remplissage. Deux octets de remplissage sont ajoutés pour que la charge utile soit un multiple de 32 bits, correspondant à 2 objets de données. Le champ ***Number of Data Objects*** est défini sur 2 pour indiquer la longueur de ce fragment et le champ ***Data Size*** est défini sur 30 pour indiquer la longueur de l'ensemble du message étendu.

Figure 6-11 Exemple de fragment 1 de message Security_Request de taille de données égale à 30 (le bit Chunked est à 1)



Anglais	Français
Message Header (16 bits)	En-tête de message (16 bits)
Extended Message Header (16 bit)	En-tête de message étendu (16 bits)
Data (4 bytes)	Données (4 octets)
Padding (2 bytes)	Remplissage (2 octets)
Data Object	Objet de données

6.3 Message de contrôle

Un message est défini comme un message de contrôle lorsque le champ ***Number of Data Objects*** de l'en-tête de message est défini sur 0. Le message de contrôle est uniquement composé d'un en-tête de message et d'un CRC. La couche protocole génère les messages de contrôle (message ***Accept***, message ***Reject***, etc.).

Les types de messages de contrôle sont spécifiés dans le champ ***Message Type*** de l'en-tête de message (bits 4…0) et sont récapitulés dans le Tableau 6-5. La colonne "Envoyé par" indique les entités qui ***Pouvoir/Peut/Peuvent*** envoyer le message concerné (source, destinataire ou fiche de câble). Les entités qui ne figurent pas dans la liste ***Ne doit/doivent pas*** émettre le message correspondant. La colonne "Début de paquet valide" indique les messages qui ne ***Devoir/Doit/Doivent*** être émis que dans des paquets SOP et les messages qui ***Pouvoir/Peut/Peuvent*** être émis dans des paquets SOP*.

Tableau 6-5 Types de messages de contrôle

Bits 4…0	Type de message	Envoyé par	Description	Début de paquet valide
0 0000	***Réservé(s/ée/ées)***	N/A	Toutes les valeurs non définies explicitement sont ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisées.	
0 0001	***GoodCRC***	Source, destinataire ou fiche de câble	Voir Section 6.3.1.	SOP*
0 0010	***GotoMin***	Source uniquement	Voir Section 6.3.2.	SOP uniquement
0 0011	***Accept***	Source, destinataire ou fiche de câble	Voir Section 6.3.3.	SOP*
0 0100	***Reject***	Source ou destinataire	Voir Section 6.3.4.	SOP uniquement
0 0101	***Ping***	Source uniquement	Voir Section 6.3.5.	SOP uniquement
0 0110	***PS_RDY***	Source ou destinataire	Voir Section 6.3.6.	SOP uniquement
0 0111	***Get_Source_Cap***	Destinataire ou DRP	Voir Section 6.3.7.	SOP uniquement
0 1000	***Get_Sink_Cap***	Source ou DRP	Voir Section 6.3.8.	SOP uniquement
0 1001	***DR_Swap***	Source ou destinataire	Voir Section 6.3.9	SOP uniquement
0 1010	***PR_Swap***	Source ou destinataire	Voir Section 6.3.10	SOP uniquement
0 1011	***VCONN_Swap***	Source ou destinataire	Voir Section 6.3.11	SOP uniquement
0 1100	***Wait***	Source ou destinataire	Voir Section 6.3.12	SOP uniquement
0 1101	***Soft_Reset***	Source ou destinataire	Voir Section 6.3.13	SOP*
01110	***Data_Reset***	Source ou destinataire	Voir Section 6.3.14	SOP uniquement
0 1111	***Data_Reset_Complete***	Source ou destinataire	Voir Section 6.3.15	SOP uniquement
1 0000	***Not_Supported***	Source, destinataire ou fiche de câble	Voir Section 6.3.16	SOP*
1 0001	***Get_Source_Cap_Extended***	Destinataire ou DRP	Voir Section 6.3.17	SOP uniquement

Bits 4...0	Type de message	Envoyé par	Description	Début de paquet valide
1 0010	*Get_Status*	Source ou destinataire	Voir Section 6.3.18	SOP*
1 0011	*FR_Swap*	Destinataire[1]	Voir Section 6.3.19	SOP uniquement
1 0100	*Get_PPS_Status*	Destinataire	Voir Section 6.3.20	SOP uniquement
1 0101	*Get_Country_Codes*	Source ou destinataire	Voir Section 6.3.21	SOP uniquement
1 0110	*Get_Sink_Cap_Extended*	Source ou DRP	Voir Section 6.3.22	SOP uniquement
1 0111-1 1111	***Réservé(s/ée/ées)***	N/A	Toutes les valeurs non définies explicitement sont ***Réservé(s/ée/ées)*** et ***Ne doivent pas*** être utilisées.	
Note 1: Dans ce cas, le port fournit *vSafe5V*, en ayant cependant Rd affirmée plutôt que Rp. Il définit le champ *Port Power Role* sur "destinataire", jusqu'à ce que l'AMS de permutation rapide des rôles soit terminée.				

6.3.1 Message GoodCRC

Le message *GoodCRC* ***Devoir/Doit/Doivent*** être envoyé par le récepteur pour confirmer que le précédent message a été reçu correctement (c'est-à-dire qu'il a un CRC correct). Le message *GoodCRC* ***Devoir/Doit/Doivent*** renvoyer le *MessageID* du message pour que l'émetteur puisse déterminer que le message correct est acquitté. Le premier bit du message *GoodCRC* ***Devoir/Doit/Doivent*** être renvoyé dans un délai *tTransmit* après réception du dernier bit du message précédent.

BIST n'envoie pas le message *GoodCRC* en étant dans un mode BIST continu (voir 6.4.3).

6.3.2 Message GotoMin

Le message *GotoMin* ne s'applique qu'aux destinataires ayant demandé une alimentation avec le fanion "GiveBack capable" défini dans l'objet de données de demande du destinataire.

Il est impératif que le port destinataire réduise son niveau de puissance opérationnelle à la valeur spécifiée dans le champ Minimum Operating Current de son dernier objet de données de demande du destinataire.

Le processus GotoMin est conçu pour permettre à la source de réaffecter temporairement de la puissance pour satisfaire à une exigence à court terme. Par exemple, une source peut réduire la consommation de puissance d'un destinataire pendant 10-20 secondes pour permettre à un autre destinataire, par exemple un DD, de se mettre en marche.

La source envoie ce message pour récolter de la puissance afin de répondre à une demande d'alimentation qu'elle ne peut satisfaire autrement. Le gestionnaire de politique d'utilisation des dispositifs détermine le ou les ports qui reçoivent le message.

Le destinataire ***Devoir/Doit/Doivent*** répondre à un message *GotoMin* en réduisant sa consommation de puissance à une valeur inférieure ou égale à la valeur prénégociée (courant opérationnel minimal) dans un délai *tSnkNewPower*.

La source envoie un message *GotoMin* en tant que raccourci dans le cadre du processus de négociation de puissance puisque la source et le destinataire ont déjà établi un contrat en ce qui concerne la puissance à renvoyer. En substance, la source n'a pas à annoncer ses capacités et le destinataire n'a pas à formuler de demande sur la base de ces capacités. La source envoie simplement le message *GotoMin* au lieu du message *Accept* envoyé normalement pendant le processus de négociation de puissance (voir étape 19 sur la Figure 8-5). Le processus de négociation de puissance s'achève alors de la façon normale à partir de ce point, la source envoyant un message *PS_RDY* dès que la transition d'alimentation électrique est effectuée. Les étapes du processus GotoMin sont entièrement décrites à la Figure 8-6.

La source ***Devoir/Doit/Doivent*** renvoyer la puissance au(x) destinataire(s) au(x)quel(s) elle l'a "empruntée" en utilisant le mécanisme GotoMin avant de pouvoir affecter une "nouvelle" puissance à d'autres dispositifs.

6.3.3 Message Accept

Le message ***Accept*** est une réponse ***Valide(s)*** dans les cas suivants:

- il ***Devoir/Doit/Doivent*** être envoyé par la source pour signaler au destinataire que la source souhaite satisfaire au message ***Request***;
- il ***Devoir/Doit/Doivent*** être envoyé par le récepteur du message ***PR_Swap*** pour signaler qu'il souhaite effectuer une permutation des rôles d'alimentation et qu'il a commencé la séquence de permutation des rôles d'alimentation;
- il ***Devoir/Doit/Doivent*** être envoyé par le récepteur du message ***DR_Swap*** pour signaler qu'il souhaite effectuer une permutation des rôles de transmission de données et qu'il a commencé la séquence de permutation des rôles de transmission de données;
- il ***Devoir/Doit/Doivent*** être envoyé par le récepteur du message ***VCONN_Swap*** pour signaler qu'il souhaite effectuer une permutation de VCONN et qu'il a commencé la séquence de permutation de VCONN;
- il ***Devoir/Doit/Doivent*** être envoyé par le récepteur du message ***FR_Swap*** pour indiquer qu'il a commencé la séquence de permutation rapide des rôles;
- il ***Devoir/Doit/Doivent*** être envoyé par le récepteur du message ***Soft_Reset*** pour indiquer qu'il a terminé sa réinitialisation logicielle.

Le message ***Accept*** ***Devoir/Doit/Doivent*** être envoyé dans un délai ***tReceiverResponse*** à partir de la réception du dernier bit du message (voir 6.6.2).

6.3.4 Message Reject

Le message ***Reject*** est une réponse ***Valide(s)*** dans les cas suivants:

- Il ***Devoir/Doit/Doivent*** être envoyé pour signaler au destinataire que la source n'est pas en mesure de satisfaire au message ***Request*** . Cela ***Pouvoir/Peut/Peuvent*** être dû à une demande ***Invalide(s)*** ou au fait que la source n'est plus en mesure de fournir ce qu'elle avait annoncé précédemment;
- il ***Devoir/Doit/Doivent*** être envoyé par le récepteur d'un message ***PR_Swap*** pour indiquer qu'il n'est pas en mesure d'effectuer une permutation des rôles d'alimentation;
- il ***Devoir/Doit/Doivent*** être envoyé par le récepteur d'un message ***DR_Swap*** pour indiquer qu'il n'est pas en mesure d'effectuer une permutation des rôles de transmission de données;
- il ***Devoir/Doit/Doivent*** être envoyé par le récepteur d'un message ***VCONN_Swap*** qui n'est actuellement pas la source VCONN pour indiquer qu'il n'est pas en mesure d'effectuer une permutation de VCONN.

Le message ***Reject*** ***Devoir/Doit/Doivent*** être envoyé dans un délai ***tReceiverResponse*** à partir de la réception du dernier bit du message (voir 6.6.2).

Le message ***Reject*** n'est pas une réponse ***Valide(s)*** lorsqu'un message n'est pas pris en charge. Dans ce cas, le message ***Not_Supported*** est renvoyé (voir 6.3.16).

6.3.5 Message Ping

Le message ***Ping*** était utilisé précédemment sur les connecteurs USB Type-A et USB Type-B pour déterminer la présence continue du destinataire lorsqu'aucun autre échange de messages n'avait lieu. Les connecteurs USB Type-C disposent d'un mécanisme pour déterminer la présence d'un destinataire, de sorte que lorsque les ports partenaires sont tous les deux connectés à l'aide de connecteurs USB Type-C, le message ***Ping*** n'est pas nécessaire, mais il ***Pouvoir/Peut/Peuvent*** être envoyé par une source si elle le souhaite. Un destinataire qui utilise un connecteur USB Type-C ***Ne doit/doivent pas*** s'attendre à recevoir de messages ***Ping***, mais il ***Ne doit/doivent pas*** traiter les messages ***Ping*** comme une erreur s'il en reçoit.

6.3.6 Message PS_RDY

Le message ***PS_RDY*** ***Devoir/Doit/Doivent*** être envoyé par la source (ou à la fois par le nouveau destinataire et la nouvelle source pendant la séquence de permutation des rôles d'alimentation ou la séquence de permutation rapide des rôles) pour indiquer que son alimentation électrique a atteint la condition de fonctionnement souhaitée (voir 8.3.2.2).

6.3.7 Message Get_Source_Cap

Le message ***Get_Source_Cap*** ***Pouvoir/Peut/Peuvent*** être envoyé par un port pour demander les capacités de la source et l'aptitude à l'alimentation double fonction de son port partenaire (par exemple apte à l'alimentation double fonction). Le port ***Devoir/Doit/Doivent*** répondre en renvoyant un message ***Source_Capabilities*** (voir 6.4.1.1.1).

6.3.8 Message Get_Sink_Cap

Le message ***Get_Sink_Cap*** ***Pouvoir/Peut/Peuvent*** être envoyé par un port pour demander les capacités du destinataire et l'aptitude à l'alimentation double fonction de son port partenaire (par exemple apte à l'alimentation double fonction). Le port ***Devoir/Doit/Doivent*** répondre en renvoyant un message ***Sink_Capabilities*** (voir 6.4.1.1.2).

6.3.9 Message DR_Swap

Le message ***DR_Swap*** sert à échanger un fonctionnement DFP et UFP entre ports partenaires tout en maintenant le sens du débit de puissance sur V_{BUS}. Le processus DR_Swap peut être utilisé par les ports partenaires, qu'ils prennent en charge ou non la capacité de communication USB. Un port DFP qui prend en charge la capacité de communication USB démarre en tant qu'hôte USB dès le branchement. Un port UFP qui prend en charge la capacité de communication USB démarre en tant que dispositif USB dès le branchement.

Les DRP ***[USB Type-C 2.0]*** ***Devoir/Doit/Doivent*** avoir la capacité d'effectuer une permutation des rôles de transmission de données à partir des états ***PE_SRC_Ready*** ou ***PE_SNK_Ready***. Les ports DFP et UFP ***Pouvoir/Peut/Peuvent*** avoir la capacité d'effectuer une permutation des rôles de transmission de données à partir des états ***PE_SRC_Ready*** ou ***PE_SNK_Ready***. Une permutation des rôles de transmission de données ***Devoir/Doit/Doivent*** être vue comme un branchement/débranchement de câble pour toute communication USB en cours entre les ports partenaires. En cas de modes actifs entre les ports partenaires lorsqu'un message ***DR_Swap*** est reçu, une réinitialisation matérielle ***Devoir/Doit/Doivent*** alors être effectuée (voir 6.4.4.3.4). Si la fiche de câble a un mode actif, le DFP ***Ne doit/doivent pas*** émettre de message ***DR_Swap*** et ***Devoir/Doit/Doivent*** faire quitter tous les modes actifs dans la fiche de câble avant d'accepter une demande de permutation DR.

La source de V_{BUS} et la source V_{CONN} ***Devoir/Doit/Doivent*** rester inchangées, de même que les résistances Rp/Rd sur le fil CC, pendant le processus de permutation des rôles de transmission de données.

Le message ***DR_Swap*** ***Pouvoir/Peut/Peuvent*** être envoyé par l'un ou l'autre des ports partenaires. Le récepteur du message ***DR_Swap*** ***Devoir/Doit/Doivent*** répondre en envoyant un message ***Accept***, un message ***Reject*** ou un message ***Wait***.

- Si un message ***Accept*** est envoyé, la source et le destinataire ***Devoir/Doit/Doivent*** échanger leurs rôles opérationnels.
- Si un message ***Reject*** est envoyé, le demandeur est informé du fait que le récepteur n'est pas en mesure d'effectuer une permutation des rôles de transmission de données, ou ne le souhaite pas, et aucune mesure ne ***Devoir/Doit/Doivent*** être prise.
- Si un message ***Wait*** est envoyé, le demandeur est informé du fait qu'une permutation des rôles de transmission de données pourra être possible dans le futur, mais qu'aucune mesure immédiate ne ***Devoir/Doit/Doivent*** être prise.

Avant une permutation des rôles de transmission de données, le port DFP initial ***Devoir/Doit/Doivent*** avoir son bit ***Port Data Role*** défini sur DFP et le port UFP initial ***Devoir/Doit/Doivent*** avoir son bit ***Port Data Role*** défini sur UFP.

Après une permutation des rôles de transmission de données réussie, le port DFP/hôte ***Devoir/Doit/Doivent*** devenir le port UFP/dispositif, et inversement. Le nouveau port DFP ***Devoir/Doit/Doivent*** avoir son bit ***Port Data Role*** défini sur DFP et le nouveau port UFP ***Devoir/Doit/Doivent*** avoir son bit ***Port Data Role*** défini sur UFP. Lorsque la communication USB est prise en charge par les deux ports partenaires, ***Il convient d'/de/qu'/que*** qu'une connexion de données USB soit établie conformément aux nouveaux rôles de transmission de données.

Si la permutation des rôles de transmission de données, après acceptation par le port partenaire, s'avère par la suite un échec, il est nécessaire de prendre des mesures de rétablissement sur erreur USB Type-C, telles qu'une déconnexion, comme défini dans ***[USB Type-C 2.0]***, afin d'essayer de rétablir la connexion sur le fil CC.

Pour plus d'informations, voir 8.3.2.8, 8.3.3.18.1 et 8.3.3.18.2.

6.3.10 Message PR_Swap

Le message ***PR_Swap*** ***Pouvoir/Peut/Peuvent*** être envoyé par l'un ou l'autre des ports partenaires pour demander un échange de rôles d'alimentation. Le récepteur du message ***Devoir/Doit/Doivent*** répondre en envoyant un message ***Accept***, un message ***Reject*** ou un message ***Wait***.

- Si un message ***Accept*** est envoyé, la source et le destinataire ***Devoir/Doit/Doivent*** effectuer une permutation des rôles d'alimentation.
- Si un message ***Reject*** est envoyé, le demandeur est informé du fait que le récepteur n'est pas en mesure d'effectuer une permutation des rôles d'alimentation, ou ne le souhaite pas, et aucune mesure ne ***Devoir/Doit/Doivent*** être prise.
- Si un message ***Wait*** est envoyé, le demandeur est informé du fait qu'une permutation des rôles d'alimentation pourra être possible dans le futur, mais qu'aucune mesure immédiate ne ***Devoir/Doit/Doivent*** être prise.

Après une permutation réussie des rôles d'alimentation, les ports partenaires ***Devoir/Doit/Doivent*** réinitialiser leurs couches protocole respectives (ce qui équivaut à une réinitialisation logicielle) en réinitialisant leur ***MessageIDCounter***, leur ***RetryCounter*** et les diagrammes d'états de leur couche protocole avant d'essayer d'établir un contrat explicite. A ce stade, la source ***Devoir/Doit/Doivent*** également réinitialiser son ***CapsCounter***.

La source ***Devoir/Doit/Doivent*** avoir Rp affirmée sur le fil CC et le destinataire ***Devoir/Doit/Doivent*** avoir Rd affirmée sur le fil CC, comme défini dans ***[USB Type-C 2.0]***. En cas de permutation des rôles d'alimentation de source à destinataire, le port ***Devoir/Doit/Doivent*** faire passer sa résistance de fil CC de Rp à Rd. En cas de permutation des rôles d'alimentation de destinataire à source, le port ***Devoir/Doit/Doivent*** faire passer sa résistance de fil CC de Rd à Rp. Les rôles de DFP (hôte), d'UFP (dispositif) et de source VCONN ***Devoir/Doit/Doivent*** rester inchangés pendant le processus de permutation des rôles d'alimentation.

Note: Pendant le processus de permutation des rôles d'alimentation, le destinataire initial ne se déconnecte pas, même si V_{BUS} descend au-dessous de ***vSafe5V***.

Pour plus d'informations sur la permutation des rôles d'alimentation, voir 7.3.9 et 7.3.10 au chapitre sur l'alimentation électrique, 8.3.2.6, 8.3.3.18.3 et 8.3.3.18.4 au chapitre sur la politique d'utilisation des dispositifs et 9.1.2 pour la mise en correspondance de V_{BUS} vers les états USB.

6.3.11 Message VCONN_Swap

Le message ***VCONN_Swap*** ***Devoir/Doit/Doivent*** être pris en charge par tout port pouvant fonctionner en tant que source VCONN.

Le message ***VCONN_Swap*** ***Pouvoir/Peut/Peuvent*** être envoyé par l'un ou l'autre des ports partenaires pour demander un échange de source VCONN. Le récepteur du message ***Devoir/Doit/Doivent*** répondre en envoyant un message ***Accept***, un message ***Reject***, un message ***Wait*** ou un message ***Not_Supported***.

- Si un message ***Accept*** est envoyé, les ports partenaires ***Devoir/Doit/Doivent*** procéder à une permutation de VCONN. La nouvelle source VCONN ***Devoir/Doit/Doivent*** envoyer un message ***PS_RDY***

dans un délai ***tVCONNSourceOn*** pour indiquer qu'elle fournit désormais VCONN. La source VCONN initiale ***Devoir/Doit/Doivent*** cesser de fournir VCONN dans un délai ***tVCONNSourceOff*** à partir de la réception du dernier bit de l'***EOP*** du message ***PS_RDY***.

- Si un message ***Reject*** est envoyé, le demandeur est informé du fait que le récepteur n'est pas en mesure d'effectuer une permutation de VCONN, ou ne le souhaite pas, et aucune mesure ne ***Devoir/Doit/Doivent*** être prise. Un message ***Reject*** ***Devoir/Doit/Doivent*** uniquement être envoyé par le port qui n'est actuellement pas la source Vconn, en réponse à un message ***VCONN_Swap***. Le port qui est actuellement la source Vconn ***Ne doit/doivent pas*** envoyer de message ***Reject*** en réponse à un message ***VCONN_Swap***.
- Si un message ***Wait*** est envoyé, le demandeur est informé du fait qu'une permutation de VCONN pourra être possible dans le futur, mais qu'aucune mesure immédiate ne ***Devoir/Doit/Doivent*** être prise. Un message ***Wait*** ***Devoir/Doit/Doivent*** uniquement être envoyé par le port qui n'est actuellement pas la source Vconn, en réponse à un message ***VCONN_Swap***. Le port qui est actuellement la source Vconn ***Ne doit/doivent pas*** envoyer de message ***Wait*** en réponse à un message ***VCONN_Swap***.
- Si un message ***Not_Supported*** est envoyé, le demandeur est informé que la permutation de VCONN n'est pas prise en charge. Le port qui n'est actuellement pas la source Vconn ***Pouvoir/Peut/Peuvent*** alimenter VCONN lorsqu'un message ***Not_Supported*** est reçu en réponse à un message ***VCONN_Swap***.

Les rôles de DFP (hôte), d'UFP (dispositif) et de source de VBUS ***Devoir/Doit/Doivent*** rester inchangés, de même que les résistances Rp/Rd sur le fil CC, pendant le processus de permutation de VCONN.

Note: VCONN ***Devoir/Doit/Doivent*** être alimentée en continu pendant le processus de permutation de VCONN afin de maintenir la ou les fiches de câbles alimentées, en établissant la nouvelle connexion avant la coupure de l'autre.

Avant de communiquer avec une fiche de câble, un port ***Devoir/Doit/Doivent*** s'assurer qu'il est la source VCONN et que les fiches de câbles sont alimentées, en effectuant une permutation de VCONN si nécessaire. Comme il ne peut pas être assuré que la source VCONN actuelle fournit VCONN, le seul moyen de s'assurer que les fiches de câbles sont alimentées consiste, pour un port qui souhaite communiquer avec une fiche de câble, à devenir la source VCONN. Si un message ***Not_Supported*** est renvoyé en réponse au message ***VCONN_Swap***, le port est autorisé à devenir la source VCONN jusqu'à une réinitialisation matérielle ou un débranchement.

Une source VCONN qui est également une source peut tenter d'envoyer une commande ***Discover Identity*** en utilisant SOP' vers une fiche de câble avant l'établissement d'un contrat explicite.

Note: Même s'il est actuellement la source VCONN, le destinataire n'est pas autorisé à initier une AMS avec une fiche de câble, sauf si Rp est définie sur ***SinkTxOk*** (voir 6.9).

6.3.12 Message Wait

Le message ***Wait*** est une réponse ***Valide(s)*** à un message ***Request***, ***PR_Swap***, ***DR_Swap*** ou ***VCONN_Swap***.

Il ***Devoir/Doit/Doivent*** être envoyé pour signaler au destinataire que la source n'est pas en mesure de satisfaire à la demande en ce moment.
Il ***Devoir/Doit/Doivent*** être envoyé par le récepteur d'un message ***PR_Swap*** pour indiquer qu'il n'est pas en mesure d'effectuer une permutation des rôles d'alimentation en ce moment.
Il ***Devoir/Doit/Doivent*** être envoyé par le récepteur d'un message ***DR_Swap*** pour indiquer qu'il n'est pas en mesure d'effectuer une permutation des rôles de transmission de données à ce moment.
Il ***Devoir/Doit/Doivent*** être envoyé par le récepteur d'un message ***VCONN_Swap*** qui n'est actuellement pas la source VCONN pour indiquer qu'il n'est pas en mesure d'effectuer une permutation de VCONN en ce moment.

Le message ***Wait*** ***Devoir/Doit/Doivent*** être envoyé dans un délai ***tReceiverResponse*** à partir de la réception du dernier bit du message (voir 6.6.2).

6.3.12.1 Wait en réponse à un message Request

Le message ***Wait*** est utilisé par la source lorsqu'un destinataire ayant de la puissance en réserve la demande. Le message ***Wait*** donne du temps à la source pour récupérer la puissance dont elle a besoin pour satisfaire à la demande au moyen du processus GotoMin. Une source ne ***Devoir/Doit/Doivent*** envoyer un message ***Wait*** en réponse à un message ***Request*** que lorsqu'il existe un contrat explicite entre les ports partenaires.

Le destinataire est autorisé à répéter le message ***Request*** en utilisant ***SinkRequestTimer*** et il ***Devoir/Doit/Doivent*** respecter un délai ***tSinkRequest*** à partir de la réception du message ***Wait*** avant d'envoyer un autre message ***Request***.

6.3.12.2 Wait en réponse à un message PR_Swap

Le message ***Wait*** est utilisé en réponse à un message ***PR_Swap*** pour indiquer qu'une permutation des rôles d'alimentation pourra être possible dans le futur. Cela peut se produire dans tous les cas où il est nécessaire que le dispositif qui reçoit le message ***PR_Swap*** évalue davantage la demande, par exemple en demandant les capacités de l'auteur du message ***PR_Swap***. Après avoir achevé cette évaluation, ***Il convient d'/de/qu'/que*** l'un des ports partenaires relance le processus de permutation des rôles d'alimentation en envoyant un message ***PR_Swap***.

Le message ***Wait*** est également utilisé lorsqu'un hub fonctionne en mode hybride quand une demande ne peut être satisfaite (voir ***[USBTypeCBridge 1.0]***).

Un port qui reçoit un message ***Wait*** en réponse à un message ***PR_Swap*** ***Devoir/Doit/Doivent*** attendre ***tPRSwapWait*** à partir de la réception du message ***Wait*** avant d'envoyer un autre message ***PR_Swap***.

6.3.12.3 Wait en réponse à un message DR_Swap

Le message ***Wait*** est utilisé en réponse à un message ***DR_Swap*** pour indiquer qu'une permutation des rôles de transmission de données pourra être possible dans le futur. Cela peut se produire dans tous les cas où il est nécessaire que le dispositif qui reçoit le message ***DR_Swap*** évalue davantage la demande. Après avoir achevé cette évaluation, ***Il convient d'/de/qu'/que*** l'un des ports partenaires relance le processus de permutation des rôles de transmission de données en envoyant un message ***DR_Swap***.

Un port qui reçoit un message ***Wait*** en réponse à un message ***DR_Swap*** ***Devoir/Doit/Doivent*** attendre ***tDRSwapWait*** à partir de la réception du message ***Wait*** avant d'envoyer un autre message ***DR_Swap***.

6.3.12.4 Wait en réponse à un message VCONN_Swap

Le message ***Wait*** est utilisé en réponse à un message ***VCONN_Swap*** pour indiquer qu'une permutation de ***VCONN_Swap*** pourra être possible dans le futur. Cela peut se produire dans tous les cas où il est nécessaire que le dispositif qui reçoit le message ***VCONN_Swap*** évalue davantage la demande. Un message ***Wait*** ***Devoir/Doit/Doivent*** uniquement être envoyé par le port qui n'est actuellement pas la source Vconn, en réponse à un message ***VCONN_Swap***. Le port qui est actuellement la source Vconn ***Ne doit/doivent pas*** envoyer de message ***Wait*** en réponse à un message ***VCONN_Swap***. Après avoir achevé cette évaluation, ***Il convient d'/de/qu'/que*** l'un des ports partenaires relance le processus de permutation de VCONN en envoyant un message ***VCONN_Swap***.

Un port qui reçoit un message ***Wait*** en réponse à un message ***VCONN_Swap*** ***Devoir/Doit/Doivent*** attendre ***tVCONNSwapWait*** à partir de la réception du message ***Wait*** avant d'envoyer un autre message ***VCONN_Swap***.

6.3.13 Message Soft Reset

Un message ***Soft_Reset*** ***Pouvoir/Peut/Peuvent*** être émis par la source ou par le destinataire à son port partenaire demandant une réinitialisation logicielle. Le message ***Soft_Reset*** ***Devoir/Doit/Doivent*** entraîner une réinitialisation logicielle de la paire de ports connectée (voir 6.8.1). Si le message ***Soft_Reset*** échoue, une réinitialisation matérielle ***Devoir/Doit/Doivent*** être initiée dans un délai ***tHardReset*** à partir de la dernière expiration de ***CRCReceiveTimer***, après avoir effectué ***nRetryCount*** relances.

Un message ***Soft_Reset*** est utilisé pour un rétablissement après des erreurs de la couche protocole, en mettant les compteurs de message à un état connu afin de récupérer la synchronisation des messages. Le message ***Soft_Reset*** n'a aucun effet sur la source ou le destinataire, c'est-à-dire sur le sens négocié précédemment. La tension et l'intensité de courant restent inchangées. Le fonctionnement modal n'est pas affecté par une réinitialisation logicielle. Cependant, après une réinitialisation logicielle, une négociation de contrat explicite a lieu afin de rétablir la communication PD et de ramener le fonctionnement d'état des deux ports partenaires aux états ***PE_SNK_Ready*** ou ***PE_SRC_Ready***, selon ce qui est approprié (voir 8.3.3.4).

Un message ***Soft_Reset*** ***Pouvoir/Peut/Peuvent*** être envoyé par la source ou par le destinataire en cas d'erreur de synchronisation des messages. Si l'erreur n'est pas corrigée par la réinitialisation logicielle, un signal ***Hard Reset*** ***Devoir/Doit/Doivent*** être émis (voir 6.8).

Un message ***Soft_Reset*** ***Devoir/Doit/Doivent*** être adressé à une entité spécifique qui dépend du type de paquet SOP* utilisé. Les messages ***Soft_Reset*** envoyés à l'aide de paquets SOP ***Devoir/Doit/Doivent*** entraîner la réinitialisation logicielle du port partenaire uniquement. Les messages ***Soft_Reset*** envoyés à l'aide de paquets SOP'/SOP'' ***Devoir/Doit/Doivent*** uniquement déclencher une réinitialisation logicielle de la fiche de câble correspondante.

Après une permutation de Vconn, il est nécessaire que la source Vconn réinitialise la couche protocole de la fiche de câble afin d'assurer la synchronisation de ***MessageID***. Si, après une permutation de VCONN, la source VCONN souhaite communiquer avec une fiche de câble en utilisant des paquets SOP', elle ***Devoir/Doit/Doivent*** émettre un message ***Soft_Reset*** en utilisant un paquet SOP' afin de réinitialiser la couche protocole de la fiche de câble. Si la source VCONN souhaite communiquer avec une fiche de câble en utilisant des paquets SOP'', elle ***Devoir/Doit/Doivent*** émettre un message ***Soft_Reset*** en utilisant un paquet SOP'' afin de réinitialiser la couche protocole de la fiche de câble.

6.3.14 Message Data_Reset

Le message ***Data_Reset*** ***Pouvoir/Peut/Peuvent*** être envoyé par le DFP ou l'UFP et ***Devoir/Doit/Doivent*** réinitialiser la connexion de données USB et quitter tous les modes alternatifs avec son port partenaire tout en maintenant l'alimentation sur VBUS. Les ports compatibles USB4™ ***Devoir/Doit/Doivent*** prendre en charge le message ***Data_Reset*** et les autres ports ***Peuvent*** prendre en charge le message ***Data_Reset***.

Le message ***Data_Reset*** ne ***Devoir/Doit/Doivent*** pas modifier:

- le contrat d'alimentation existant;
- les rôles de données existants (c'est-à-dire quel port est le DFP et quel port est l'UFP).

Le destinataire du message ***Data_Reset*** ***Devoir/Doit/Doivent*** répondre en envoyant un message ***Accept***, puis suivre le processus décrit dans les étapes suivantes. Ni l'expéditeur ni le destinataire ne ***Devoir/Doit/Doivent*** initier une permutation de VCONN tant que le processus de réinitialisation des données n'est pas terminé. Après réception du message ***Accept*** ou ***GoodCRC*** à la suite du message ***Accept***, selon le port qui envoie le message ***Data_Reset***:

1. le DFP ***Devoir/Doit/Doivent***:
   - déconnecter les signaux D+/D ***[USB 2.0]*** du port;
   - s'il fonctionne en mode ***[USB 3.2]***, supprimer les terminaisons Rx du port (voir ***[USB 3.2]***);
   - s'il fonctionne en mode ***[USB4]***, piloter le SBTX du port sur un signal logique bas (voir ***[USB4]***);
2. le DFP et l'UFP ***Devoir/Doit/Doivent*** quitter tous les modes alternatifs, le cas échéant;
3. réinitialiser le câble:
   - si le port source VCONN est également l'UFP, il ***Devoir/Doit/Doivent*** exécuter le processus de cycle de mise sous/hors tension VCONN de l'UFP, comme décrit en 7.1.15.1;
   - si le port source VCONN est également l'DFP, il ***Devoir/Doit/Doivent*** exécuter le processus de cycle de mise sous/hors tension VCONN de l'DFP, comme décrit en 7.1.15.2;
   - le DFP ***Devoir/Doit/Doivent*** quitter le processus de mise sous/hors tension de VCONN en tant que source VCONN et alimenter VCONN;
4. après ***tDataReset***, le DFP ***Devoir/Doit/Doivent***:
   - reconnecter les signaux D+/D- ***[USB 2.0]***;
   - si le port fonctionnait en mode ***[USB 3.2]*** ou ***[USB4]***, réappliquer les terminaisons Rx du port (voir ***[USB 3.2]***);

5. le processus de réinitialisation des données est terminé; le DFP ***Devoir/Doit/Doivent*** envoyer un message ***Data_Reset_Complete*** et entrer dans le flux de découverte et d'entrée USB4 (voir ***[USB Type-C 2.0]***).

Si l'initiateur du message ***Data_Reset*** ne reçoit pas le message ***Accept*** dans un délai ***tSenderResponse***, il ***Devoir/Doit/Doivent*** passer à l'état ***ErrorRecovery***.

6.3.15 Message Data_Reset_Complete

Le message ***Data_Reset_Complete*** ***Devoir/Doit/Doivent*** être envoyé par le DFP à l'UFP pour indiquer que le processus de réinitialisation des données est terminé (voir 6.3.14).

6.3.16 Message Not_Supported

Le message ***Not_Supported*** ***Devoir/Doit/Doivent*** être envoyé par un port ou une fiche de câble en réponse à tout message qu'il ne prend pas en charge. Le fait de renvoyer un message ***Not_Supported*** est admis par hypothèse dans la présente spécification et n'a pas été évoqué de façon explicite à la section 6.12, qui définit les cas où le message ***Not_Supported*** est renvoyé.

6.3.17 Message Get_Source_Cap_Extended

Le message ***Get_Source_Cap_Extended*** (obtention des capacités étendues de la source) est envoyé par un port pour demander des informations supplémentaires sur les capacités de source d'un port. ***Il convient d'/de/qu'/que*** le port réponde en renvoyant un message ***Source_Capabilities_Extended*** (voir 6.5.1).

6.3.18 Message Get_Status

Le message ***Get_Status*** est envoyé par un port par ***SOP*** pour demander le statut actuel du port partenaire.

La source ou le destinataire ***Devoir/Doit/Doivent*** répondre en renvoyant un message ***Statut*** (voir 6.5.2). Un port recevant un message ***Alert*** (voir 6.4.6) indique que le statut de la source ou du destinataire a changé, et ***Il convient d'/de/qu'/que*** il soit relu en utilisant un message ***Get_Status***.

Le message ***Get_Status*** ***Pouvoir/Peut/Peuvent*** également être envoyé à un câble actif pour obtenir son statut actuel en utilisant ***SOP'***/***SOP''***.

Le câble actif ***Devoir/Doit/Doivent*** répondre en renvoyant un message ***Statut*** (voir 6.5.2).

6.3.19 Message FR_Swap

Le message ***FR_Swap*** ***Devoir/Doit/Doivent*** être envoyé par la nouvelle source dans un délai ***tFRSwapInit*** après avoir détecté un signal de permutation rapide des rôles (voir 5.8.6.3 et 6.6.17.3). L'AMS de permutation rapide des rôles est nécessaire pour appliquer Rp à la nouvelle source et Rd au nouveau destinataire, et pour resynchroniser les diagrammes d'état. La durée ***tFRSwapInit*** ***Devoir/Doit/Doivent*** être mesurée à partir de l'envoi du signal FRS pendant ***tFRSwapRx*** (max) jusqu'au moment où le dernier bit de l'***EOP*** du message ***FR_Swap*** a été transmis par la couche physique.

Le récepteur du message ***FR_Swap*** ***Devoir/Doit/Doivent*** répondre en envoyant un message ***Accept***.

Après une permutation rapide des rôles réussie, les ports partenaires ***Devoir/Doit/Doivent*** réinitialiser leurs couches protocole respectives (ce qui équivaut à une réinitialisation logicielle) en réinitialisant leur ***MessageIDCounter***, leur ***RetryCounter*** et les diagrammes d'états de leur couche protocole avant d'essayer d'établir un contrat explicite. A ce stade, la source ***Devoir/Doit/Doivent*** également réinitialiser son ***CapsCounter***.

Cela permet de s'assurer que seule la fiche de câble répond par un message ***GoodCRC*** à la commande ***Discover Identity***.

Avant l'AMS de permutation rapide des rôles, la nouvelle source ***Devoir/Doit/Doivent*** avoir Rd affirmée sur le fil CC et le nouveau destinataire ***Devoir/Doit/Doivent*** avoir Rp affirmée sur le fil CC. Noter qu'il s'agit d'une attribution incorrecte de Rp/Rd (puisque Rp suit la source et que Rd suit le destinataire, comme défini dans ***[USB Type-C 2.0]***) qui est corrigée par l'AMS de permutation rapide des rôles.

Pendant l'AMS de permutation rapide des rôles, la nouvelle source ***Devoir/Doit/Doivent*** faire passer sa résistance de fil CC de Rd à Rp et le nouveau destinataire ***Devoir/Doit/Doivent*** faire passer sa résistance

de fil CC de Rp à Rd. Les rôles de DFP (hôte), d'UFP (dispositif) et de source VCONN ***Devoir/Doit/Doivent*** rester inchangés pendant le processus de permutation rapide des rôles.

Il convient d'/de/qu'/que que la source initiale évite d'être la source VCONN (en utilisant le processus de permutation de VCONN) chaque fois qu'elle ne communique pas activement avec le câble, puisqu'il est difficile pour la source initiale de maintenir l'alimentation VCONN pendant le processus de permutation rapide des rôles.

Note: Une permutation rapide des rôles est une solution du "meilleur effort" face à une situation dans laquelle un dispositif PDUSB a perdu son alimentation externe. Ce processus peut se produire à tout moment, même pendant une AMS non interruptible, auquel cas un traitement d'erreur tel qu'une réinitialisation matérielle ou un rétablissement sur erreur ***[USB Type-C 2.0]*** est déclenché.

Note: Pendant le processus de permutation rapide des rôles, le destinataire initial ne se déconnecte pas, même si VBUS descend au-dessous de ***vSafe5V***.

Pour plus d'informations sur le processus de permutation rapide des rôles, voir 7.1.13 et 7.2.10 au chapitre sur l'alimentation électrique, 8.3.3.18.5 et 8.3.3.18.6 au chapitre sur la politique d'utilisation des dispositifs et 9.1.2 pour la mise en correspondance de VBUS et des états USB.

6.3.20 Get_PPS_Status

Le message ***Get_PPS_Status*** est envoyé par le destinataire pour demander des informations supplémentaires sur le statut d'une source. Le port ***Devoir/Doit/Doivent*** répondre en renvoyant un message ***PPS_Status*** (voir 6.5.10).

6.3.21 Get_Country_Codes

Le message ***Get_Country_Codes*** est envoyé par un port pour demander les codes pays alpha-2 que son port partenaire prend en charge, comme défini dans l'***[ISO3166]***. Le port partenaire ***Devoir/Doit/Doivent*** répondre en renvoyant un message ***Country_Codes*** (voir 6.5.11).

6.3.22 Message Get_Sink_Cap_Extended

Le message ***Get_Sink_Cap_Extended*** (obtention des capacités étendues de la source) est envoyé par un port pour demander des informations supplémentaires sur les capacités de destinataire d'un port. Le port ***Devoir/Doit/Doivent*** répondre en renvoyant un message ***Sink_Capabilities_Extended*** (voir 6.5.13).

6.4 Message de données

Un message de données ***Devoir/Doit/Doivent*** être constitué d'un en-tête de message et être suivi d'un ou plusieurs objets de données. Les messages de données sont facilement identifiables dans la mesure où le champ ***Number of Data Objects*** est une valeur non nulle.

Il existe plusieurs types d'objets de données:

- l'objet de données BIST (BDO) utilisé pour l'essai de conformité de la couche PHY;
- l'objet de données d'alimentation (PDO) utilisé pour exposer les capacités d'alimentation d'un port source ou les exigences de puissance d'un destinataire;
- l'objet de données de demande (RDO) utilisé par un port destinataire pour négocier un contrat;
- l'objet de données définies par le fournisseur (VDO) utilisé pour transporter des informations spécifiques au fournisseur;
- l'objet de données de statut de batterie (BSDO) utilisé pour transporter des informations sur le statut de batterie;
- l'objet de données d'alerte (ADO) utilisé pour indiquer des événements survenant sur la source ou le destinataire.

Le type d'objet de données utilisé dans un message de données est défini par le champ ***Message Type*** de l'en-tête de message et est résumé dans le Tableau 6-6. La colonne "Envoyé par" indique les entités qui ***Pouvoir/Peut/Peuvent*** envoyer le message concerné (source, destinataire ou fiche de câble). Les entités qui ne figurent pas dans la liste ***Ne doit/doivent pas*** émettre le message correspondant. La colonne

"Début de paquet valide" indique les messages qui ***Devoir/Doit/Doivent*** uniquement être émis dans des paquets SOP et les messages qui ***Pouvoir/Peut/Peuvent*** être émis dans des paquets SOP*.

Tableau 6-6 Types de messages de données

Bits 4...0	Type	Envoyé par	Description	Début de paquet valide
0 0000	***Réservé(s/ée/ées)***		Toutes les valeurs non définies explicitement sont ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisées.	
0 0001	***Source_Capabilities***	Source ou alimentation double fonction	Voir Section 6.4.1.2	SOP uniquement
0 0010	***Request***	Destinataire uniquement	Voir Section 6.4.2	SOP uniquement
0 0011	***BIST***	Dispositif d'essai, source ou destinataire	Voir Section 6.4.3	SOP*
0 0100	***Sink_Capabilities***	Destinataire ou alimentation double fonction	Voir Section 6.4.1.3	SOP uniquement
0 0101	***Battery_Status***	Source ou destinataire	Voir Section 6.4.5	SOP uniquement
0 0110	***Alert***	Source ou destinataire	Voir Section 6.4.6	SOP uniquement
0 0111	***Get_Country_Info***	Source ou destinataire	Voir Section 6.4.7	SOP uniquement
0 1000	***Enter_USB***	DFP	Voir Section 6.4.8	SOP*
0 1001 -0 1110	***Réservé(s/ée/ées)***		Toutes les valeurs non définies explicitement sont ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisées.	
0 1111	***Vendor_Defined***	Source, destinataire ou fiche de câble	Voir Section 6.4.4	SOP*
1 0000-1 1111	***Réservé(s/ée/ées)***		Toutes les valeurs non définies explicitement sont ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisées.	

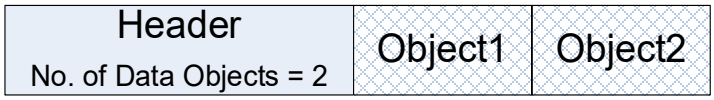
6.4.1 Message de capacités

Un message de capacités (message ***Source_Capabilities*** ou message ***Sink_Capabilities***) ***Devoir/Doit/Doivent*** comporter au moins un objet de données d'alimentation pour ***vSafe5V***. Le message de capacités ***Devoir/Doit/Doivent*** également comporter l'information du port émetteur suivie d'au maximum 6 objets de données d'alimentation supplémentaires. Les objets de données d'alimentation d'un message de capacités ***Devoir/Doit/Doivent*** être envoyés dans l'ordre suivant:

1. l'objet d'alimentation fixe ***vSafe5V*** ***Devoir/Doit/Doivent*** toujours être le premier objet;
2. le reste des objets d'alimentation fixe, le cas échéant, ***Devoir/Doit/Doivent*** être envoyé selon l'ordre de tension, de la plus faible vers la plus élevée;
3. les objets d'alimentation par batterie, le cas échéant, ***Devoir/Doit/Doivent*** être envoyés selon l'ordre minimal de tension, de la plus faible vers la plus élevée;
4. les objets d'alimentation variable (batterie exceptée), le cas échéant, ***Devoir/Doit/Doivent*** être envoyés selon l'ordre minimal de tension, de la plus faible vers la plus élevée;

5. les objets d'alimentation électrique programmable, le cas échéant, ***Devoir/Doit/Doivent*** être envoyés selon l'ordre maximal de tension, de la plus faible vers la plus élevée.

Figure 6-12 Exemple de message de capacités avec 2 objets de données d'alimentation



Anglais	Français
Header	En-tête
No. of Data Objects	Nombre d'objets de données
Object	Objet

Sur la Figure 6-12, le champ ***Number of Data Objects*** est à 2: ***vSafe5V***, plus une autre tension.

Les objets de données d'alimentation (PDO) et les objets de données d'alimentation augmentée (APDO) sont identifiés par le champ de type d'en-tête de message. Ils sont utilisés pour former des messages ***Source_Capabilities*** et des messages ***Sink_Capabilities***.

Il existe trois types d'objets de données d'alimentation. Ils contiennent des informations supplémentaires au-delà de ce qui est codé dans l'en-tête de message pour identifier chacun des trois types d'objets de données d'alimentation électrique:

- l'objet de données d'alimentation fixe est utilisé pour exposer des alimentations électriques à tension constante correctement régulée;
- l'objet de données d'alimentation électrique variable est utilisé pour exposer des alimentations électriques très mal régulées;
- l'objet de données de batterie est utilisé pour exposer des batteries pouvant être directement connectées à V_{BUS}.

Il existe un type d'objet de données d'alimentation augmentée:

- l'objet de données d'alimentation électrique programmable est utilisé pour exposer une alimentation électrique dont la tension de sortie peut être réglée par programme sur la plage de tension annoncée.

Les objets de données d'alimentation sont également utilisés pour exposer des capacités supplémentaires qui ***Pouvoir/Peut/Peuvent*** être utilisées, comme dans le cas d'une permutation des rôles d'alimentation.

Une liste d'un ou plusieurs objets de données d'alimentation ***Devoir/Doit/Doivent*** être envoyée par la source afin de transmettre ses capacités. Le destinataire ***Pouvoir/Peut/Peuvent*** alors demander une de ces capacités en renvoyant un objet de données de demande contenant un indice vers un objet de données d'alimentation, afin de négocier un contrat établi d'un commun accord.

Lorsque les valeurs de tension et de courant maximales et minimales sont données dans des PDO, ces valeurs ***Devoir/Doit/Doivent*** être des valeurs absolues.

La source et le destinataire ***Ne doit/doivent pas*** négocier un niveau de puissance susceptible de permettre au courant de dépasser le courant maximal supporté par leurs embases ou par la fiche branchée (voir ***[USB Type-C 2.0]***). La source ***Devoir/Doit/Doivent*** limiter ses capacités offertes à l'intensité de courant maximale supportée par son embase et par la fiche branchée. Un destinataire ne ***Devoir/Doit/Doivent*** formuler une demande qu'à partir des capacités offertes par la source. Pour plus d'informations, voir 4.4.

Les sources exposent leurs capacités d'alimentation en envoyant un message ***Source_Capabilities***. Les destinataires exposent leurs exigences de puissance en envoyant un message ***Sink_Capabilities***. Les deux se composent d'un certain nombre d'objets de données d'alimentation de 32 bits (voir Tableau 6-7).

Tableau 6-7 Objet de données d'alimentation

Bit(s)	Description	
B31…30	**Valeur**	**Paramètre**
	00b	Alimentation fixe (Vmin = Vmax)
	01b	Batterie
	10b	Alimentation variable (batterie exceptée)
	11b	Objet de données d'alimentation augmentée (APDO - Augmented Power Data Object)
B29…0	Les capacités d'alimentation spécifiques sont décrites par les PDO dans les sections qui suivent.	

L'objet de données d'alimentation augmentée (APDO) est défini pour permettre la prise en charge de plus que les quatre types de PDO, en étendant le champ Power Data Object de 2 à 4 bits lorsque les B31…B30 ont pour valeur 11b. La structure APDO générique est indiquée dans le Tableau 6-8.

Tableau 6-8 Objet de données d'alimentation augmentée

Bit(s)	Description
B31…30	11b – Objet de données d'alimentation augmentée (APDO)
B29…28	00b – Alimentation électrique programmable 01b-11b - ***Réservé(s/ée/ées)***
B27…0	Les capacités d'alimentation spécifiques sont décrites par les APDO dans les sections qui suivent.

6.4.1.1 Utilisation du message de capacités

6.4.1.1.1 Utilisation par les sources

Les sources envoient un message ***Source_Capabilities*** (voir 6.4.1) soit dans le cadre de l'annonce des capacités du port, soit en réponse à un message ***Get_Source_Cap.***

A la suite d'une réinitialisation matérielle, d'une mise sous tension ou de l'insertion d'une fiche, un port source ***Devoir/Doit/Doivent*** émettre un message ***Source_Capabilities*** après chaque temporisation ***SourceCapabilityTimer*** en tant qu'annonce qui ***Devoir/Doit/Doivent*** être interprétée par le port destinataire au moment du branchement. La source ***Devoir/Doit/Doivent*** continuer d'envoyer un nombre minimal de ***nCapsCount*** messages ***Source_Capabilities*** jusqu'à réception d'un message ***GoodCRC***.

De plus, un message ***Source_Capabilities*** ***Devoir/Doit/Doivent*** être envoyé par un port uniquement dans les cas suivants:

- par le port source à partir de l'état ***PE_SRC_Ready*** en cas de modification de son aptitude à fournir l'alimentation électrique sur ce port;
- par un port source ou un port d'alimentation double fonction en réponse à un message ***Get_Source_Cap***.
- ***En option***, par un port source à partir de l'état ***PE_SRC_Ready*** en cas de modification de l'alimentation disponible dans un système à ports multiples, même sans modification des capacités de source de ce port.

6.4.1.1.2 Utilisation par les destinataires

Les destinataires envoient un message ***Sink_Capabilities*** (voir 6.4.1.3) en réponse à un message ***Get_Sink_Cap.***

Lorsqu'il détecte ***vSafe5V*** sur V_{BUS} et après une temporisation ***SinkWaitCapTimer*** sans avoir reçu de message ***Source_Capabilities***, un destinataire apte à l'alimentation électrique par port USB ***Devoir/Doit/Doivent*** envoyer une commande Hard Reset. Si la source branchée est apte à l'alimentation électrique par port USB, elle répond par l'envoi de messages ***Source_Capabilities***, permettant ainsi aux négociations de puissance de commencer.

6.4.1.1.3 Utilisation par les dispositifs d'alimentation double fonction

Les dispositifs d'alimentation double fonction émettent un message ***Source_Capabilities*** (voir 6.4.1) dans le cadre de l'annonce des capacités du port lorsque celui-ci fonctionne en tant que source. Les dispositifs d'alimentation double fonction émettent un message ***Source_Capabilities*** (voir 6.4.1) en réponse à un message ***Get_Source_Cap*** quel que soit leur rôle opérationnel actuel. De même, les dispositifs d'alimentation double fonction émettent un message ***Sink_Capabilities*** (voir 6.4.1.3) en réponse à un message ***Get_Sink_Cap*** quel que soit leur rôle opérationnel actuel.

6.4.1.2 Message Source_Capabilities

Un port source ***Devoir/Doit/Doivent*** signaler ses capacités dans une série d'objets de données d'alimentation de 32 bits (voir Tableau 6-7) au sein d'un message ***Source_Capabilities*** (voir Figure 6-12). Les objets de données d'alimentation sont utilisés pour transmettre les capacités d'un port source à fournir l'alimentation électrique, y compris les ports d'alimentation double fonction fonctionnant actuellement en tant que destinataire.

Chaque objet de données d'alimentation ***Devoir/Doit/Doivent*** décrire une capacité spécifique de la source, telle qu'alimentation par batterie (par exemple 2,8-4,1 V) ou alimentation électrique fixe (par exemple 12 V) pour une intensité de courant admissible maximale. Le champ ***Number of Data Objects*** dans l'en-tête de message ***Devoir/Doit/Doivent*** définir le nombre d'objets de données d'alimentation qui suivent l'en-tête de message d'un message de données. Toutes les sources ***Devoir/Doit/Doivent*** offrir au minimum un objet de données d'alimentation qui présente ***vSafe5V***. Une source ***Ne doit/doivent pas*** offrir de multiples objets de données d'alimentation de même type (fixe, variable, batterie) et de même tension, mais elle ***Devoir/Doit/Doivent*** à la place offrir un objet de données d'alimentation ayant l'intensité de courant disponible la plus élevée pour cette capacité de source et cette tension.

Les destinataires avec prise en charge d'accessoires n'assurent pas l'alimentation de V_{BUS} (voir ***[USB Type-C 2.0]***). Les destinataires avec prise en charge d'accessoires sont toujours considérés comme des sources lorsqu'il fournissent V_{CONN} à un accessoire, même si V_{BUS} n'est pas appliquée; dans ce cas, ils ***Devoir/Doit/Doivent*** annoncer ***vSafe5V*** avec l'intensité maximale de courant fixée à 0 mA dans le premier objet de données d'alimentation. L'objectif principal est de permettre au destinataire avec prise en charge d'accessoires de passer à l'état ***PE_SRC_Ready*** afin d'entrer dans un mode alternatif.

Un destinataire ***Devoir/Doit/Doivent*** évaluer chacun des messages ***Source_Capabilities*** qu'il reçoit et ***Devoir/Doit/Doivent*** répondre par un message ***Request***. Si sa consommation de puissance dépasse les capacités de la source, il ***Devoir/Doit/Doivent*** renégocier de manière à ne pas dépasser les capacités de la source annoncées le plus récemment.

Un destinataire qui évalue le message ***Source_Capabilities*** qu'il reçoit et identifie un APDO PPS ***Devoir/Doit/Doivent*** périodiquement redemander l'APDO PPS, au moins à intervalles de ***tPPSRequest***, jusqu'à l'un des événements suivants:

- le destinataire demande quelque chose d'autre que l'APDO PPS;
- une permutation des rôles d'alimentation a lieu;
- une réinitialisation matérielle a lieu.

Une source qui a accepté un message ***Request*** avec un RDO programmable ***Devoir/Doit/Doivent*** émettre un signal ***Hard Reset*** si elle n'a pas reçu de message ***Request*** avec un RDO programmable dans un délai ***tPPSTimeout***. La source ***Devoir/Doit/Doivent*** mettre fin à ce comportement après l'un des événements suivants:

- la réception d'une ***Request*** avec un RDO fixe, variable ou de batterie;
- une permutation des rôles d'alimentation a lieu;
- une réinitialisation matérielle a lieu.

6.4.1.2.1 Gestion de la réserve de puissance

Une réserve de puissance ***Pouvoir/Peut/Peuvent*** être affectée à un destinataire lorsqu'il formule une demande à partir des capacités de la source incluant une intensité de courant/puissance de fonctionnement maximale. La taille de la réserve de puissance pour un destinataire particulier se calcule

comme la différence entre son champ Maximum Operating Current/Power et son champ Operating Current/Power. Pour un hub comprenant des ports multiples, cette même réserve de puissance ***Pouvoir/Peut/Peuvent*** être partagée entre plusieurs destinataires. La réserve de puissance ***Pouvoir/Peut/Peuvent*** également être utilisée temporairement par un destinataire qui a indiqué qu'il peut rendre de la puissance, en définissant le fanion GiveBack.

Lorsqu'une réserve de puissance a été affectée à un destinataire, la source ***Devoir/Doit/Doivent*** indiquer la réserve de puissance à l'intérieur de chaque message ***Source_Capabilities*** qu'elle envoie. Lorsque la même puissance de réserve est partagée entre plusieurs destinataires, la source ***Devoir/Doit/Doivent*** indiquer la réserve de puissance à l'intérieur de chaque message ***Source_Capabilities*** qu'elle envoie à chaque destinataire. Chaque fois qu'une source envoie ses capacités avec la capacité de réserve de puissance, puis accepte une demande provenant d'un destinataire, y compris la réserve de puissance indiquée par l'intensité de courant/la puissance de fonctionnement maximale, elle confirme que la réserve de puissance fait partie du contrat explicite avec le destinataire.

Lorsque la réserve est utilisée temporairement par un destinataire apte à la restitution, la source ***Devoir/Doit/Doivent*** indiquer la réserve de puissance disponible dans chacun des messages ***Source_Capabilities*** qu'elle envoie. Cependant, dans une telle situation, lorsque la réserve de puissance est demandée par un destinataire, la source ***Devoir/Doit/Doivent*** renvoyer un message ***Wait*** pendant qu'elle récupère cette puissance en utilisant un message ***GotoMin***. Dès que la puissance supplémentaire a été récupérée, la source ***Devoir/Doit/Doivent*** envoyer un nouveau message ***Source_Capabilities*** afin de déclencher une nouvelle demande du destinataire qui demande la réserve de puissance.

La réserve de puissance ***Pouvoir/Peut/Peuvent*** être réaffectée par la source à tout moment, mais cette réaffectation ***Devoir/Doit/Doivent*** être indiquée au(x) destinataire(s) qui utilise(nt) la réserve de puissance en envoyant un nouveau message ***Source_Capabilities***.

6.4.1.2.2 Objet de données d'alimentation électrique fixe

Le Tableau 6-9 décrit le PDO d'alimentation électrique fixe (00b). Voir 7.1.3 pour les exigences électriques applicables à l'alimentation.

Puisque tous les fournisseurs USB prennent en charge ***vSafe5V***, l'objet de données d'alimentation électrique fixe de la tension ***vSafe5V*** exigée est également utilisé pour acheminer des informations supplémentaires renvoyées dans les bits 29 à 25. Tous les autres objets de données d'alimentation électrique fixe ***Devoir/Doit/Doivent*** avoir les bits 29…22 mis à zéro.

Pour une source qui n'offre aucune capacité, la tension (B19…10) ***Devoir/Doit/Doivent*** être réglée à 5 V et l'intensité de courant maximale ***Devoir/Doit/Doivent*** être réglée à 0 mA. Cela est utilisé dans certains cas tels que celui d'un dispositif d'alimentation double fonction qui n'offre aucune capacité dans son rôle par défaut, ou lorsqu'une alimentation externe est exigée afin d'offrir une alimentation.

Lorsqu'une source souhaite qu'un destinataire, qui consomme la puissance fournie par V_{BUS}, passe à son état de puissance le plus faible, la tension (B19…10) ***Devoir/Doit/Doivent*** être réglée à 5 V et l'intensité de courant maximale ***Devoir/Doit/Doivent*** être réglée à 0 mA. Cela est utilisé dans les cas où la source souhaite que l'appel de courant du destinataire soit égal à ***pSnkSusp***.

Tableau 6-9 PDO d'alimentation électrique fixe - Source

Bit(s)	Description
B31…30	Alimentation fixe
B29	Alimentation double fonction
B28	Prise en charge de la veille USB
B27	Puissance non contrainte
B26	Aptitude aux communications USB
B25	Données double fonction
B24	Prise en charge des messages étendus fragmentés
B23…22	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B21…20	Courant de crête
B19…10	Tension en unités de 50 mV

Bit(s)	Description
B9…0	Courant maximal en unités de 10 mA

6.4.1.2.2.1 Alimentation double fonction

Le bit Dual-Role Power ***Devoir/Doit/Doivent*** être défini lorsque le port est apte à l'alimentation double fonction, c'est-à-dire lorsqu'il prend en charge le message ***PR_Swap***.

Il s'agit d'une capacité statique qui ***Devoir/Doit/Doivent*** rester fixe pour un dispositif donné, quel que soit le rôle d'alimentation actuel du dispositif. Si le bit Dual-Role Power est défini sur 1 dans le message ***Source_Capabilities*** , le bit Dual-Role Power du message ***Sink_Capabilities*** ***Devoir/Doit/Doivent*** également être défini sur 1. Si le bit Dual-Role Power est défini sur 0 dans le message ***Source_Capabilities***, le bit Dual-Role Power du message ***Sink_Capabilities*** ***Devoir/Doit/Doivent*** également être défini sur 0.

6.4.1.2.2.2 Prise en charge de la veille USB

Avant un contrat ou lorsque le bit USB Communications Capable est défini sur 0, ce fanion est indéfini et les destinataires ***Devoir/Doit/Doivent*** suivre les règles de veille définies dans ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** ou ***[USBBC 1.2]***. Après la négociation d'un contrat:

- si le fanion USB Suspend Supported est défini, le destinataire ***Devoir/Doit/Doivent*** alors suivre les règles ***[USB 2.0]*** ou ***[USB 3.2]*** pour la veille et la reprise. Un périphérique PDUSB ***Pouvoir/Peut/Peuvent*** appeler un courant allant jusqu'à ***pSnkSusp*** et un hub PDUSB ***Pouvoir/Peut/Peuvent*** appeler un courant allant jusqu'à ***pHubSusp*** pendant une veille (voir 7.2.3);
- si le fanion USB Suspend Supported est effacé, le destinataire ***Ne doit/doivent pas*** appliquer les règles ***[USB 2.0]*** ou ***[USB 3.2]*** pour la veille, et il ***Pouvoir/Peut/Peuvent*** continuer d'utiliser la puissance négociée. Noter que lorsque l'interface USB est en veille, l'état du dispositif USB est également en veille.

Les destinataires ***Pouvoir/Peut/Peuvent*** indiquer à la source qu'ils préfèrent avoir le fanion USB Suspend Supported effacé en définissant le fanion No USB Suspend d'un message ***Request*** (voir 6.4.2.5).

6.4.1.2.2.3 Puissance non contrainte

Le bit Unconstrained Power ***Devoir/Doit/Doivent*** être activé lorsqu'une source de puissance externe est disponible et suffisante pour alimenter le système de façon appropriée tout en chargeant les dispositifs externes, ou lorsque la fonction primaire de l'appareil consiste à charger des dispositifs externes.

Pour activer le bit Unconstrained Power en fonction du résultat d'une source externe, ***Il convient d'/de/qu'/que*** la source de puissance externe soit:

- une alimentation en courant alternatif (par exemple un bloc d'alimentation) directement connectée au destinataire;
- ou, dans le cas d'un hub PDUSB:
  - une source PD ayant son bit Unconstrained Power défini;
  - plusieurs sources PD ayant toutes leur bit Unconstrained Power défini.

6.4.1.2.2.4 Aptitude aux communications USB

Le bit USB Communications Capable ***Devoir/Doit/Doivent*** uniquement être défini pour les sources aptes à la communication sur les circuits de données USB (par exemple D+/- ou SS Tx/Rx).

6.4.1.2.2.5 Données double fonction

Le bit Dual-Role Data ***Devoir/Doit/Doivent*** être défini lorsque le port est apte aux données double fonction, c'est-à-dire lorsqu'il prend en charge le message ***DR_Swap***. Il s'agit d'une capacité statique qui ***Devoir/Doit/Doivent*** rester fixe pour un dispositif donné, quel que soit le rôle actuel d'alimentation ou de transmission de données du dispositif. Si le bit Dual-Role Data est défini sur 1 dans le message ***Source_Capabilities***, le bit Dual-Role Data du message ***Sink_Capabilities*** ***Devoir/Doit/Doivent*** également

être défini sur 1. Si le bit Dual-Role Data est défini sur 0 dans le message ***Source_Capabilities***, le bit Dual-Role Data du message ***Sink_Capabilities*** ***Devoir/Doit/Doivent*** également être défini sur 0.

6.4.1.2.6 Prise en charge des messages étendus fragmentés

Le bit Unchunked Extended Messages Supported ***Devoir/Doit/Doivent*** être activé lorsque le port peut envoyer et recevoir des messages étendus avec ***Data Size*** > ***MaxExtendedMsgLegacyLen*** octets dans un message unique non fragmenté.

6.4.1.2.7 Courant de crête

L'alimentation électrique fixe par port USB est seulement tenue de fournir la quantité de courant demandée dans le champ Operating Current (IOC) d'un RDO. Cependant, pour certaines utilisations, par exemple dans le cas des systèmes informatiques qui connaissent de brèves pointes d'activité, il peut être souhaitable de surcharger la source d'alimentation pendant de courtes périodes.

Par exemple, lorsqu'un système informatique essaie de maintenir une consommation de puissance moyenne, plus le courant de crête est élevé, plus la période à faible intensité de courant nécessaire pour maintenir une telle puissance moyenne est longue (voir 7.2.8). Le champ Peak Current permet à une source d'alimentation d'annoncer cette capacité supplémentaire. Cette capacité est prévue uniquement pour les connexions directes port à port et elle ***Ne doit/doivent pas*** être offerte à des destinataires en aval par l'intermédiaire d'un hub.

Chacun des PDO d'alimentation électrique fixe ***Devoir/Doit/Doivent*** comporter un champ Peak Current. Les alimentations qui souhaitent offrir un ensemble de capacités de surcharge ***Devoir/Doit/Doivent*** l'annoncer au moyen du champ Peak Current à l'intérieur du PDO d'alimentation électrique fixe correspondant (voir Tableau 6-10). Les alimentations qui ne prennent pas en charge la capacité de surcharge ***Devoir/Doit/Doivent*** définir ces bits sur 00b dans le PDO d'alimentation électrique fixe correspondant. Les alimentations qui prennent en charge une capacité de surcharge étendue spécifiée dans les champs PeakCurrent1…3 du message ***Source_Capabilities_Extended*** (voir 6.5.1) ***Devoir/Doit/Doivent*** également établir ces bits à 00b. Les destinataires qui souhaitent utiliser ces capacités étendues ***Devoir/Doit/Doivent*** tout d'abord envoyer le message ***Get_Source_Cap_Extended*** pour déterminer quelles sont les capacités prises en charge par la source, le cas échéant.

Tableau 6-10 Capacité de courant de crête d'une source d'alimentation fixe

Bits 21…20	Description
00	Courant de crête égal à IOC (valeur par défaut) ou rechercher les capacités de source étendues (envoyer le message ***Get_Source_Cap_Extended***)
01	Capacités de surcharge: 1. Courant de crête égal à 150 % I_{OC} pendant 1 ms à 5 % du cycle de service (bas niveau de courant égal à 97 % I_{OC} pendant 19 ms) 2. Courant de crête égal à 125 % I_{OC} pendant 2 ms à 10 % du cycle de service (bas niveau de courant égal à 97 % I_{OC} pendant 18 ms) 3. Courant de crête égal à 110 % I_{OC} pendant 10 ms à 50 % du cycle de service (bas niveau de courant égal à 90 % I_{OC} pendant 10 ms)
10	Capacités de surcharge: 1. Courant de crête égal à 200 % I_{OC} pendant 1 ms à 5 % du cycle de service (bas niveau de courant égal à 95 % I_{OC} pendant 19 ms) 2. Courant de crête égal à 150 % I_{OC} pendant 2 ms à 10 % du cycle de service (bas niveau de courant égal à 94 % I_{OC} pendant 18 ms) 3. Courant de crête égal à 125 % I_{OC} pendant 10 ms à 50 % du cycle de service (bas niveau de courant égal à 75 % I_{OC} pendant 10 ms)
11	Capacités de surcharge: 1. Courant de crête égal à 200 % I_{OC} pendant 1 ms à 5 % du cycle de service (bas niveau de courant égal à 95 % I_{OC} pendant 19 ms) 2. Courant de crête égal à 175 % I_{OC} pendant 2 ms à 10 % du cycle de service (bas niveau de courant égal à 92 % I_{OC} pendant 18 ms) 3. Courant de crête égal à 150 % I_{OC} pendant 10 ms à 50 % du cycle de service (bas niveau de courant égal à 50 % I_{OC} pendant 10 ms)

6.4.1.2.3 Objet de données d'alimentation électrique variable (batterie exceptée)

Le Tableau 6-11 décrit un PDO d'alimentation électrique variable (batterie exceptée) (10b) pour une source. Voir 7.1.3 pour les exigences électriques applicables à l'alimentation.

Les champs de tension ***Devoir/Doit/Doivent*** définir la plage dans laquelle la tension de sortie ***Devoir/Doit/Doivent*** se trouver. Cela n'indique pas la tension qui est effectivement fournie, mais seulement qu'elle ***Devoir/Doit/Doivent*** se trouver dans cette plage. La tension absolue, y compris toute variation de tension, ***Ne doit/doivent pas*** descendre au-dessous de la tension minimale et ***Ne doit/doivent pas*** dépasser la tension maximale.

Tableau 6-11 PDO d'alimentation électrique variable (batterie exceptée) - Source

Bit(s)	Description
B31…30	Alimentation variable (batterie exceptée)
B29…20	Tension maximale en unités de 50 mV
B19…10	Tension minimale en unités de 50 mV
B9…0	Courant maximal en unités de 10 mA

6.4.1.2.4 Objet de données d'alimentation électrique par batterie

Le Tableau 6-12 décrit un PDO d'alimentation électrique par batterie (01b) pour une source. Voir 7.1.3 pour les exigences électriques applicables à l'alimentation.

Les champs de tension ***Devoir/Doit/Doivent*** représenter la plage de tensions de la batterie. La batterie ***Devoir/Doit/Doivent*** être capable de fournir la valeur de puissance sur la totalité de la plage de tensions. La tension absolue, y compris toute variation de tension, ***Ne doit/doivent pas*** descendre au-dessous de la tension minimale et ***Ne doit/doivent pas*** dépasser la tension maximale. Noter que seul le PDO de batterie utilise la puissance à la place du courant.

Le destinataire ***Pouvoir/Peut/Peuvent*** surveiller la tension de la batterie.

Tableau 6-12 PDO d'alimentation électrique par batterie - Source

Bit(s)	Description
B31…30	Batterie
B29…20	Tension maximale en unités de 50 mV
B19…10	Tension minimale en unités de 50 mV
B9…0	Puissance maximale admissible en unités de 250 mW

6.4.1.2.5 Objet de données d'alimentation augmentée par une alimentation électrique programmable

Le Tableau 6-13 ci-dessous décrit un APDO d'alimentation électrique programmable (1100b) pour une source. Voir 7.1.3 pour les exigences électriques applicables à l'alimentation. Cet APDO est utilisé essentiellement pour la charge commandée par le destinataire d'une batterie au sein du destinataire. Lorsqu'un courant supérieur à celui supporté par le câble est appliqué à la batterie, un démultiplicateur fixe à haut rendement ***Pouvoir/Peut/Peuvent*** être utilisé au sein du destinataire pour réduire le courant dans le câble.

Les champs de tension définissent la plage de tension de sortie sur laquelle l'alimentation électrique ***Devoir/Doit/Doivent*** être réglable par pas de 20 mV. Le champ Maximum Current contient l'intensité de courant que l'alimentation électrique programmable ***Devoir/Doit/Doivent*** être capable de fournir sur la plage de tension annoncée.

Tableau 6-13 APDO d'alimentation électrique programmable - Source

Bit(s)	Description
B31…30	11b – Objet de données d'alimentation augmentée (APDO)
B29…28	00b – Alimentation électrique programmable

Bit(s)	Description
	01b…11b - ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé
B27	PPS Power Limited
B26…25	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B24…17	Tension maximale par pas de 100 mV
B16	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B15…8	Tension minimale par pas de 100 mV
B7	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B6…0	Courant maximal par pas de 50 mA

6.4.1.2.5.1 PPS Power Limited

Lorsque le bit PPS Power Limited est défini, la source PPS ***Ne doit/doivent pas*** fournir une puissance qui dépasse la PDP assignée de la source; si la tension de sortie demandée au RDO dépasse la tension Prog nominale (par exemple 5 V pour 5 VProg), la PPS ***Devoir/Doit/Doivent*** limiter son courant de sortie de telle sorte que, pour une PDP assignée de x Watts, la limite du courant de sortie limite soit égale à RoundDown(x/tension de sortie demandée) arrondie à 50 mA. La source PPS ***Ne doit/doivent pas*** rejeter un RDO avec un courant de sortie inférieur ou égal au courant maximal de l'APDO, même si le courant de sortie demandé est supérieur à la limite de courant de la source. Une source PPS qui définit le bit Power Output Limited ***Devoir/Doit/Doivent*** limiter automatiquement son courant de sortie de manière à ne pas dépasser sa PDP assignée (voir Figure 7-7).

Lorsque le bit PPS Power Limited est effacé, la source PPS ***Devoir/Doit/Doivent*** délivrer le courant maximal jusqu'à la tension maximale annoncée dans son APDO.

6.4.1.3 Message de capacités du destinataire

Un port destinataire ***Devoir/Doit/Doivent*** signaler les niveaux de puissance auxquels il est apte à fonctionner dans une série d'objets de données d'alimentation de 32 bits (voir Tableau 6-7). Ces derniers sont renvoyés au sein d'un message ***Sink_Capabilities*** en réponse à un message ***Get_Sink_Cap*** (voir Figure 6-12). Cela est semblable à la procédure utilisée pour les capacités d'un port source avec des objets de données d'alimentation équivalents pour les alimentations électriques fixe, variable et sur batterie, comme défini dans la présente section. Les objets de données d'alimentation sont utilisés pour transmettre les exigences de puissance opérationnelle du port destinataire, y compris les ports d'alimentation double fonction fonctionnant actuellement en tant que source.

Chaque objet de données d'alimentation ***Devoir/Doit/Doivent*** décrire un niveau de puissance opérationnelle spécifique pour le destinataire, tel qu'une alimentation par batterie (par exemple 2,8-4,1 V) ou une alimentation électrique fixe (par exemple 12 V). Le champ ***Number of Data Objects*** dans l'en-tête de message ***Devoir/Doit/Doivent*** définir le nombre d'objets de données d'alimentation qui suivent l'en-tête de message d'un message de données.

Tous les destinataires ***Devoir/Doit/Doivent*** offrir au minimum un objet de données d'alimentation avec un niveau de puissance auquel le destinataire peut fonctionner. Un destinataire ***Ne doit/doivent pas*** offrir de multiples objets de données d'alimentation de même type (fixe, variable, batterie) et de même tension, mais il ***Devoir/Doit/Doivent*** à la place offrir un objet de données d'alimentation ayant l'intensité de courant disponible la plus élevée pour cette capacité de destinataire et cette tension.

Tous les destinataires ***Devoir/Doit/Doivent*** comporter un objet de données d'alimentation qui signale ***vSafe5V*** même s'ils ont besoin de puissance supplémentaire pour fonctionner pleinement. Dans le cas où une puissance supplémentaire est nécessaire pour fonctionner pleinement, le bit Higher Capability ***Devoir/Doit/Doivent*** être défini.

6.4.1.3.1 Objet de données d'alimentation électrique fixe pour le destinataire

Le Tableau 6-14 décrit le PDO d'alimentation électrique fixe pour le destinataire (00b). Voir 7.1.3 pour les exigences électriques applicables à l'alimentation. Le destinataire ***Devoir/Doit/Doivent*** définir la tension et le courant opérationnel aux valeurs exigées. Le courant opérationnel exigé est défini par l'intensité de courant nécessaire à un dispositif donné pour qu'il soit fonctionnel. Cette valeur peut être l'intensité de

courant maximale que le destinataire exigera éventuellement ou la valeur suffisante pour faire fonctionner le destinataire dans l'un de ses modes de fonctionnement.

Puisque tous les consommateurs USB prennent en charge ***vSafe5V***, l'objet de données d'alimentation électrique fixe de ***vSafe5V*** exigé est également utilisé pour acheminer des informations supplémentaires, renvoyées dans les bits 29 à 20. Tous les autres objets de données d'alimentation électrique fixe ***Devoir/Doit/Doivent*** avoir les bits 29…20 mis à zéro.

Pour un destinataire qui ne demande pas de puissance à la source, la tension (B19…10) ***Devoir/Doit/Doivent*** être réglée à 5 V et le courant de fonctionnement ***Devoir/Doit/Doivent*** être réglée à 0 mA.

Tableau 6-14 PDO d'alimentation électrique fixe - Destinataire

<table>
<tr><th>Bit(s)</th><th>Description</th></tr>
<tr><td>B31…30</td><td>Alimentation fixe</td></tr>
<tr><td>B29</td><td>Alimentation double fonction</td></tr>
<tr><td>B28</td><td>Capacité supérieure</td></tr>
<tr><td>B27</td><td>Puissance non contrainte</td></tr>
<tr><td>B26</td><td>Aptitude aux communications USB</td></tr>
<tr><td>B25</td><td>Données double fonction</td></tr>
<tr><td>B24…23</td><td>Intensité de courant USB Type-C exigée par la permutation rapide des rôles (voir aussi [USB Type-C 2.0]):
<table>
<tr><th>Valeur</th><th>Description</th></tr>
<tr><td>00b</td><td>Permutation rapide non prise en charge (par défaut)</td></tr>
<tr><td>01b</td><td>Puissance USB par défaut</td></tr>
<tr><td>10b</td><td>1,5 A @ 5 V</td></tr>
<tr><td>11b</td><td>3,0 A @ 5 V</td></tr>
</table></td></tr>
<tr><td>B22…20</td><td>Réservé(s/ée/ées) – Devoir/Doit/Doivent être défini sur 0.</td></tr>
<tr><td>B19…10</td><td>Tension en unités de 50 mV</td></tr>
<tr><td>B9…0</td><td>Courant opérationnel en unités de 10 mA</td></tr>
</table>

6.4.1.3.1.1 Alimentation double fonction

Le bit Dual-Role Power ***Devoir/Doit/Doivent*** être défini lorsque le port est apte à l'alimentation double fonction, c'est-à-dire lorsqu'il prend en charge le message ***PR_Swap***. Il s'agit d'une capacité statique qui ***Devoir/Doit/Doivent*** rester fixe pour un dispositif donné, quel que soit le rôle d'alimentation actuel du dispositif. Si le bit Dual-Role Power est défini sur 1 dans le message ***Source_Capabilities***, le bit Dual-Role Power du message ***Sink_Capabilities*** ***Devoir/Doit/Doivent*** également être défini sur 1. Si le bit Dual-Role Power est défini sur 0 dans le message ***Sink_Capabilities***, le bit Dual-Role Power du message ***Sink_Capabilities*** ***Devoir/Doit/Doivent*** également être défini sur 0.

6.4.1.3.1.2 Capacité supérieure

Si le destinataire a besoin de plus que ***vSafe5V*** (par exemple 12 V) pour assurer une fonctionnalité complète, le bit Higher Capability ***Devoir/Doit/Doivent*** être défini.

6.4.1.3.1.3 Puissance non contrainte

Le bit Unconstrained Power ***Devoir/Doit/Doivent*** être activé lorsqu'une source de puissance externe est disponible et suffisante pour alimenter le système de façon appropriée tout en chargeant les dispositifs externes, ou lorsque la fonction primaire de l'appareil consiste à charger des dispositifs externes.

Pour activer le bit Unconstrained Power en fonction du résultat d'une source externe, ***Il convient d'/de/qu'/que*** la source de puissance externe soit:

- une alimentation en courant alternatif (par exemple un bloc d'alimentation) directement connectée au destinataire;

- ou, dans le cas d'un hub PDUSB:
  - une source PD ayant son bit Unconstrained Power défini;
  - plusieurs sources PD ayant toutes leur bit Unconstrained Power défini.

6.4.1.3.1.4 Aptitude aux communications USB

Le bit USB Communications Capable ***Devoir/Doit/Doivent*** uniquement être défini pour les destinataires aptes à la communication sur les circuits de données USB (par exemple D+/- ou SS Tx/Rx).

6.4.1.3.1.5 Données double fonction

Le bit Dual-Role Data ***Devoir/Doit/Doivent*** être défini lorsque le port est apte aux données double fonction, c'est-à-dire lorsqu'il prend en charge le message ***DR_Swap***. Il s'agit d'une capacité statique qui ***Devoir/Doit/Doivent*** rester fixe pour un dispositif donné, quel que soit le rôle actuel d'alimentation ou de transmission de données du dispositif. Si le bit Dual-Role Data est défini sur 1 dans le message ***Source_Capabilities***, le bit Dual-Role Data du message ***Sink_Capabilities*** ***Devoir/Doit/Doivent*** également être défini sur 1. Si le bit Dual-Role Data est défini sur 0 dans le message ***Source_Capabilities*** , le bit Dual-Role Data du message ***Sink_Capabilities*** ***Devoir/Doit/Doivent*** également être défini sur 0.

6.4.1.3.1.6 Intensité de courant USB Type-C pour la permutation rapide des rôles

Le champ Fast Role Swap USB Type-C Current ***Devoir/Doit/Doivent*** indiquer l'intensité de courant exigée par le destinataire après une permutation rapide des rôles.

La source initiale ***Ne doit/doivent pas*** transmettre un signal de permutation rapide des rôles si le champ Fast Role Swap USB Type-C Current est défini sur 0.

Initialement, lorsque la nouvelle source applique ***vSafe5V***, elle a Rd affirmée, mais elle ***Devoir/Doit/Doivent*** fournir l'intensité de courant USB Type-C indiqué par le nouveau destinataire dans ce champ. Si la nouvelle source n'est pas en mesure de fournir ce niveau de courant, elle ***Ne doit/doivent pas*** effectuer de permutation rapide des rôles. Lorsque Rp est affirmée par la nouvelle source pendant l'AMS de permutation rapide des rôles (voir 6.3.19), la valeur de l'intensité de courant USB Type-C indiquée par Rp ***Devoir/Doit/Doivent*** être supérieure ou égale à celle qui est indiquée dans le champ Fast Role Swap USB Type-C Current.

6.4.1.3.2 Objet de données d'alimentation électrique variable (batterie exceptée)

Le Tableau 6-15 décrit un PDO d'alimentation électrique variable (batterie exceptée) (10b) utilisé par un destinataire. Voir 7.1.3 pour les exigences électriques applicables à l'alimentation.

Les champs de tension ***Devoir/Doit/Doivent*** être définis sur la plage de tensions de sortie dont le destinataire a besoin pour fonctionner. Le champ Operational Current ***Devoir/Doit/Doivent*** être défini sur la valeur de l'intensité de courant opérationnel exigée par le destinataire, sur la plage de tension donnée. La tension absolue, y compris toute variation de tension, ***Ne doit/doivent pas*** descendre au-dessous de la tension minimale et ***Ne doit/doivent pas*** dépasser la tension maximale. Le courant opérationnel exigé est défini par l'intensité de courant nécessaire à un dispositif donné pour qu'il soit fonctionnel. Cette valeur peut être l'intensité de courant maximale que le destinataire exigera éventuellement ou la valeur suffisante pour faire fonctionner le destinataire dans l'un de ses modes de fonctionnement.

Tableau 6-15 PDO d'alimentation électrique variable (batterie exceptée) - Destinataire

Bit(s)	Description
B31…30	Alimentation variable (batterie exceptée)
B29…20	Tension maximale en unités de 50 mV
B19…10	Tension minimale en unités de 50 mV
B9…0	Courant opérationnel en unités de 10 mA

6.4.1.3.3 Objet de données d'alimentation électrique par batterie

Le Tableau 6-16 décrit un PDO de batterie (01b) utilisé par un destinataire. Voir 7.1.3 pour les exigences électriques applicables à l'alimentation.

Les champs de tension ***Devoir/Doit/Doivent*** être définis sur la plage de tensions de sortie dont le destinataire a besoin pour fonctionner. Le champ Operational Power ***Devoir/Doit/Doivent*** être défini sur la valeur de la puissance opérationnelle exigée par le destinataire, sur la plage de tension donnée. La tension absolue, y compris toute variation de tension, ***Ne doit/doivent pas*** descendre au-dessous de la tension minimale et ***Ne doit/doivent pas*** dépasser la tension maximale. Noter que seul le PDO de batterie utilise la puissance à la place du courant. La puissance opérationnelle exigée est définie par la quantité de puissance nécessaire à un dispositif donné pour qu'il soit fonctionnel. Cette valeur peut être la puissance maximale que le destinataire exigera éventuellement ou la valeur suffisante pour faire fonctionner le destinataire dans l'un de ses modes de fonctionnement.

Tableau 6-16 PDO d'alimentation par batterie - Destinataire

Bit(s)	Description
B31…30	Batterie
B29…20	Tension maximale en unités de 50 mV
B19…10	Tension minimale en unités de 50 mV
B9…0	Puissance opérationnelle en unités de 250 mW

6.4.1.3.4 Objet de données d'alimentation augmentée par une alimentation électrique programmable

Le Tableau 6-17 ci-dessous décrit un APDO d'alimentation électrique programmable (1100b) utilisé par un destinataire. Voir 7.1.3 pour les exigences électriques applicables à l'alimentation.

Les champs Maximum Voltage et Minimum Voltage ***Devoir/Doit/Doivent*** être définis sur la plage de tensions de sortie dont le destinataire a besoin pour fonctionner. Le champ Operational Current ***Devoir/Doit/Doivent*** être défini sur l'intensité de courant maximale exigée par le destinataire, sur la plage de tension. Le courant opérationnel est défini comme la quantité maximale de courant nécessaire à l'appareil pour assurer pleinement sa fonction (par exemple la charge commandée par le destinataire).

Tableau 6-17 APDO d'alimentation électrique programmable - Destinataire

Bit(s)	Description
B31…30	11b – Objet de données d'alimentation augmentée (APDO)
B29…28	00b – Alimentation électrique programmable
B27…25	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B24…17	Tension maximale par pas de 100 mV
B16	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B15…8	Tension minimale par pas de 100 mV
B7	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B6…0	Courant maximal par pas de 50 mA

6.4.2 Message de demande

Un récepteur ***Devoir/Doit/Doivent*** envoyer un message ***Request*** pour réclamer de la puissance, généralement lors de la phase de demande d'une négociation de puissance. L'objet de données de demande ***Devoir/Doit/Doivent*** être renvoyé par le destinataire qui émet la demande de puissance. Il ***Devoir/Doit/Doivent*** être envoyé en réponse au message ***Source_Capabilities*** le plus récent (voir 8.3.2.2). Un message ***Request*** ***Devoir/Doit/Doivent*** renvoyer un seul et unique objet de données de demande de destinataire, qui ***Devoir/Doit/Doivent*** identifier l'objet de données d'alimentation demandé.

Le message ***Request*** inclut le niveau de puissance demandé. Par exemple, si le message ***Source_Capabilities*** inclut un PDO d'alimentation fixe qui fournit 12 V @ 1,5 A, et que le destinataire ne demande que 12 V @0,5 A, il définira le champ Operating Current sur 50 (c'est-à-dire 10 mA * 50 = 0,5 A).

Le message ***Request*** demande, dans le champ Maximum Operating Current, l'intensité de courant la plus élevée que le destinataire exige (dans cet exemple, la valeur est 100 (100 * 10 mA = 1,0 A)).

La demande prend différentes formes en fonction du type d'alimentation demandé. L'objet de données d'alimentation fixe et l'objet de données d'alimentation variable partagent un format commun, représenté dans le Tableau 6-18 et le Tableau 6-19. L'objet de données d'alimentation de batterie utilise le format indiqué dans le Tableau 6-20 et le Tableau 6-21. L'objet de demande programmable est au format indiqué dans le Tableau 6-22.

Tableau 6-18 Objet de données de demande fixe et variable

Bits	Description
B31	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B30…28	Position de l'objet (000b est ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisé)
B27	Fanion GiveBack = 0
B26	Incohérence de capacité
B25	Aptitude aux communications USB
B24	Pas de veille USB
B23	Prise en charge des messages étendus fragmentés
B22…20	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B19…10	Courant de fonctionnement en unités de 10 mA
B9…0	Intensité de courant de fonctionnement maximale en unités de 10 mA

Tableau 6-19 Objets de données de demande fixe et variable avec prise en charge de GiveBack

Bits	Description
B31	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B30…28	Position de l'objet (000b est ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisé)
B27	Fanion GiveBack = 1
B26	Incohérence de capacité
B25	Aptitude aux communications USB
B24	Pas de veille USB
B23	Prise en charge des messages étendus fragmentés
B22…20	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B19…10	Courant de fonctionnement en unités de 10 mA
B9…0	Courant de fonctionnement minimal en unités de 10 mA

Tableau 6-20 Objets de données de demande de batterie

Bits	Description
B31	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B30…28	Position de l'objet (000b est ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisé)
B27	GiveBackFlag = 0
B26	Incohérence de capacité
B25	Aptitude aux communications USB
B24	Pas de veille USB
B23	Prise en charge des messages étendus fragmentés
B22…20	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B19…10	Puissance de fonctionnement en unités de 250 mW
B9…0	Puissance de fonctionnement maximale en unités de 250 mW

Tableau 6-21 Objet de données de demande de batterie avec prise en charge de GiveBack

Bits	Description
B31	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B30…28	Position de l'objet (000b est ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas***être utilisé)
B27	GiveBackFlag = 1
B26	Incohérence de capacité
B25	Aptitude aux communications USB
B24	Pas de veille USB
B23	Prise en charge des messages étendus fragmentés
B22…20	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B19…10	Puissance de fonctionnement en unités de 250 mW
B9…0	Puissance de fonctionnement minimale en unités de 250 mW

Tableau 6-22 Objet de données de demande programmable

Bits	Description
B31	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B30…28	Position de l'objet (000b est ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas***être utilisé)
B27	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B26	Incohérence de capacité
B25	Aptitude aux communications USB
B24	Pas de veille USB
B23	Prise en charge des messages étendus fragmentés
B22…20	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B19…9	Tension de sortie en unités de 20 mV
B8…7	***Réservé(s/ée/ées)*** – ***Devoir/Doit/Doivent*** être défini sur 0.
B6…0	Courant d'exploitation en unités de 50 mA

6.4.2.1 Position de l'objet

La valeur du champ Object Position ***Devoir/Doit/Doivent*** indiquer à quel objet du message ***Source_Capabilities*** se réfère le RDO. La valeur 1 indique toujours le PDO d'alimentation fixe de 5 V, en tant que premier objet suivant l'en-tête de message ***Source_Capabilities***. La valeur 2 fait référence au PDO suivant, etc.

6.4.2.2 Fanion GiveBack

Le fanion GiveBack ***Devoir/Doit/Doivent*** être défini de manière à indiquer que le destinataire répond à un message ***GotoMin*** en réduisant sa charge au courant de fonctionnement minimal. Il est généralement utilisé par un dispositif USB pendant la charge de sa batterie, car une courte interruption de la charge n'a qu'un impact mineur sur l'utilisateur et permet à la source de mieux gérer sa charge.

6.4.2.3 Incohérence de capacité

Une incohérence de capacité se produit lorsque les capacités offertes par la source ne peuvent satisfaire aux exigences de puissance du destinataire. Dans ce cas, le destinataire ***Devoir/Doit/Doivent*** émettre une demande ***Valide(s)*** à partir des capacités offertes et ***Devoir/Doit/Doivent*** définir le bit Capability Mismatch (voir 8.2.5.2).

Lorsque le destinataire renvoie un objet de données de demande, avec ce bit défini, en réponse aux capacités annoncées, il indique qu'il souhaite disposer d'une puissance que la source n'est pas en mesure de fournir. La raison peut être une tension indisponible ou une intensité de courant insuffisante. A ce stade, la source peut utiliser les informations du message de ***Request*** conjointement avec le contenu du message ***Sink_Capabilities*** pour déterminer la tension et l'intensité de courant exigées par le destinataire pour fonctionner de manière optimale.

Dans ce contexte, un message ***Request Valide(s)*** signifie que:

- le champ Object position ***Devoir/Doit/Doivent*** contenir une référence vers un objet du dernier message ***Source_Capabilities*** reçu;
- le champ Operating Current/Power ***Devoir/Doit/Doivent*** contenir une valeur inférieure ou égale à l'intensité de courant/la puissance maximale offerte dans le message ***Source_Capabilities***;
- si le fanion GiveBack est défini sur 0, c'est-à-dire que le champ Maximum Operating Current/Power est renseigné:
  - si le bit Capability Mismatch est défini sur 1:
    - le champ Maximum Operating Current/Power ***Pouvoir/Peut/Peuvent*** contenir une valeur supérieure à celle de l'intensité de courant/la puissance maximale fournie dans le PDO du message ***Source_Capabilities***, référencé par le champ Object position. Cela permet au destinataire d'indiquer qu'il exige davantage de courant/puissance que ce qui lui est offert. Si le destinataire exige une tension différente, cette information est indiquée par son message ***Sink_Capabilities***;
  - sinon, si le bit Capability Mismatch est défini sur 0:
    - le champ Maximum Operating Current/Power ***Devoir/Doit/Doivent*** contenir une valeur inférieure ou égale à celle de l'intensité de courant/la puissance maximale fournie dans le PDO du message ***Source_Capabilities***, référencé par le champ Object position;
- sinon, si le fanion GiveBack est défini sur 1, c'est-à-dire que le champ Minimum Operating Current/Power est renseigné:
  - le champ Minimum Operating Current/Power ***Devoir/Doit/Doivent*** contenir une valeur inférieure à celle du champ Operating Current/Power.

6.4.2.4 Aptitude aux communications USB

Le fanion USB Communications Capable ***Devoir/Doit/Doivent*** être défini sur 1 lorsque le récepteur dispose de circuits de données USB et qu'il est capable de communiquer en utilisant le protocole ***[USB 2.0]*** ou le protocole ***[USB 3.2]***. Ce fanion ***Devoir/Doit/Doivent*** être défini sur 0 lorsque le récepteur ne dispose pas de circuits de données USB ou qu'il n'est pas capable de communiquer en utilisant le protocole ***[USB 2.0]*** ou le protocole ***[USB 3.2]***. Cela permet à la source de déterminer le fonctionnement dans certains cas tels que la veille USB. Si le destinataire a défini le fanion USB Communications Capable sur 0, il est nécessaire que la source sache que les règles de veille USB ne peuvent pas être respectées par le destinataire.

6.4.2.5 Pas de veille USB

Le fanion No USB Suspend ***Pouvoir/Peut/Peuvent*** être défini par le destinataire pour indiquer à la source que ce dispositif demande à poursuivre son contrat lors de la veille USB. Les destinataires qui définissent ce fanion disposent généralement de fonctionnalités qui peuvent utiliser l'alimentation à d'autres fins qu'une communication USB, par exemple pour charger une batterie.

La source utilise ce fanion pour déterminer s'***Il convient d'/de/qu'/que*** elle émette à nouveau le message ***Source_Capabilities***, avec le fanion USB Suspend effacé.

6.4.2.6 Prise en charge des messages étendus fragmentés

Le bit Unchunked Extended Messages Supported ***Devoir/Doit/Doivent*** être activé lorsque le port peut envoyer et recevoir des messages étendus avec ***Data Size*** > ***MaxExtendedMsgLegacyLen*** octets dans un message unique non fragmenté.

6.4.2.7 Courant de fonctionnement

Le champ Operating Current de l'objet de données de demande ***Devoir/Doit/Doivent*** être défini sur l'intensité de courant effective nécessaire au destinataire pour fonctionner à un instant donné. Un nouveau message ***Request***, avec une valeur d'intensité de courant de fonctionnement mise à jour, ***Devoir/Doit/Doivent*** être émis dès lors qu'il est nécessaire de modifier la puissance du destinataire, par exemple, du courant de fonctionnement maximal à une intensité de courant inférieure. Conjointement

avec le champ Maximum Operating Current ou le Minimum Operating Current, il fournit à la source des informations supplémentaires lui permettant de mieux gérer la distribution de puissance.

Le destinataire utilise également le champ Operating Current de l'objet de données de demande programmable pour demander à la source le niveau de la limite de courant nécessaire. Lorsque la demande est acceptée, le courant de sortie fourni par la source dans n'importe quelle charge ***Devoir/Doit/Doivent*** être inférieur ou égal au courant de fonctionnement. Lorsque le destinataire tente de consommer davantage de courant, la source ***Devoir/Doit/Doivent*** réduire la tension de sortie de manière à ne pas dépasser la valeur du courant de fonctionnement.

La valeur du champ Operating Current ***Ne doit/doivent pas*** dépasser la valeur du champ Maximum Current.

Ce champ ***Devoir/Doit/Doivent*** s'appliquer aux RDO fixe, variable et programmable.

6.4.2.8 Courant de fonctionnement maximal

Le champ Maximum Operating Current du message ***Request*** ***Devoir/Doit/Doivent*** avoir pour valeur l'intensité de courant la plus élevée que le destinataire exigera éventuellement. La différence entre les champs Operating Current et Maximum Operating Current (lorsque le fanion GiveBack est effacé) est utilisée par le gestionnaire de politique d'utilisation des dispositifs de la source pour calculer la taille à maintenir de la réserve de puissance (voir 8.2.5.1). La valeur du champ Operating Current ***Devoir/Doit/Doivent*** être inférieure ou égale à la valeur de l'intensité de courant de fonctionnement maximale.

Lorsque le bit Capability Mismatch est défini sur 0, l'intensité de courant de fonctionnement maximale demandée ***Devoir/Doit/Doivent*** être inférieure ou égale au courant offert par les capacités de la source, car il est nécessaire que cette dernière réserve cette puissance pour une utilisation ultérieure. Le champ Maximum Operating Current ***Devoir/Doit/Doivent*** continuer d'être défini sur l'intensité de courant nécessaire la plus élevée, afin de maintenir l'allocation de la réserve de puissance. Si l'intensité de courant de fonctionnement maximale est demandée alors que la réserve de puissance est utilisée par un dispositif compatible GotoMin, un message ***Wait*** est envoyé pour permettre à la source de réclamer le courant supplémentaire (voir 6.3.12.1 et 8.2.5.1).

Lorsque le bit Capability Mismatch est défini sur 1, l'intensité de courant de fonctionnement maximale demandée ***Pouvoir/Peut/Peuvent*** être supérieure au courant offert par les capacités de la source, car cette information est nécessaire à cette dernière pour déterminer les besoins effectifs du destinataire.

Pour plus d'informations sur l'utilisation du bit Capability Mismatch, voir 6.4.2.3.

Ce champ ***Devoir/Doit/Doivent*** s'appliquer aux RDO fixe et variable.

6.4.2.9 Courant de fonctionnement minimal

Le champ Minimum Operating Current du message ***Request*** ***Devoir/Doit/Doivent*** être défini sur l'intensité de courant la plus faible exigée par le destinataire pour fonctionner. La différence entre les champs Operating Current et Minimum Operating Current (lorsque le fanion GiveBack est défini) est utilisée par le gestionnaire de politique d'utilisation des dispositifs pour calculer la quantité de puissance qui peut être réclamée au moyen d'un message ***GotoMin***. La valeur du champ Operating Current ***Devoir/Doit/Doivent*** être supérieure à la valeur de l'intensité de courant de fonctionnement minimale.

Ce champ ***Devoir/Doit/Doivent*** s'appliquer aux RDO fixe et variable.

6.4.2.10 Puissance d'exploitation

Le champ Operating Power de l'objet de données de demande ***Devoir/Doit/Doivent*** être défini sur la quantité de puissance effective souhaitée par le destinataire à cet instant. Conjointement avec le champ Maximum Operating Power, il fournit à la source des informations supplémentaires lui permettant de mieux gérer la distribution de puissance.

Ce champ ***Devoir/Doit/Doivent*** s'appliquer aux RDO de batterie.

6.4.2.11 Puissance d'exploitation maximale

Le champ Maximum Operating Power du message ***Request*** ***Devoir/Doit/Doivent*** avoir pour valeur la puissance la plus élevée que le destinataire exigera éventuellement. Cela permet à une source fournissant une alimentation partagée entre plusieurs ports de répartir intelligemment la puissance.

Lorsque le bit Capabilities Mismatch est défini sur 0, la puissance de fonctionnement maximale demandée ***Devoir/Doit/Doivent*** être inférieure ou égale à la puissance offerte par les capacités de la source, car il est nécessaire que cette dernière réserve cette puissance pour une utilisation ultérieure. Le champ Maximum Operating Power ***Devoir/Doit/Doivent*** continuer d'être défini sur la puissance nécessaire la plus élevée, afin de maintenir l'allocation de la réserve de puissance. Si la puissance de fonctionnement maximale est demandée alors que la réserve de puissance est utilisée par un dispositif compatible GotoMin, un message ***Wait*** est envoyé pour permettre à la source de réclamer la puissance supplémentaire (voir 6.3.12.1 et 8.2.5.1).

Lorsque le bit Capabilities Mismatch est défini sur 1, la puissance de fonctionnement maximale demandée ***Pouvoir/Peut/Peuvent*** être supérieure à la puissance offerte par les capacités de la source, car cette information est nécessaire à cette dernière pour déterminer les besoins effectifs du destinataire.

Pour plus d'informations sur l'utilisation du bit Capability Mismatch, voir 6.4.2.3.

Ce champ ***Devoir/Doit/Doivent*** s'appliquer aux RDO de batterie.

6.4.2.12 Puissance d'exploitation minimale

Le champ Minimum Operating Power du message ***Request*** ***Devoir/Doit/Doivent*** être défini sur la puissance la plus faible exigée par le destinataire pour fonctionner. Conjointement avec le champ Operating Power, il fournit à une source fournissant une alimentation partagée entre plusieurs ports des informations relatives à la quantité de puissance qu'elle peut temporairement se rapproprier de manière à répartir intelligemment la puissance.

Ce champ ***Devoir/Doit/Doivent*** s'appliquer aux RDO de batterie.

6.4.2.13 Tension de sortie

Le destinataire ***Devoir/Doit/Doivent*** définir le champ Output Voltage de l'objet de données de demande programmable sur la tension qu'il exige, mesurée au niveau du connecteur de sortie de la source. La valeur du champ Output Voltage ***Devoir/Doit/Doivent*** être supérieure ou égale à celle du champ Minimum Voltage et inférieure ou égale à celle du champ Maximum Voltage de l'APDO d'alimentation électrique programmable.

Ce champ ***Devoir/Doit/Doivent*** s'appliquer aux RDO programmable.

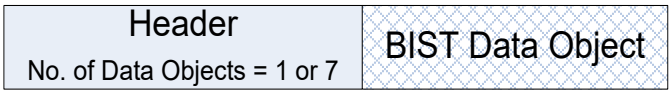
6.4.3 Message BIST

Le message ***BIST*** est envoyé pour demander au port d'entrer dans un mode d'essai de couche physique (voir 5.9) qui assure l'une des fonctions suivantes:

- entrer dans un mode BIST continu pour envoyer un flux continu de données d'essai au dispositif d'essai;
- entrer et quitter le mode d'essai d'un groupe à capacité partagée.

Le format de message est le suivant:

Figure 6-13 Message BIST



Anglais	Français
Header	En-tête

No. of Data Objects	Nombre d'objets de données
BIST Data Object	Objet de données BIST

Tous les ports ***Devoir/Doit/Doivent*** uniquement être en mesure d'être une unité à l'essai (UUT) lors d'un fonctionnement à ***vSafe5V***. Tous les modes BIST suivants ***Devoir/Doit/Doivent*** être pris en charge:

- traitement de la réception d'un objet de données BIST ***Mode porteur BIST***, dont le résultat ***Devoir/Doit/Doivent*** être la génération du signal porteur approprié;
- traitement de la réception d'un objet de données BIST ***BIST Test Data*** dont le résultat ***Devoir/Doit/Doivent*** être que le message est ***Ignoré(s/ée/ées)***.

Les UUT dont les ports constituent un groupe à capacité partagée (voir ***[USB Type-C 2.0]Devoir/Doit/Doivent***) prennent en charge le mode BIST suivant:

- traitement de la réception d'un objet de données BIST ***BIST Shared Test Mode Entry*** qui ***Devoir/Doit/Doivent*** faire entrer l'UUT en mode d'essai BIST de capacité partagée, un mode dans lequel l'UUT offre ses capacités de source complètes sur chaque port du groupe à capacité partagée;
- traitement de la réception d'un objet de données BIST ***BIST Shared Test Mode Exit*** qui ***Devoir/Doit/Doivent*** amener l'UUT à quitter le mode d'essai à capacité partagée.

Lorsqu'un port reçoit un objet de données BIST de message ***BIST*** pour un mode BIST dans lequel le port ne fonctionne pas à ***vSafe5V***, le message ***BIST Devoir/Doit/Doivent*** être ***Ignoré(s/ée/ées)***.

Lorsqu'un port reçoit un objet de données BIST de message ***BIST*** pour un mode BIST qu'il ne prend pas en charge, le message ***BIST Devoir/Doit/Doivent*** être ***Ignoré(s/ée/ées)***.

Lorsqu'un port ou une fiche de câble reçoit un objet de données BIST de message ***BIST*** pour un mode BIST continu, le port ou la fiche de câble entre dans le mode BIST demandé et ***Devoir/Doit/Doivent*** rester dans ce mode BIST pendant ***tBISTContMode***, puis ***Devoir/Doit/Doivent*** retourner à son fonctionnement normal (voir 6.6.7.2).

Le modèle d'utilisation des modes BIST de la couche PHY admet généralement par hypothèse qu'un agent de contrôle demande de soumettre son port partenaire à l'essai.

La Section 8.3.2.14 décrit une séquence d'essais utilisée pour attester de la conformité.

Les champs de l'objet de données BIST sont définis dans le Tableau 6-23.

Tableau 6-23 Objet de données BIST

Bit(s)	Valeur	Paramètre	Description	Référence	Applicabilité
B31...28	0000b...0100b	Réservé***(s/ée/ées)***	***Ne doit/doivent pas*** être utilisé	Section 1.4.2.10	-
	0101b	***Mode porteur BIST***	Demande à l'émetteur d'entrer en mode porteur BIST	Section 6.4.3.1	Obligatoire
	0110b...0111b	Réservé***(s/ée/ées)***	***Ne doit/doivent pas*** être utilisé	Section 1.4.2.10	-
	1000b	***BIST Test Data***	Envoie une trame de données d'essai	Section 6.4.3.2	Obligatoire
	1001b	***BIST Shared Test Mode Entry***	Demande à l'UUT d'entrer en mode d'essai de capacité partagée.		Obligatoire pour les UUT à capacité partagée
	1010b	***BIST Shared Test Mode Exit***	Demande à l'UUT de quitter le mode d'essai de capacité partagée.		Obligatoire pour les UUT à capacité partagée
	1011b...1111b	Réservé***(s/ée/ées)***	***Ne doit/doivent pas*** être utilisé	Section 1.4.2.10	-

Bit(s)	Valeur	Paramètre	Description	Référence	Applicabilité
B27…0		Réservé***(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.	Section 1.4.2.10	-

6.4.3.1 Mode porteur BIST

Dès la réception d'un message ***BIST***, associé à un objet de données BIST ***Mode porteur BIST***, l'UUT ***Devoir/Doit/Doivent*** envoyer une chaîne continue de «1» et de «0» en alternance.

L'UUT ***Devoir/Doit/Doivent*** quitter le mode BIST continu dans un délai ***tBISTContMode*** à partir de l'activation de ce mode BIST continu (voir 6.6.7.2).

6.4.3.2 BIST Test Data

Dès la réception d'un message ***BIST***, associé à un objet de données BIST ***BIST Test Data***, l'UUT ***Devoir/Doit/Doivent*** renvoyer un message ***GoodCRC*** et ***Devoir/Doit/Doivent*** entrer en mode d'essai, au cours duquel elle n'envoie plus de messages, à l'exception des messages ***GoodCRC*** en réponse aux messages reçus. Voir 5.9.2 pour la définition de la trame de données d'essai.

L'essai ***Devoir/Doit/Doivent*** se terminer avec l'envoi d'un signal ***Hard Reset*** afin de réinitialiser l'UUT.

6.4.3.3 Mode d'essai BIST de capacité partagée

Un groupe de ports à capacité partagée partage une source d'alimentation commune qui n'est pas capable d'alimenter simultanément tous les ports à leurs capacités de source complètes (voir ***[USB Type-C 2.0]***). Le mode d'essai BIST de capacité partagée ne ***Devoir/Doit/Doivent*** être mis en œuvre que par les ports d'un groupe à capacité partagée.

Le groupe de ports de l'UUT à capacité partagée ***Devoir/Doit/Doivent*** contenir un ou plusieurs ports, désignés comme ports maîtres, qui reconnaissent à la fois l'objet de données BIST ***BIST Shared Test Mode Entry*** et l'objet de données BIST ***BIST Shared Test Mode Exit***.

6.4.3.3.1 BIST Shared Test Mode Entry

Lorsqu'un port maître d'un groupe à capacité partagée reçoit un message BIST avec un objet de données BIST ***BIST Shared Test Mode Entry***, alors qu'il est à l'état ***PE_SRC_Ready***, l'UUT doit entrer en mode d'essai de conformité où la capacité de source maximale est toujours délivrée sur chaque port, indépendamment de la disponibilité de la puissance partagée, c'est-à-dire que toute la gestion de la puissance partagée est désactivée.

Les ports du groupe à capacité partagée qui ne sont pas des ports maîtres ***Ne doit/doivent pas*** entrer en mode de conformité à la réception de l'objet de données BIST ***BIST Shared Test Mode Entry***.

A la réception d'un message ***BIST*** avec un objet de données BIST ***BIST Shared Test Mode Entry***, l'UUT ***Devoir/Doit/Doivent*** retourner un message ***GoodCRC*** et ***Devoir/Doit/Doivent*** entrer en mode d'essai BIST de capacité partagée.

Lorsqu'elle entre dans ce mode, l'UUT ***Devoir/Doit/Doivent*** envoyer un nouveau message ***Source_Capabilities*** à partir de chaque port du groupe à capacité partagée dans un délai ***tBISTSharedTestMode***. Le dispositif d'essai ne dépasse pas la capacité partagée dans ce mode.

6.4.3.3.2 BIST Shared Test Mode Exit

A la réception d'un message ***BIST*** avec un objet de données BIST ***BIST Shared Test Mode Exit***, l'UUT ***Devoir/Doit/Doivent*** retourner un message ***GoodCRC*** et ***Devoir/Doit/Doivent*** quitter le mode d'essai BIST de capacité partagée. En mode d'essai BIST de capacité partagée, la réception de tout autre message, à l'exception d'un message ***BIST***, avec un objet de données BIST ***BIST Shared Test Mode Exit***, ***Ne doit/doivent pas*** amener l'UUT à quitter le mode d'essai BIST de capacité partagée.

Lorsqu'elle quitte ce mode, l'UUT peut envoyer un nouveau message ***Source_Capabilities*** à chaque port du groupe à capacité partagée ou l'UUT peut exécuter ***ErrorRecovery*** sur chaque port.

Les ports du groupe à capacité partagée qui ne sont pas des ports maîtres ***Ne doit/doivent pas*** quitter le mode de conformité à la réception de l'objet de données BIST ***BIST Shared Test Mode Entry***.

Les ports du groupe à capacité partagée qui ne sont pas des ports maîtres ***Il convient de ne pas*** quitter le mode de conformité à la réception de l'objet de données BIST ***BIST Shared Test Mode Exit***.

- L'UUT ***Devoir/Doit/Doivent*** quitter le mode d'essai BIST de capacité partagée lorsqu'il est hors tension.
- L'UUT ***Devoir/Doit/Doivent*** rester en mode d'essai BIST de capacité partagée lors de tout événement PD (sauf en cas de réception d'un objet de données BIST ***BIST Shared Test Mode Exit***); plus précisément, l'UUT ***Devoir/Doit/Doivent*** rester en mode d'essai BIST de capacité partagée lorsque l'un des événements PD suivants se produit:
  - Réinitialisation matérielle
  - Réinitialisation de câble
  - Réinitialisation logicielle
  - Permutation des rôles de transmission de données
  - Permutation des rôles d'alimentation
  - Permutation rapide des rôles
  - Permutation de VCONN.
- L'UUT ***Pouvoir/Peut/Peuvent*** quitter le mode d'essai si le dispositif d'essai effectue une demande qui dépasse les capacités de l'UUT.

6.4.4 Message Vendor Defined

Le message ***Vendor_Defined*** (VDM) permet aux fournisseurs d'échanger des informations autres que celles définies par la présente spécification.

Un message ***Vendor_Defined*** ***Devoir/Doit/Doivent*** être constitué d'au moins un objet de données de fournisseur, de l'entête VDM, et ***Pouvoir/Peut/Peuvent*** contenir jusqu'à six objets VDM (VDO) supplémentaires.

Afin d'assurer l'unicité en ce qui concerne le fournisseur des messages ***Vendor_Defined***, tous les messages ***Vendor_Defined*** ***Devoir/Doit/Doivent*** contenir une norme USB ***Valide(s)*** ou un ID de fournisseur (SVID) alloué par USB-IF dans l'en-tête de VDM.

Deux types de messages ***Vendor_Defined*** sont définis: les VDM structurés et les VDM non structurés. Un VDM structuré définit une structure extensible conçue pour prendre en charge un fonctionnement modal. Un VDM non structuré ne définit pas de structure et les messages ***Pouvoir/Peut/Peuvent*** être créés de la manière choisie par le fournisseur.

Les messages ***Vendor_Defined*** ***Ne doit/doivent pas*** être utilisés pour une négociation de puissance directe. Ils ***Pouvoir/Peut/Peuvent*** toutefois être utilisés pour modifier la politique locale, et affecter ce qui est offert ou consommé par le biais des messages d'alimentation USB classiques. Par exemple, un message ***Vendor_Defined*** peut être utilisé pour permettre à la source d'offrir davantage de puissance par le biais d'un message ***Source_Capabilities***.

Le format de message ***Devoir/Doit/Doivent*** être conforme aux indications de la Figure 6-14.

Figure 6-14 Message Vendor Defined

Anglais	Français
Header	En-tête
No. of Data Objects	Nombre d'objets de données
VDM Header	En-tête de VDM

L'en-tête de VDM ***Devoir/Doit/Doivent*** être le premier objet de 4 octets d'un message Vendor Defined. L'en-tête de VDM fournit l'espace de commande nécessaire aux fournisseurs pour personnaliser les messages selon leurs propres besoins. Les fournisseurs ***Pouvoir/Peut/Peuvent*** également utiliser les commandes dans un VDM structuré.

La définition des champs de l'en-tête de VDM pour un VDM non structuré, lorsque le bit VDM Type est défini sur 0, ***Devoir/Doit/Doivent*** être conforme à la définition du Tableau 6-24. La définition des champs de l'en-tête de VDM pour un VDM structuré, lorsque le bit VDM Type est défini sur 1, ***Devoir/Doit/Doivent*** être conforme à la définition du Tableau 6-25.

Les VDM structurés et non structurés ne ***Devoir/Doit/Doivent*** être envoyés et reçus qu'après l'établissement d'un contrat explicite. La seule exception concerne la commande ***Discover Identity***, qui ***Pouvoir/Peut/Peuvent*** être envoyée par la source lorsqu'aucun contrat ou un contrat implicite (établi après une permutation des rôles d'alimentation ou une permutation rapide des rôles) est en place, afin de découvrir les capacités du câble (voir 8.3.3.24.3). Une séquence de messages VDM ***Ne doit/doivent pas*** interrompre toute autre séquence de messages d'alimentation USB. Une séquence de messages VDM ***Devoir/Doit/Doivent*** être interruptible par toute autre séquence de messages d'alimentation USB.

6.4.4.1 VDM non structuré

Le VDM non structuré ne définit pas le contenu des bits B14…0 de l'en-tête de VDM. Leur définition et leur utilisation relèvent de la responsabilité exclusive du fournisseur désigné par le VID. Les ports partenaires et les fiches de câbles ***Devoir/Doit/Doivent*** sortir de tout état dans lequel ils sont entrés à l'aide d'un VDM non structuré en cas de réinitialisation matérielle de l'alimentation.

Les règles suivantes s'appliquent dans le cadre de l'utilisation de messages VDM non structurés:

- les VDM non structurés ne ***Devoir/Doit/Doivent*** être utilisés que lorsqu'un contrat explicite est en place;
- les VDM non structurés ***Ne doit/doivent pas*** être envoyés avant l'établissement d'un contrat explicite, et ***Devoir/Doit/Doivent*** être ***Ignoré(s/ée/ées)*** en cas de réception;
- seul le port orienté en aval ***Devoir/Doit/Doivent*** être initiateur de VDM non structurés;
- seuls le port orienté en amont ou une fiche de câble ***Devoir/Doit/Doivent*** répondre à des VDM non structurés;
- les VDM non structurés ***Ne doivent*** en aucune autre circonstance être initiés ou faire l'objet d'une réponse;
- une séquence de "commandes" ***Devoir/Doit/Doivent*** être interruptible, par exemple si une AMS relative à l'alimentation est nécessaire;
- les VDM non structurés ne ***Devoir/Doit/Doivent*** être utilisés qu'en fonctionnement modal dans le contexte d'un mode actif;
- les VDM non structurés ***Pouvoir/Peut/Peuvent*** être utilisés avec des paquets SOP*;
- lorsqu'un port en aval ou en amont ne prend pas en charge les VDM non structurés ou ne reconnaît pas le VID, il ***Devoir/Doit/Doivent*** renvoyer un message ***Not_Supported***.

Le Tableau 6-24 décrit les bits de l'en-tête de VDM.

Tableau 6-24 En-tête de VDM non structuré

Bit(s)	Paramètre	Description
B31…16	ID de fournisseur (VID)	Entier non signé unique de 16 bits. Affecté par l'USB-IF au fournisseur.
B15	Type de VDM	0 = VDM non structuré
B14…0	Disponibles pour utilisation par le fournisseur	Le contenu de ce champ est défini par le fournisseur.

6.4.4.1.1 ID de fournisseur USB

Le champ Vendor ID ***Devoir/Doit/Doivent*** contenir l'ID de fournisseur de 16 bits affecté par l'USB-IF au fournisseur (VID). Aucune autre valeur ne ***Devoir/Doit/Doivent*** être présente dans ce champ.

6.4.4.1.2 Type de VDM

Le champ VDM Type ***Devoir/Doit/Doivent*** être défini sur 0, indiquant qu'il s'agit d'un VDM non structuré.

6.4.4.2 VDM structuré

Définir le champ VDM Type sur 1 (VDM structuré) définit l'utilisation des bits B14...0 dans l'en-tête de VDM structuré. Les champs de l'en-tête de VDM structuré sont définis dans le Tableau 6-25.

Les règles suivantes s'appliquent dans le cadre de l'utilisation de messages VDM structurés.

- Les VDM structurés ne ***Devoir/Doit/Doivent*** être utilisés que lorsqu'un contrat explicite est en place, à ces exceptions près:
  - avant d'établir un contrat explicite, une source ***Pouvoir/Peut/Peuvent*** émettre des messages ***Discover Identity***, à destination d'une fiche de câble qui utilise des paquets SOP', en tant qu'initiatrice (voir 8.3.3.24.3).
- Les deux types de ports ***Pouvoir/Peut/Peuvent*** être initiateurs de VDM structurés, sauf pour les commandes ***Enter Mode*** et ***Exit Mode***, qui ne ***Devoir/Doit/Doivent*** être initiées que par le port orienté en aval.
- Une fiche de câble ***Devoir/Doit/Doivent*** uniquement répondre à des VDM structurés.
- Des VDM structurés ***Ne doit/doivent pas*** être initiés ou faire l'objet d'une réponse, en aucune autre circonstance.
- Lorsqu'un port en aval ou en amont ne prend pas en charge les VDM structurés, il ***Devoir/Doit/Doivent*** renvoyer un message ***Not_Supported*** dès lors qu'il reçoit un VDM structuré.
- Lorsqu'une fiche de câble ne prend pas en charge les VDM structurés, tout VDM structuré reçu ***Devoir/Doit/Doivent*** être ***Ignoré(s/ée/ées)***.
- Un port orienté en amont, un port orienté en aval ou une fiche de câble qui prend en charge les VDM structurés et qui reçoit un VDM structuré pour un SVID qu'il/elle ne reconnaît pas ***Devoir/Doit/Doivent*** répondre avec une commande NAK.
- Une séquence de commandes de VDM structuré ***Devoir/Doit/Doivent*** être interruptible, par exemple si une AMS relative à l'alimentation est nécessaire.

Tableau 6-25 En-tête de VDM structuré

Bit(s)	Champ	Description
B31...16	ID normalisé ou ID de fournisseur (SVID)	Entier non signé unique de 16 bits, affecté par l'USB-IF
B15	Type de VDM	1 = VDM structuré
B14...13	Version de VDM structuré	Numéro de version du VDM structuré (pas la version de la présente spécification): • Version 1.0 = 00b (***Ne doit/doivent pas*** être utilisé) • Version 2.0 = 01b • Les valeurs 2-3 sont ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisées
B12...11	***Réservé(s/ée/ées)***	Pour les commandes 0...15, ***Devoir/Doit/Doivent*** être défini sur 0 et ***Devoir/Doit/Doivent*** être ***Ignoré(s/ée/ées)***. Commandes spécifiques au SVID (16...31), définies par le SVID.

Bit(s)	Champ	Description
B10…8	Position de l'objet	Pour les commandes ***Enter Mode, Exit Mode*** et ***Attention*** (demandes/réponses): • 000b = ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisé. • 001b…110b = Indice dans la liste des VDO, identifiant le VDO du mode souhaité • 111b = Sortie de tous les modes actifs (équivaut à une réinitialisation à la mise sous tension). Ne ***Devoir/Doit/Doivent*** être utilisé qu'avec la commande ***Exit Mode***. Commandes 0…3, 7…15: • 000b • 001b…111b = ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisé. Commandes spécifiques au SVID (16…31), définies par le SVID.
B7…6	Type de commande	00b = REQ (demande du port initiateur) 01b = ACK (réponse d'acquittement du port répondeur) 10b = NAK (réponse d'acquittement négative du port répondeur) 11b = BUSY (réponse du port répondeur "occupé")
B5	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini à 0 et ***Devoir/Doit/Doivent*** être ***Ignoré(s/ée/ées)***
B4…0	Commande[1]	0 = ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé 1 = ***Discover Identity*** 2 = ***Discover SVIDs*** 3 = ***Discover Modes*** 4 = ***Enter Mode*** 5 = ***Exit Mode*** 6 = ***Attention*** 7-15 = ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé 16…31 = Commandes spécifiques au SVID
Note 1: Dans le cas où un SID est utilisé, les modes sont définis par une norme. Lorsqu'un VID est utilisé, les modes sont définis par le fournisseur.		

Le Tableau 6-26 représente les commandes, quel SVID utiliser avec quelle commande, et les valeurs de ***SOP**** qui ***Devoir/Doit/Doivent*** être utilisées.

Tableau 6-26 Commandes relatives au VDM structuré

Commande	Champ SVID de l'en-tête de VDM	SOP* utilisé
Discover Identity	***Devoir/Doit/Doivent*** utiliser uniquement le PD SID.	***Devoir/Doit/Doivent*** utiliser uniquement ***SOP***/***SOP'***.
Discover SVIDs	***Devoir/Doit/Doivent*** utiliser uniquement le PD SID.	***Devoir/Doit/Doivent*** utiliser uniquement ***SOP***/***SOP'***.
Discover Modes	***Valide(s)*** avec n'importe quel SVID.	***Devoir/Doit/Doivent*** utiliser uniquement ***SOP***/***SOP'***.
Enter Mode	***Valide(s)*** avec n'importe quel SVID.	***Valide(s)*** avec ***SOP****.
Exit Mode	***Valide(s)*** avec n'importe quel SVID.	***Valide(s)*** avec ***SOP****.
Attention	***Valide(s)*** avec n'importe quel SVID.	***Valide(s)*** avec ***SOP***.
Commandes spécifiques au SVID	***Valide(s)*** avec n'importe quel SVID.	***Valide(s)*** avec ***SOP**** (défini par SVID).

6.4.4.2.1 SVID

Le champ SVID ***Devoir/Doit/Doivent*** contenir la valeur de l'ID normalisé USB de 16 bits (SID) ou l'ID de 16 bits affecté par l'USB-IF au fournisseur (VID). Aucune autre valeur ne ***Devoir/Doit/Doivent*** être présente dans ce champ.

Le Tableau 6-27 répertorie les valeurs de SVID référencées dans la présente spécification.

Tableau 6-27 Valeurs de SVID

Paramètre	Valeur	Description
PD SID	0xFF00	ID normalisé alloué à la présente spécification

6.4.4.2.2 Type de VDM

Le champ VDM Type ***Devoir/Doit/Doivent*** être défini sur 1, indiquant qu'il s'agit d'un VDM structuré.

6.4.4.2.3 Version de VDM structuré

Le champ Structured VDM Version indique le niveau de fonctionnalité pris en charge dans la partie VDM structuré de la spécification. Il ne s'agit pas de la version de la présente spécification. Ce champ ***Devoir/Doit/Doivent*** être défini sur 01b pour indiquer la Version 2.0.

Pour assurer l'interopérabilité avec les produits USB-PD existants, ces derniers ***Devoir/Doit/Doivent*** prendre en charge chaque numéro de version de VDM structuré à partir de la Version 1.0.

A réception d'un en-tête de VDM dont le numéro de version est supérieur à celui pris en charge, un port ***Devoir/Doit/Doivent*** répondre en utilisant le numéro de version le plus élevé qu'il prend en charge.

Le champ Structured VDM Version de la commande ***Discover Identity*** envoyée et reçue lors de la découverte de VDM ***Devoir/Doit/Doivent*** être utilisé pour déterminer la version commune de VDM structuré la plus ancienne prise en charge par les ports partenaires ou la fiche de câble, et ***Devoir/Doit/Doivent*** poursuivre son fonctionnement à l'aide de cette révision de spécification, jusqu'à ce que ceux-ci soient débranchés. A la découverte de la version de VDM, le champ Structured VDM Version ***Devoir/Doit/Doivent*** correspondre à la version de VDM structuré commune fixée.

6.4.4.2.4 Position de l'objet

Le champ Object Position ***Devoir/Doit/Doivent*** être utilisé par les commandes ***Enter Mode*** et ***Exit Mode***. La commande ***Discover Modes*** renvoie une liste de zéro à six VDO, chacun décrivant un mode. La valeur du champ Object Position est un indice tiré de cette liste, qui indique le VDO (mode) de la liste auquel les commandes ***Enter Mode*** et ***Exit Mode*** font référence. La position de l'objet ***Devoir/Doit/Doivent*** commencer par un pour le premier mode de la liste. Si le SVID est un VID, le contenu du VDO pour le mode ***Devoir/Doit/Doivent*** être défini par le fournisseur. Si le SVID est un SID, le contenu ***Devoir/Doit/Doivent*** être défini par la norme. Le contenu du VDO ***Pouvoir/Peut/Peuvent*** être aussi simple qu'une valeur numérique, ou aussi complexe qu'une image binaire de description des capacités du mode. Dans tous les cas, le répondeur est responsable du déchiffrement du contenu, pour savoir s'il prend en charge ou non le mode correspondant à la position de l'objet.

Ce champ ***Devoir/Doit/Doivent*** être défini sur 0 dans la demande ou la réponse (REQ; ACK, NAK ou BUSY) lorsqu'il n'est pas exigé par la spécification de la commande concernée.

6.4.4.2.5 Type de commande

6.4.4.2.5.1 Commandes autres qu'Attention

Le champ Command Type ***Devoir/Doit/Doivent*** être utilisé pour indiquer le type de demande/réponse de commande en cours d'envoi.

Un initiateur ***Devoir/Doit/Doivent*** définir ce champ sur REQ pour indiquer qu'il s'agit d'une demande de commande lancée par un initiateur.

Si les VDM structurés sont pris en charge, les réponses sont les suivantes:

- "Responder ACK" est le retour normal et ***Devoir/Doit/Doivent*** être envoyé pour indiquer que la demande de commande a été reçue et traitée normalement;
- "Responder NAK" ***Devoir/Doit/Doivent*** être envoyé lorsque la demande de commande:
  - contient un paramètre ***Invalide(s)*** (par exemple, SVID ou mode ***Invalide(s)***);

- ne peut pas être traitée, car la configuration est incorrecte (par exemple, un mode dépendant d'un autre mode ou une demande de sortie d'un mode non actif);
- est un message non reconnu;
- la gestion de "Responder NAK" appartient à l'initiateur;

- "Responder BUSY" ***Devoir/Doit/Doivent*** être envoyé en réponse à un VDM lorsque le répondeur n'est pas capable de répondre immédiatement à la demande de commande, mais que la demande de commande ***Pouvoir/Peut/Peuvent*** être renouvelée. Lors de la réception d'une réponse "Responder BUSY", l'initiateur ***Devoir/Doit/Doivent*** attendre un délai ***tVDMBusy*** avant de renouveler la demande de commande.

6.4.4.2.5.2 Commande Attention

Le champ Command Type ***Devoir/Doit/Doivent*** être utilisé pour indiquer le type de demande de commande en cours d'envoi. Un initiateur ***Devoir/Doit/Doivent*** définir ce champ sur REQ pour indiquer qu'il s'agit d'une demande de commande lancée par un initiateur. Si les VDM structurés sont pris en charge, la commande ***Attention*** ne ***Devoir/Doit/Doivent*** entraîner aucune réponse.

6.4.4.2.6 Commande

6.4.4.2.6.1 Commandes autres qu'Attention

Ce champ contient la valeur de la commande de VDM en cours d'envoi. Les commandes explicitement répertoriées dans ce champ permettent d'identifier les dispositifs et de gérer leurs modes de fonctionnement. Le fournisseur dispose de davantage de valeurs de commande pour gérer les extensions supplémentaires.

Une commande de VDM structuré consiste en une demande de commande et une réponse de commande (ACK, NAK ou BUSY). Une commande de VDM structuré est réputée terminée (et, le cas échéant, la transition vers la fonction demandée est effectuée) lorsque le message ***GoodCRC*** est reçu avec succès par le répondeur, en réponse à sa réponse de commande.

Si les VDM structurés sont pris en charge, mais que la demande de commande de VDM structuré n'est pas reconnue, la réponse ***Devoir/Doit/Doivent*** être NAK (voir Tableau 6-28).

6.4.4.2.6.2 Commande Attention

Ce champ contient la valeur de la commande de VDM en cours d'envoi (***Attention***). La commande ***Attention*** ***Pouvoir/Peut/Peuvent*** être utilisée par l'initiateur pour indiquer au répondeur qu'il exige un service.

Une commande ***Attention*** de VDM structuré consiste en une demande de commande, mais pas en une réponse de commande. Une commande ***Attention*** de VDM structuré est réputée terminée lorsque le message ***GoodCRC*** est reçu avec succès par l'initiateur, en réponse à sa demande de commande ***Attention***.

Si les VDM structurés sont pris en charge, mais que la demande de commande de VDM structuré ***Attention*** n'est pas reconnue, la réponse ***Devoir/Doit/Doivent*** être ***Ignoré(s/ée/ées)*** (voir Tableau 6-28).

6.4.4.3 Utilisation des commandes

L'en-tête de VDM pour un message de VDM structuré définit les commandes utilisées pour extraire une liste de SVID que le dispositif prend en charge, pour découvrir les modes associés à chaque SVID, et pour entrer/sortir des modes. Les commandes incluent:

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

L'espace de commande supplémentaire est également réservé pour l'utilisation normalisée et pour l'utilisation par le fournisseur, ainsi que pour de futures extensions.

Les séquences de commandes utilisent les termes d'initiateur et de répondeur pour identifier les rôles que les ports assument les uns par rapport aux autres en ce qui concerne les messages. Ce rôle est indépendant des capacités d'alimentation du port (fournisseur, consommateur, etc.) ou de son rôle d'alimentation actuel (source ou destinataire). L'initiateur est le port qui envoie la demande de commande initiale, le répondeur est le port qui répond par la réponse de commande. Voir Section 6.4.4.3.6.

Tous les ports qui prennent en charge les modes ***Devoir/Doit/Doivent*** prendre en charge les commandes ***Discover Identity***, ***Discover SVIDs***, ***Discover Modes***, ***Enter Mode*** et ***Exit Mode***.

Le Tableau 6-28 décrit les réponses qu'un répondeur ***Pouvoir/Peut/Peuvent*** émettre à chaque demande de commande. Le répondeur ***Ne doit/doivent pas*** envoyer de réponse qui n'est pas répertoriée pour une commande donnée. ***Il convient d'/de/qu'/que*** une réponse NAK soit interprétée comme une indication de ne pas relancer cette commande particulière.

Tableau 6-28 Commandes et réponses

Commande	Réponse admise	Référence
Discover Identity	ACK, NAK, BUSY	Section 6.4.4.3.1
Discover SVIDs	ACK, NAK, BUSY	Section 6.4.4.3.2
Discover Modes	ACK, NAK, BUSY	Section 6.4.4.3.3
Enter Mode	ACK, NAK	Section 6.4.4.3.4
Exit Mode	ACK, NAK	Section 6.4.4.3.5
Attention	Aucune	Section 6.4.4.3.6

Des exemples d'utilisation des commandes peuvent être consultés à l'Annexe G.

6.4.4.3.1 Découverte d'identité

La commande ***Discover Identity*** permet à un initiateur d'identifier son port partenaire, et elle permet à un initiateur (source VCONN) d'identifier le répondeur (fiche de câble). La commande ***Discover Identity*** permet également de déterminer si une fiche de câble est apte à l'alimentation USB, en recherchant un message de réponse ***GoodCRC***.

La commande ***Discover Identity*** ne ***Devoir/Doit/Doivent*** être envoyée au ***SOP*** qu'en cas de contrat explicite.

La commande ***Discover Identity*** ***Devoir/Doit/Doivent*** être utilisée pour déterminer si une fiche de câble donnée est apte à l'alimentation USB (voir 8.3.3.20.1 et 8.3.3.24.3). Dans ce cas, une demande de commande ***Discover Identity*** envoyée au SOP' ***Ne doit/doivent pas*** entraîner de réinitialisation logicielle si un message de réponse ***GoodCRC*** n'est pas renvoyé, car cela peut indiquer que le câble n'est pas apte à l'alimentation PD. Noter qu'une fiche de câble n'est prête pour une communication PD que lorsque le délai tVCONNStable est atteint après application de VCONN (voir ***[USB Type-C 2.0]***). Lors de la découverte de fiche de câble, en cas de contrat explicite, les commandes ***Discover Identity*** sont envoyées à une fréquence définie par ***DiscoverIdentityTimer*** (voir 6.6.15), jusqu'à une valeur maximale de ***nDiscoverIdentityCount*** fois (voir 6.7.5).

En réponse à une demande de commande ***Discover Identity*** envoyée au ***SOP'***, une fiche de câble apte à l'alimentation USB ***Devoir/Doit/Doivent*** renvoyer une confirmation de commande ***Discover Identity*** ACK.

La commande ***Discover Identity*** ***Devoir/Doit/Doivent*** être utilisée pour déterminer l'identité et/ou les capacités du port partenaire. En réponse à une demande de commande ***Discover Identity*** envoyée au ***SOP***, les produits suivants ***Devoir/Doit/Doivent*** renvoyer une confirmation de commande ***Discover Identity*** ACK:

- un UFP apte à l'alimentation USB qui prend en charge le fonctionnement modal;
- un produit apte à l'alimentation USB qui possède plusieurs DFP;
- un produit ***[USB4]*** apte à l'alimentation USB.

Le SVID de la demande de commande ***Discover Identity*** ***Devoir/Doit/Doivent*** être défini sur le ***PD SID*** (voir Tableau 6-27).

Le champ ***Number of Data Objects*** de l'en-tête de message de la demande de commande ***Discover Identity*** ***Devoir/Doit/Doivent*** être défini sur 1, car la demande de commande ***Discover Identity*** ***Ne doit/doivent pas*** contenir de VDO.

La confirmation ACK de commande ***Discover Identity*** renvoyée par le répondeur ***Devoir/Doit/Doivent*** contenir un VDO d'en-tête d'ID, un VDO Cert Stat, un VDO de produit et les VDO de type de produit, définis par le type de produit, comme indiqué à la Figure 6-15. La présente spécification définit les VDO de type de produit suivants:

- VDO de câble passif (voir 6.4.4.3.1.6);
- VDO de câble actif (voir 6.4.4.3.1.7);
- VDO d'adaptateur de mode alternatif (voir 6.4.4.3.1.8);
- VDO de dispositif USB alimenté par VCONN (voir 6.4.4.3.1.9);
- VDO d'UFP (voir 6.4.4.3.1.4);
- VDO de DFP (voir 6.4.4.3.1.5).

Aucun VDO autre que ceux définis dans la présente spécification ne ***Devoir/Doit/Doivent*** être envoyé dans le cadre de la réponse de commande ***Discover Identity***. Lorsqu'aucun VDO de type de produit n'est défini pour un type de produit spécifique, aucun VDO ne ***Devoir/Doit/Doivent*** être envoyé dans le cadre de la réponse de commande ***Discover Identity***. Tout autre VDO reçu par l'initiateur ***Devoir/Doit/Doivent*** être ***Ignoré(s/ée/ées)***.

Figure 6-15 Réponse de commande Discover Identity

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)

Anglais	Français
Header	En-tête
No. of Data Objects	Nombre d'objets de données
VDM Header	En-tête de VDM
ID Header VDO	VDO d'en-tête d'ID
Cert Stat VDO	VDO Cert Stat
Product VDO	VDO de produit
0..3[2] Product Type VDO(s)	0...3[2] VDO de type de produit

[1] Seuls les objets de données définis dans la présente spécification peuvent être envoyés dans le cadre de la commande ***Discover Identity***.

[2] Les sections suivantes définissent le nombre et le contenu des VDO pour chaque type de produit.

Le champ ***Number of Data Objects*** de l'en-tête de message des réponses de commande ***Discover Identity*** NAK et BUSY ***Devoir/Doit/Doivent*** être défini sur 1, car elles ***Ne doit/doivent pas*** contenir de VDO.

Si le produit est un DRD, un type de produit (UFP) et un type de produit (DFP) sont déclarés dans l'en-tête d'ID. Ces produits ***Devoir/Doit/Doivent*** définir les bits PDUSB Host et PDUSB Peripheral dans l'en-tête d'ID et ***Devoir/Doit/Doivent*** retourner les VDO de type de produit pour le périphérique PDUSB et pour l'hôte PDUSB, en commençant par les VDO d'UFP suivis du VDO de DFP, comme indiqué à la Figure 6-16.

Figure 6-16 Réponse de commande Discover Identity pour un DRD

Header No. of Data Objects = 7	VDM Header	ID Header VDO	Cert Stat VDO	Product VDO	Product Type VDO(s)		
					UFP1	UFP2	DFP

Anglais	Français
Header	En-tête

No. of Data Objects	Nombre d'objets de données
VDM Header	En-tête de VDM
ID Header VDO	VDO d'en-tête d'ID
Cert Stat VDO	VDO Cert Stat
Product VDO	VDO de produit
Product Type VDO(s)	VDO de type de produit

6.4.4.3.1.1 VDO d'en-tête d'ID

Le VDO d'en-tête d'identification contient des informations correspondant au produit d'alimentation électrique. Les champs du VDO d'en-tête d'ID ***Devoir/Doit/Doivent*** être conformes à la définition du Tableau 6-29.

Tableau 6-29 VDO d'en-tête d'ID

Bit(s)	Description	Référence
B31	Apte aux communications USB en tant qu'hôte USB: • ***Devoir/Doit/Doivent*** être défini sur 1 si le produit est capable d'énumérer des dispositifs USB. • ***Devoir/Doit/Doivent*** être défini sur 0 sinon.	Section 6.4.4.3.1.1.1
B30	Apte aux communications USB en tant que dispositif USB: • ***Devoir/Doit/Doivent*** être défini sur si le produit est en mesure d'être compris dans une énumération de dispositifs USB. • ***Devoir/Doit/Doivent*** être défini sur 0 sinon.	Section 6.4.4.3.1.1.2
B29…27	Type de produit (UFP) • 000b – Non défini • 001b – Hub PDUSB • 010b – Dispositif PDUSB • 011b – PSD • 100b – ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé. • 101b – Adaptateur de mode alternatif (AMA) • 110b – Dispositif USB alimenté par VCONN (VPD) • 111b – ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé. Type de produit (Fiche de câble): • 000b – Non défini • 001b…010b – ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé. • 011b – Câble passif • 100b – Câble actif • 101b…111b – ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé.	Section 6.4.4.3.1.1.3
B26	Fonctionnement modal pris en charge: • ***Devoir/Doit/Doivent*** être défini sur 1 si le produit prend en charge le fonctionnement modal (modes alternatifs). • ***Devoir/Doit/Doivent*** être défini sur 0 sinon.	Section 6.4.4.3.1.1.4
B25…23	Type de produit (DFP) • 000b – Non défini • 001b – Hub PDUSB • 010b – Hôte PDUSB • 011b – Transformateur • 100b – Contrôleur de mode alternatif (AMC) • 101b…111b – ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé.	

Bit(s)	Description	Référence
B22…16	***Réservé(s/ée/ées)***, ***Devoir/Doit/Doivent*** être défini sur 0.	
B15…0	ID de fournisseur USB.	*[USB 2.0]*/*[USB 3.2]*/*[USB4]*

6.4.4.3.1.1.1 Apte aux communications USB en tant qu'hôte USB:

Le champ Data Capable as a USB Host permet d'indiquer si un port présente ou non une capacité d'hôte USB.

6.4.4.3.1.1.2 Apte aux communications USB en tant que dispositif USB:

Le champ Data Capable as a USB Device permet d'indiquer si un port présente ou non une capacité de dispositif USB.

6.4.4.3.1.1.3 Type de produit (UFP)

Le champ Product Type (UFP) indique le type de produit pour un port de données orienté en amont, si un VDO est renvoyé et, le cas échéant, le type de VDO à renvoyer. Le type de produit indiqué dans le champ Product Type (UFP) ***Devoir/Doit/Doivent*** être la catégorisation la plus proche de la fonctionnalité principale du produit dont le rôle de transmission de données est UFP ou "Indéfini" lorsqu'il n'existe pas de catégorie adaptée pour le produit. Pour les produits DRD, ce champ ***Devoir/Doit/Doivent*** toujours indiquer le type de produit pour un port de données orienté en amont, quel que soit le rôle de transmission de données actuel. Le Tableau 6-30 définit les VDO de type de produit qui ***Devoir/Doit/Doivent*** être renvoyés.

Tableau 6-30 Types de produits (UFP)

Type de produit	Description	VDO de type de produit	Référence
Non défini	***Devoir/Doit/Doivent*** être utilisé lorsqu'aucune autre valeur de type de produit n'est appropriée.	Aucune	
Hub PDUSB	***Devoir/Doit/Doivent*** être utilisé lorsque le produit est un hub PDUSB.	VDO d'UFP	Section 6.4.4.3.1.4
Périphérique PDUSB	***Devoir/Doit/Doivent*** être utilisé lorsque le produit est un dispositif PDUSB autre qu'un hub PDUSB.	VDO d'UFP	Section 6.4.4.3.1.4
PSD	***Devoir/Doit/Doivent*** être utilisé lorsque le produit est une PSD, par exemple une réserve de puissance.	Aucune	
Adaptateur de mode alternatif	***Devoir/Doit/Doivent*** être utilisé lorsque le produit est un dispositif PDUSB prenant en charge un ou plusieurs modes alternatifs.	VDO AMA	Section 6.4.4.3.1.8
Dispositif USB alimenté par VCONN	***Devoir/Doit/Doivent*** être utilisé lorsque le produit est un dispositif PDUSB alimenté par Vconn.	VDO VPD	Section 6.4.4.3.1.9

6.4.4.3.1.1.4 Type de produit (fiche de câble)

Le champ Product Type (Cable Plug) indique le type de produit lorsqu'il s'agit d'une fiche de câble, si un VDO est renvoyé et, le cas échéant, le type de VDO à renvoyer. Le Tableau 6-31 définit les VDO de type de produit qui ***Devoir/Doit/Doivent*** être renvoyés.

Tableau 6-31 Types de produits (fiche de câble)

Type de produit	Description	VDO de type de produit	Référence
Non défini	***Devoir/Doit/Doivent*** être utilisé lorsqu'aucune autre valeur de type de produit n'est appropriée.	Aucune	
Câble actif	***Devoir/Doit/Doivent*** être utilisé lorsque le produit est un câble qui comporte des circuits adaptateurs de signaux.	VDO Câble actif	Section 6.4.4.3.1.7
Câble passif	***Devoir/Doit/Doivent*** être utilisé lorsque le produit est un câble qui ne comporte pas de circuits adaptateurs de signaux.	VDO de câble passif	Section 6.4.4.3.1.6

6.4.4.3.1.1.5 Fonctionnement modal pris en charge

Le bit Modal Operation Supported permet d'indiquer si le produit prend ou non en charge les modes.

6.4.4.3.1.1.6 Type de produit (DFP)

Le champ Product Type (DFP) indique le type de produit pour un port de données orienté en aval, si un VDO est renvoyé et, le cas échéant, le type de VDO à renvoyer. Le type de produit indiqué dans le champ Product Type (DFP) ***Devoir/Doit/Doivent*** être la catégorisation la plus proche de la fonctionnalité principale du produit dont le rôle de transmission de données est DFP ou "Indéfini" lorsqu'il n'existe pas de catégorie adaptée pour le produit. Pour les produits DRD, ce champ ***Devoir/Doit/Doivent*** toujours indiquer le type de produit pour un port de données orienté en aval, quel que soit le rôle de transmission de données actuel. Le Tableau 6-32 définit les VDO de type de produit qui ***Devoir/Doit/Doivent*** être renvoyés.

Tableau 6-32 Types de produits (DFP)

Type de produit	Description	VDO de type de produit	Référence
Non défini	***Devoir/Doit/Doivent*** être utilisé lorsqu'aucune autre valeur de type de produit n'est appropriée.	Aucune	
Hub PDUSB	***Devoir/Doit/Doivent*** être utilisé lorsque le produit est un hub PDUSB.	VDO de DFP	Section 6.4.4.3.1.5
Hôte PDUSB	***Devoir/Doit/Doivent*** être utilisé lorsque le produit est un hôte PDUSB.	VDO de DFP	Section 6.4.4.3.1.5
Transformateur	***Devoir/Doit/Doivent*** être utilisé lorsque le produit est un transformateur/adaptateur.	VDO de DFP	Section 6.4.4.3.1.5
Contrôleur de mode alternatif	***Devoir/Doit/Doivent*** être utilisé lorsque le produit est un hôte PDUSB ou un port DFP prenant en charge un ou plusieurs modes alternatifs.	Aucune	

6.4.4.3.1.1.7 ID de fournisseur

Les fabricants ***Devoir/Doit/Doivent*** définir le champ Vendor ID sur la valeur de l'ID de fournisseur qui leur a été affectée par l'USB-IF. Pour les dispositifs USB ou les hubs qui prennent en charge les communications USB, le champ Vendor ID ***Devoir/Doit/Doivent*** être identique au champ Vendor ID défini dans le descripteur de dispositif USB du produit (voir ***[USB 2.0]*** et ***[USB 3.2]***).

6.4.4.3.1.2 VDO Cert Stat

Le VDO Cert Stat ***Devoir/Doit/Doivent*** contenir le XID affecté au format binaire par l'USB-IF au produit avant sa certification. Les champs du VDO Cert Stat ***Devoir/Doit/Doivent*** être conformes à la définition du Tableau 6-33.

Tableau 6-33 VDO Cert Stat

Bit(s)	Description	Référence
B31…0	Entier non signé de 32 bits, XID	Affecté par l'USB-IF

6.4.4.3.1.3 VDO de produit

Le VDO de produit contient des informations d'identité relatives au produit. Les champs du VDO de produit ***Devoir/Doit/Doivent*** être conformes à la définition du Tableau 6-34.

Tableau 6-34 VDO de produit

Bit(s)	Description	Référence
B31…16	Entier non signé de 16 bits. ID de produit USB	*[USB 2.0]*/*[USB 3.2]*
B15…0	Entier non signé de 16 bits. BcdDevice	*[USB 2.0]*/*[USB 3.2]*

Il convient d'/de/qu'/que le fabricant définisse le champ USB Product ID sur une valeur unique, qui identifie le produit, et ***Il convient d'/de/qu'/que*** celui-ci définisse le champ bcdDevice sur un numéro de version correspondant à la version du produit.

6.4.4.3.1.4 VDO d'UFP

Les VDO d'UFP définis dans la présente section ***Devoir/Doit/Doivent*** être renvoyés par les ports capables de fonctionner en tant qu'UFP, y compris les périphériques USB traditionnels, le port en amont du hub USB et les ports hôtes capables de fonctionner en tant que DRD. Les VDO d'UFP définis dans la présente section ***Devoir/Doit/Doivent*** être renvoyés lorsque le champ Product Type (UFP) du VDO d'en-tête d'ID est donné en tant que périphérique PDUSB ou en tant que hub PDUSB. Le Tableau 6-35 et le Tableau 6-36 définissent les VDO d'UFP qui ***Devoir/Doit/Doivent*** être envoyés en fonction du type de produit.

Un UFP *[USB4]* ***Devoir/Doit/Doivent*** prendre en charge la commande de VDM structuré *Discover Identity*.

Tableau 6-35 VDO d'UFP 1

Bit(s)	Champ	Description
B31…29	UFP VDO Version	Numéro de version du VDO (pas la version de la présente spécification): • Version 1.0 = 000b Les valeurs 001b…111b sont ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisées
B28	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B27…24	Device Capability	**Bit** / **Description** 0 / *[USB 2.0]* Dispositif compatible 1 / *[USB 2.0]* Dispositif compatible (Billboard uniquement) 2 / Dispositif compatible *[USB 3.2]* 3 / Dispositif compatible *[USB4]*
B23…6	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B5…3	Modes alternatifs	**Bit** / **Description** 0 / Prend en charge le mode alternatif *[TBT3]* 1 / Prend en charge les modes alternatifs qui reconfigurent les signaux sur le connecteur *[USB Type-C 2.0]*, sauf pour *[TBT3]*. 2 / Prend en charge les modes alternatifs qui ne reconfigurent pas les signaux sur le connecteur *[USB Type-C 2.0]*

Bit(s)	Champ	Description
B2…0	USB Highest Speed	000b = *[USB 2.0]* uniquement, pas de prise en charge SuperSpeed 001b = *[USB 3.2]* Gen1 010b = *[USB 3.2]*/*[USB4]* Gen2 011b = *[USB4]* Gen3 100b…111b = ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé

6.4.4.3.1.4.1 Champ VDO Version

Le champ UFP VDO Version contient une version de VDO pour ce numéro de version de VDM. Ce champ indique le contenu attendu pour les VDO d'UFP.

6.4.4.3.1.4.2 Champ Device Capability

Le champ de bits Device Capability décrit les capacités de l'UFP lorsqu'il fonctionne en tant que dispositif PDUSB ou en tant que hub PDUSB.

Les bits du champ de bits ne ***Devoir/Doit/Doivent*** pas être nuls lorsque la vitesse correspondante du dispositif USB est prise en charge et ***Devoir/Doit/Doivent*** être définis sur 0 lorsque la vitesse correspondante du dispositif USB n'est pas prise en charge.

Les bits "Dispositif compatible" *[USB 2.0]* et "Dispositif compatible Billboard uniquement" (bits 0 et 1) ***Ne doit/doivent pas*** être définis simultanément.

6.4.4.3.1.4.3 Champ Alternate Modes

Le champ Alternate Modes ***Devoir/Doit/Doivent*** être utilisé pour identifier tous les types de modes alternatifs pris en charge par un dispositif, le cas échéant.

6.4.4.3.1.4.4 Champ USB Highest Speed

Le champ USB Highest Speed doit indiquer la capacité de signalisation la plus élevée du port.

Tableau 6-36 VDO d'UFP 2

Bit(s)	Champ	Description
B31…30	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B29…23	USB4 Min Power	Puissance minimale, en watts, exigée pour le fonctionnement *[USB4]*.
B22…16	USB4 Max Power	Puissance, en watts, exigée pour une pleine fonctionnalité, à l'exclusion de toute puissance exigée pour la charge de la batterie ou pour la redistribution en fonctionnement *[USB4]*.
B15…14	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B13…7	USB3 Min Power	Puissance minimale, en watts, exigée pour le fonctionnement *[USB 3.2]*.
B6…0	USB3 Max Power	Puissance, en watts, exigée pour une pleine fonctionnalité, à l'exclusion de toute puissance exigée pour la charge de la batterie ou pour la redistribution en fonctionnement *[USB 3.2]*.

6.4.4.3.1.4.5 Champ USB4 Min Power

Le champ USB4 Min Power ***Devoir/Doit/Doivent*** contenir la quantité minimale de puissance, arrondie à la valeur entière supérieure, nécessaire à un dispositif à fonctionnement *[USB4]* pour être fonctionnel. A ce niveau de puissance minimal, le dispositif peut être énuméré et au moins une de ses fonctions doit être opérationnelle, même si ses performances peuvent être réduites.

6.4.4.3.1.4.6 Champ USB4 Max Power

Le champ USB4 Max Power ***Devoir/Doit/Doivent*** contenir la quantité de puissance, arrondie à la valeur entière supérieure, nécessaire à un dispositif à fonctionnement *[USB4]* pour être pleinement fonctionnel, aux performances maximales. Toutefois, cela exclut toute puissance supplémentaire nécessaire pour la

charge d'une batterie ou pour une redistribution, par exemple pour utiliser une partie de la puissance pour alimenter un autre port sur un hub.

6.4.4.3.1.4.7 Champ USB3 Min Power

Le champ USB3 Min Power ***Devoir/Doit/Doivent*** contenir la quantité minimale de puissance, arrondie à la valeur entière supérieure, nécessaire à un dispositif à fonctionnement ***[USB 3.2]*** pour être fonctionnel. A ce niveau de puissance minimal, le dispositif peut être énuméré et au moins une de ses fonctions doit être opérationnelle, même si ses performances peuvent être réduites.

6.4.4.3.1.4.8 Champ USB3 Max Power

Le champ USB3 Max Power ***Devoir/Doit/Doivent*** contenir la quantité de puissance, arrondie à la valeur entière supérieure, nécessaire à un dispositif à fonctionnement ***[USB 3.2]*** pour être pleinement fonctionnel, aux performances maximales. Toutefois, cela exclut toute puissance supplémentaire nécessaire pour la charge d'une batterie ou pour une redistribution, par exemple pour utiliser une partie de la puissance pour alimenter un autre port sur un hub.

6.4.4.3.1.5 VDO de DFP

Le VDO de DFP ***Devoir/Doit/Doivent*** être renvoyé par des ports capables de fonctionner en tant que DFP; y compris ceux mis en œuvre par des hôtes, des hubs et des transformateurs. Le VDO de DFP ***Devoir/Doit/Doivent*** être renvoyé lorsque le champ Product Type (DFP) du VDO d'en-tête d'ID est donné en tant que transformateur, hôte PDUSB ou hub PDUSB. Le Tableau 6-37 définit le VDO de DFP qui ***Devoir/Doit/Doivent*** être envoyé.

Tableau 6-37 VDO de DFP

Bit(s)	Champ	Description
B31…29	DFP VDO Version	Numéro de version du VDO (pas la version de la présente spécification): • Version 1.0 = 000b Les valeurs 001b…111b sont ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisées
B28…27	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B26…24	Host Capability	**Bit** / **Description** 0 / ***[USB 2.0]*** Hôte compatible 1 / Hôte compatible ***[USB 3.2]*** 2 / Hôte compatible ***[USB4]***
B23…5	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B4…0	Port Number	Numéro de port unique qui permet d'identifier un port spécifique sur un dispositif à ports multiples.

6.4.4.3.1.5.1 Champ VDO Version

Le champ DFP VDO Version ***Devoir/Doit/Doivent*** contenir une version de VDO pour ce numéro de version de VDM. Ce champ indique le contenu attendu pour le VDO de DFP.

6.4.4.3.1.5.2 Champ Host Capability

Le champ de bits Host Capability ***Devoir/Doit/Doivent*** indiquer si le DFP peut fonctionner en tant qu'hôte PDUSB et décrire les capacités du DFP lorsqu'il fonctionne en tant qu'hôte PDUSB.

Les transformateurs et les hubs PDHUB ***Devoir/Doit/Doivent*** définir les bits Host Capability sur 0.

6.4.4.3.1.5.3 Champ Port Number

Le champ Port Number ***Devoir/Doit/Doivent*** être un nombre unique statique qui identifie sans ambiguïté chaque DFP ***[USB Type-C 2.0]***, y compris les DRP, sur le dispositif. Noter que ce nombre est indépendant du nombre de ports USB.

6.4.4.3.1.6 VDO de câble passif

Le VDO de câble défini dans cette section ***Devoir/Doit/Doivent*** être envoyé lorsque le type de produit est un câble passif. Le Tableau 6-38 définit le VDO de câble qui ***Devoir/Doit/Doivent*** être envoyé.

Un câble passif dispose d'une fiche USB à chaque extrémité, et au moins l'une d'entre elles est une fiche de câble qui prend en charge la communication SOP'. Un câble passif ***Ne doit/doivent pas*** comporter de circuits adaptateurs de signaux de bus de données, et n'a donc pas la notion de directionnalité Super Speed. Un câble passif ***Devoir/Doit/Doivent*** inclure un câble V_{BUS} et ne ***Devoir/Doit/Doivent*** répondre qu'aux communications SOP'. Les câbles passifs ***Devoir/Doit/Doivent*** prendre en charge la commande ***Discover Identity*** des VDM structurés et ***Devoir/Doit/Doivent*** renvoyer le VDO de câble passif dans une confirmation de commande ***Discover Identity*** ACK, comme indiqué dans le Tableau 6-38.

Tableau 6-38 VDO de câble passif

Bit(s)	Champ	Description
B31…28	Version HW	0000b…1111b affectés par le propriétaire du VID
B27…24	Version de micrologiciel	0000b…1111b affectés par le propriétaire du VID
B23…21	Version de VDO	Numéro de version du VDO (pas la version de la présente spécification): • Version 1.0 = 000b Les valeurs 001b…111b sont ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisées
B20	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B19…18	Fiche USB Type-C vers USB Type-C/Captive	00b = ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé 01b = ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé 10b = USB Type-C 11b = Captif
B17	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B16…13	Latence de câble	0000b – ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé 0001b – <10 ns (~1 m) 0010b – 10 ns à 20 ns (~2 m) 0011b – 20 ns à 30 ns (~3 m) 0100b – 30 ns à 40 ns (~4 m) 0101b – 40 ns à 50 ns (~5 m) 0110b – 50 ns à 60 ns (~6 m) 0111b – 60 ns à 70 ns (~7 m) 1000b – > 70 ns (>~7 m) 1001b ….1111b ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé Inclut la latence des composants électroniques du câble actif
B12…11	Type de terminaison de câble	00b = V_{CONN} non exigé. Les fiches de câbles qui ne prennent en charge que les commandes ***Discover Identity*** ***Devoir/Doit/Doivent*** définir ces bits sur 00b. 01b = V_{CONN} exigé 10b…11b = ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé
B10…9	Tension V_{BUS} maximale	Tension de câble V_{BUS} maximale: 00b – 20 V 01b – 30 V 10b – 40V 11b – 50V
B8…7	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B6…5	Capacité de gestion du courant V_{BUS}	00b = ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé. 01b = 3 A 10b = 5 A 11b = ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé.

Bit(s)	Champ	Description
B4…3	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B2…0	USB Highest Speed	000b = ***[USB 2.0]*** uniquement, pas de prise en charge SuperSpeed 001b = ***[USB 3.2]*** Gen1 010b = ***[USB 3.2]***/***[USB4]*** Gen2 011b = ***[USB4]*** Gen3 100b…111b = ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé

6.4.4.3.1.6.1 Champ HW Version

Le champ HW Version (B31...28) contient une version matérielle affectée par le propriétaire du VID.

6.4.4.3.1.6.2 Champ FW Version

Le champ FW Version (B27...24) contient une version de micrologiciel affectée par le propriétaire du VID.

6.4.4.3.1.6.3 Champ VDO Version

Le champ VDO Version (B23...20) contient une version de VDO pour ce numéro de version de VDM. Ce champ indique le contenu attendu pour ce VDO.

6.4.4.3.1.6.4 Champ Connector Type

Le champ Connector Type (B19...18) ***Devoir/Doit/Doivent*** contenir une valeur qui correspond au type de connecteur à l'autre extrémité du connecteur USB Type-C.

6.4.4.3.1.6.5 Champ Cable Latency

Le champ Cable Latency (B16...13) ***Devoir/Doit/Doivent*** contenir une valeur qui correspond à la latence de signal à travers le câble, qui peut être utilisée comme approximation de sa longueur.

6.4.4.3.1.6.6 Champ Cable Termination Type

Le champ Cable Termination Type (B12…11) ***Devoir/Doit/Doivent*** contenir une valeur qui indique si le câble passif n'a besoin de VCONN qu'initialement, afin de prendre en charge la commande ***Discover Identity***, après quoi il peut être retiré, ou si le câble passif exige VCONN en continu, afin d'alimenter certaines fonctions de la fiche de câble.

6.4.4.3.1.6.7 Champ Maximum VBUS Voltage

Le champ Maximum V_{BUS} Voltage (B10…9) ***Devoir/Doit/Doivent*** contenir la tension maximale qui ***Devoir/Doit/Doivent*** être négociée avec une alimentation fixe par le biais du câble dans le cadre d'un contrat explicite, où la tension maximale qui ***Devoir/Doit/Doivent*** être appliquée au câble est ***vSrcNew*** max + ***vSrcValid*** max. Par exemple, lorsque le champ Maximum V_{BUS} Voltage a pour valeur 20 V, une alimentation fixe de 20 V peut être négociée dans le cadre d'un contrat explicite, où la tension maximale absolue pouvant être appliquée au câble est de 21,55 V.

6.4.4.3.1.6.8 Champ VBUS Current Handling Capability

Le champ V_{BUS} Current Handling Capability (B6...5) ***Devoir/Doit/Doivent*** indiquer si le câble est capable d'acheminer un courant de 3 A ou 5 A.

6.4.4.3.1.6.9 Champ USB Highest Speed

Le champ USB Highest Speed (B2...0) ***Devoir/Doit/Doivent*** indiquer le taux de signalisation le plus élevé pris en charge par le câble.

6.4.4.3.1.7 VDO de câble actif

Un câble actif dispose d'une fiche USB à chaque extrémité, et au moins l'une d'entre elles est une fiche de câble qui prend en charge la communication SOP'. Un câble actif ***Devoir/Doit/Doivent*** comporter des circuits adaptateurs de signaux de bus de données, et ***Pouvoir/Peut/Peuvent*** avoir la notion de

directionnalité Super Speed dans ses câbles Super Speed. Un câble actif ***Pouvoir/Peut/Peuvent*** comprendre un fil V_{BUS}.

Un câble actif:

- ***Devoir/Doit/Doivent*** répondre aux communications SOP'
- ***Pouvoir/Peut/Peuvent*** répondre aux communications SOP''
- ***Devoir/Doit/Doivent*** prendre en charge la commande de VDM structuré ***Discover Identity***
- Dans la confirmation de commande ***Discover Identity*** ACK:
  - ***Devoir/Doit/Doivent*** définir le type de produit dans le VDO d'en-tête d'ID sur Active Cable
  - ***Devoir/Doit/Doivent*** renvoyer les VDO de câbles actifs définis dans le Tableau 6-39 et le Tableau 6-40

Tableau 6-39 VDO Active Cable 1

Bit(s)	Champ	Description
B31…28	Version HW	0000b…1111b affectés par le propriétaire du VID
B27…24	Version de micrologiciel	0000b…1111b affectés par le propriétaire du VID
B23…21	Version de VDO	Numéro de version du VDO (pas la version de la présente spécification): • Version 1.3 = 011b Les valeurs 000b, 100b…111b sont ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisées
B20	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B19…18	Type de connecteur	00b = ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé 01b = ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé 10b = USB Type-C 11b = Captif
B17	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B16…13	Latence de câble	0000b – ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé 0001b – <10 ns (~1 m) 0010b – 10 ns à 20 ns (~2 m) 0011b – 20 ns à 30 ns (~3 m) 0100b – 30 ns à 40 ns (~4 m) 0101b – 40 ns à 50 ns (~5 m) 0110b – 50 ns à 60 ns (~6 m) 0111b – 60 ns à 70 ns (~7 m) 1000b –1000 ns (~100 m) 1001b –2000 ns (~200 m) 1010b – 3000 ns (~300 m) 1011b ….1111b ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé Inclut la latence des composants électroniques du câble actif
B12…11	Type de terminaison de câble	00b…01b = ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé 10b = Une extrémité active, une extrémité passive, V_{CONN} exigé 11b = Les deux extrémités actives, V_{CONN} exigé
B10…9	Tension V_{BUS} maximale	Tension de câble V_{BUS} maximale: 00b – 20 V 01b – 30 V 10b – 40 V 11b – 50 V
B8	SBU pris en charge	0 = connexions SBU prises en charge 1 = les connexions SBU ne sont pas prises en charge

Bit(s)	Champ	Description
B7	Type de SBU	Lorsque SBU Supported = 1, ce bit ***Devoir/Doit/Doivent*** être ***Ignoré(s/ée/ées)*** Lorsque SBU Supported = 0: 0 = SBU passif 1 = SBU actif
B6…5	Capacité de gestion du courant V_{BUS}	Lorsque V_{BUS} Through Cable = No, ce champ ***Devoir/Doit/Doivent*** être ***Ignoré(s/ée/ées)***. Lorsque V_{BUS} Through Cable = Yes: 00b = Courant de défaut USB Type-C 01b = 3A 10b = 5 A 11b = ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé.
B4	V_{BUS} Through Cable	0 = Non 1 = Oui
B3	Contrôleur SOP" présent	0 = Aucun contrôleur SOP" présent 1 = Contrôleur SOP" présent
B2…0	USB Highest Speed	000b = ***[USB 2.0]*** uniquement, pas de prise en charge SuperSpeed 001b = ***[USB 3.2]*** Gen1 010b = ***[USB 3.2]***/ ***[USB4]*** Gen2 011b = ***[USB4]*** Gen3 100b…111b = ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé

6.4.4.3.1.7.1 Champ HW Version

Le champ HW Version (B31…28) contient une version matérielle affectée par le propriétaire du VID.

6.4.4.3.1.7.2 Champ FW Version

Le champ FW Version (B27…24) contient une version de micrologiciel affectée par le propriétaire du VID.

6.4.4.3.1.7.3 Champ VDO Version

Le champ VDO Version (B23…20) contient une version de VDO pour ce numéro de version de VDM. Ce champ indique le contenu attendu pour les VDO Active Cable.

6.4.4.3.1.7.4 Champ Connector Type

Le champ Connector Type (B19…18) ***Devoir/Doit/Doivent*** contenir une valeur qui correspond au type de connecteur à l'autre extrémité du connecteur USB Type-C.

6.4.4.3.1.7.5 Champ Cable Latency

Le champ Cable Latency (B16…13) ***Devoir/Doit/Doivent*** contenir une valeur qui correspond à la latence de signal à travers le câble, qui peut être utilisée comme approximation de sa longueur.

6.4.4.3.1.7.6 Champ Cable Termination Type

Le champ Cable Termination Type (B12…11) ***Devoir/Doit/Doivent*** contenir une valeur qui indique si le câble actif dispose d'une ou deux fiches exigeant une alimentation de V_{CONN}.

6.4.4.3.1.7.7 Champ Maximum VBUS Voltage

Le champ V_{BUS} Voltage (B10...9) ***Devoir/Doit/Doivent*** contenir la tension maximale qui ***Devoir/Doit/Doivent*** être négociée dans le cadre d'un contrat explicite lorsque la tension maximale qui ***Devoir/Doit/Doivent*** être appliquée au câble est ***vSrcNew*** max + ***vSrcValid*** max. Lorsque ce champ est défini sur 20 V, le câble aura en toute sécurité un APDO d'alimentation électrique programmable de 20 V, la tension maximale absolue pouvant être appliquée au câble étant de 21,55 V.

6.4.4.3.1.7.8 Champ SBU Supported

Le champ SBU Supported (B8) ***Devoir/Doit/Doivent*** indiquer si le câble prend en charge les SBU dans le câble.

6.4.4.3.1.7.9 Champ SBU Type

Le champ SBU Type (B7) ***Devoir/Doit/Doivent*** indiquer si les SBU sont passifs ou actifs (par exemple numérique).

6.4.4.3.1.7.10 Champ VBUS Current Handling Capability

Le champ VBUS Current Handling Capability (B6…5) ***Devoir/Doit/Doivent*** indiquer si le câble est capable d'acheminer un courant par défaut de 3 A ou 5 A (500 mA USB2, 900 mA USB3.2 x1, 1,5 A USB3.2 x2). La capacité de gestion de courant V_{BUS} ***Devoir/Doit/Doivent*** uniquement être ***Valide(s)*** lorsque le champ V_{BUS} Through Cable indique un câble V_{BUS} de bout en bout.

6.4.4.3.1.7.11 Champ VBUS Through Cable

Le champ V_{BUS} Through Cable (B4) ***Devoir/Doit/Doivent*** indiquer si le câble contient un câble V_{BUS} de bout en bout.

6.4.4.3.1.7.12 Champ SOP'' Controller Present

Le champ SOP'' Controller Present (B3) ***Devoir/Doit/Doivent*** indiquer si l'une des fiches de câbles est compatible avec la communication SOP'' en plus de la communication SOP' ***Normatif(s/ve/ves)***.

6.4.4.3.1.7.13 Champ USB Highest Speed

Le champ USB Highest Speed (B2...0) ***Devoir/Doit/Doivent*** indiquer le taux de signalisation le plus élevé pris en charge par le câble.

Tableau 6-40 VDO Active Cable 2

Bit(s)	Champ	Description
B31…24	Température d'exploitation maximale	La température de fonctionnement interne maximale. Elle peut ou peut ne pas refléter la température externe de la fiche.
B23…16	Température d'arrêt	La température à laquelle le câble procède à l'arrêt thermique de manière à ne pas dépasser la fiche admissible de température de la peau.
B15	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B14…12	U3/CLd Power	000b: >10 mW 001b: 5-10 mW 010b: 1-5 mW 011b: 0,5-1 mW 100b: 0,2-0,5 mW 101b: 50-200 µW 110b: <50 µW 111b: ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé
B11	Mode de transition U3 à U0	0b: U3 to U0 direct 1b: U3 to U0 through U3S
B10	Physical connection	0b = Cuivre 1b = Optique
B9	Active element	0b – Circuit d'amplification actif 1b – Circuit de retemporisation actif
B8	USB4 Supported	0b = ***[USB4]*** pris en charge 1b = ***[USB4]*** non pris en charge
B7…6	Sauts de hub USB 2.0 consommés	Nombre de "sauts de hub" ***[USB 2.0]*** que consomme le câble. ***Devoir/Doit/Doivent*** être défini sur 0 si USB 2.0 n'est pas prise en charge.
B5	Prise en charge de la veille USB 2.0	0b = ***[USB 2.0]*** pris en charge 1b = ***[USB 2.0]*** non pris en charge
B4	Prise en charge de la veille USB 3.2	0b = ***[USB 3.2]*** SuperSpeed pris en charge 1b = ***[USB 3.2]*** SuperSpeed non pris en charge
B3	USB Lanes Supported	0b = Une voie 1b = Deux voies
B2	Câble actif à isolation optique	0b = Non 1b = Oui
B1	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B0	USB Gen	0b = Gen 1 1b = Gen 2 ou supérieure Note: Pour plus d'informations sur la génération prise en charge, voir VDO1 USB Highest Speed.

6.4.4.3.1.7.14 Champ Maximum Operating Temperature

Le champ Maximum Operating Temperature (B31…24) ***Devoir/Doit/Doivent*** indiquer la température de fonctionnement maximale admissible à l'intérieur de la fiche.

6.4.4.3.1.7.15 Champ Shutdown Temperature

Le champ Shutdown Temperature (B23…16) ***Devoir/Doit/Doivent*** indiquer la température à l'intérieur de la fiche à laquelle la fiche procède à l'arrêt de ses composants de signalisation actifs. Lorsque cette température est atteinte, elle est consignée dans le message ***Statut*** du câble actif par le bit Thermal Shutdown.

6.4.4.3.1.7.16 Champ U3/CLd Power

Le champ U3/CLd Power (B14…12) ***Devoir/Doit/Doivent*** indiquer la puissance consommée par le câble en U3 ***[USB 3.2]*** ou en CLd ***[USB4]***.

6.4.4.3.1.7.17 Champ U3 to U0 Transition Mode

Le champ U3 to U0 transition mode (B11) ***Devoir/Doit/Doivent*** indiquer le mode pris en charge par le câble, de U3 à U0. La puissance en U3S, si ce mode est pris en charge, n'est pas incluse.

6.4.4.3.1.7.18 Champ Physical Connection

Le champ Physical Connection (B10) ***Devoir/Doit/Doivent*** indiquer la construction du câble, c'est-à-dire si la connexion entre les éléments actifs est de type cuivre ou optique.

6.4.4.3.1.7.19 Champ Active element

Le champ Active Element (B9) ***Devoir/Doit/Doivent*** indiquer l'élément actif du câble, c'est-à-dire si l'élément actif est un circuit de retemporisation ou un circuit d'amplification.

6.4.4.3.1.7.20 Champ USB4 Supported

Le champ USB4 Supported (B8) doit indiquer si le câble prend ou non en charge le fonctionnement ***[USB4]***.

6.4.4.3.1.7.21 Champ USB 2.0 Hub Hops Consumed

Le champ USB 2.0 Hub Hops Consumed (B7…6) ***Devoir/Doit/Doivent*** indiquer le nombre de "sauts de hub" USB 2.0 perdus en raison de la durée de transmission du câble.

6.4.4.3.1.7.22 Champ USB 2.0 Supported

Le champ USB 2.0 Supported (B5) ***Devoir/Doit/Doivent*** indiquer si le câble prend ou non en charge les signaux ***[USB 2.0]*** uniquement.

6.4.4.3.1.7.23 Champ USB 3.2 Supported

Le champ USB3.2 Supported (B4) ***Devoir/Doit/Doivent*** indiquer si le câble prend ou non en charge les signaux ***[USB 3.2]*** SuperSpeed.

6.4.4.3.1.7.24 Champ USB Lanes Supported

Le champ USB Lanes Supported (B3) ***Devoir/Doit/Doivent*** indiquer si le câble prend en charge une ou deux voies de signaux ***[USB 3.2]*** SuperSpeed.

6.4.4.3.1.7.25 Champ Optically Isolated Active Cable

Le champ Optically Isolated Active Cable (B2) ***Devoir/Doit/Doivent*** indiquer si ce câble est ou non un câble actif à isolement optique (comme défini dans ***[USB Type-C 2.0]***). Les câbles actifs à isolement optique ***Devoir/Doit/Doivent*** utiliser un circuit de retemporisation comme élément actif et ne prennent en charge ni ***[USB 2.0]*** ni le transport VBUS.

6.4.4.3.1.7.26 Champ USB Gen

Le champ USB Gen (B0) ***Devoir/Doit/Doivent*** indiquer la génération de signaux prise en charge par le câble. Gen 1 ne ***Devoir/Doit/Doivent*** être utilisée que par les câbles ***[USB 3.2]***, comme indiqué par le champ USB 3.2 Supported. Gen 2 (ou supérieure) ***Pouvoir/Peut/Peuvent*** être utilisée par les câbles ***[USB 3.2]*** ou ***[USB4]***, comme indiqué par leurs champs de prise en charge respectifs. Lorsque Gen 2 ou supérieure est indiquée, le champ USB Highest Speed dans VDO1 ***Devoir/Doit/Doivent*** indiquer la génération effectivement prise en charge.

6.4.4.3.1.8 VDO d'adaptateur de mode alternatif

Le VDO Alternate Mode Adapter (AMA) défini dans la présente section ***Devoir/Doit/Doivent*** être envoyé lorsque le type de produit est un adaptateur de mode alternatif. Le Tableau 6-41 définit le VDO d'AMA qui ***Devoir/Doit/Doivent*** être envoyé.

Tableau 6-41 VDO d'AMA

Bit(s)	Champ	Description
B31…28	Version HW	0000b…1111b affectés par le propriétaire du VID
B27…24	Version de micrologiciel	0000b…1111b affectés par le propriétaire du VID
B23…21	Version de VDO	Numéro de version du VDO (pas la version de la présente spécification): • Version 1.0 = 000b Les valeurs 001b…111b sont ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisées
B20…8	***Réservé(s/ée/ées)***.	***Devoir/Doit/Doivent*** être défini sur 0.
B7…5	Alimentation V_{CONN}	Lorsque le champ V_{CONN} required est défini sur "Yes", l'alimentation V_{CONN} est nécessaire à l'adaptateur pour une fonctionnalité totale 000b = 1 W 001b = 1,5 W 010b = 2 W 011b = 3 W 100b = 4 W 101b = 5 W 110b = 6 W 111b = ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé Lorsque le champ V_{CONN} required est défini sur "No", ***Réservé(s/ée/ées)***, ***Devoir/Doit/Doivent*** être défini sur 0.
B4	V_{CONN} exigée	0 = Non 1 = Oui
B3	V_{BUS} exigée	0 = Non 1 = Oui
B2…0	USB Highest Speed	000b = ***[USB 2.0]*** uniquement 001b = ***[USB 3.2]*** Gen1 et USB 2.0 010b = ***[USB 3.2]*** Gen1, Gen2 et USB 2.0 011b = ***[USB 2.0]*** Billboard uniquement 100b…111b = ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé

6.4.4.3.1.8.1 Champ HW Version

Le champ HW Version (B31…28) contient une version matérielle affectée par le propriétaire du VID.

6.4.4.3.1.8.2 Champ FW Version

Le champ FW Version (B27…24) contient une version de micrologiciel affectée par le propriétaire du VID.

6.4.4.3.1.8.3 Champ VDO Version

Le champ VDO Version (B23…20) contient une version de VDO pour ce numéro de version de VDM. Ce champ indique le contenu attendu pour ce VDO.

6.4.4.3.1.8.4 Champ VCONN Required

Lorsque le champ V_{CONN} required indique que V_{CONN} est nécessaire, le champ V_{CONN} power ***Devoir/Doit/Doivent*** indiquer la puissance dont l'AMA a besoin pour fonctionner pleinement.

Le champ V_{CONN} required ***Devoir/Doit/Doivent*** indiquer si V_{CONN} est nécessaire à l'AMA pour fonctionner.

6.4.4.3.1.8.5 Champ VBUS Required

Le champ V_{BUS} required ***Devoir/Doit/Doivent*** indiquer si V_{BUS} est nécessaire à l'AMA pour fonctionner.

6.4.4.3.1.8.6 Champ USB Highest Speed

Le champ USB Highest Speed (B2…0) ***Devoir/Doit/Doivent*** indiquer si le câble ne prend en charge que ***[USB 2.0]*** , ou s'il prend également en charge ***[USB 3.2]*** Gen1, ou Gen1 et Gen2, ou ***[USB 2.0]*** Billboard uniquement.

6.4.4.3.1.9 VDO de dispositif USB alimenté par Vconn

Le VDO de dispositif USB alimenté par VCONN (VPD) défini dans la présente section ***Devoir/Doit/Doivent*** être envoyé lorsque le type de produit indiqué est celui de dispositif USB alimenté par VCONN. Le Tableau 6-42 définit le VDO d'VPD qui ***Devoir/Doit/Doivent*** être envoyé.

Tableau 6-42 VDO d'VPD

Bit(s)	Champ	Description
B31…28	Version HW	0000b…1111b affectés par le propriétaire du VID
B27…24	Version de micrologiciel	0000b…1111b affectés par le propriétaire du VID
B23…21	Version de VDO	Numéro de version du VDO (pas la version de la présente spécification): • Version 1.0 = 000b Les valeurs 001b…111b sont ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisées
B20…17	***Réservé(s/ée/ées)***	***Doit*** être défini sur 0.
B16…15	Tension VBUS maximale	Tension de câble VBUS maximale: 00b – 20 V 01b – 30 V 10b – 40 V 11b – 50 V
B14	Prise en charge du courant de charge simultanée	Bit Charge Through Support = 1b: 0b = capacité de 3 A; 1b = capacité de 5 A. Bit Charge Through Support = 0b: ***Réservé(s/ée/ées)***, ***Devoir/Doit/Doivent*** être défini sur 0.
B14…13	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B12…7	Impédance de VBUS	Bit Charge Through Support = 1b: Impédance de Vbus dans le VPD par incréments de 2 mΩ. Les valeurs inférieures à 10 mΩ sont Réservées et***Ne doit/doivent pas*** être utilisées. Bit Charge Through Support = 0b: ***Réservé(s/ée/ées), Devoir/Doit/Doivent*** être défini sur 0.
B6…1	Impédance de masse	Bit Charge Through Support = 1b: Impédance de masse dans le VPD par incréments de 1 mΩ. Les valeurs inférieures à 10 mΩ sont***Réservé(s/ée/ées)*** et***Ne doit/doivent pas*** être utilisées. Bit Charge Through Support = 0b: ***Réservé(s/ée/ées), Devoir/Doit/Doivent*** être défini sur 0.
B0	Prise en charge de la charge simultanée	1b – le VPD prend en charge la charge simultanée 0b – le VPD ne prend pas en charge la charge simultanée

6.4.4.3.1.9.1 Champ HW Version

Le champ HW Version (B31…28) contient une version matérielle affectée par le propriétaire du VID.

6.4.4.3.1.9.2 Champ FW Version

Le champ FW Version (B27…24) contient une version de micrologiciel affectée par le propriétaire du VID.

6.4.4.3.1.9.3 Champ VDO Version

Le champ VDO Version (B23…20) contient une version de VDO pour ce numéro de version de VDM. Ce champ indique le contenu attendu pour ce VDO.

6.4.4.3.1.9.4 Champ Maximum VBUS Voltage

Le champ Maximum VBUS Voltage (B16…15) ***Devoir/Doit/Doivent*** contenir la tension maximale qu'un destinataire ***Devoir/Doit/Doivent*** négocier à travers le port VPD en charge simultanée, dans le cadre d'un contrat explicite. La tension maximale qui est appliquée au câble est égale à ***vSrcNew*** max + ***vSrcValid***

max. Par exemple, lorsque le champ Maximum VBUS Voltage a pour valeur 20 V, une alimentation fixe de 20 V peut être négociée dans le cadre d'un contrat explicite, où la tension maximale absolue pouvant être appliquée au câble est de 21,55 V.

6.4.4.3.1.9.5 Champs VBUS Impedance

Le champ VBUS Impedance (B12…7) ***Devoir/Doit/Doivent*** contenir l'impédance que le VPD ajoute en série entre la source et le destinataire. Le destinataire ***Devoir/Doit/Doivent*** prendre cette valeur en compte lors de sa demande de courant, afin de ne pas dépasser la limite de chute de tension ohmique VBUS de 0,5 V entre lui et la source. Si le destinataire est en mesure de tolérer une chute de tension ohmique supérieure sur VBUS, il ***Pouvoir/Peut/Peuvent*** le faire.

6.4.4.3.1.9.6 Champ Ground Impedance

Le champ Ground Impedance (B6…1) ***Devoir/Doit/Doivent*** contenir l'impédance que le VPD ajoute en série entre la source et le destinataire. Le destinataire ***Devoir/Doit/Doivent*** prendre cette valeur en compte lors de sa demande de courant, afin de ne pas dépasser la limite de chute de tension ohmique de terre de 0,25 V entre lui et la source.

6.4.4.3.1.9.7 Champ Charge Through

Le champ Charge Through (B0) ***Devoir/Doit/Doivent*** être défini sur 1b lorsque le VPD prend en charge la charge simultanée, sur 0b sinon.

6.4.4.3.2 Découverte de SVID

La commande ***Discover SVIDs*** est utilisée par un initiateur afin de déterminer les SVID pour lesquels un répondeur dispose de modes. La commande ***Discover SVIDs*** s'utilise conjointement avec la commande ***Discover Modes*** dans les processus de découverte, pour déterminer les modes qu'un dispositif prend en charge. La liste des SVID se termine toujours par un ou deux SVID 0x0000.

L'initiateur et le répondeur de la commande ***Discover SVIDs*** ***Devoir/Doit/Doivent*** définir le SVID de la commande sur le ***PD SID*** (voir Tableau 6-27).

Le champ ***Number of Data Objects*** de l'en-tête de message de la demande de commande ***Discover SVIDs*** ***Devoir/Doit/Doivent*** être défini sur 1, car la demande de commande ***Discover SVIDs*** ***Ne doit/doivent pas*** contenir de VDO.

La confirmation de commande ***Discover SVIDs*** ACK renvoyée par le répondeur ***Devoir/Doit/Doivent*** contenir un ou plusieurs SVID. 2 SVID sont renvoyés par VDO (voir Tableau 6-43). En cas de nombre impair de SVID pris en charge, dans la dernière partie du dernier VDO, la commande ***Discover SVIDs*** renvoyée se termine par une valeur de SVID de 0x0000. En cas de nombre pair de SVID pris en charge, la commande ***Discover SVIDs*** renvoyée se termine par un VDO supplémentaire, contenant deux SVID de valeurs 0x0000. Un répondeur ***Devoir/Doit/Doivent*** uniquement renvoyer des SVID pour lesquels une demande de commande ***Discover Modes*** pour ces SVID renverra au moins un mode.

Un répondeur qui ne prend en charge aucun SVID ***Devoir/Doit/Doivent*** renvoyer une réponse NAK.

Le champ ***Number of Data Objects*** de l'en-tête de message des réponses de commande ***Discover SVIDs*** NAK et BUSY ***Devoir/Doit/Doivent*** être défini sur 1, car elles ***Ne doit/doivent pas*** contenir de VDO.

Si le répondeur prend en charge 12 SVID ou plus, la commande ***Discover SVIDs*** ***Devoir/Doit/Doivent*** être exécutée plusieurs fois, jusqu'à ce qu'un VDO de découverte de SVID renvoyé se termine par une valeur de SVID de 0x0000 dans la dernière partie du dernier VDO, ou avec un VDO qui contient deux SVID de valeur 0x0000. Chaque message ACK de découverte de SVID, outre celui qui contient le SVID 0x0000 final, ***Devoir/Doit/Doivent*** acheminer 12 SVID. Le répondeur ***Devoir/Doit/Doivent*** recommencer la liste de SVID à chaque fois qu'une demande de commande ***Discover Identity*** est reçue de l'initiateur.

Note: Comme une fiche de câble ne relance pas de messages, si le message ***GoodCRC*** de l'initiateur est corrompu, la fiche juge que la confirmation de commande ***Discover SVIDs*** ACK n'est pas envoyée, et renvoie la même liste de SVID.

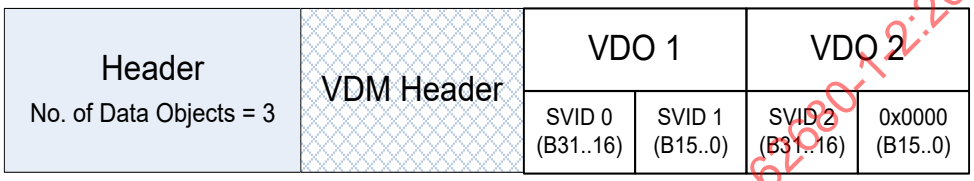
La Figure 6-17 donne un exemple de réponse à la demande de commande ***Discover SVIDs*** avec deux VDO contenant trois SVID. La Figure 6-18 représente un exemple de réponse, avec deux VDO qui contiennent

quatre SVID, suivis d'un VDO vide pour terminer la réponse. La Figure 6-19 représente un exemple de réponse avec six VDO, qui contiennent douze SVID suivis d'une demande supplémentaire qui renvoie un VDO vide indiquant qu'il n'y a plus de SVID à renvoyer.

Tableau 6-43 VDO de répondeur de découverte de SVID

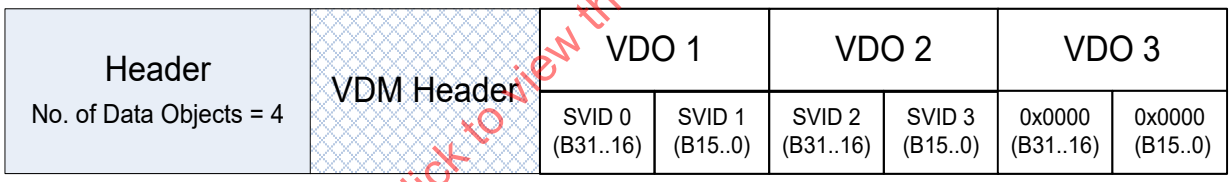
Bit(s)	Champ	Description
B31…16	SVID n	Entier non signé de 16 bits, affecté par l'USB-IF ou 0x0000 s'il s'agit du dernier VDO et que le répondeur prend en charge un nombre pair de SVID.
B15…0	SVID n+1	Entier non signé de 16 bits, affecté par l'USB-IF ou 0x0000 s'il s'agit du dernier VDO et que le répondeur prend en charge un nombre pair ou impair de SVID.

Figure 6-17 Exemple de réponse de découverte de SVID avec 3 SVID



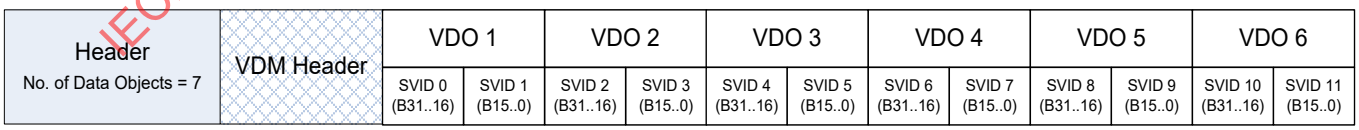
Anglais	Français
Header	En-tête
No. of Data Objects	Nombre d'objets de données
VDM Header	En-tête de VDM

Figure 6-18 Exemple de réponse de découverte de SVID avec 4 SVID



Anglais	Français
Header	En-tête
No. of Data Objects	Nombre d'objets de données
VDM Header	En-tête de VDM

Figure 6-19 Exemple de réponse de découverte de SVID avec 12 SVID, suivi par une réponse vide



Anglais	Français
Header	En-tête
No. of Data Objects	Nombre d'objets de données

VDM Header	En-tête de VDM

6.4.4.3.3 Découverte de modes

La commande ***Discover Modes*** permet à un initiateur de déterminer les modes qu'un répondeur prend en charge pour un SVID donné.

L'initiateur et le répondeur de la commande ***Discover Modes*** ***Devoir/Doit/Doivent*** définir le SVID de la commande sur le SVID correspondant aux modes demandés.

Le champ ***Number of Data Objects*** de l'en-tête de message de la demande de commande ***Discover Modes*** ***Devoir/Doit/Doivent*** être défini sur 1, car la demande de commande ***Discover Modes*** ***Ne doit/doivent pas*** contenir de VDO.

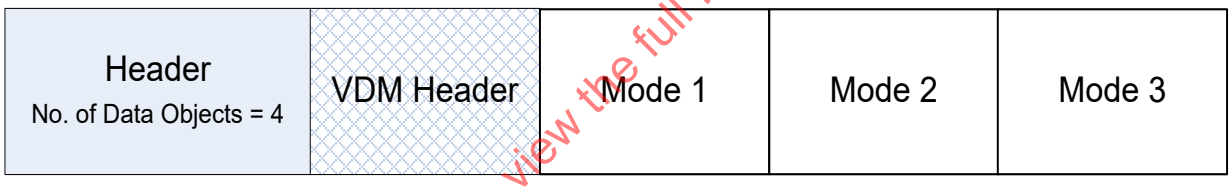
La confirmation de commande ***Discover Modes*** ACK renvoyée par le répondeur ***Devoir/Doit/Doivent*** contenir un ou plusieurs modes. La confirmation de commande ***Discover Modes*** ACK ***Devoir/Doit/Doivent*** contenir un en-tête de message dont le champ ***Number of Data Objects*** a une valeur comprise entre 1 et 7 (la valeur réelle correspond au nombre d'objets de mode plus un). Si l'ID est un VID, la structure et le contenu du VDO sont laissés au choix du fournisseur. Si l'ID est un SID, la structure et le contenu du VDO sont définis par la norme appropriée.

Un répondeur qui ne prend en charge aucun mode ***Devoir/Doit/Doivent*** renvoyer une réponse NAK.

Le champ ***Number of Data Objects*** de l'en-tête de message des réponses de commande ***Discover Modes*** NAK et BUSY ***Devoir/Doit/Doivent*** être défini sur 1, car elles ***Ne doit/doivent pas*** contenir de VDO.

La Figure 6-20 représente un exemple de réponse à la commande ***Discover Modes*** d'un répondeur qui prend en charge trois modes pour un SVID donné.

Figure 6-20 Exemple de réponse de découverte de modes pour un SVID à 3 modes



Anglais	Français
Header	En-tête
No. of Data Objects	Nombre d'objets de données
VDM Header	En-tête de VDM

6.4.4.3.4 Commande d'entrée dans un mode

La commande ***Enter Mode*** permet à un initiateur (port orienté en aval) d'ordonner à un répondeur (port orienté en amont ou fiche de câble) d'entrer dans un mode donné de fonctionnement. Seul un port orienté en aval ***Devoir/Doit/Doivent*** initier le processus d'entrée dans un mode, qui commence à l'issue du processus de découverte.

La valeur du champ Object Position de l'en-tête de VDM ***Devoir/Doit/Doivent*** indiquer le mode de la commande ***Discover Modes*** auquel le VDO se réfère (voir Figure 6-20). La valeur 1 indique toujours le premier mode, étant le premier objet suivant l'en-tête de VDM. La valeur 2 fait référence au mode suivant, et ainsi de suite.

Le champ ***Number of Data Objects*** de l'en-tête de message de la demande de commande ***Devoir/Doit/Doivent*** être défini sur 1 ou 2 à partir de l'***Enter Mode***, car la demande de commande ***Ne doit/doivent pas*** contenir plus d'un VDO. Lorsqu'un VDO est inclus dans une demande de commande ***Enter Mode***, le contenu du VDO de 32 bits est défini par le mode.

Le champ ***Number of Data Objects*** de la réponse de commande ***Devoir/Doit/Doivent*** être défini sur 1, car une réponse de commande ***Enter Mode*** (ACK, NAK, BUSY) ***Ne doit/doivent pas*** contenir de VDO.

Avant d'entrer dans un mode, en envoyant la demande de commande ***Enter Mode***, qui exige la reconfiguration de broches dès l'entrée dans ce mode, l'initiateur ***Devoir/Doit/Doivent*** s'assurer que ces broches en reconfiguration sont placées en état de sécurité USB. Avant d'entrer dans un mode qui exige la reconfiguration de broches, le répondeur ***Devoir/Doit/Doivent*** s'assurer que ces broches en cours de reconfiguration sont placées en fonctionnement USB ou en état de sécurité USB.

Un dispositif ***Pouvoir/Peut/Peuvent*** prendre en charge plusieurs modes, avec un ou plusieurs actifs à tout moment. Toute interaction entre eux relève de la responsabilité de la norme ou du fournisseur. Lorsque plusieurs modes actifs fonctionnent simultanément, le fonctionnement modal ***Devoir/Doit/Doivent*** démarrer dès l'entrée dans le premier mode.

Lors de la réception de la demande de commande ***Enter Mode***, le répondeur ***Devoir/Doit/Doivent*** répondre avec une réponse ACK ou NAK. Le répondeur n'est pas autorisé à renvoyer une réponse BUSY. La valeur du champ Object Position de la réponse de commande ***Enter Mode*** ***Devoir/Doit/Doivent*** contenir la même valeur que la demande de commande ***Enter Mode*** reçue.

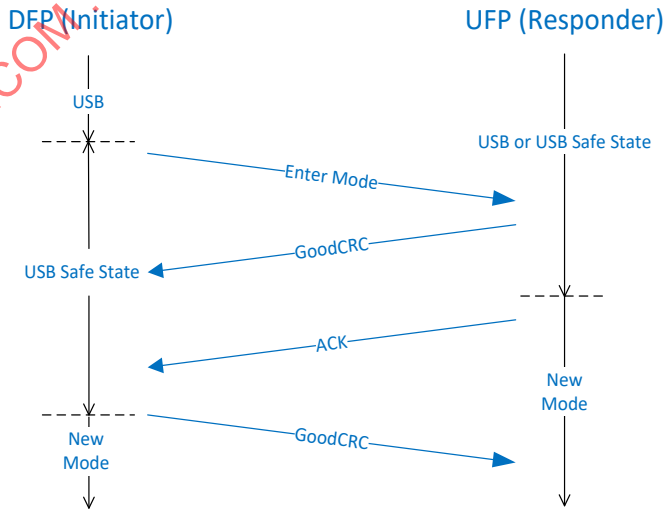
Si le répondeur répond à la demande de commande ***Enter Mode*** par un message ACK, le répondeur ***Devoir/Doit/Doivent*** entrer dans le mode avant l'envoi de ce message. L'initiateur ***Devoir/Doit/Doivent*** entrer dans le mode à la réception de la confirmation ACK. La réception du message ***GoodCRC*** correspondant à la confirmation ACK confirme au répondeur que l'initiateur est dans un mode actif et est prêt à fonctionner.

Si le répondeur répond à la demande de commande ***Enter Mode*** par un message NAK, le répondeur n'entre pas dans le mode. L'initiateur, s'il n'est pas actuellement en fonctionnement modal, ***Devoir/Doit/Doivent*** reprendre un fonctionnement USB. Le répondeur, s'il n'est pas actuellement en fonctionnement modal, ***Devoir/Doit/Doivent*** rester en fonctionnement USB ou à l'état de sécurité USB.

Si l'initiateur ne parvient pas à recevoir de réponse dans un délai ***tVDMWaitModeEntry***, il ***Ne doit/doivent pas*** entrer dans le mode, mais retourner au fonctionnement USB.

La Figure 6-21 représente la séquence d'événements lors de la transition entre le fonctionnement USB et l'entrée dans un mode. Elle indique les instants de changement de mode pour le répondeur et pour l'initiateur. La Figure 6-22 représente une séquence interrompue par un message ***Source_Capabilities***, qui achève une négociation de contrat, à la suite de quoi la séquence est réexécutée. La Figure 6-23 montre que lorsque le répondeur renvoie un message NAK, la transition vers un mode n'a pas lieu, et le répondeur et l'initiateur conservent leurs rôles USB par défaut.

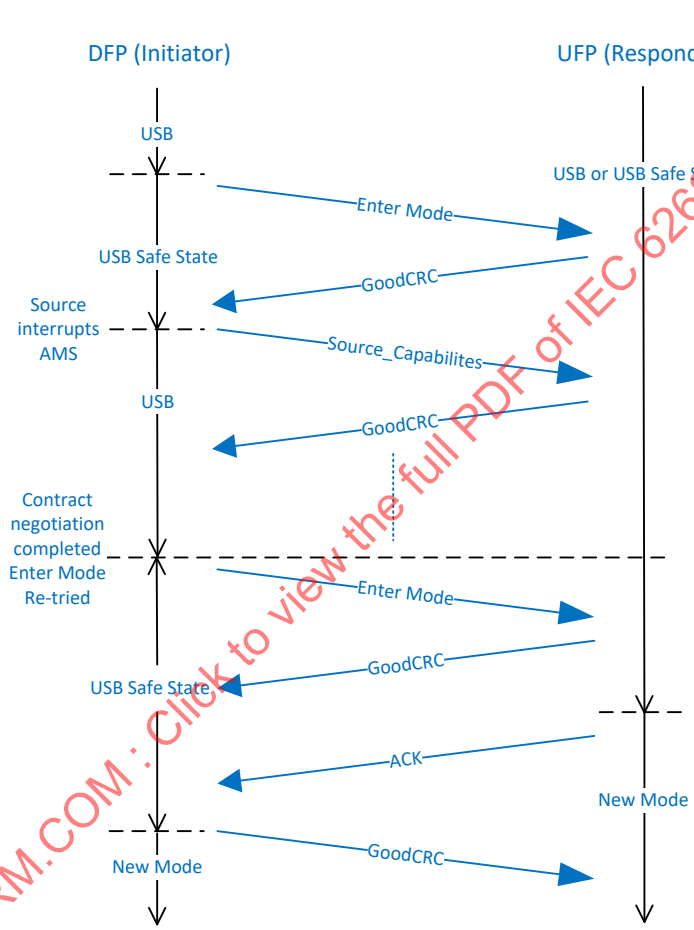
Figure 6-21 Séquence d'entrée dans un mode réussie



Anglais	Français

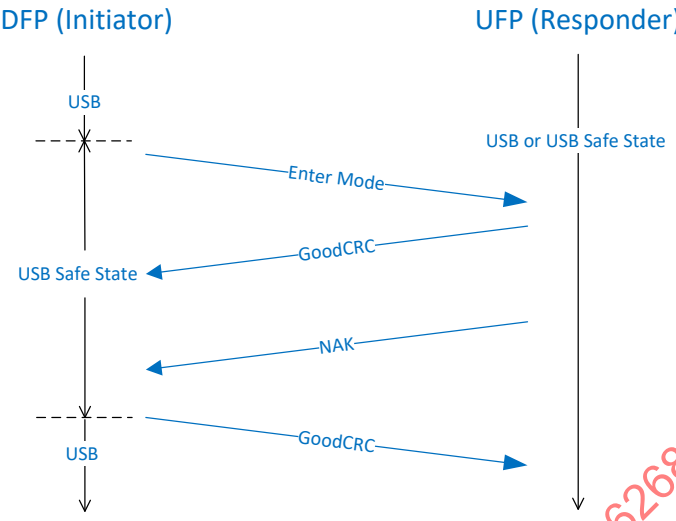
Initiator	Initiateur
Responder	Répondeur
USB or USB Safe State	USB ou état de sécurité USB
Enter Mode	Entrer dans un mode
New mode	Nouveau mode

Figure 6-22 Séquence d'entrée dans un mode interrompue par un message Source Capabilities, puis réexécutée



Anglais	Français
Initiator	Initiateur
Responder	Répondeur
USB or USB Safe State	USB ou état de sécurité USB
Enter Mode	Entrer dans un mode
Source interrupts AMS	La source interrompt l'AMS
Contract negotiation completed	Négociation du contrat terminée
Enter Mode Re-tried	Entrée dans le mode relancée
New Mode	Nouveau mode

Figure 6-23 Séquence d'entrée dans un mode échouée en raison d'un message NAK



Anglais	Français
Initiator	Initiateur
Responder	Répondeur
USB or USB Safe State	USB ou état de sécurité USB
Enter Mode	Entrer dans un mode

Lorsqu'il est entré dans le mode, le dispositif ***Devoir/Doit/Doivent*** rester dans ce mode actif jusqu'à réception de la commande ***Exit Mode*** (voir 6.4.4.3.5).

Les événements suivants ***Devoir/Doit/Doivent*** également faire sortir les ports partenaires et les fiches de câbles de tous les modes actifs:

- une réinitialisation matérielle de l'alimentation USB est réalisée;
- les ports partenaires ou les fiches de câbles sont débranchés;
- une réinitialisation de câble (ne fait quitter que les modes actifs des fiches de câbles) est réalisée.

L'initiateur ***Devoir/Doit/Doivent*** reprendre un fonctionnement USB dans un délai ***tVDMExitMode*** suivant une déconnexion ou la détection d'un signal ***Hard Reset***.

Le répondeur ***Devoir/Doit/Doivent*** reprendre un fonctionnement USB ou un état de sécurité USB dans un délai ***tVDMExitMode*** suivant une déconnexion ou la détection d'un signal ***Hard Reset***.

Un message ***DR_Swap*** ***Ne doit/doivent pas*** être envoyé au cours du fonctionnement modal, entre les ports partenaires (voir 6.3.9).

6.4.4.3.5 Commande de sortie de mode

La commande ***Exit Mode*** permet à un initiateur (DFP) d'ordonner à un répondeur (UFP ou fiche de câble) de quitter son mode actif et de reprendre un fonctionnement USB normal. Seul le port orienté en aval ***Devoir/Doit/Doivent*** initier le processus de sortie de mode.

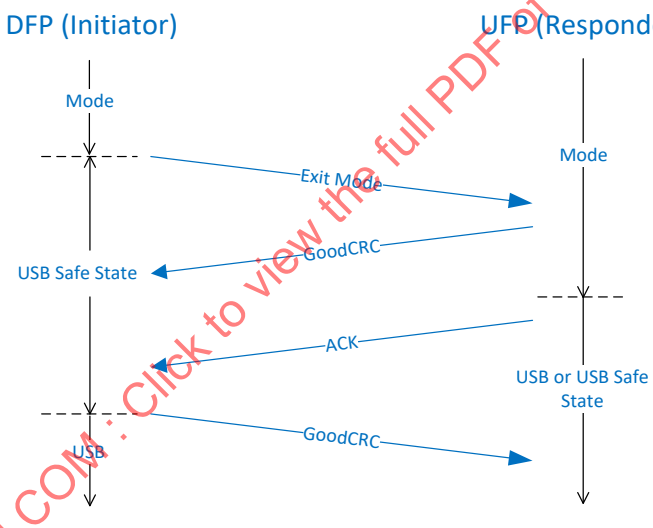
La valeur du champ Object Position ***Devoir/Doit/Doivent*** indiquer le mode de la commande ***Discover Modes*** auquel le VDO se réfère (voir Figure 6-20) et ***Devoir/Doit/Doivent*** avoir été précédemment utilisée dans une demande de commande ***Enter Mode*** pour un mode actif. La valeur 1 indique toujours le premier mode, étant le premier objet suivant l'en-tête de VDM. La valeur 2 fait référence au mode suivant, et ainsi de suite. Une valeur de 111b du champ Object Position ***Devoir/Doit/Doivent*** indiquer que tous les modes actifs ***Devoir/Doit/Doivent*** être quittés.

Le champ ***Number of Data Objects*** de la demande et de la réponse de commande (ACK, NAK, BUSY) ***Devoir/Doit/Doivent*** être défini sur 1, car une commande ***Exit Mode*** ***Ne doit/doivent pas*** contenir de VDO.

Le répondeur ***Devoir/Doit/Doivent*** quitter son mode actif avant d'envoyer le message de réponse. L'initiateur ***Devoir/Doit/Doivent*** quitter son mode actif avant d'envoyer le message ***GoodCRC*** en réponse à la confirmation ACK. La réception du message ***GoodCRC*** confirme au répondeur que l'initiateur est sorti du mode. Le répondeur ***Ne doit/doivent pas*** renvoyer d'acquittement BUSY et ***Devoir/Doit/Doivent*** uniquement renvoyer un acquittement NAK à une demande qui ne contient pas de mode actif (position d'objet ***Invalide(s)***). Un initiateur qui ne parvient pas à recevoir de confirmation ACK au cours de ***tVDMWaitModeExit*** ou qui reçoit une réponse NAK ou BUSY ***Devoir/Doit/Doivent*** quitter son mode actif.

La Figure 6-24 représente la séquence d'événements lors de la transition entre la sortie d'un mode actif et le fonctionnement USB. Elle indique les instants de changement de mode pour le répondeur et pour l'initiateur.

Figure 6-24 Séquence de sortie de mode



Anglais	Français
Initiator	Initiateur
Responder	Répondeur
USB or USB Safe State	USB ou état de sécurité USB
Exit Mode	Sortir d'un mode

6.4.4.3.6 Attention

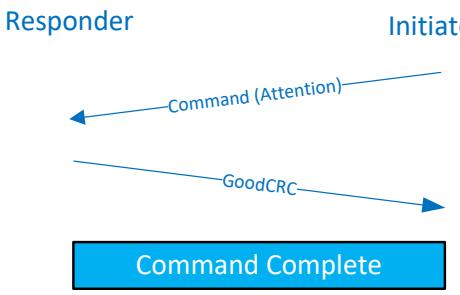
La commande ***Attention*** ***Pouvoir/Peut/Peuvent*** être utilisée par l'initiateur pour indiquer au répondeur qu'il exige un service.

La valeur du champ Object Position ***Devoir/Doit/Doivent*** indiquer le mode de la commande ***Discover Modes*** auquel le VDO se réfère (voir Figure 6-20) et ***Devoir/Doit/Doivent*** avoir été précédemment utilisée dans une demande de commande ***Enter Mode*** pour un mode actif. La valeur 1 indique toujours le premier mode, étant le premier objet suivant l'en-tête de VDM. La valeur 2 fait référence au mode suivant,

et ainsi de suite. La commande ***Attention*** ***Ne doit/doivent pas*** utiliser une valeur de 000 b ou de 111 b dans le champ Object Position.

Le champ ***Number of Data Objects*** de l'en-tête de message ***Devoir/Doit/Doivent*** être défini sur 1 ou 2, car la commande ***Attention*** ***Ne doit/doivent pas*** contenir plus d'un VDO. Lorsqu'un VDO est inclus dans une commande ***Attention***, le contenu du VDO de 32 bits est défini par le mode.

Figure 6-25 Séquence de demande/réponse de commande Attention

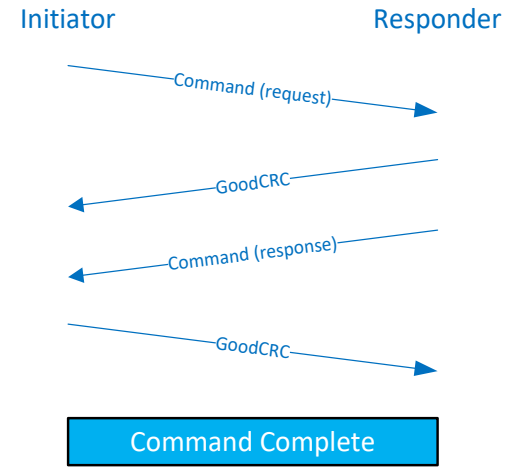


Anglais	Français
Initiator	Initiateur
Responder	Répondeur
Command (Attention)	Commande (Attention)
Command Complete	Commande terminée

6.4.4.4 Processus de commande

Le flux de messages des commandes lors d'un processus consiste en une requête suivie d'une réponse. Chaque demande de commande envoyée doit faire l'objet d'une réponse avec un message ***GoodCRC***. Le message ***GoodCRC*** indique seulement que la demande de commande a été correctement reçue, et non que le répondeur a compris ou simplement qu'il prend en charge un SVID donné. La Figure 6-26 représente la séquence de demande/réponse, y compris les messages ***GoodCRC***.

Figure 6-26 Séquence de demande/réponse de commande



Anglais	Français
Initiator	Initiateur

Responder	Répondeur
Command (Request)	Commande (Demande)
Command (Response)	Commande (Réponse)
Command Complete	Commande terminée

Pour que l'initiateur sache que la demande de commande a été effectivement assimilée, un acquittement du répondeur est nécessaire. Il existe trois réponses pour indiquer que le répondeur a reçu et traité la demande de commande:

- ACK;
- NAK;
- BUSY.

Le répondeur ***Devoir/Doit/Doivent*** satisfaire à ce qui suit:

- demandes d'entrée dans un mode, dans un délai ***tVDMEnterMode***;
- demandes de sortie de mode, dans un délai ***tVDMExitMode***;
- pour les autres demandes, dans un délai ***tVDMReceiverResponse***.

Un initiateur qui ne reçoit pas de réponse au cours des durées suivantes ***Devoir/Doit/Doivent*** expirer et reprendre son état ***PE_SRC_Ready*** ou ***PE_SNK_Ready*** (selon ce qui est approprié):

- demandes d'entrée dans un mode, dans un délai ***tVDMWaitModeEntry***;
- demandes de sortie de mode, dans un délai ***tVDMWaitModeExit***;
- pour les autres demandes, dans un délai ***tVDMSenderResponse***.

Le répondeur ***Devoir/Doit/Doivent*** répondre par:

- ACK s'il reconnaît le SVID et s'il peut le traiter à cet instant;
- NAK:
  - s'il reconnaît le SVID, mais ne peut pas traiter la demande de commande;
  - ou s'il ne reconnaît pas le SVID;
  - ou s'il ne prend pas en charge la commande;
  - ou si un VDO ***Invalide(s).*** a été fourni;
- BUSY s'il reconnaît le SVID et la commande, mais ne peut pas traiter la demande de commande à cet instant.

La réponse ACK, NAK ou BUSY ***Devoir/Doit/Doivent*** contenir le même SVID que la demande de commande.

6.4.4.4.1 Processus de découverte

L'initiateur (en général le port orienté en aval) commence toujours le processus de découverte. Ce dernier consiste en deux phases. Dans la première, la demande de commande ***Discover SVIDs*** est envoyée par l'initiateur pour obtenir la liste des SVID que le répondeur prend en charge. Dans la seconde phase, l'initiateur envoie une demande de commande ***Discover Modes*** pour chaque SVID que l'initiateur et le répondeur prennent tous deux en charge.

6.4.4.4.2 Processus d'entrée dans un mode fournisseur/de sortie de mode fournisseur

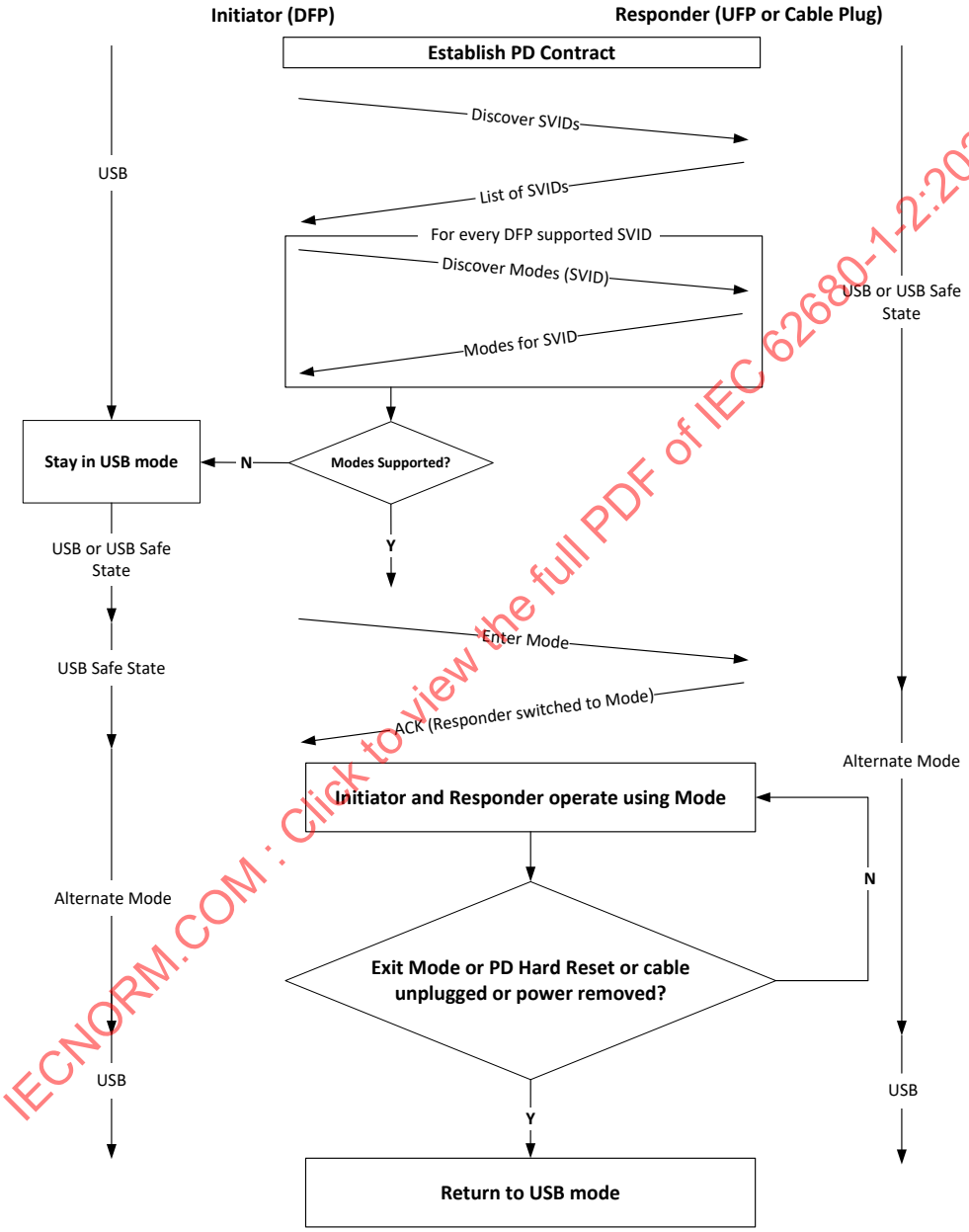
Par le biais du processus de découverte, l'initiateur et le répondeur identifient tous deux les modes qu'ils prennent mutuellement en charge. L'initiateur (port orienté en aval), lorsqu'il trouve un mode approprié, utilise la commande ***Enter Mode*** pour activer le mode.

Le répondeur (UFP ou fiche de câble) et l'initiateur continuent d'utiliser le mode actif jusqu'à ce qu'ils le quittent. Lors d'un achèvement maîtrisé, à l'aide de la commande ***Exit Mode***, le mode actif ***Devoir/Doit/Doivent*** être quitté d'une façon contrôlée, décrite en 6.4.4.3.5. Lors d'un achèvement non

maîtrisé, déclenché par une réinitialisation matérielle de l'alimentation (envoi d'un signal ***Hard Reset*** par l'un des ports partenaires) ou par le débranchement d'un câble, le mode actif ***Devoir/Doit/Doivent*** tout de même être quitté, mais la transition vers l'état de sécurité USB ***Ne doit/doivent pas*** être effectuée. Dans les terminaisons gérées comme non gérées, à la sortie d'un mode, l'initiateur et le répondeur reprennent un fonctionnement USB, comme défini dans ***[USB Type-C 2.0]***.

Le flux global de messages est représenté à la Figure 6-27.

Figure 6-27 Processus d'entrée dans un mode/de sortie de mode



Anglais	Français
Initiator	Initiateur
Responder	Répondeur
UFP or Cable Plug	UFP ou fiche de câble
USB or USB Safe State	USB ou état de sécurité USB
Establish PD Contract	Etablir un contrat PD

Discover SVIDs	Découverte de SVDI
List of SVIDs	Liste de SVID
For every supported SVIDs	Pour tous les SVID pris en charge
Discover Modes (SVID)	Découverte de modes (SVID)
Modes for SVID	Modes pour SVID
Modes supported?	Modes pris en charge?
Stay in USB mode	Rester dans le mode USB
Enter Mode	Entrer dans un mode
ACK (Responder switched to Mode)	ACK (Répondeur commuté vers le mode)
Alternate Mode	Mode alternatif
Initiator and Responder operate using Mode	L'initiateur et le répondeur fonctionnent en utilisant le mode
Exit Mode or PD Hard Reset or cable unplugged or power removed?	Sortie du mode ou réinitialisation matérielle de l'alimentation ou câble débranché ou alimentation retirée?
Return to USB mode	Retourner au mode USB

6.4.4.5 Temporisation des messages VDM et messages d'alimentation normaux

Tout processus de commande ou autre séquence VDM ***Pouvoir/Peut/Peuvent*** être interrompu(e) par tout autre message d'alimentation USB. Le fonctionnement d'état défini par le fournisseur ou les normes ***Devoir/Doit/Doivent*** prendre cela en compte et se poursuivre comme attendu lors du traitement de tout autre message d'alimentation USB.

La dispersion dans le temps des VDM entre les messages d'alimentation USB réguliers ***Devoir/Doit/Doivent*** être telle que les séquences de ces messages ne soient pas perturbées. Cette exigence ***Devoir/Doit/Doivent*** s'appliquer aux VDM non structurés comme aux VDM structurés.

L'utilisation d'un VDM structuré par un initiateur ***Ne doit/doivent pas*** interférer avec les exigences temporelles d'un message normal d'alimentation USB, et ni l'initiateur ni le répondeur ne ***Devoir/Doit/Doivent*** interrompre une séquence de messages d'alimentation USB (par exemple, négociation de puissance, permutation des rôles d'alimentation, permutation de rôles de données, etc.). L'utilisation de VDM non structurés ***Ne doit/doivent pas*** interférer avec les exigences temporelles d'un message d'alimentation USB.

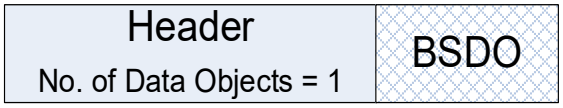
Les séquences VDM ***Devoir/Doit/Doivent*** être interruptibles après le renvoi d'un message ***GoodCRC***. Dans le cas où une erreur se produit lors de la transmission du message ***Vendor_Defined***, comme pour tout autre message d'alimentation USB, la tentative d'émission du message ***Vendor_Defined*** n'est pas renouvelée, au lieu de quoi le message entrant est traité par le moteur de politique. Cela signifie qu'il est nécessaire de réexécuter la séquence de messages ***Vendor_Defined*** à l'issue de la séquence de messages d'alimentation USB.

6.4.5 Message Battery_Status

Le message ***Battery_Status*** ***Devoir/Doit/Doivent*** être envoyé en réponse au message ***Get_Battery_Status***. Le message ***Battery_Status*** contient un objet de données de statut de batterie (BSDO) pour l'une des batteries qu'il prend en charge, comme indiqué par le champ Battery du message ***Source_Capabilities_Extended***. Le BSDO renvoyé ***Devoir/Doit/Doivent*** correspondre à la batterie demandée dans le champ ***Battery Status Ref*** contenu dans le message ***Get_Battery_Status***.

Le message ***Battery_Status*** renvoie un BSDO dont le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-28 et au Tableau 6-44. Le champ ***Number of Data Objects*** du message ***Battery_Status*** ***Devoir/Doit/Doivent*** être défini sur 1.

Figure 6-28 Message Battery_Status



Anglais	Français
Header	En-tête
No. of Data Objects	Nombre d'objets de données

Tableau 6-44 Objet de données de statut de batterie (BSDO)

Bit(s)	Champ	Description
B31…16	Capacité actuelle de batterie	Etat de charge (SoC) de la batterie en incréments de 0,1 Wh Note: 0xFFFF = SoC de la batterie inconnu
B15…8	Informations sur la batterie	<table><tr><th>Bit</th><th>Description</th></tr><tr><td>0</td><td>Référence de batterie non valide</td></tr><tr><td>1</td><td>Lorsqu'il est défini, la batterie est présente</td></tr><tr><td>3…2</td><td>En présence de batterie ***Devoir/Doit/Doivent*** contenir le statut de charge de la batterie: 00b: Batterie en charge 01b: Batterie en cours de décharge 10b: Batterie inactive 11b: ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé Lorsque la batterie est absente: 11b…00b: ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé</td></tr><tr><td>7…4</td><td>***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0.</td></tr></table>
B7…0	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.

6.4.5.1 Capacité actuelle de batterie

Le champ Battery Present Capacity ***Devoir/Doit/Doivent*** renvoyer l'état de charge (SoC) de la batterie en dixièmes de Wh ou indiquer que le SoC actuel de la batterie est inconnu.

6.4.5.2 Informations sur la batterie

Le champ Battery Info ***Devoir/Doit/Doivent*** être utilisé pour consigner des informations supplémentaires sur le statut actuel de la batterie. Les bits du champ Battery Info ***Devoir/Doit/Doivent*** refléter les conditions actuelles sous lesquelles la batterie fonctionne dans les systèmes.

6.4.5.2.1 Référence de batterie non valide

Le bit Invalid Battery Reference ***Devoir/Doit/Doivent*** être défini lorsque le message ***Get_Battery_Status*** contient une référence à une batterie ou des emplacements de batterie inexistants (voir Section 6.5.1.13).

6.4.5.2.2 Battery is Present

Le bit Battery is Present ***Devoir/Doit/Doivent*** être défini dès lors que la batterie est présente. Il ***Devoir/Doit/Doivent*** toujours être défini pour les batteries qui ne sont pas remplaçables à chaud. Pour les batteries remplaçables à chaud, le bit Battery is Present ***Devoir/Doit/Doivent*** indiquer si la batterie est branchée ou débranchée.

6.4.5.2.3 Statut de charge de la batterie

Les bits de statut de charge de la batterie indiquent si la batterie est en charge, se décharge, ou si elle est inactive (elle n'est ni en charge, ni en cours de décharge). Ces bits ***Devoir/Doit/Doivent*** être définis lorsque le bit Battery is present est défini. Sinon, lorsque le bit Battery is present vaut zéro, les bits de statut de charge de la batterie ***Devoir/Doit/Doivent*** également valoir zéro.

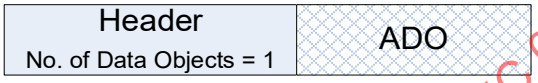
6.4.6 Message d'alerte

Le message ***Alert*** permet aux ports partenaires de s'informer l'un l'autre en cas d'événement relatif à une modification de statut. Certains événements sont essentiels, comme OCP, OVP et OTP, tandis que d'autres sont informatifs, par exemple lorsque la batterie passe du statut "en charge" au statut "inactive".

Le message ***Alert*** ne ***Devoir/Doit/Doivent*** être envoyé que lorsque la source ou le destinataire détecte un changement de statut.

Le message ***Alert*** ***Devoir/Doit/Doivent*** contenir exactement un objet de données d'alerte (ADO) et le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-29 et au Tableau 6-45.

Figure 6-29 Message d'alerte



Anglais	Français
Header	En-tête
No. of Data Objects	Nombre d'objets de données

Tableau 6-45 Objet de données d'alerte

Bit(s)	Champ	Description
B31…24	***Type of Alert***	<table><tr><th>Bit</th><th>Description</th></tr><tr><td>0</td><td>***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0.</td></tr><tr><td>1</td><td>Evénement de changement de statut de batterie (branchée/débranchée/en charge/en cours de décharge/inactive)</td></tr><tr><td>2</td><td>Evénement OCP lorsque défini (source uniquement, pour le destinataire: ***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0)</td></tr><tr><td>3</td><td>Evénement OTP lorsque défini</td></tr><tr><td>4</td><td>Changement de condition de fonctionnement lorsque défini</td></tr><tr><td>5</td><td>Evénement de modification d'entrée de source lorsque défini</td></tr><tr><td>6</td><td>Evénement OVP lorsque défini</td></tr><tr><td>7</td><td>***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0.</td></tr></table>
B23…20	***Fixed Batteries***	Lorsque le bit Battery Status Change de la batterie est défini, indique quelles batteries fixes ont changé de statut. B20 correspond à la batterie 0 et B23 correspond à la batterie 3.
B19…16	***Hot Swappable Batteries***	Lorsque le bit Battery Status Change de la batterie est défini, indique quelles batteries remplaçables à chaud ont changé de statut. B16 correspond à la batterie 4 et B19 correspond à la batterie 7.
B15…0	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.

6.4.6.1 Type of Alert

Le champ ***Type of Alert*** ***Devoir/Doit/Doivent*** être utilisé pour consigner les changements de statut de la source ou du récepteur. Seul un message ***Alert*** ***Devoir/Doit/Doivent*** être généré par événement ou changement. En revanche, plusieurs bits Type of Alert ***Pouvoir/Peut/Peuvent*** être définis dans un message ***Alert***. Lorsque le message ***Alert*** a été envoyé, le champ ***Type of Alert*** ***Devoir/Doit/Doivent*** être effacé.

Il convient d'/de/qu'/que un message ***Get_Battery_Status*** soit envoyé en réponse à un changement de statut de la batterie dans un message ***Alert***, afin d'obtenir les détails du changement.

Il convient d'/de/qu'/que un message ***Get_Status*** soit envoyé en réponse à un changement qui ne concerne pas le statut de la batterie dans un message ***Alert***, afin d'obtenir les détails du changement.

6.4.6.1.1 Changement de statut de la batterie

Le bit Battery Status Change ***Devoir/Doit/Doivent*** être défini lors de tout changement d'état d'alimentation de la batterie, entre les statuts "en charge", "en cours de décharge", ou ni l'un ni l'autre ("inactive"). Pour les batteries remplaçables à chaud, il ***Devoir/Doit/Doivent*** également être défini si la batterie est branchée ou débranchée.

6.4.6.1.2 Evénement de protection contre les surintensités

Le bit Over-Current Protection Event ***Devoir/Doit/Doivent*** être défini lorsqu'une source détecte que son courant de sortie dépasse ses limites, ce qui actionne son circuit de protection. Pour un destinataire, ce bit est ***Réservé(s/ée/ées)***.

6.4.6.1.3 Evénement de protection contre les surchauffes

Le bit Over-Temperature Protection Event ***Devoir/Doit/Doivent*** être défini lorsqu'une source ou un destinataire s'éteint en raison d'une surchauffe qui actionne son circuit de protection.

6.4.6.1.4 Changement de condition de fonctionnement

Le bit Operating Condition Change ***Devoir/Doit/Doivent*** être défini lorsqu'une source ou un destinataire détecte que sa condition de fonctionnement entre ou sort d'un état de température "avertissement" ou "surchauffe".

Le bit Operating Condition Change ***Devoir/Doit/Doivent*** être défini lorsque la source qui exploite le mode d'alimentation électrique programmable détecte que sa condition de fonctionnement est passée de tension constante (CV) à limite de fonctionnement (CL), ou inversement.

6.4.6.1.5 Evénement de modification d'entrée de source

Le bit Source Input Event ***Devoir/Doit/Doivent*** être défini lorsque l'entrée de la source/du destinataire est modifiée. Par exemple, lorsque l'entrée de courant alternatif est retirée et que la source/le destinataire continue d'être alimenté(e) par une ou plusieurs de ses batteries, ou lorsque le courant alternatif revient et que la source/le destinataire passe d'un fonctionnement par batterie à un fonctionnement en courant alternatif, ou encore lorsque la source/le destinataire passe d'un fonctionnement avec une ou plusieurs batteries à un fonctionnement avec une ou plusieurs autres batteries.

6.4.6.1.6 Evénement de protection contre les surtensions

Le bit Over-Voltage Protection Event ***Devoir/Doit/Doivent*** être défini lorsqu'un destinataire détecte que sa tension de sortie dépasse ses limites, ce qui actionne son circuit de protection.

Le bit Over-Voltage Protection Event ***Pouvoir/Peut/Peuvent*** être défini lorsqu'une source détecte que sa tension de sortie dépasse ses limites, ce qui actionne son circuit de protection.

6.4.6.2 Fixed Batteries

Le champ ***Fixed Batteries*** indique quelles batteries fixes ont changé de statut. B20 correspond à la batterie 0 t B23 correspond à la batterie 3.

Lorsque le message ***Alert*** a été envoyé, le champ ***Fixed Batteries*** ***Devoir/Doit/Doivent*** être effacé.

6.4.6.3 Hot Swappable Batteries

Le champ ***Hot Swappable Batteries*** indique quelles batteries remplaçables à chaud ont changé de statut. B16 correspond à la batterie 0 et B19 correspond à la batterie 3.

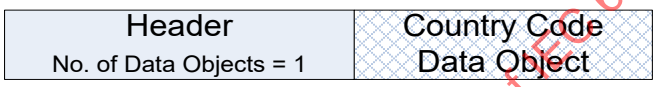
Lorsque le message ***Alert*** a été envoyé, le champ ***Hot Swappable Batteries*** ***Devoir/Doit/Doivent*** être effacé.

6.4.7 Message Get_Country_Info

Le message ***Get_Country_Info*** ***Devoir/Doit/Doivent*** être envoyé par un port pour obtenir des informations relatives au pays auprès de son port partenaire, à l'aide du code pays Alpha-2 spécifique au pays, défini par l'***[ISO 3166]***. Le port partenaire répond par un message ***Country_Info*** qui contient des informations relatives au pays. Le format du message ***Get_Country_Info*** ***Devoir/Doit/Doivent*** être conforme aux indications de la Figure 6-30 et du Tableau 6-46.

Par exemple, si la demande concerne des informations relatives à la Chine, alors l'objet de données de code pays est CCDO[31:0] = 434E0000h pour le code pays "CN".

Figure 6-30 Message Get_Country_Info



Anglais	Français
Header	En-tête
No. of Data Objects	Nombre d'objets de données
Country Code	Code pays
Data Objects	Objets de données

Tableau 6-46 Objet de données de code pays

Bit(s)	Description
B31…24	Premier caractère du code pays Alpha-2 défini par l'***[ISO 3166]***
B23…16	Second caractère du code pays Alpha-2 défini par l'***[ISO 3166]***
B15…0	***Réservé(s/ée/ées)***, ***Devoir/Doit/Doivent*** être défini sur 0.

6.4.8 Message Enter_USB

Le message ***Enter_USB*** ***Devoir/Doit/Doivent*** être envoyé par le DFP à son port partenaire UFP et aux fiches de câble, en présence d'un contrat explicite, pour entrer dans un mode de fonctionnement USB spécifié. Le destinataire du message ***Devoir/Doit/Doivent*** répondre en envoyant soit un message ***Accept*** en réponse à une demande ***Valide(s)***, soit un message ***Reject*** en réponse à une demande ***Invalide(s)***.

Le message ***Enter_USB*** ***Devoir/Doit/Doivent*** être envoyé par le ou les DFP d'un hub PDUSB ***[USB4]*** ou par le ou les DFP d'un hôte PDUSB ***[USB4]*** dans un délai ***tEnterUSB*** après la mise sous tension initiale ou après une réinitialisation des données pour entrer en mode de fonctionnement ***[USB4]***.

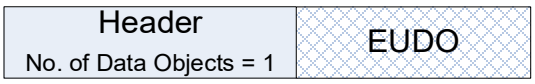
Le message ***Enter_USB*** ***Pouvoir/Peut/Peuvent*** être envoyé par le ou les DFP d'un hub PDUSB ou par le ou les DFP d'un hôte PDUSB dans un délai ***tEnterUSB*** après la mise sous tension initiale ou après une réinitialisation des données pour entrer en mode de fonctionnement ***[USB 3.2]*** ou ***[USB 2.0]***.

Le message ***Enter_USB*** ***Devoir/Doit/Doivent*** être utilisé par le ou les DFP d'un hub PDUSB pour former de façon spéculative les liaisons USB ou entrer dans les modes alternatifs ***[DPTC1.0]***ou ***[TBT3]*** avant la présence d'un hôte. Dans ce cas, le bit Host Present ***Devoir/Doit/Doivent*** être effacé. Lorsque l'hôte est

connecté, le message ***Enter_USB*** ***Devoir/Doit/Doivent*** être présent avec le bit Host Present défini. L'objet de données Enter USB (EUDO) du message ***Enter_USB*** reçu du hub racine lorsque l'hôte USB est connecté ***Devoir/Doit/Doivent*** être propagé dans l'arborescence du hub.

Voir les exigences de connexion au hub USB4 ***[USB Type-C 2.0]***.

Figure 6-31 Message Enter_USB



Anglais	Français
Header	En-tête
No. of Data Objects	Nombre d'objets de données

Tableau 6-47 Objet de données Enter_USB

Bit(s)	Champ	Description
B31	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B30…28	***USB Mode***[1]	000b: ***[USB 2.0]*** 001b: ***[USB 3.2]*** 010b: ***[USB4]*** 111b…011b: ***Réservé(s/ée/ées)***, ne ***Devoir/Doit/Doivent*** pas être utilisé
B27	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B26	***USB4 DRD***[2]	0b: Non apte à fonctionner en tant que dispositif ***[USB4]*** 1b: Apte à fonctionner en tant que dispositif ***[USB4]***
B25	***USB3 DRD***[2]	0b: Non apte à fonctionner en tant que dispositif ***[USB 3.2]*** 1b: Apte à fonctionner en tant que dispositif ***[USB 3.2]***
B24	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
B23…21	***Cable Speed***[2]	000b: ***[USB 2.0]*** uniquement, pas de prise en charge SuperSpeed 001b: ***[USB 3.2]*** Gen1 010b: ***[USB 3.2]*** Gen2 et ***[USB4]*** Gen2 011b: ***[USB4]*** Gen3 111b…100b: ***Réservé(s/ée/ées)***, ne ***Devoir/Doit/Doivent*** pas être utilisé
B20…19	***Cable Type***[2]	00b: Passif 01b: Circuit de retemporisation actif 10b: Circuit d'amplification actif 11b: A isolation optique
B18…17	***Cable Current***[2]	00b = VBUS n'est pas pris en charge 01b = ***Réservé(s/ée/ées)*** 10b = 3A 11b = 5A
B16	***PCIe Support***[2]	Tunnellisation PCIe ***[USB4]*** prise en charge par l'hôte
B15	***DP Support***[2]	Tunnellisation DP ***[USB4]*** prise en charge par l'hôte
B14	***TBT Support***[2]	***[TBT3]*** est pris en charge par le gestionnaire de connexions USB4 de l'hôte
B13	***Host Present***[2]	Connecté à un hôte. Lorsque ce bit est défini, ***PCIe Support***, ***DP Support*** et ***TBT Support*** représentent les capacités de l'hôte qui ***Devoir/Doit/Doivent*** être propagées dans l'arborescence du hub.
B12…0	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
Note 1: Une entrée en mode ***[USB 3.2]*** et ***[USB4]*** inclut une entrée en mode ***[USB 2.0]***. Note 2: ***Devoir/Doit/Doivent*** être ***Ignoré(s/ée/ées)*** si la réception passe par une fiche de câble (par exemple, SOP' ou SOP").		

6.4.8.1 Champ USB Mode

Le champ ***USB Mode*** ***Devoir/Doit/Doivent*** être utilisé par le DFP pour indiquer le mode USB dans lequel le port partenaire doit entrer.

6.4.8.2 Champ USB4 DRD

Le champ ***USB4 DRD*** ***Devoir/Doit/Doivent*** être défini lorsque le DFP hôte est capable de fonctionner en tant que dispositif ***[USB4]***. Un DFP hôte ***[USB4]*** qui définit le champ ***USB4 DRD*** ***Devoir/Doit/Doivent*** également être capable de fonctionner en tant que dispositif ***[USB 2.0]***.

6.4.8.3 Champ USB3 DRD

Le champ ***USB3 DRD*** ***Devoir/Doit/Doivent*** être défini lorsque le DFP hôte est capable de fonctionner en tant que dispositif ***[USB 3.2]***. Un DFP hôte ***[USB 3.2]*** qui définit le champ ***USB3 DRD*** ***Devoir/Doit/Doivent*** également être capable de fonctionner en tant que dispositif ***[USB 2.0]***.

6.4.8.4 Champ Cable Speed

Le champ ***Cable Speed*** ***Devoir/Doit/Doivent*** être utilisé pour indiquer la vitesse maximale du câble.

6.4.8.5 Champ Cable Type

Le champ ***Cable Type*** ***Devoir/Doit/Doivent*** être utilisé pour indiquer si le câble est passif ou actif. De plus, si le câble est actif, il précise le type de circuits actifs dans le câble et indique si le câble est de type à isolement optique.

6.4.8.6 Champ Cable Current

Le champ ***Cable Current*** ***Devoir/Doit/Doivent*** être utilisé pour indiquer la capacité de transport de courant du câble.

6.4.8.7 Champ PCIe Support

Le champ ***PCIe Support*** ***Devoir/Doit/Doivent*** être défini lorsque le DFP hôte est capable de prendre en charge la tunnellisation PCIe sur ***[USB4]***.

Le champ ***PCIe Support*** ***Pouvoir/Peut/Peuvent*** être défini de façon spéculative lorsque le DFP du hub est capable de prendre en charge la tunnellisation PCIe sur ***[USB4]***.

6.4.8.8 Champ DP Support

Le champ ***DP Support*** ***Devoir/Doit/Doivent*** être défini lorsque le DFP hôte est capable de prendre en charge la tunnellisation DP sur ***[USB4]***.

Le champ ***DP Support*** ***Pouvoir/Peut/Peuvent*** être défini de façon spéculative lorsque le DFP du hub est capable de prendre en charge la tunnellisation DP sur ***[USB4]***.

6.4.8.9 Champ TBT Support

Le champ ***TBT Support*** ***Devoir/Doit/Doivent*** être défini lorsque le DFP hôte est capable de prendre en charge la tunnellisation Thunderbolt™ sur ***[USB4]*** et que le gestionnaire de connexions (CM, *Connection Manager*) prend en charge la découverte et la configuration des dispositifs Thunderbolt™ 3 connectés au DFP des hubs ***[USB4]***.

Le champ ***TBT Support*** ***Pouvoir/Peut/Peuvent*** être défini de façon spéculative lorsque le DFP du hub est capable de prendre en charge la tunnellisation Thunderbolt sur ***[USB4]***.

6.4.8.10 Champ Host Present

Le champ ***Host Present*** ***Devoir/Doit/Doivent*** indiquer qu'un hôte est présent en amont.

6.5 Message étendu

Un message étendu ***Devoir/Doit/Doivent*** contenir un en-tête de message étendu (indiqué par le champ ***Extended*** dans l'en-tête de message alors défini), et être suivi par zéro ou par plusieurs octets de données.

Le format du message étendu est défini par le champ ***Message Type*** de l'en-tête du message et est résumé dans le Tableau 6-48. La colonne "Envoyé par" indique les entités qui ***Pouvoir/Peut/Peuvent*** envoyer le message concerné (source, destinataire ou fiche de câble). Les entités qui ne figurent pas dans la liste ***Ne doit/doivent pas*** émettre le message correspondant. La colonne "Début de paquet valide" indique les messages qui ***Devoir/Doit/Doivent*** uniquement être émis dans des paquets SOP et les messages qui ***Pouvoir/Peut/Peuvent*** être émis dans des paquets SOP*.

Tableau 6-48 Types de messages étendus

Bits 4...0	Type	Envoyé par	Description	Début de paquet valide
0 0000	***Réservé(s/ée/ées)***		Toutes les valeurs non définies explicitement sont ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisées.	
0 0001	***Source_Capabilities_Extended***	Source ou alimentation double fonction	Voir Section 6.5.1	SOP uniquement
0 0010	***Statut***	Source ou destinataire	Voir Section 6.5.2	SOP*
0 0011	***Get_Battery_Cap***	Source ou destinataire	Voir Section 6.5.3	SOP uniquement
0 0100	***Get_Battery_Status***	Source ou destinataire	Voir Section 6.5.4	
0 0101	***Battery_Capabilities***	Source ou destinataire	Voir Section 6.5.5	SOP uniquement
0 0110	***Get_Manufacturer_Info***	Source ou destinataire	Voir Section 6.5.6	SOP*
0 0111	***Manufacturer_Info***	Source, destinataire ou fiche de câble	Voir Section 6.5.7	SOP*
0 1000	***Security_Request***	Source ou destinataire	Voir Section 6.5.8.1	SOP*
0 1001	***Security_Response***	Source, destinataire ou fiche de câble	Voir Section 6.5.8.2	SOP*
0 1010	***Firmware_Update_Request***	Source ou destinataire	Voir Section 6.5.9.1	SOP*
0 1011	***Firmware_Update_Response***	Source, destinataire ou fiche de câble	Voir Section 6.5.9.2	SOP*
0 1100	***PPS_Status***	Source	Voir Section 6.5.10	SOP uniquement
0 1101	***Country_Info***	Source ou destinataire	Voir Section 6.5.12	SOP uniquement
0 1110	***Country_Codes***	Source ou destinataire	Voir Section 6.5.11	SOP uniquement
0 1111	***Sink_Capabilities_Extended***	Destinataire ou alimentation double fonction	Voir Section 6.5.13	SOP uniquement

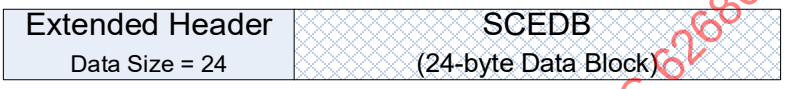
Bits 4...0	Type	Envoyé par	Description	Début de paquet valide
1 0000 - 1 1111	***Réservé(s/ée/ées)***		Toutes les valeurs non définies explicitement sont ***Réservé(s/ée/ées)*** et ***Ne doivent pas*** être utilisées.	

6.5.1 Message Source_Capabilities_Extended

Il convient d'/de/qu'/que le message ***Source_Capabilities_Extended*** soit envoyé en réponse au message ***Get_Source_Cap_Extended***. Le message ***Source_Capabilities_Extended*** permet à une source ou à une alimentation double fonction d'informer le destinataire de ses capacités en tant que source.

Le message ***Source_Capabilities_Extended*** ***Devoir/Doit/Doivent*** renvoyer un bloc de données étendues de capacités de source (SCEDB) de 24 octets, dont le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-32 et au Tableau 6-49.

Figure 6-32 Message Source_Capabilities_Extended



Anglais	Français
Extended Header	En-tête étendu
Data Size	Taille de données
24-byte Data Block	Bloc de données de 24 octets

Tableau 6-49 Bloc de données étendues de capacités de source (SCEDB)

Décalage	Champ	Description
0	VID	ID de fournisseur (affecté par l'USB-IF)
2	PID	ID de produit (affecté par le fabricant).
4	XID	Valeur affectée au produit, fournie par l'USB-IF
8	Version FW	Numéro de version de micrologiciel
9	Version HW	Numéro de version matérielle
10	Régulation de tension	<table><tr><th>Bit</th><th>Description</th></tr><tr><td>1...0</td><td>00b: Palier de charge de 150 mA/µs (par défaut) 01b: Palier de charge de 500 mA/µs 11b...10b: ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisé</td></tr><tr><td>2</td><td>0b: 25 %de l'IoC (par défaut) 1b: 90 % de l'IoC</td></tr><tr><td>3...7</td><td>***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0.</td></tr></table>
11	Temps de maintien	La sortie garde des limites régulées pendant ce nombre de millisecondes après le retrait du courant alternatif en entrée. 0x00 = fonction non prise en charge Note: ***Il convient d'/de/qu'/que*** une valeur de 3 ms soit utilisée

<table>
<tr><th>Décalage</th><th>Champ</th><th>Description</th></tr>
<tr><td>12</td><td>Conformité</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>Compatible LPS lorsque défini</td></tr>
<tr><td>1</td><td>Compatible PS1 lorsque défini</td></tr>
<tr><td>2</td><td>Compatible PS2 lorsque défini</td></tr>
<tr><td>3…7</td><td>Réservé(s/ée/ées) et Devoir/Doit/Doivent être défini sur 0.</td></tr>
</table>
</td></tr>
<tr><td>13</td><td>Courant de contact</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>EPS à courant de contact faible lorsque défini</td></tr>
<tr><td>1</td><td>Broche à la masse prise en charge lorsque défini</td></tr>
<tr><td>2</td><td>Broche à la masse prévue pour mise à la terre de protection lorsque défini</td></tr>
<tr><td>3…7</td><td>Réservé(s/ée/ées) et Devoir/Doit/Doivent être défini sur 0.</td></tr>
</table>
</td></tr>
<tr><td>14</td><td>Courant de crête 1</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0…4</td><td>Pourcentage de surcharge, par incréments de 10 %
Les valeurs supérieures à 25 (11001b) sont écrêtées à 250 %.</td></tr>
<tr><td>5…10</td><td>Période de surcharge, par paliers de 20 ms</td></tr>
<tr><td>11.14</td><td>Cycle de service, par incréments de 5 %</td></tr>
<tr><td>15</td><td>Ecart de statisme V_{BUS}</td></tr>
</table>
</td></tr>
<tr><td>16</td><td>Courant de crête 2</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0…4</td><td>Pourcentage de surcharge, par incréments de 10 %
Les valeurs supérieures à 25 (11001b) sont écrêtées à 250 %.</td></tr>
<tr><td>5…10</td><td>Période de surcharge, par paliers de 20 ms</td></tr>
<tr><td>11.14</td><td>Cycle de service, par incréments de 5 %</td></tr>
<tr><td>15</td><td>Ecart de statisme V_{BUS}</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Décalage</th><th>Champ</th><th>Description</th></tr>
<tr><td>18</td><td>Courant de crête 3</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0…4</td><td>Pourcentage de surcharge, par incréments de 10 %
Les valeurs supérieures à 25 (11001b) sont écrêtées à 250 %.</td></tr>
<tr><td>5…10</td><td>Période de surcharge, par paliers de 20 ms</td></tr>
<tr><td>11.14</td><td>Cycle de service, par incréments de 5 %</td></tr>
<tr><td>15</td><td>Ecart de statisme V_{BUS}</td></tr>
</table>
</td></tr>
<tr><td>20</td><td>Température de contact</td><td>La température se conforme à:
0 = [IEC 60950-1] (défaut)
1 = [IEC 62368-1] TS1
2 = [IEC 62368-1] TS2
Note: Toutes les autres valeurs sont Réservé(s/ée/ées)</td></tr>
<tr><td>21</td><td>Entrées source</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>0b: Pas d'alimentation externe
1b: Alimentation externe présente</td></tr>
<tr><td>1</td><td>Si le bit 0 est défini:

0b: L'alimentation externe est contrainte
1b: L'alimentation externe est non contrainte

Si le bit 0 n'est pas défini: Réservé(s/ée/ées) et Devoir/Doit/Doivent être défini sur 0</td></tr>
<tr><td>2</td><td>0b: Pas de batterie interne
1b: Batterie interne présente</td></tr>
<tr><td>3…7</td><td>Réservé(s/ée/ées) et Devoir/Doit/Doivent être défini sur 0.</td></tr>
</table>
</td></tr>
<tr><td>22</td><td>Nombre de batteries/d'emplacements de batteries</td><td>Quartet supérieur = Nombre d'emplacements de batteries remplaçables à chaud (0...4)
Quartet inférieur = Nombre de batteries fixes (0...4)</td></tr>
<tr><td>23</td><td>PDP assignée de la source</td><td>0…6: PDP assignée de la source
7: Réservé(s/ée/ées) et Devoir/Doit/Doivent être défini sur 0.</td></tr>
</table>

6.5.1.1 Champ Vendor ID (VID)

Le champ VID ***Devoir/Doit/Doivent*** contenir la valeur de l'ID de fournisseur (VID) de 16 bits affecté par l'USB-IF au fournisseur de la source. Si le fournisseur ne dispose pas de VID, le champ Vendor ID ***Devoir/Doit/Doivent*** être défini sur 0. Les dispositifs dotés d'une interface de données USB ***Devoir/Doit/Doivent*** renseigner un VID identique à l'idVendor du descripteur Standard Device (voir *[USB 2.0]* et *[USB 3.2]*).

6.5.1.2 Champ Product ID (PID)

Le champ PID ***Devoir/Doit/Doivent*** contenir la valeur de l'ID de produit de 16 bits affecté par le fournisseur de la source. Les dispositifs dotés d'une interface de données USB ***Devoir/Doit/Doivent*** renseigner un PID identique à l'idProduct du descripteur Standard Device (voir *[USB 2.0]* et *[USB 3.2]*).

6.5.1.3 Champ XID

Le champ XID ***Devoir/Doit/Doivent*** contenir le XID de 32 bits fourni par l'USB-IF au fournisseur qui, à son tour, l'affecte à un produit. Si le fournisseur ne dispose pas de XID, il ***Devoir/Doit/Doivent*** renvoyer zéro dans ce champ (voir ***[USB 2.0]*** et ***[USB 3.2]***).

6.5.1.4 Champ Firmware Version

Le champ Firmware version ***Devoir/Doit/Doivent*** contenir un numéro de version de micrologiciel de 8 bits affecté au dispositif par le fournisseur.

6.5.1.5 Champ Hardware Version

Le champ Hardware Version ***Devoir/Doit/Doivent*** contenir un numéro de version matérielle de 8 bits affecté au dispositif par le fournisseur.

6.5.1.6 Champ Voltage Regulation

Le champ Voltage Regulation contient des bits concernant la magnitude et la vitesse de balayage des paliers de charge.

Voir 7.1.12.1 pour plus d'informations.

6.5.1.6.1 Vitesse de balayage des paliers de charge

La source ***Devoir/Doit/Doivent*** renseigner ses capacités de réponse aux paliers de charge dans les bits 0...1 du champ de bits Voltage Regulation.

6.5.1.6.2 Magnitude des paliers de charge

La source ***Devoir/Doit/Doivent*** renseigner sa magnitude de paliers de charge en pourcentage de l'IoC dans le bit 2 du champ Voltage Regulation.

6.5.1.7 Champ Holdup Time

Le champ Holdup Time ***Devoir/Doit/Doivent*** contenir le temps de maintien de la source (voir 7.1.12.2).

6.5.1.8 Champ Compliance

Le champ Compliance ***Devoir/Doit/Doivent*** contenir les normes auxquelles se conforme la source (voir 7.1.12.3).

6.5.1.9 Champ Touch Current

Le champ Touch Current indique si la source atteint certains niveaux de courant de fuite et si elle dispose d'une broche à la masse.

Une source ***Devoir/Doit/Doivent*** définir le bit Touch Current (bit 0) lorsque son courant de fuite est inférieur à une valeur efficace de 65 µA, pour capacité maximale de la source inférieure ou égale à 30 W, ou lorsque son courant de fuite est inférieur à une valeur efficace de 100 µA, pour une capacité d'alimentation comprise entre 30 W et 100 W. Le courant de fuite total combiné ***Devoir/Doit/Doivent*** être mesuré conformément à l'***[IEC 60950-1]***, dans le cadre d'un essai à une valeur efficace de 250 V en courant alternatif à 50 Hz.

Une source avec une broche à la masse ***Devoir/Doit/Doivent*** définir le bit Ground pin (bit 1).

Une source dont la broche à la masse est prévue pour une connexion à une mise à la terre de protection ***Devoir/Doit/Doivent*** définir les bits 1 et 2.

6.5.1.10 Champ Peak Current

Le champ Peak Current ***Devoir/Doit/Doivent*** contenir la combinaison des courants de crête que la source prend en charge (voir 7.1.12.4).

Le courant de crête permet à la source d'indiquer sa capacité à fournir un courant dépassant la quantité négociée pour de courtes périodes. Le descripteur Peak Current définit jusqu'à trois combinaisons de pourcentage de surcharge, de durée et de cycle de service que la source prend en charge, définis par PeakCurrent1, PeakCurrent2 et PeakCurrent3. Une source ***Pouvoir/Peut/Peuvent*** n'offrir aucune capacité de courant de crête. Une source ***Devoir/Doit/Doivent*** renseigner les champs de bits Peak Current qu'elle n'utilise pas avec des zéro.

Les champs de bits de Peak Current1, Peak Current2, et Peak Current3 contiennent les sous-champs suivants:

- **Percentage Overload** ***Devoir/Doit/Doivent*** être le courant de crête maximal, indiqué par incréments de 10 %, exprimé en pourcentage du courant de fonctionnement négocié (IoC) offert par la source. Les valeurs supérieures à 25 (11001b) sont écrêtées à 250 %.
- **Overload Period** ***Devoir/Doit/Doivent*** être la fenêtre minimale de temps moyen de mise en place, par incréments de 20 ms, où une valeur de 20 ms est recommandée.
- **Duty Cycle** ***Devoir/Doit/Doivent*** être le pourcentage maximal de période de surcharge, indiqué par incréments de 5 %. ***Il convient d'/de/qu'/que*** les valeurs soient respectivement de 5 %, 10 % et 50 % pour PeakCurrent1, PeakCurrent2 et PeakCurrent3.
- **V_{BUS} Droop** ***Devoir/Doit/Doivent*** être défini sur 1 pour indiquer un écart de statisme supplémentaire de 5 % sur V_{BUS}, lorsque les conditions de surcharge se manifestent, comme défini par ***vSrcPeak***. Toutefois, à titre de recommandation, ***Il convient d'/de/qu'/que***, lorsque les conditions de surcharge se manifestent, que la source fournisse V_{BUS} à hauteur de ***vSrcNew*** et définisse ce bit sur zéro.

6.5.1.11 Champ Touch Temp

Le champ Touch Temp ***Devoir/Doit/Doivent*** indiquer la norme IEC utilisée pour déterminer la température de surface de l'enceinte de la source. Les limites de sécurité pour la température de contact de la source sont définies dans les normes applicables en matière de sécurité des produits (par exemple, l'***[IEC 60950-1]*** ou l'***[IEC 62368-1]***). La source ***Pouvoir/Peut/Peuvent*** indiquer lorsque ses performances relatives à la température de contact se conforment aux limites TS1 et TS2 décrites dans l'***[IEC 62368-1]***.

6.5.1.12 Champ Source Inputs

Le champ Source Inputs ***Devoir/Doit/Doivent*** identifier les entrées possibles qui alimentent la source. Noter que certaines sources ne sont alimentées que par une batterie (une automobile, par exemple) et non par le secteur, ce qui est pourtant plus commun.

- Lorsque le bit 0 est défini, la source peut être alimentée par une alimentation externe.
- Lorsque les bits 0 et 1 sont définis, la source peut être alimentée par une alimentation externe admise comme "infinie", c'est-à-dire qui ne s'épuise pas avec le temps.
- Lorsque le bit 2 est défini, la source peut être alimentée par une batterie interne.

Le bit 2 ***Pouvoir/Peut/Peuvent*** être défini indépendamment des bits 0 et 1.

6.5.1.13 Champ Number of Batteries/Battery Slots

Le champ Number of Batteries/Battery Slots ***Devoir/Doit/Doivent*** indiquer le nombre de batteries fixes et d'emplacements de batteries remplaçables à chaud que la source prend en charge. Ce champ ***Devoir/Doit/Doivent*** indiquer indépendamment le nombre d'emplacements de batteries et le nombre de batteries fixes.

Une source ne ***Devoir/Doit/Doivent*** pas disposer de plus de 4 batteries fixes ni de plus de 4 emplacements de batteries.

Les batteries fixes ***Devoir/Doit/Doivent*** être numérotées consécutivement de 0 à 3. Le numéro assigné à une batterie fixe donnée ***Ne doit/doivent pas*** varier lorsqu'elle est branchée ou débranchée.

Les emplacements de batteries ***Devoir/Doit/Doivent*** être numérotés consécutivement de 4 à 7. Le numéro assigné à un emplacement de batterie donné ***Ne doit/doivent pas*** varier lorsque cette dernière est branchée ou débranchée.

6.5.1.14 Champ Source PDP Rating

Le champ Source PDP Rating ***Devoir/Doit/Doivent*** indiquer la partie entière de la PDP assignée de la source, conformément au Tableau 10-2.

Le champ Source PDP Rating qui est déclaré ***Devoir/Doit/Doivent*** être invariant et ***Devoir/Doit/Doivent*** respecter les exigences ***[USB Type-C 2.0]*** relatives aux chargeurs simple port ou multiports à capacité assurée, ou aux chargeurs multiports à capacité partagée.

6.5.2 Message Status

Le message ***Statut*** ***Devoir/Doit/Doivent*** être envoyé en réponse au message ***Get_Status***. Le contenu du message ***Statut*** dépend de la cible du message ***Get_Status***. Lorsqu'il est envoyé à ***SOP***, le message Status renvoie le statut du port partenaire du port. Lorsqu'il est envoyé à ***SOP'*** ou ***SOP''***, le message ***Statut*** renvoie le statut de l'une des fiches câbles du câble actif.

6.5.2.1 Message de statut SOP

Un message ***Statut***, envoyé en réponse à un message ***Get_Status*** à ***SOP***, permet à un port d'informer son port partenaire du statut actuel de la source ou du destinataire. En général, un message ***Get_Status*** est envoyé par le port après réception d'un message ***Alert***. Certains événements signalés sont essentiels, comme OCP, OVP et OTP, tandis que d'autres sont informatifs, par exemple lorsque la batterie passe du statut "en charge" au statut "inactive".

Le message ***Statut*** renvoie un bloc de données de statut (SDB) de 6 octets, dont le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-33 et au Tableau 6-50.

Figure 6-33 Message Status

Anglais	Français
Extended Header	En-tête étendu
Data Size	Taille de données
5-byte block	Bloc de 5 octets

Tableau 6-50 Bloc de données de statut SOP (SDB)

Décalage (octet)	Champ	Description
0	Température interne	Température interne de la source ou du destinataire en degrés Celsius. 0 = fonction non prise en charge 1 = température inférieure à 2 °C. 2-255 = température en °C.

<table>
<tr><th>Décalage (octet)</th><th>Champ</th><th>Description</th></tr>
<tr><td>1</td><td>Entrée actuelle</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>Réservé(s/ée/ées) et Devoir/Doit/Doivent être défini sur 0.</td></tr>
<tr><td>1</td><td>Alimentation externe, lorsque défini</td></tr>
<tr><td>2</td><td>Alimentation externe CA/CC (Valide(s) lorsque le bit 1 est défini)
0: CC
1: CA
Réservé(s/ée/ées) lorsque le bit 1 vaut 0</td></tr>
<tr><td>3</td><td>Lorsqu'il est défini, alimentation interne par batterie</td></tr>
<tr><td>4</td><td>Lorsqu'il est défini, alimentation interne qui n'est pas une batterie</td></tr>
<tr><td>5…7</td><td>Réservé(s/ée/ées) et Devoir/Doit/Doivent être défini sur 0.</td></tr>
</table>
</td></tr>
<tr><td>2</td><td>Entrée de batterie actuelle</td><td>Lorsque le bit 3 du champ Present Input est défini, il Devoir/Doit/Doivent contenir le bit correspondant aux batteries qui fournissent l'alimentation:

Quartet supérieur = Batterie remplaçable à chaud (b7…4)
Quartet inférieur = Batterie fixe (b3…0)

Lorsque le bit 3 du champ Present Input n'est pas défini, ce champ est Réservé(s/ée/ées) et Devoir/Doit/Doivent être défini sur 0.</td></tr>
<tr><td>3</td><td>Fanions d'événements</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>Réservé(s/ée/ées) et Devoir/Doit/Doivent être défini sur 0.</td></tr>
<tr><td>1</td><td>Evénement OCP lorsque défini</td></tr>
<tr><td>2</td><td>Evénement OTP lorsque défini</td></tr>
<tr><td>3</td><td>Evénement OVP lorsque défini</td></tr>
<tr><td>4</td><td>Mode CF lorsque défini, mode CV sinon</td></tr>
<tr><td>5…7</td><td>Réservé(s/ée/ées) et Devoir/Doit/Doivent être défini sur 0.</td></tr>
</table>
</td></tr>
<tr><td>4</td><td>Statut de température</td><td>
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>Réservé(s/ée/ées) et Devoir/Doit/Doivent être défini sur 0.</td></tr>
<tr><td>1…2</td><td>00 – Non pris en charge
01 – Normal
10 – Avertissement
11 – Surchauffe</td></tr>
<tr><td>3…7</td><td>Réservé(s/ée/ées) et Devoir/Doit/Doivent être défini sur 0.</td></tr>
</table>
</td></tr>
</table>

Décalage (octet)	Champ	Description
5	Statut d'alimentation	<table><tr><th>Bit</th><th>Description</th></tr><tr><td>0</td><td>***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0.</td></tr><tr><td>1</td><td>Puissance de la source limitée en raison du courant pris en charge par le câble</td></tr><tr><td>2</td><td>Puissance de la source limitée en raison d'une puissance disponible insuffisante lorsqu'elle alimente d'autres ports</td></tr><tr><td>3</td><td>Puissance de la source limitée en raison d'une alimentation externe insuffisante</td></tr><tr><td>4</td><td>Puissance de la source limitée en raison de la présence de fanions d'événements (les fanions d'événements doivent également être définis)</td></tr><tr><td>5</td><td>Puissance de la source limitée en raison de la température</td></tr><tr><td>6...7</td><td>***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0.</td></tr></table>

6.5.2.1.1 Champ Internal Temp

Le champ Internal Temp indique la température instantanée d'une portion de la source ou du destinataire.

6.5.2.1.2 Champ Present Input

Le champ Present Input indique quelle alimentation est actuellement exploitée par la source ou le destinataire.

Les bits suivants sont définis:

- le bit 1 indique la présence d'une source externe;
- le bit 2 indique si la source non contrainte externe est en courant alternatif ou en courant continu;
- le bit 3 indique que l'alimentation provient d'une batterie;
- le bit 4 indique la présence d'une source interne alternative qui n'est pas une batterie.

6.5.2.1.3 Champ Present Battery Input

Le champ Present Battery Input indique quelles batteries alimentent actuellement la source ou le destinataire. Ce champ n'est ***Valide(s)*** que lorsque le champ Present Input indique que l'alimentation est interne et provient d'une batterie.

Le quartet supérieur du champ indique quelles batteries remplaçables à chaud fournissent l'alimentation, le bit 4 du quartet supérieur correspondant à la batterie 4 et le bit 7 à la batterie 7 (voir 6.5.3 et 6.5.4).

Le quartet inférieur du champ indique quelles batteries fixes fournissent l'alimentation, le bit 0 du quartet inférieur correspondant à la batterie 0 et le bit 3 à la batterie 3 (voir 6.5.3 et 6.5.4).

6.5.2.1.4 Champ Event Flags

Le champ Event Flags renvoie des fanions d'événements. Les fanions d'événements OTP, OVP et OCP ***Devoir/Doit/Doivent*** être définis dans le cas d'un événement et ne ***Devoir/Doit/Doivent*** être effacés que lorsqu'ils sont lus avec le message ***Get_Status***.

Lorsque le fanion d'événement OTP est défini, le champ Temperature Status ***Devoir/Doit/Doivent*** également être défini sur "over temperature" (surchauffe).

Le bit CF/CV mode n'est ***Valide(s)*** que lors d'un fonctionnement en tant qu'alimentation électrique programmable, et ***Devoir/Doit/Doivent*** être ***Ignoré(s/ée/ées)*** dans le cas contraire. Lorsque la source fonctionne en tant qu'alimentation électrique programmable, le bit CF/CV mode ***Devoir/Doit/Doivent***

être défini lors d'un fonctionnement en mode Current Limit (mode CL) et ***Devoir/Doit/Doivent*** être effacé lors d'un fonctionnement en mode de tension constante (mode CV).

6.5.2.1.5 Champ Temperature Status

Le champ Temperature Status renvoie le statut actuel de température de l'appareil, c'est-à-dire "normal", "warning" (avertissement) ou "over temperature" (surchauffe). Lorsque le champ Temperature Status est défini sur "over temperature", le fanion d'événement OTP ***Devoir/Doit/Doivent*** également être défini.

6.5.2.1.6 Champ Power Status

Le champ Power Status indique le statut actuel d'une source. Un retour à une valeur différente de zéro du champ indique que la source d'alimentation annoncée est réduite soit: parce que le câble ne prend pas en charge la totalité du courant de la source, parce que la source alimente d'autres ports et ne peut pas fournir sa pleine puissance, parce que l'alimentation externe à la source est insuffisante pour prendre en charge la pleine puissance, ou parce qu'un événement est survenu, provoquant la réduction de la puissance annoncée par la source.

Un destinataire ***Devoir/Doit/Doivent*** définir ce champ sur zéro.

6.5.2.1 Message Status SOP'/SOP''

Un message ***Statut***, envoyé en réponse à un message ***Get_Status*** à ***SOP'*** ou à ***SOP''***, permet à une source ou à un destinataire d'obtenir le statut actuel des fiches de câbles du câble actif. En règle générale, un message ***Get_Status*** est utilisé par l'hôte USB et/ou le dispositif USB pour contrôler la température de la ou des fiches de câbles du câble actif. Le message ***Statut*** renvoie un bloc de données de statut (SDB) de 2 octets, dont le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-34 et au Tableau 6-51.

Figure 6-34 Message de statut SOP'/SOP''

Anglais	Français
Extended Header	En-tête étendu
Data Size	Taille de données
2-byte block	Bloc de 2 octets

Tableau 6-51 Bloc de données de statut SOP'/SOP'' (SDB)

Décalage (octet)	Champ	Valeur	Description
0	Température interne	Unsigned Int	Température interne de la fiche du câble actif en °C. 0 = fonction non prise en charge 1 = température inférieure à 2 °C. 2…255 = température en °C.
1	Fanions	Champ de bits	**Bit** / **Description** 0 / Thermal Shutdown 1…7 / ***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0.

6.5.2.1.1 Champ Internal Temp

Le champ Internal Temp indique la température instantanée de la fiche en °C. La température interne ***Devoir/Doit/Doivent*** être monotone. Le câble actif ***Devoir/Doit/Doivent*** indiquer sa température interne à intervalles ***TACTempUpdate***.

6.5.2.1.2 Champ Thermal Shutdown

Le fanion Thermal Shutdown ***Devoir/Doit/Doivent*** également être défini lorsque la température interne de la fiche dépasse la température interne maximale indiquée dans le VDO Active Cable. Lorsque ce bit a été défini, il ***Devoir/Doit/Doivent*** rester défini et la fiche ***Devoir/Doit/Doivent*** rester en arrêt thermique jusqu'à une réinitialisation matérielle ou jusqu'à ce que l'alimentation du câble actif soit coupée. Le fanion Thermal Shutdown ***Ne doit/doivent pas*** être effacé par une réinitialisation de câble.

6.5.3 Message Get_Battery_Cap

Le message ***Get_Battery_Cap*** (Obtenir les capacités de la batterie) permet de demander les capacités d'une batterie présente dans son port partenaire. Le port ***Devoir/Doit/Doivent*** répondre en renvoyant un message ***Battery_Capabilities*** (voir 6.5.5) qui contient un bloc de données de capacités de batterie (BCDB) relatif à la batterie concernée.

Le message ***Get_Battery_Cap*** contient un bloc de données d'obtention de capacité de batterie (GBCDB) de 1 octet, dont le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-35 et au Tableau 6-52. Ce bloc définit la batterie pour laquelle la demande est effectuée.

Le champ ***Data Size*** du message ***Get_Battery_Cap*** ***Devoir/Doit/Doivent*** être défini sur 1.

Figure 6-35 Message Get_Battery_Cap

Anglais	Français
Extended Header	En-tête étendu
Data Size	Taille de données

Tableau 6-52 Bloc de données d'obtention de capacités de batterie (GBCDB)

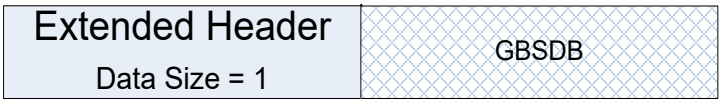
Décalage	Champ	Description
0	***Battery Cap Ref***	Numéro de la batterie indexée à partir de zéro: • les valeurs 0…3 représentent les batteries fixes; • les valeurs 4…7 représentent les batteries remplaçables à chaud; • les valeurs 8…255 sont ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisées.

6.5.4 Message Get_Battery_Status

Le message ***Get_Battery_Status*** permet de demander le statut d'une batterie présente dans son port partenaire. Le port ***Devoir/Doit/Doivent*** répondre en renvoyant un message ***Battery_Status*** (voir 6.4.5) contenant un objet de données de statut de batterie (BSDO) pour la batterie concernée.

Le message ***Get_Battery_Status*** contient un bloc de données d'obtention de statut de batterie (GBSDB) de 1 octet, dont le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-36 et au Tableau 6-53. Ce bloc contient les détails relatifs à la batterie demandée. Le champ ***Data Size*** du message ***Get_Battery_Status*** ***Devoir/Doit/Doivent*** être défini sur 1.

Figure 6-36 Message Get_Battery_Status



Anglais	Français
Extended Header	En-tête étendu
Data Size	Taille de données

Tableau 6-53 Bloc de données d'obtention de statut de batterie (GBSDB)

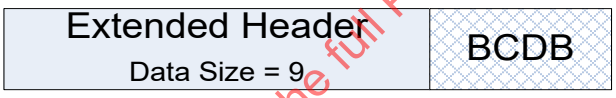
Décalage	Champ	Description
0	*Battery Status Ref*	Numéro de la batterie indexée à partir de zéro: • les valeurs 0…3 représentent les batteries fixes; • les valeurs 4…7 représentent les batteries remplaçables à chaud; • les valeurs 8…255 sont ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisées.

6.5.5 Message Battery_Capabilities

Le message ***Battery_Capabilities*** est envoyé en réponse au message ***Get_Battery_Cap***. Le message ***Battery_Capabilities*** contient un bloc de données de capacité de batterie (BCDB) pour l'une des batteries qu'il prend en charge, indiquée par le champ Battery du message ***Source_Capabilities_Extended***. Le BCDB renvoyé ***Devoir/Doit/Doivent*** correspondre à la batterie demandée dans le champ ***Battery Cap Ref*** contenu dans le message ***Get_Battery_Cap***.

Le message ***Battery_Capabilities*** renvoie un BCDB de 9 octets, dont le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-37 et au Tableau 6-50.

Figure 6-37 Message Battery_Capabilities



Anglais	Français
Extended Header	En-tête étendu
Data Size	Taille de données

Tableau 6-54 Bloc de données de capacité de batterie (BCDB)

Décalage (octet)	Champ	Description
0	VID	ID de fournisseur (affecté par l'USB-IF)
2	PID	ID de produit (affecté par le fabricant)
4	Battery Design Capacity	Capacité théorique de la batterie en 0,1 Wh Note: 0x0000 = batterie absente 0xFFFF = capacité théorique inconnue
6	Battery Last Full Charge Capacity	Dernière capacité à pleine charge de la batterie en 0,1 Wh Note: 0x0000 = batterie absente 0xFFFF = dernière capacité à pleine charge inconnue
8	Battery Type	**Bit** / **Description** 0 / Référence de batterie non valide 1-7 / ***Réservé(s/ée/ées)***

6.5.5.1 Champ Battery Design Capacity

Le champ Battery Design Capacity ***Devoir/Doit/Doivent*** renvoyer la capacité théorique de la batterie en dixièmes de Wh. Si la batterie est remplaçable à chaud et n'est pas présente, le champ Battery Design Capacity ***Devoir/Doit/Doivent*** être défini sur 0. Si la batterie est incapable de renseigner sa capacité théorique, ce champ ***Devoir/Doit/Doivent*** renvoyer 0xFFFF.

6.5.5.2 Champ Battery Last Full Charge Capacity

Le champ Battery Last Full Charge Capacity ***Devoir/Doit/Doivent*** renvoyer la dernière capacité à pleine charge de la batterie en dixièmes de Wh. Si la batterie est remplaçable à chaud et n'est pas présente, le champ Battery Last Full Charge Capacity ***Devoir/Doit/Doivent*** être défini sur 0. Si la batterie est incapable de renseigner sa capacité théorique, le champ Battery Last Full Charge Capacity ***Devoir/Doit/Doivent*** être défini sur 0xFFFF.

6.5.5.3 Champ Battery Type

Le champ Battery Type est utilisé pour renseigner des informations supplémentaires sur les capacités de la batterie.

6.5.5.3.1 Référence de batterie non valide

Le bit Invalid Battery Reference ***Devoir/Doit/Doivent*** être défini lorsque le message ***Get_Battery_Cap*** contient une référence à une batterie qui n'existe pas.

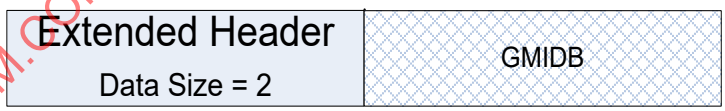
6.5.6 Message Get_Manufacturer_Info

Le message ***Get_Manufacturer_Info*** est envoyé par un port pour demander des informations spécifiques au fabricant vis-à-vis de son port partenaire, d'une fiche de câble ou d'une batterie derrière un port. Le port ***Devoir/Doit/Doivent*** répondre en renvoyant un message ***Manufacturer_Info*** (6.5.7) qui contient un bloc de données d'informations de fabricant (MIDB). La prise en charge de cette fonctionnalité par la fiche de câble est ***Normatif(s/ve/ves) Facultatif(s/ve/ves)***,.

Le message ***Get_Manufacturer_Info*** contient un bloc de données d'obtention d'informations de fabricant (GMIDB) de 2 octets. Ce bloc définit si les informations du fabricant demandées concernent le dispositif ou la batterie, et quelle est la batterie concernée par la demande.

Le message ***Get_Manufacturer_Info*** renvoie un GMIDB dont le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-36 et au Tableau 6-55.

Figure 6-38 Message Get_Manufacturer_Info



Anglais	Français
Extended Header	En-tête étendu
Data Size	Taille de données

Tableau 6-55 Bloc de données d'obtention d'informations de fabricant (GMIDB)

Décalage	Champ	Description
0	***Manufacturer Info Target***	0: Port/fiche de câble 1: Batterie 255…2: ***Réservé(s/ée/ées)***, ***Ne doit/doivent pas*** être utilisé.

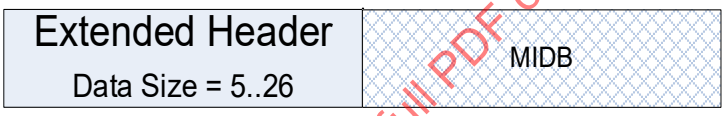
Décalage	Champ	Description
1	***Manufacturer Info Ref***	Si la valeur du sous-champ ***Manufacturer Info Target*** est "Battery (01b)", le champ ***Manufacturer Info Ref*** ***Devoir/Doit/Doivent*** contenir le numéro de référence de la batterie, qui est le numéro de la batterie indexé à partir de 0: • les valeurs 0…3 représentent les batteries fixes; • les valeurs 4…7 représentent les batteries remplaçables à chaud. Sinon, ce champ est ***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0.

6.5.7 Message Manufacturer_Info

Le message ***Manufacturer_Info*** ***Devoir/Doit/Doivent*** être envoyé en réponse au message ***Get_Manufacturer_Info***. Le message ***Manufacturer_Info*** contient le VID USB et le PID de fournisseur pour identifier la matrice d'octets correspondant au dispositif, à la batterie et au dispositif ou au fabricant de la batterie dans un bloc de données de longueur variable, jusqu'à ***MaxExtendedMsgLegacyLen***.

Le message ***Manufacturer_Info*** renvoie un bloc de données d'informations de fabricant (MIDB) dont le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-37 et au Tableau 6-50.

Figure 6-39 Message Manufacturer_Info



Anglais	Français
Extended Header	En-tête étendu
Data Size	Taille de données

Tableau 6-56 Bloc de données d'informations de fabricant (MIDB)

Décalage	Champ	Description
0	VID	ID de fournisseur (affecté par l'USB-IF)
2	PID	ID de produit (affecté par le fabricant)
4	Manufacturer String	Chaîne de 0…21 caractères terminée par une valeur nulle, définie par le fournisseur Si le champ Manufacturer Info Target ou le champ Manufacturer Info Ref du message ***Get_Manufacturer_Info*** n'est pas reconnu, le champ doit renvoyer une chaîne de texte ASCII terminée par une valeur nulle "Not Supported".

6.5.7.1 Vendor ID (VID)

Le champ VID ***Devoir/Doit/Doivent*** contenir la chaîne du fabricant du dispositif ou de la batterie, définie par le fournisseur.

Si le champ ***Manufacturer Info Target*** du message ***Get_Manufacturer_Info*** est ***Invalide(s)***, ce champ VID ***Devoir/Doit/Doivent*** être 0xFFFF et ***Il convient d'/de/qu'/que*** le champ PID associé soit défini sur 0x0000. Si le champ ***Manufacturer Info Target*** du message ***Get_Manufacturer_Info*** a la valeur "Battery (01b)" et que le champ ***Manufacturer Info Ref*** est ***Invalide(s)***, ce champ VID ***Devoir/Doit/Doivent*** être 0xFFFF et ***Il convient d'/de/qu'/que*** le champ PID associé soit défini sur 0x0000.

6.5.7.2 Product ID (PID)

Le champ PID ***Devoir/Doit/Doivent*** contenir l'identifiant de produit de 16 bits du dispositif ou de la batterie, désigné par le fournisseur.

Si le champ ***Manufacturer Info Target*** du message ***Get_Manufacturer_Info*** est ***Invalide(s)***, ***Il convient d'/de/qu'/que*** ce champ PID soit défini sur 0x0000. Si le champ ***Manufacturer Info Target*** du message ***Get_Manufacturer_Info*** a la valeur "Battery (01b)" et que le champ ***Manufacturer Info Ref*** est ***Invalide(s)***, ***Il convient d'/de/qu'/que*** ce champ PID soit défini sur 0x0000.

Lors de la réception d'un message ***Manufacturer_Info***, où le VID est défini sur 0xFFFF, le champ PID ***Devoir/Doit/Doivent*** être ***Ignoré(s/ée/ées)***.

6.5.7.3 Manufacturer String

Ce champ ***Devoir/Doit/Doivent*** contenir la chaîne du fabricant du dispositif ou de la batterie, définie par le fournisseur.

Si le champ ***Manufacturer Info Target*** ou le champ ***Manufacturer Info Ref*** du message ***Get_Manufacturer_Info*** n'est pas reconnu, le champ ***Devoir/Doit/Doivent*** renvoyer zéro octet.

6.5.8 Messages de sécurité

Le processus d'authentification entre des ports partenaires ou entre un port et une fiche de câble est entièrement décrit dans ***[USBTypeCAuthentication 1.0]***. Cette spécification décrit deux messages étendus utilisés par le processus d'authentification lorsqu'il est appliqué à l'alimentation USB.

Dans le processus d'authentification décrit dans ***[USBTypeCAuthentication 1.0]***, trois échanges de base servent à:

- obtenir les certificats du port ou de la fiche de câble;
- obtenir le résumé du port ou de la fiche de câble;
- demander l'accès au port partenaire ou à la fiche de câble.

Les certificats sont utilisés pour transmettre des informations et attestés par un signataire, qui atteste de l'authenticité du port partenaire ou de la fiche de câble. Les certificats du port ou de la fiche de câble sont nécessaires lorsqu'un port rencontre un port partenaire ou une fiche de câble auquel/à laquelle il n'a pas été branché auparavant. Pour réduire le plus possible les calculs après le premier branchement, un port peut également utiliser un résumé composé de hachages de certificats plutôt que les certificats eux-mêmes. Lorsque le port a obtenu les certificats et qu'il a calculé les hachages, il mémorise les hachages et utilise le résumé pour les échanges ultérieurs. Lorsque le port a obtenu les certificats ou le résumé, il demande l'accès à son port partenaire ou à la fiche de câble pour détecter les attaques par relecture.

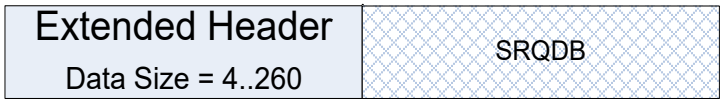
Pour plus d'informations, voir ***[USBTypeCAuthentication 1.0]***.

6.5.8.1 Security_Request

Le message ***Security_Request*** est utilisé par un port pour transmettre une structure de données de sécurité à son port partenaire ou à une fiche de câble.

Le message ***Security_Request*** contient un bloc de données de demande de sécurité (SRQDB) dont le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-40. Le contenu du SRQDB et son utilisation sont définis dans ***[USBTypeCAuthentication 1.0]***.

Figure 6-40 Message Security_Request



Anglais	Français
Extended Header	En-tête étendu

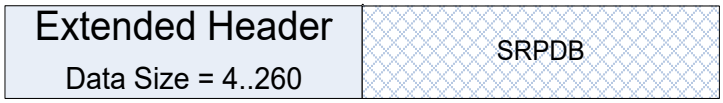
Data Size	Taille de données

6.5.8.2 Security_Response

Le message ***Security_Response*** est utilisé par un port ou par une fiche de câble pour transmettre une structure de données de sécurité au port qui a envoyé le message ***Security_Request***.

Le message ***Security_Response*** contient un bloc de données de réponse de sécurité (SRPDB) dont le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-41. Le contenu du SRPDB et son utilisation sont définis dans ***[USBTypeCAuthentication 1.0]***.

Figure 6-41 Message Security_Response



Anglais	Français
Extended Header	En-tête étendu
Data Size	Taille de données

6.5.9 Messages de mise à jour du micrologiciel

Le processus de mise à jour du micrologiciel entre des ports partenaires ou entre un port et une fiche de câble est entièrement décrit dans ***[USBPDFirmwareUpdate 1.0]***. La présente spécification décrit deux messages étendus utilisés par le processus de mise à jour du micrologiciel lorsqu'il est appliqué à l'alimentation USB.

6.5.9.1 Firmware_Update_Request

Le message ***Firmware_Update_Request*** est utilisé par un port pour transmettre une structure de données de mise à jour du micrologiciel à son port partenaire ou à une fiche de câble.

Le message ***Firmware_Update_Request*** contient un bloc de données de demande de mise à jour du micrologiciel (FRQDB) dont le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-42. Le contenu du FRQDB et son utilisation sont définis dans ***[USBPDFirmwareUpdate 1.0]***.

Figure 6-42 Message Firmware_Update_Request

Anglais	Français
Extended Header	En-tête étendu
Data Size	Taille de données

6.5.9.2 Firmware_Update_Response

Le message ***Firmware_Update_Response*** est utilisé par un port ou une fiche de câble pour transmettre une structure de données de mise à jour du micrologiciel au port qui a envoyé le message ***Firmware_Update_Request***.

Le message ***Firmware_Update_Response*** contient un bloc de données de réponse de mise à jour du micrologiciel (FRPDB) dont le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-43. Le contenu du FRPDB et son utilisation sont définis dans ***[USBPDFirmwareUpdate 1.0]***.

Figure 6-43 Message Firmware_Update_Response

Anglais	Français
Extended Header	En-tête étendu
Data Size	Taille de données

6.5.10 Message PPS_Status

Le message ***PPS_Status*** ***Devoir/Doit/Doivent*** être envoyé en réponse au message ***Get_PPS_Status***. Le message ***PPS_Status*** permet à un destinataire de demander à la source d'obtenir des informations supplémentaires sur son état de fonctionnement. Le message ***Get_PPS_Status*** et le message ***PPS_Status*** ***Devoir/Doit/Doivent*** être pris en charge uniquement lorsque le message ***Alert*** est également pris en charge.

Le message ***PPS_Status*** ***Devoir/Doit/Doivent*** renvoyer un bloc de données de statut PPS (PPSSDB) de 4 octets, dont le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-44 et au Tableau 6-57.

Figure 6-44 Message PPS_Status

Anglais	Français
Extended Header	En-tête étendu
Data Size	Taille de données
4-byte Data Block	Bloc de données de 4 octets

Tableau 6-57 Bloc de données de statut PPS (PPSSDB)

Décalage	Champ	Taille	Description
0	Tension de sortie	2	Tension de sortie de la source en unités de 20 mV. Lorsqu'il est défini sur 0xFFFF, la source ne prend pas ce champ en charge.
2	Courant de sortie	1	Courant de sortie de la source en unités de 50 mA. Lorsqu'il est défini sur 0xFF, la source ne prend pas ce champ en charge.
3	Fanions de temps réel	1	**Bit** / **Description** 0 – ***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0. 1…2 – PTF: 00 – Non pris en charge PTF: 01 – Normal PTF: 10 – Avertissement PTF: 11 – Surchauffe 3 – OMF défini en mode Current Limit, et effacé en mode Constant Voltage. 4…7 – ***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0.

6.5.10.1 Champ Output Voltage

Le champ Output Voltage ***Devoir/Doit/Doivent*** renvoyer la tension de sortie de la source au moment de la demande. Celle-ci est mesurée à l'embase de la source ou, si la source possède un câble captif, elle est mesurée à l'endroit où la tension est appliquée au câble.

La précision de la mesure ***Devoir/Doit/Doivent*** être de +/- 3 %, arrondie à 20 mV.

Si la source ne prend pas ce champ en charge, il ***Devoir/Doit/Doivent*** être défini sur 0xFFFF.

6.5.10.2 Champ Output Current

Le champ Output Current ***Devoir/Doit/Doivent*** renvoyer le courant de sortie de la source au moment de la demande.

La précision de la mesure ***Devoir/Doit/Doivent*** être de +/- 150 mA.

Si la source ne prend pas ce champ en charge, il ***Devoir/Doit/Doivent*** être défini sur 0xFF.

6.5.10.3 Champ Real Time Flags

Les fanions de temps réel fournissent une indication en temps réel de l'état de fonctionnement de la source.

- Le PTF (Present Temperature Flag, fanion de température actuelle) ***Devoir/Doit/Doivent*** fournir une indication en temps réel du statut de température interne de la source. Si le PTF n'est pas pris en charge, il est défini sur 0.
  - L'état normal indique que la source fonctionne dans son enveloppe thermique normale.
  - L'état d'avertissement indique que la source est en surchauffe, mais qu'elle n'est pas en danger imminent de mise à l'arrêt.
  - L'état de surchauffe indique que la source est en surchauffe, qu'elle va bientôt se mettre à l'arrêt ou qu'elle s'est déjà mise à l'arrêt, et qu'elle a envoyé un OTP dans un message ***Alert***.
- Les OMF (Operating Mode Flag, fanions de mode de fonctionnement) ***Devoir/Doit/Doivent*** fournir une indication en temps réel du mode de fonctionnement de la source. Lorsqu'ils sont définis, la

source fonctionne en mode Current Limit; lorsqu'ils sont effacés, elle fonctionne en mode Constant Voltage.

6.5.11 Message Country_Codes

Le message ***Country_Codes*** ***Devoir/Doit/Doivent*** être envoyé en réponse au message ***Get_Country_Codes***. Le message ***Country_Codes*** permet à un port de formuler une requête à son port partenaire afin d'obtenir une liste de codes pays alpha-2, définis dans l'***[ISO 3166]***, pour lesquels le port partenaire dispose d'informations spécifiques au pays.

Le message Country_Codes ***Devoir/Doit/Doivent*** contenir un bloc de données de code pays (CCDB) de 4 à 26 octets, dont le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-45 et au Tableau 6-58.

Figure 6-45 Message Country_Codes

Anglais	Français
Extended Header	En-tête étendu
Data Size	Taille de données
4-260 byte Data Block	Bloc de données de 4-260 octets

Tableau 6-58 Bloc de données de codes pays (CCDB)

Décalage	Champ	Description
0	Longueur	Nombre de codes pays dans le message
1	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
2	1er code pays	Premier caractère du code pays Alpha-2 défini par l'***[ISO 3166]***
3		Second caractère du code pays Alpha-2 défini par l'***[ISO 3166]***
4	2e code pays	Premier caractère du code pays Alpha-2 défini par l'***[ISO 3166]***
5		Second caractère du code pays Alpha-2 défini par l'***[ISO 3166]***
	…	
Longueur * 2n	ne code pays	

6.5.11.1 Champ Country Code

Le champ Country Code ***Devoir/Doit/Doivent*** contenir le code pays Alpha-2 défini par l'***[ISO 3166]***.

6.5.12 Message Country_Info

Le message ***Country_Info*** ***Devoir/Doit/Doivent*** être envoyé en réponse au message ***Get_Country_Info***. Le message ***Country_Info*** permet à un port d'obtenir davantage d'informations spécifiques au pays auprès de son port partenaire.

Le message ***Country_Info*** ***Devoir/Doit/Doivent*** contenir un bloc de données d'informations de pays (CIDB) de 4 à 26 octets dont le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-46 et au Tableau 6-59.

Figure 6-46 Message Country_Info

Anglais	Français
Extended Header	En-tête étendu
Data Size	Taille de données
4-260 byte Data Block	Bloc de données de 4-260 octets

Tableau 6-59 Bloc de données d'informations de pays (CIDB)

Décalage	Champ	Description
0	Code pays	Premier caractère du code pays Alpha-2 défini par le message ***Get_Country_Info***
1		Second caractère du code pays Alpha-2 défini par le message ***Get_Country_Info***
2…3	***Réservé(s/ée/ées)***	***Devoir/Doit/Doivent*** être défini sur 0.
4	Données spécifiques au pays	0…22 octets pour le contenu défini par les autorités du pays.

6.5.12.1 Champ Country Code

Le champ Country Code ***Devoir/Doit/Doivent*** contenir le code pays Alpha-2 reçu par le message ***Get_Country_Info*** correspondant.

6.5.12.2 Champ Country Specific Data

Le champ Country Specific Data ***Devoir/Doit/Doivent*** contenir le contenu défini et formaté de la manière déterminée par un organisme officiel du pays indiqué dans le champ Country Code.

Si le champ Country Code du message ***Get_Country_Info*** n'est pas reconnu, le champ Country Specific Data ***Devoir/Doit/Doivent*** renvoyer la chaîne de texte ASCII terminée par une valeur nulle "Unsupported Country Code".

6.5.13 Message Sink_Capabilities_Extended

Le message ***Sink_Capabilities_Extended*** ***Devoir/Doit/Doivent*** être envoyé en réponse au message ***Get_Sink_Cap_Extended***. Le message ***Sink_Capabilities_Extended*** permet à un destinataire ou à une alimentation double fonction d'informer la source de ses capacités en tant que destinataire.

Le message ***Sink_Capabilities_Extended*** ***Devoir/Doit/Doivent*** renvoyer un bloc de données étendues de capacités de destinataire (SCEDB) de 21 octets, dont le format ***Devoir/Doit/Doivent*** être conforme à la Figure 6-46 et au Tableau 6-60.

Figure 6-47 Message Sink_Capabilities_Extended

Anglais	Français
Extended Header	En-tête étendu
Data Size	Taille de données
21 byte Data Block	Bloc de données de 21 octets

Tableau 6-60 Bloc de données étendues de capacités de destinataire (SCEDB)

Décalage	Champ	Taille	Valeur	Description
0	VID	2	Numérique	ID de fournisseur (affecté par l'USB-IF)
2	PID	2	Numérique	ID de produit (affecté par le fabricant)
4	XID	4	Numérique	Valeur affectée au produit, fournie par l'USB-IF
8	Version FW	1	Numérique	Numéro de version de micrologiciel
9	Version HW	1	Numérique	Numéro de version matérielle
10	Version SKEDB	1	Numérique	Version SKEDB (pas la version de la spécification): Version 1.0 = 1 Les valeurs de 0 à 2-255 sont ***Réservées*** et ***Ne doit/doivent pas*** être utilisées.
11	Palier de charge	1	Champ de bits	**Bit** / **Description** 1…0 : 00b: Palier de charge de 150 mA/µs (par défaut) 01b: Palier de charge de 500 mA/µs 11b…10b: ***Réservé(s/ée/ées)*** et ***Ne doit/doivent pas*** être utilisé 2…7 : ***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0.
12	Sink Load Characteristics	2	Champ de bits	**Bit** / **Description** 0…4 : Pourcentage de surcharge, par incréments de 10 % Les valeurs supérieures à 25 (11001b) sont écrêtées à 250 %. 00000b est la valeur par défaut. 5…10 : Période de surcharge de 20 ms lorsque les bits 0-4 sont différents de zéro. 11.14 : Cycle de service par incréments de 5 % lorsque les bits 0-4 sont différents de zéro 15 : Peut tolérer l'écart de statisme V_{BUS}
14	Conformité	1	Champ de bits	**Bit** / **Description** 0 : Exige une source LPS lorsqu'il est défini 1 : Exige une source PS1 lorsqu'il est défini 2 : Exige une source PS2 lorsqu'il est défini 3…7 : ***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0.
15	Température de contact	1	Valeur	La température se conforme à: 0 = Non applicable 1 = ***[IEC 60950-1]*** (défaut) 2 = ***[IEC 62368-1]*** TS1 3 = ***[IEC 62368-1]*** TS2 Note: Toutes les autres valeurs sont ***Réservé(s/ée/ées)***
16	Informations sur la batterie	1	Octet	Quartet supérieur = Nombre d'emplacements de batteries remplaçables à chaud (0…4) Quartet inférieur = Nombre de batteries fixes (0…4)

Décalage	Champ	Taille	Valeur	Description
17	Sink Modes	1	Champ de bits	**Bit** / **Description** 0 : 1: Charge PPS prise en charge 1 : 1: Alimenté par VBUS 2 : 1: Alimenté par le réseau 3 : 1: Alimenté par batterie 4 : 1: Batterie substantiellement illimitée 5…7 : ***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0.
18	Sink Minimum PDP	1	Octet	**Bit** / **Description** 0…6 : La PDP minimale exigée par le destinataire pour fonctionner sans consommer la puissance de sa ou ses batteries, le cas échéant. 7 : ***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0.
19	Sink Operational PDP	1	Octet	**Bit** / **Description** 0…6 : La PDP exigée par le destinataire pour fonctionner normalement. Pour les destinataires avec une batterie, il s'agit de la PDP assignée du chargeur associé ou recommandé. 7 : ***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0.
20	Sink Maximum PDP	1	Octet	**Bit** / **Description** 0…6 : La PDP maximale que le destinataire peut consommer pour fonctionner et pour charger sa ou ses batteries, le cas échéant. 7 : ***Réservé(s/ée/ées)*** et ***Devoir/Doit/Doivent*** être défini sur 0.

6.5.13.1 Champ Vendor ID (VID)

Le champ VID ***Devoir/Doit/Doivent*** contenir la valeur de l'ID de fournisseur (VID) de 16 bits affecté par l'USB-IF au fournisseur du destinataire. Si le fournisseur ne dispose pas de VID, le champ Vendor ID doit être défini sur 0. Les dispositifs dotés d'une interface de données USB ***Devoir/Doit/Doivent*** renseigner un VID identique à l'idVendor du descripteur Standard Device (voir ***[USB 2.0]*** et ***[USB 3.2]***).

6.5.13.2 Champ Product ID (PID)

Le champ PID ***Devoir/Doit/Doivent*** contenir la valeur de l'ID de produit de 16 bits affecté par le fournisseur du destinataire. Les dispositifs dotés d'une interface de données USB ***Devoir/Doit/Doivent*** renseigner un PID identique à l'idProduct du descripteur Standard Device (voir ***[USB 2.0]*** et ***[USB 3.2]***).

6.5.13.3 Champ XID

Le champ XID ***Devoir/Doit/Doivent*** contenir le XID de 32 bits fourni par l'USB-IF au fournisseur qui, à son tour, l'affecte à un produit. Si le fournisseur ne dispose pas de XID, il ***Devoir/Doit/Doivent*** renvoyer zéro dans ce champ (voir ***[USB 2.0]*** et ***[USB 3.2]***).

6.5.13.4 Champ Firmware Version

Le champ Firmware version ***Devoir/Doit/Doivent*** contenir un numéro de version de micrologiciel de 8 bits affecté au dispositif par le fournisseur.

6.5.13.5 Champ Hardware Version

Le champ Hardware Version ***Devoir/Doit/Doivent*** contenir un numéro de version matérielle de 8 bits affecté au dispositif par le fournisseur.

6.5.13.6 Champ SKEDB Version

Le champ SKEDB version contient le niveau de version du SCEDB. Actuellement, seule la version 1 est définie.

6.5.13.7 Champ Load Step

Le champ Voltage Regulation contient des bits indiquant la magnitude et la vitesse de balayage des paliers de charge préférentielles du destinataire. Voir 7.1.12.1 pour plus d'informations.

6.5.13.8 Champ Sink Load Characteristics

Le destinataire ***Devoir/Doit/Doivent*** signaler ses caractéristiques de charge préférentielles. Indépendamment de cette valeur, lors du fonctionnement, sa charge ***Ne doit/doivent pas*** dépasser les capacités indiquées dans le message ***Source_Capabilities_Extended***.

6.5.13.9 Champ Compliance

Le champ Compliance ***Devoir/Doit/Doivent*** contenir les types de sources avec lesquels le destinataire a été soumis à l'essai et certifié (voir 7.1.12.3).

6.5.13.10 Température de contact

Le champ Touch Temp ***Devoir/Doit/Doivent*** indiquer la norme IEC utilisée pour déterminer la température de surface de l'enceinte du destinataire. Les limites de sécurité pour la température de contact du destinataire sont définies dans les normes applicables en matière de sécurité des produits (par exemple, l'***[IEC 60950-1]*** ou l'***[IEC 62368-1]***). Le destinataire ***Pouvoir/Peut/Peuvent*** indiquer lorsque ses performances relatives à la température de contact se conforment aux limites TS1 et TS2 décrites dans l'***[IEC 62368-1]***.

6.5.13.11 Informations sur la batterie

Le champ Batteries Info ***Devoir/Doit/Doivent*** indiquer le nombre de batteries fixes et d'emplacements de batteries remplaçables à chaud que le destinataire prend en charge. Ce champ ***Devoir/Doit/Doivent*** indiquer indépendamment le nombre d'emplacements de batteries et le nombre de batteries fixes. Les informations indiquées dans le champ Battery Info ***Devoir/Doit/Doivent*** correspondre à celles indiquées dans le champ Battery Info du message ***Source_Capabilities_Extended***.

Un destinataire ne ***Devoir/Doit/Doivent*** pas disposer de plus de 4 batteries fixes ni de plus de 4 emplacements de batteries.

Les batteries fixes ***Devoir/Doit/Doivent*** être numérotées consécutivement de 0 à 3. Le numéro assigné à une batterie fixe donnée ***Ne doit/doivent pas*** varier lorsqu'elle est branchée ou débranchée.

Les emplacements de batteries ***Devoir/Doit/Doivent*** être numérotés consécutivement de 4 à 7. Le numéro assigné à un emplacement de batterie donné ***Ne doit/doivent pas*** varier lorsque cette dernière est branchée ou débranchée.

6.5.13.12 Sink Modes

Le champ de bits Sink Modes ***Devoir/Doit/Doivent*** identifier les capacités de charge et les sources d'alimentation qui peuvent être utilisées par le destinataire. Lorsque le bit 0 est défini, le destinataire a la possibilité d'utiliser une source PPS pour une charge rapide.

La source d'alimentation qu'un destinataire peut utiliser:

- lorsque le bit 1 est défini, le destinataire a la possibilité d'être alimenté par Vbus;
- lorsque le bit 2 est défini, le destinataire a la possibilité d'être alimenté par un réseau électrique externe;
- lorsque le bit 3 est défini, le destinataire a la possibilité d'être alimenté par une batterie;
- lorsque le bit 4 est défini, le destinataire a la possibilité d'être alimenté par une batterie dont la capacité est substantiellement infinie (par exemple, une batterie de voiture).

Les bits 1-4 ***Pouvoir/Peut/Peuvent*** être définis indépendamment les uns des autres. La combinaison indique les sources d'alimentation que le destinataire peut utiliser. Par exemple, certains destinataires ne sont alimentés que par une batterie (par exemple une batterie automobile) et non par le secteur, ce qui est pourtant plus commun, et certains destinataires ne sont alimentés que par VBUS ou VCONN.

6.5.13.13 Sink Minimum PDP

Le champ Sink Minimum PDP ***Devoir/Doit/Doivent*** contenir la puissance minimale exigée par le destinataire, arrondie à l'entier supérieur, pour utiliser tous ses modes de fonctionnement, à l'exception de la charge de sa batterie, le cas échéant. Le champ Sink Minimum PDP ***Devoir/Doit/Doivent*** être inférieur ou égal à la PDP opérationnelle du destinataire. La valeur est utilisée par la source pour déterminer si elle dispose ou non d'une puissance suffisante pour prendre en charge au minimum le destinataire branché.

6.5.13.14 Sink Operational PDP

Le champ Sink Operational PDP ***Devoir/Doit/Doivent*** contenir la PDP recommandée par le fabricant du destinataire, arrondie à l'entier supérieur. Cela correspond à la PDP assignée des sources avec lesquelles le destinataire est conçu pour fonctionner (voir 10.3.2). La PDP opérationnelle du destinataire ***Devoir/Doit/Doivent*** être suffisante pour utiliser normalement tous les modes de fonctionnement du destinataire ET pour charger la batterie du destinataire, le cas échéant. Pour les destinataires avec une ou plusieurs batteries, elle ***Devoir/Doit/Doivent*** correspondre à la PDP assignée du chargeur fourni avec le destinataire ou à la PDP assignée du chargeur recommandé.

6.5.13.15 Sink Maximum PDP

La PDP maximale du destinataire ***Devoir/Doit/Doivent*** être la plus importante quantité de puissance que le destinataire consomme dans toutes les conditions de fonctionnement, arrondie à l'entier supérieur, y compris la charge de sa batterie, le cas échéant. Le champ Sink Maximum PDP ***Ne doit/doivent pas*** être inférieur à la PDP opérationnelle du destinataire, mais ***Pouvoir/Peut/Peuvent*** être le même. La valeur est utilisée par la source pour déterminer la quantité maximale de puissance dont elle dispose pour le destinataire branché.

6.6 Temporisateurs

Tous les temporisateurs suivants sont définis en bits sur le bus, quel que soit l'endroit de l'architecture logique où ils sont mis en œuvre. Cela permet d'assurer une référence fixe pour le démarrage et l'arrêt des temporisateurs. Il appartient à l'implémenteur de faire en sorte que cette temporisation soit observée dans un système réel.

6.6.1 CRCReceiveTimer

Le ***CRCReceiveTimer*** ***Devoir/Doit/Doivent*** être utilisé par la couche protocole de l'expéditeur pour s'assurer qu'un message n'a pas été perdu. L'échec de réception de l'acquittement d'un message (un message ***GoodCRC***), du fait d'un mauvais CRC côté réception ou d'un message altéré, dans un délai ***tReceive*** est détecté à l'expiration du ***CRCReceiveTimer***.

La réponse de la couche protocole de l'expéditeur à l'expiration d'un ***CRCReceiveTimer*** ***Devoir/Doit/Doivent*** être de réessayer un nombre de fois égal à ***nRetryCount***. Note: Les fiches de câbles ne réessayent pas d'envoyer les messages, et les messages étendus de taille importante qui ne sont pas fragmentés ne sont pas relancés (voir 6.7.2). L'envoi du préambule correspondant au message relancé ***Devoir/Doit/Doivent*** commencer dans un délai ***tRetry*** après l'expiration du ***CRCReceiveTimer***.

Le ***CRCReceiveTimer*** ***Devoir/Doit/Doivent*** être lancé lorsque dernier bit de l'***EOP*** du message a été transmis par la couche physique. Le ***CRCReceiveTimer*** ***Devoir/Doit/Doivent*** être arrêté lorsque dernier bit de l'***EOP*** correspondant au message ***GoodCRC*** a été reçu par la couche physique.

La couche protocole qui reçoit un message ***Devoir/Doit/Doivent*** répondre par un message ***GoodCRC*** dans un délai ***tTransmit*** pour s'assurer que le ***CRCReceiveTimer*** de l'expéditeur n'expire pas. Le ***tTransmit*** ***Devoir/Doit/Doivent*** être mesuré à partir du moment où le dernier bit de l'***EOP*** du message a été reçu par

la couche physique, jusqu'au moment où le premier bit du préambule du message ***GoodCRC*** a été transmis par la couche physique.

6.6.2 SenderResponseTimer

Le ***SenderResponseTimer*** ***Devoir/Doit/Doivent*** être utilisé par le moteur de politique de l'expéditeur pour s'assurer qu'un message demandant une réponse (par exemple, un message ***Get_Source_Cap***) fasse l'objet d'une réponse dans un délai limité ***tSenderResponse***. L'échec de réception de la réponse attendue est détecté à l'expiration du ***SenderResponseTimer***.

La réponse du moteur de politique à l'expiration du ***SenderResponseTimer*** ***Devoir/Doit/Doivent*** dépendre du message envoyé (voir 8.3).

Le ***SenderResponseTimer*** ***Devoir/Doit/Doivent*** être lancé au moment où le dernier bit de l'***EOP*** du message ***GoodCRC*** (c'est-à-dire le message ***GoodCRC*** correspondant au message demandant une réponse) a été reçu par la couche physique. Le ***SenderResponseTimer*** ***Devoir/Doit/Doivent*** être arrêté lorsque le dernier bit de l'***EOP*** du message de réponse attendu a bien été reçu par la couche physique.

Le récepteur d'un message demandant une réponse ***Devoir/Doit/Doivent*** répondre dans un délai ***tReceiverResponse*** pour s'assurer que le ***SenderResponseTimer*** de l'expéditeur n'expire pas.

Le délai ***tReceiverResponse*** ***Devoir/Doit/Doivent*** être mesuré à partir du moment où le dernier bit de l'***EOP*** du message a été reçu par la couche physique, jusqu'au moment où le premier bit du préambule du message de réponse a été transmis par la couche physique.

6.6.3 Temporisateurs de capacité

Les sources et les destinataires utilisent des temporisateurs de capacité pour déterminer si un dispositif apte à l'alimentation USB est branché. Par le biais d'envois ou de demandes périodiques de capacités, il est possible de déterminer si un dispositif PD est branché lorsqu'une réponse est reçue.

6.6.3.1 SourceCapabilityTimer

Avant une négociation réussie, une source ***Devoir/Doit/Doivent*** utiliser le ***SourceCapabilityTimer*** pour envoyer périodiquement un message ***Source_Capabilities*** à intervalles de ***tTypeCSendSourceCap*** lorsque:

- le port est branché;
- la source n'est pas dans une connexion active avec un port destinataire d'alimentation USB.

S'il y a un temporisateur ***SourceCapabilityTimer***, la source ***Devoir/Doit/Doivent*** envoyer un message ***Source_Capabilities***. Elle ***Devoir/Doit/Doivent*** ensuite réinitialiser et relancer le ***SourceCapabilityTimer***. Le ***SourceCapabilityTimer*** ***Devoir/Doit/Doivent*** être arrêté lorsque dernier bit de l'***EOP*** correspondant au message ***GoodCRC*** a été reçu par la couche physique, à la suite de l'établissement d'une connexion d'alimentation. A ce stade, la source attend un message ***Request*** ou une temporisation de réponse.

Voir 8.3.3.2 pour une description plus détaillée du moment où les messages ***Source_Capabilities*** sont transmis.

6.6.3.2 SinkWaitCapTimer

Le destinataire ***Devoir/Doit/Doivent*** prendre en charge ***SinkWaitCapTimer***. Lorsqu'un destinataire observe une absence de messages ***Source_Capabilities*** après la présence de V_{BUS}, pendant une durée ***tTypeCSinkWaitCap***, le destinataire ***Devoir/Doit/Doivent*** émettre un signal ***Hard Reset*** pour relancer l'envoi des messages ***Source_Capabilities*** par la source (voir 6.7.4).

Voir 8.3.3.3 pour une description plus détaillée du moment où les temporisateurs ***SinkWaitCapTimer*** sont exécutés.

6.6.3.3 tFirstSourceCap

Après le branchement des ports partenaires, après une réinitialisation matérielle ou après une permutation des rôles d'alimentation ou une permutation rapide des rôles, une source ***Devoir/Doit/Doivent*** envoyer son premier message ***Source_Capabilities*** pendant le délai ***tFirstSourceCap*** du V_{BUS} atteignant ***vSafe5V***. Cela permet de s'assurer que le destinataire reçoit un message ***Source_Capabilities*** avant l'expiration de son ***SinkWaitCapTimer***.

6.6.4 Temporisateurs d'attente et temps d'attente

6.6.4.1 SinkRequestTimer

SinkRequestTimer est utilisé pour s'assurer que le délai avant le prochain message ***Request*** du destinataire, après un message ***Wait*** reçu en provenance de la source en réponse à un message ***Request*** du destinataire, est au minimum de ***tSinkRequest*** min (voir 6.3.12).

Le ***SinkRequestTimer*** ***Devoir/Doit/Doivent*** être lancé lorsque l'***EOP*** d'un message ***Wait*** a été reçu et ***Devoir/Doit/Doivent*** être arrêté si un autre message est reçu, ou bien pendant une réinitialisation matérielle.

Le destinataire ***Devoir/Doit/Doivent*** attendre au moins ***tSinkRequest***, après la réception de l'***EOP*** d'un message ***Wait*** envoyé en réponse à un message ***Request*** du destinataire, avant d'envoyer un nouveau message ***Request***. S'il y a une temporisation ***SinkRequestTimer***, le destinataire ***Pouvoir/Peut/Peuvent*** envoyer un message ***Request***. Elle ***Devoir/Doit/Doivent*** ensuite réinitialiser et relancer le ***SinkRequestTimer***.

6.6.4.2 tPRSwapWait

Le délai avant le prochain message ***PR_Swap***, après la réception d'un message ***Wait*** en réponse à un message ***PR_Swap***, est au minimum de ***tPRSwapWait*** min (voir 6.3.12). Le port ***Devoir/Doit/Doivent*** attendre au moins ***tPRSwapWait***, après la réception de l'***EOP*** d'un message ***Wait*** envoyé en réponse à un message ***PR_Swap***, avant d'envoyer un nouveau message ***PR_Swap***.

6.6.4.3 tDRSwapWait

Le délai avant le prochain message ***DR_Swap***, après la réception d'un message ***Wait*** en réponse à un message ***DR_Swap***, est au minimum de ***tDRSwapWait*** min (voir 6.3.12). Le port ***Devoir/Doit/Doivent*** attendre au moins ***tDRSwapWait***, après la réception de l'***EOP*** d'un message ***Wait*** envoyé en réponse à un message ***DR_Swap***, avant d'envoyer un nouveau message ***DR_Swap***.

6.6.4.4 tVconnSwapWait

Le délai avant le prochain message ***VCONN_Swap***, après la réception d'un message ***Wait*** en réponse à un message ***VCONN_Swap***, est au minimum de ***tVCONNSwapWait*** min (voir 6.3.12). Le port ***Devoir/Doit/Doivent*** attendre au moins ***tVCONNSwapWait***, après la réception de l'***EOP*** d'un message ***Wait*** envoyé en réponse à un message ***VCONN_Swap***, avant d'envoyer un nouveau message ***VCONN_Swap***.

6.6.5 Temporisateurs d'alimentation électrique

6.6.5.1 PSTransitionTimer

PSTransitionTimer est utilisé par le moteur de politique pour temporiser à la suite d'un message ***PS_RDY***. Il est lancé lorsqu'une demande de nouvelle capacité a été acceptée et expire après ***tPSTransition*** si aucun message ***PS_RDY*** n'a été reçu. Cette condition conduit à une réinitialisation matérielle et à un retour au fonctionnement USB par défaut. Le ***PSTransitionTimer*** se rapporte au temps nécessaire à la source pour passer d'un niveau de tension ou de courant à un autre (voir 7.1).

Le ***PSTransitionTimer*** ***Devoir/Doit/Doivent*** être lancé à la réception du dernier bit de l'***EOP*** d'un message ***Accept*** ou ***GotoMin*** par la couche physique. Le ***PSTransitionTimer*** ***Devoir/Doit/Doivent*** être arrêté lorsque dernier bit de l'***EOP*** du message ***PS_RDY*** a été reçu par la couche physique.

6.6.5.2 PSSourceOffTimer

6.6.5.2.1 Utilisation pendant la permutation des rôles d'alimentation

PSSourceOffTimer est utilisé par le moteur de politique d'un dispositif d'alimentation double fonction qui agit actuellement comme un destinataire, pour temporiser à la suite d'un message ***PS_RDY*** pendant une séquence de permutation des rôles d'alimentation. Cette condition donne lieu à un rétablissement sur erreur USB Type-C.

Si une demande de message ***PR_Swap*** a été envoyée par le dispositif d'alimentation double fonction qui agit actuellement comme une source, le destinataire peut répondre par un message ***Accept***. Lorsque le dernier bit de l'***EOP*** du message ***GoodCRC*** correspondant à ce message ***Accept*** est reçu par le destinataire, alors le ***PSSourceOffTimer*** ***Devoir/Doit/Doivent*** être lancé.

Si une demande de message ***PR_Swap*** a été envoyée par le dispositif d'alimentation double fonction qui agit actuellement comme un destinataire, la source peut répondre par un message ***Accept***. Lorsque le dernier bit de l'***EOP*** de ce message ***Accept*** est reçu par le destinataire, alors le ***PSSourceOffTimer*** ***Devoir/Doit/Doivent*** être lancé.

Le ***PSSourceOffTimer*** ***Devoir/Doit/Doivent*** être arrêté quand:

- le dernier bit de l'***EOP*** du message ***PS_RDY*** est reçu;

PSSourceOffTimer se rapporte au temps nécessaire au dispositif d'alimentation double fonction pour arrêter d'alimenter (voir 7.3.9 et 7.3.10). Le temporisateur ***Devoir/Doit/Doivent*** expirer si aucun message ***PS_RDY*** n'a été reçu en provenance du dispositif d'alimentation double fonction distant pendant ***tPSSourceOff***, indiquant que la permutation a eu lieu.

6.6.5.2.2 Utilisation pendant la permutation rapide des rôles

PSSourceOffTimer est utilisé par le moteur de politique d'un dispositif d'alimentation double fonction qui était initialement le destinataire (et qui fournit actuellement ***vSafe5V***), pour temporiser à la suite d'un message ***PS_RDY*** pendant une séquence de permutation rapide des rôles. Cette condition donne lieu à un rétablissement sur erreur USB Type-C.

Lorsque le message de demande ***FR_Swap*** a été envoyé par le destinataire initial, la source initiale ***Devoir/Doit/Doivent*** répondre par un message ***Accept***. Lorsque dernier bit de l'***EOP*** du message ***GoodCRC*** correspondant à ce message ***Accept*** est reçu par le destinataire initial, alors le ***PSSourceOffTimer*** ***Devoir/Doit/Doivent*** être lancé.

Le ***PSSourceOffTimer*** ***Devoir/Doit/Doivent*** être arrêté quand:

- le dernier bit de l'***EOP*** du message ***PS_RDY*** est reçu.

Le ***PSSourceOffTimer*** se rapporte au temps nécessaire à la source initiale pour arrêter d'alimenter et à V_{BUS} pour revenir à ***vSafe5V*** (voir 7.2.10 et 7.3.15). Le temporisateur ***Devoir/Doit/Doivent*** expirer si aucun message ***PS_RDY*** n'a été reçu en provenance de la source initiale pendant ***tPSSourceOff***, indiquant que la permutation a eu lieu.

6.6.5.3 PSSourceOnTimer

6.6.5.3.1 Utilisation pendant la permutation des rôles d'alimentation

PSSourceOnTimer est utilisé par le moteur de politique d'un dispositif d'alimentation double fonction qui vient d'arrêter de fonctionner en tant que source et qui attend de fonctionner en tant que destinataire, pour temporiser à la suite d'un message ***PS_RDY*** pendant une permutation des rôles d'alimentation. Cette condition donne lieu à un rétablissement sur erreur USB Type-C.

Le ***PSSourceOnTimer*** ***Devoir/Doit/Doivent*** être lancé quand:

- le dernier bit de l'***EOP*** du message ***GoodCRC*** correspondant au message ***PS_RDY*** émis est reçu par la couche physique.

Le ***PSSourceOnTimer*** ***Devoir/Doit/Doivent*** être arrêté quand:

- le dernier bit de l'***EOP*** du message ***PS_RDY*** est reçu par la couche physique.

PSSourceOnTimer se réfère au temps nécessaire au dispositif d'alimentation double fonction pour commencer à fonctionner en tant source (voir aussi 7.3.9 et 7.3.10) et expire si aucun message ***PS_RDY*** indiquant cela n'a été reçu pendant ***tPSSourceOn***.

6.6.5.3.2 Utilisation pendant la permutation rapide des rôles

PSSourceOnTimer est utilisé par le moteur de politique d'un dispositif d'alimentation double fonction qui vient d'arrêter de fonctionner en tant source et qui attend de fonctionner en tant destinataire, pour temporiser à la suite d'un message ***PS_RDY*** pendant une permutation rapide des rôles. Cette condition donne lieu à un rétablissement sur erreur USB Type-C.

Le ***PSSourceOnTimer*** ***Devoir/Doit/Doivent*** être lancé quand:

- le dernier bit de l'***EOP*** du message ***GoodCRC*** correspondant au message ***PS_RDY*** émis est reçu par la couche physique.

Le ***PSSourceOnTimer*** ***Devoir/Doit/Doivent*** être arrêté quand:

- le dernier bit de l'***EOP*** du message ***PS_RDY*** est reçu par la couche physique.

PSSourceOnTimer se réfère au temps nécessaire au dispositif d'alimentation double fonction pour commencer à fonctionner en tant source (voir aussi 7.2.10 et 7.3.15) et expire si aucun message ***PS_RDY*** indiquant cela n'a été reçu pendant ***tPSSourceOn***.

6.6.6 NoResponseTimer

Le ***NoResponseTimer*** est utilisé par le moteur de politique d'une source ou d'un destinataire pour déterminer si son port partenaire ne répond pas après une réinitialisation matérielle. Lorsque ***NoResponseTimer*** expire, le moteur de politique ***Devoir/Doit/Doivent*** effectuer jusqu'à ***nHardResetCount*** réinitialisations matérielles supplémentaires avant de déterminer que le port partenaire ne répond pas aux messages d'alimentation USB.

Si la source ne reçoit pas de message ***GoodCRC*** en réponse à un message ***Source_Capabilities*** dans un délai ***tNoResponse*** à partir:

- du dernier bit d'un signal ***Hard Reset*** envoyé par la couche PHY, si le signal ***Hard Reset*** a été lancé par le destinataire;
- du dernier bit d'une signalisation ***Hard Reset*** reçu par la couche PHY, si le signal ***Hard Reset*** a été lancé par la source;

alors la source ***Devoir/Doit/Doivent*** émettre des signaux Hard Reset supplémentaires jusqu'à un nombre de fois égal à ***nHardResetCount*** (voir 6.8.3).

Si un dispositif ne répond pas, le moteur de politique d'une source ***Pouvoir/Peut/Peuvent*** décider soit de continuer à envoyer des messages ***Source_Capabilities***, soit de passer en fonctionnement sans alimentation électrique par port USB et d'arrêter d'envoyer des messages ***Source_Capabilities.***

6.6.7 Temporisateurs BIST

6.6.7.1 tBISTCarrierMode

tBISTCarrierMode est utilisé pour déterminer la durée maximale après laquelle l'UUT doit entrer en mode porteur BIST à la demande d'un dispositif d'essai.

Une UUT ***Devoir/Doit/Doivent*** entrer dans le mode porteur BIST dans un délai ***tBISTCarrierMode*** à partir de la réception du dernier bit ***EOP*** du message ***BIST*** utilisé pour démarrer l'essai par la couche physique. Dans ***Mode porteur BIST***, lors de l'émission d'un signal porteur continu, l'émission ***Devoir/Doit/Doivent*** commencer dès que l'UUT entre en mode BIST.

6.6.7.2 BISTContModeTimer

BISTContModeTimer est utilisé par une UUT pour s'assurer qu'un mode BIST continu (c'est-à-dire ***Mode porteur BIST***) est quitté en temps utile. Une UUT qui est entrée en mode BIST continu ***Devoir/Doit/Doivent*** revenir au fonctionnement normal (***PE_SRC_Transition_to_default***, ***PE_SNK_Transition_to_default*** ou ***PE_CBL_Ready***) au cours de ***tBISTContMode*** avant de commencer la transmission d'un signal porteur continu.

6.6.7.3 tBISTSharedTestMode

tBISTSharedTestMode est utilisé pour déterminer la durée maximale après laquelle l'UUT doit entrer en mode d'essai BIST de capacité partagée à la demande d'un dispositif d'essai.

Une UUT ***Devoir/Doit/Doivent*** entrer en mode d'essai BIST de capacité partagée et envoyer un nouveau message ***Source_Capabilities*** à partir de tous les ports du groupe à capacité partagée dans un délai ***tBISTSharedTestMode*** à partir de la réception du dernier bit ***EOP*** du message ***BIST*** utilisé pour démarrer l'essai par la couche physique.

6.6.8 Temporisateurs de permutation des rôles d'alimentation

6.6.8.1 SwapSourceStartTimer

Le ***SwapSourceStartTimer*** ***Devoir/Doit/Doivent*** être utilisé par la nouvelle source, après une permutation des rôles d'alimentation ou une permutation rapide des rôles, pour s'assurer qu'elle n'envoie pas de message ***Source_Capabilities*** avant que le nouveau destinataire ne soit prêt à recevoir le message ***Source_Capabilities***. La nouvelle source ***Ne doit/doivent pas*** envoyer le message ***Source_Capabilities*** avant un délai ***tSwapSourceStart*** après le dernier bit de l'***EOP*** du message ***GoodCRC*** envoyé en réponse au message ***PS_RDY*** envoyé par la nouvelle source, indiquant que son alimentation est prête. Le destinataire ***Devoir/Doit/Doivent*** être prêt à recevoir un message ***Source_Capabilities*** dans un délai ***tSwapSinkReady*** après avoir envoyé le dernier bit de l'***EOP*** du message ***GoodCRC*** envoyé en réponse au message ***PS_RDY*** envoyé par la nouvelle source, indiquant que son alimentation est prête.

6.6.9 Temporisateurs de réinitialisation logicielle

6.6.9.1 tSoftReset

L'échec de réception d'un message ***GoodCRC*** en réponse à tout message dans un délai ***tReceive*** (après ***nRetryCount*** nouvelles tentatives), lorsqu'une paire de ports est connectée, indique un défaut des communications. Cet événement ***Devoir/Doit/Doivent*** donner lieu à l'envoi d'un message ***Soft_Reset*** par la source ou le destinataire, dont la transmission ***Devoir/Doit/Doivent*** être terminée dans un délai ***tSoftReset*** à partir de l'expiration de ***CRCReceiveTimer***.

6.6.9.2 tProtErrSoftReset

Si une erreur de protocole se produit et qu'elle donne lieu à l'envoi d'un message ***Soft_Reset*** par la source ou le destinataire, la transmission du message ***Soft_Reset*** ***Devoir/Doit/Doivent*** être terminée dans un délai ***tProtErrSoftReset*** à partir de l'***EOP*** du ***GoodCRC*** envoyé en réponse au message à l'origine de l'erreur de protocole.

6.6.10 Temporisateurs de réinitialisation des données

6.6.10.1 VCONNDischargeTimer

Le ***VCONNDischargeTimer*** est utilisé par le moteur de politique dans le DFP pour s'assurer que l'UFP décharge activement VCONN en temps opportun afin de s'assurer que le câble restaure Ra. Lorsque l'UFP a déchargé VCONN au-dessous de la valeur vRaReconnect (voir ***[USB Type-C 2.0]***), il envoie un message ***PS_RDY*** (voir également 7.1.15.1).

Si le DFP ne reçoit pas de message ***PS_RDY*** de l'UFP dans un délai ***tVCONNSourceDischarge*** à partir de la réception du dernier bit du ***GoodCRC*** qui acquitte le message ***Accept*** en réponse au message ***Data_Reset***,

le ***VCONNDischargeTimer*** expire et le moteur de politique ***Devoir/Doit/Doivent*** passer à l'état ***ErrorRecovery***.

6.6.10.2 tDataReset

Le DFP ***Devoir/Doit/Doivent*** terminer le processus Data_Reset (défini en 6.3.14) dans un délai ***tDataReset***:

- à partir de la réception du dernier bit du ***GoodCRC*** qui acquitte le message ***Accept*** lors de l'envoi du message ***Data_Reset*** par le DFP; ou
- à partir de la réception du dernier bit du message ***Accept*** lors de l'envoi du message ***Data_Reset*** par l'UFP.

6.6.10.3 DataResetFailTimer

Le ***DataResetFailTimer*** ***Devoir/Doit/Doivent*** être utilisé par le moteur de politique du DFP pour s'assurer que le processus de réinitialisation des données se termine dans un délai ***tDataResetFail*** à partir de la réception du dernier bit du ***GoodCRC*** qui acquitte le message ***Accept*** en réponse au message ***Data_Reset***. Si le ***DataResetFailTimer*** du DFP expire, le DFP ***Devoir/Doit/Doivent*** passer à l'état ***ErrorRecovery***.

6.6.11 Temporisateurs de réinitialisation matérielle

6.6.11.1 HardResetCompleteTimer

HardResetCompleteTimer est utilisé par la couche protocole dans le cas où elle a demandé à la couche PHY d'envoyer un signal ***Hard Reset*** et que la couche PHY est incapable d'envoyer le signal dans un délai raisonnable du fait d'un canal non inactif. Si la couche PHY n'indique pas que le signal ***Hard Reset*** a été envoyé dans un délai ***tHardResetComplete*** après la demande de transmission de la couche protocole, alors la couche protocole ***Devoir/Doit/Doivent*** informer le moteur de politique que le signal ***Hard Reset*** a été envoyé pour s'assurer que l'alimentation a été réinitialisée en temps utile.

6.6.11.2 PSHardResetTimer

PSHardResetTimer est utilisé par le moteur de politique d'une source pour s'assurer que le destinataire disposait d'un temps suffisant pour traiter le signal ***Hard Reset*** avant de désactiver son alimentation vers V_{BUS}.

Lors d'une réinitialisation matérielle, la source arrête de piloter V_{CONN}, supprime Rp de la broche V_{CONN} et commence à faire passer la tension de V_{BUS} à ***vSafe0V***, soit:

- ***tPSHardReset*** lorsque le dernier bit du signal ***Hard Reset*** a été reçu en provenance du destinataire; ou
- ***tPSHardReset*** après envoi du dernier bit du signal ***Hard Reset*** par la source.

Voir Section 7.1.5.

6.6.11.3 tDRSwapHardReset

Si un message ***DR_Swap*** est reçu pendant le fonctionnement modal, alors une réinitialisation matérielle ***Devoir/Doit/Doivent*** être initiée par le récepteur du message ***DR_Swap*** inattendu; le signal ***Hard Reset*** ***Devoir/Doit/Doivent*** être généré dans un délai ***tDRSwapHardReset*** après l'EOP du ***GoodCRC*** envoyé en réponse au message ***DR_Swap***.

6.6.11.4 tProtErrHardReset

Si une erreur de protocole se produit et qu'elle donne directement lieu à une réinitialisation matérielle, la transmission du signal ***Hard Reset*** ***Devoir/Doit/Doivent*** être terminée dans un délai ***tProtErrHardReset*** à partir de l'***EOP*** du ***GoodCRC*** envoyé en réponse au message à l'origine de l'erreur de protocole.

6.6.12 Temporisateurs de VDM structuré

6.6.12.1 VDMResponseTimer

Le ***VDMResponseTimer*** ***Devoir/Doit/Doivent*** être utilisé par le moteur de politique de l'initiateur pour s'assurer qu'une demande de commande de VDM structuré qui nécessite une réponse (par exemple, une demande de commande ***Discover Identity***) fasse l'objet d'une réponse dans un délai limité de ***tVDMSenderResponse***. Le ***VDMResponseTimer*** ***Devoir/Doit/Doivent*** être appliqué à toutes les commandes VDM structurées à l'exception des commandes ***Enter Mode*** et***Exit Mode*** qui ont leurs propres temporisateurs (respectivement ***VDMModeEntryTimer*** et ***VDMModeExitTimer***). L'échec de réception de la réponse attendue est détecté à l'expiration du ***VDMResponseTimer***.

La réponse du moteur de politique à l'expiration du ***VDMResponseTimer*** ***Devoir/Doit/Doivent*** dépendre du message envoyé (voir 8.3).

Le ***VDMResponseTimer*** ***Devoir/Doit/Doivent*** être lancé au moment où le dernier bit de l'***EOP*** du message ***GoodCRC*** (c'est-à-dire le message ***GoodCRC*** correspondant à la commande de VDM qui demande une réponse) a été reçu par la couche physique. Le ***VDMResponseTimer*** ***Devoir/Doit/Doivent*** être arrêté lorsque le dernier bit de l'***EOP*** de la réponse de commande de VDM attendue a été reçu par la couche physique.

Le récepteur d'un message demandant une réponse ***Devoir/Doit/Doivent*** répondre dans un délai ***tVDMReceiverResponse*** pour s'assurer que le ***VDMResponseTimer*** de l'expéditeur n'expire pas.

Le délai ***tVDMReceiverResponse*** ***Devoir/Doit/Doivent*** être mesuré à partir du moment où le dernier bit de l'***EOP*** du message a été reçu par la couche physique, jusqu'au moment où le premier bit du préambule du message de réponse a été transmis par la couche physique.

6.6.12.2 VDMModeEntryTimer

Le ***VDMModeEntryTimer*** ***Devoir/Doit/Doivent*** être utilisé par le moteur de politique de l'initiateur pour s'assurer que la réponse à une demande de commande de VDM structuré ***Enter Mode*** (ACK ou NAK avec ACK indiquant que le mode demandé est activé) arrive dans un délai limité ***tVDMWaitModeEntry***. L'échec de réception de la réponse attendue est détecté à l'expiration du ***VDMModeEntryTimer***.

La réponse du moteur de politique à l'expiration de ***VDMModeEntryTimer*** permet d'informer le gestionnaire de politique d'utilisation des dispositifs (voir 8.3.3.22.1).

Le ***VDMModeEntryTimer*** ***Devoir/Doit/Doivent*** être lancé au moment où le dernier bit de l'***EOP*** du message ***GoodCRC*** (c'est-à-dire le message ***GoodCRC*** correspondant à la demande de commande de VDM) a été reçu par la couche physique. Le ***VDMModeEntryTimer*** ***Devoir/Doit/Doivent*** être arrêté lorsque le dernier bit de l'***EOP*** de la réponse de commande de VDM structuré attendue (ACK, NAK ou BUSY) a été reçu par la couche physique.

Le récepteur d'un message demandant une réponse ***Devoir/Doit/Doivent*** répondre dans un délai ***tVDMEnterMode*** pour s'assurer que le ***VDMModeEntryTimer*** de l'expéditeur n'expire pas.

Le délai ***tVDMEnterMode*** ***Devoir/Doit/Doivent*** être mesuré à partir du moment où le dernier bit de l'***EOP*** du message a été reçu par la couche physique, jusqu'au moment où le premier bit du préambule du message de réponse a été transmis par la couche physique.

6.6.12.3 VDMModeExitTimer

Le ***VDMModeExitTimer*** ***Devoir/Doit/Doivent*** être utilisé par le moteur de politique de l'initiateur pour s'assurer que la réponse ACK à une commande de VDM structuré ***Exit Mode***, indiquant que le mode demandé a été quitté, arrive dans un délai limité de ***tVDMWaitModeExit***. L'échec de réception de la réponse attendue est détecté à l'expiration du ***VDMModeExitTimer***.

La réponse du moteur de politique à l'expiration de ***VDMModeExitTimer*** permet d'informer le gestionnaire de politique d'utilisation des dispositifs (voir 8.3.3.22.2).

Le ***VDMModeExitTimer*** ***Devoir/Doit/Doivent*** être lancé au moment où le dernier bit de l'***EOP*** du message ***GoodCRC*** (c'est-à-dire le message ***GoodCRC*** correspondant à la commande de VDM qui demande une

réponse) a été reçu par la couche physique. Le ***VDMModeExitTimer*** ***Devoir/Doit/Doivent*** être arrêté lorsque le dernier bit de l'***EOP*** de la réponse de commande de VDM structuré attendue ACK a été reçu par la couche physique.

Le récepteur d'un message demandant une réponse ***Devoir/Doit/Doivent*** répondre dans un délai ***tVDMExitMode*** pour s'assurer que le ***VDMModeExitTimer*** de l'expéditeur n'expire pas.

Le délai ***tVDMExitMode*** ***Devoir/Doit/Doivent*** être mesuré à partir du moment où le dernier bit de l'***EOP*** du message a été reçu par la couche physique, jusqu'au moment où le premier bit du préambule du message de réponse a été transmis par la couche physique.

6.6.12.4 tVDMBusy

L'initiateur ***Devoir/Doit/Doivent*** attendre au moins ***tVDMBusy***, après la réception de la réponse de commande BUSY, avant de répéter la demande de VDM structuré.

6.6.13 Temporisateurs de VCONN

6.6.13.1 VCONNOnTimer

VCONNOnTimer est utilisé pendant une permutation de VCONN.

Le ***VCONNOnTimer*** ***Devoir/Doit/Doivent*** être lancé quand:

- le dernier bit de l'***EOP*** du message ***Accept*** est reçu;
- le dernier bit de l'***EOP*** du message ***GoodCRC*** correspondant au message ***Accept*** est reçu.

Le ***VCONNOnTimer*** ***Devoir/Doit/Doivent*** être arrêté quand:

- le dernier bit de l'***EOP*** du message ***PS_RDY*** est reçu;

Avant d'envoyer le message ***PS_RDY***, le port ***Devoir/Doit/Doivent*** avoir activé VCONN.

6.6.13.2 tVCONNSourceOff

Le délai ***tVCONNSourceOff*** s'applique pendant une permutation de Vconn. La source VCONN initiale ***Devoir/Doit/Doivent*** cesser de fournir VCONN dans un délai ***tVCONNSourceOff*** à partir de la réception du dernier bit de l'***EOP*** du message ***PS_RDY***.

6.6.14 tCableMessage

Les ports conformes à la présente révision de la spécification ***Ne doit/doivent pas*** attendre ***tCableMessage*** avant d'envoyer un paquet SOP' ou SOP'', même lorsqu'ils communiquent avec une fiche de câble à l'aide de ***[USBPD 2.0]***. La présente spécification définit les mécanismes anticollisions qui rendent ce délai inutile.

Les fiches de câbles ***Devoir/Doit/Doivent*** attendre ***tCableMessage*** avant d'envoyer un paquet SOP' ou SOP'' seulement lorsqu'elles fonctionnent conformément à ***[USBPD 2.0]***. Lorsqu'elles fonctionnent à l'aide de révisions ultérieures à ***[USBPD 2.0]***, les fiches de câbles ***Ne doit/doivent pas*** attendre ***tCableMessage*** avant d'envoyer un paquet SOP' ou SOP''.

6.6.15 DiscoverIdentityTimer

DiscoverIdentityTimer est utilisé pendant un contrat explicite, en découvrant si une fiche de câble est apte à l'alimentation USB au moyen du SOP'. Pendant la découverte des câbles au cours d'un contrat explicite, la demande de commande ***Discover Identity*** ***Devoir/Doit/Doivent*** être envoyée à intervalles de ***tDiscoverIdentity***. Pas plus de ***nDiscoverIdentityCount*** messages ***Discover Identity*** sans message de réponse ***GoodCRC*** ne ***Devoir/Doit/Doivent*** être envoyés. Si aucun message de réponse ***GoodCRC*** n'est reçu lorsque ***nDiscoverIdentityCount*** demandes de commande ***Discover Identity*** ont été envoyées par un port, le port ***Ne doit/doivent pas*** envoyer d'autre message SOP'/SOP''.

6.6.16 Temporisateurs anticollisions

SinkTxTimer est utilisé par la couche protocole d'une source pour permettre au destinataire de terminer sa transmission avant de lancer une AMS.

La source ***Devoir/Doit/Doivent*** attendre au moins ***tSinkTx*** après le passage de Rp de ***SinkTxOk*** à ***SinkTxNG*** avant d'initier une AMS en envoyant un message.

Un destinataire ne ***Devoir/Doit/Doivent*** initier une AMS que lorsqu'il a déterminé que Rp est définie sur ***SinkTxOk***.

6.6.17 Temporisateurs de permutation rapide des rôles

6.6.17.1 tFRSwap5V

Au cours d'une permutation rapide des rôles, la source initiale ***Devoir/Doit/Doivent*** lancer le message ***PS_RDY*** au cours de ***tFRSwap5V*** après avoir envoyé le message ***Accept***, V_{BUS} étant à ***vSafe5V***. La durée ***tFRSwap5V*** Doit être mesurée depuis le dernier bit de l'***EOP*** pour le message ***GoodCRC*** correspondant au message ***Accept***, V_{BUS} étant comprise dans ***vSafe5V***, jusqu'au premier bit du préambule de message ***PS_RD*** transmis en réponse par la couche physique.

6.6.17.2 tFRSwapComplete

Au cours d'une permutation rapide des rôles, le destinataire initial ***Devoir/Doit/Doivent*** répondre par un message ***PS_RDY*** au cours de ***tFRSwapComplete*** après avoir reçu le message ***PS_RDY*** de la source initiale. La durée ***tFRSwapComplete*** Doit être mesurée à partir du moment où le dernier bit de l'***EOP*** du message ***PS_RDY*** a été reçu par la couche physique, jusqu'au moment où le premier bit du préambule du message de réponse ***PS_RD*** a été transmis par la couche physique.

6.6.17.3 tFRSwapInit

Ce dernier bit de l'***EOP*** du message ***FR_Swap*** ***Devoir/Doit/Doivent*** être transmis par la nouvelle source au plus tard dans un délai ***tFRSwapInit*** après détection de la demande de permutation rapide des rôles (voir 5.8.6.3).

6.6.18 Temporisateurs de fragmentation

6.6.18.1 ChunkingNotSupportedTimer

ChunkingNotSupportedTimer est utilisé par une source qui ne prend pas en charge la fragmentation en plusieurs fragments, mais qui a reçu un fragment de message.

Le ***ChunkingNotSupportedTimer*** ***Devoir/Doit/Doivent*** être lancé quand:

le dernier bit de l'EOP du fragment de message d'un message en plusieurs fragments est reçu. Le moteur de politique ***Ne doit/doivent pas*** envoyer son message Not_Supported avant l'expiration de ChunkingNotSupportedTimer.

6.6.18.2 ChunkSenderRequestTimer

ChunkSenderRequestTimer est utilisé pendant une transmission de message fragmenté.

Le ***ChunkSenderRequestTimer*** ***Devoir/Doit/Doivent*** être utilisé par le diagramme d'états de fragmentation de l'expéditeur pour s'assurer qu'une réponse de fragment fasse l'objet d'une réponse dans un délai limité ***tChunkSenderRequest***. L'échec de réception de la réponse attendue est détecté à l'expiration du ***ChunkSenderRequestTimer***.

Le ***ChunkSenderRequestTimer*** ***Devoir/Doit/Doivent*** être lancé quand:

- le dernier bit de l'***EOP*** du message ***GoodCRC*** correspondant au message de réponse de fragment est reçu.

Le ***ChunkSenderRequestTimer*** ***Devoir/Doit/Doivent*** être arrêté quand:

- le dernier bit de l'***EOP*** du message de demande de fragment est reçu;
- un message autre qu'une demande de fragment est reçu du Rx de la couche protocole.

Le récepteur d'une réponse de fragment exigeant une demande de fragment ***Devoir/Doit/Doivent*** répondre par une demande de fragment dans un délai ***tChunkReceiverRequest*** pour s'assurer que le ***ChunkSenderRequestTimer*** de l'expéditeur n'expire pas.

Le délai ***tChunkReceiverRequest*** ***Devoir/Doit/Doivent*** être mesuré à partir du moment où le dernier bit de l'***EOP*** du message a été reçu par la couche physique, jusqu'au moment où le premier bit du préambule du message de réponse a été transmis par la couche physique.

6.6.18.3 ChunkSenderResponseTimer

ChunkSenderResponseTimer est utilisé pendant une transmission de message fragmenté.

Le ***ChunkSenderResponseTimer*** ***Devoir/Doit/Doivent*** être utilisé par le diagramme d'états de fragmentation de l'expéditeur pour s'assurer qu'une demande de fragment fasse l'objet d'une réponse dans un délai limité ***tChunkSenderResponse***. L'échec de réception de la réponse attendue est détecté à l'expiration du ChunkSenderResponseTimer.

Le ***ChunkSenderResponseTimer*** ***Devoir/Doit/Doivent*** être lancé quand:

- le dernier bit de l'EOP du message GoodCRC correspondant au message de demande de fragment est reçu.

Le ***ChunkSenderResponseTimer*** ***Devoir/Doit/Doivent*** être arrêté quand:

- le dernier bit de l'EOP du message de réponse de fragment est reçu;
- un message autre qu'un fragment est reçu de la couche protocole.

Le récepteur d'une demande de fragment exigeant une réponse de fragment ***Devoir/Doit/Doivent*** répondre par une réponse de fragment dans un délai ***tChunkReceiverResponse*** pour s'assurer que le ***ChunkSenderResponseTimer*** de l'expéditeur n'expire pas.

Le délai ***tChunkReceiverResponse*** ***Devoir/Doit/Doivent*** être mesuré à partir du moment où le dernier bit de l'***EOP*** du message a été reçu par la couche physique, jusqu'au moment où le premier bit du préambule du message de réponse a été transmis par la couche physique.

6.6.19 Temporisateurs d'alimentation électrique programmable

6.6.19.1 SinkPPSPeriodicTimer

Le ***SinkPPSPeriodicTimer*** ***Devoir/Doit/Doivent*** être utilisé par le moteur de politique du destinataire pour s'assurer que la communication entre le destinataire et la source a lieu dans un délai ***tPPSRequest*** lors du fonctionnement PPS. En l'absence de toute autre circulation, un message ***Request*** demandant un APDO PPS est envoyé périodiquement en tant que mécanisme de maintien.

SinkPPSPeriodicTimer ***Devoir/Doit/Doivent*** être réinitialisé et relancé lorsque le dernier bit de l'***EOP*** d'un message quelconque est reçu, provoquant l'entrée du destinataire dans l'état ***PE_SNK_Ready***.

Le destinataire ***Devoir/Doit/Doivent*** arrêter le ***SinkPPSPeriodicTimer*** lorsque le dernier bit de l'***EOP*** d'un message quelconque ou le dernier bit d'un signal quelconque est reçu en provenance de la source et par le destinataire, provoquant la sortie du destinataire de l'état ***PE_SNK_Ready***.

6.6.19.2 SourcePPSCommTimer

Le ***SourcePPSCommTimer*** ***Devoir/Doit/Doivent*** être utilisé par le moteur de politique de la source pour s'assurer que la communication entre le destinataire et la source a lieu dans un délai ***tPPSTimeout*** lors du fonctionnement PPS. En l'absence de toute autre circulation, un message ***Request*** demandant un APDO PPS est reçu périodiquement en tant que mécanisme de maintien.

SourcePPSCommTimer ***Devoir/Doit/Doivent*** être réinitialisé et relancé lorsque le dernier bit de l'***EOP*** d'un message quelconque est reçu, provoquant l'entrée de la source dans l'état ***PE_SRC_Ready***.

La source ***Devoir/Doit/Doivent*** arrêter le ***SourcePPSCommTimer*** lorsque le dernier bit de l'***EOP*** d'un message quelconque ou le dernier bit d'un signal quelconque est reçu en provenance du destinataire par la source, provoquant la sortie de la source de l'état ***PE_SRC_Ready***.

A l'expiration du ***SourcePPSCommTimer***, la source ***Devoir/Doit/Doivent*** émettre un signal ***Hard Reset***.

6.6.20 tEnterUSB

Le DFP ***Devoir/Doit/Doivent*** envoyer le message ***Enter_USB*** dans un délai ***tEnterUSB***:

- à partir de la réception du dernier bit ***GoodCRC*** qui acquitte le message ***Accept*** en réponse au message ***Data_Reset***; ou
- à partir de la mise sous tension initiale.

Un non-respect de ce paramètre d'expiration peut empêcher les ports de passer en fonctionnement ***[USB4]***.

6.6.21 Valeurs de temporisation et temporisateurs

Le Tableau 6-61 rassemble les valeurs des temporisateurs énumérés dans le présent paragraphe. Pour chaque valeur de temporisation, une mise en œuvre donnée ***Devoir/Doit/Doivent*** choisir une valeur fixe comprise dans la plage spécifiée. Le Tableau 6-62 répertorie les temporisateurs.

Tableau 6-61 Valeurs de temporisation

Paramètre	Valeur (min)	Valeur (max)	Unités	Référence
tACTempUpdate		500	ms	Section 6.5.2.1.1
tBISTContMode	30	60	ms	Section 6.6.7.2
tBISTCarrierMode		300	ms	Section 6.6.7.1
tBISTSharedTestMode		1	s	Section 6.6.7.3
tCableMessage	750		µs	Section 6.6.14
tChunkingNotSupported	40	50	ms	Section 6.6.18.1
tChunkReceiverRequest		15	ms	Section 6.6.18.2
tChunkReceiverResponse		15	ms	Section 6.6.18.3
tChunkSenderRequest	24	30	ms	Section 6.6.18.2
tChunkSenderResponse	24	30	ms	Section 6.6.18.3
tDataReset	200	250	ms	Section 6.6.10.2
tDataResetFail	300		ms	Section 6.6.10.3
tDiscoverIdentity	40	50	ms	Section 6.6.14
tDRSwapHardReset		15	ms	Section 6.6.11.3
tDRSwapWait	100		ms	Section 6.6.4.3
tEnterUSB		500	ms	Section 6.6.20
tFirstSourceCap		250	ms	Section 6.6.3.3
tFRSwap5V		15	ms	Section 6.6.17.1
tFRSwapComplete		15	ms	Section 6.6.17.2
tFRSwapInit		15	ms	Section 6.6.17.3
tHardReset		5	ms	Section 6.3.13
tHardResetComplete	4	5	ms	Section 6.6.9
tNoResponse	4,5	5,5	s	Section 6.6.6
tPPSRequest		10	s	Section 6.6.19.1
tPPSTimeout	12	15	s	Section 6.6.19.2
tProtErrHardReset		15	ms	Section 6.6.11.4
tProtErrSoftReset		15	ms	Section 6.6.9.2
tPRSwapWait	100		ms	Section 6.6.4.2
tPSHardReset	25	35	ms	Section 6.6.11.2
tPSSourceOff	750	920	ms	Section 6.6.5.2
tPSSourceOn	390	480	ms	Section 6.6.5.3
tPSTransition	450	550	ms	Section 6.6.5.1
tReceive	0,9	1,1	ms	Section 6.6.1
tReceiverResponse		15	ms	Section 6.6.2
tRetry		195	µs	Section 6.6.1
tSenderResponse	24	30	ms	Section 6.6.2
tSinkRequest	100		ms	Section 6.6.4.1
tSinkTx	16	20	ms	Section 6.6.16
tSoftReset		15	ms	Section 6.8.1
tSwapSinkReady		15	ms	Section 6.6.8.1

Paramètre	Valeur (min)	Valeur (max)	Unités	Référence
tSwapSourceStart	20		ms	Section 6.6.8.1
tTransmit		195	µs	Section 6.6.1
tTypeCSendSourceCap	100	200	ms	Section 6.6.3.1
tTypeCSinkWaitCap	310	620	ms	Section 6.6.3.2
tVCONNSourceDischarge	160	240	ms	Section 6.6.10.1
tVCONNSourceOff		25	ms	Section 6.6.13
tVCONNSourceOn		50	ms	Section 6.3.11
TVCONNSourceTimeout	100	200	ms	Section 6.6.13
tVCONNSwapWait	100		ms	Section 6.6.4.4
tVDMBusy	50		ms	Section 6.6.12.4
tVDMEnterMode		25	ms	Section 6.6.12.2
tVDMExitMode		25	ms	Section 6.6.12.3
tVDMReceiverResponse		15	ms	Section 6.6.12.1
tVDMSenderResponse	24	30	ms	Section 6.6.12.1
tVDMWaitModeEntry	40	50	ms	Section 6.6.12.2
tVDMWaitModeExit	40	50	ms	Section 6.6.12.3

Tableau 6-62 Temporisateurs

Temporisateur	Paramètre	Utilisé par	Référence
BISTContModeTimer	*tBISTContMode*	Moteur de politique	Section 6.6.7.2
ChunkingNotSupportedTimer	*tChunkingNotSupported*	Moteur de politique	Section 6.6.18.1
ChunkSenderRequestTimer	*tChunkSenderRequest*	Protocole	Section 6.6.18.2
ChunkSenderResponseTimer	*tChunkSenderResponse*	Protocole	Section 6.6.18.3
CRCReceiveTimer	*tReceive*	Protocole	Section 6.6.1
DataResetFailTimer	*tDataResetFail*	Moteur de politique	Section 6.6.10.3
DiscoverIdentityTimer	*tDiscoverIdentity*	Moteur de politique	Section 6.6.15
HardResetCompleteTimer	*tHardResetComplete*	Protocole	Section 6.6.9
NoResponseTimer	*tNoResponse*	Moteur de politique	Section 6.6.6
PSHardResetTimer	*tPSHardReset*	Moteur de politique	Section 6.6.11.2
PSSourceOffTimer	*tPSSourceOff*	Moteur de politique	Section 6.6.5.2
PSSourceOnTimer	*tPSSourceOn*	Moteur de politique	Section 6.6.5.3
PSTransitionTimer	*tPSTransition*	Moteur de politique	Section 6.6.5.1
SenderResponseTimer	*tSenderResponse*	Moteur de politique	Section 6.6.2
SinkPPSPeriodicTimer	*tPPSRequest*	Moteur de politique	Section 6.6.19.1
SinkRequestTimer	*tSinkRequest*	Moteur de politique	Section 6.6.4

Temporisateur	Paramètre	Utilisé par	Référence
SinkWaitCapTimer	*tTypeCSinkWaitCap*	Moteur de politique	Section 6.6.3.2
SourceCapabilityTimer	*tTypeCSendSourceCap*	Moteur de politique	Section 6.6.3.1
SourcePPSCommTimer	*tPPSTimeout*	Moteur de politique	Section 6.6.19.2
SinkTxTimer	*tSinkTx*	Couche protocole	Section 6.6.16
SwapSourceStartTimer	*tSwapSourceStart*	Moteur de politique	Section 6.6.8.1
VCONNDischargeTimer	*tVCONNSourceDischarge*	Moteur de politique	Section 6.6.10.1
VCONNOnTimer	*TVCONNSourceTimeout*	Moteur de politique	Section 6.6.13.1
VDMModeEntryTimer	*tVDMWaitModeEntry*	Moteur de politique	Section 6.6.12.2
VDMModeExitTimer	*tVDMWaitModeExit*	Moteur de politique	Section 6.6.12.3
VDMResponseTimer	*tVDMSenderResponse*	Moteur de politique	Section 6.6.12.1

6.7 Compteurs

6.7.1 Compteur MessageID

MessageIDCounter est un compteur perpétuel qui s'étend de 0 à ***nMessageIDCount***, utilisé pour détecter les messages dupliqués. Cette valeur est utilisée pour le champ ***MessageID*** dans l'en-tête de message pour chaque message transmis.

Chaque port ***Devoir/Doit/Doivent*** conserver une copie de la dernière valeur ***MessageID*** reçue de son port partenaire. Les dispositifs qui prennent en charge les ports multiples, comme les hubs, ***Devoir/Doit/Doivent*** conserver des copies du dernier ***MessageID*** pour chaque port. Un port qui communique en utilisant des paquets SOP* ***Devoir/Doit/Doivent*** conserver des copies du dernier ***MessageID*** pour chaque type de ***SOP**** qu'il utilise.

L'émetteur ***Devoir/Doit/Doivent*** utiliser le ***MessageID*** d'un message ***GoodCRC*** pour vérifier qu'un message en particulier a été reçu correctement. Le récepteur ***Devoir/Doit/Doivent*** utiliser le ***MessageID*** pour détecter les messages dupliqués.

6.7.1.1 Utilisation par l'émetteur

L'émetteur ***Devoir/Doit/Doivent*** utiliser le ***MessageID*** de la manière suivante:

- après réception d'un signal ***Hard Reset*** ou d'un message ***Soft_Reset***, l'émetteur ***Devoir/Doit/Doivent*** mettre son ***MessageIDCounter*** à zéro et réinitialiser son mécanisme de relance;
- si aucun message ***GoodCRC*** avec un ***MessageID*** correspondant au ***MessageIDCounter*** n'est reçu avant l'expiration de ***CRCReceiveTimer***, le même paquet ***Devoir/Doit/Doivent*** être relancé jusqu'à ***nRetryCount*** fois, en utilisant le même ***MessageID***;
- si un message ***GoodCRC*** est reçu avec un ***MessageID*** correspondant au ***MessageIDCounter*** actuel avant l'expiration de ***CRCReceiveTimer***, l'émetteur ***Devoir/Doit/Doivent*** réinitialiser son mécanisme de relance et incrémenter son ***MessageIDCounter***;
- si le message est abandonné par le moteur de politique, l'émetteur ***Devoir/Doit/Doivent*** supprimer le message de sa mémoire tampon d'émission, réinitialiser son mécanisme de relance et incrémenter son ***MessageIDCounter***.

6.7.1.2 Utilisation par le récepteur

Le récepteur ***Devoir/Doit/Doivent*** utiliser le ***MessageID*** de la manière suivante:

- lorsque le premier paquet correct est reçu après une réinitialisation, le récepteur ***Devoir/Doit/Doivent*** enregistrer une copie de la valeur ***MessageID*** reçue;
- pour les messages suivants, si la valeur ***MessageID*** d'un message reçu est la même que la valeur mémorisée, le récepteur ***Devoir/Doit/Doivent*** renvoyer un message ***GoodCRC*** avec cette valeur ***MessageID*** et ignorer le message (il s'agit d'une relance concernant un message déjà reçu). Note: Cela ***Ne doit/doivent pas*** s'appliquer au message ***Soft_Reset*** dont la valeur ***MessageID*** est toujours zéro;
- si la valeur ***MessageID*** du message reçu est différente de la valeur mémorisée, le récepteur ***Devoir/Doit/Doivent*** renvoyer un message ***GoodCRC*** avec la nouvelle valeur ***MessageID***, enregistrer une copie de la nouvelle valeur ***MessageID*** et traiter le message.

6.7.2 Compteur de relances

RetryCounter est utilisé par un port en cas d'échec de transmission d'un message (expiration de ***CRCReceiveTimer***). Si la relance égale à ***nRetryCount*** échoue, alors la liaison ***Devoir/Doit/Doivent*** être réinitialisée au moyen du mécanisme de réinitialisation logicielle.

Les règles suivantes s'appliquent pour les relances en cas d'échec de transmission d'un message (voir aussi 6.11.2.1).

- Les fiches de câbles ***Ne doit/doivent pas*** relancer les messages.

- Les messages étendus de ***Data Size*** > ***MaxExtendedMsgLegacyLen*** qui ne sont pas fragmentés (fanion ***Chunked*** à 0) ***Ne doit/doivent pas*** être relancés.
- Les messages étendus de ***Data Size*** ≤ ***MaxExtendedMsgLegacyLen*** (fanion ***Chunked*** à 0 ou 1) ***Devoir/Doit/Doivent*** être relancés.
- Pour les messages étendus de ***Data Size*** > ***MaxExtendedMsgLegacyLen*** qui sont fragmentés (fanion ***Chunked*** à 1), les fragments individuels ***Devoir/Doit/Doivent*** être relancés.

Lorsque les messages ne sont pas relancés, ***RetryCounter*** n'est pas utilisé. Il est attendu des protocoles des couches supérieures qu'ils prennent en compte les échecs de délivrance des messages ou les échecs de réception d'un message ***GoodCRC***.

6.7.3 Compteur de réinitialisations matérielles

HardResetCounter est utilisé pour relancer la réinitialisation matérielle, en l'absence de réponse du dispositif distant (voir 6.6.6). Lorsque le signal Hard Reset a été relancé un nombre de fois égal à ***nHardResetCount***, alors l'hypothèse qui ***Devoir/Doit/Doivent*** être retenue est que le dispositif distant ne répond pas.

6.7.4 Compteur de capacités

CapsCounter est utilisé pour compter le nombre de messages ***Source_Capabilities*** qui ont été envoyés par une source à la mise sous tension ou après une réinitialisation matérielle. La mise en œuvre de ***CapsCounter*** est ***Facultatif(s/ve/ves),***, mais il ***Pouvoir/Peut/Peuvent*** être utilisé par toute source souhaitant préserver son énergie en cessant d'envoyer un message ***Source_Capabilities*** après un certain laps de temps.

Lorsque ***CapsCounter*** est mis en œuvre et que la source détecte qu'un destinataire est branché, après l'envoi de ***nCapsCount*** messages ***Source_Capabilities***, la source ***Devoir/Doit/Doivent*** alors décider que le destinataire ne répond pas, arrêter d'envoyer des messages ***Source_Capabilities*** et désactiver l'alimentation USB.

Un destinataire ***Devoir/Doit/Doivent*** utiliser ***SinkWaitCapTimer*** pour déclencher le renvoi de messages ***Source_Capabilities*** par une source apte à l'alimentation électrique par port USB qui a auparavant arrêté d'envoyer des messages ***Source_Capabilities***. Tout destinataire branché qui ne détecte aucun message ***Source_Capabilities*** ***Devoir/Doit/Doivent*** émettre un signal ***Hard Reset*** à l'expiration de ***SinkWaitCapTimer*** pour réinitialiser la source. La réinitialisation de la source ***Devoir/Doit/Doivent*** aussi réinitialiser ***CapsCounter*** et relancer l'envoi de messages ***Source_Capabilities***.

6.7.5 Compteur de découvertes d'identités

Lorsqu'il envoie des messages ***Discover Identity*** vers une fiche de câble, un port ***Devoir/Doit/Doivent*** tenir le compte des messages envoyés (***DiscoverIdentityCounter***). Le nombre de messages ***Discover Identity*** envoyés par le port qui reçoit un message de réponse ***GoodCRC*** ne ***Devoir/Doit/Doivent*** pas excéder ***nDiscoverIdentityCount***. Une permutation VCONN ***Devoir/Doit/Doivent*** remettre le ***DiscoverIdentityCounter*** à zéro.

6.7.6 VDMBusyCounter

Lorsqu'ils envoient des réponses Responder Busy à un message ***Vendor_Defined*** structuré, les UFP ou les fiches de câbles ***Devoir/Doit/Doivent*** tenir le compte des messages envoyés (***VDMBusyCounter***). Le nombre de réponses Responder Busy envoyées ne ***Devoir/Doit/Doivent*** pas excéder ***nBusyCount***. Le ***VDMBusyCounter*** ***Devoir/Doit/Doivent*** être réinitialisé après l'envoi d'une réponse autre que Busy. ***Il convient d'/de/qu'/que*** utiliser un ***nBusyCount*** de 1 pour les produits souhaitant respecter les exigences ***[USB Type-C 2.0]*** en ce qui concerne l'entrée dans un mode.

6.7.7 Valeurs de compteurs et compteurs

Le Tableau 6-64 contient la liste des compteurs utilisés dans la présente section et le Tableau 6-63 présente les paramètres correspondants.

Tableau 6-63 Paramètres des compteurs

Paramètre	Valeur	Référence
nBusyCount	5	Section 6.7.6
nCapsCount	50	Section 6.7.4
nDiscoverIdentityCount	20	Section 6.7.5
nHardResetCount	2	Section 6.7.3
nMessageIDCount	7	Section 6.7.1
nRetryCount	2	Section 6.7.2

Tableau 6-64 Compteurs

Compteur	Max	Référence
CapsCounter	*nCapsCount*	Section 6.7.4
DiscoverIdentityCounter	*nDiscoverIdentityCount*	Section 6.7.5
HardResetCounter	*nHardResetCount*	Section 6.7.3
MessageIDCounter	*nMessageIDCount*	Section 6.7.1
RetryCounter	*nRetryCount*	Section 6.7.2
VDMBusyCounter	*nBusyCount*	Section 6.7.6

6.8 Réinitialisation

Les réinitialisations sont une réponse nécessaire aux erreurs de protocole ou aux autres erreurs. L'alimentation électrique par port USB définit quatre types différents de réinitialisation:

- la réinitialisation logicielle, qui réinitialise le protocole;
- la réinitialisation des données, qui réinitialise les communications USB;
- la réinitialisation matérielle, qui réinitialise à la fois les blocs d'alimentation et le protocole;
- la réinitialisation de câble, qui réinitialise le câble.

6.8.1 Réinitialisation logicielle et erreur de protocole

Un message *Soft_Reset* est utilisé pour provoquer une réinitialisation logicielle de la communication du protocole lorsqu'elle s'est coupée d'une manière ou d'une autre. Elle ***Ne doit/doivent pas*** avoir d'impact sur le fonctionnement de l'alimentation, mais est utilisée pour corriger une erreur de protocole au cours d'une séquence atomique de messages (AMS). La réinitialisation logicielle ***Pouvoir/Peut/Peuvent*** être déclenchée par l'un des deux ports partenaires en réponse à l'erreur de protocole.

Les erreurs de protocole correspondent à tout message inattendu pendant une AMS. Si le premier message d'une AMS a été transmis à la couche protocole par le moteur de politique, mais qu'il n'a pas encore été reçu (message *GoodCRC* non reçu) lorsque l'erreur de protocole se produit, le moteur de politique ***Ne doit/doivent pas*** émettre de signal Soft Reset, mais il ***Devoir/Doit/Doivent*** revenir à l'état *PE_SNK_Ready* ou *PE_SRC_Ready*, puis traiter le message entrant. Si l'erreur de protocole se produit pendant une AMS interruptible, alors le moteur de politique ***Ne doit/doivent pas*** émettre de signal Soft Reset, mais il ***Devoir/Doit/Doivent*** revenir à l'état *PE_SNK_Ready* ou *PE_SRC_Ready*, puis traiter le message entrant. Si le message entrant est un message inattendu reçu pendant l'état *PE_SNK_Ready* ou *PE_SRC_Ready*, le moteur de politique ***Devoir/Doit/Doivent*** émettre un signal Soft Reset. Si l'erreur de protocole se produit pendant une AMS non interruptible, elle ***Devoir/Doit/Doivent*** donner lieu à une réinitialisation logicielle pour resynchroniser les diagrammes d'états du moteur de politique (voir 8.3.3.4), sauf lorsque la tension est en transition et que l'erreur de protocole ***Devoir/Doit/Doivent*** donner lieu à une réinitialisation matérielle (voir 6.6.11.4 et 8.3.3.2). La description des AMS interruptibles et non interruptibles peut être consultée en 8.3.2.1.3.

Les messages non reconnus ou non pris en charge reçus pendant les états *PE_SNK_Ready* ou *PE_SRC_Ready* ***Ne doit/doivent pas*** entraîner la génération d'un message *Soft_Reset*, mais un message *Not_Supported* ***Devoir/Doit/Doivent*** être généré.

Un message ***Soft_Reset*** ***Devoir/Doit/Doivent*** être envoyé quelle que soit la valeur de Rp, ***SinkTxOk*** ou ***SinkTxNG***, s'il s'agit de la réponse appropriée dans cet état. Note: Cela signifie qu'un message ***Soft_Reset*** peut être envoyé au cours d'une AMS, quelle que soit la valeur de Rp, ***SinkTxOk*** ou ***SinkTxNG***, en réponse à une erreur de protocole.

Le Tableau 6-65 et le Tableau 6-66 récapitulent les réponses qui ***Devoir/Doit/Doivent*** être envoyées pour un message entrant, y compris les VDM.

Tableau 6-65 Réponse à un message entrant (hors VDM)

Rôle d'alimentation du récepteur	Etat du récepteur	Message entrant			
		Reconnu			Non reconnu
		Pris en charge		Non pris en charge	
		Attendu	Inattendu		
Source	***PE_SRC_Ready***	Traiter le Message	Message ***Soft_Reset***[2]	Message ***Not_Supported***[3]	Message ***Not_Supported***[3] (sauf pour les VDM) Voir 6.4.4.1 pour les UVDM, 6.12.4 pour les SVDM
	Pendant les AMS interruptibles (avec contrat explicite)	Traiter le Message	Revenir à l'état ***PE_SRC_Ready*** et traiter le message		
	Pendant les AMS interruptibles (sans contrat explicite)	Traiter le Message	Message ***Soft_Reset***[2]		
	Pendant les AMS non interruptibles (sans transition de l'alimentation[1])	Traiter le Message	Message ***Soft_Reset***[2]		
	Pendant les AMS non interruptibles (avec transition de l'alimentation[1])	Traiter le Message	Signalisation ***Hard Reset***		
Destinataire	***PE_SNK_Ready***	Traiter le Message	Message ***Soft_Reset***[2]	Message ***Not_Supported***[3]	Message ***Not_Supported***[3] (sauf pour les VDM) Voir 6.4.4.1 pour les UVDM, 6.12.4 pour les SVDM
	Pendant les AMS interruptibles (avec contrat explicite)	Traiter le Message	Revenir à l'état ***PE_SNK_Ready*** et traiter le message		
	Pendant les AMS interruptibles (sans contrat explicite)	Traiter le Message	Message ***Soft_Reset***[2]		
	Pendant les AMS non interruptibles (sans transition de l'alimentation)	Traiter le Message	Message ***Soft_Reset***[2]		
	Pendant les AMS non interruptibles (avec transition de l'alimentation)	Traiter le Message	Signalisation ***Hard Reset***		

Rôle d'alimenta-tion du récepteur	Etat du récepteur	Message entrant			
		Reconnu			Non reconnu
		Pris en charge		Non pris en charge	
		Attendu	Inattendu		
1. "transition d'alimentation" signifie que le moteur de politique est à l'état *PE_SRC_Transition_Supply*, à l'état *PE_SNK_Transition_Sink* ou à l'état *PE_FRS_SNK_SRC_Send_Swap*. 2. Le message *Soft_Reset* ***Devoir/Doit/Doivent*** être envoyé à l'aide du SOP* du message entrant. 3. Le message *Not_Supported* ***Devoir/Doit/Doivent*** être envoyé à l'aide du SOP* du message entrant.					

Tableau 6-66 Réponse à un VDM entrant

Rôle du récepteur	UVDM pris en charge	UVDM non pris en charge	UVDM non reconnu	SVDM pris en charge	SVDM non pris en charge	SVDM non reconnu
DFP ou UFP	Défini par le fournisseur	*Not_Supported* Message	*Not_Supported* Message	Voir 6.12.4	*Not_Supported* Message	Commande NAK
Fiche de câble	Défini par le fournisseur	Message ***Ignoré(s/ée/ées)***	Message ***Ignoré(s/ée/ées)***	Voir 6.12.4	Message ***Ignoré(s/ée/ées)***	Commande NAK

L'échec de réception d'un message *GoodCRC* en réponse à tout message dans un délai *tReceive* (après *nRetryCount* nouvelles tentatives), lorsqu'une paire de ports est connectée, indique un défaut des communications donnant lieu à une réinitialisation logicielle (voir 6.6.9.1).

Une réinitialisation logicielle ***Devoir/Doit/Doivent*** affecter les couches d'alimentation électrique par port USB comme suit:

- Couche physique: Réinitialisation non exigée, car la couche physique redémarre à chaque émission/réception de paquet.
- Couche protocole: Réinitialisation de *MessageIDCounter*, de *RetryCounter* et des diagrammes d'états.
- Moteur de politique: Réinitialisation d'un comportement lié à l'état au moyen d'une négociation de contrat explicite.
- Alimentation: ***Ne doit/doivent pas*** varier.

Une réinitialisation logicielle est effectuée au moyen d'une séquence de messages de protocole (voir Tableau 8-8). Les numéros des messages ***Devoir/Doit/Doivent*** être définis sur 0 avant d'envoyer le message *Soft_Reset*/*Accept*, car le problème peut être lié aux compteurs. L'expéditeur d'un message *Soft_Reset* ***Devoir/Doit/Doivent*** réinitialiser son *MessageIDCounter* et son *RetryCounter*, le récepteur du message ***Devoir/Doit/Doivent*** réinitialiser son *MessageIDCounter* et son *RetryCounter* avant d'envoyer le message de réponse *Accept*. Tout échec dans le processus de réinitialisation logicielle déclenche une réinitialisation matérielle lorsque des paquets SOP sont utilisés, ou une réinitialisation de câble pour tous les autres paquets SOP*, par exemple si aucun message *GoodCRC* n'est reçu pendant le processus de réinitialisation logicielle (voir 6.8.3 et 6.8.4).

6.8.2 Réinitialisation des données

Un message *Data_Reset* est utilisé par un port pour réinitialiser sa connexion de données USB et quitter tous les modes alternatifs, à la fois avec son port partenaire et dans la ou les fiches de câble.

Le processus de réinitialisation des données ***Pouvoir/Peut/Peuvent*** être initié par l'un ou l'autre des ports partenaires en envoyant un message *Data_Reset*.

Une réinitialisation des données affecte l'alimentation électrique par port USB de la manière suivante:

- ***Ne doit/doivent pas*** modifier les rôles d'alimentation des ports (source/destinataire) ou les rôles de données des ports (DFP/UFP).
- ***Ne doit/doivent pas*** modifier le contrat explicite existant.

- ***Devoir/Doit/Doivent*** faire quitter tous les modes actifs.
- ***Devoir/Doit/Doivent*** réinitialiser le câble par un cycle d'alimentation de VCONN.
- Le DFP ***Devoir/Doit/Doivent*** devenir la source VCONN.

En cas d'échec du processus de réinitialisation des données, le port ***Devoir/Doit/Doivent*** passer à l'état ***ErrorRecovery*** défini dans ***[USB Type-C 2.0]***.

Voir 6.3.14 pour plus d'informations sur le fonctionnement de la réinitialisation des données.

6.8.3 Réinitialisation matérielle

Les réinitialisations matérielles sont signalées par un ensemble ordonné défini en 5.6.4. L'expéditeur et le récepteur ***Devoir/Doit/Doivent*** provoquer le retour de leurs alimentations respectives à leur état par défaut (voir 7.3.12 et 7.3.13 pour une description des transitions de tension). De plus, leurs couches protocole respectives ***Devoir/Doit/Doivent*** être réinitialisées, comme pour la réinitialisation logicielle. Cela permet aux dispositifs branchés d'être dans un état où ils peuvent rétablir la communication USB avec l'alimentation USB. La réinitialisation matérielle est relancée un nombre de fois allant jusqu'à ***nHardResetCount*** (voir également 6.6.6 et 6.7.3). Note: Bien que VBUS chute à ***vSafe0V*** pendant une réinitialisation matérielle, un destinataire ne voit pas cela comme une déconnexion, car il s'agit du comportement attendu.

Une réinitialisation matérielle ***Ne doit/doivent pas*** donner lieu à une variation dans la résistance Rp/Rd qui est affirmée.

Si une permutation des rôles de transmission de données a eu lieu, la réinitialisation matérielle ***Devoir/Doit/Doivent*** provoquer le retour du rôle de transmission de données du port à DFP pour un port dont la résistance Rp est affirmée et à UFP pour un port dont la résistance Rd est affirmée.

Lorsque VCONN est prise en charge (voir ***[USB Type-C 2.0]***), la réinitialisation matérielle ***Devoir/Doit/Doivent*** entraîner le fait que le port dont la résistance Rp est affirmée alimente VCONN et que le port dont la résistance Rd est affirmée coupe VCONN.

En effet, la réinitialisation matérielle fait revenir les ports à leur état par défaut selon leurs résistances de ligne CC. Le fait de couper et de rétablir VCONN dans les fiches de câbles permet également de s'assurer qu'elles rétablissent leur configuration en SOP' ou SOP'' selon l'emplacement de VCONN (voir ***[USB Type-C 2.0]***).

Si la réinitialisation matérielle ne suffit pas pour effacer la condition d'erreur, ***Il convient d'/de/qu'/que*** le port utilise des mécanismes de rétablissement sur erreur définis dans ***[USB Type-C 2.0]***.

Un destinataire ***Devoir/Doit/Doivent*** être capable d'envoyer un signal ***Hard Reset*** quelle que soit la valeur de Rp (voir 5.7).

6.8.3.1 Fiches de câbles et réinitialisation matérielle

Les fiches de câbles ***Ne doit/doivent pas*** générer de signal ***Hard Reset***, mais elles ***Devoir/Doit/Doivent*** surveiller la présence d'un signal ***Hard Reset*** entre les ports partenaires et elles ***Devoir/Doit/Doivent*** procéder à une réinitialisation si un tel signal est détecté (voir 8.3.3.24.2.2). Les fiches de câbles ***Devoir/Doit/Doivent*** exécuter l'équivalent d'un cycle d'alimentation en revenant à leur état de démarrage initial. Cela permet aux produits branchés d'être dans un état où ils peuvent rétablir la communication USB avec l'alimentation USB.

6.8.3.2 Fonctionnement modal et réinitialisation matérielle

Un signal Hard Reset ***Devoir/Doit/Doivent*** faire sortir les deux ports partenaires et toutes les fiches de câbles de tous les modes actifs (voir 6.4.4.3.4).

6.8.4 Réinitialisation de câble

Les réinitialisations de câbles sont signalées par un ensemble ordonné défini en 5.6.5. L'expéditeur et le récepteur du signal ***Cable Reset*** ***Devoir/Doit/Doivent*** réinitialiser leur couche protocole respective. Les fiches de câbles ***Devoir/Doit/Doivent*** exécuter l'équivalent d'un cycle d'alimentation en revenant à leur

état de démarrage initial. Cela permet aux produits branchés d'être dans un état où ils peuvent rétablir la communication USB avec l'alimentation USB.

Le DFP doit alimenter VCONN avant une réinitialisation de câble. Si VCONN a été coupée, le DFP ***Devoir/Doit/Doivent*** activer VCONN avant de générer le signal ***Cable Reset***. Si une permutation de VCONN a eu lieu et que l'UFP alimente actuellement VCONN, le DFP ***Devoir/Doit/Doivent*** effectuer une permutation de VCONN de manière à alimenter VCONN avant de générer un signal ***Cable Reset***.

Seul un DFP ***Devoir/Doit/Doivent*** générer le signal ***Cable Reset***. Un DFP ne ***Devoir/Doit/Doivent*** générer le signal ***Cable Reset*** que dans le cadre d'un contrat explicite.

Un signal Cable Reset ***Devoir/Doit/Doivent*** faire quitter tous les modes actifs des fiches de câbles (voir 6.4.4.3.4).

6.9 Anticollision

Pour éviter les collisions entre les messages du fait de l'envoi asynchrone de messages par le destinataire, la source définit Rp sur ***SinkTxOk*** pour indiquer au destinataire qu'il est autorisé à initier une AMS. Lorsque la source souhaite initier une AMS, elle définit Rp sur ***SinkTxNG***. Lorsque le destinataire détecte que Rp est définie sur ***SinkTxOk***, il ***Pouvoir/Peut/Peuvent*** initier une AMS. Lorsque le destinataire détecte que Rp est définie sur ***SinkTxNG***, il ***Ne doit/doivent pas*** initier d'AMS et ne ***Devoir/Doit/Doivent*** envoyer que des messages faisant partie d'une AMS initiée par la source. Noter que cette restriction s'applique aux AMS de SOP*, c'est-à-dire pour les communications port à port et pour les communications port à fiche de câble.

Note: Un destinataire peut toujours envoyer un signal ***Hard Reset*** à tout moment.

6.10 Rejet de message

A la réception d'un message en SOP, la couche protocole ***Devoir/Doit/Doivent Rejeter*** tous les messages SOP* en attente. Un message reçu en SOP'/SOP'' ***Ne doit/doivent pas Rejeté(s/ée/ées)*** les messages SOP* en attente.

Par hypothèse, les messages qui utilisent SOP'/SOP'' constituent une simple AMS de demande/réponse où la fiche de câble fournit la réponse, il n'y a donc pas de raison qu'un message SOP* en attente soit ***Rejeté(s/ée/ées)***. Il ne peut y avoir qu'une AMS entre les ports partenaires, et ceux-ci ont également la priorité sur les communications avec les fiches de câbles, aussi un message en attente sur SOP* est ***Rejeté(s/ée/ées)*** du fait de la réception d'un message reçu sur SOP.

Voir Tableau 6-67 pour une description des messages qui ***Devoir/Doit/Doivent***/***Ne doit/doivent pas*** être ***Rejeté(s/ée/ées)***.

Tableau 6-67 Rejet de message

Transmission de message en attente	Message reçu	Rejeter la transmission en attente?
SOP	SOP	Oui
SOP	SOP'/SOP''	Non
SOP'	SOP	Oui
SOP'	SOP'	Non
SOP'	SOP''	Non
SOP''	SOP	Oui
SOP''	SOP'	Non
SOP''	SOP''	Non

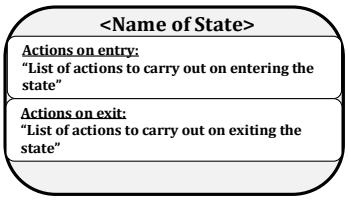
6.11 Comportement d'état

6.11.1 Présentation des diagrammes d'état utilisés au Chapitre 6

Les diagrammes d'état définis en 6.11 sont ***Normatif(s/ve/ves)*** et ***Devoir/Doit/Doivent*** définir le fonctionnement de la couche de protocole de l'alimentation électrique. Noter que ces diagrammes d'état n'ont pas pour objet de remplacer une conception robuste et bien rédigée.

La Figure 6-48 représente les grandes lignes des états définis dans les sections suivantes. Le nom de l'état se trouve en haut. Il est suivi par "Actions à l'entrée", une liste des actions exécutées à l'entrée dans cet état, et pour certains états par "Actions à la sortie", une liste des actions exécutées à la sortie de cet état.

Figure 6-48 Présentation des états



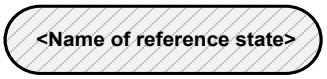
Anglais	Français
Name of State	Nom de l'état
Actions on entry	Actions à l'entrée
List of actions to carry out on entering the state	Liste des actions à exécuter à l'entrée dans l'état
Actions on exit	Actions à la sortie
List of actions to carry out on exiting the state	Liste des actions à exécuter à la sortie de l'état

Les transitions d'un état à l'autre sont indiquées par des flèches avec les conditions mentionnées sur la flèche. En présence de plusieurs conditions, celles-ci sont connectées au moyen d'un OU logique "|" ou d'un ET logique "&". L'inverse d'une condition est indiqué par "NOT" (NON) devant la condition.

Dans certains cas, des transitions peuvent avoir lieu à partir de tout état vers un état particulier. Elles sont indiquées par une flèche qui n'est pas connectée à un état d'un côté, mais dont l'autre extrémité (la pointe) est connectée à l'état final.

Dans certains diagrammes d'état, il est nécessaire d'entrer ou de sortir des états des autres diagrammes. La Figure 6-49 indique comment ces références sont faites. La référence est indiquée dans un cadre hachuré. Le cadre contient le nom de l'état de référence.

Figure 6-49 Références aux états



Anglais	Français
Name of reference state	Nom de l'état de référence

Les temporisateurs sont inclus dans un grand nombre des états. Les temporisateurs sont initialisés (définis sur leur condition de démarrage) et exécutés (le temporisateur compte) dans l'état particulier où ils sont référencés. Dès que l'état est quitté, le temporisateur n'est plus actif. Les expirations des temporisateurs sont indiquées comme des conditions dans les transitions d'état.

Les conditions mentionnées dans les transitions d'état viendront de l'une des trois sources suivantes:

- messages reçus de la couche PHY;
- événements déclenchés dans la couche protocole, par exemple l'expiration des temporisateurs;

- messages et indications qui s'y réfèrent, transférés au moteur de politique depuis la couche protocole (message envoyé, message reçu, etc.);

6.11.2 Fonctionnement d'état

Le paragraphe suivant décrit le fonctionnement d'état de la couche protocole lorsqu'elle envoie et reçoit des paquets SOP*.

Pour chaque communication SOP* envoyée et reçue, les instances de diagrammes d'états pour la transmission par la couche protocole, la réception par la couche protocole et la réinitialisation matérielle ***Devoir/Doit/Doivent*** être séparées, avec leurs propres instances pour les compteurs et les temporisateurs. Lorsque la fragmentation est prise en charge, les diagrammes d'états de Rx fragmenté, Tx fragmenté et du routeur de message fragmenté ***Devoir/Doit/Doivent*** être séparés.

La réinitialisation logicielle ***Devoir/Doit/Doivent*** s'appliquer seulement aux instances de diagrammes d'états qu'elle vise, selon le type de paquet SOP* utilisé pour envoyer le message ***Soft_Reset***. Le diagramme d'état de la réinitialisation matérielle (y compris la réinitialisation de câble) ***Devoir/Doit/Doivent*** s'appliquer simultanément à toutes les instances de diagrammes d'états de la couche protocole actives dans le DFP, l'UFP et la fiche de câble (le cas échéant).

6.11.2.1 Fragmentation de la couche protocole

6.11.2.1.1 Architecture d'un dispositif incluant une couche de fragmentation

Le composant de fragmentation se trouve dans la couche protocole entre le moteur de politique et le Tx/Rx de protocole. La Figure 6-50 représente la relation entre les composants.

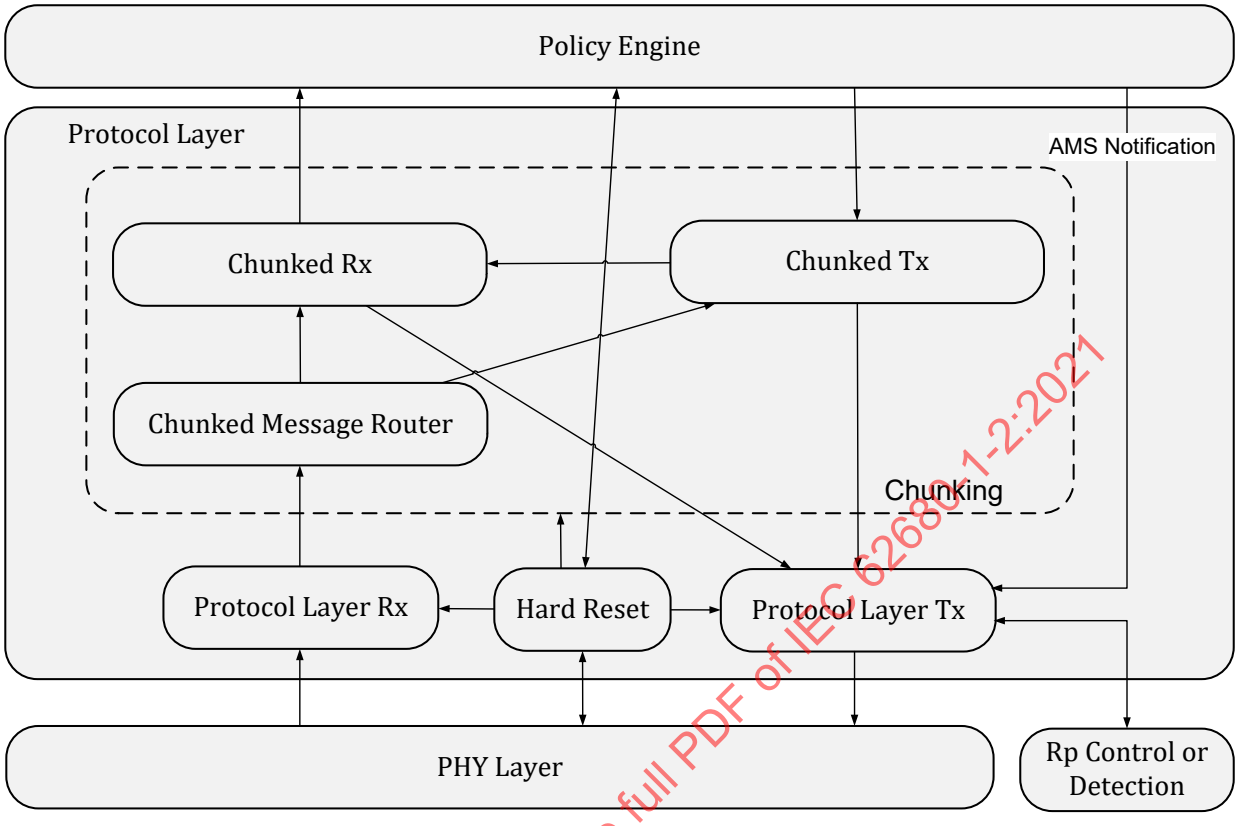
La couche de fragmentation comprend trois diagrammes d'états liés:

- Rx fragmenté;
- Tx fragmenté;
- routeur de message fragmenté.

Noter que la conséquence de cette architecture est que le moteur de politique s'occupe entièrement des messages non fragmentés. Il ne reçoit pas le message (et n'y répond probablement pas), jusqu'à ce que tous les fragments liés aient été assemblés.

Si un dispositif apte à l'alimentation USB ou un repère de câble ne fait l'objet d'aucune exigence concernant la gestion des messages qui exigent plus d'un fragment de tout message étendu, il ***Pouvoir/Peut/Peuvent*** omettre la couche de fragmentation. Dans ce cas, il ***Devoir/Doit/Doivent*** mettre en œuvre ***ChunkingNotSupportedTimer*** pour assurer un fonctionnement compatible avec des partenaires qui prennent en charge le fractionnement (voir 6.6.18.1 et 8.3.3.6).

Figure 6-50 Architecture de fragmentation représentant le message et le flux de contrôle



Anglais	Français
Policy Engine	Moteur de politique
Protocol Layer	Couche protocole
AMS Notification	Notification d'AMS
Chunked	Fragmenté
Chunked Message Router	Routeur de message fragmenté
Hard Reset	Réinitialisation matérielle
Chunking	Fragmentation
PHY Layer	Couche PHY
Rp Control or Detection	Contrôle ou détection de Rp

6.11.2.1.1.1 Mécanisme d'abandon facultatif

Les longs messages fragmentés s'accompagnent d'un éventuel problème susceptible d'empêcher les messages urgents d'être transmis en temps utile. Un mécanisme d'abandon facultatif est fourni pour résoudre ce problème.

Le fanion d'abandon mentionné dans les diagrammes ci-dessous ***Pouvoir/Peut/Peuvent*** être défini et examiné par le moteur de politique. Les moyens spécifiques sont laissés au choix de l'implémenteur.

6.11.2.1.1.2 Abandon d'envoi d'un long message fragmenté

Un long message fragmenté envoyé ***Pouvoir/Peut/Peuvent*** être abandonné en définissant le fanion d'abandon ***Facultatif(s/ve/ves),***. Le message ***Devoir/Doit/Doivent*** être jugé comme abandonné lorsque le fanion d'abandon est effacé à nouveau par le diagramme d'état de Tx fragmenté.

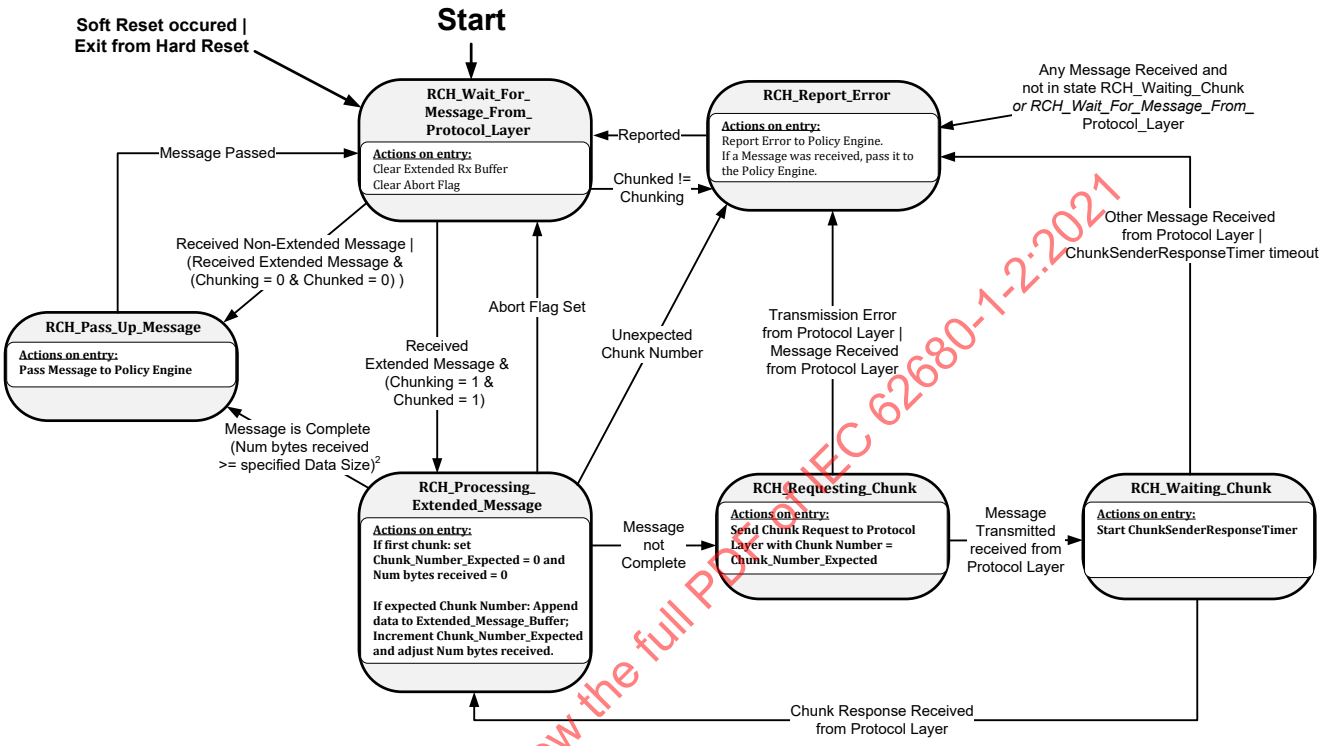
6.11.2.1.1.3 Abandon de réception d'un long message fragmenté

Si le mécanisme d'abandon facultatif a été mis en œuvre, tout message envoyé pendant qu'un message fragmenté est en cours de réception donne lieu à un rapport d'erreur reçu par le moteur de politique, pour indiquer que la demande de message a été ***Rejeté(s/ée/ées)***. Si le message était urgent, le moteur de politique peut activer le fanion d'abandon, ce qui donne lieu à l'abandon du message fragmenté entrant. Le fait que le fanion d'abandon soit effacé par le diagramme d'état de Rx fragmenté indique que le message urgent peut désormais être envoyé.

6.11.2.1.2 Diagramme d'état de Rx fragmenté

La Figure 6-51 représente le comportement d'état pour le diagramme d'état de Rx fragmenté. Celui-ci reconnaît lorsque des messages reçus fragmentés sont impliqués et gère les demandes de fragments, le cas échéant. Il vérifie également la validité de tous les messages liés à la fragmentation.

Figure 6-51 Diagramme d'état de Rx fragmenté



Anglais	Français
Soft Reset occurred	Une réinitialisation logicielle a eu lieu
Exit from Hard Reset	Sortir de la réinitialisation matérielle
Start	Début
Actions on entry	Actions à l'entrée
Clear Extended Rx Buffer	Effacer la mémoire tampon de Rx étendu
Clear Abort Flag	Effacer le fanion d'abandon
Message passed	Message transmis
Received non-extended Message	Message non étendu reçu
Chunking	Fragmentation
Pass Message to Policy Engine	Transmettre le message au moteur de politique
Message is Complete	Le message est complet
Num bytes received >= specified Data Size	Nombre d'octets reçus >= taille de données spécifiée
Chunked	Fragmenté
Received Extended Message	Message étendu reçu
Abort Flag Set	Fanion d'abandon défini
Unexpected Chunk Number	Nombre de fragments inattendu
Reported	Rapport
Message not Complete	Message non complet

If first chunk: set	Si premier fragment: défini
and	et
If expected Chunk Number	Si nombre de fragments attendu
Append data to	Ajouter des données à
Increment Chunk_Number_Expected and adjust Num bytes received	Incrémenter Chunk_Number_Expected et ajuster le nombre d'octets reçus
Send Chunk Request to Protocol Layer with Chunk Number	Envoyer la demande de fragment à la couche protocole avec le nombre de fragments
Message Transmitted received from Protocol Layer	Message transmis reçu de la couche protocole
Transmission Error from Protocol Layer	Erreur de transmission de la couche protocole
Message received from Protocol Layer	Message reçu de la couche protocole
Any Message Received and not in state RCH_Waiting_Chunk or *RCH_Wait_for_Message_From_Protocol_Layer*	Tout message reçu qui n'est pas à l'état RCH_Waiting_Chunk ou *RCH_Wait_for_Message_From_Protocol_Layer*
Other Message Received from Protocol Layer	Autre message reçu de la couche protocole
ChunkSenderResponseTimer timeout	Temporisation ChunkSenderResponseTimer
Start ChunkSenderResponseTimer	Lancer ChunkSenderResponseTimer
Chunk Response Received from Protocol Layer	Réponse de fragment reçu de la couche protocole

[1] La fragmentation est un état interne qui est défini sur 1 si le bit "Unchunked Extended Messages Supported" des capacités de source ou de la demande est à 0. La valeur par défaut est 1 et elle est définie après le premier échange de capacités de source et de demande. Il est également défini sur 1 pour la communication ***SOP'*** ou ***SOP''***.

[2] Les octets supplémentaires reçus par rapport à la valeur de ***Data Size*** spécifiée sont le résultat du remplissage du dernier fragment.

6.11.2.1.2.1 Etat RCH_Wait_For_Message_From_Protocol_Layer

Le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent*** entrer à l'état ***RCH_Wait_For_Message_From_Protocol_Layer***:

- au démarrage;
- à la suite d'une réinitialisation logicielle;
- à la sortie d'une réinitialisation matérielle.

En entrant dans l'état ***RCH_Wait_For_Message_From_Protocol_Layer***, le diagramme d'état de Rx fragmenté efface la mémoire tampon Rx étendue et efface le fanion d'abandon facultatif.

Dans l'état ***RCH_Wait_For_Message_From_Protocol_Layer***, le diagramme d'état de Rx fragmenté attend que le routeur de message fragmenté transfère un message reçu.

Le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent*** entrer à l'état ***RCH_Pass_Up_Message*** lorsque:

- un message non étendu est transféré par le routeur de message fragmenté;
- un message étendu est transféré par le routeur de message fragmenté, et que le moteur de politique a déterminé qu'il n'y a pas de fragmentation, et que le bit ***Chunked*** du message est défini sur 0b.

Le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent*** entrer à l'état ***RCH_Processing_Extended_Message*** lorsque:

- un message étendu est transféré par le routeur de message fragmenté, et que le moteur de politique a déterminé qu'il y a une fragmentation, et que le bit ***Chunked*** du Message est défini sur 1b.

6.11.2.1.2.2 Etat RCH_Pass_Up_Message

En entrant dans l'état ***RCH_Pass_Up_Message***, le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent*** transférer le message reçu au moteur de politique.

Le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent*** entrer à l'état ***RCH_Wait_For_Message_From_Protocol_Layer*** lorsque:

- le message a été transféré.

6.11.2.1.2.3 Etat RCH_Processing_Extended_Message

En entrant dans l'état ***RCH_Processing_Extended_Message***, le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent***:

- s'il s'agit du premier fragment:
  - définir Chunk_Number_Expected = 0;
  - définir le nombre d'octets reçus sur 0;
- si le fragment contient le nombre de fragments attendu:
  - adjoindre ses données à Extended_Message_Buffer;
  - incrémenter Chunk_Number_Expected;
  - ajuster le nombre d'octets reçus.

Le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent*** entrer à l'état ***RCH_Pass_Up_Message*** lorsque:

- le message est complet (c'est-à-dire que le nombre d'octets reçus >= ***Data Size*** spécifiée. Noter que l'inégalité permet l'utilisation d'octets de remplissage dans le dernier fragment, qui ne font pas réellement partie du message étendu).

Le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent*** entrer à l'état ***RCH_Requesting_Chunk*** lorsque:

- le message n'est pas encore complet.

Le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent*** entrer à l'état ***RCH_Report_Error*** lorsque:

- un nombre de fragments inattendu est reçu.

Le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent*** entrer à l'état ***RCH_Wait_For_Message_From_Protocol_Layer*** lorsque:

- le fanion d'abandon facultatif est défini.

6.11.2.1.2.4 Etat RCH_Requesting_Chunk

En entrant dans l'état ***RCH_Requesting_Chunk***, le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent***:

- envoyer une demande de fragment à la couche protocole avec ***Chunk Number*** = Chunk_Number_Expected.

Le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent*** entrer à l'état ***RCH_Waiting_Chunk*** lorsque:

- "Message Transmitted" est reçu de la couche protocole.

Le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent*** entrer à l'état ***RCH_Report_Error*** lorsque:

- une erreur de transmission est reçue de la couche de protocole; ou
- un message est reçu de la couche protocole.

6.11.2.1.2.5 Etat RCH_Waiting_Chunk

En entrant dans l'état ***RCH_Waiting_Chunk***, le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent***:

- Lancer le ***ChunkSenderResponseTimer.***

Le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent*** entrer à l'état ***RCH_Processing_Extended_Message*** lorsque:

- un fragment est reçu de la couche protocole.

Le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent*** entrer à l'état ***RCH_Report_Error*** lorsque:

- un message autre qu'un fragment est reçu de la couche protocole; ou
- ***ChunkSenderResponseTimer*** expire.

6.11.2.1.2.6 Etat RCH_Report_Error

Le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent*** entrer à l'état ***RCH_Report_Error***:

- Lorsque un message est reçu et que le diagramme d'état de Rx fragmenté n'est pas à l'état ***RCH_Waiting_Chunk*** ou ***RCH_Wait_For_Message_From_Protocol_Layer.***

En entrant dans l'état ***RCH_Report_Error***, le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent***:

- faire un rapport d'erreur au moteur de politique.
- Si l'état a été activé parce qu'un message a été reçu, ce message ***Devoir/Doit/Doivent*** être transféré au moteur de politique.

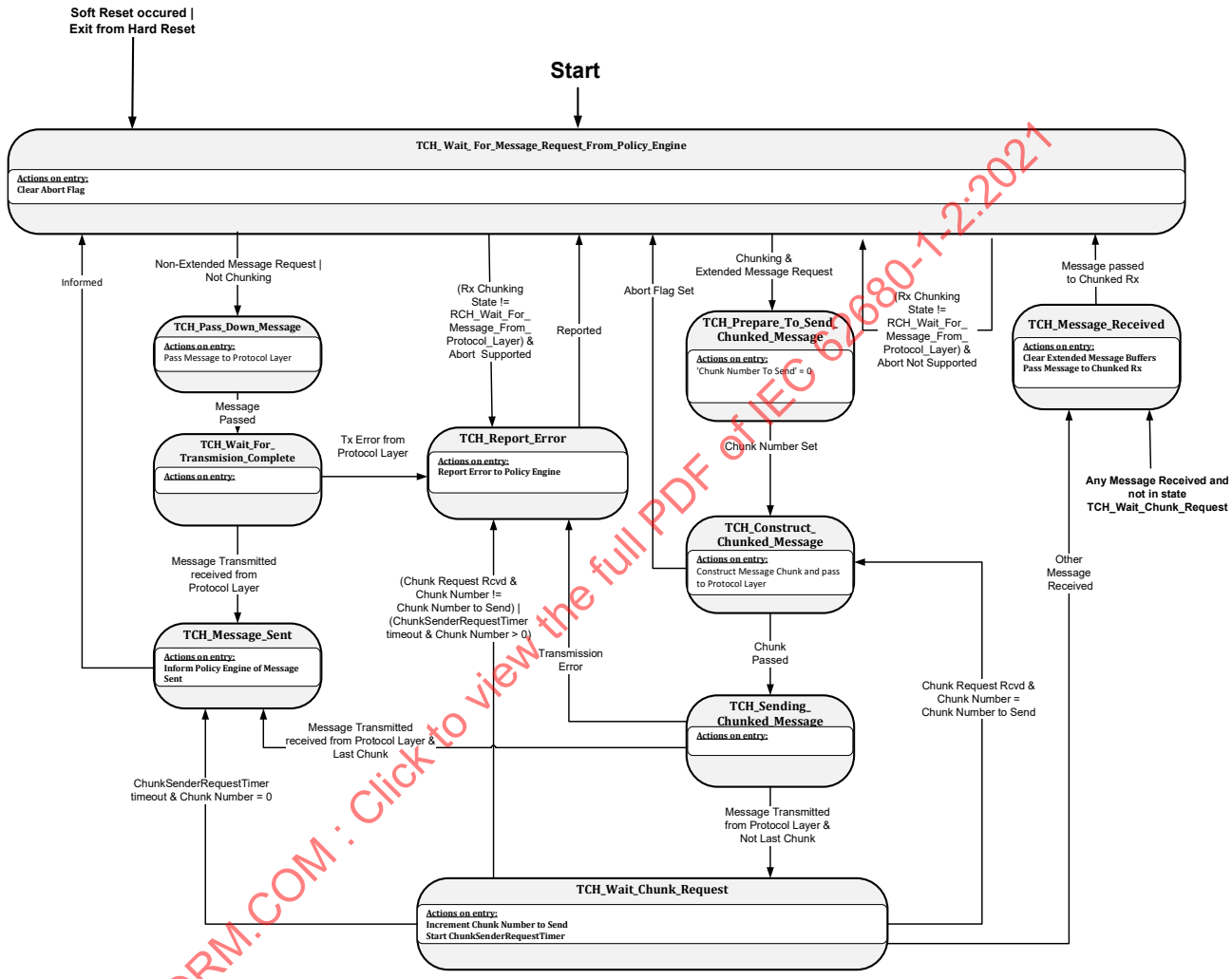
Le diagramme d'état de Rx fragmenté ***Devoir/Doit/Doivent*** entrer à l'état ***RCH_Wait_For_Message_From_Protocol_Layer*** lorsque:

- l'erreur a fait l'objet d'un rapport;
- tout message reçu a été transféré au moteur de politique.

6.11.2.1.3 Diagramme d'état de Tx fragmenté

La Figure 6-52 représente le comportement d'état pour le diagramme d'état de Tx fragmenté. Celui-ci reconnaît lorsque des messages transmis fragmentés sont impliqués et gère les envois de fragments et l'attente des demandes de fragments, le cas échéant. Il vérifie également la validité de tous les messages liés à la fragmentation.

Figure 6-52 Diagramme d'état de Tx fragmenté



Anglais	Français
Soft Reset occurred	Une réinitialisation logicielle a eu lieu
Exit from Hard Reset	Sortir de la réinitialisation matérielle
Start	Début
Actions on entry	Actions à l'entrée
Clear Abort Flag	Effacer le fanion d'abandon
Pass Message to Protocol Layer	Transmettre le message à la couche protocole
Message Passed	Message transmis
Inform Policy Engine of Message Sent	Informer le moteur de politique de l'envoi du message
Abort Flag Set	Fanion d'abandon défini
Tx Error from Protocol Layer	Erreur Tx de la couche protocole

Reported	Rapport
Rx Chunking State	Etat de fragmentation Rx
Abort not Supported	Abandon non pris en charge
Report Error to Policy Engine	Rapport d'erreur au moteur de politique
Chunk Request Rcvd & Chunk Number to Send	Demande de fragment reçue et nombre de fragments à envoyer
Transmission Error	Erreur de transmission
Increment Chunk Number to Send	Incrémenter le nombre de fragments à envoyer
Start ChunkSenderRequestTimer	Lancer ChunkSenderRequestTimer
Message Transmitted received from Protocol Layer	Message transmis reçu de la couche protocole
ChunkSenderRequestTimer timeout	Temporisation ChunkSenderRequestTimer
Chunk Request Rcvd & Chunk Number = Chunk Number to Send	Demande de fragment reçu et nombre de fragments = nombre de fragments à envoyer
Other Message Received	Autre message reçu
Last Chunk	Dernier fragment
Chunk Number Set	Nombre de fragments défini
Chunking & Extended Message Request	Fragmentation et demande de message étendu
Chunking State	Etat de fragmentation
Construct Message Chunk and pass to Protocol Layer	Construire un fragment de message et transmettre à la couche protocole
Chunk passed	Fragment transmis
Message Transmitted from Protocol Layer & Not Last Chunk	Message transmis de la couche protocole et pas le dernier fragment
Message Passed to Chunked Rx	Message transmis à Rx fragmenté
Clear Extended Message Buffers	Effacer les mémoires tampons des messages étendus
Pass Message to Chunked Rx	Transmettre le message à Rx fragmenté
Any Message Received and not in state TCH_Wait_Chunk_Request	Tout message reçu qui n'est pas à l'état TCH_Wait_Chunk_Request

6.11.2.1.3.1 Etat TCH_Wait_For_Message_Request_From_Policy_Engine

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** entrer à l'état ***TCH_Wait_For_Message_Request_From_Policy_Engine***:

- au démarrage;
- à la suite d'une réinitialisation logicielle;
- à la sortie d'une réinitialisation matérielle.

En entrant dans l'état ***TCH_Wait_For_Message_Request_From_Policy_Engine***, le diagramme d'état de Tx fragmenté efface le fanion d'abandon facultatif.

A l'état ***TCH_Wait_For_Message_Request_From_Policy_Engine***, le diagramme d'état de Tx fragmenté attend que le moteur de politique lui envoie une demande de message.

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** passer à l'état ***TCH_Pass_Down_Message*** lorsque:

- une demande de message non étendu est reçue du moteur de politique; ou
- une demande de message est reçue du moteur de politique et que le lien ne fragmente pas.

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** passer à l'état ***TCH_Prepare_To_Send_Chunked_Message*** lorsque:

- une demande de message étendu est reçue du moteur de politique et le lien fragmente.

Le diagramme d'état du Tx fragmenté ***Devoir/Doit/Doivent Rejeter*** la demande de message et rester à l'état ***TCH_Wait_For_Message_Request_From_Policy_Engine*** lorsque:

- l'état de Rx fragmenté est tout autre que ***TCH_Wait_For_Message_Request_From_Policy_Engine*** et que le fanion d'abandon facultatif n'a pas été défini.

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent Rejeter*** la demande de message et passer à l'état ***TCH_Report_Error*** lorsque:

- l'état de Rx fragmenté est tout autre que ***TCH_Wait_For_Message_Request_From_Policy_Engine*** et que le fanion d'abandon facultatif a été défini.

6.11.2.1.3.2 Etat TCH_Pass_Down_Message

En entrant dans l'état ***TCH_Pass_Down_Message***, le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** transférer le message au moteur de politique.

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** passer à l'état ***TCH_Wait_For_Transmision_Complete*** lorsque:

- le message a été transféré à la couche protocole.

6.11.2.1.3.3 Etat TCH_Wait_For_Transmision_Complete

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** passer à l'état ***TCH_Message_Sent*** lorsque:

- le message transmis a été reçu de la couche protocole.

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** passer à l'état ***TCH_Report_Error*** lorsque:

- l'erreur de transmission a été reçue de la couche protocole.

6.11.2.1.3.4 Etat TCH_Message_Sent

En entrant dans l'état ***TCH_Message_Sent***, le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent***:

- informer le moteur de politique que le message a été envoyé.

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** passer à l'état ***TCH_Wait_For_Message_Request_From_Policy_Engine*** lorsque:

- le moteur de politique a été informé.

6.11.2.1.3.5 Etat TCH_Prepare_To_Send_Chunked_Message

En entrant dans l'état ***TCH_Prepare_To_Send_Chunked_Message***, le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent***:

- définir "Chunk Number To Send" sur zéro.

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** passer à l'état ***TCH_Construct_Chunked_Message*** lorsque:

- "Chunk Number To Send" a été défini sur 0.

6.11.2.1.3.6 Etat TCH_Construct_Chunked_Message

En entrant dans l'état ***TCH_Construct_Chunked_Message***, le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent***:

- construire un fragment de message et le transférer à la couche protocole.

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** passer à l'état ***TCH_Sending_Chunked_Message*** lorsque:

- le fragment de message a été transféré à la couche protocole.

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** passer à l'état ***TCH_Wait_For_Message_Request_From_Policy_Engine*** lorsque:

- le fanion d'abandon facultatif est défini.

6.11.2.1.3.7 Etat TCH_Sending_Chunked_Message

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** passer à l'état ***TCH_Wait_Chunk_Request*** lorsque:

- "Message Transmitted" est reçu de la couche protocole et qu'il ne s'agissait pas du dernier fragment.

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** passer à l'état ***TCH_Message_Sent*** lorsque:

- "Message Transmitted" est reçu de la couche protocole et qu'il s'agissait du dernier fragment.

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** passer à l'état ***TCH_Report_Error*** lorsque:

- l'erreur de transmission a été reçue de la couche protocole.

6.11.2.1.3.8 Etat TCH_Wait_Chunk_Request

En entrant dans l'état ***TCH_Wait_Chunk_Request***, le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent***:

- incrémente le nombre de fragments à envoyer;
- lance le ***ChunkSenderRequestTimer.***

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** passer à l'état ***TCH_Report_Error*** lorsque:

- une demande de fragment a été reçue et le nombre de fragments n'est pas égal au nombre de fragments à envoyer; ou
- le ***ChunkSenderRequestTimer*** a expiré et le nombre de fragments est supérieur à zéro.

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** passer à l'état ***TCH_Message_Sent*** lorsque:

- le ***ChunkSenderRequestTimer*** a expiré et le nombre de fragments est égal à zéro.

Noter qu'il s'agit du mécanisme qui permet au port partenaire distant ou au repère de câble d'omettre la couche de fragmentation. Le moteur de politique reçoit un signal Message Sent si le port partenaire distant ou le repère de câble sont présents (Message ***GoodCRC*** reçu) mais qu'ils ne renvoient pas de demande de fragment. Après cela, le port partenaire distant envoie un message ***Not_Supported***, ou le repère de câble ***Ignorer*** le message fragmenté.

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** passer à l'état ***TCH_Message_Received*** lorsque:

- tout message autre qu'une demande de fragment est reçu.

6.11.2.1.3.9 Etat TCH_Message_Received

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** entrer à l'état ***TCH_Message_Received***:

- lorsque tout message est reçu et que le diagramme d'état de Tx fragmenté n'est pas à l'état ***TCH_Wait_Chunk_Request***.

En entrant dans l'état ***TCH_Message_Received***, le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent***:

- effacer les mémoires tampons des messages étendus;
- transférer le message reçu au moteur de Rx fragmenté.

Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** passer à l'état ***TCH_Wait_For_Message_Request_From_Policy_Engine*** lorsque:

- le message reçu a été transféré au moteur de Rx fragmenté.

6.11.2.1.3.10 Etat TCH_Report_Error

En entrant dans l'état ***TCH_Report_Error***, le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent***:

- faire un rapport d'erreur au moteur de politique.

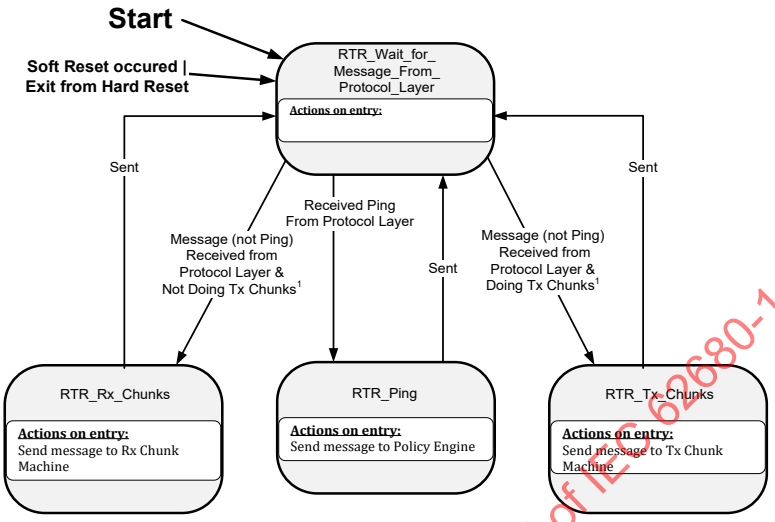
Le diagramme d'état de Tx fragmenté ***Devoir/Doit/Doivent*** passer à l'état ***TCH_Wait_For_Message_Request_From_Policy_Engine*** lorsque:

- l'erreur a fait l'objet d'un rapport.

6.11.2.1.4 Diagramme d'état du routeur de messages fragmentés

La Figure 6-53 représente le comportement d'état du routeur de messages fragmentés. Cela permet de déterminer le diagramme d'état vers lequel un message entrant est acheminé (Rx fragmenté, Tx fragmenté ou directement vers le moteur de politique).

Figure 6-53 Diagramme d'état du routeur de messages fragmentés



Anglais	Français
Start	Début
Soft Reset occurred	Une réinitialisation logicielle a eu lieu
Exit from Hard Reset	Sortie de la réinitialisation matérielle
Actions on entry	Actions à l'entrée
Sent	Envoyé
Message (not Ping) Received from Protocol Layer & Not Doing Tx Chunks[1]	Message (pas de Ping) reçu de la couche de protocole et ne générant pas de fragments Tx[1]
Received Ping From Protocol Layer	Réception d'un Ping de la couche de protocole
Message (not Ping) Received from Protocol Layer & Doing Tx Chunks[1]	Message (pas de Ping) reçu de la couche de protocole et générant des fragments Tx[1]
Send message to Rx Chunk Machine	Envoyer un message à la machine de fragments Rx
Send message to Policy Engine	Envoyer un message au moteur de règles
Send message to Tx Chunk Machine	Envoyer un message à la machine de fragments Tx

[1] Générer des fragments Tx signifie que le diagramme d'état de Tx fragmenté n'est pas à l'état ***TCH_Wait_For_Message_Request_From_Policy_Engine***.

[2] Les messages sont pris pour inclure la notification de succès de transmission ou autre des messages.

6.11.2.1.4.1 Etat RTR_Wait_for_Message_From_Protocol_Layer

Lorsqu'il est à l'état ***RTR_Wait_for_Message_From_Protocol_Layer***, le routeur de messages fragmentés attend que la couche de protocole lui envoie un message reçu.

Le routeur de messages fragmentés ***Devoir/Doit/Doivent*** passer à l'état ***RTR_Rx_Chunks*** lorsque:

- un message autre qu'un message ***Ping*** est reçu de la couche protocole, et la fragmentation combinée ne génère pas de fragments Tx.

Le routeur de messages fragmentés ***Devoir/Doit/Doivent*** passer à l'état ***RTR_Tx_Chunks*** lorsque:

- un message autre qu'un message ***Ping*** est reçu de la couche protocole, et la fragmentation combinée génère des fragments Tx.

Le routeur de messages fragmentés ***Devoir/Doit/Doivent*** passer à l'état ***RTR_Ping*** lorsque:

- un message ***Ping*** est reçu de la couche protocole.

6.11.2.1.4.2 Etat RTR_Rx_Chunks

En entrant dans l'état ***RTR_Rx_Chunks***, le routeur de messages fragmentés ***Devoir/Doit/Doivent***:

- envoyer le message au diagramme d'état de Rx fragmenté;
- passer à l'état ***RTR_Wait_for_Message_From_Protocol_Layer***.

6.11.2.1.4.3 Etat RTR_Ping

En entrant dans l'état ***RTR_Ping***, le routeur de messages fragmentés ***Devoir/Doit/Doivent***:

- envoyer le message au moteur de politique;
- passer à l'état ***RTR_Wait_for_Message_From_Protocol_Layer***.

6.11.2.1.4.4 Etat RTR_Tx_Chunks

En entrant dans l'état ***RTR_Tx_Chunks***, le routeur de messages fragmentés ***Devoir/Doit/Doivent***:

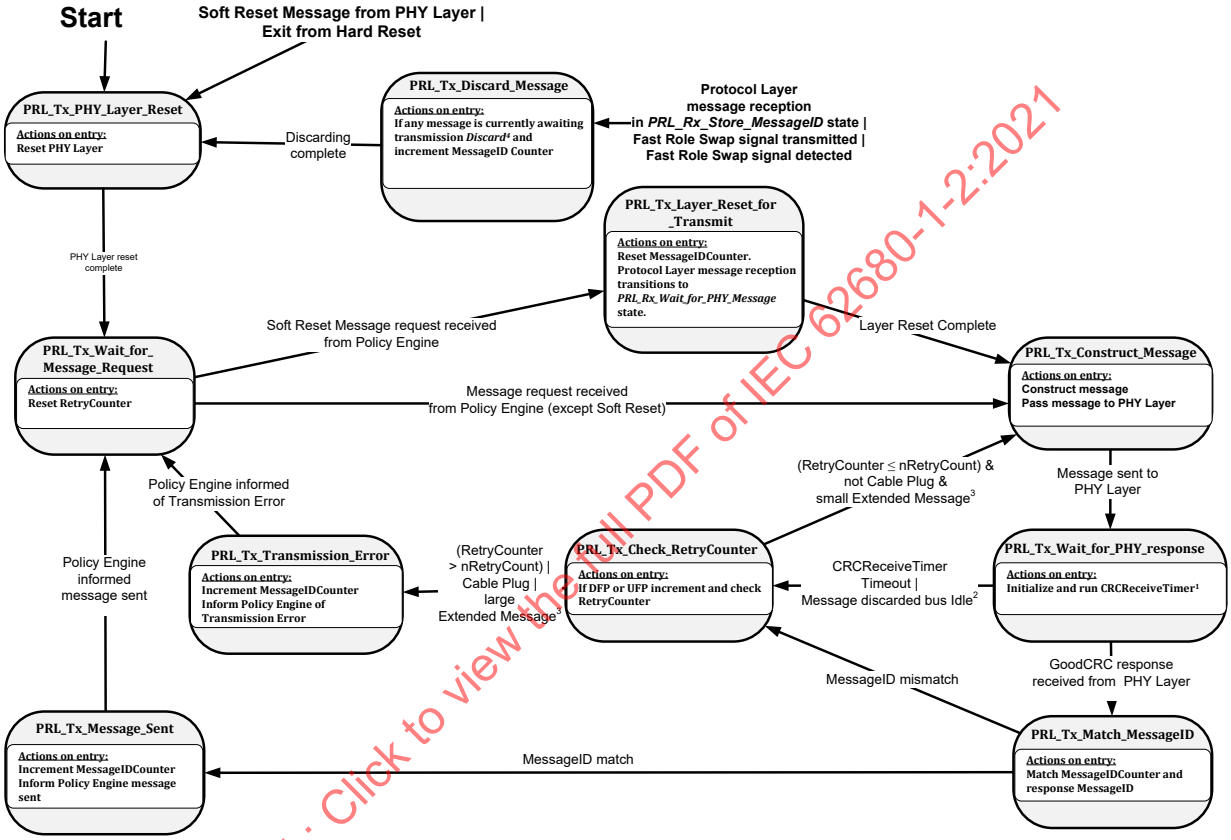
- envoyer le message au diagramme d'état de Tx fragmenté;
- passer à l'état ***RTR_Wait_for_Message_From_Protocol_Layer***.

6.11.2.2 Transmission d'un message de la couche protocole

6.11.2.2.1 Diagramme d'état commun de transmission d'un message de la couche protocole

La Figure 6-54 représente le comportement d'état, commun entre la source et le destinataire, de la couche protocole lors de la transmission d'un message.

Figure 6-54 Diagramme d'états commun de transmission d'un message de la couche protocole



Anglais	Français
Start	Début
Soft Reset occurred from PHY Layer	Une réinitialisation logicielle a eu lieu depuis la couche PHY
Exit from Hard Reset	Sortie de la réinitialisation matérielle
Reset PHY Layer	Réinitialiser la couche PHY
Discarding complete	Elimination terminée
If any message is currently awaiting transmission Discard[4] and increment MessageID Counter	Si aucun message n'est en attente de transmission, ignorer[4] et incrémenter le compteur de MessageID
Protocol Layer message reception in PRL_Rx_Store_MessageID state | Fast Role Swap signal transmitted | Fast Role Swap signal detected	Réception du message de la couche de protocole à l'état PRL_Rx_Store_MessageID | Signal de permutation rapide des rôles transmis | Signal de permutation rapide des rôles détecté
PHY Layer reset complete	Réinitialisation de la couche PHY terminée
Soft Reset Message request received from Policy Engine	Demande de message de réinitialisation logicielle reçue du moteur de politique
Actions on entry	Actions à l'entrée

Reset MessageIDCounter. Protocol Layer message reception transitions to PRL_Rx_Wait_for_PHY_Message state.	Réinitialiser le MessageIDCounter. La réception du message de la couche de protocole passe à l'état PRL_Rx_Wait_for_PHY_Message
Layer Reset Complete	Réinitialisation de la couche terminée
Reset RetryCounter	Réinitialiser le RetryCounter
Message request received from Policy Engine (except Soft Reset)	Demande de message reçue du moteur de politique (sauf pour la réinitialisation logicielle)
Construct message Pass message to PHY Layer	Construire le message Transmettre le message à la couche PHY
Policy engine informed of Transmission Error	Moteur de politique informé de l'erreur de transmission
(RetryCounter ≤ nRetryCount) & not Cable Plug & small Extended Message[3]	(RetryCounter ≤ nRetryCount), pas de fiche de câble, petit message étendu[3]
Message sent to PHY Layer	Message envoyé à la couche PHY
Policy Engine informed message sent	Le moteur de politique est informé de l'envoi du message
Increment MessageIDCounter Informe Policy Engine of Transmission Error	Incrémenter le MessageIDCounter Informer le moteur de politique de l'erreur de transmission
(RetryCounter > nRetryCount) | Cable Plug | large Extended Message[3]	(RetryCounter > nRetryCount) | Fiche de câble | grand message étendu[3]
If DFP or UFP increment and check RetryCounter	Si DFP ou UFP, incrémenter et vérifier le RetryCounter
CRCReceiveTimer Timeout | Message discarded bus Idle[2]	Temporisation CRCReceiveTimer | Message éliminé, bus inactif[2]
Initialize and run CRCReceiveTimer[1]	Initialiser et lancer le CRCReceiveTimer[1]
MessageID mismatch	Le MessageID ne correspond pas
GoodCRC response received from PHY Layer	Réponse GoodCRC reçue de la couche physique
Inform Policy Engine message sent	Informer le moteur de politique de l'envoi du message
MessageID match	Le MessageID correspond
Match MessageIDCounter and response MessageID	Faire correspondre le MessageIDCounter et le MessageID de la réponse

[1] Le ***CRCReceiveTimer*** n'est lancé qu'après l'envoi du message par la couche PHY. Si le message n'est pas envoyé en raison d'une voie occupée, le ***CRCReceiveTimer*** n'est pas lancé (voir 6.6.1).

[2] Cette indication est envoyée par la couche PHY lorsqu'un message a été ***Rejeté(s/ée/ées)*** en raison de l'occupation de la voie de communication, redevenue inactive ensuite (voir 5.7). Le ***CRCReceiveTimer*** n'est pas lancé dans ce cas puisqu'aucun message n'a été envoyé.

[3] Un "petit" message étendu est un message étendu pour lequel ***Data Size*** ≤ ***MaxExtendedMsgLegacyLen*** octets, ou pour lequel ***Data Size*** > ***MaxExtendedMsgLegacyLen*** octets, mais qui a été fragmenté. Un "grand" message étendu est un message étendu pour lequel ***Data Size*** > ***MaxExtendedMsgLegacyLen*** octets et qui n'a pas été fragmenté.

[4] Voir 6.10 pour savoir quand des messages sont ***Rejeté(s/ée/ées)***.

6.11.2.2.1.1 Etat PRL_Tx_PHY_Layer_Reset

La couche protocole ***Devoir/Doit/Doivent*** entrer dans l'état ***PRL_Tx_PHY_Layer_Reset***:

- au démarrage;
- à la suite d'une demande de réinitialisation logicielle reçue par la couche PHY;
- à la sortie d'une réinitialisation matérielle.

En entrant dans l'état ***PRL_Tx_PHY_Layer_Reset***, la couche protocole ***Devoir/Doit/Doivent*** réinitialiser la couche PHY (effacer les messages en attente et activer les communications).

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Wait_for_Message_Request*** lorsque:

- la réinitialisation de la couche PHY est terminée.

6.11.2.2.1.2 Etat PRL_Tx_Wait_for_Message_Request

Dans l'état ***PRL_Tx_Wait_for_Message_Request***, la couche protocole attend que le moteur de politique lui demande d'envoyer un message.

En entrant dans l'état ***PRL_Tx_Wait_for_Message_Request***, la couche protocole ***Devoir/Doit/Doivent*** réinitialiser le ***RetryCounter***.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Construct_Message*** lorsque:

- une demande de message qui n'est pas un message ***Soft_Reset*** est reçue du moteur de politique.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Layer_Reset_for_Transmit*** lorsque:

- une demande de message qui est un message ***Soft_Reset*** est reçue du moteur de politique.

6.11.2.2.1.3 Etat PRL_Tx_Layer_Reset_for_Transmit

En entrant dans l'état ***PRL_Tx_Layer_Reset_for_Transmit***, la couche protocole ***Devoir/Doit/Doivent*** réinitialiser le ***MessageIDCounter***. La couche protocole ***Devoir/Doit/Doivent*** passer la réception du message de couche protocole à l'état ***PRL_Rx_Wait_for_PHY_Message*** (voir 6.11.2.3.1) afin de réinitialiser le ***MessageID*** stocké.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Construct_Message*** lorsque:

- les actions de réinitialisation de la couche de cet état sont terminées.

6.11.2.2.1.4 Etat PRL_Tx_Construct_Message

A l'entrée dans l'état ***PRL_Tx_Construct_Message***, la couche protocole ***Devoir/Doit/Doivent*** construire le message demandé par le moteur de politique, ou renvoyer un message précédemment construit, puis transmettre ce message à la couche PHY.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Wait_for_PHY_Response*** lorsque:

- le message a été envoyé à la couche PHY.

6.11.2.2.1.5 Etat PRL_Tx_Wait_for_PHY_Response

A l'entrée dans l'état ***PRL_Tx_Wait_for_PHY_Response***, après que le message est envoyé, la couche protocole ***Devoir/Doit/Doivent*** initialiser et lancer le ***CRCReceiveTimer*** (voir 6.6.1).

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Match_MessageID*** lorsque:

- un message de réponse ***GoodCRC*** est reçu de la couche PHY.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Check_RetryCounter*** lorsque:

- Le ***CRCReceiveTimer*** a expiré.
- La couche PHY indique qu'un message a été ***Rejeté(s/ée/ées)*** en raison de l'occupation de la voie qui est maintenant inactive (voir 5.7).

6.11.2.2.1.6 Etat PRL_Tx_Match_MessageID

En entrant dans l'état ***PRL_Tx_Match_MessageID***, la couche protocole ***Devoir/Doit/Doivent*** comparer le ***MessageIDCounter*** et le ***MessageID*** du message ***GoodCRC*** reçu.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Message_Sent*** lorsque:

- le MessageIDCounter et le ***MessageID*** du message ***GoodCRC*** reçu correspondent.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Check_RetryCounter*** lorsque:

- le MessageIDCounter et le ***MessageID*** du message ***GoodCRC*** reçu ne correspondent pas.

6.11.2.2.1.7 Etat PRL_Tx_Message_Sent

A l'entrée dans l'état ***PRL_Tx_Message_Sent***, la couche protocole ***Devoir/Doit/Doivent*** incrémenter le ***MessageIDCounter*** et informer le moteur de politique que le message a été envoyé.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Wait_for_Message_Request*** lorsque:

- le moteur de politique a été informé que le message a été envoyé.

6.11.2.2.1.8 Etat PRL_Tx_Check_RetryCounter

En entrant dans l'état ***PRL_Tx_Check_RetryCounter***, la couche protocole d'un DFP ou d'un UFP ***Devoir/Doit/Doivent*** incrémenter la valeur du ***RetryCounter***, puis la vérifier afin de déterminer s'il est nécessaire de retenter d'envoyer le message. Noter que les fiches de câbles ne relancent pas les messages et n'utilisent donc pas le ***RetryCounter***.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Construct_Message*** afin de retenter d'envoyer un message lorsque:

- ***RetryCounter*** ≤ ***nRetryCount*** et
- il ne s'agit pas d'une fiche de câble; et
- il s'agit d'un message étendu pour lequel ***Data Size*** ≤ ***MaxExtendedMsgLegacyLen***; ou
- il s'agit d'un message étendu qui a été fragmenté.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Transmission_Error*** lorsque:

- ***RetryCounter*** > ***nRetryCount***; ou
- il s'agit d'une fiche de câble qui n'effectue pas de relance;
- il s'agit d'un message étendu pour lequel ***Data Size*** > ***MaxExtendedMsgLegacyLen*** et qui n'a pas été fragmenté.

6.11.2.2.1.9 Etat PRL_Tx_Transmission_Error

En entrant dans l'état ***PRL_Tx_Transmission_Error***, la couche protocole ***Devoir/Doit/Doivent*** incrémenter le ***MessageIDCounter*** et informer le moteur de politique de l'erreur de transmission.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Wait_for_Message_Request*** lorsque:

- le moteur de politique a été informé de l'erreur de transmission.

6.11.2.2.1.10 Etat PRL_Tx_Discard_Message

La transmission d'un message de la couche protocole ***Devoir/Doit/Doivent*** entrer dans l'état ***PRL_Tx_Discard_Message*** lorsque:

- la réception du message de la couche protocole a reçu un message entrant; ou
- le signal de permutation rapide des rôles est en cours de transmission (voir 5.8.5.6);
- le signal de permutation rapide des rôles est détecté (voir 5.8.6.3).

En entrant dans l'état ***PRL_Tx_Discard_Message***, si un message est dans la file en attente de transmission, la couche protocole ***Devoir/Doit/Doivent Rejeter*** le message selon les règles de la section 6.10, et incrémenter le ***MessageIDCounter***.

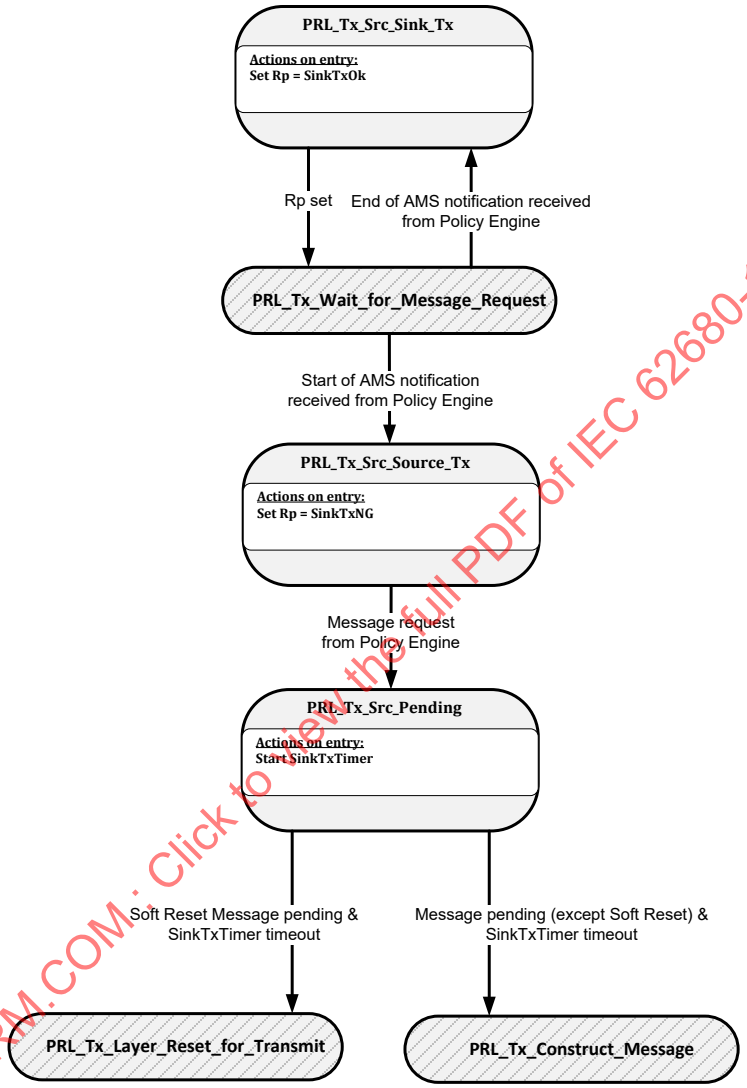
La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_PHY_Layer_Reset*** lorsque:

- l'élimination est terminée; la file d'attente des messages est vide.

6.11.2.2.2 Diagramme d'état de transmission d'un message de la couche protocole source

La Figure 6-55 représente le comportement d'état de la couche protocole d'un destinataire lors de la transmission d'un message.

Figure 6-55 Diagramme d'états de transmission d'un message de la couche protocole Source



Anglais	Français
Actions on entry	Actions à l'entrée
Set Rp = SinkTxOk	Définir Rp = SinkTxOk
Rp set	Rp définie
End of AMS notification received from Policy Engine	Fin de la notification d'AMS reçue du moteur de politique
Start of AMS notification received from Policy Engine	Début de la notification d'AMS reçue du moteur de politique
Set Rp = SinkTxNG	Définir Rp = SinkTxNG
Message request from Policy Engine	Demande de message du moteur de politique
Start SinkTxTimer	Lancer le SinkTxTimer

Soft Reset Message pending & SinkTxTimer timeout	Message de réinitialisation logicielle en attente et SinkTxTimer expiré
Message pending (except Soft Reset) & SinkTxTimer timeout	Message en attente (sauf réinitialisation logicielle) et SinkTxTimer expiré

6.11.2.2.2.1 Etat PRL_Tx_Src_Sink_Tx

Dans l'état ***PRL_Tx_Src_Sink_Tx***, la source définit Rp sur ***SinkTxOk***, permettant au destinataire de démarrer une séquence atomique de messages (AMS).

La couche protocole d'une source ***Devoir/Doit/Doivent*** passer de l'état ***PRL_Tx_Wait_for_Message_Request*** à l'état ***PRL_Tx_Src_Sink_Tx*** lorsque:

- une notification est reçue du moteur de politique, indiquant que la fin d'une AMS a été atteinte.

A l'entrée dans l'état ***PRL_Tx_Src_Sink_Tx***, la couche protocole ***Devoir/Doit/Doivent*** demander à la couche PHY de définir la Rp sur ***SinkTxOk***.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Wait_for_Message_Request*** lorsque:

- la Rp a été définie.

6.11.2.2.2.2 Etat PRL_Tx_Src_Source_Tx

Dans l'état ***PRL_Tx_Src_Source_Tx***, la source définit Rp sur ***SinkTxNG***, permettant à la source de démarrer une séquence atomique de messages (AMS).

La couche protocole d'une source ***Devoir/Doit/Doivent*** passer de l'état ***PRL_Tx_Wait_for_Message_Request*** à l'état ***PRL_Tx_Src_Source_Tx*** lorsque:

- une notification est reçue du moteur de politique indiquant qu'une AMS va être démarrée.

En entrant dans l'état ***PRL_Tx_Src_Source_Tx***, la couche protocole ***Devoir/Doit/Doivent*** définir la Rp sur ***SinkTxNG***.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Src_Pending*** lorsque:

- une demande de message est reçue du moteur de politique.

6.11.2.2.2.3 Etat PRL_Tx_Src_Pending

Dans l'état ***PRL_Tx_Src_Pending***, la couche protocole a un message mis en mémoire tampon et prêt à être transmis.

En entrant dans l'état ***PRL_Tx_Src_Pending***, le ***SinkTxTimer*** ***Devoir/Doit/Doivent*** être initialisé et exécuté.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Construct_Message*** lorsque:

- la demande de message en attente provenant du moteur de politique n'est pas un message ***Soft_Reset***; et
- le ***SinkTxTimer*** a expiré.

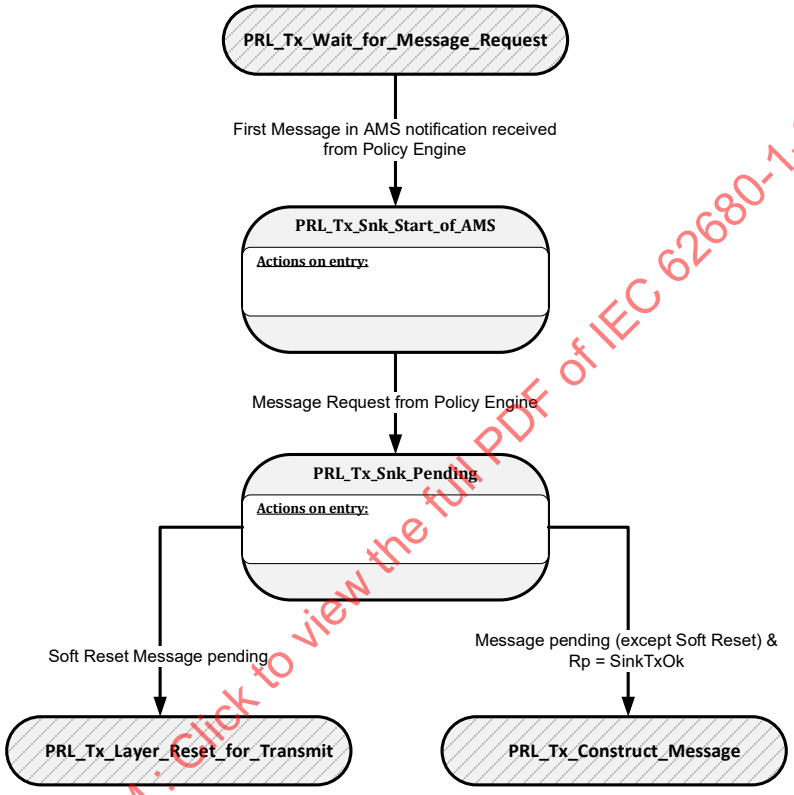
La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Layer_Reset_for_Transmit*** lorsque:

- la demande de message en attente provenant du moteur de politique est un message ***Soft_Reset***; et
- le ***SinkTxTimer*** a expiré.

6.11.2.2.3 Diagramme d'état de transmission d'un message de la couche protocole destinataire

La Figure 6-56 représente le comportement d'état de la couche protocole d'un destinataire lors de la transmission d'un message.

Figure 6-56 Diagramme d'états de transmission d'un message de la couche protocole destinataire



Anglais	Français
First Message in AMS notification received from Policy Engine	Premier message de la notification d'AMS reçu du moteur de politique
Actions on entry:	Actions à l'entrée:
Message Request from Policy Engine	Demande de message du moteur de politique
Soft Reset Message pending & Rp = SinkTxOk	Message de réinitialisation logicielle en attente et Rp = SinkTxOk[2]
Message pending (except Soft Reset) & Rp = SinkTxOk	Message en attente (sauf réinitialisation logicielle) et Rp = SinkTxOk

6.11.2.2.3.1 Etat PRL_Tx_Snk_Start_of_AMS

Dans l'état ***PRL_Tx_Snk_Start_of_AMS***, la couche protocole attend le premier message d'une AMS initiée par un destinataire.

La couche protocole d'un destinataire ***Devoir/Doit/Doivent*** passer de l'état ***PRL_Tx_Wait_for_Message_Request*** à l'état ***PRL_Tx_Snk_Start_of_AMS*** lorsque:

- une notification est reçue du moteur de politique, indiquant que le prochain message envoyé par le destinataire sera le début d'une AMS.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Snk_Pending*** lorsque:

- une demande de message est reçue du moteur de politique.

6.11.2.2.3.2 Etat PRL_Tx_Snk_Pending

Dans l'état ***PRL_Tx_Snk_Pending***, le premier message d'une AMS initiée par un destinataire de la couche protocole est prêt à être envoyé. La couche protocole attend le passage de la Rp à ***SinkTxOk*** pour envoyer le message.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Construct_Message*** lorsque:

- un message est en attente, mais ce n'est pas un message ***Soft_Reset***; et
- la Rp est définie sur***SinkTxOk.***

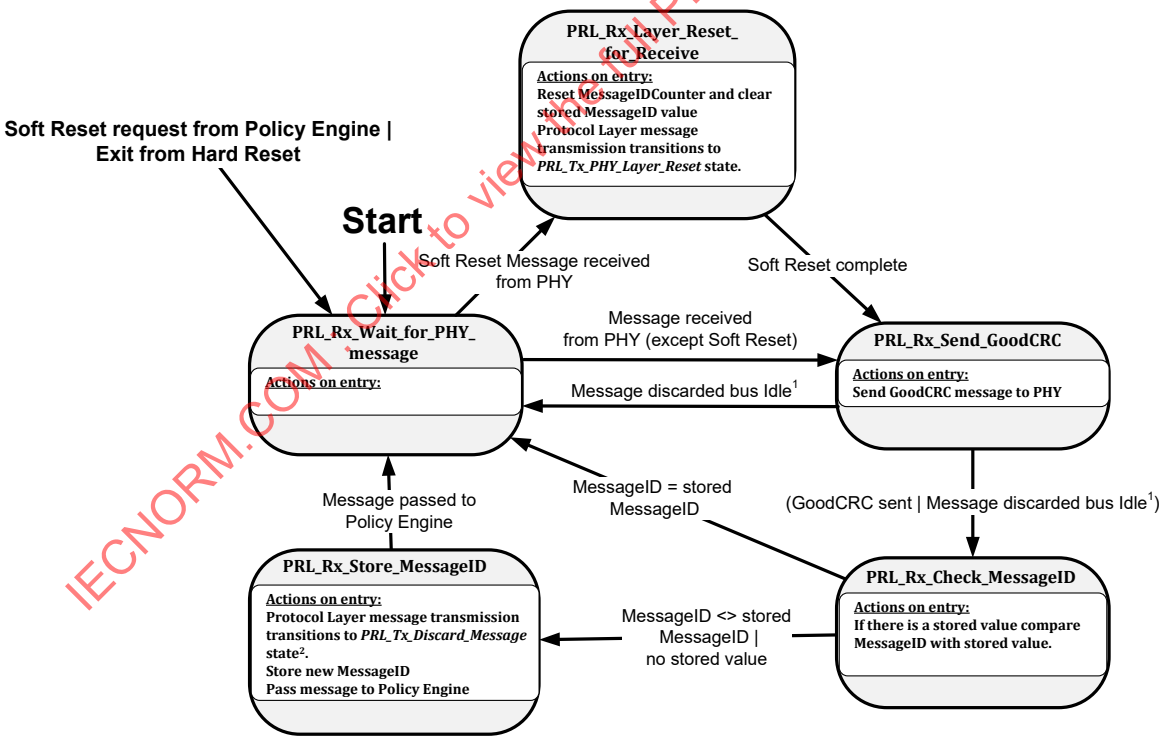
La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Layer_Reset_for_Transmit*** lorsque:

- un message ***Soft_Reset*** est en attente.

6.11.2.3 Réception d'un message de la couche protocole

La Figure 6-57 représente le comportement d'état de la couche protocole lors de la réception d'un message.

Figure 6-57 Réception d'un message de la couche protocole



Anglais	Français
Soft Reset request from Policy Engine | Exit from Hard Reset	Demande de réinitialisation logicielle du moteur de politique | Sortie de la réinitialisation matérielle
Actions on entry: Reset MessageIDCounter and clear stored MessageID value	Actions à l'entrée: Réinitialiser le MessageIDCounter et effacer la valeur de MessageID stockée

Protocol Layer message transmission transitions to PRL_Tx_PHY_Layer_Reset state.	La transmission du message de la couche de protocole passe à l'état PRL_Tx_PHY_Layer_Reset.
Start	Début
Soft Reset Message received from PHY	Message de réinitialisation logicielle reçu de la couche physique
Soft Reset complete	Réinitialisation logicielle terminée
Actions on entry:	Actions à l'entrée:
Message received from PHY (except Soft Reset)	Message reçu de la couche physique (sauf réinitialisation logicielle)
Message discarded bus Idle[1]	Message rejeté, bus inactif[1]
Actions on entry: Send GoodCRC message to PHY	Actions à l'entrée: Envoyer le message GoodCRC à la couche physique
Message passed to Policy Engine	Message transmis au moteur de politique
MessageID = stored MessageID	MessageID = MessageID stocké
(GoodCRC sent | Message discarded bus Idle[1])	(GoodCRC envoyé | Message rejeté, bus inactif[1])
Actions on entry: Protocol Layer message transmission transitions to PRL_Tx_Discard_Message state[2]. Store new MessageID Pass message to Policy Engine	Actions à l'entrée: La transmission du message de la couche de protocole passe à l'état PRL_Tx_Discard_Message[2]. Stocker un nouveau MessageID Transmettre le message au moteur de politique
MessageID <> stored MessageID | no stored value	MessageID <> MessageID stocké | aucune valeur stockée
Actions on entry: If there is a stored value compare MessageID with stored value.	Actions à l'entrée: S'il existe une valeur stockée, comparer le MessageID à la valeur stockée.

[1] Cette indication est envoyée par la couche PHY lorsqu'un message a été ***Rejeté(s/ée/ées)*** en raison de l'occupation de la voie de communication, redevenue inactive ensuite (voir 5.7). Deux autres transitions possibles sont représentées.

[2] Dans le cas de la réception d'un message PING, ***Il convient de ne pas Rejeté(s/ée/ées)*** le message sortant, afin de maintenir des communications robustes en présence de collisions.

6.11.2.3.1 Etat PRL_Rx_Wait_for_PHY_Message

La couche protocole ***Devoir/Doit/Doivent*** entrer dans l'état ***PRL_Rx_Wait_for_PHY_Message***:

- au démarrage;
- à la suite d'une demande de réinitialisation logicielle provenant du moteur de politique;
- à la sortie d'une réinitialisation matérielle.

Dans l'état ***PRL_Rx_Wait_for_PHY_Message***, la couche protocole attend que la couche PHY lui transmette un message reçu.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Rx_Send_GoodCRC*** lorsque:

- un message est transmis par la couche PHY.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Rx_Layer_Reset_for_Receive*** lorsque:

- un message ***Soft_Reset*** est reçu de la couche PHY.

6.11.2.3.2 Etat PRL_Rx_Layer_Reset_for_Receive

A l'entrée dans l'état ***PRL_Rx_Layer_Reset_for_Receive***, la couche protocole ***Devoir/Doit/Doivent*** réinitialiser le ***MessageIDCounter*** et effacer le ***MessageID*** stocké. La couche protocole ***Devoir/Doit/Doivent*** passer la transmission du message de la couche protocole à l'état ***PRL_Tx_Wait_for_Message_Request*** (voir 6.11.2.2.1.1).

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Rx_Send_GoodCRC*** lorsque:

- les actions de réinitialisation logicielle dans cet état sont terminées.

6.11.2.3.3 Etat PRL_Rx_Send_GoodCRC

A l'entrée dans l'état ***PRL_Rx_Send_GoodCRC***, la couche protocole ***Devoir/Doit/Doivent*** construire un message ***GoodCRC*** et demander à la couche PHY de le transmettre.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Rx_Check_MessageID*** lorsque:

- le message ***GoodCRC*** a été transmis à la couche PHY.

Lorsque la couche PHY indique qu'un message a été ***Rejeté(s/ée/ées)*** en raison de l'occupation de la voie de communication qui est maintenant inactive (voir 5.7), la couche protocole ***Devoir/Doit/Doivent***:

- passer à l'état ***PRL_Rx_Check_MessageID*** ou
- passer à l'état ***PRL_Rx_Wait_for_PHY_Message***.

6.11.2.3.4 Etat PRL_Rx_Check_MessageID

A l'entrée dans l'état ***PRL_Rx_Check_MessageID***, la couche protocole ***Devoir/Doit/Doivent*** comparer le ***MessageID*** du message reçu avec sa valeur stockée, le cas échéant.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Rx_Wait_for_PHY_Message*** lorsque:

- le ***MessageID*** du message reçu est égal à la valeur de ***MessageID*** stockée, puisqu'il s'agit d'une relance de message qui ***Devoir/Doit/Doivent*** être ***Rejeté(s/ée/ées)***.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Rx_Store_MessageID*** lorsque:

- le ***MessageID*** du message reçu n'est pas égal à la valeur de ***MessageID*** stockée, puisqu'il s'agit d'un nouveau message; ou
- ce message reçu est le premier, et aucune valeur de ***MessageID*** n'est encore stockée.

6.11.2.3.5 Etat PRL_Rx_Store_MessageID

En entrant dans l'état ***PRL_Rx_Store_MessageID***, la couche protocole ***Devoir/Doit/Doivent*** passer la transmission du message de la couche protocole à l'état ***PRL_Tx_Discard_Message*** (sauf lorsqu'un message ***Ping*** a été reçu, ***Il convient de ne pas*** utiliser l'état ***PRL_Tx_Discard_Message***), remplacer la valeur stockée du ***MessageID*** par la valeur du ***MessageID*** contenue dans le message reçu, et transmettre le message au moteur de politique.

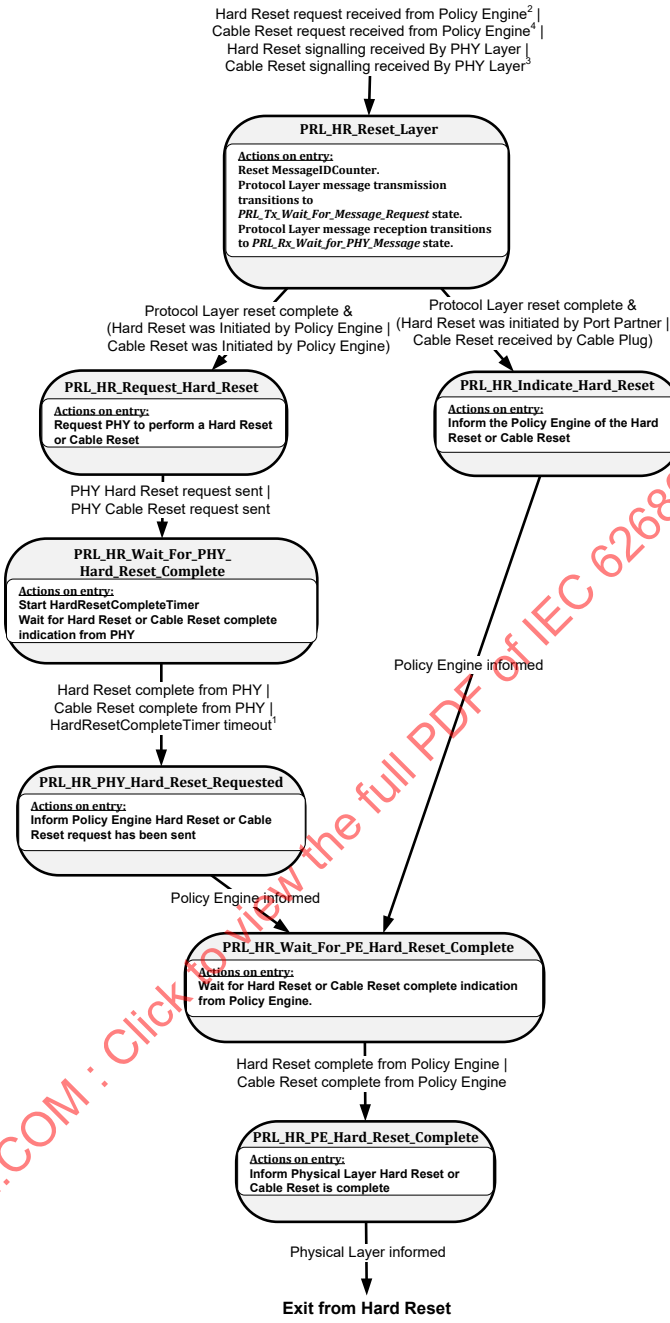
La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Rx_Wait_for_PHY_Message*** lorsque:

- le message a été transmis au moteur de politique.

6.11.2.4 Opération de réinitialisation matérielle

La Figure 6-58 représente le comportement d'état de la couche protocole lors de la réception d'une demande de réinitialisation matérielle ou de réinitialisation de câble provenant du moteur de politique, ou d'un signal ***Hard Reset*** ou ***Cable Reset*** provenant de la couche physique (voir aussi 6.8.3 et 6.8.4).

Figure 6-58 Réinitialisation matérielle/de câble



Anglais	Français
Hard Reset request received from Policy Engine[2] | Cable Reset request received form Policy Engine[4] | Hard Reset signalling received By PHY Layer | Cable Reset signalling received By PHY Layer[3]	Demande de réinitialisation matérielle reçue du moteur de politique[2] | Demande de réinitialisation de câble reçue du moteur de politique [4] | Signal de réinitialisation matérielle reçu par la couche PHY | Signal de réinitialisation de câble reçu par la couche PHY[3]
Actions on entry: Reset MessageIDCounter.	Actions à l'entrée: Réinitialiser le MessageIDCounter.

Protocol Layer message transmission transitions to *PRL_Tx_Wait_For_Message_Request* state. Protocol Layer message reception transitions to *PRL_Rx_Wait_For_PHY_Message* state.	La transmission du message de la couche de protocole passe à l'état *PRL_Tx_Wait_For_Message_Request.* La réception du message de la couche de protocole passe à l'état *PRL_Rx_Wait_For_PHY_Message.*
Protocol Layer reset complete & (Hard Reset was Initiated by Policy Engine | Cable Reset was Initiated by Policy Engine)	Réinitialisation de la couche de protocole terminée et (la réinitialisation matérielle a été initiée par le moteur de politique | la réinitialisation de câble a été initiée par le moteur de politique)
Protocol Layer reset complete & (Hard Reset was Initiated by Port Partner | Cable Reset received by Cable Plug)	Réinitialisation de la couche de protocole terminée et (la réinitialisation matérielle a été initiée par le port partenaire | la réinitialisation de câble a été reçue par la fiche de câble)
Actions on entry: Request PHY to perform a Hard Reset or Cable Reset	Actions à l'entrée: Demander à la couche physique d'effectuer une réinitialisation matérielle ou de câble
Actions on entry: Inform the Policy Engine of the Hard Reset or Cable Reset	Actions à l'entrée: Informer le moteur de politique de la réinitialisation matérielle ou de câble
PHY Hard Reset request sent | PHY Cable Reset request sent	Demande de réinitialisation matérielle à la couche physique envoyée | Demande de réinitialisation de câble à la couche physique envoyée
Actions on entry: Start HardResetCompleteTimer Wait for Hard Reset or Cable Reset complete indication from PHY	Actions à l'entrée: Lancer le HardResetCompleteTimer Attendre l'annonce de la couche physique de la fin de la réinitialisation matérielle ou de câble
Policy Engine informed	Moteur de politique informé
Hard Reset complete from PHY | Cable Reset complete from PHY | HardResetCompleteTimer timeout[1]	Fin de la réinitialisation matérielle annoncée par la couche physique | Fin de la réinitialisation de câble annoncée par la couche physique | Expiration du HardResetCompleteTimer[1]
Actions on entry: Inform Policy Engine Hard Reset or Cable Reset has been sent	Actions à l'entrée: Informer le moteur de politique qu'une réinitialisation matérielle ou de câble a été envoyée
Actions on entry: Wait for Hard Reset or Cable Reset complete indication from Policy Engine.	Actions à l'entrée: Attendre l'annonce du moteur de politique de la fin de la réinitialisation matérielle ou de câble
Hard Reset complete from Policy Engine | Cable Reset complete from Policy Engine	Fin de la réinitialisation matérielle annoncée par le moteur de politique | Fin de la réinitialisation de câble annoncée par le moteur de politique
Actions on entry: Inform Physical Layer Hard Reset or Cable Reset is complete	Actions à l'entrée: Informer la couche physique de la fin de la réinitialisation matérielle ou de câble
Physical Layer informed	Couche physique informée
Exit from Hard Reset	Sortie de la réinitialisation matérielle

[1] Si le ***HardResetCompleteTimer*** a expiré, cela signifie que la couche PHY attend toujours d'envoyer la commande Hard Reset en raison d'un canal non inactif. Cette condition est effacée après que la réinitialisation matérielle du moteur de politique est terminée.

[2] Les fiches de câbles ne génèrent pas de signal ***Hard Reset***, mais elles sont exigées pour surveiller la présence d'un signal ***Hard Reset*** entre les ports partenaires et répondre le cas échéant en les réinitialisant.

[3] Le signal de réinitialisation de câble n'est reconnu que par une fiche de câble.

[4] Le signal de réinitialisation de câble ne peut pas être généré par les fiches de câbles.

6.11.2.4.1 Etat PRL_HR_Reset_Layer

L'état ***PRL_HR_Reset_Layer*** définit le mode opératoire des diagrammes d'états de transmission et de réception de la couche protocole au cours d'une réinitialisation matérielle ou de câble. Au cours d'une réinitialisation matérielle, aucun message de protocole d'alimentation électrique par port USB n'est envoyé ou reçu; seul le signal ***Hard Reset*** est présent, après lequel il est admis par hypothèse que la voie de communication est désactivée par la couche physique jusqu'à l'achèvement de la réinitialisation matérielle. Au cours d'une réinitialisation de câble, aucun message de protocole d'alimentation électrique par port USB n'est envoyé à la fiche de câble ou reçu de celle-ci, mais les autres communications d'alimentation électrique par port USB ***Pouvoir/Peut/Peuvent*** se poursuivre.

La couche protocole ***Devoir/Doit/Doivent*** entrer dans l'état ***PRL_HR_Reset_Layer*** depuis n'importe quel autre état lorsque:

- une demande de réinitialisation matérielle est reçue du moteur de politique; ou
- un signal ***Hard Reset*** est reçu de la couche physique; ou
- une demande de réinitialisation de câble est reçue du moteur de politique; ou
- un signal ***Cable Reset*** est reçu de la couche physique.

En entrant dans l'état ***PRL_HR_Reset_Layer***, la couche protocole ***Devoir/Doit/Doivent*** réinitialiser le ***MessageIDCounter***. Elle ***Devoir/Doit/Doivent*** également réinitialiser les états des diagrammes d'états de transmission et de réception de la couche protocole à leur point de départ. Le diagramme d'état de transmission de la couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Tx_Wait_for_Message_Request***. Le diagramme d'état de réception de la couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_Rx_Wait_for_PHY_Message***.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_HR_Request_Hard_Reset*** lorsque:

- la réinitialisation de la couche protocole est terminée; et
  - la demande de réinitialisation matérielle provient du moteur de politique; ou
  - la demande de réinitialisation de câble provient du moteur de politique.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_HR_Indicate_Hard_Reset*** lorsque:

- la réinitialisation de la couche protocole est terminée; et
  - la demande de réinitialisation matérielle a été soumise par la couche physique; ou
  - une demande de réinitialisation de câble a été soumise par la couche physique (fiche de câble uniquement).

6.11.2.4.2 Etat PRL_HR_Indicate_Hard_Reset

En entrant dans l'état ***PRL_HR_Indicate_Hard_Reset***, la couche protocole ***Devoir/Doit/Doivent*** indiquer au moteur de politique qu'un signal ***Hard Reset*** ou qu'un signal ***Cable Reset*** a été reçu.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_HR_Wait_for_PE_Hard_Reset_Complete*** lorsque:

- l'indication a été envoyée au moteur de politique.

6.11.2.4.3 Etat PRL_HR_Request_Hard_Reset

En entrant dans l'état ***PRL_HR_Request_Hard_Reset***, la couche protocole ***Devoir/Doit/Doivent*** demander à la couche physique d'envoyer un signal ***Hard Reset*** ou un signal ***Cable Reset***.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_HR_Wait_for_PHY_Hard_Reset_Complete*** lorsque:

- la demande de signal ***Hard Reset*** de la couche physique a été envoyée; ou
- la demande de signal ***Cable Reset*** de la couche physique a été envoyée.

6.11.2.4.4 Etat PRL_HR_Wait_for_PHY_Hard_Reset_Complete

Dans l'état ***PRL_HR_Wait_for_PHY_Hard_Reset_Complete***, la couche protocole ***Devoir/Doit/Doivent*** lancer le ***HardResetCompleteTimer*** et attendre que la couche PHY annonce l'achèvement de la réinitialisation matérielle ou de la réinitialisation de câble.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_HR_PHY_Hard_Reset_Requested*** lorsque:

- l'indication de l'achèvement de la réinitialisation matérielle est reçue de la couche PHY; ou
- l'indication de l'achèvement de la réinitialisation de câble est reçue de la couche PHY; ou
- le ***HardResetCompleteTimer*** a expiré.

6.11.2.4.5 Etat PRL_HR_PHY_Hard_Reset_Requested

En entrant dans l'état ***PRL_HR_PHY_Hard_Reset_Requested***, la couche protocole ***Devoir/Doit/Doivent*** informer le moteur de politique que la couche PHY a été sollicitée pour effectuer une réinitialisation matérielle ou de câble.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_HR_Wait_for_PE_Hard_Reset_Complete*** lorsque:

- l'indication a été envoyée au moteur de politique.

6.11.2.4.6 Etat PRL_HR_Wait_for_PE_Hard_Reset_Complete

Dans l'état ***PRL_HR_Wait_for_PE_Hard_Reset_Complete***, la couche protocole ***Devoir/Doit/Doivent*** attendre que le moteur de politique annonce l'achèvement de la réinitialisation matérielle ou de câble.

La couche protocole ***Devoir/Doit/Doivent*** passer à l'état ***PRL_HR_PE_Hard_Reset_Complete*** lorsque:

- l'indication de l'achèvement de la réinitialisation matérielle est reçue du moteur de politique; ou
- l'indication de l'achèvement de la réinitialisation de câble est reçue du moteur de politique.

6.11.2.4.7 PRL_HR_PE_Hard_Reset_Complete

En entrant dans l'état ***PRL_HR_PE_Hard_Reset_Complete***, la couche protocole ***Devoir/Doit/Doivent*** informer la couche physique de l'achèvement de la réinitialisation matérielle ou de câble.

La couche protocole ***Devoir/Doit/Doivent*** sortir de la réinitialisation matérielle et revenir au fonctionnement normal lorsque:

- la couche physique a été informée de l'achèvement de la réinitialisation matérielle et réactive ainsi la voie de communication. Si le signal ***Hard Reset*** est toujours en attente en raison d'une voie non inactive, il ***Devoir/Doit/Doivent*** être effacé et ne pas être envoyé; ou
- la couche physique a été informée de l'achèvement de la réinitialisation de câble.

6.11.3 Liste des états de la couche protocole

Le Tableau 6-68 répertorie les états utilisés par les différents diagrammes d'états.

Tableau 6-68 Etats de la couche protocole

Nom de l'état	Référence
Transmission d'un message de la couche protocole	
Transmission d'un message de la couche protocole commune	
PRL_Tx_PHY_Layer_Reset	Section 6.11.2.2.1.1
PRL_Tx_Wait_for_Message_Request	Section 6.11.2.2.1.2
PRL_Tx_Layer_Reset_for_Transmit	Section 6.11.2.2.1.3
PRL_Tx_Construct_Message	Section 6.11.2.2.1.4
PRL_Tx_Wait_for_PHY_Response	Section 6.11.2.2.1.5
PRL_Tx_Match_MessageID	Section 6.11.2.2.1.6
PRL_Tx_Message_Sent	Section 6.11.2.2.1.7
PRL_Tx_Check_RetryCounter	Section 6.11.2.2.1.8
PRL_Tx_Transmission_Error	Section 6.11.2.2.1.9
PRL_Tx_Discard_Message	Section 6.11.2.2.1.10
Transmission d'un message de la couche protocole source	
PRL_Tx_Src_Sink_Tx	Section 6.11.2.2.2.1
PRL_Tx_Src_Source_Tx	Section 6.11.2.2.2.2
PRL_Tx_Src_Pending	Section 6.11.2.2.2.3
Transmission d'un message de la couche protocole destinataire	
PRL_Tx_Snk_Start_of_AMS	Section 6.11.2.2.3.1
PRL_Tx_Snk_Pending	Section 6.11.2.2.3.2
Réception d'un message de la couche protocole	
PRL_Rx_Wait_for_PHY_Message	Section 6.11.2.3.1
PRL_Rx_Layer_Reset_for_Receive	Section 6.11.2.3.2
PRL_Rx_Send_GoodCRC	Section 6.11.2.3.3
PRL_Rx_Check_MessageID	Section 6.11.2.3.4
PRL_Rx_Store_MessageID	Section 6.11.2.3.5
Opération de réinitialisation matérielle	
PRL_HR_Reset_Layer	Section 6.11.2.4.1
PRL_HR_Indicate_Hard_Reset	Section 6.11.2.4.2
PRL_HR_Request_Hard_Reset	Section 6.11.2.4.3
PRL_HR_Wait_for_PHY_Hard_Reset_Complete	Section 6.11.2.4.4
PRL_HR_PHY_Hard_Reset_Requested	Section 6.11.2.4.5
PRL_HR_Wait_for_PE_Hard_Reset_Complete	Section 6.11.2.4.6
PRL_HR_PE_Hard_Reset_Complete	Section 6.11.2.4.7
Fragmentation	
Rx fragmenté	
RCH_Wait_For_Message_From_Protocol_Layer	Section 6.11.2.1.2.1
RCH_Pass_Up_Message	Section 6.11.2.1.2.2
RCH_Processing_Extended_Message	Section 6.11.2.1.2.3
RCH_Requesting_Chunk	Section 6.11.2.1.2.4

Nom de l'état	Référence
RCH_Waiting_Chunk	Section 6.11.2.1.2.5
RCH_Report_Error	Section 6.11.2.1.2.6
Tx fragmenté	
TCH_Wait_For_Message_Request_From_Policy_Engine	Section 6.11.2.1.3.1
TCH_Pass_Down_Message	Section 6.11.2.1.3.2
TCH_Wait_For_Transmision_Complete	Section 6.11.2.1.3.3
TCH_Message_Sent	Section 6.11.2.1.3.4
TCH_Prepare_To_Send_Chunked_Message	Section 6.11.2.1.3.5
TCH_Construct_Chunked_Message	Section 6.11.2.1.3.6
TCH_Sending_Chunked_Message	Section 6.11.2.1.3.7
TCH_Wait_Chunk_Request	Section 6.11.2.1.3.8
TCH_Message_Received	Section 6.11.2.1.3.9
TCH_Report_Error	Section 6.11.2.1.3.10
Routeur de messages fragmentés	
RTR_Wait_for_Message_From_Protocol_Layer	Section 6.11.2.1.4.1
RTR_Rx_Chunks	Section 6.11.2.1.4.2
RTR_Ping	Section 6.11.2.1.4.3
RTR_Tx_Chunks	Section 6.11.2.1.4.4

6.12 Applicabilité des messages

Les tableaux suivants décrivent les messages pris en charge par un port donné, selon sa capacité.

Lorsqu'un message est pris en charge, la caractéristique et la séquence de messages induites par le message ***Devoir/Doit/Doivent*** également être prises en charge. Par exemple, les destinataires qui utilisent la puissance de charge et prennent en charge le message ***GotoMin*** ***Devoir/Doit/Doivent*** être capable réduire leur consommation de courant lorsque cela est demandé par un message ***GotoMin***.

Les abréviations suivantes sont utilisées:

- N – ***Normatif(s/ve/ves)***; ***Devoir/Doit/Doivent*** être pris en charge par ce port/cette fiche de câble;
- CN – ***Normatif(s/ve/ves) conditionnel(s/le/les)***; ***Devoir/Doit/Doivent*** être pris en charge par un port/une fiche de câble donné(e) selon ses caractéristiques;
- R – Recommandé; ***Il convient d'/de/qu'/que*** le prendre en charge par ce port/cette fiche de câble;
- – ***Facultatif(s/ve/ves),***; ***Pouvoir/Peut/Peuvent*** être pris en charge par ce port/cette fiche de câble;
- NS – Non pris en charge, ***Devoir/Doit/Doivent*** aboutir à un message de réponse ***Not_Supported*** par ce port/cette fiche de câble lorsqu'il est reçu;
- I – ***Ignorer***; ***Devoir/Doit/Doivent*** être ***Ignoré(s/ée/ées)*** par ce port/cette fiche de câble lorsqu'il est reçu;
- NK – NAK; ce port/cette fiche de câble ***Devoir/Doit/Doivent*** renvoyer le répondeur NAK lorsque cette commande est reçue;
- NA – Non admis; ***Ne doit/doivent pas*** être transmis par ce port/cette fiche de câble;
- DR – Ne pas reconnaître; aucune réponse ne ***Devoir/Doit/Doivent*** avoir lieu (pas même un message ***GoodCRC***) de ce port/cette fiche de câble lorsqu'il est reçu.

Dans le cas ***Normatif(s/ve/ves) conditionnel(s/le/les)***, une note a été ajoutée pour indiquer la condition. La notation "CN/" est utilisée pour indiquer le niveau de prise en charge lorsque la condition n'est pas présente.

Les notations "R/" et "O/" sont utilisées pour indiquer la réponse lorsque le message Recommandé ou ***Facultatif(s/ve/ves),***n'est pas pris en charge.

Si NS/RJ/NK est indiqué pour les messages reçus, cela ne ***Devoir/Doit/Doivent*** s'appliquer qu'aux états ***PE_CBL_Ready***, ***PE_SNK_Ready*** ou ***PE_SRC_Ready***, car des messages inattendus reçus au cours d'une séquence de messages constituent des erreurs de protocole (voir 6.8.1).

La présente section couvre la prise en charge des messages de contrôle et de données pour les sources, les destinataires et les fiches de câbles. Il couvre également la prise en charge des commandes de VDM pour les DFP, les UFP et les fiches de câbles.

6.12.1 Applicabilité des messages de contrôle

Le Tableau 6-69 répertorie les messages de contrôle qui ***Devoir/Doit/Doivent/Ne doit/doivent pas*** être transmis et reçus/qu'***Il convient d'/de/qu'/que*** transmettre et de recevoir pour une source, un destinataire, une fiche de câble ou un VPD. Les exigences relatives aux ports d'alimentation double fonction et aux ports de données double fonction ***Devoir/Doit/Doivent*** remplacer les exigences relatives aux ports uniquement sources ou uniquement destinataires.

Tableau 6-69 Applicabilité des messages de contrôle

Type de message	Source	Destinataire	Alimentation double fonction	Données double fonction	Fiche de câble	VPD[12]
Message transmis						
Accept	N	N			N	N
DR_Swap	O	O		N	NA	NA
FR_Swap	NA	NA	R		NA	NA
Get_Country_Codes	CN[10]/NA	CN[10]/NA			NA	NA
Get_PPS_Status	NA	CN[9]			NA	NA
Get_Sink_Cap	R	NA	N		NA	NA
Get_Sink_Cap_Extended	R	NA	R		NA	NA
Get_Source_Cap	NA	R	N		NA	NA
Get_Source_Cap_Extended	NA	R	R		NA	NA
Get_Status	R	R			NA	NA
GoodCRC	N	N			N	N
GotoMin	CN[1]/O	NA			NA	NA
Not_Supported	N	N			N	NA
Ping	O	NA			NA	NA
Data_Reset	CN[13]/R	CN[13]/R			NA	NA
PR_Swap	NA	NA	N		NA	NA
PS_RDY	N	CN[4]/NA	N		NA	NA
Reject	N	NA	O	O	NA	NA
Soft_Reset	N	N			NA	NA
VCONN_Swap	R	R			NA	NA
Wait	CN[2]/O	NA	O	O	NA	NA
Message reçu						
Accept	N	N	N	N	I	I
DR_Swap	O/NS	O/NS		N	I	I
FR_Swap	NS	NS	CN[7]/NS		I	I
Get_Country_Codes	CN[10]/NS	CN[10]/NS			I	I
Get_PPS_Status	CN[9]/NS	NS			I	I
Get_Sink_Cap	NS	N	N		I	I
Get_Sink_Cap_Extended	NS	N	N		I	I
Get_Source_Cap	N	NS	N		I	I
Get_Source_Cap_Extended	CN[5]/NS	NS	CN[5]/NS		I	I
Get_Status	CN[6]/NS	CN[6]/NS	CN[6]/NS		CN[11]/I	I

Type de message	Source	Destinataire	Alimentation double fonction	Données double fonction	Fiche de câble	VPD[12]
GoodCRC	N	N			N	N
GotoMin	NS	R[3]			I	I
Not_Supported	N	N			CN[11]/I	I
Ping	NS	I			I	I
Data_Reset	CN[13]/R	CN[13]/R			I	I
PR_Swap	NS	NS	N		I	I
PS_RDY	CN[4]/NS	N	N		I	I
Reject	CN[8]/NS	N	N	N	I	I
Soft_Reset	N	N			N	N
VCONN_Swap	CN[4]/ NS	CN[4]/ NS			I	I
Wait	CN[8]/NS	N	N	N	I	I

Note 1: ***Devoir/Doit/Doivent*** être pris en charge par un hub avec plusieurs ports en aval. ***Il convient d'/de/qu'/que*** ils soient pris en charge par un hôte avec plusieurs ports en aval.
Note 2: ***Devoir/Doit/Doivent***être pris en charge lorsque la transmission des messages*GotoMin* est prise en charge.
Note 3: ***Il convient d'/de/qu'/que*** ils soient pris en charge par les destinataires qui utilisent l'alimentation électrique USB pour la charge.
Note 4: ***Devoir/Doit/Doivent*** être pris en charge par tout port qui peut fournir VCONN.
Note 5: ***Devoir/Doit/Doivent*** être pris en charge par des produits qui prennent en charge le message *Source_Capabilities_Extended*.
Note 6: ***Devoir/Doit/Doivent*** être pris en charge par des sources qui prennent en charge le message *Alert*.
Note 7: ***Devoir/Doit/Doivent*** être pris en charge lorsque le signal de permutation rapide des rôles est pris en charge.
Note 8: ***Devoir/Doit/Doivent*** être pris en charge lorsque *VCONN_Swap* est prise en charge.
Note 9: ***Devoir/Doit/Doivent*** être pris en charge lorsque la PPS est prise en charge.
Note 10: ***Devoir/Doit/Doivent*** être pris en charge lorsque cela est exigé par l'autorité d'un pays.
Note 11: ***Devoir/Doit/Doivent*** être pris en charge par des câbles actifs.
Note 12: Un VPD inclut les CT-VPD lorsqu'ils ne sont pas connectés à un chargeur. La communication PD avec un CT-VPD ***Devoir/Doit/Doivent*** être établie uniquement lorsqu'il n'est pas connecté à un chargeur.
Note 13: ***Devoir/Doit/Doivent*** être pris en charge par des produits qui prennent en charge *[USB4]*.

6.12.2 Applicabilité des messages de données

Le Tableau 6-70 répertorie les messages de données (sauf pour les commandes de VDM) qui ***Devoir/Doit/Doivent/Ne doit/doivent pas*** être transmis et reçus/qu'***Il convient d'/de/qu'/que*** transmettre et de recevoir pour une source, un destinataire, une fiche de câble ou un VPD. Les exigences relatives aux ports d'alimentation double fonction ***Devoir/Doit/Doivent*** remplacer les exigences relatives aux ports source uniques ou aux ports destinataire uniques.

Tableau 6-70 Applicabilité des messages de données

Message Type	Source	Destinataire	Alimentation double fonction	Fiche de câble SOP'	Fiche de câble SOP"	VPD[6]
Message transmis						
Source_Capabilities	N	NA	N	NA	NA	NA
Request	NA	N		NA	NA	NA
Get_Country_Info	CN[5]/O	CN[5]/O		NA	NA	NA
BIST	N[1]	N[1]		NA	NA	NA
Sink_Capabilities	NA	N	N	NA	NA	NA
Battery_Status	CN[2]	CN[2]		NA	NA	NA

Message Type	Source	Destinataire	Alimentation double fonction	Fiche de câble SOP'	Fiche de câble SOP"	VPD[6]
Alert	R	R		NA	NA	NA
Enter_USB	CN[7]/O	CN[7]/O		NA	NA	NA
Message reçu						
Source_Capabilities	NS	N	N	I	I	I
Request	N	NS		I	I	I
Get_Country_Info	CN[5]/NS	CN[5]/NS		I	I	I
BIST	N[1]	N[1]		N[1]	N[1]	N[1]
Sink_Capabilities	CN[4]	NS	CN[4]	I	I	I
Battery_Status	CN[3]/NS	CN[3]/NS		I	I	I
Alert	R/NS	R/NS		I	I	I
Enter_USB	CN[7]/O	CN[7]/O		CN[7]/O	I	I

Note 1: Pour plus d'informations sur les modes et messages BIST qui ***Devoir/Doit/Doivent*** être pris en charge, voir 5.9 et 6.4.3.
Note 2: ***Devoir/Doit/Doivent*** être pris en charge par des produits équipés de batteries.
Note 3: ***Devoir/Doit/Doivent*** être pris en charge par des produits qui prennent en charge le message *Get_Battery_Status*.
Note 4: ***Devoir/Doit/Doivent*** être pris en charge par des produits qui prennent en charge le message *Get_Sink_Cap* .
Note 5: ***Devoir/Doit/Doivent*** être pris en charge lorsque cela est exigé par l'autorité d'un pays.
Note 6: Un VPD inclut les CT-VPD lorsqu'ils ne sont pas connectés à un chargeur. La communication PD avec un CT-VPD ***Devoir/Doit/Doivent*** être établie uniquement lorsqu'il n'est pas connecté à un chargeur.
Note 7: ***Devoir/Doit/Doivent*** être pris en charge par des produits qui prennent en charge *[USB4]*.

6.12.3 Applicabilité des messages étendus

Le Tableau 6-71 répertorie les messages étendus (sauf pour les commandes de VDM étendus) qui ***Devoir/Doit/Doivent*/*Ne doit/doivent pas*** être transmis et reçus/qu'***Il convient d'/de/qu'/que*** transmettre et de recevoir pour une source, un destinataire, une fiche de câble ou un VPD. Les exigences relatives aux ports d'alimentation double fonction ***Devoir/Doit/Doivent*** remplacer les exigences relatives aux ports source uniques ou aux ports destinataire uniques.

Tableau 6-71 Applicabilité des messages étendus

Message Type	Source	Destinataire	Alimentation double fonction	Fiche de câble SOP'	Fiche de câble SOP"	VPD[1,3]
Message transmis						
Battery_Capabilities	CN[1]/NA	CN[1]/NA		NA	NA	NA
Country_Codes	CN[10]/NA	CN[10]/NA		NA	NA	NA
Country_Info	CN[10]/NA	CN[10]/NA		NA	NA	NA
Firmware_Update_Request	CN[7]/NA	CN[7]/NA		NA	NA	NA
Firmware_Update_Response	CN[7]/NA	CN[7]/NA		CN[7]/NA	O	NA
Get_Battery_Cap	R	R		NA	NA	NA
Get_Battery_Status	R	R		NA	NA	NA
Get_Manufacturer_Info	R	R		NA	NA	NA
Manufacturer_Info	R	R		R	NA	NA
PPS_Status	CN[8]/NA	NA		NA	NA	NA
Security_Request	CN[6]/NA	CN[6]/NA		NA	NA	NA
Security_Response	CN[6]/NA	CN[6]/NA		CN[6]/NA	NA	NA

Message Type	Source	Destinataire	Alimentation double fonction	Fiche de câble SOP'	Fiche de câble SOP"	VPD[1,3]
Sink_Capabilities_Extended	NA	N	N	NA	NA	NA
Source_Capabilities_Extended	R	NA	R	NA	NA	NA
Statut	R	R	R	CN[12]/NA	CN[12]/NA	NA
Message reçu						
Battery_Capabilities	CN[4]/NS	CN[4]/NS		I	I	I
Country_Codes	CN[10]/NS	CN[10]/NS		I	I	I
Country_Info	CN[10]/NS	CN[10]/NS		I	I	I
Firmware_Update_Request	CN[7]/NS	CN[7]/NS		CN[7]/I	O	I
Firmware_Update_Response	CN[7]/NS	CN[7]/NS		I	I	I
Get_Battery_Cap	CN[1]/NS	CN[1]/NS		I	I	I
Get_Battery_Status	CN[1]/NS	CN[1]/NS		I	I	I
Get_Manufacturer_Info	R/NS	R/NS		R/I	I	I
Manufacturer_Info	CN[5]/NS	CN[5]/NS		I	I	I
PPS_Status	NS	CN9/NS		I	I	I
Security_Request	CN[6]/NS	CN[6]/NS		CN[6]/I	I	I
Security_Response	CN[6]/NS	CN[6]/NS		I	I	I
Sink_Capabilities_Extended	CN[11]/NS	NS	CN[11]/NS	I	I	I
Source_Capabilities_Extended	NS	CN[2]/NS	CN[2]/NS	I	I	I
Statut	CN[3]/NS	CN[3]/NS		I	I	I

Note 1: ***Devoir/Doit/Doivent*** être pris en charge par des produits équipés de batteries.
Note 2: ***Devoir/Doit/Doivent*** être pris en charge par des produits qui peuvent transmettre le message *Get_Source_Cap_Extended*.
Note 3: ***Devoir/Doit/Doivent*** être pris en charge par des produits qui peuvent transmettre le message *Get_Status*.
Note 4: ***Devoir/Doit/Doivent*** être pris en charge par des produits qui peuvent transmettre le message *Get_Battery_Cap*.
Note 5: ***Devoir/Doit/Doivent*** être pris en charge par des produits qui peuvent transmettre le message *Get_Manufacturer_Info*.
Note 6: ***Devoir/Doit/Doivent*** être pris en charge par des produits qui prennent en charge la communication sécurisée USB définie dans *[USBTypeCAuthentication 1.0]*
Note 7: ***Devoir/Doit/Doivent*** être pris en charge par les produits qui prennent en charge la communication de mise à jour du micrologiciel USB définie dans *[USBPDFirmwareUpdate 1.0]*.
Note 8: ***Devoir/Doit/Doivent*** être pris en charge lorsque la PPS est prise en charge.
Note 9: ***Devoir/Doit/Doivent*** être pris en charge par des produits qui peuvent transmettre le *Get_PPS_Status*.
Note 10: ***Devoir/Doit/Doivent*** être pris en charge lorsque cela est exigé par l'autorité d'un pays.
Note 11: ***Devoir/Doit/Doivent*** être pris en charge par des produits qui peuvent transmettre le message *Get_Sink_Cap_Extended*.
Note 12: ***Devoir/Doit/Doivent*** être pris en charge par des câbles actifs.
Note 13: Un VPD inclut les CT-VPD lorsqu'ils ne sont pas connectés à un chargeur. La communication PD avec un CT-VPD ***Devoir/Doit/Doivent*** être établie uniquement lorsqu'il n'est pas connecté à un chargeur.

6.12.4 Applicabilité des commandes de VDM structuré

Le Tableau 6-72 répertorie les commandes de VDM structuré qui ***Devoir/Doit/Doivent*/*Ne doit/doivent pas*** être transmises et reçues/qu'***Il convient d'/de/qu'/que*** transmettre et de recevoir pour un DFP, un UFP, une fiche de câble ou un VDP. Si les VDM structurés ne sont pas pris en charge, le DFP ou l'UFP qui reçoit une commande de VDM ***Devoir/Doit/Doivent*** envoyer un message *Not_Supported* en réponse.

Tableau 6-72 Applicabilité des commandes de VDM structuré

Type de commande	DFP	UFP	Fiche de câble SOP'	Fiche de câble SOP"	VPD[4]
Demande de commande transmise					
Discover Identity	CN[1,6]/R	R[2]	NA	NA	NA
Discover SVIDs	CN[1]/O	O	NA	NA	NA
Discover Modes	CN[1]/O	O	NA	NA	NA
Enter Mode	CN[1]/NA	NA	NA	NA	NA
Exit Mode	CN[1]/NA	NA	NA	NA	NA
Attention	O	O	NA	NA	NA
Demande de commande reçue/Réponse de commande transmise					
Discover Identity	CN[5,6]/R/NK[3]	CN[1,6]/R/NK[3]	N	I	N
Discover SVIDs	O/NK[3]	CN[1]/NK[3]	CN[1]/NK	I	NK
Discover Modes	O/NK[3]	CN[1]/NK[3]	CN[1]/NK	I	NK
Enter Mode	NK[3]	CN[1]/NK[3]	CN[1]/NK	O	NK
Exit Mode	NK[3]	CN[1]/NK[3]	CN[1]/NK	O	NK
Attention	O/I[3]	O/I[3]	I	I	I

Note 1: ***Devoir/Doit/Doivent*** être pris en charge lorsque le fonctionnement modal est pris en charge.
Note 2: ***Pouvoir/Peut/Peuvent*** être transmis par un UFP ou une source au cours de la découverte (voir 6.4.4.3.1 et 8.3.3.24.3).
Note 3: Si les VDM structurés ne sont pas pris en charge, le DFP ou l'UFP qui reçoit une commande de VDM ***Devoir/Doit/Doivent*** envoyer un message ***Not_Supported*** en réponse.
Note 4: Un VPD inclut les CT-VPD lorsqu'ils ne sont pas connectés à un chargeur. La communication PD avec un CT-VPD ***Devoir/Doit/Doivent*** être établie uniquement lorsqu'il n'est pas connecté à un chargeur.
Note 5: ***Devoir/Doit/Doivent*** être pris en charge par des produits avec plusieurs DFP.
Note 6: ***Devoir/Doit/Doivent*** être pris en charge par des produits qui prennent en charge ***[USB4]***.

6.12.5 Applicabilité des signaux de réinitialisation

Le Tableau 6-73 répertorie les signaux de réinitialisation qui ***Devoir/Doit/Doivent/Ne doit/doivent pas*** être transmis et reçus/qu'***Il convient d'/de/qu'/que*** transmettre et de recevoir pour un DFP, un UFP ou une fiche de câble.

Tableau 6-73 Applicabilité des signaux de réinitialisation

Type de signal	DFP	UFP	Fiche de câble SOP'	Cable Plug SOP''	VPD[2]
Message/signal transmis					
Soft_Reset	N	N	NA	NA	NA
Hard Reset	N	N	NA	NA	NA
Cable Reset	CN[1]	CN[1]	NA	NA	NA
Message/signal reçu					
Soft_Reset	N	N	N	N	N
Hard Reset	N	N	N	N	N
Cable Reset	DR	DR	N	N	N

Note 1: ***Devoir/Doit/Doivent*** être pris en charge lorsque la transmission de paquets SOP' est prise en charge et que le port peut fournir VCONN.
Note 2: Un VPD inclut les CT-VPD lorsqu'ils ne sont pas connectés à un chargeur. La communication PD avec un CT-VPD ***Devoir/Doit/Doivent*** être établie uniquement lorsqu'il n'est pas connecté à un chargeur.

6.12.6 Applicabilité des signaux de permutation rapide des rôles

Le Tableau 6-73 répertorie les signaux de permutation rapide des rôles qui ***Devoir/Doit/Doivent/Ne doit/doivent pas*** être transmis et reçus/qu'***Il convient d'/de/qu'/que*** transmettre et de recevoir pour une source ou un destinataire.

Tableau 6-74 Applicabilité des signaux de permutation rapide des rôles

Type de commande	Source	Destinataire	Alimentation double fonction
Message/signal transmis			
Permutation rapide des rôles	NA	NA	R
Message/signal reçu			
Permutation rapide des rôles	NA	NA	R

6.13 Paramètres de valeur

Le Tableau 6-75 contient les paramètres de valeur utilisés dans la présente section.

Tableau 6-75 Paramètres de valeur

Paramètre	Description	Valeur	Unité	Référence
MaxExtendedMsgLen	Longueur maximale d'un message étendu, exprimée dans le champ ***Data Size***.	260	Octet	Section 6.4.8
MaxExtendedMsgChunkLen		26	Octet	Section 6.4.8
MaxExtendedMsgLegacyLen		26	Octet	Section 6.4.8

7. Alimentation

7.1 Exigences relatives à la source

7.1.1 Aspects comportementaux

Une source d'alimentation électrique par port USB présente les comportements suivants:

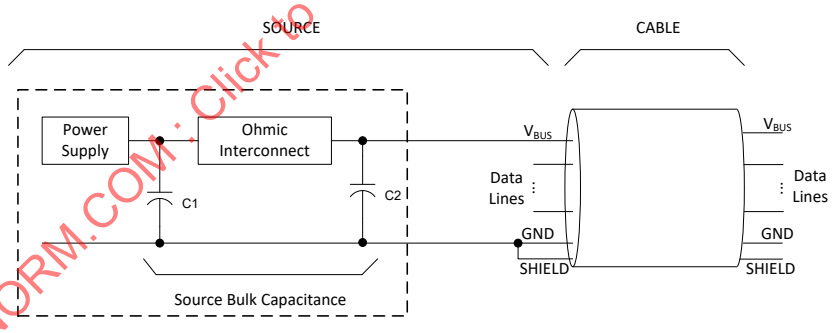
- ***Devoir/Doit/Doivent*** fournir la tension et le courant par défaut ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** (USB Type-C®) ou ***[USBBC 1.2]*** à V_{BUS} en l'absence de contrat (fonctionnement USB par défaut);
- ***Devoir/Doit/Doivent*** suivre les exigences spécifiées en 7.1.5 lorsqu'un signal ***Hard Reset*** est reçu;
- ***Devoir/Doit/Doivent*** contrôler les transitions de tension de V_{BUS} liées aux exigences de sous-dépassement, de surdépassement et de délai de transition.

7.1.2 Capacité de masse de la source

La capacité de masse de la source ***Ne doit/doivent pas*** être placée entre l'impédance d'isolation de l'émetteur-récepteur et l'embase USB. La capacité de masse de la source comprend C1 et C2, comme représenté à la Figure 7-1. L'interconnexion ohmique peut être constituée de pistes de CCI pour les dispositifs de distribution ou de commutation de puissance. La capacité peut être un condensateur unique, une batterie de condensateurs ou une capacité distribuée. Si l'alimentation est partagée entre plusieurs ports, la capacité de masse est définie par ***cSrcBulkShared***. Si l'alimentation est dédiée à un seul port, la capacité de masse minimale est définie par ***cSrcBulk***.

Il est admis que la capacité de masse de la source soit modifiée pour un nouveau niveau de puissance négocié. La modification de capacité ***Devoir/Doit/Doivent*** intervenir avant que la source ne soit prête à fonctionner au nouveau niveau de puissance. Au cours d'une permutation des rôles d'alimentation, la source par défaut ***Devoir/Doit/Doivent*** passer en veille de permutation avant de fonctionner en tant que nouveau destinataire. Toute modification de la capacité de masse exigée pour achever la permutation des rôles d'alimentation ***Devoir/Doit/Doivent*** intervenir au cours de la veille de permutation.

Figure 7-1 Placement de la capacité de masse de la source



Anglais	Français
CABLE	CÂBLE
Power Supply	Alimentation
Ohmic Interconnect	Interconnexion ohmique
Source Bulk Capacitance	Capacité de masse de la source
Data Lines	Lignes de données
GND	TERRE
SHIELD	BLINDAGE

7.1.3 Types de sources

Conformément aux objets de données de puissance décrits en 6.4.1, les types d'alimentation disponibles en tant que sources dans un système d'alimentation électrique par port USB sont les suivants:

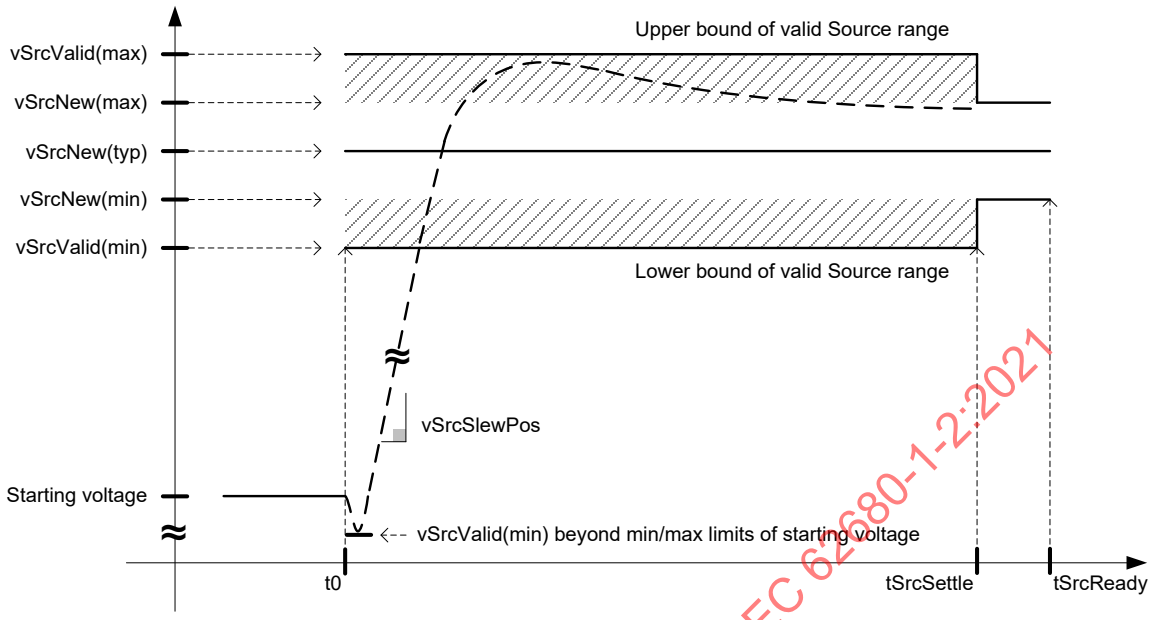
- le PDO d'alimentation fixe expose des alimentations de tension fixe bien régulées. Les sources ***Devoir/Doit/Doivent*** prendre en charge au moins une alimentation fixe en mesure de fournir ***vSafe5V***. La tension de sortie d'une alimentation fixe ***Devoir/Doit/Doivent*** rester dans la plage définie par la tolérance relative ***vSrcNew*** et la bande absolue ***vSrcValid***, répertoriées dans le Tableau 7-22 et décrites en 7.1.8;
- le PDO Variable Supply (non-Battery) expose des sources très mal régulées. La tension de sortie d'une alimentation variable (batterie exceptée) ***Devoir/Doit/Doivent*** rester entre la tension de sortie maximale absolue et la tension de sortie minimale absolue, exposées dans le PDO Variable Supply;
- le PDO Battery Supply expose les batteries qui peuvent être connectées directement en tant que source à V_{BUS}. La tension de sortie d'une alimentation par batterie ***Devoir/Doit/Doivent*** rester entre la tension de sortie maximale absolue et la tension de sortie minimale absolue, exposées dans le PDO Battery Supply;
- le PDO augmenté (APDO) Programmable Power Supply (PPS) dévoile une source dont la tension de sortie peut être ajustée par programmation sur une plage définie. La tension de sortie de l'alimentation électrique programmable ***Devoir/Doit/Doivent*** rester dans une plage définie par la tolérance relative ***vPpsNew*** et la bande absolue ***vPpsValid***.

7.1.4 Transitions de source

7.1.4.1 Transitions de tension positives de l'alimentation fixe

La source ***Devoir/Doit/Doivent*** faire passer V_{BUS} de la tension de départ à la nouvelle tension plus élevée de manière contrôlée. La nouvelle tension négociée (par exemple, 5 V, 9 V, 15 V ou 20 V) définit la valeur nominale de ***vSrcNew***. Au cours de la transition positive, la source ***Devoir/Doit/Doivent*** être en mesure de fournir l'alimentation de veille au destinataire ainsi que le courant transitoire pour modifier la capacité de masse totale sur V_{BUS}. La vitesse de balayage de la transition positive ***Ne doit/doivent pas*** dépasser ***vSrcSlewPos***. La tension de sortie d'une source en transition ***Devoir/Doit/Doivent*** être établie à ***vSrcNew*** dans un délai ***tSrcSettle***. La source ***Devoir/Doit/Doivent*** être en mesure de fournir le niveau de puissance négocié à la nouvelle tension dans un délai ***tSrcReady***. La transition de tension positive ***Devoir/Doit/Doivent*** rester monotone tant que la tension de transition est inférieure à ***vSrcValid*** min, et elle ***Devoir/Doit/Doivent*** rester dans la plage ***vSrcValid*** au croisement avec ***vSrcValid*** min, comme le montre la Figure 7-2. Sur la Figure 7-2, l'heure de début t0 lance ***tSrcTransition*** après réception du dernier bit de l'***EOP*** du message ***GoodCRC*** par la source.

Figure 7-2 Enveloppe des transitions de tension positives



Anglais	Français
Starting voltage	Tension de départ
Upper bound of valid Source range	Limite supérieure de la plage source valide
Lower bound of valid Source range	Limite inférieure de la plage source valide
vSrcValid(min) beyond min/max limits of starting voltage	vSrcValid(min) au-delà des limites min. et max. de la tension de départ

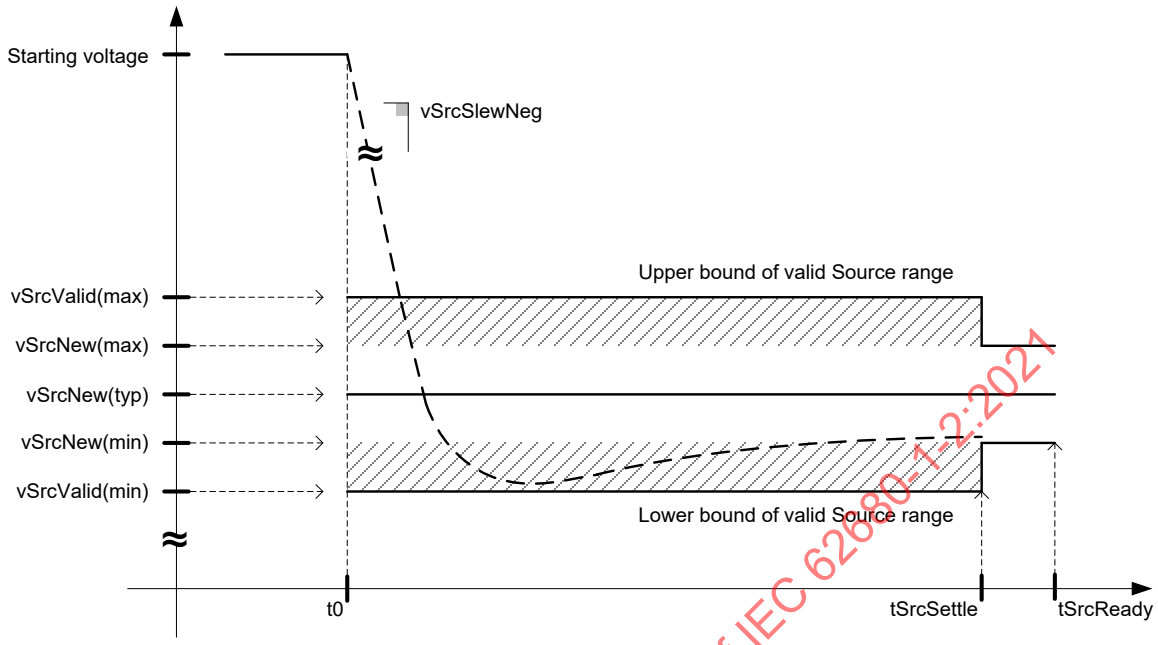
Au début de la transition de tension positive, le niveau de tension V_{BUS} ***Ne doit/doivent pas*** faire tomber ***vSrcValid*** min sous ***vSrcNew*** (si le niveau de tension V_{BUS} initial n'est pas ***vSafe5V***) ni sous ***vSafe5V***, selon le cas.

La Section 7.1.14 répertorie les transitions exemptes de la limite ***vSrcSlewPos***.

7.1.4.2 Transitions de tension négatives de l'alimentation fixe

Les transitions de tension négatives sont définies à la Figure 7-3 et sont spécifiées de manière semblable aux transitions de tension positives. La Figure 7-3 ne s'applique pas aux transitions ***vSafe0V***. La vitesse de balayage de la transition négative ***Ne doit/doivent pas*** dépasser ***vSrcSlewNeg***. La transition de tension négative ***Devoir/Doit/Doivent*** rester monotone tant que la tension de transition est supérieure à ***vSrcValid*** max et elle ***Devoir/Doit/Doivent*** rester dans la plage ***vSrcValid*** au croisement avec ***vSrcValid*** max, comme le montre la Figure 7-3 . Sur la Figure 7-3 , l'heure de début t0 lance ***tSrcTransition*** après réception du dernier bit de l'***EOP*** du message ***GoodCRC*** par la source.

Figure 7-3 Enveloppe des transitions de tension négatives



Anglais	Français
Starting voltage	Tension de départ
Upper bound of valid Source range	Limite supérieure de la plage source valide
Lower bound of valid Source range	Limite inférieure de la plage source valide

Si la tension nouvellement négociée est ***vSafe5V***, les limites de ***vSrcValid*** ***Devoir/Doit/Doivent*** déterminer la fenêtre de transition, et la source en transition ***Devoir/Doit/Doivent*** être établie dans les limites de ***vSafe5V*** dans un délai ***tSrcSettle***.

La Section 7.1.14 répertorie les transitions exemptes de la limite ***vSrcSlewNeg***.

7.1.4.3 Transitions de tension de l'alimentation électrique programmable

L'alimentation programmable (PPS) ***Devoir/Doit/Doivent*** faire passer V_{BUS} sur la plage de tensions définie de manière contrôlée. La valeur de la tension de sortie du RDO programmable définit la valeur nominale de la tension de sortie PPS après une variation de tension et ***Devoir/Doit/Doivent*** se stabiliser dans les limites définies par ***vPpsNew*** au cours de ***TPpsSrcTransSmall*** pour des paliers inférieurs ou égaux à ***VPpsSmallStep***, ou sinon, dans les limites définies par ***vPpsNew*** au cours de ***TPpsSrcTransLarge***, mais seulement dans le cas où l'alimentation électrique programmable n'est pas en mode CL. Tout sur-dépassement au-delà de ***vPpsNew*** ***Ne doit/doivent pas*** dépasser ***vPpsValid***, quel que soit le moment. Tout sous-dépassement au-delà de ***vPpsNew*** ***Ne doit/doivent pas*** dépasser ***vPpsValid*** pour des courants qui n'entraînent pas de mode CL. La tension de sortie de la PPS ***Pouvoir/Peut/Peuvent*** varier de manière graduelle ou linéaire, et la vitesse de balayage de l'un ou l'autre des types de variation ***Ne doit/doivent pas*** dépasser ***vPpsSlewPos*** pour les augmentations de tension ou ***vPpsSlewNeg*** pour les baisses de tension. La tension assignée demandée pour toutes les modifications de tension linéaires ***Devoir/Doit/Doivent*** être égale à un nombre entier de modifications dont le poids en octets est faible. Une modification d'octet de poids faible de la tension de sortie de PPS est définie comme ***vPpsStep***. Une PPS ***Devoir/Doit/Doivent*** être capable de fournir le niveau de courant négocié à mesure qu'elle modifie sa tension de sortie au niveau demandé. Toute augmentation de la tension PPS ***Devoir/Doit/Doivent*** entraîner une tension supérieure à la tension de sortie PPS précédente. De même, toute baisse de la tension PPS ***Devoir/Doit/Doivent*** entraîner une tension inférieure à la tension de sortie PPS précédente.

Puisqu'un destinataire peut appeler du courant jusqu'au niveau de courant de l'APDO négocié en cas de palier de tension, la tension peut ne pas augmenter jusqu'au niveau exigé du fait que l'alimentation

électrique fonctionne en mode CL. De même, puisqu'un destinataire peut avoir une batterie connectée à VBUS, la tension peut ne pas descendre jusqu'au niveau exigé du fait que la tension de la batterie est supérieure à la valeur de tension de sortie de consigne à laquelle la source est passée. Que la source vérifie ou non la tension sur VBUS, un PS_RDY n'est jamais envoyé pour déterminer lorsque son alimentation électrique est prête.

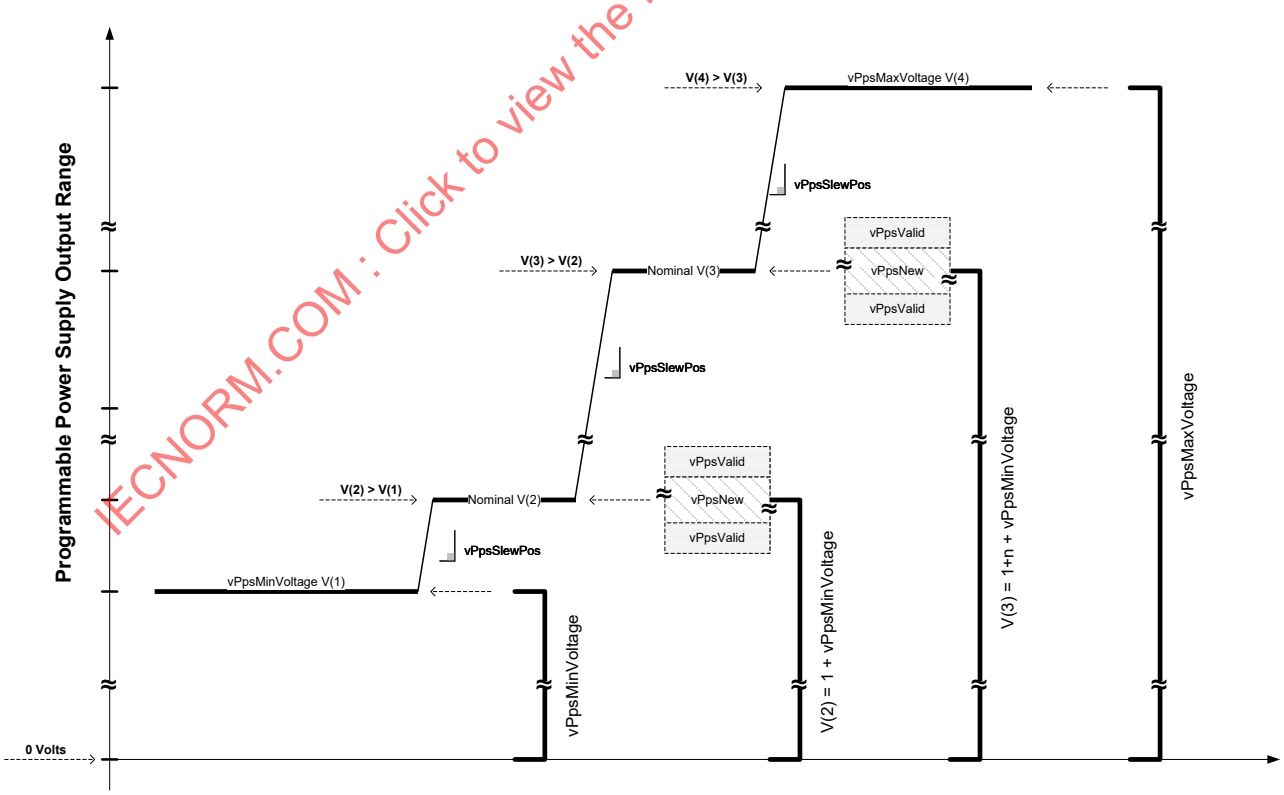
Lorsque la tension PPS augmente ou diminue, un message ***PS_RDY*** ***Devoir/Doit/Doivent*** être envoyé au cours de:

- ***TPpsSrcTransLarge*** après le dernier bit du message ***GoodCRC*** suivant le message ***Accept*** pour des paliers supérieurs à ***VPpsSmallStep.***
- ***TPpsSrcTransSmall*** après le dernier bit du message ***GoodCRC*** suivant le message ***Accept*** pour des paliers inférieurs ou égaux à ***VPpsSmallStep***, à condition que la tension sur V_{BUS} ait atteint ***vPpsNew*** ou que l'alimentation électrique soit en mode CL.

Lorsque ***vPpsNew*** est inférieure à la tension de la batterie ou que l'alimentation principale de la source est coupée, le destinataire ***Devoir/Doit/Doivent*** immédiatement déconnecter sa batterie de V_{BUS}. Dans de telles situations, le courant de sortie peut inverser la polarité et le destinataire n'est pas autorisé à fournir du courant (voir 7.2.1 et 7.2.9).

La Figure 7-4 et la Figure 7-5 ci-dessous représentent le comportement de la tension de sortie d'une alimentation programmable en réponse à des demandes de variation de tension positives et négatives lors du fonctionnement avec une PPS. Les paramètres ***vPpsMinVoltage*** et ***vPpsMaxVoltage*** définissent respectivement les limites inférieure et supérieure de la plage PPS (voir Tableau 10-8 pour les modifications exigées). ***vPpsMinVoltage*** correspond au champ Minimum Voltage de l'APDO PPS, et ***vPpsMaxVoltage*** correspond au champ Maximum Voltage de l'APDO PPS. Si le destinataire négocie un nouvel APDO PPS, alors la transition entre les deux APDO PPS ***Devoir/Doit/Doivent*** se dérouler comme décrit en 7.3.18.

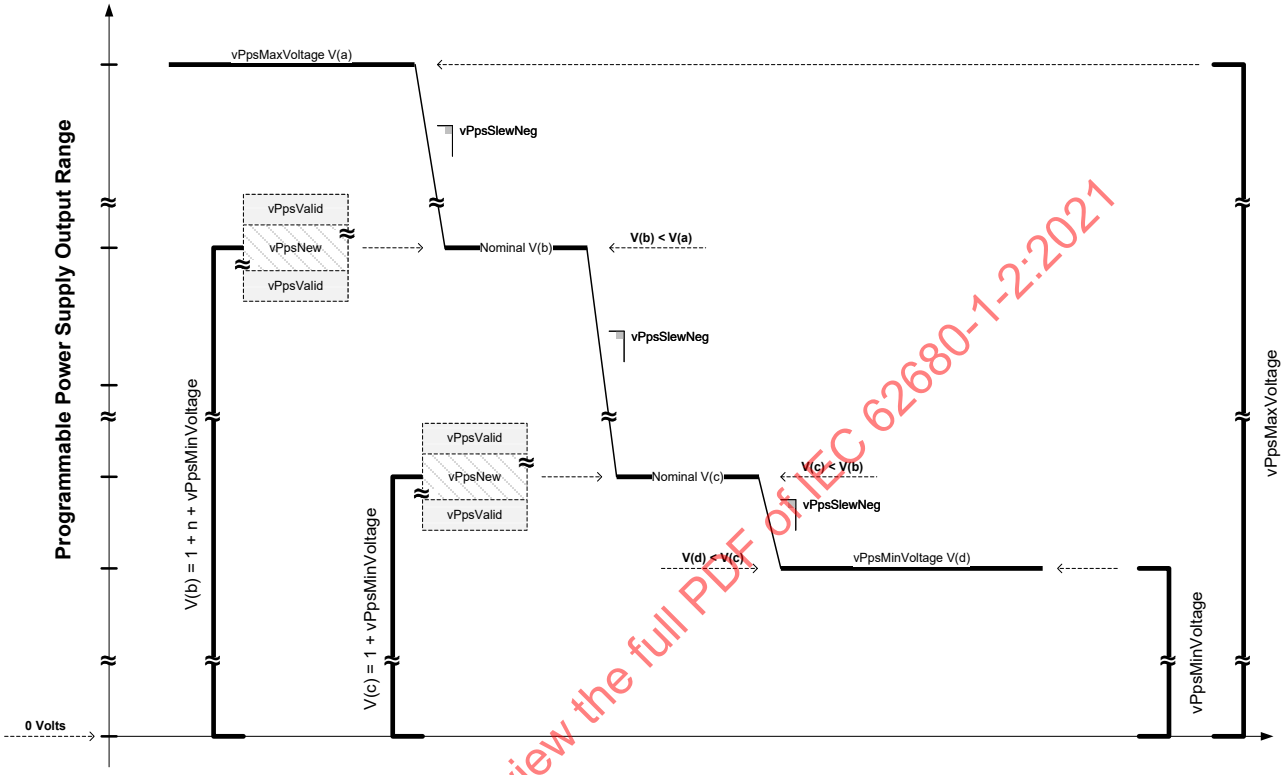
Figure 7-4 Transitions de tension positives de la PPS



Anglais	Français
Programmable Power Supply Output Range	Plage de sortie de l'alimentation programmable

0 Volts	0 volt

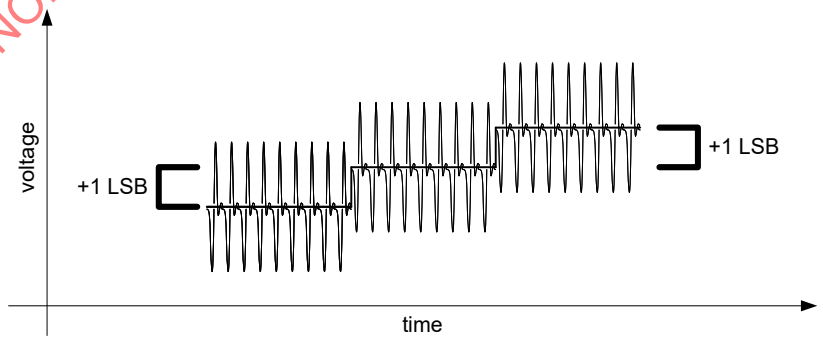
Figure 7-5 Transitions de tension négatives de la PPS



Anglais	Français
Programmable Power Supply Output Range	Plage de sortie de l'alimentation programmable
0 Volts	0 volt

Il est prévu que l'ondulation de la tension de sortie PPS dépasse l'amplitude d'un ou plusieurs octets de poids faible, comme le montre la Figure 7-6.

Figure 7-6 Ondulation prévue de la PPS relative à un octet de poids faible



Anglais	Français
voltage	tension
time	temps

Le paragraphe 7.1.14 répertorie les transitions exemptes des limites ***vPpsSlewNeg*** et ***vPpsSlewPos***.

7.1.4.4 Limite du courant de l'alimentation électrique programmable

L'alimentation programmable ***Devoir/Doit/Doivent*** limiter son courant de sortie à la valeur du courant de fonctionnement du RDO programmable lorsque le destinataire tente de prélever plus de courant que le niveau du courant de sortie. La taille du palier de programmation du courant de sortie est ***IPpsCLStep***. Toutes les modifications de programmation du courant de fonctionnement ***Devoir/Doit/Doivent*** être établies sur la nouvelle valeur du courant de fonctionnement dans un délai ***TPpsCLProgram*** Settle. La précision de la régulation du courant de fonctionnement de la PPS lors de la limitation de courant est définie par ***IPpsCLNew***. Le niveau minimal programmable de la limitation de courant est ***IPpsCLMin***. Une source qui prend en charge la PPS ***Devoir/Doit/Doivent*** prendre en charge la programmabilité de la limitation de courant entre ***IPpsCLMin*** et la valeur du courant maximal de l'APDO PPS.

La réponse d'une PPS à une variation de charge dépend du mode de fonctionnement de la PPS et de l'amplitude de la variation de charge. Ces dépendances entraînent l'une des quatre réponses possibles d'une PPS à une variation de charge. Elles se distinguent par la valeur du fanion PPS Status OMF avant et après la variation de charge:

- Si le fanion PPS Status OMF est effacé avant et après la variation de charge, la PPS répond seulement en maintenant la tension de sortie. La tension de sortie PPS doit rester dans la plage ***vPpsValid***. La réponse de la PPS à la variation de charge ***Devoir/Doit/Doivent*** rester dans la plage de tolérance ***vPpsNew*** pendant la durée ***tPpsTransient***. Le fanion Operating Mode ***Devoir/Doit/Doivent*** rester effacé pendant la réponse de la PPS à la variation de charge.
- Si le fanion PPS Status OMF est effacé avant la variation de charge et défini après la variation de charge, la PPS répond en réduisant sa tension de sortie pour limiter le courant de sortie PPS. Le courant de sortie PPS ***Devoir/Doit/Doivent*** rester dans la plage ***IPpsCVCLTransient*** après avoir atteint la plage ***IPpsCVCLTransient***. La réponse de la PPS à la variation de charge ***Devoir/Doit/Doivent*** rester dans la plage de tolérance ***IPpsCLNew*** pendant la durée ***TPpsCVCLTransient***. Le fanion Operating Mode ***Devoir/Doit/Doivent*** être défini lorsque la réponse de la PPS à la variation de charge se stabilise.
- Si le fanion PPS Status OMF est défini avant et après la variation de charge, la PPS répond en ajustant sa tension de sortie pour maintenir le courant de sortie. Le courant de sortie PPS ***Devoir/Doit/Doivent*** rester dans la plage ***IPpsCLTransient***. La réponse de la PPS à la variation de charge ***Devoir/Doit/Doivent*** rester dans la plage de tolérance ***IPpsCLNew*** pendant la durée ***TPpsCLSettle***. Le fanion Operating Mode ***Devoir/Doit/Doivent*** rester défini pendant la réponse de la PPS à la variation de charge.
- Si le fanion PPS Status OMF est défini avant la variation de charge et effacé après la variation de charge, la PPS répond à la variation de charge en augmentant sa tension de sortie ***vPpsNew*** et en la maintenant. La tension de sortie PPS ***Devoir/Doit/Doivent*** rester dans la plage ***TPpsCLCVTransient***. La réponse de la PPS à la variation de charge ***Devoir/Doit/Doivent*** rester dans la plage de tolérance ***vPpsNew*** pendant la durée ***TPpsCLCVTransient***. Le fanion Operating Mode ***Devoir/Doit/Doivent*** rester effacé lorsque la réponse de la PPS à la variation de charge se stabilise.

La PPS doit maintenir sa tension de sortie à la valeur demandée dans le RDO PPS pour toutes les conditions de charge statiques et dynamiques pendant le fonctionnement en Current Limit. En réponse à toute condition de charge statique ou dynamique pendant le fonctionnement en Current Limit qui provoque une chute de la tension de sortie PPS au-dessous de ***vPpsShutdown***, la source ***Pouvoir/Peut/Peuvent*** envoyer un signal ***Hard Reset*** et ***Devoir/Doit/Doivent*** décharger V_{BUS} à ***vSafe0V*** avant de reprendre son fonctionnement par défaut à ***vSafe5V***.

Lorsque le destinataire tente d'appeler plus de courant que le courant de fonctionnement du RDO, la source ***Devoir/Doit/Doivent*** limiter son courant de sortie. Le courant disponible à partir de la source en mode Current Limit doit être conforme à ***IPpsCLNew*** plus ***IPpsCLOperating***. Le destinataire ***Peut ne pas/Peuvent ne pas*** réduire sa demande de courant de fonctionnement dans le RDO lorsque le fanion PPS Status OM est défini.

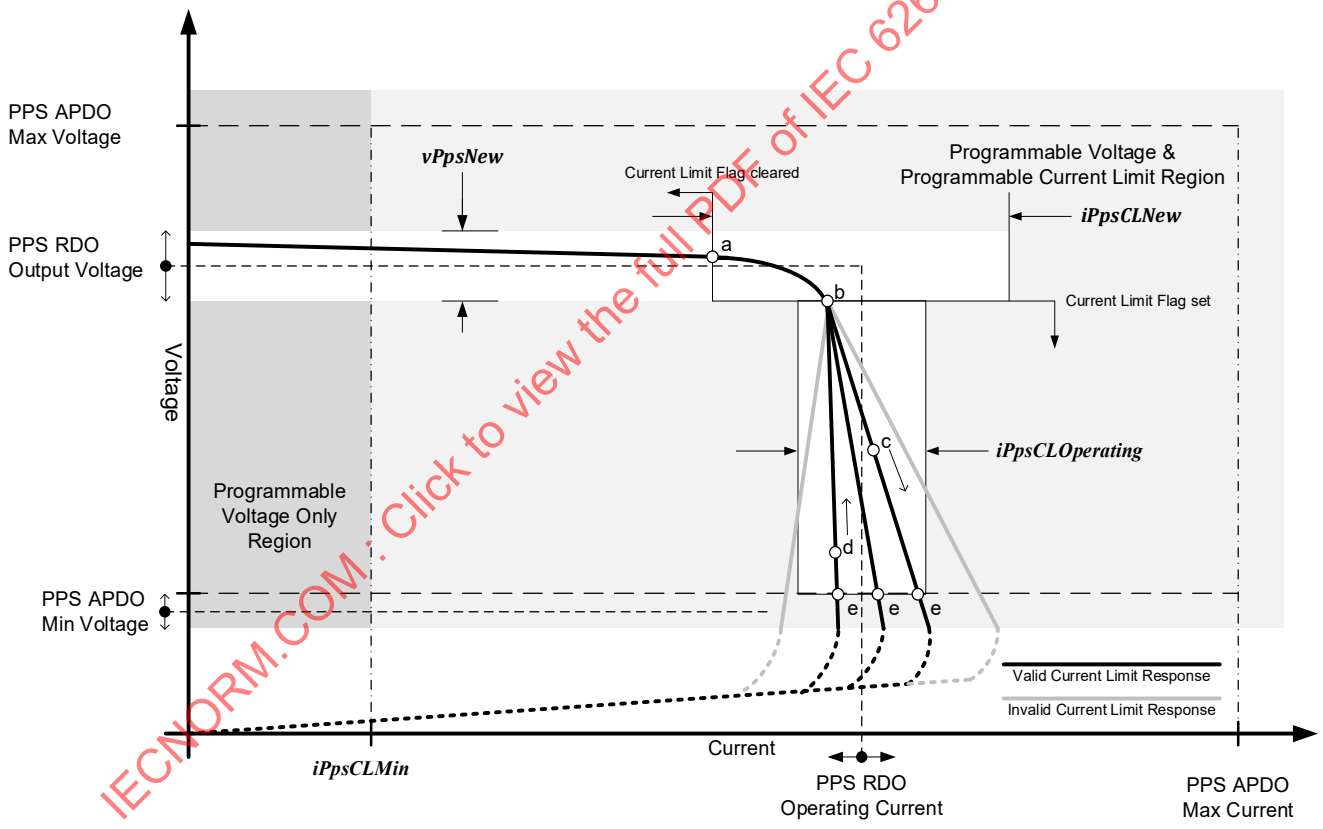
La limitation du courant ***Devoir/Doit/Doivent*** être effectuée par la source PPS. Les destinataires qui dépendent de la limitation de courant PPS ***Devoir/Doit/Doivent*** satisfaire aux exigences données en 7.2.9. La source ***Ne doit/doivent pas*** arrêter ou autrement interrompre la puissance de sortie disponible en

mode Current Limit, sauf si un autre mécanisme de protection décrit en 7.1.7 est engagé pour protéger la source d'éventuels dommages.

La relation entre la tension de sortie programmable PPS et la limitation de courant programmable PPS ***Devoir/Doit/Doivent*** être conforme à la Figure 7-7. La transition entre le mode Constant Voltage et le mode Current Limit a lieu entre les points *a* et *b*. Le fanion PPS Status OM doit être défini ou effacé dans cette région. En mode Current Limit, en cas de modification de la résistance de charge, le courant de sortie de la source reste dans les limites de ***IPpsCLOperating***, qui sont déterminées par le point b (une valeur mesurée). A mesure que la résistance de charge diminue, il convient que le courant de sortie reste le même ou qu'il augmente légèrement; à mesure que la résistance de charge augmente, il convient que le courant de sortie reste le même ou qu'il diminue légèrement. Le niveau d'augmentation ou de diminution admissible ne doit pas dépasser ***iPpsCLTolerance*** par rapport à une ligne droite tracée entre les points b et e, comme représenté à la Figure 7-8.

Le comportement adéquat est représenté par le point *c*. De même, comme la résistance de charge augmente, le courant de sortie de la source ***Ne doit/doivent pas*** augmenter. Le comportement adéquat est représenté par le point *d*.

Figure 7-7 Tension et limite de courant programmables PPS



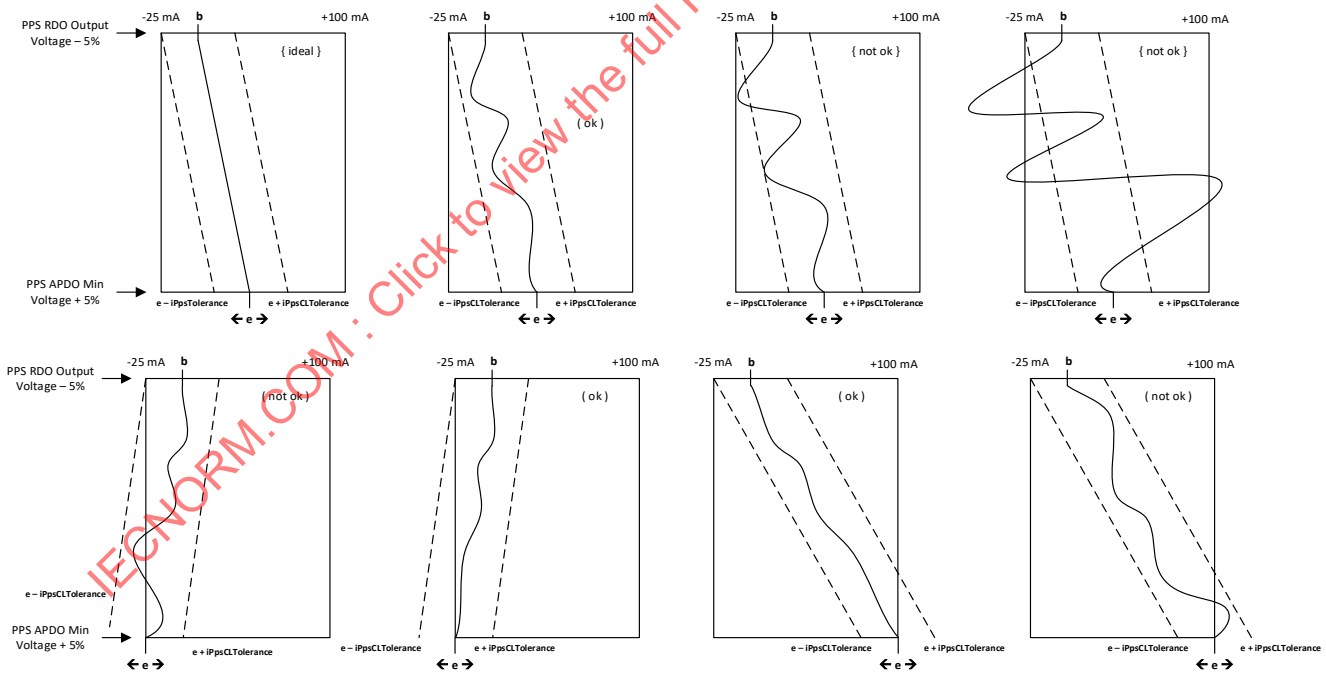
Anglais	Français
PPS APDO Max Voltage	Tension max. de l'APDO PPS
PPS RDO Output Voltage	Tension de sortie du RDO PPS
PPS APDO Min Voltage	Tension min. de l'APDO PPS
Programmable Voltage Only Region	Domaine de tension programmable uniquement
Programmable Voltage & Programmable Current Limit Region	Tension programmable et domaine de limite de courant programmable
Current Limit Flag cleared	Fanion Current Limit effacé

Current Limit Flag set	Fanion Current Limit défini
Valid Current Limit Response	Réponse Current Limit valide
Invalid Current Limit Response	Réponse Current Limit invalide
Voltage	Tension
Current	Courant
PPS RDO Operating Current	Courant de fonctionnement du RDO PPS
PPS APDO Max Current	Courant max. de l'APDO PPS

Notes:

- Le point *a* représente l'entrée dans la région de transition entre le mode Constant Voltage et le mode Current Limit.
- Le point *b* représente la sortie de la région de transition entre le mode Constant Voltage et le mode Current Limit.
- Le point *b* correspond au début de l'augmentation admissible du courant jusqu'à ***IPpsCLOperating***.
- Le point *c* représente le comportement lorsque la résistance de charge diminue en mode Current Limit. Voir Tableau 7-22 pour la variation admise du courant de fonctionnement (***IPpsCLOperating***) pendant ce comportement.
- Le point *d* représente le comportement lorsque la résistance de charge augmente en mode Current Limit. Voir Tableau 7-22 pour la variation admise du courant de fonctionnement (***IPpsCLOperating***) pendant ce comportement.
- Le point e représente la sortie de la région ***IPpsCLOperating***.

Figure 7-8 iPpsCLOperatingDetail



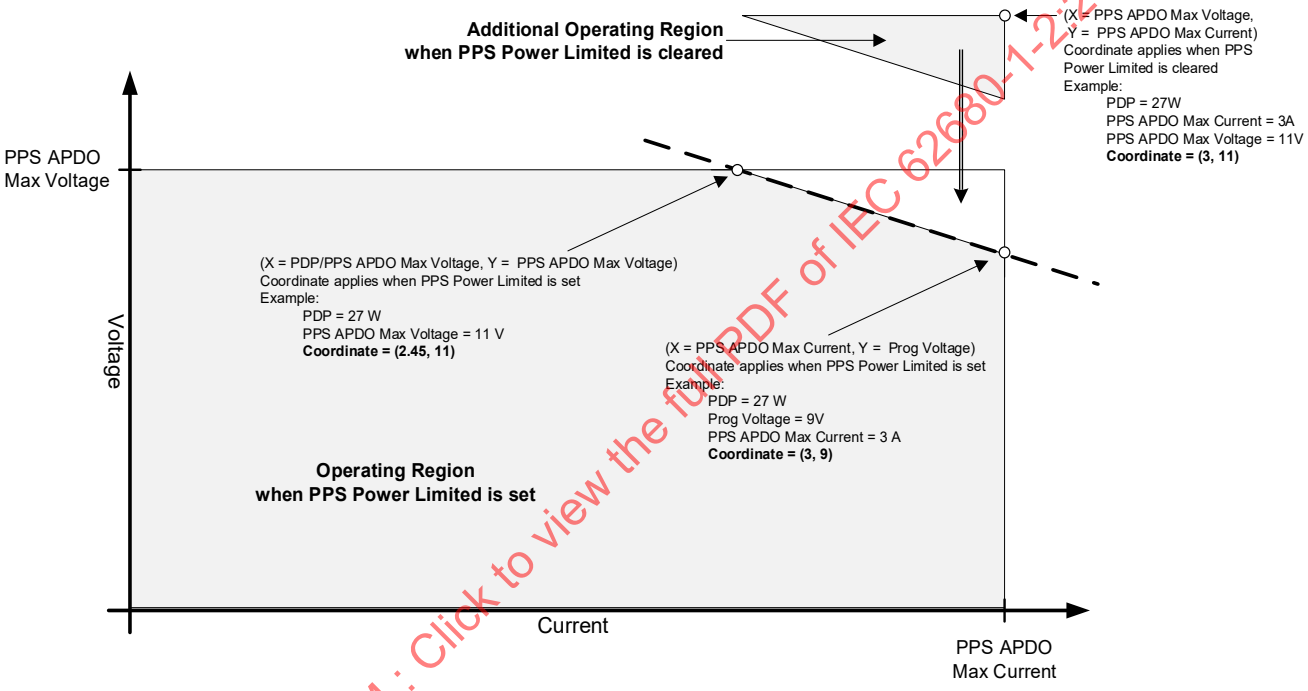
Anglais	Français
PPS RDO Output Voltage	Tension de sortie du RDO PPS
ideal	idéal
ok	correct
not ok	non correct
PPS APDO Min Voltage	Tension min. de l'APDO PPS

7.1.4.4.1 Mode Constant Power

En mode Constant Power (lorsque le bit PPS Power Limited est défini), la source ***Devoir/Doit/Doivent*** limiter son courant de sortie, de sorte que le produit du courant de sortie par la tension de sortie ne dépasse pas la PDP assignée de la source. Les destinataires ***Peut ne pas/Peuvent ne pas*** limiter leur demande de courant de fonctionnement dans le RDO, mais ***Devoir/Doit/Doivent*** satisfaire aux exigences du 7.2.9.

La relation entre la tension de sortie programmable PPS et la limite de courant programmable PPS Doit être conforme à la Figure 7-9.

Figure 7-9 Tension et limite de courant programmables PPS



Anglais	Français
PPS APDO Max Voltage	Tension max. de l 'APDO PPS
Voltage	Tension
Additional Operation Region when PPS Power Limited is cleared	Domaine de fonctionnement supplémentaire lorsque PPS Power Limited est effacé
(X= PPS APDO Max Voltage Y = PPS APDO Max Current)	(X = Tension max. de l 'APDO PPS Y= Courant max. de l 'APDO PPS)
Coordinate applies when PPS Power Limited is cleared	Les coordonnées s'appliquent lorsque PPS Power Limited est effacé
Coordinate applies when PPS Power Limited is set	Les coordonnées s'appliquent lorsque PPS Power Limited est défini

Example: PDP = 27W PPS APDO Max Current = 3A PPS APDO Max Voltage = 11V Coordinate = (3, 11)	Exemple: PDP = 27 W Courant max. de l 'APDO PPS = 3 A Tension max. de l 'APDO PPS = 11V Coordonnées = (3, 11)
(X= PPS APDO Max Current Y = Prog Voltage)	(X = Courant max. de l 'APDO PPS Y = Tension prog.)
Example: PDP = 27 W PPS APDO Max Voltage = 11 V Coordinate = (2.45, 11)	Exemple: PDP = 27 W Tension max. de l 'APDO PPS = 11 V Coordonnées = (2,45, 11)
Example: PDP = 27 W Prog Voltage = 9V PPS APDO Max Current = 3 A Coordinate = (3, 9)	Exemple: PDP = 27 W Tension prog. = 9 V Courant max. de l 'APDO PPS = 3 A Coordonnées = (3, 9)
Operating Region when PPS Power Limited is set	Domaine de fonctionnement lorsque PPS Power Limited est défini
Current	Courant
PPS APDO Max Current	Courant max. de l 'APDO PPS

7.1.5 Réponse aux réinitialisations matérielles

Un signal ***Hard Reset*** indique qu'une défaillance de communication s'est produite et que la source ***Devoir/Doit/Doivent*** arrêter de conduire VCONN, ***Devoir/Doit/Doivent*** supprimer la Rp de la broche VCONN et ***Devoir/Doit/Doivent*** amener V_{BUS} à ***vSafe0V***, comme indiqué à la Figure 7-10. La connexion USB ***Pouvoir/Peut/Peuvent*** être réinitialisée lors d'une réinitialisation matérielle, car la tension V_{BUS} est inférieure à ***vSafe5V*** pendant une période prolongée. Après avoir établi la condition de tension ***vSafe0V*** sur V_{BUS}, la source ***Devoir/Doit/Doivent*** attendre ***tSrcRecover*** avant de réappliquer VCONN et de restaurer V_{BUS} sur ***vSafe5V***. Une source ***Devoir/Doit/Doivent*** être conforme à la temporisation VCONN spécifiée dans ***[USB Type-C 2.0]***.

Pendant et après une réinitialisation matérielle, un dispositif fonctionne comme suit:

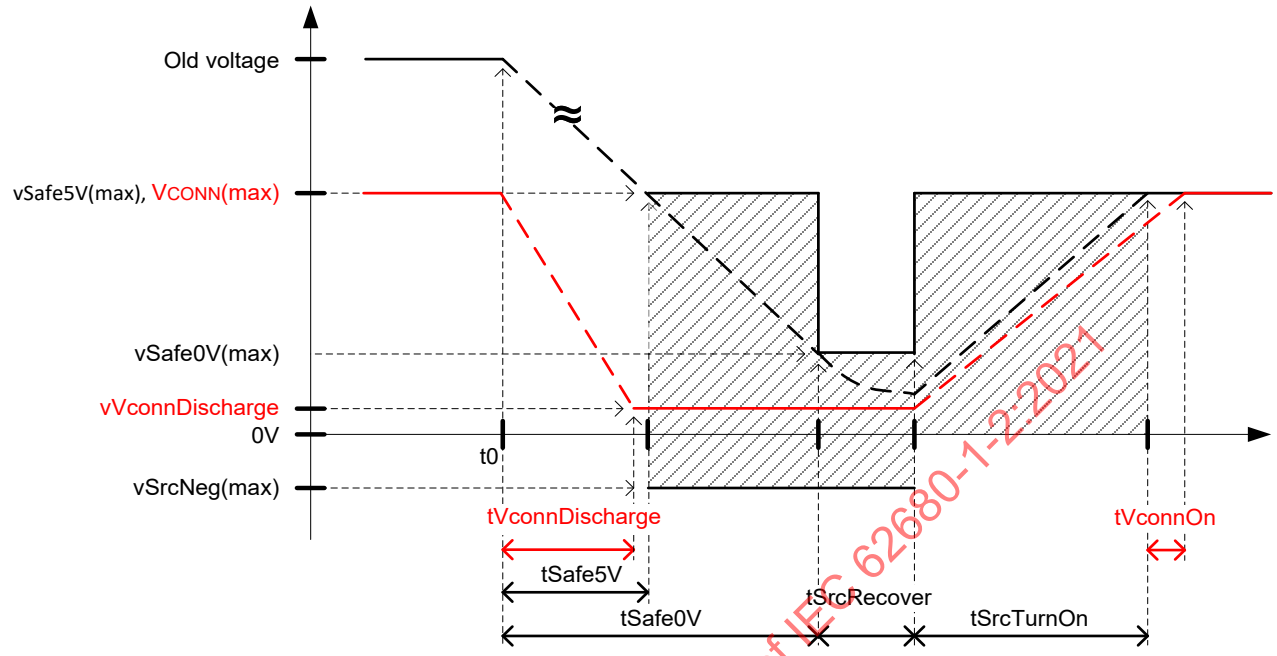
- ***Il convient de ne pas*** déconnecter les dispositifs autoalimentés de l'USB lors d'une réinitialisation matérielle (voir 9.1.2);
- les dispositifs autoalimentés qui fonctionnent à plus de ***vSafe5V*** ***Peut ne pas/Peuvent ne pas*** conserver toutes les fonctionnalités après une ***Hard Reset***;
- les dispositifs alimentés par bus se déconnectent de l'USB lors d'une réinitialisation matérielle en raison de la perte de leur source d'alimentation.

Lorsqu'une réinitialisation matérielle se produit, la source ***Devoir/Doit/Doivent*** arrêter de conduire VCONN, ***Devoir/Doit/Doivent*** supprimer la Rp de la broche VCONN et ***Devoir/Doit/Doivent*** commencer à faire passer la tension V_{BUS} à ***vSafe0V***, soit:

- ***tPSHardReset*** lorsque le dernier bit du signal ***Hard Reset*** a été reçu en provenance du destinataire; ou
- ***tPSHardReset*** après envoi du dernier bit du signal ***Hard Reset*** par la source.

La source ***Devoir/Doit/Doivent*** satisfaire à la fois à ***tSafe5V*** et ***tSafe0V*** par rapport au début de la transition de la tension, comme indiqué à la Figure 7-10.

Figure 7-10 Réponse à une réinitialisation matérielle de V_{BUS} et VCONN Source



Anglais	Français
Old voltage	Ancienne tension

VCONN satisfait à ***tVconnDischarge*** par rapport au début de la transition de la tension, comme le montre la Figure 7-10, en raison du circuit de décharge qui se trouve dans la fiche de câble. VCONN ***Devoir/Doit/Doivent*** satisfaire à ***tVconnOn*** par rapport à V_{BUS}, qui atteint ***vSafe5V***. Noter que ***tVconnOn*** et ***tVconnDischarge*** sont définis dans ***[USB Type-C 2.0]***.

7.1.6 Modification de la capacité de puissance de sortie

Certaines négociations sur l'alimentation électrique par port USB nécessitent que la source ajuste sa capacité de puissance de sortie sans modifier la tension de sortie. Dans ce cas, la source ***Devoir/Doit/Doivent*** être en mesure de fournir un courant de charge plus élevé ou plus faible dans un délai ***tSrcReady***.

7.1.7 Fonctionnement robuste de la source

7.1.7.1 Protection contre les surintensités de sortie

Les sources ***Devoir/Doit/Doivent*** mettre en œuvre une protection contre les surintensités de sortie pour éviter les dommages causés par un courant de sortie qui dépasse la capacité de traitement du courant de la source. La définition de la capacité de traitement du courant est laissée libre selon la mise en œuvre de la source; elle ***Devoir/Doit/Doivent*** tenir compte de la capacité de traitement du courant des contacts du connecteur. La réponse à la surintensité ***Ne doit/doivent pas*** interférer avec le niveau de courant V_{BUS} négocié.

Il convient d'/de/qu'/que les sources tentent d'envoyer un message ***Hard Reset*** lorsque la protection contre les surintensités se déclenche, suivi d'un message ***Alert*** indiquant un événement de protection contre les surintensités après établissement d'un contrat explicite. La réponse à la protection contre les surintensités ***Pouvoir/Peut/Peuvent*** se déclencher au niveau du port ou du système. ***Il convient d'/de/qu'/que*** les systèmes ou les ports qui ont déclenché la protection contre les surintensités tentent de reprendre leur fonctionnement par défaut après avoir déterminé que la cause de la surintensité n'est plus présente, et ils ***Pouvoir/Peut/Peuvent*** se verrouiller pour se protéger. La manière de détecter si la cause de la surintensité est toujours présente est laissée libre selon la mise en œuvre de la source.

La source ***Devoir/Doit/Doivent*** renégocier avec le ou les destinataires après avoir choisi de reprendre le fonctionnement par défaut. La manière de renégocier après un événement de surintensité est laissée libre selon la mise en œuvre de la source.

La source ***Devoir/Doit/Doivent*** empêcher la continuité cyclique du système ou du port si la protection contre les surintensités continue à se déclencher après la reprise initiale du fonctionnement par défaut ou de la renégociation. Le verrouillage du port ou du système est une réponse acceptable à une surintensité récurrente.

Au cours de la réponse à la surintensité et de l'arrêt subséquent du système ou du port, tous les ports sources concernés fonctionnant avec V_{BUS} supérieure à ***vSafe5V*** ***Devoir/Doit/Doivent***décharger V_{BUS} jusqu'à ***vSafe5V*** dans le délai ***tSafe5V*** et jusqu'à ***vSafe0V*** dans le délai ***tSafe0V***.

7.1.7.2 Protection contre les surchauffes

Les sources ***Devoir/Doit/Doivent*** mettre en œuvre une protection contre les surchauffes pour éviter les dommages causés par une température qui dépasse la capacité thermique de la source. La définition de la capacité thermique et les emplacements de surveillance utilisés pour déclencher la protection contre les surchauffes sont laissés libres selon la mise en œuvre de la source.

Il convient d'/de/qu'/que les sources tentent d'envoyer un message ***Hard Reset*** lorsque la protection contre les surchauffes se déclenche, suivi d'un message ***Alert*** indiquant un événement OTP après établissement d'un contrat explicite. La réponse à la protection contre les surchauffes ***Pouvoir/Peut/Peuvent*** se déclencher au niveau du port ou du système. ***Il convient d'/de/qu'/que*** les systèmes ou les ports qui ont déclenché la protection contre les surchauffes tentent de reprendre leur fonctionnement par défaut, et ils ***Pouvoir/Peut/Peuvent*** se verrouiller pour se protéger.

La source ***Devoir/Doit/Doivent*** renégocier avec le ou les destinataires après avoir choisi de reprendre le fonctionnement par défaut. La manière de renégocier après un événement de surchauffe est laissée libre selon la mise en œuvre de la source.

La source ***Devoir/Doit/Doivent*** empêcher la continuité cyclique du système ou du port si la protection contre les surchauffes continue à se déclencher après la reprise initiale du fonctionnement par défaut ou de la renégociation. Le verrouillage du port ou du système est une réponse acceptable à une surchauffe récurrente.

Au cours de la réponse de surchauffe et de l'arrêt subséquent du système ou du port, tous les ports source concernés qui fonctionnent avec une V_{BUS} supérieure à ***vSafe5V*** ***Devoir/Doit/Doivent*** décharger la V_{BUS} jusqu'à ***vSafe5V*** dans un délai de ***tSafe5V*** et jusqu'à ***vSafe0V*** dans un délai de ***tSafe0V***.

7.1.7.3 vSafe5V appliqué de manière externe aux ports fournissant vSafe5V

Un fonctionnement sûr oblige les sources d'alimentation électrique à ***Devoir/Doit/Doivent*** tolérer la présence de ***vSafe5V*** sur V_{BUS} lors de l'application simultanée de puissance à V_{BUS}. La communication de l'alimentation électrique par port USB normale ***Devoir/Doit/Doivent*** être prise en charge lorsque cette connexion ***vSafe5V*** à ***vSafe5V*** existe.

7.1.7.4 Débranchement

Un débranchement USB est détecté électriquement par la détection de la voie de communication sur le connecteur USB Type-C®. Lorsque la source est débranchée, la source ***Devoir/Doit/Doivent*** passer à ***vSafe0V*** dans un délai ***tSafe0V*** par rapport au moment où l'événement de débranchement a eu lieu. Au cours de la transition à ***vSafe0V***, la tension V_{BUS} ***Devoir/Doit/Doivent*** être inférieure à ***vSafe5V*** max dans un délai ***tSafe5V*** par rapport au moment où l'événement de débranchement a eu lieu et ***Ne doit/doivent pas*** dépasser ***vSafe5V*** max après cette durée.

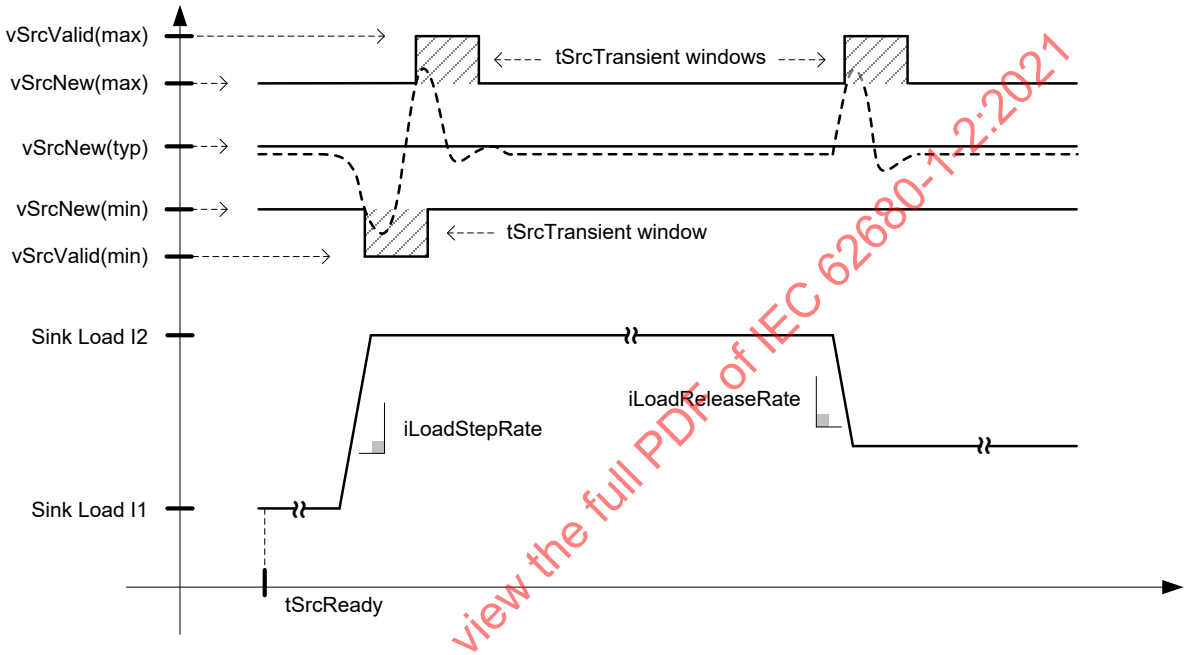
7.1.8 Tolérance et plage de tensions de sortie

Après l'achèvement d'une transition de tension (c'est-à-dire après ***tSrcReady***) et dans des conditions de charge statique, la tension de sortie de la source ***Devoir/Doit/Doivent*** rester dans les limites de ***vSrcNew*** ou ***vSafe5V***, selon le cas. Les plages définies par ***vSrcNew*** et ***vSafe5V*** tiennent compte de la précision de la régulation du courant continu, de la régulation de la ligne, de la régulation de la charge et de l'ondulation

de sortie. Après l'achèvement d'une transition de tension (c'est-à-dire après ***tSrcReady***) et dans des conditions de charge transitoire, la tension de sortie de la source ***Ne doit/doivent pas*** sortir de la plage spécifiée par ***vSrcValid***. La durée pendant laquelle la tension de sortie de la source peut se trouver dans la plage comprise entre ***vSrcNew*** ou ***vSafe5V*** et ***vSrcValid*** ***Ne doit/doivent pas*** dépasser ***tSrcTransient***. Se reporter au Tableau 7-22 pour connaître les spécifications de tolérance de la tension de sortie. La Figure 7-11 représente l'application de ***vSrcNew*** et ***vSrcValid*** lorsque la transition de tension est achevée.

Les limites ***vSrcNew*** et ***vSrcValid*** ***Ne doit/doivent pas*** s'appliquer à V_{BUS} lors de la décharge et du basculement de V_{BUS} qui surviennent lors d'une permutation rapide de rôles, comme décrit en 7.1.13.

Figure 7-11 Application des limites vSrcNew et vSrcValid après tSrcReady



Anglais	Français
tSrcTransient windows	fenêtres tSrcTransient
tSrcTransient window	fenêtre tSrcTransient
Sink Load	Charge destinataire

La tension de sortie de la source ***Devoir/Doit/Doivent*** être mesurée au niveau de l'embase du connecteur. La stabilité de la source ***Devoir/Doit/Doivent*** être soumise à l'essai par incréments de charge de 25 %, de la charge minimale à la charge maximale, ainsi que de la charge maximale à la charge minimale. Le comportement transitoire du courant de charge est défini en 7.2.6. Le temps entre chaque palier de charge ***Devoir/Doit/Doivent*** être suffisant pour permettre à la tension de sortie de se stabiliser. Dans certains systèmes, il peut être nécessaire de concevoir la source de manière à compenser la chute de tension entre l'étape de sortie de l'électronique d'alimentation et le contact de l'embase. L'identification de la nécessité d'une compensation est laissée libre selon la mise en œuvre de la source.

7.1.8.1 Tolérance et plage de tensions de sortie de l'alimentation électrique programmable

Après l'achèvement d'une transition de tension d'une alimentation électrique programmable (c'est-à-dire après ***TPpsSrcTransSmall*** ou ***TPpsSrcTransLarge***) et dans des conditions de charge statique, la tension de sortie de la source ***Devoir/Doit/Doivent*** rester dans les limites de ***vPpsNew***. La plage définie par ***vPpsNew*** tient compte de la précision de la régulation du courant continu, de la régulation de la ligne, de la régulation de la charge et de l'ondulation de sortie. Après l'achèvement d'une transition de tension (c'est-à-dire après ***TPpsSrcTransSmall*** ou ***TPpsSrcTransLarge***) et dans des conditions de charge

transitoire, la tension de sortie de la source ***Ne doit/doivent pas*** sortir de la plage spécifiée par ***vPpsValid***. La durée pendant laquelle la tension de sortie de la source peut être comprise entre ***vPpsNew*** et ***vPpsValid*** ***Ne doit/doivent pas*** dépasser ***tPpsTransient***.

7.1.9 Charge et décharge de la capacité de masse sur V_{BUS}

La source ***Devoir/Doit/Doivent*** charger et décharger la capacité de masse sur V_{BUS} chaque fois que la tension de la source est négociée à une valeur différente. La charge et la décharge se produisent pendant la transition de la tension et ***Ne doit/doivent pas*** interférer avec la capacité de la source à satisfaire à ***tSrcReady***.

7.1.10 Veille de permutation pour les sources

Les sources et les destinataires d'un port d'alimentation double fonction ***Devoir/Doit/Doivent*** prendre en charge la veille de permutation. La veille de permutation se produit pour la source lorsque l'alimentation électrique de la source a déchargé la capacité de masse sur V_{BUS} vers ***vSafe0V*** dans le cadre de la permutation des rôles d'alimentation.

Lorsque la source est en veille de permutation:

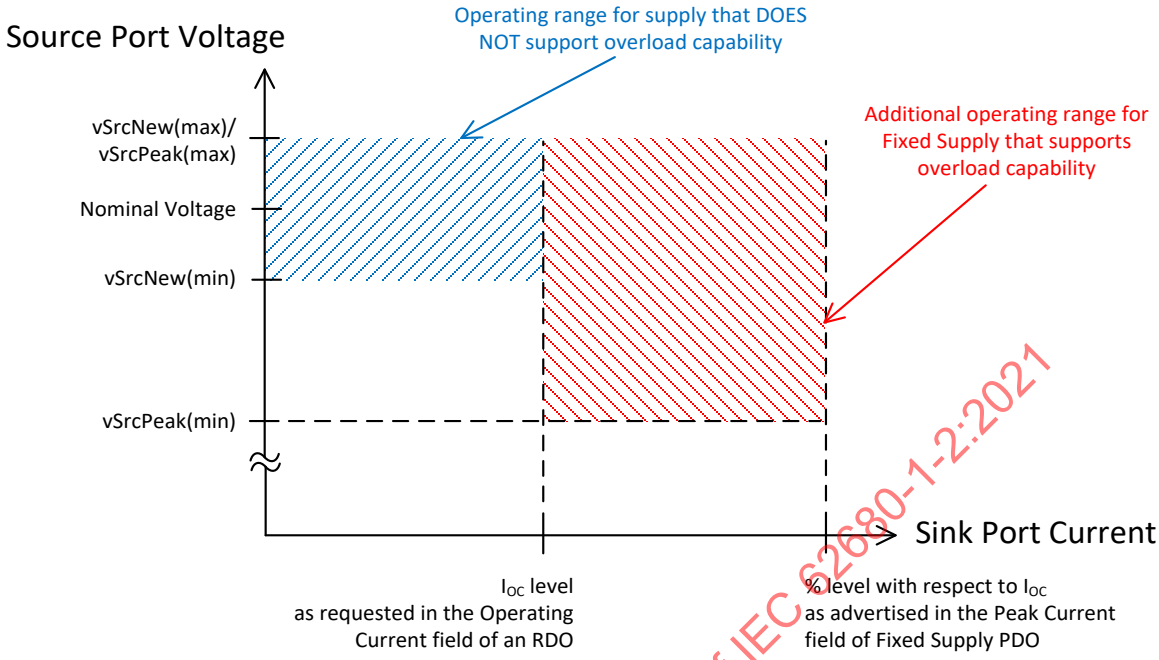
- elle ***Ne doit/doivent pas*** commander V_{BUS}, qui est donc réputée être maintenue à ***vSafe0V***;
- tout circuit de décharge utilisé pour atteindre ***vSafe0V*** ***Devoir/Doit/Doivent*** être retiré de V_{BUS};
- le port d'alimentation à double fonction ***Devoir/Doit/Doivent*** être configuré en tant que destinataire;
- la connexion USB ***Ne doit/doivent pas*** être réinitialisée, même si ***vSafe5V*** n'est plus présent sur V_{BUS} (voir 9.1.2).

Le message ***PS_RDY*** associé à la source en veille de permutation ***Devoir/Doit/Doivent*** être envoyé après la suppression du pilote V_{BUS}. Le temps nécessaire à la source pour passer en veille de permutation ***Ne doit/doivent pas*** dépasser ***tSrcSwapStdby***. En entrant en veille de permutation, la source renonce à son rôle de source et est prête à devenir le nouveau destinataire. Le temps de transition entre la veille de permutation et l'état de nouveau destinataire ***Devoir/Doit/Doivent*** être inférieur ou égal à ***tNewSnk***. Le nouveau destinataire ***Pouvoir/Peut/Peuvent*** commencer à utiliser la puissance après envoi du message ***PS_RDY*** par la nouvelle source.

7.1.11 Fonctionnement en courant de crête de la source

Une source dont les bits de courant de crête du PDO d'alimentation fixe sont définis sur 01b, 10b et 11b ***Devoir/Doit/Doivent*** être conçue pour prendre en charge une des capacités de surcharge définies dans le Tableau 6-10. Les conditions de surcharge sont liées à l'amplitude, à la durée et au cycle de fonctionnement répertoriés dans le Tableau 6-10. Les sources ne sont pas tenues de prendre en charge le fonctionnement en surcharge continue. Lorsque des conditions de surcharge se produisent, il est admis que la source utilise la plage de ***vSrcPeak*** (au lieu de ***vSrcNew***) en fonction de la valeur nominale (voir Figure 7-12). Lorsque la capacité de surcharge est dépassée, la source est censée prendre toutes les mesures nécessaires pour empêcher tout dommage électrique ou thermique. Elle ***Pouvoir/Peut/Peuvent*** envoyer un nouveau message ***Source_Capabilities*** dont les bits de courant de crête du PDO à alimentation fixe sont définis sur 00b pour interdire le fonctionnement en surcharge, même si une capacité de surcharge a précédemment été négociée avec le destinataire.

Figure 7-12 Surcharge du courant de crête de la source



Anglais	Français
Source Port Voltage	Tension du port source
Operating range for supply that DOES NOT support overload capability	Plage de fonctionnement pour une alimentation qui NE prend PAS en charge la capacité de surcharge
Nominal Voltage	Tension nominale
Additional operating range for Fixed Supply that supports overload capability	Plage de fonctionnement supplémentaire pour une alimentation fixe qui prend en charge la capacité de surcharge
Sink Port Current	Courant du port destinataire
I_{oc} level as requested in the Operating Current field of an RDO	Niveau I_{oc} demandé dans le champ Operating Current d'un RDO
% level with respect to I_{oc} as advertised in the Peak Current field of Fixed Supply PDO	niveau en % par rapport à I_{oc} annoncé dans le champ Peak Current du PDO d'alimentation fixe

7.1.12 Paramètres étendus des capacités de source

Les implémenteurs peuvent choisir de mettre à disposition certaines caractéristiques d'une source d'alimentation électrique par port USB comme un ensemble de paramètres statiques et/ou dynamiques permettant d'améliorer l'interopérabilité entre les sources d'alimentation externes et les dispositifs informatiques portatifs. Tous les paramètres statiques transposables sont décrits en détail en 6.5.1 et sont répertoriés à la Figure 6-32. Le sous-ensemble des paramètres énumérés ci-dessous représente directement les capacités de source. Ils sont décrits dans le reste de la présente section:

- régulation de la tension;
- temps de maintien;
- conformité;
- courant de crête;
- entrées source;
- batteries.

7.1.12.1 Champ Voltage Regulation

La consommation électrique d'un dispositif peut varier de manière dynamique. La capacité de la source à réguler sa tension de sortie peut être un point important si le dispositif est sensible aux fluctuations de tension. Le champ de bit Voltage Regulation est utilisé pour transmettre des informations sur la régulation et la tolérance de sortie des sources à différents paliers de charge.

7.1.12.1.1 Vitesse de balayage des paliers de charge

La vitesse de balayage par défaut des paliers de charge est établie à 150 mA/µs. Une source ***Devoir/Doit/Doivent*** répondre aux exigences suivantes sous le palier de charge indiqué dans les capacités étendues de source:

- la source ***Devoir/Doit/Doivent*** maintenir la régulation de V_{BUS} dans la plage ***vSrcValid***;
- le bruit sur la voie de communication ***Devoir/Doit/Doivent*** rester inférieur à ***vNoiseIdle*** et ***vNoiseActive.***

Les conditions d'essai exigent une modification des paliers de charge positifs et négatifs de 1 Hz à 5 000 Hz, jusqu'à l'amplitude de palier de charge indiquée de la sortie à pleine charge, y compris de la charge initiale de 10 mA et 10 %. La source ***Devoir/Doit/Doivent*** s'assurer que les communications de l'alimentation électrique satisfont aux masques de transmission et de réception spécifiés en 5.8.2 dans toutes les conditions de charge.

7.1.12.1.2 Amplitude des paliers de charge

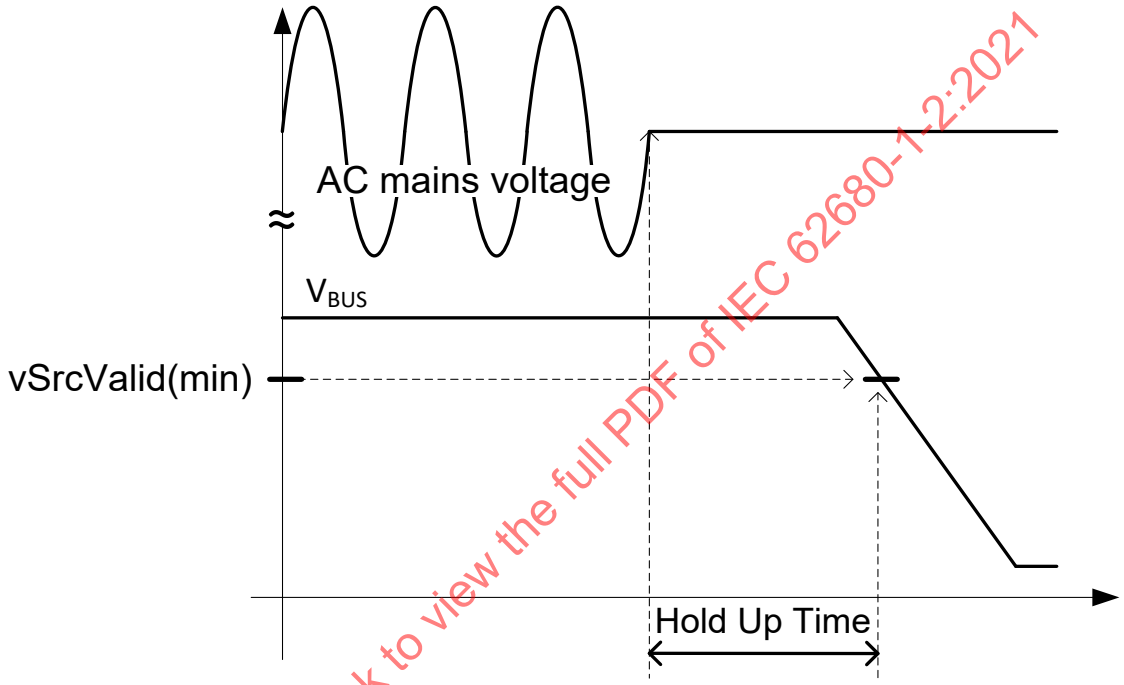
Le taux d'amplitude par défaut des paliers de charge ***Devoir/Doit/Doivent*** être de 25 % de l'IoC. La source ***Pouvoir/Peut/Peuvent*** indiquer une capacité plus élevée tolérant un palier de charge de 90 % de l'IoC.

7.1.12.2 Champ Holdup Time

Le champ Holdup Time ***Devoir/Doit/Doivent*** renvoyer une valeur numérique du nombre de millisecondes pendant lequel la tension de sortie maintient sa régulation après une courte interruption de l'alimentation en courant alternatif.

Une source alimentée par le secteur ***Devoir/Doit/Doivent*** indiquer son temps de maintien dans ce champ. Le temps de maintien est mesuré avec la charge à la valeur maximale assignée, et une alimentation secteur en courant alternatif à 115 V efficace et 60 Hz (ou à 230 V efficace et 50 Hz pour une source qui ne prend pas en charge l'alimentation secteur à 115 V en courant alternatif). Le temps indiqué décrit la durée minimale depuis le dernier cycle d'entrée complet de l'alimentation secteur en courant alternatif (angle de phase de zéro degré), jusqu'à ce que la tension de sortie soit réduite à une valeur inférieure à ***vSrcValid*** (min). Il est recommandé que les sources d'alimentation prennent en charge un temps de maintien minimal de 3 ms, mais il est préférable qu'elles prennent en charge un temps de maintien de 10 ms (équivalent à une chute sur un demi-cycle de l'alimentation secteur en courant alternatif).

Figure 7-13 Mesure du temps de maintien



Anglais	Français
AC mains voltage	Tension de l'alimentation secteur
Hold Up Time	Temps de maintien

7.1.12.3 Champ Compliance

Une source revendiquant la conformité LPS, PS1 ou PS2 (voir ***[IEC 62368-1]***) ***Devoir/Doit/Doivent*** indiquer ses capacités dans le champ Compliance. Etant donné que la source ***Pouvoir/Peut/Peuvent*** avoir plusieurs paramètres de tension et de courant de sortie potentiels, chaque alimentation (indiquée par un PDO) ***Devoir/Doit/Doivent*** être conforme aux exigences LPS.

Note: Selon les exigences de l'***[IEC 60950-1]***, un dispositif soumis à l'essai et certifié avec une source LPS a l'interdiction d'utiliser une source non LPS. Alternativement, l'***[IEC 62368-1]*** classe les sources d'alimentation en fonction de leur puissance de sortie maximale et limitée (15 watts ou 100 watts).

7.1.12.4 Courant de crête

La source indique dans le champ Peak Current sa capacité à distribuer un courant de crête supérieur à l'intensité négociée. La durée du courant de crête ***Devoir/Doit/Doivent*** être suivie d'une consommation de courant inférieure au courant de fonctionnement (IoC) afin de maintenir une distribution de puissance moyenne inférieure au courant de fonctionnement.

Une source ***Pouvoir/Peut/Peuvent*** disposer d'une capacité de distribution du courant de crête supérieure à celle qui peut être indiquée dans le champ Peak Current du PDO d'alimentation fixe. Dans ce cas, la source ***Devoir/Doit/Doivent*** indiquer sa capacité supplémentaire dans le champ Peak Current du message ***Source_Capabilities_Extended***.

Chaque période de surcharge ***Devoir/Doit/Doivent*** être suivie d'une période de consommation de courant réduite. Le courant moyen de roulement sur la valeur du champ Overload Period, avec la valeur spécifiée du champ Duty Cycle (voir 6.5.1.10), ***Ne doit/doivent pas*** dépasser le courant négocié. Ce courant moyen est calculé par:

*Période de courant réduit = (1 – valeur du champ Duty Cycle/100) * valeur du champ Overload Period*

7.1.12.5 Entrées source

Le champ Source Inputs identifie les entrées possibles qui alimentent la source. Noter que certaines sources ne sont alimentées que par batterie (une automobile, par exemple) et non par le secteur, ce qui est pourtant plus commun.

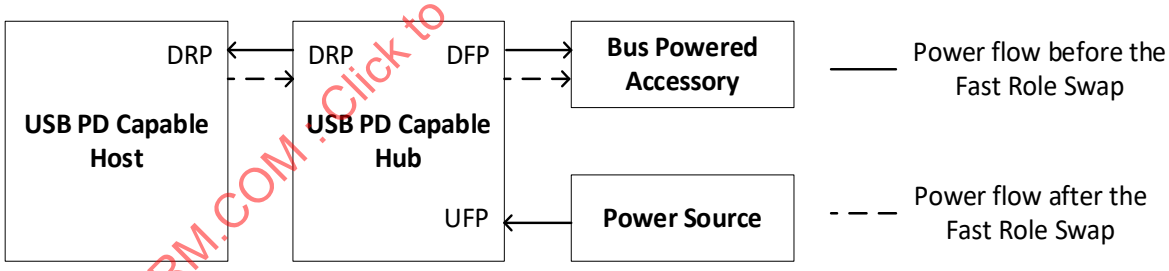
7.1.12.6 Batteries

Le champ Batteries ***Devoir/Doit/Doivent*** indiquer le nombre de batteries prises en charge par la source. La source ***Devoir/Doit/Doivent*** indiquer de manière indépendante le nombre de batteries remplaçables à chaud et le nombre de batteries fixes.

7.1.13 Permutation rapide des rôles

Une permutation rapide des rôles limite l'interruption de la puissance V_{BUS} à un accessoire alimenté par bus connecté à un hub DFP, lui-même équipé d'un UFP branché à une source d'alimentation et d'une alimentation double fonction branchée à un port hôte prenant en charge l'alimentation double fonction, comme indiqué à la Figure 7-14 Puissance V_{BUS} au cours d'une permutation rapide des rôles.

Figure 7-14 Puissance V_{BUS} au cours d'une permutation rapide des rôles



Anglais	Français
USB PD Capable Host	Hôte apte à l'alimentation électrique par port USB
USB PD Capable Hub	Hub apte à l'alimentation électrique par port PD USB
Bus Powered Accessory	Accessoire alimenté par bus
Power Source	Source de puissance
Power flow before the Fast Role Swap	Flux de puissance avant la permutation rapide des rôles
Power flow after the Fast Role Swap	Flux de puissance après la permutation rapide des rôles

Si V_{BUS} a été négociée à une tension supérieure à ***vSafe5V*** ou ***vSafe5V*** (min), lorsque la source d'alimentation connectée au hub UFP cesse de fournir l'alimentation et que V_{BUS} sur le connecteur du hub DRP se décharge sous ***vSrcValid*** (min), le signal de permutation rapide des rôles ***Devoir/Doit/Doivent*** être envoyé du hub DRP à l'hôte DRP, et le hub DRP ***Devoir/Doit/Doivent*** recevoir l'alimentation. Dans le

cas d'utilisation de la permutation rapide des rôles, le hub DRP se comporte comme un chemin de puissance bidirectionnel. Le hub DRP ***Ne doit/doivent pas*** permettre à V_{BUS} de décharger le circuit lors du changement de fonctionnement de la source initiale en nouveau destinataire. Le nouveau destinataire ***Devoir/Doit/Doivent*** être limité au courant USB Type-C (voir ***[USB Type-C 2.0]***) jusqu'à ce qu'un nouveau contrat explicite soit négocié.

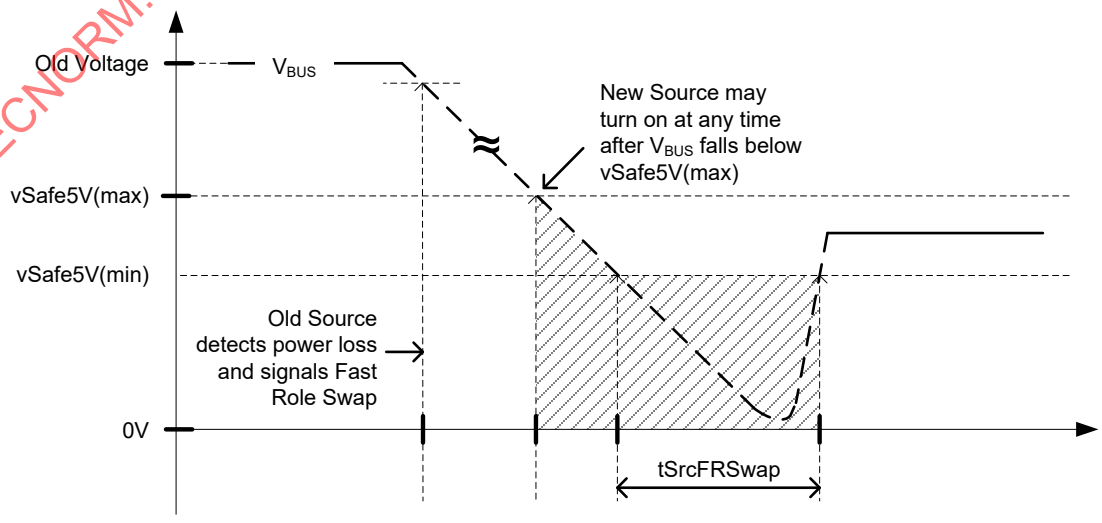
Après l'envoi du signal FRS et alors que V_{BUS} > ***vSafe5V*** (min), le nouveau destinataire ***Ne doit pas*** consommer plus que ***INewFrsSink***, jusqu'à ce que la nouvelle source ait appliqué sa Rp. Le nouveau destinataire ***Ne doit pas*** consommer plus que ***pSnkStdby*** de V_{BUS} avant ***TSnkFRSwap*** après avoir commencé à envoyer le signal FRS ou que V_{BUS} a chuté au-dessous ***vSafe5V*** (min). La durée ***TSnkFRSwap*** ***Devoir/Doit/Doivent*** commencer au début du signal FRS ou lorsque V_{BUS} chute au-dessous de ***vSafe5V*** (min), selon ce qui se produit le plus tard. Après avoir attendu ***TSnkFRSwap***, le nouveau destinataire ***Ne doit pas*** consommer plus que ***INewFrsSink***, jusqu'à ce que la nouvelle source ait appliqué sa Rp. Lorsque la nouvelle source a appliqué sa Rp, le nouveau destinataire ***Devoir/Doit/Doivent*** être limité au courant USB Type-C (voir ***[USB Type-C 2.0]***) d'un contrat implicite jusqu'à ce qu'un nouveau contrat explicite soit négocié. Toutes les exigences relatives au destinataire ***Devoir/Doit/Doivent*** s'appliquer au nouveau destinataire à l'issue de la permutation rapide des rôles. La réponse à la permutation rapide des rôles de l'hôte DRP est décrite en 7.2.10, dans la mesure où l'hôte DRP fonctionne comme le destinataire initial avant la permutation rapide des rôles.

Lorsque le niveau de tension V_{BUS} au connecteur du hub DRP passe sous ***vSafe5V***, un message ***PS_RDY*** ***Devoir/Doit/Doivent*** être envoyé à l'hôte DRP, comme indiqué dans le diagramme de transition de la permutation rapide des rôles donné en 7.3.15.

La Figure 7-15 représente la détection et la temporisation de V_{BUS} pour la nouvelle source lors d'une permutation rapide des rôles après la réception du signal de permutation rapide des rôles. La nouvelle source ***Pouvoir/Peut/Peuvent*** activer le commutateur de sortie V_{BUS} lorsque V_{BUS} est au-dessous de ***vSafe5V*** (max). Dans ce cas, la nouvelle source empêche V_{BUS} de descendre sous ***vSafe5V*** (min). La nouvelle source ***Devoir/Doit/Doivent*** activer le commutateur de sortie V_{BUS} dans un délai ***tSrcFRSwap*** de chute sous ***vSafe5V*** (min).

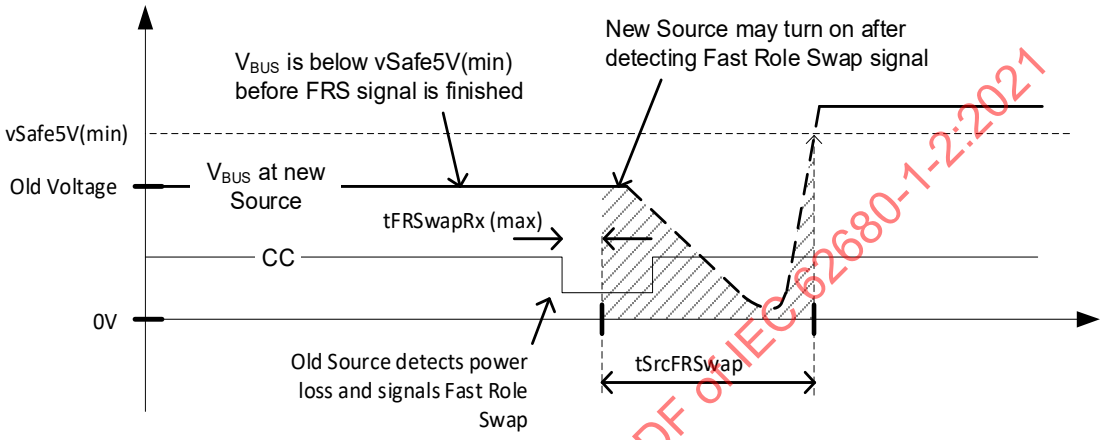
V_{BUS} peut avoir démarré à ***vSafe5V*** ou à une tension plus élevée. Lorsque le signal de permutation rapide des rôles est détecté, V_{BUS} peut être au-dessus de ***vSafe5V*** (max), dans la plage ***vSafe5V***, ou au-dessous de ***vSafe5V*** (min). Si le signal de permutation rapide des rôles est détecté lorsque V_{BUS} est inférieure à ***vSafe5V*** (min), alors la nouvelle source ***Devoir/Doit/Doivent*** activer le commutateur de sortie V_{BUS} dans un délai ***tSrcFRSwap*** après la détection du signal de permutation rapide des rôles. Dans ce cas, la durée maximale entre le début du signal de permutation rapide des rôles à l'alimentation de V_{BUS} ***Pouvoir/Peut/Peuvent*** être de ***tSrcFRSwap*** (max) + ***tFRSwapRx*** (max).

Figure 7-15 Détection et temporisation de V_{BUS} lors d'une permutation rapide des rôles, avec V_{BUS} initiale (à la nouvelle source) > vSafe5V (min).



Anglais	Français
Old voltage	Ancienne tension
New Source may turn on at any time after V_{BUS} falls below vSafe5V(max)	Une nouvelle source peut s'activer à tout moment après la chute de V_{BUS} sous vSafe5V(max)
Old Source detects power loss and signals Fast Role Swap	L'ancienne source détecte une perte de puissance et signale une permutation rapide des rôles

Figure 7-16 Détection et temporisation de VBUS lors d'une permutation rapide des rôles, avec VBUS initiale (à la nouvelle source) < vSafe5V (min).



Anglais	Français
Old Voltage	Ancienne tension
Vbus is below vSafe5V(min) before FRS signal is finished	V_{BUS} est inférieure à vSafe5V(min) avant la finalisation du signal FRS
New Source may turn on after detecting Fast Role Swap signal	La nouvelle source peut être activée après avoir détecté le signal de permutation rapide des rôles
Vbus at new Source	V_{BUS} de la nouvelle source
Old Source detects Power loss and signals Fast Roles Swap	L'ancienne source détecte une perte d'alimentation et signale la permutation rapide des rôles

7.1.14 Non application des limites de la vitesse de balayage de V_{BUS}

Voici les scénarios dans lesquels les limites ***vSrcSlewPos*** et ***vPpsSlewPos*** de la vitesse de balayage de V_{BUS} ne s'appliquent pas, et dans lesquels V_{BUS} ***Pouvoir/Peut/Peuvent*** effectuer une transition plus rapide que celle spécifiée:

- lors de la première application de V_{BUS} après un branchement;
- lors de l'augmentation de V_{BUS} de ***vSafe0V*** à ***vSafe5V*** au cours d'une réinitialisation matérielle;
- au cours d'une permutation rapide des rôles lorsque le destinataire initial applique V_{BUS}.

Voici les scénarios dans lesquels les limites ***vSrcSlewNeg*** et ***vPpsSlewNeg*** de la vitesse de balayage de V_{BUS} ne s'appliquent pas, et dans lesquels V_{BUS} ***Pouvoir/Peut/Peuvent*** effectuer une transition plus rapide que celle spécifiée:

- lors de la décharge de V_{BUS} à ***vSafe0V*** au cours d'une réinitialisation matérielle;
- lors de la décharge de V_{BUS} à ***vSafe0V*** après un débranchement;
- au cours d'une permutation rapide des rôles lorsque la source d'alimentation de V_{BUS} connectée au hub UFP cesse de délivrer de la puissance.

7.1.15 Cycle d'alimentation VCONN

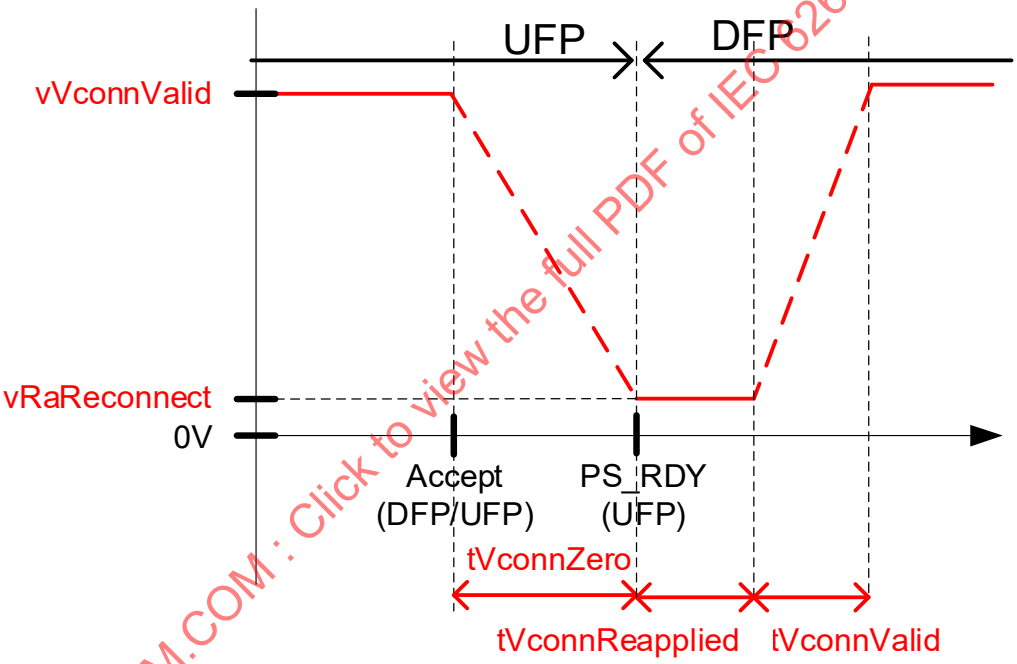
7.1.15.1 Cycle d'alimentation VCONN UFP

Le processus de réinitialisation des données exige que le DFP devienne la source VCONN avant la fin du processus. Dans le cas où l'UFP est la source VCONN, les étapes suivantes ***Devoir/Doit/Doivent*** être respectées:

1. à partir de la réception du dernier bit du ***GoodCRC*** qui acquitte le message ***Accept*** en réponse au message ***Data_Reset***, l'UFP doit désactiver VCONN et s'assurer qu'elle se situe au-dessous de la valeur vRaReconnect (voir ***[USB Type-C 2.0]***) dans un délai ***tVconnZero***;
2. lorsque VCONN se situe au-dessous de la valeur vRaReconnect, l'UFP ***Devoir/Doit/Doivent*** envoyer un message ***PS_RDY***. Noter que si l'UFP n'alimente pas VCONN, il envoie quand même le message ***PS_RDY***;
3. le DFP doit attendre le délai ***tVconnReapplied*** à partir de la réception du dernier bit du ***GoodCRC*** qui acquitte le message ***PS_RDY*** avant d'alimenter VCONN. le DFP doit s'assurer que VCONN se trouve dans la plage vVconnValid (voir ***[USB Type-C 2.0]***) dans un délai ***tVconnValid***.

La Figure 7-17 ci-dessous représente le processus du cycle d'alimentation VCONN UFP.

Figure 7-17 Cycle d'alimentation VCONN UFP dans une réinitialisation des données



7.1.15.2 Cycle d'alimentation VCONN DFP

Le processus de réinitialisation des données exige que le DFP devienne la source VCONN avant la fin du processus. Dans le cas où le DFP est la source VCONN, les étapes suivantes ***Devoir/Doit/Doivent*** être respectées:

1. si le DFP a envoyé le message ***Data_Reset*** et qu'il alimente VCONN, il ***Devoir/Doit/Doivent*** désactiver VCONN et s'assurer qu'elle se situe au-dessous de la valeur vRaReconnect (voir ***[USB Type-C 2.0]***) dans un délai ***tVconnZero*** à partir de la réception du dernier bit ***GoodCRC*** qui acquitte le message ***Accept*** envoyé en réponse au message ***Data_Reset***;

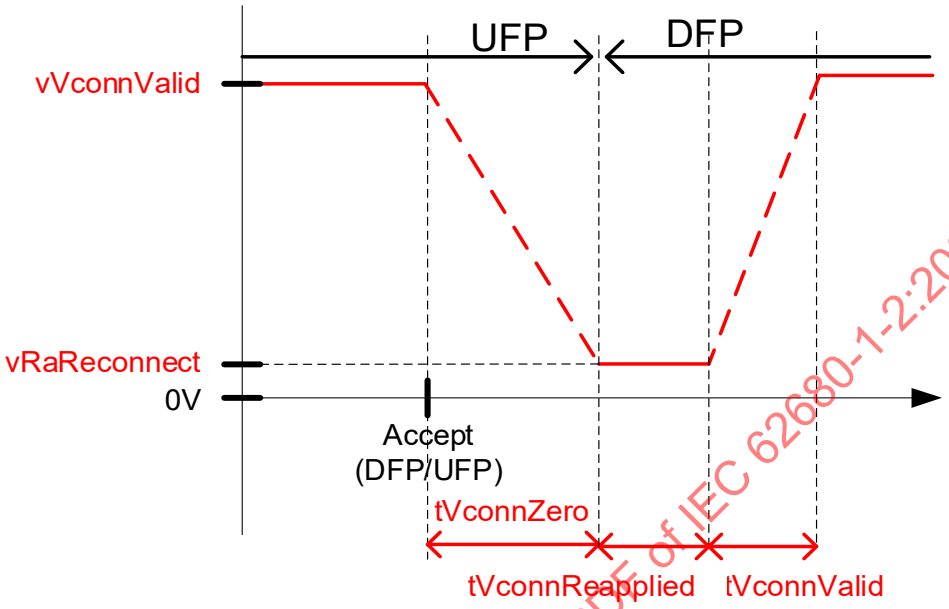
2. si l'UFP a envoyé le message ***Data_Reset***, il doit désactiver VCONN et s'assurer qu'elle se situe au-dessous de la valeur vRaReconnect (voir ***[USB Type-C 2.0]***) dans un délai ***tVconnZero*** à partir de la réception du dernier bit du ***GoodCRC*** qui acquitte le message ***Accept*** envoyé en réponse au message ***Data_Reset***;

3. lorsque VCONN se situe au-dessous de la valeur vRaReconnect, le DFP doit attendre ***tVconnReapplied*** avant de fournir VCONN;

4. le DFP doit s'assurer que VCONN se trouve dans la plage vVconnValid (voir ***[USB Type-C 2.0]***) dans un délai ***tVconnValid***.

La Figure 7-18 ci-dessous représente le processus du cycle d'alimentation VCONN DFP.

Figure 7-18 Cycle d'alimentation VCONN DFP dans une réinitialisation des données



7.2 Exigences relatives au destinataire

7.2.1 Aspects comportementaux

Un destinataire d'alimentation électrique par port USB présente les comportements suivants:

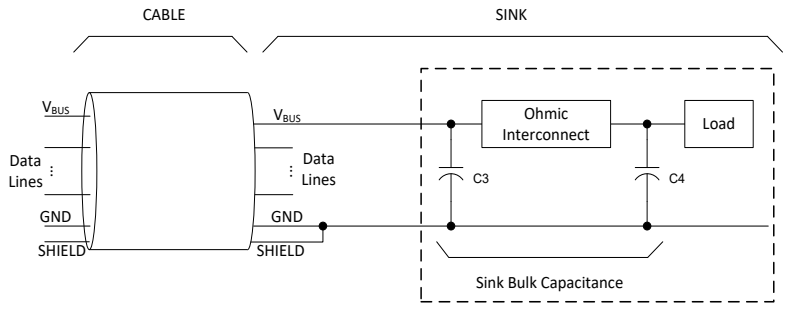
- il ***Devoir/Doit/Doivent*** prélever le courant V_{BUS} ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** ou ***[USBBC 1.2]*** par défaut en l'absence de contrat (fonctionnement USB par défaut);
- il ***Devoir/Doit/Doivent*** suivre les exigences spécifiées en 7.1.5 lorsqu'un signal ***Hard Reset*** est reçu;
- il ***Devoir/Doit/Doivent*** contrôler le courant d'appel de V_{BUS} lors de l'augmentation de la consommation de courant.

7.2.2 Capacité de masse du destinataire

La capacité de masse du destinataire consiste en C3 et C4, comme représenté à la Figure 7-19. L'interconnexion ohmique peut être constituée de pistes de CCI pour les dispositifs de distribution ou de commutation de puissance. La capacité peut être un condensateur unique, une batterie de condensateurs ou une capacité distribuée. La limite supérieure de ***cSnkBulkPd*** ***Ne doit/doivent pas*** être dépassée pour que la charge (ou décharge) transitoire de la capacité de masse totale sur V_{BUS} puisse être prise en compte lors des transitions de tension.

Il est admis que la capacité de masse du destinataire qui se situe dans les limites de ***cSnkBulk*** max ou de ***cSnkBulkPd*** max soit modifiée pour prendre en charge un nouveau niveau de puissance négocié. La capacité peut être modifiée lorsque le destinataire entre en veille, pendant une transition de tension ou lorsque le destinataire commence à fonctionner au nouveau niveau de puissance. La modification de la capacité de masse du destinataire ***Ne doit/doivent pas*** entraîner de courant transitoire sur V_{BUS} qui enfreindrait le contrat actuel. Au cours d'une permutation des rôles d'alimentation, le destinataire par défaut ***Devoir/Doit/Doivent*** passer en veille de permutation afin de devenir la nouvelle source. Toute modification de la capacité de masse exigée pour achever la permutation des rôles d'alimentation ***Devoir/Doit/Doivent*** intervenir au cours de la veille de permutation.

Figure 7-19 Placement de la capacité de masse du destinataire



Anglais	Français
CABLE	CÂBLE
SINK	DESTINATAIRE
Ohmic Interconnect	Interconnexion ohmique
Load	Charge
Data Lines	Lignes de données
GND	TERRE
SHIELD	BLINDAGE
Sink Bulk Capacitance	Capacité de masse du destinataire

7.2.3 Veille du destinataire

Le destinataire ***Devoir/Doit/Doivent*** passer par le mode veille avant une transition de tension positive ou négative de V_{BUS}. Pendant la veille, le destinataire ***Devoir/Doit/Doivent*** réduire sa consommation électrique à ***pSnkStdby.*** Cela permet à la source de gérer la transition de tension et de fournir au destinataire un courant de fonctionnement suffisant afin de maintenir l'alimentation électrique pendant la transition. Après avoir évalué le message ***Accept*** provenant de la source, le destinataire ***Devoir/Doit/Doivent*** effectuer cette transition vers le mode veille dans un délai ***tSnkStdby***. Au réveil du destinataire, la transition ***Devoir/Doit/Doivent*** s'effectuer dans un délai ***tSnkNewPower***. L'exigence ***pSnkStdby*** ne ***Devoir/Doit/Doivent*** s'appliquer que si la consommation électrique du destinataire est supérieure à ce niveau.

Voir 7.3 pour savoir dans quels cas ***pSnkStdby*** ***Devoir/Doit/Doivent*** être appliqué pour une transition donnée.

7.2.3.1 Veille du destinataire de l'alimentation programmable

Il n'est pas nécessaire qu'un destinataire passe en mode veille lorsqu'il fonctionne avec l'APDO PPS négocié. Un destinataire ***Pouvoir/Peut/Peuvent*** consommer la valeur du courant de fonctionnement indiquée dans le RDO programmable pendant les changements de tension de sortie de la PPS. Toutefois, avant de faire fonctionner la PPS en mode Current Limit, le destinataire ***Devoir/Doit/Doivent*** programmer la tension de fonctionnement de la PPS au niveau pratique le plus bas qui satisfait à l'exigence de charge du destinataire. Cela réduit le plus possible le courant d'appel qui se produit lors de la transition vers le mode Current Limit. Lors du fonctionnement avec une PPS en mode Current Limit, le destinataire ***Ne doit/doivent pas*** modifier sa charge de sorte qu'elle dépasse ***IPpsCLLoadStepRate*** ou ***IPpsCLLoadReleaseRate***. L'amplitude de la variation de charge ***Ne doit/doivent pas*** dépasser ***IPpsCLLoadStep*** ou ***IPpsCLLoadRelease***.

Si le destinataire négocie un nouvel APDO PPS, il ***Devoir/Doit/Doivent*** passer en mode veille en interchangeant les APDO PPS, comme décrit en 7.3.18.

7.2.4 Consommation électrique pendant la veille

Lorsqu'une source a défini son fanion USB Suspend Supported (voir 6.4.1.2.2.2), un destinataire ***Devoir/Doit/Doivent*** passer à l'état de puissance le plus faible pendant la veille USB. L'état de puissance le plus faible ***Devoir/Doit/Doivent*** être ***pSnkSusp*** ou inférieur pour un périphérique PDUSB, et ***pHubSusp*** ou inférieur pour un hub PDUSB. Il n'est pas exigé que la tension source soit modifiée pendant la veille USB.

7.2.5 Courant négocié à zéro

Lorsqu'un destinataire demande un courant nul dans le cadre d'une négociation de puissance avec une source, il ***Devoir/Doit/Doivent*** passer à l'état de puissance le plus faible, ***pSnkSusp*** ou inférieur, dans lequel il peut continuer à communiquer à l'aide de la signalisation d'alimentation électrique.

7.2.6 Comportement de charge transitoire

Lorsque le courant de fonctionnement d'un destinataire varie en raison d'un palier de charge, d'une libération de charge ou de toute autre modification du niveau de charge, le surdépassement positif ou négatif du nouveau courant de charge ***Ne doit/doivent pas*** dépasser la plage définie par ***iOvershoot***. Aux fins de mesure d'***iOvershoot***, la nouvelle valeur du courant de charge est définie comme la valeur moyenne du régime établi du courant de charge après l'établissement du palier de charge. La variation de tout décalage du courant de charge du destinataire pendant le fonctionnement normal ***Ne doit/doivent pas*** dépasser ***iLoadStepRate*** (pour les paliers de charge) et ***iLoadReleaseRate*** (pour les libérations de charge), mesurés à l'embase du destinataire.

Le courant de fonctionnement du destinataire ***Ne doit/doivent pas*** varier plus rapidement que la valeur indiquée dans le champ Load Step Slew Rate de la source, et ***Devoir/Doit/Doivent*** veiller à ce que les communications d'alimentation par USB satisfassent aux masques de transmission et de réception spécifiés en 5.8.2.

7.2.7 Veille de permutation pour les destinataires

Dans un port d'alimentation double fonction, le destinataire ***Devoir/Doit/Doivent*** prendre en charge la veille de permutation. La veille de permutation se produit pour le destinataire après l'évaluation du message ***Accept*** provenant de la source au cours d'une négociation de permutation des rôles d'alimentation. En veille de permutation, la consommation de courant du destinataire ***Ne doit/doivent pas*** dépasser ***iSnkSwapStdby*** de V_{BUS} et le port d'alimentation double fonction ***Devoir/Doit/Doivent*** être configuré en tant que source après la décharge de V_{BUS} à ***vSafe0V*** par la source initiale. ***Il convient de ne pas*** réinitialiser la connexion USB du destinataire, même si ***vSafe5V*** n'est pas présent sur le conducteur V_{BUS} (voir 9.1.2). Le temps nécessaire au destinataire pour passer en veille de permutation ne ***Devoir/Doit/Doivent*** pas être supérieur à ***tSnkSwapStdby***. Lorsque le destinataire est en veille de permutation, il renonce à son rôle de destinataire et se prépare à devenir la nouvelle source. Le temps nécessaire à la transition du mode veille de permutation à la nouvelle source ne ***Devoir/Doit/Doivent*** pas être supérieur à ***tNewSrc***.

7.2.8 Fonctionnement en courant de crête du destinataire

Les destinataires ne ***Devoir/Doit/Doivent*** utiliser la capacité de surcharge d'une source que lorsque les bits correspondants du courant crête du PDO à alimentation fixe sont définis sur 01b, 10b et 11b (voir 6.4.1.2.2.7). Les destinataires ***Devoir/Doit/Doivent*** gérer les aspects thermiques de l'événement de surcharge en ne dépassant pas la sortie moyenne négociée d'une alimentation fixe prenant en charge le fonctionnement en courant de crête.

Les destinataires qui dépendent de la capacité de courant de crête pour améliorer les performances du système ***Devoir/Doit/Doivent*** également fonctionner correctement lorsqu'ils sont branchés à une source qui n'offre pas la capacité de courant de crête, ou lorsque la capacité de courant de crête a été inhibée par la source.

7.2.9 Fonctionnement robuste du destinataire

7.2.9.1 Décharge de la capacité de masse du destinataire au débranchement

Lorsqu'une source est débranchée d'un destinataire, le destinataire ***Devoir/Doit/Doivent*** continuer à être alimenté par sa capacité de masse d'entrée, jusqu'à ce que V_{BUS} soit déchargée à ***vSafe5V*** ou moins, dans un temps inférieur à ***tSafe5V*** à compter de l'événement de débranchement. Cette exigence de sécurité relative au destinataire ***Devoir/Doit/Doivent*** s'appliquer à tous les destinataires qui fonctionnent avec un niveau V_{BUS} négocié supérieur à ***vSafe5V***, et elle ***Devoir/Doit/Doivent*** s'appliquer à tous les modes de fonctionnement de faible puissance et de grande puissance du destinataire.

Si le débranchement est détecté au cours d'un état de faible puissance du destinataire, comme une veille USB, le destinataire peut alors prélever autant de puissance que nécessaire de sa capacité de masse, puisque plus aucune source n'est branchée. Afin de réussir une détection de débranchement fondée sur la chute du niveau de tension V_{BUS}, la consommation électrique du destinataire ***Devoir/Doit/Doivent*** être suffisamment élevée pour que V_{BUS} tombe bien au-dessous de ***vSrcValid*** (min) dans un délai ***tSafe5V*** après le retrait de la capacité de masse de la source dû au débranchement. Lorsque V_{BUS} a suffisamment diminué, un circuit de décharge peut être activé pour satisfaire à l'exigence de sécurité du destinataire.

Pour représenter ce point, l'ensemble suivant de conditions du destinataire ne satisfait pas aux exigences de sécurité sans circuit de décharge supplémentaire:

- V_{BUS} négociée = 20 V;
- tension V_{BUS} maximale admissible fournie = 21,55V;
- capacité de masse maximale = 30 µF;
- consommation électrique au débranchement = 12,5 mW.

Au moment du débranchement (et donc du retrait de la capacité de masse de la source), la consommation électrique de 12,5 mW fait baisser la tension V_{BUS} du niveau maximal le plus défavorable (21,55 V) à 17 V en environ 205 ms. A ce stade, avec V_{BUS} bien au-dessous de ***vSrcValid*** (min), un circuit de décharge d'environ 100 mW peut être activé pour augmenter la vitesse de décharge de la capacité de masse du destinataire et satisfaire aux exigences de sécurité du destinataire. Le niveau de puissance du circuit de décharge dépend du temps restant pour décharger la tension restante de la capacité de masse du destinataire. Si un destinataire est en mesure de détecter le débranchement d'une autre manière et en moins de temps que ***tSafe5V***, cette autre méthode de détection peut être utilisée pour activer un circuit de décharge, permettant une dissipation de puissance encore plus faible dans les modes de faible puissance tels que la veille USB.

Dans la plupart des applications, l'exigence de sécurité du destinataire limite sa capacité de masse maximale bien au-dessous de la limite ***cSnkBulkPd***. Un débranchement survenant pendant les modes de fonctionnement à haute puissance du destinataire doit rapidement décharger la capacité de masse de celui-ci à ***vSafe5V*** ou moins, tant que le destinataire continue à prélever suffisamment de puissance, jusqu'à ce que V_{BUS} tombe à ***vSafe5V*** ou moins.

7.2.9.2 Protection contre les surtensions d'entrée

Les destinataires ***Devoir/Doit/Doivent*** mettre en œuvre une protection contre les surtensions d'entrée pour éviter les dommages dus à une tension d'entrée supérieure à leur capacité de traitement de la tension. La définition de la capacité de traitement de la tension est laissée libre selon la mise en œuvre du destinataire. La réponse à la surtension ***Ne doit/doivent pas*** interférer avec le niveau de tension V_{BUS} négocié.

Il convient d'/de/qu'/que les destinataires tentent d'envoyer un message ***Hard Reset*** lorsque la protection contre les surtensions se déclenche, suivi d'un message ***Alert*** indiquant un événement de protection contre les surtensions après établissement d'un contrat explicite. La réponse à la protection contre les surtensions ***Pouvoir/Peut/Peuvent*** se déclencher au niveau du port ou du système. Les systèmes ou les ports qui ont déclenché une protection contre les surtensions ***Devoir/Doit/Doivent*** reprendre leur fonctionnement par défaut lorsque la source a rétabli ***vSafe5V*** sur V_{BUS}.

Le destinataire ***Devoir/Doit/Doivent*** être en mesure de renégocier avec la source après la reprise du fonctionnement par défaut. La manière de répondre à une renégociation après un événement de surtension est laissée libre selon la mise en œuvre du destinataire.

Le destinataire ***Devoir/Doit/Doivent*** empêcher la continuité cyclique du système ou du port si la protection contre les surtensions continue à se déclencher après la reprise initiale du fonctionnement par défaut ou de la renégociation. Le verrouillage du port ou du système est une réponse acceptable à une surtension récurrente.

7.2.9.3 Protection contre les surchauffes

Les destinataires ***Devoir/Doit/Doivent*** mettre en œuvre une protection contre les surchauffes pour éviter les dommages causés par une température qui dépasse la capacité thermique de la source. La définition de la capacité thermique et les emplacements de surveillance utilisés pour déclencher la protection contre les surchauffes sont laissés libres selon la mise en œuvre du destinataire.

Les destinataires ***Devoir/Doit/Doivent*** tenter d'envoyer un message ***Hard Reset*** lorsque la protection contre les surchauffes se déclenche, suivi d'un message ***Alert*** indiquant un événement OTP après établissement d'un contrat explicite. La réponse à la protection contre les surchauffes ***Pouvoir/Peut/Peuvent*** se déclencher au niveau du port ou du système. ***Il convient d'/de/qu'/que*** les systèmes ou les ports qui ont déclenché la protection contre les surchauffes tentent de reprendre leur fonctionnement par défaut après un refroidissement suffisant, et ils ***Pouvoir/Peut/Peuvent*** se verrouiller pour se protéger. La définition d'un refroidissement suffisant est laissée libre selon la mise en œuvre du destinataire.

Le destinataire ***Devoir/Doit/Doivent*** être en mesure de renégocier avec la source après la reprise du fonctionnement par défaut. La manière de répondre à une renégociation après un événement de surchauffe est laissée libre selon la mise en œuvre du destinataire.

Le destinataire ***Devoir/Doit/Doivent*** empêcher la continuité cyclique du système ou du port si la protection contre les surchauffes continue à se déclencher après la reprise initiale du fonctionnement par défaut ou de la renégociation. Le verrouillage du port ou du système est une réponse acceptable à une surchauffe récurrente.

7.2.9.4 Protection contre les surintensités

Les destinataires qui fonctionnent avec une alimentation programmable ***Devoir/Doit/Doivent*** mettre en œuvre leur propre mécanisme interne de protection du courant pour se protéger contre les défauts d'intensité V_{BUS} internes ainsi que contre la régulation erratique du courant de la source. Le destinataire ne ***Devoir/Doit/Doivent*** jamais prélever un courant plus élevé que la valeur du courant maximal de l'APDO PPS.

7.2.10 Permutation rapide des rôles

Comme décrit en 7.1.13, une permutation rapide des rôles limite l'interruption de la puissance V_{BUS} à un accessoire alimenté par bus connecté à un hub DFP, lui-même équipé d'un UFP branché à une source d'alimentation et d'une alimentation double fonction branchée à un port hôte prenant en charge l'alimentation double fonction. Cette configuration est représentée à la Figure 7-14 Puissance V_{BUS} au cours d'une permutation rapide des rôles.

L'hôte DRP, après avoir établi un contrat explicite, ***Devoir/Doit/Doivent*** interroger les capacités de destinataire de la source initiale pour déterminer si la source initiale prend en charge la permutation rapide des rôles, ainsi que le niveau de courant dont elle a besoin. Si au moins un bit de permutation rapide des rôles est défini dans le message ***Sink_Capabilities*** reçu de la source initiale, et si l'hôte DRP est en mesure de délivrer le courant demandé à 5 V, l'hôte DRP peut s'armer pour la permutation rapide des rôles. Si l'hôte DRP n'a pas interrogé les capacités de destinataire de la source initiale, ou si le message ***Sink_Capabilities*** indique que la permutation rapide des rôles n'est pas prise en charge ou un courant supérieur à ce que l'hôte DRP est capable de fournir ou souhaite fournir en cas de permutation rapide des rôles, l'hôte DRP ***Ne doit/doivent pas*** s'armer pour la permutation rapide des rôles et ***Devoir/Doit/Doivent Ignorer*** tout signal de permutation rapide des rôles qui peut être détecté.

Lorsque l'hôte DRP qui prend en charge la permutation rapide des rôles détecte le signal de permutation, l'hôte DRP ***Devoir/Doit/Doivent*** arrêter de recevoir du courant et il ***Devoir/Doit/Doivent*** être prêt et en mesure de délivrer ***vSafe5V*** si le niveau de tension V_{BUS} résiduelle au connecteur de l'hôte DRP est supérieur à ***vSafe5V***. Lorsque le niveau de tension V_{BUS} résiduelle au connecteur de l'hôte DRP passe sous ***vSafe5V*** (min), l'hôte DRP, en tant que nouvelle source, ***Devoir/Doit/Doivent*** fournir ***vSafe5V*** au hub DRP dans un délai ***tSrcFRSwap***. L'hôte DRP ***Ne doit/doivent pas*** activer le circuit de décharge V_{BUS} lors du changement du rôle de destinataire initial en nouvelle source.

La nouvelle source ***Devoir/Doit/Doivent*** fournir ***vSafe5V*** au courant USB de type C (voir ***[USB Type-C 2.0]***) à la valeur annoncée dans le champ Fast Role Swap USB Type-C Current (voir 6.4.1.3.1.6). Toutes les exigences relatives à la source ***Devoir/Doit/Doivent*** s'appliquer à la nouvelle source à l'issue de la permutation rapide des rôles. La réponse à la permutation rapide des rôles du hub DRP est décrite en 7.1.13, dans la mesure où le hub DRP fonctionne comme la source initiale avant la permutation rapide des rôles.

Lorsque l'hôte DRP a fourni la puissance V_{BUS} au hub DRP, un message ***PS_RDY*** ***Devoir/Doit/Doivent*** être envoyé au hub DRP, comme défini par la séquence de signaux et de messages de permutation rapide des rôles décrite en 7.3.15.

7.3 Transitions

Les sections suivantes indiquent la réponse de l'alimentation aux différents types de négociations. Les cas de négociation pris en compte pour les exemples sont les suivants:

- transitions vers une puissance plus élevée:
  - augmenter le courant;
  - augmenter la tension;
  - augmenter la tension et le courant;
- transitions vers une puissance relativement constante:
  - augmenter la tension et diminuer le courant;
  - diminuer la tension et augmenter le courant;
- transitions vers une puissance plus faible:
  - diminuer le courant;
  - diminuer la tension;
  - diminuer la tension et le courant;
- transitions vers une permutation des rôles d'alimentation:
  - la source demande une permutation des rôles d'alimentation;
  - le destinataire demande une permutation des rôles d'alimentation;
- transition vers le courant minimal;
- réponse au signal ***Hard Reset***:
  - la source émet un signal ***Hard Reset***;
  - le destinataire émet un signal ***Hard Reset***;
- aucune modification du courant ou de la tension.

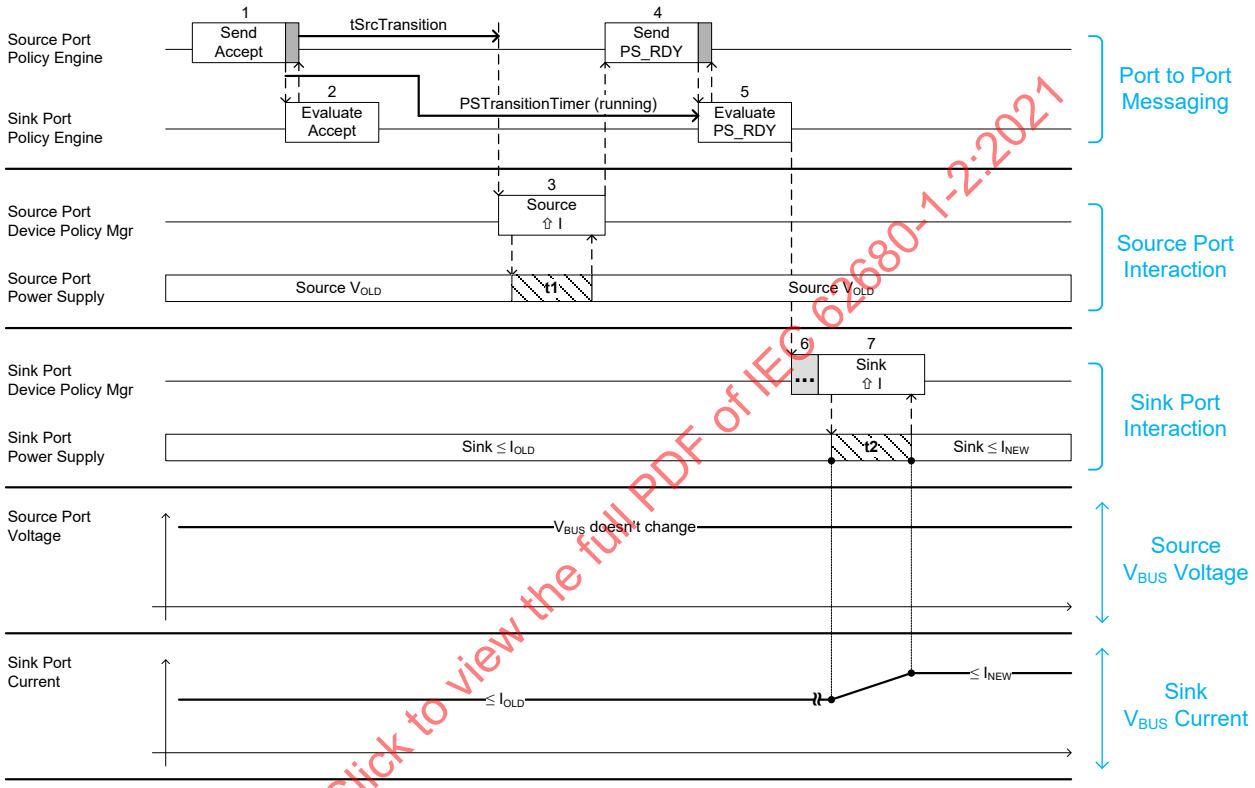
La transition du fonctionnement ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** ou ***[USBBC 1.2]*** vers le mode de distribution de puissance peut également entraîner une transition de puissance, puisqu'il s'agit de la négociation initiale du contrat. Les types de transitions de puissance suivants ***Devoir/Doit/Doivent*** également être appliqués lors du passage du fonctionnement ***[USB 2.0]***, ***[USB 3.2]***, ***[USB Type-C 2.0]*** ou ***[USBBC 1.2]*** vers le mode de distribution de puissance:

- puissance élevée;
- puissance relativement constante;
- transitions vers une puissance plus faible;
- aucune modification du courant ou de la tension.

7.3.1 Augmentation du courant

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectée lors de l'augmentation du courant est représentée à la Figure 7-20. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-1. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22 et le Tableau 7-23. Noter que sur cette figure, le destinataire a déjà envoyé un message ***Request*** à la source.

Figure 7-20 Diagramme de transition pour l'augmentation du courant



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Send Accept	Envoyer Accept
Send PS_RDY	Envoyer PS_RDY
Sink Port Policy Engine	Moteur de politique du port destinataire
Evaluate Accept	Evaluer Accept
PSTransitionTimer (running)	PSTransitionTimer (exécution)
Evaluate PS_RDY	Evaluer PS_RDY
Port to Port Messaging	Messagerie port à port
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Source V_{OLD}	V_{OLD} source
Source Port Interaction	Interaction du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire

Sink	Destinataire
Sink Port Power Supply	Alimentation du port destinataire
Sink Port Interaction	Interaction du port destinataire
Source Port Voltage	Tension du port source
V_{BUS} doesn't change	V_{BUS} ne change pas
Source V_{BUS} Voltage	Tension V_{BUS} source
Sink Port Current	Courant du port destinataire
Sink V_{BUS} Current	Courant V_{BUS} destinataire

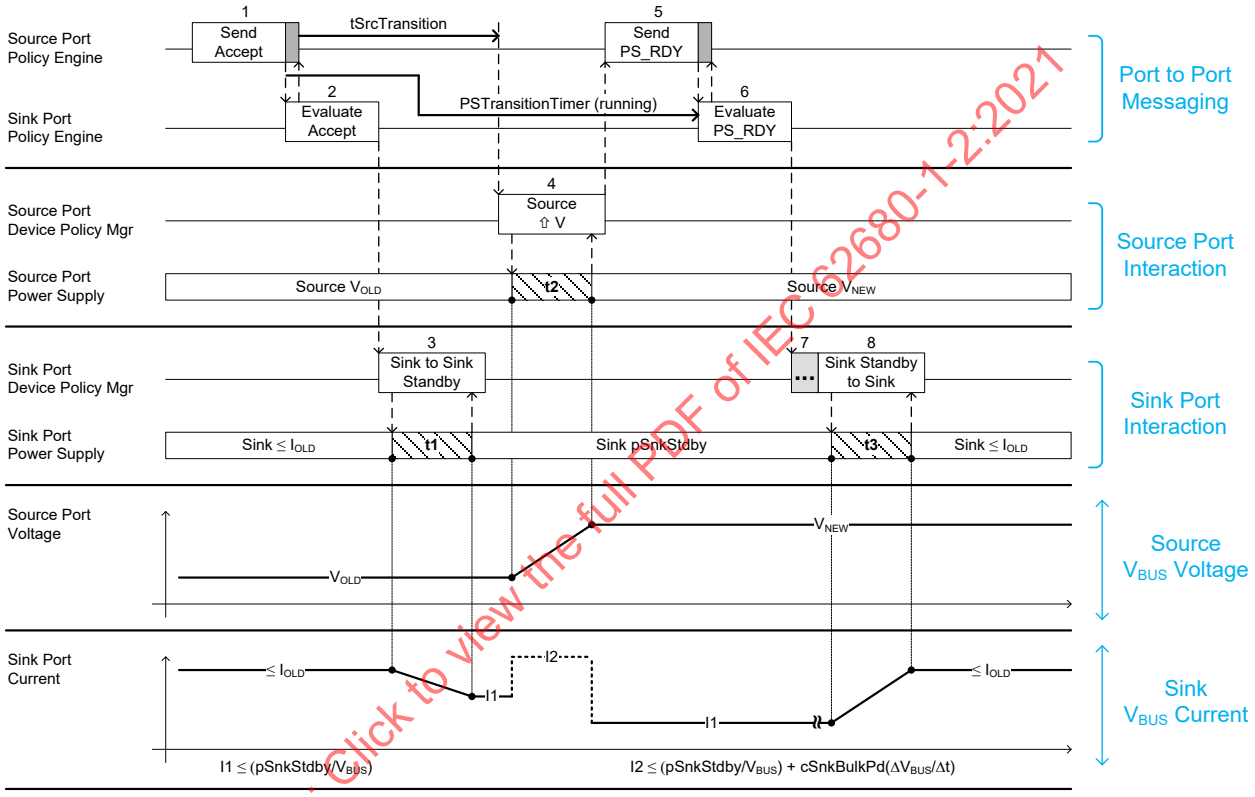
Tableau 7-1 Description de séquence d'augmentation du courant

Etape	Port source	Port destinataire
1	Le moteur de politique envoie le message ***Accept*** au destinataire.	Le moteur de politique reçoit le message ***Accept*** et lance le ***PSTransitionTimer***.
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire. Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de modifier sa puissance de sortie.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***Accept***.
3	***tSrcTransition*** après la réception du message ***GoodCRC***, l'alimentation commence à modifier la capacité de sa puissance de sortie. L'alimentation ***Devoir/Doit/Doivent*** être prête à fonctionner au nouveau niveau de puissance dans un délai ***tSrcReady*** (t1). L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle est prête à fonctionner au nouveau niveau de puissance. Le statut de l'alimentation est transmis au moteur de politique.	
4	Le moteur de politique envoie le message ***PS_RDY*** au destinataire.	Le moteur de politique reçoit le message ***PS_RDY*** de la source.
5	La couche protocole reçoit le message ***GoodCRC*** du destinataire.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***PS_RDY*** provenant de la source, et indique au gestionnaire de politique d'utilisation des dispositifs qu'il est d'accord pour fonctionner au nouveau niveau de puissance.
6		Le destinataire ***Pouvoir/Peut/Peuvent*** commencer à fonctionner au nouveau niveau de puissance à tout moment après l'évaluation du message ***PS_RDY***. Cette durée est indéterminée.
7		Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau. La durée (t2) dépend de l'amplitude de la variation de charge.

7.3.2 Augmentation de la tension

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectée lors de l'augmentation de la tension est représentée à la Figure 7-21. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-2. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22, le Tableau 7-23 et le Tableau 7-24. Noter que sur cette figure, le destinataire a déjà envoyé un message ***Request*** à la source.

Figure 7-21 Diagramme de transition pour l'augmentation de la tension



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Send Accept	Envoyer Accept
Send PS_RDY	Envoyer PS_RDY
Sink Port Policy Engine	Moteur de politique du port destinataire
Evaluate Accept	Evaluer Accept
PSTransitionTimer (running)	PSTransitionTimer (exécution)
Evaluate PS_RDY	Evaluer PS_RDY
Port to Port Messaging	Messagerie port à port
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Source V_{OLD}	V_{OLD} source
Source V_{NEW}	V_{NEW} source

Source Port Interaction	Interaction du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink to Sink Standby	Destinataire à destinataire en attente
Sink Standby to Sink	Destinataire en attente à destinataire
Sink Port Power Supply	Alimentation du port destinataire
Sink pSnkStdby	Destinataire pSnkStdby
Sink Port Interaction	Interaction du port destinataire
Source Port Voltage	Tension du port source
Source V_{BUS} Voltage	Tension V_{BUS} source
Sink Port Current	Courant du port destinataire
Sink V_{BUS} Current	Courant V_{BUS} destinataire

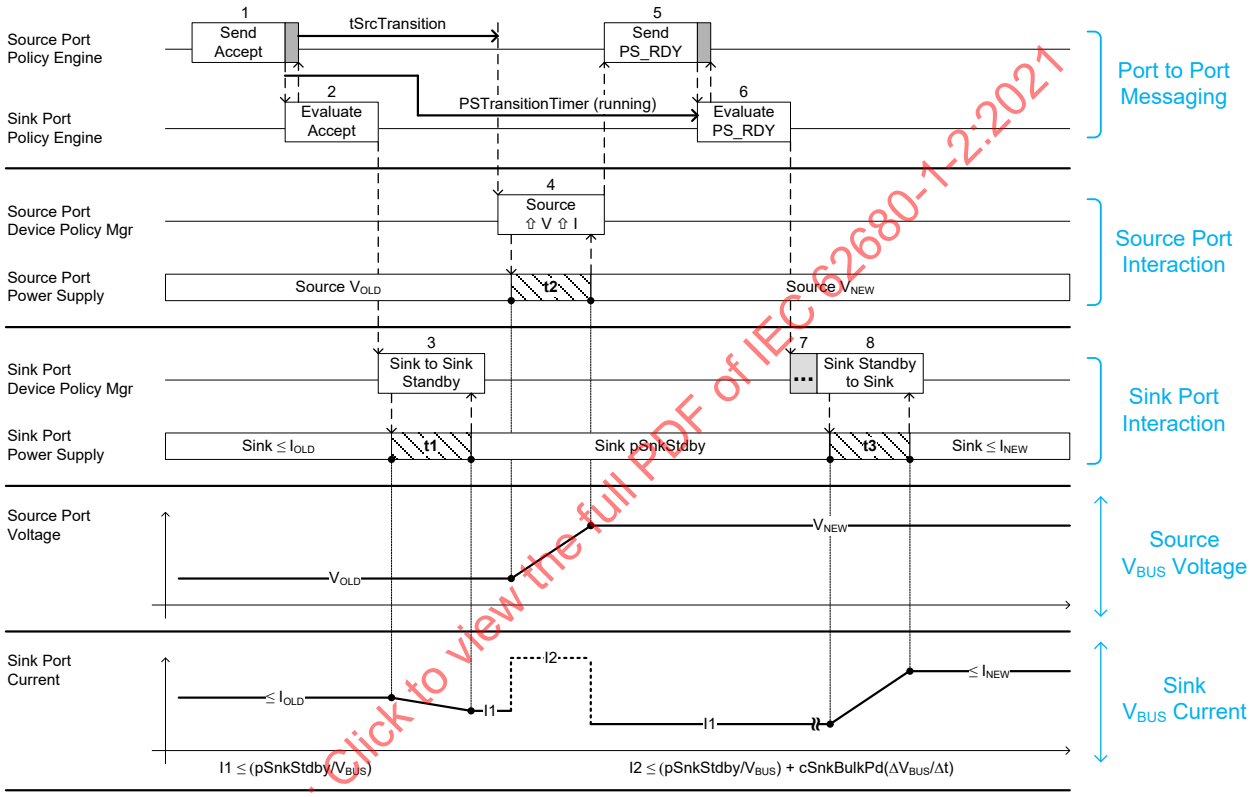
Tableau 7-2 Description de séquence d'augmentation de la tension

Etape	Port source	Port destinataire
1	Le moteur de politique envoie le message ***Accept*** au destinataire.	Le moteur de politique reçoit le message ***Accept*** et lance le ***PSTransitionTimer***.
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire. Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de modifier sa puissance de sortie.	La couche protocole envoie le message ***GoodCRC*** à la source. Moteur de politique. Le moteur de politique évalue ensuite le message ***Accept***.
3		Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de réduire sa consommation électrique à ***pSnkStdby*** dans un délai ***tSnkStdby*** (t1); t1 ***Devoir/Doit/Doivent*** s'achever avant ***tSrcTransition***. Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau.
4	***tSrcTransition*** après la réception du message ***GoodCRC***, l'alimentation commence à modifier la capacité de sa puissance de sortie. L'alimentation ***Devoir/Doit/Doivent*** être prête à fonctionner au nouveau niveau de puissance dans un délai ***tSrcReady*** (t2). L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle est prête à fonctionner au nouveau niveau de puissance. Le statut de l'alimentation est transmis au moteur de politique.	
5	Le moteur de politique envoie le message ***PS_RDY*** au destinataire.	Le moteur de politique reçoit le message ***PS_RDY*** de la source.
6	La couche protocole reçoit le message ***GoodCRC*** du destinataire.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***PS_RDY*** provenant de la source, et indique au gestionnaire de politique d'utilisation des dispositifs qu'il est d'accord pour fonctionner au nouveau niveau de puissance.
7		Le destinataire ***Pouvoir/Peut/Peuvent*** commencer à fonctionner au nouveau niveau de puissance à tout moment après l'évaluation du message ***PS_RDY***. Cette durée est indéterminée.
8		Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau. La durée (t3) dépend de l'amplitude de la variation de charge.

7.3.3 Augmentation de la tension et du courant

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectées lors de l'augmentation de la tension et du courant est représentée à la Figure 7-22. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-3. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22, le Tableau 7-23 et le Tableau 7-24. Noter que sur cette figure, le destinataire a déjà envoyé un message ***Request*** à la source.

Figure 7-22 Diagramme de transition pour l'augmentation de la tension et du courant



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Send Accept	Envoyer Accept
Send PS_RDY	Envoyer PS_RDY
Sink Port Policy Engine	Moteur de politique du port destinataire
Evaluate Accept	Evaluer Accept
PSTransitionTimer (running)	PSTransitionTimer (exécution)
Evaluate PS_RDY	Evaluer PS_RDY
Port to Port Messaging	Messagerie port à port
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Source V_{OLD}	V_{OLD} source
Source V_{NEW}	V_{NEW} source

Source Port Interaction	Interaction du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink to Sink Standby	Destinataire à destinataire en attente
Sink Standby to Sink	Destinataire en attente à destinataire
Sink Port Power Supply	Alimentation du port destinataire
Sink pSnkStdby	Destinataire pSnkStdby
Sink Port Interaction	Interaction du port destinataire
Source Port Voltage	Tension du port source
Source V_{BUS} Voltage	Tension V_{BUS} source
Sink Port Current	Courant du port destinataire
Sink V_{BUS} Current	Courant V_{BUS} destinataire

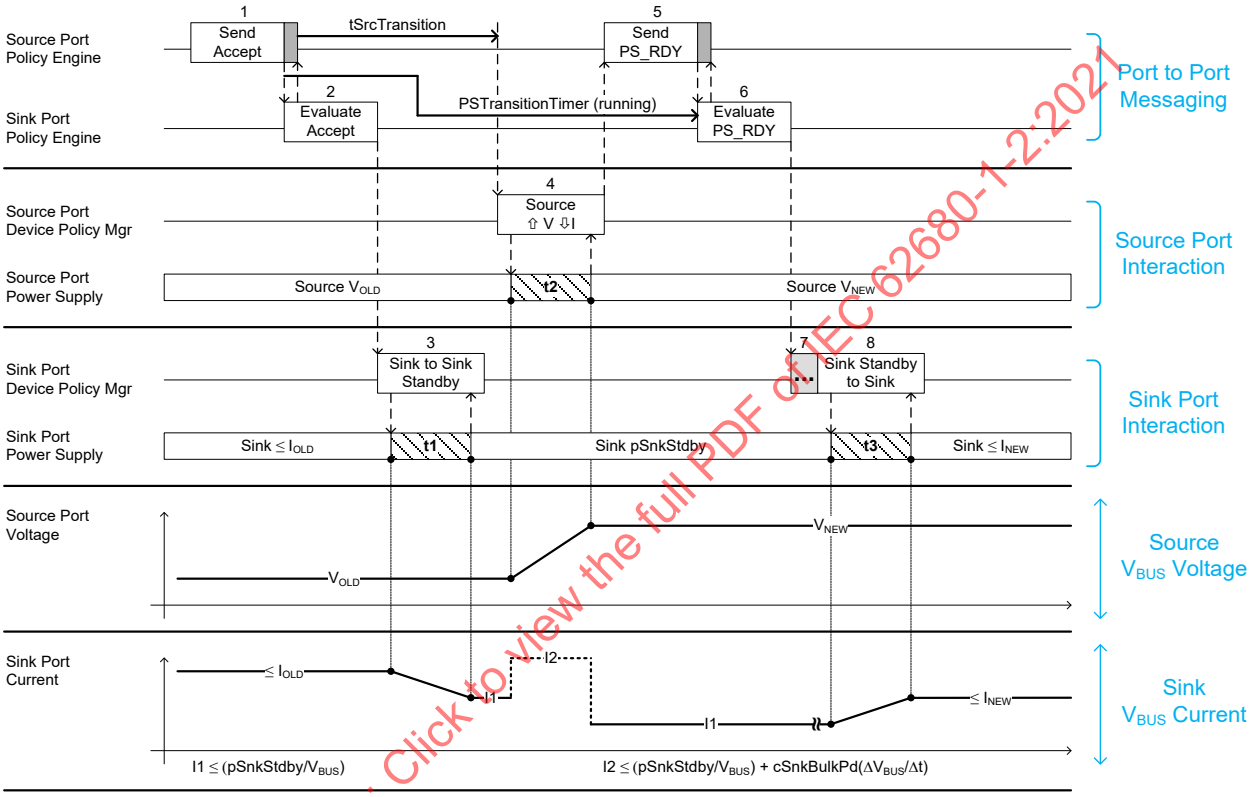
Tableau 7-3 Description de séquence d'augmentation de la tension et du courant

Etape	Port source	Port destinataire
1	Le moteur de politique envoie le message ***Accept*** au destinataire.	Le moteur de politique reçoit le message ***Accept*** et lance le ***PSTransitionTimer***.
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire. Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de modifier sa puissance de sortie.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***Accept.***
3		Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de réduire sa consommation électrique à ***pSnkStdby*** dans un délai ***tSnkStdby*** (t1); t1 ***Devoir/Doit/Doivent*** s'achever avant ***tSrcTransition***. Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau.
4	***tSrcTransition*** après la réception du message ***GoodCRC***, l'alimentation commence à modifier la capacité de sa puissance de sortie. L'alimentation ***Devoir/Doit/Doivent*** être prête à fonctionner au nouveau niveau de puissance dans un délai ***tSrcReady*** (t2). L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle est prête à fonctionner au nouveau niveau de puissance. Le statut de l'alimentation est transmis au moteur de politique.	
5	Le moteur de politique envoie le message ***PS_RDY*** au destinataire.	Le moteur de politique reçoit le message ***PS_RDY*** de la source.
6	La couche protocole reçoit le message ***GoodCRC*** du destinataire.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***PS_RDY*** provenant de la source, et indique au gestionnaire de politique d'utilisation des dispositifs qu'il est d'accord pour fonctionner au nouveau niveau de puissance.
7		Le destinataire ***Pouvoir/Peut/Peuvent*** commencer à fonctionner au nouveau niveau de puissance à tout moment après l'évaluation du message ***PS_RDY***. Cette durée est indéterminée.
8		Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau. La durée (t3) dépend de l'amplitude de la variation de charge.

7.3.4 Augmentation de la tension et diminution du courant

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectées lors de l'augmentation de la tension et la diminution du courant est représentée à la Figure 7-23. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-4. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22, le Tableau 7-23 et le Tableau 7-24. Noter que sur cette figure, le destinataire a déjà envoyé un message ***Request*** à la source.

Figure 7-23 Diagramme de transition pour l'augmentation de la tension et la diminution du courant



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink Port Power Supply	Alimentation du port destinataire
Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Send Accept	Envoyer Accept
Evaluate Accept	Evaluer Accept
Source	Source
Sink	Destinataire

Sink to Sink Standby	Destinataire à destinataire en attente
Send	Envoyer
Evaluate	Evaluer
Sink Standby	Destinataire en attente
Port to Port Messaging	Messagerie port à port
Source Port Interaction	Interaction du port source
Sink Port Interaction	Interaction du port destinataire
Source V_{BUS} Voltage	Tension V_{BUS} source
Sink V_{BUS} Current	Courant V_{BUS} destinataire

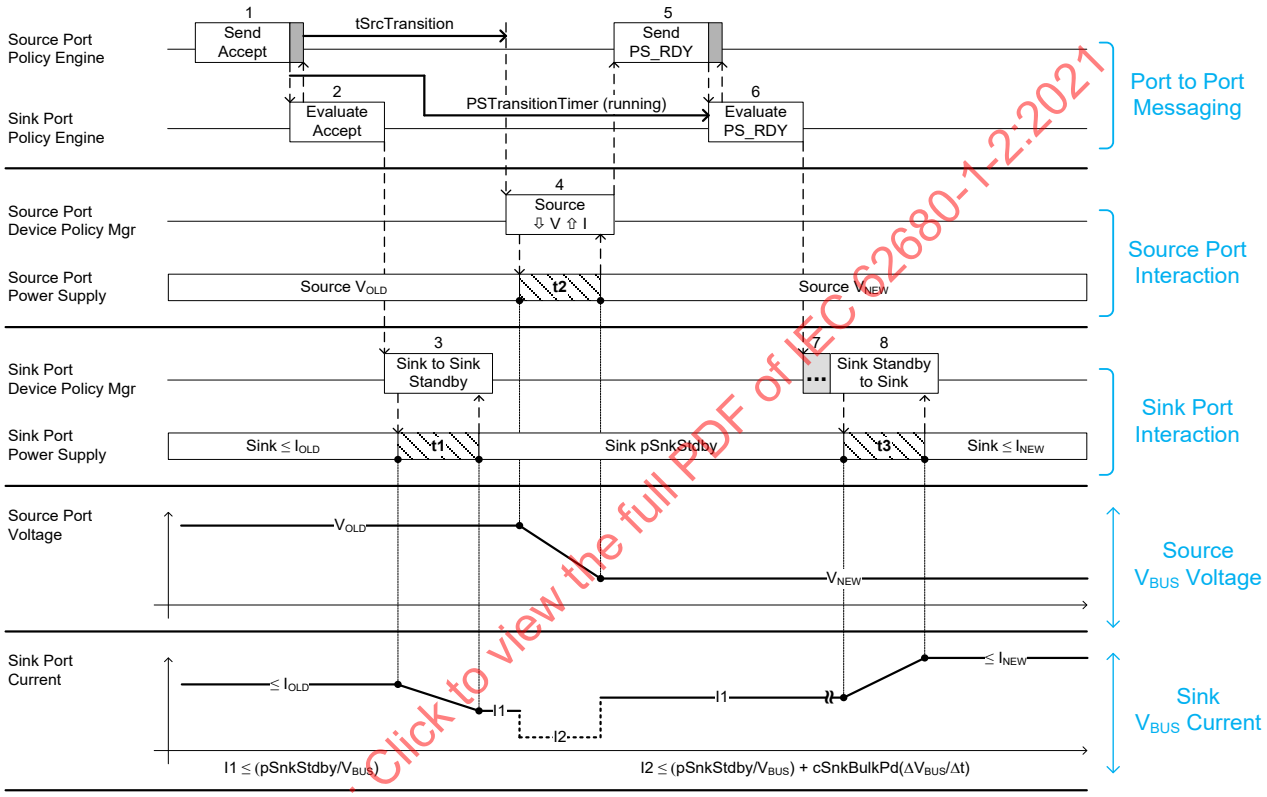
Tableau 7-4 Description de séquence d'augmentation de la tension et de diminution du courant

Etape	Port source	Port destinataire
1	Le moteur de politique envoie le message ***Accept*** au destinataire.	Le moteur de politique évalue le message ***Accept*** et lance le ***PSTransitionTimer***.
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire. Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de modifier sa puissance de sortie.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***Accept***.
3		Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de réduire sa consommation électrique à ***pSnkStdby*** dans un délai ***tSnkStdby*** (t1); t1 ***Devoir/Doit/Doivent*** s'achever avant ***tSrcTransition***. Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau.
4	***tSrcTransition*** après la réception du message ***GoodCRC***, l'alimentation commence à modifier la capacité de sa puissance de sortie. L'alimentation ***Devoir/Doit/Doivent*** être prête à fonctionner au nouveau niveau de puissance dans un délai ***tSrcReady*** (t2). L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle est prête à fonctionner au nouveau niveau de puissance. Le statut de l'alimentation est transmis au moteur de politique.	
5	Le moteur de politique envoie le message ***PS_RDY*** au destinataire.	Le moteur de politique reçoit le message ***PS_RDY*** de la source.
6	La couche protocole reçoit le message ***GoodCRC*** du destinataire.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***PS_RDY*** provenant de la source, et indique au gestionnaire de politique d'utilisation des dispositifs qu'il est d'accord pour fonctionner au nouveau niveau de puissance.
7		Le destinataire ***Pouvoir/Peut/Peuvent*** commencer à fonctionner au nouveau niveau de puissance à tout moment après l'évaluation du message ***PS_RDY***. Cette durée est indéterminée.
8		Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau. La durée (t3) dépend de l'amplitude de la variation de charge.

7.3.5 Diminution de la tension et augmentation du courant

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectées lors de la diminution de la tension et de l'augmentation du courant est représentée à la Figure 7-24. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-5. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22, le Tableau 7-23 et le Tableau 7-24. Noter que sur cette figure, le destinataire a déjà envoyé un message ***Request*** à la source.

Figure 7-24 Diagramme de transition pour la diminution de la tension et l'augmentation du courant



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink Port Power Supply	Alimentation du port destinataire
Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Send Accept	Envoyer Accept
Evaluate Accept	Evaluer Accept
Source	Source
Sink	Destinataire

Sink to Sink Standby	Destinataire à destinataire en attente
Send	Envoyer
Evaluate	Evaluer
Sink Standby	Destinataire en attente
Port to Port Messaging	Messagerie port à port
(running)	(exécution)
V_{BUS} does not change	V_{BUS} ne varie pas
Source Port Interaction	Interaction du port source
Sink Port Interaction	Interaction du port destinataire
Source V_{BUS} Voltage	Tension V_{BUS} source
Sink V_{BUS} Current	Courant V_{BUS} destinataire

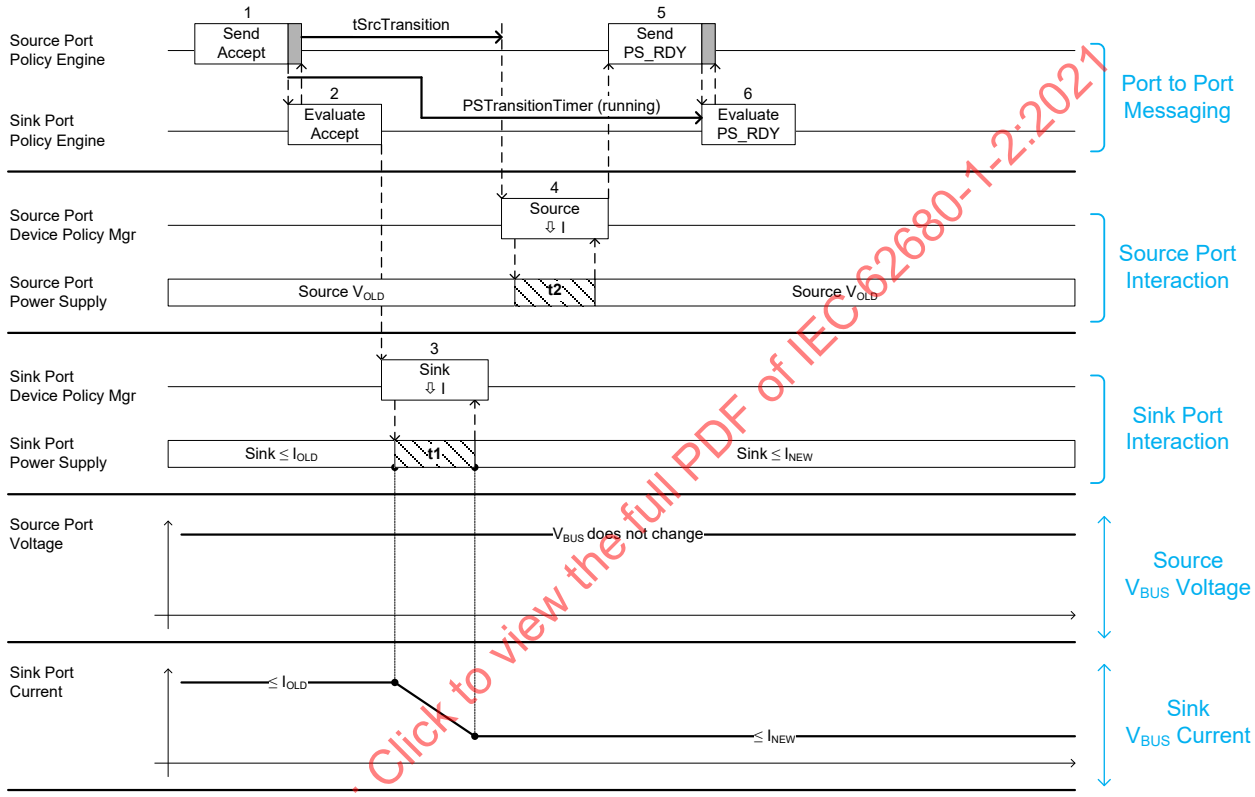
Tableau 7-5 Description de séquence de diminution de la tension et d'augmentation du courant

Etape	Port source	Port destinataire
1	Le moteur de politique envoie le message ***Accept*** au destinataire.	Le moteur de politique reçoit le message ***Accept*** et lance le ***PSTransitionTimer***.
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire. Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de modifier sa puissance de sortie.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***Accept***.
3		Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de réduire sa consommation électrique à ***pSnkStdby*** dans un délai ***tSnkStdby*** (t1); t1 ***Devoir/Doit/Doivent*** s'achever avant ***tSrcTransition***. Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau.
4	***tSrcTransition*** après la réception du message ***GoodCRC***, l'alimentation commence à modifier la capacité de sa puissance de sortie. L'alimentation ***Devoir/Doit/Doivent*** être prête à fonctionner au nouveau niveau de puissance dans un délai ***tSrcReady*** (t2). L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle est prête à fonctionner au nouveau niveau de puissance. Le statut de l'alimentation est transmis au moteur de politique.	
5	Le moteur de politique envoie le message ***PS_RDY*** au destinataire.	Le moteur de politique reçoit le message ***PS_RDY*** de la source.
6	La couche protocole reçoit le message ***GoodCRC*** du destinataire.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***PS_RDY*** provenant de la source, et indique au gestionnaire de politique d'utilisation des dispositifs qu'il est d'accord pour fonctionner au nouveau niveau de puissance.
7		Le destinataire ***Pouvoir/Peut/Peuvent*** commencer à fonctionner au nouveau niveau de puissance à tout moment après l'évaluation du message ***PS_RDY***. Cette durée est indéterminée.
8		Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau. La durée (t3) dépend de l'amplitude de la variation de charge.

7.3.6 Diminution du courant

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectées lors de la diminution du courant est représentée à la Figure 7-25. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-6. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22, le Tableau 7-23 et le Tableau 7-24. Noter que sur cette figure, le destinataire a déjà envoyé un message ***Request*** à la source.

Figure 7-25 Diagramme de transition pour la diminution du courant



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink Port Power Supply	Alimentation du port destinataire
Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Send Accept	Envoyer Accept
Evaluate Accept	Evaluer Accept
Source	Source
Sink	Destinataire

Sink to Sink Standby	Destinataire à destinataire en attente
Send	Envoyer
Evaluate	Evaluer
Sink Standby	Destinataire en attente
Port to Port Messaging	Messagerie port à port
(running)	(exécution)
Source Port Interaction	Interaction du port source
Sink Port Interaction	Interaction du port destinataire
Source V_{BUS} Voltage	Tension V_{BUS} source
Sink V_{BUS} Current	Courant V_{BUS} destinataire
Initial Sink	Destinataire initial
Initial Source	Source initiale
New Source	Nouvelle source
New Sink	Nouveau destinataire

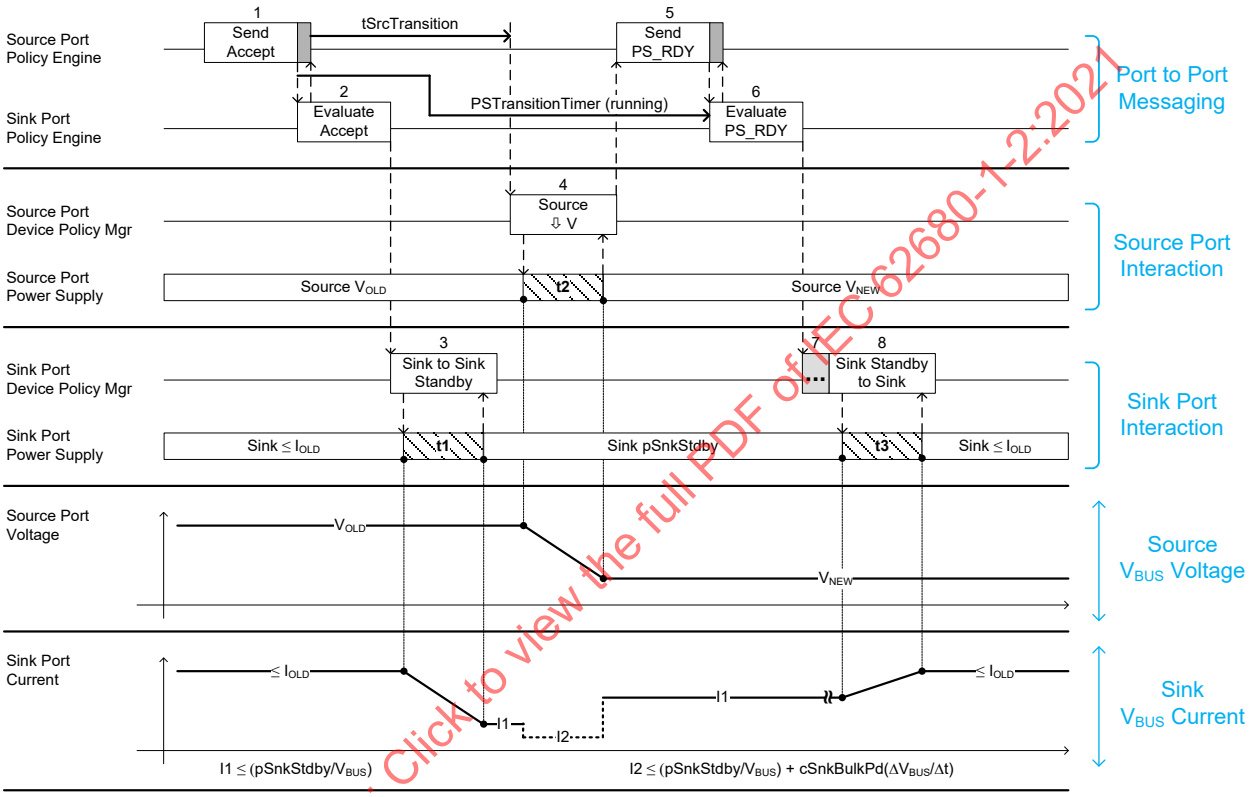
Tableau 7-6 Description de séquence de diminution du courant

Etape	Port source	Port destinataire
1	Le moteur de politique envoie le message ***Accept*** au destinataire.	Le moteur de politique reçoit le message ***Accept*** et lance le ***PSTransitionTimer***.
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire. Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de modifier sa puissance de sortie.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***Accept***. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de réduire la consommation de puissance.
3		Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau. Le destinataire ***Devoir/Doit/Doivent*** être capable de fonctionner à faible courant dans un délai ***tSnkNewPower*** (t1); t1 ***Devoir/Doit/Doivent*** s'achever avant ***tSrcTransition***.
4	***tSrcTransition*** après la réception du message ***GoodCRC***, l'alimentation commence à modifier la capacité de sa puissance de sortie. L'alimentation ***Devoir/Doit/Doivent*** être prête à fonctionner au nouveau niveau de puissance dans un délai ***tSrcReady*** (t2). L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle est prête à fonctionner au nouveau niveau de puissance. Le statut de l'alimentation est transmis au moteur de politique.	
5	Le moteur de politique envoie le message ***PS_RDY*** au destinataire.	Le moteur de politique reçoit le message ***PS_RDY*** de la source.
6	La couche protocole reçoit le message ***GoodCRC*** du destinataire.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue le message ***PS_RDY*** de la source. Le destinataire fonctionne déjà au nouveau niveau de puissance, aucune action supplémentaire n'est donc exigée.

7.3.7 Diminution de la tension

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectée lors de la diminution de la tension est représentée à la Figure 7-26. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-7. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22, le Tableau 7-23 et le Tableau 7-24. Noter que sur cette figure, le destinataire a déjà envoyé un message ***Request*** à la source.

Figure 7-26 Diagramme de transition pour la diminution de la tension



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink Port Power Supply	Alimentation du port destinataire
Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Send Accept	Envoyer Accept
Evaluate Accept	Evaluer Accept
Source	Source
Sink	Destinataire

Sink to Sink Standby	Destinataire à destinataire en attente
Send	Envoyer
Evaluate	Evaluer
Sink Standby	Destinataire en attente
Port to Port Messaging	Messagerie port à port
(running)	(exécution)
Source Port Interaction	Interaction du port source
Sink Port Interaction	Interaction du port destinataire
Source V_{BUS} Voltage	Tension V_{BUS} source
Sink V_{BUS} Current	Courant V_{BUS} destinataire
Initial Sink	Destinataire initial
Initial Source	Source initiale
New Source	Nouvelle source
New Sink	Nouveau destinataire

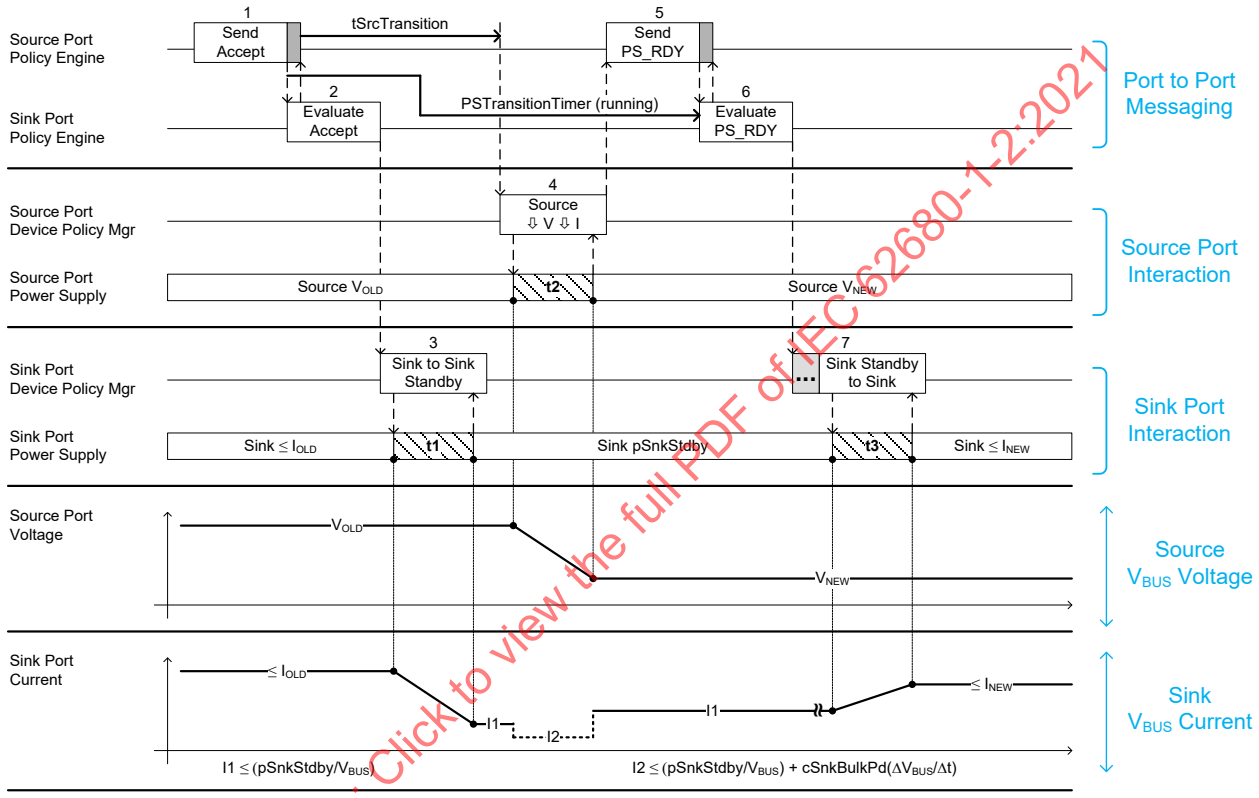
Tableau 7-7 Description de séquence de diminution de la tension

Etape	Port source	Port destinataire
1	Le moteur de politique envoie le message ***Accept*** au destinataire.	Le moteur de politique reçoit le message ***Accept*** et lance le ***PSTransitionTimer***.
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire. Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de modifier sa puissance de sortie.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***Accept*** .
3		Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de réduire sa consommation électrique à ***pSnkStdby*** dans un délai ***tSnkStdby*** (t1); t1 ***Devoir/Doit/Doivent*** s'achever avant ***tSrcTransition***. Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau.
4	***tSrcTransition*** après la réception du message ***GoodCRC***, l'alimentation commence à modifier la capacité de sa puissance de sortie. L'alimentation ***Devoir/Doit/Doivent*** être prête à fonctionner au nouveau niveau de puissance dans un délai ***tSrcReady*** (t2). L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle est prête à fonctionner au nouveau niveau de puissance. Le statut de l'alimentation est transmis au moteur de politique.	
5	Le moteur de politique envoie le message ***PS_RDY*** au destinataire.	Le moteur de politique reçoit le message ***PS_RDY*** de la source.
6	La couche protocole reçoit le message ***GoodCRC*** du destinataire.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***PS_RDY*** provenant de la source, et indique au gestionnaire de politique d'utilisation des dispositifs qu'il est d'accord pour fonctionner au nouveau niveau de puissance.
7		Le destinataire ***Pouvoir/Peut/Peuvent*** commencer à fonctionner au nouveau niveau de puissance à tout moment après l'évaluation du message ***PS_RDY***. Cette durée est indéterminée.
8		Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau. La durée (t3) dépend de l'amplitude de la variation de charge.

7.3.8 Diminution de la tension et du courant

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectées lors de la diminution de la tension et du courant est représentée à la Figure 7-27. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-8. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22, le Tableau 7-23 et le Tableau 7-24. Noter que sur cette figure, le destinataire a déjà envoyé un message ***Request*** à la source.

Figure 7-27 Diagramme de transition pour la diminution de la tension et du courant



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink Port Power Supply	Alimentation du port destinataire
Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Send Accept	Envoyer Accept
Evaluate Accept	Evaluer Accept
Source to Swap Standby	Source vers veille de permutation
Source	Source

Sink	Destinataire
Sink to Swap Standby	Destinataire vers veille de permutation
Swap Standby	Veille de permutation
Swap Standby to Sink	Veille de permutation vers destinataire
Swap Standby to Source	Veille de permutation vers source
Send	Envoyer
Sink Default Current	Courant par défaut du destinataire
Not Driven	Non conduit
Evaluate	Evaluer
Sink Standby	Destinataire en attente
Port to Port Messaging	Messagerie port à port
(running)	(exécution)
Source Port Interaction	Interaction du port source
Sink Port Interaction	Interaction du port destinataire
Source V_{BUS} Voltage	Tension V_{BUS} source
Sink V_{BUS} Current	Courant V_{BUS} destinataire
Initial Sink	Destinataire initial
Initial Source	Source initiale
New Source	Nouvelle source
New Sink	Nouveau destinataire

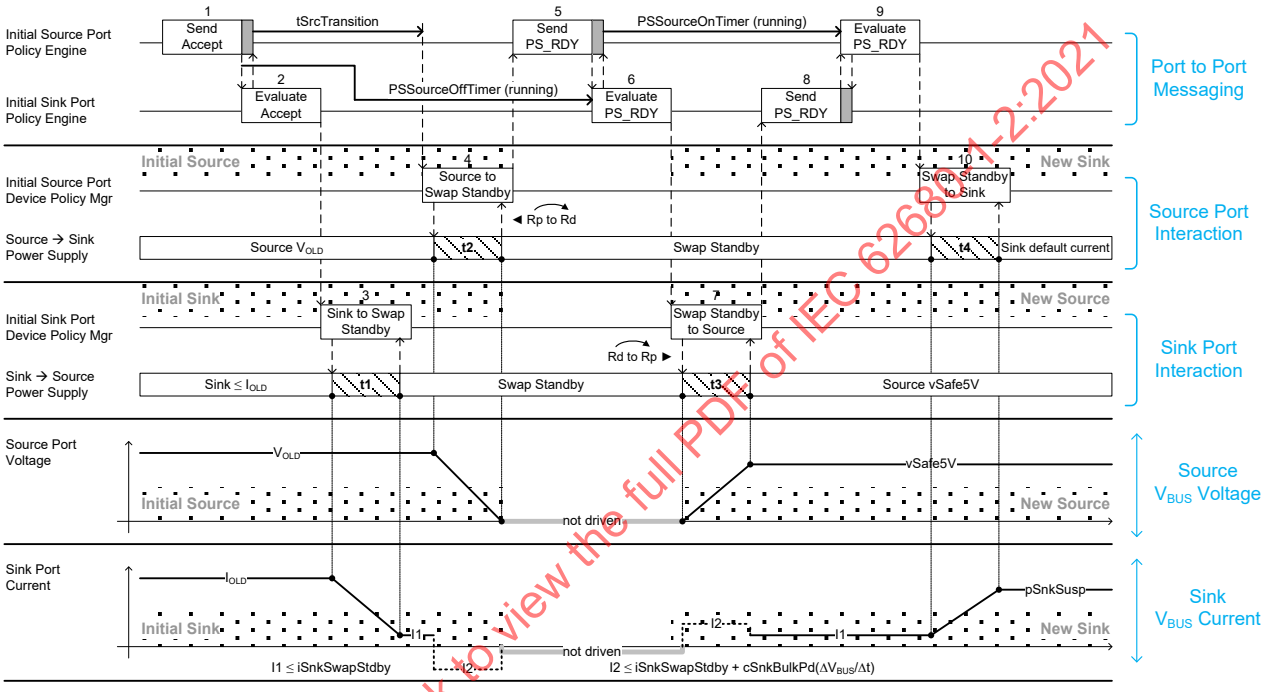
Tableau 7-8 Description de séquence de diminution de la tension et du courant

Etape	Port source	Port destinataire
1	Le moteur de politique envoie le message ***Accept*** au destinataire.	Le moteur de politique reçoit le message ***Accept*** et lance le ***PSTransitionTimer***.
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire. Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de modifier sa puissance de sortie.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***Accept***.
3		Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de réduire sa consommation électrique à ***pSnkStdby*** dans un délai ***tSnkStdby*** (t1); t1 ***Devoir/Doit/Doivent*** s'achever avant ***tSrcTransition***. Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau.
4	***tSrcTransition*** après la réception du message ***GoodCRC***, l'alimentation commence à modifier la capacité de sa puissance de sortie. L'alimentation ***Devoir/Doit/Doivent*** être prête à fonctionner au nouveau niveau de puissance dans un délai ***tSrcReady*** (t2). L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle est prête à fonctionner au nouveau niveau de puissance. Le statut de l'alimentation est transmis au moteur de politique.	
5	Le moteur de politique envoie le message ***PS_RDY*** au destinataire.	Le moteur de politique reçoit le message ***PS_RDY*** de la source.
6	La couche protocole reçoit le message ***GoodCRC*** du destinataire.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***PS_RDY*** provenant de la source, et indique au gestionnaire de politique d'utilisation des dispositifs qu'il est d'accord pour fonctionner au nouveau niveau de puissance.
7		Le destinataire ***Pouvoir/Peut/Peuvent*** commencer à fonctionner au nouveau niveau de puissance à tout moment après l'évaluation du message ***PS_RDY***. Cette durée est indéterminée.
8		Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau. La durée (t3) dépend de l'amplitude de la variation de charge.

7.3.9 Permutation des rôles d'alimentation demandée par le destinataire

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectée lors d'une permutation des rôles d'alimentation demandée par le destinataire est représentée à la Figure 7-28. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-9. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-23. Noter que sur cette figure, le destinataire a déjà envoyé un message ***PR_Swap*** à la source.

Figure 7-28 Diagramme de transition pour une permutation des rôles d'alimentation demandée par le destinataire



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink Port Power Supply	Alimentation du port destinataire
Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Send Accept	Envoyer Accept
Evaluate Accept	Evaluer Accept
Source to Swap Standby	Source vers veille de permutation
Source	Source
Sink	Destinataire
Sink to Swap Standby	Destinataire vers veille de permutation

Swap Standby	Veille de permutation
Swap Standby to Sink	Veille de permutation vers destinataire
Swap Standby to Source	Veille de permutation vers source
Send	Envoyer
Sink Default Current	Courant par défaut du destinataire
Not Driven	Non conduit
Evaluate	Evaluer
Sink Standby	Destinataire en attente
Port to Port Messaging	Messagerie port à port
(running)	(exécution)
Source Port Interaction	Interaction du port source
Sink Port Interaction	Interaction du port destinataire
Source V_{BUS} Voltage	Tension V_{BUS} source
Sink V_{BUS} Current	Courant V_{BUS} destinataire
Initial Sink	Destinataire initial
Initial Source	Source initiale
New Source	Nouvelle source
New Sink	Nouveau destinataire

Tableau 7-9 Description de séquence pour une permutation des rôles d'alimentation demandée par le destinataire

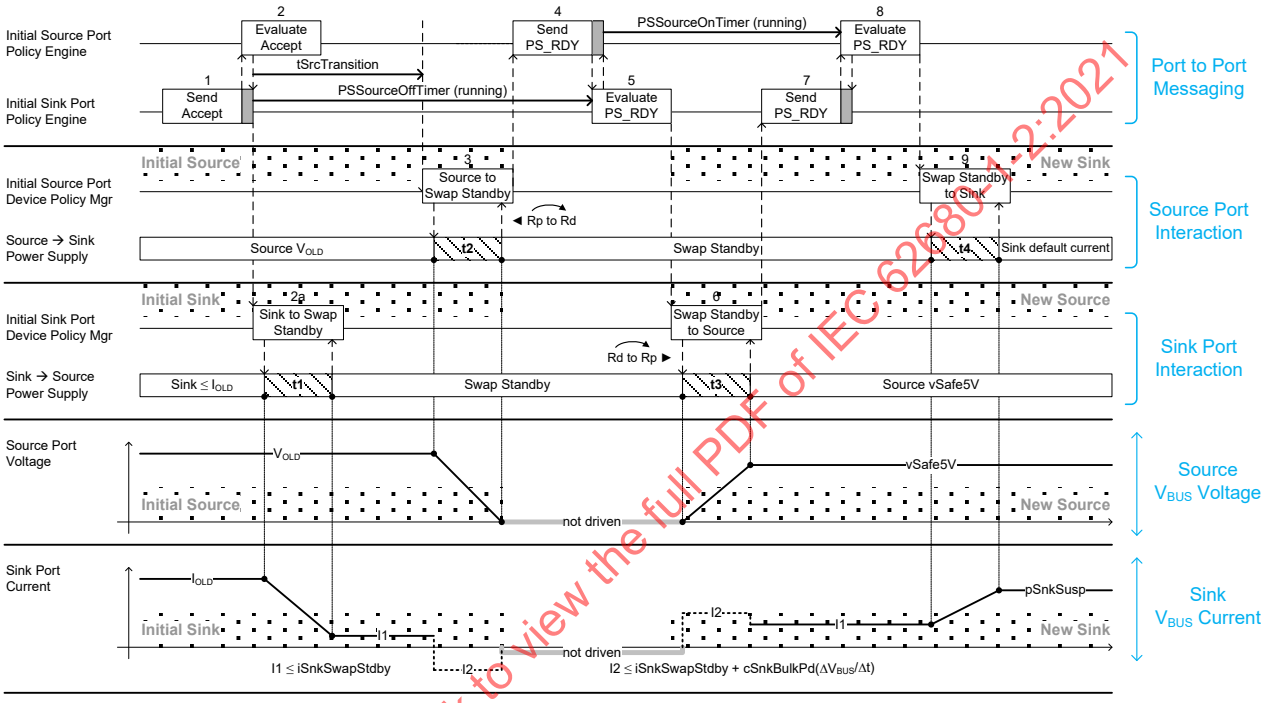
Etape	Port source initial → Nouveau port destinataire	Port destinataire initial → Nouveau port source
1	Le moteur de politique envoie le message ***Accept*** au destinataire initial.	Le moteur de politique reçoit le message ***Accept*** et lance le ***PSSourceOffTimer***.
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire. Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de modifier sa puissance de sortie.	La couche protocole envoie le message ***GoodCRC*** à la source initiale. Le moteur de politique évalue ensuite le message ***Accept***.
3		Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de passer en veille de permutation dans un délai ***tSnkStdby*** (t1); t1 ***Devoir/Doit/Doivent*** s'achever avant ***tSrcTransition***. Si le destinataire est en veille, le destinataire initial ***Ne doit/doivent pas*** consommer plus de ***iSnkSwapStdby*** (I1). Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau.
4	***tSrcTransition*** après la réception du message ***GoodCRC***, l'alimentation commence à passer sa capacité de puissance de sortie en veille de permutation (voir 7.1.10). L'alimentation ***Devoir/Doit/Doivent*** passer en veille de permutation dans un délai ***tSrcSwapStdby*** (t2). L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle est prête à fonctionner en tant que nouveau destinataire. La terminaison CC passe de Rp à Rd (voir ***[USB Type-C 2.0]***). Le statut de l'alimentation est transmis au moteur de politique.	
5	L'alimentation est prête et le moteur de politique envoie le message ***PS_RDY*** au dispositif qui va devenir la nouvelle source.	
6	La couche protocole reçoit le message ***GoodCRC*** provenant du dispositif qui va devenir la nouvelle source. Le moteur de politique lance le ***PSSourceOnTimer***. Dès l'envoi du message ***PS_RDY*** et la réception du message ***GoodCRC*** , la source initiale est prête à être le nouveau destinataire.	Le moteur de politique arrête le ***PSSourceOffTimer.*** La couche protocole envoie le message ***GoodCRC*** au nouveau destinataire. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de fonctionner en tant que nouvelle source.
7		La terminaison CC passe de Rd à Rp (voir ***[USB Type-C 2.0]***). En tant que nouvelle source, l'alimentation passe du mode veille de permutation à l'alimentation de ***vSafe5V*** par défaut dans un délai ***tNewSrc*** (t3). L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle fonctionne en tant que nouvelle source.
8	Le moteur de politique reçoit le message ***PS_RDY*** de la source.	Le gestionnaire de politique d'utilisation des dispositifs informe le moteur de politique que l'alimentation est prête, et le moteur de politique envoie le message ***PS_RDY*** au nouveau destinataire.

Etape	Port source initial → Nouveau port destinataire	Port destinataire initial → Nouveau port source
9	Le moteur de politique arrête le ***PSSourceOnTimer***. La couche protocole envoie le message ***GoodCRC*** à la nouvelle source. Le moteur de politique évalue le message ***PS_RDY*** provenant de la nouvelle source et demande au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de consommer le courant en tant que nouveau destinataire.	La couche protocole reçoit le message ***GoodCRC*** provenant du nouveau destinataire.
10	En tant que nouveau destinataire, l'alimentation passe du mode veille de permutation à la consommation de ***pSnkSusp*** dans un délai ***tNewSnk*** (t4). L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle fonctionne en tant que nouveau destinataire. A ce stade, les négociations qui s'ensuivent entre la nouvelle source et le nouveau destinataire ***Pouvoir/Peut/Peuvent*** se dérouler normalement. Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau. La durée (t4) dépend de l'amplitude de la variation de charge.	

7.3.10 Permutation des rôles d'alimentation demandée par la source

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectée lors d'une permutation des rôles d'alimentation demandée par la source est représentée à la Figure 7-29. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-10. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22. Noter que sur cette figure, le destinataire a déjà envoyé un message ***PR_Swap*** à la source.

Figure 7-29 Diagramme de transition pour une permutation des rôles d'alimentation demandée par la source



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink Port Power Supply	Alimentation du port destinataire
Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Send Go To Min	Envoyer Go To Min
Evaluate Go To Min	Evaluer Go To Min
Source	Source
Sink	Destinataire
Sink previously negotiated of to min current	Le destinataire a préalablement négocié le courant Go To Min
V_{BUS} doesn't change	V_{BUS} ne varie pas

Go To Min Current	Courant Go To Min
Port to Port Messaging	Messagerie port à port
Source Port Interaction	Interaction du port source
Sink Port Interaction	Interaction du port destinataire
Source V_{BUS} Voltage	Tension V_{BUS} source
Sink V_{BUS} Current	Courant V_{BUS} destinataire

Tableau 7-10 Description de séquence pour une permutation des rôles d'alimentation demandée par la source

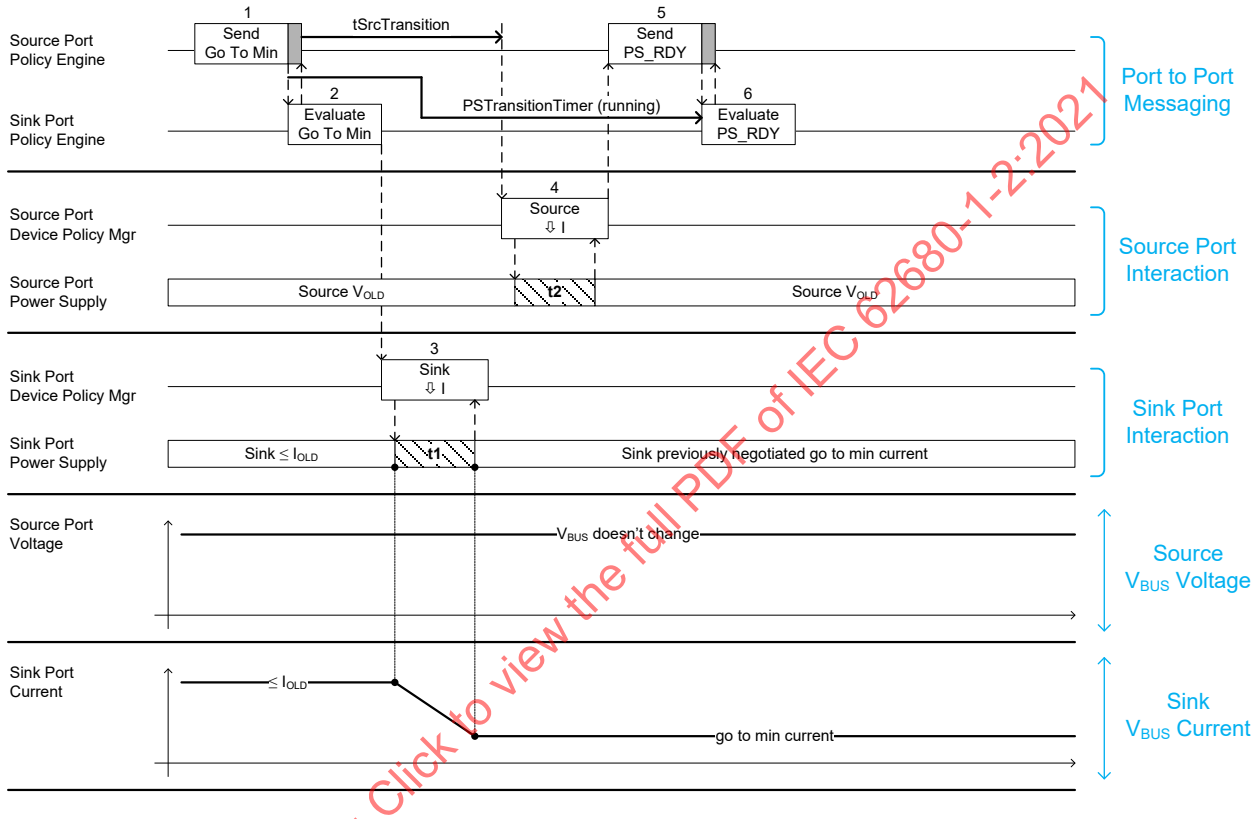
Etape	Port source initial → Nouveau port destinataire	Port destinataire initial → Nouveau port source
1	Le moteur de politique reçoit le message ***Accept***.	Le moteur de politique envoie le message ***Accept*** à la source initiale.
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire. Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de modifier sa puissance de sortie.	La couche protocole reçoit le message ***GoodCRC*** provenant de la source initiale. Le moteur de politique lance le ***PSSourceOffTimer***.
2a		Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de passer en veille de permutation. L'alimentation ***Devoir/Doit/Doivent*** achever le passage en veille de permutation dans un délai ***tSnkStdby*** (t1); t1 ***Devoir/Doit/Doivent*** s'achever avant ***tSrcTransition***. Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau. Le moteur de politique lance le ***PSSourceOffTimer***. Si le destinataire est en veille, le destinataire initial ***Ne doit/doivent pas*** consommer plus de ***iSnkSwapStdby*** (I1).
3	***tSrcTransition*** après la réception du message ***GoodCRC***, l'alimentation commence à passer sa capacité de puissance de sortie en veille de permutation (voir 7.1.10). L'alimentation ***Devoir/Doit/Doivent*** passer en veille de permutation dans un délai ***tSrcSwapStdby*** (t2). L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle est prête à fonctionner en tant que nouveau destinataire. La terminaison CC passe de Rp à Rd (voir ***[USB Type-C 2.0]***). Le statut de l'alimentation est transmis au moteur de politique.	
4	Le moteur de politique envoie le message ***PS_RDY*** au dispositif qui va bientôt devenir la nouvelle source.	Le moteur de politique reçoit le message ***PS_RDY*** et arrête le ***PSSourceOffTimer***.
5	La couche protocole reçoit le message ***GoodCRC*** provenant du dispositif qui va bientôt devenir la nouvelle source. Le moteur de politique lance le ***PSSourceOnTimer***. A ce stade, la source initiale est prête à devenir le nouveau destinataire.	La couche protocole envoie le message ***GoodCRC*** au nouveau destinataire. Dès que le message ***PS_RDY*** est évalué, le destinataire initial est prêt à fonctionner en tant que nouvelle source. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de fonctionner en tant que nouvelle source.
6		La terminaison CC passe de Rd à Rp (voir ***[USB Type-C 2.0]***). En tant que nouvelle source, l'alimentation passe du mode veille de permutation à l'alimentation de ***vSafe5V*** par défaut dans un délai ***tNewSrc*** (t3). L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle fonctionne en tant que nouvelle source.
7	Le moteur de politique reçoit le message ***PS_RDY*** et arrête le ***PSSourceOnTimer***.	Le gestionnaire de politique d'utilisation des dispositifs informe le moteur de politique que l'alimentation est prête, et le moteur de politique envoie le message ***PS_RDY*** au nouveau destinataire.

Etape	Port source initial → Nouveau port destinataire	Port destinataire initial → Nouveau port source
8	La couche protocole envoie le message ***GoodCRC*** à la nouvelle source. Le moteur de politique évalue le message ***PS_RDY*** provenant de la nouvelle source et demande au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de consommer le courant en tant que nouveau destinataire.	La couche protocole reçoit le message ***GoodCRC*** provenant du nouveau destinataire.
9	En tant que nouveau destinataire, l'alimentation passe du mode veille de permutation à la consommation de ***pSnkSusp*** dans un délai ***tNewSnk*** (t4). L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle fonctionne en tant que nouveau destinataire. A ce stade, les négociations qui s'ensuivent entre la nouvelle source et le nouveau destinataire ***Pouvoir/Peut/Peuvent*** se dérouler normalement. Le nouveau destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau. La durée (t4) dépend de l'amplitude de la variation de charge.	

7.3.11 Diminution de courant GotoMin

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectée lors de la diminution de courant GotoMin est représentée à la Figure 7-30. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-11. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22 et le Tableau 7-11.

Figure 7-30 Diagramme de transition pour la diminution de courant GotoMin



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink Port Power Supply	Alimentation du port destinataire
Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Send Hard Reset	Envoyer Hard Reset
Process Hard Reset	Traiter Hard Reset
Source Hard Reset	Réinitialisation matérielle de la source
Source Recover	Reprise de la source
Source	Source

Sink Prepare	Préparation du destinataire
Sink	Destinataire
Ready to recover and power up	Prêt à la reprise et au démarrage
Default current draw	Appel de courant par défaut
Port to Port Messaging	Messagerie port à port
Source Port Interaction	Interaction du port source
Sink Port Interaction	Interaction du port destinataire
Source V_{BUS} Voltage	Tension V_{BUS} source
Sink VBUS Current	Courant VBUS source

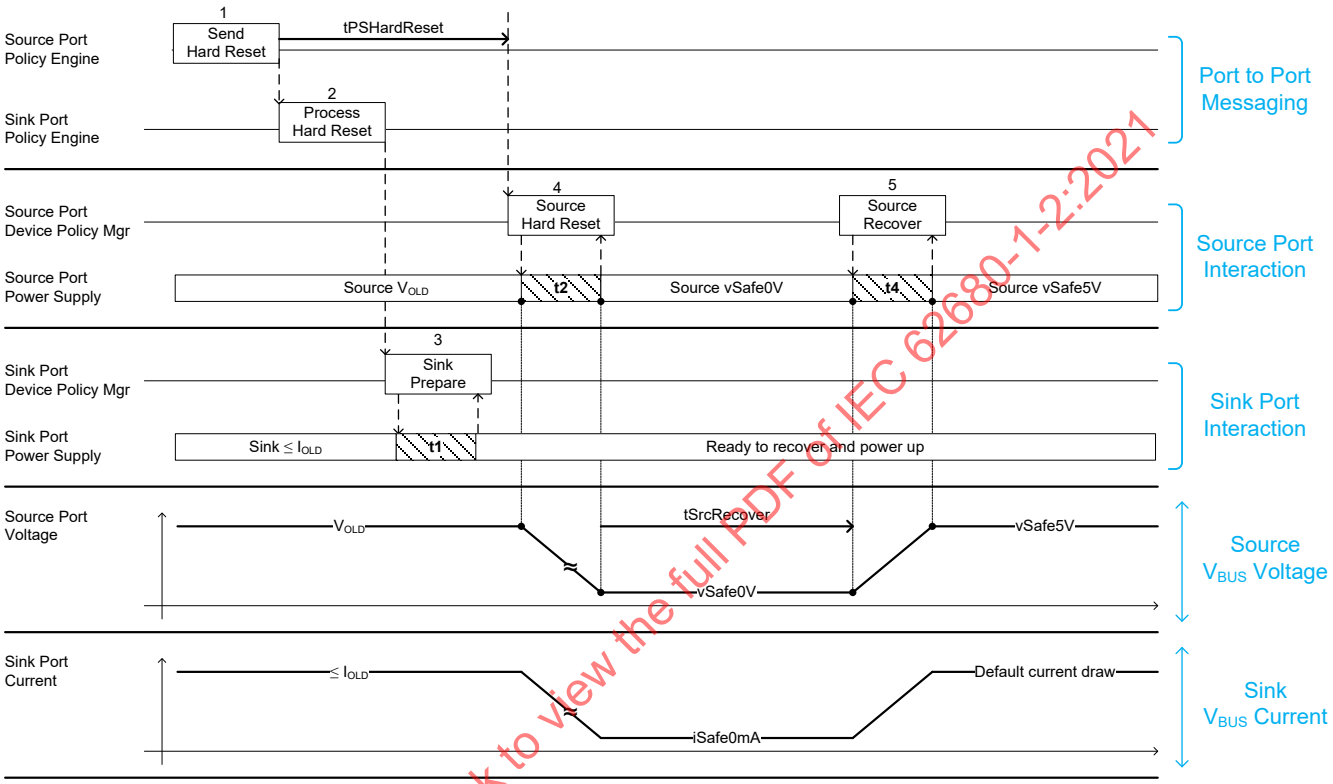
Tableau 7-11 Description de séquence de diminution de courant GotoMin

Etape	Port source	Port destinataire
1	Le moteur de politique envoie le message ***GotoMin*** au destinataire.	Le moteur de politique reçoit le message ***GotoMin*** et lance le ***PSTransitionTimer.***
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire. Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de modifier sa puissance de sortie.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***GotoMin***.
3		Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de réduire la consommation de puissance, dans un délai ***tSnkNewPower*** (t1), au niveau de puissance réduite préalablement négociée; t1 ***Devoir/Doit/Doivent*** s'achever avant ***tSrcTransition***. Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau.
4	***tSrcTransition*** après la réception du message ***GoodCRC***, l'alimentation commence à modifier la capacité de sa puissance de sortie. L'alimentation ***Devoir/Doit/Doivent*** être prête à fonctionner au nouveau niveau de puissance dans un délai ***tSrcReady*** (t2). L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle est prête à fonctionner au nouveau niveau de puissance. Le statut de l'alimentation est transmis au moteur de politique.	
5	Le moteur de politique envoie le message ***PS_RDY*** au destinataire.	Le moteur de politique reçoit le message ***PS_RDY***.
6	La couche protocole reçoit le message ***GoodCRC*** du destinataire.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue le message ***PS_RDY*** de la source et aucune action supplémentaire n'est exigée.

7.3.12 Réinitialisation matérielle déclenchée par la source

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectée lors d'une réinitialisation matérielle déclenchée par la source est représentée à la Figure 7-31. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-12. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être appliqués sont répertoriés dans le Tableau 7-22 et le Tableau 7-23.

Figure 7-31 Diagramme de transition pour une réinitialisation matérielle déclenchée par la source



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink Port Power Supply	Alimentation du port destinataire
Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Send Hard Reset	Envoyer Hard Reset
Evaluate Hard Reset	Evaluer Hard Reset
Process Hard Reset	Traiter Hard Reset
Source Hard Reset	Réinitialisation matérielle de la source
Source Recover	Reprise de la source
Source	Source

Sink Prepare	Préparation du destinataire
Sink	Destinataire
Ready to recover and power up	Prêt à la reprise et au démarrage
Default current draw	Appel de courant par défaut
Port to Port Messaging	Messagerie port à port
Source Port Interaction	Interaction du port source
Sink Port Interaction	Interaction du port destinataire
Source V_{BUS} Voltage	Tension V_{BUS} source
Sink VBUS Current	Courant VBUS source

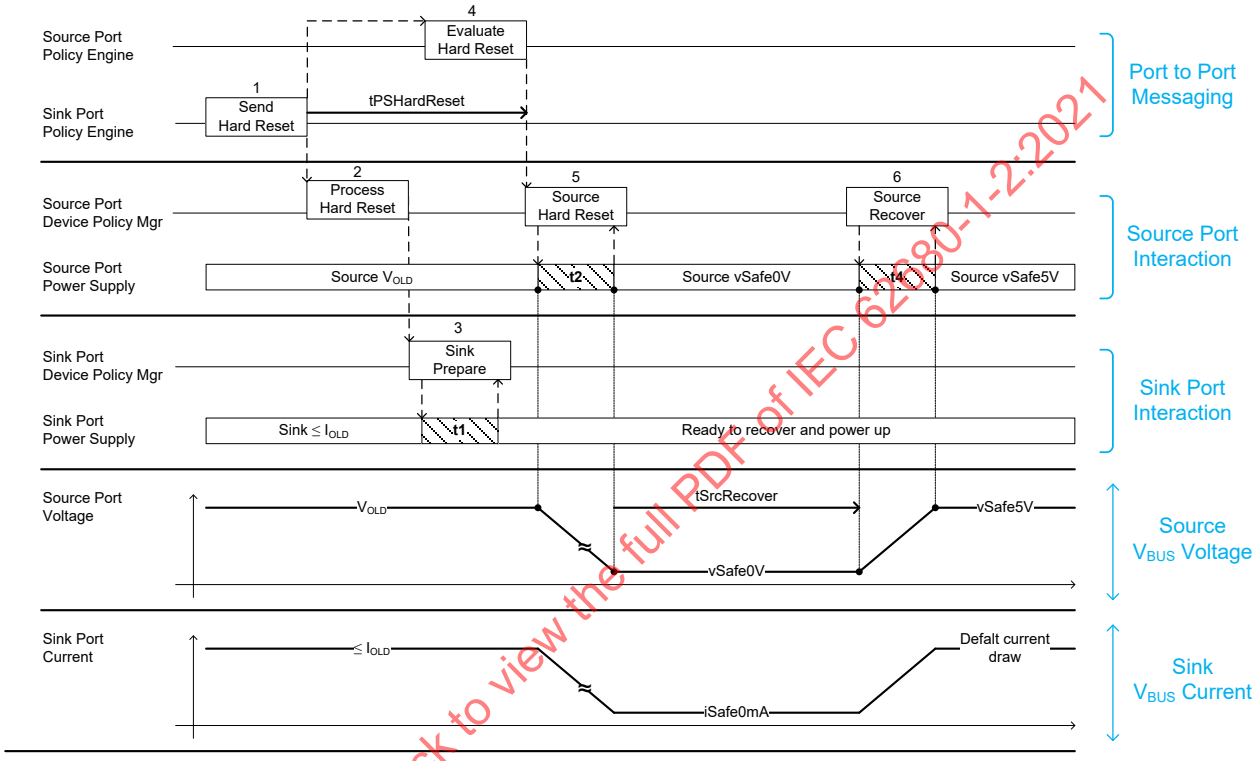
Tableau 7-12 Description de séquence pour une réinitialisation matérielle déclenchée par la source

Etape	Port source	Port destinataire
1	Le moteur de politique envoie un signal ***Hard Reset*** au destinataire.	Le destinataire reçoit le signal ***Hard Reset***.
2		Le moteur de politique est informé de la réinitialisation matérielle. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de se préparer pour une réinitialisation matérielle.
3		Le destinataire se prépare pour la réinitialisation matérielle dans un délai ***tSnkHardResetPrepare*** (t1) et transmet une indication au gestionnaire de politique d'utilisation des dispositifs. Le destinataire ***Ne doit/doivent pas*** consommer plus de ***iSafe0mA*** lorsque V_{BUS} est entraînée vers ***vSafe0V***.
4	Le moteur de politique attend ***tPSHardReset*** après l'envoi du signal ***Hard Reset***, puis demande au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de procéder à une réinitialisation matérielle. La transition vers ***vSafe0V*** ***Devoir/Doit/Doivent*** se produire dans un délai ***tSafe0V*** (t2).	
5	Après ***tSrcRecover***, la source applique la puissance à V_{BUS} dans le cadre d'une tentative de rétablissement de la communication avec le destinataire et reprend le fonctionnement USB par défaut. La transition vers ***vSafe5V*** ***Devoir/Doit/Doivent*** se produire dans un délai ***tSrcTurnOn*** (t4).	Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau.

7.3.13 Réinitialisation matérielle déclenchée par le destinataire

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectée lors d'une réinitialisation matérielle déclenchée par le destinataire est représentée à la Figure 7-32. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-13. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22 et le Tableau 7-23.

Figure 7-32 Diagramme de transition pour une réinitialisation matérielle déclenchée par le destinataire



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink Port Power Supply	Alimentation du port destinataire
Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Send Accept	Envoyer Accept
Evaluate Accept	Evaluer Accept
Source	Source
(running)	(exécution)
Sink	Destinataire

V_{BUS} doesn't change	V_{BUS} ne varie pas
Current doesn't change	Le courant ne varie pas
Port to Port Messaging	Messagerie port à port
Source Port Interaction	Interaction du port source
Sink Port Interaction	Interaction du port destinataire
Source V_{BUS} Voltage	Tension V_{BUS} source
Sink V_{BUS} Current	Courant V_{BUS} destinataire

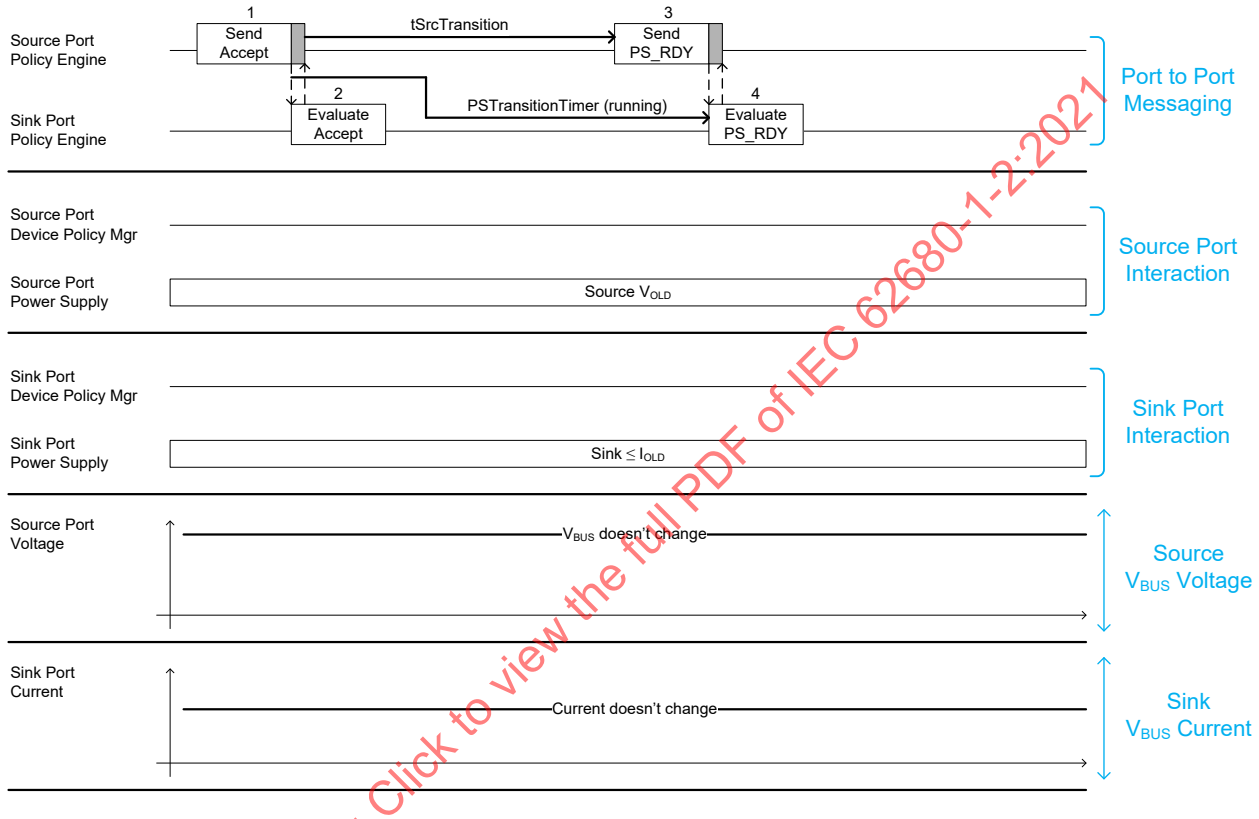
Tableau 7-13 Description de séquence pour une réinitialisation matérielle déclenchée par le destinataire

Etape	Port source	Port destinataire
1		Le moteur de politique envoie un signal ***Hard Reset*** à la source.
2		Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de se préparer pour une réinitialisation matérielle.
3		Le destinataire se prépare pour la réinitialisation matérielle dans un délai ***tSnkHardResetPrepare*** (t1) et transmet une indication au gestionnaire de politique d'utilisation des dispositifs. Le destinataire ***Ne doit/doivent pas*** consommer plus de ***iSafe0mA*** lorsque V_{BUS} est entraînée vers ***vSafe0V***.
4	Le moteur de politique est informé de la réinitialisation matérielle.	
5	Le moteur de politique attend ***tPSHardReset*** après la réception du signal ***Hard Reset***, puis demande au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de procéder à une réinitialisation matérielle. La transition vers ***vSafe0V*** ***Devoir/Doit/Doivent*** se produire dans un délai ***tSafe0V*** (t2).	
6	Après ***tSrcRecover***, la source applique la puissance à V_{BUS} dans le cadre d'une tentative de rétablissement de la communication avec le destinataire et reprend le fonctionnement USB par défaut. La transition vers ***vSafe5V*** ***Devoir/Doit/Doivent*** se produire dans un délai ***tSrcTurnOn*** (t4).	Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau.

7.3.14 Pas de variation de courant ou de tension

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectée lorsque le destinataire demande la même tension et le même courant, étant donné qu'il fonctionne actuellement comme indiqué à la Figure 7-33. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-14. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22 et le Tableau 7-23.

Figure 7-33 Diagramme de transition en cas d'absence de variation de courant ou de tension



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink Port Power Supply	Alimentation du port destinataire
Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Signal Fast Swap	Signal Fast Swap
Detect Fast Swap	Détecter Fast Swap
Evaluate	Evaluer
(running)	(exécution)
Accept	Accept

Send	Envoyer
Evaluate Accept	Evaluer Accept
Source Stops	La source s'arrête
Rp changed to Rd	Rp changée en Rd
Sink	Destinataire
Source	Source
Rd changed to Rp	Rd changée en Rp
Ready & Able to Source vSafe5V	Prêt et capable d'alimenter vSafe5V
Old Source	Ancienne source
Discharging to vSafe0V	Déchargement à vSafe0V
Source Port Policy Engine	Nouvelle source
Sink Port Policy Engine	Ancien destinataire
Source Port Device Policy Mgr	Représente l'augmentation du courant alors que la tension V_{BUS} se décharge
Source Port Power Supply	Nouveau destinataire
Sink Port Device Policy Mgr	Messagerie port à port
Sink Port Power Supply	Interaction du port source
Source Port Voltage	Interaction du port destinataire
Source Port Current	Tension V_{BUS} source
Signal Fast Swap	Courant V_{BUS} destinataire

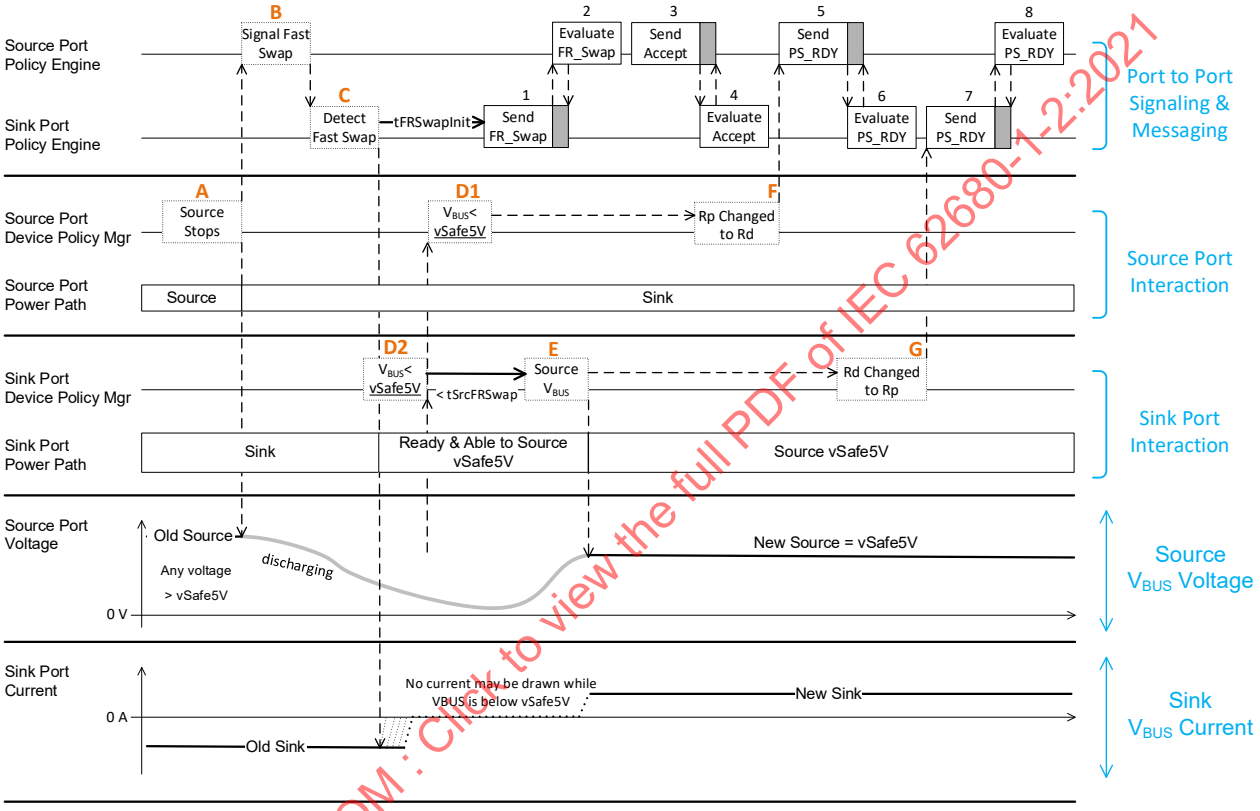
Tableau 7-14 Description de séquence en cas d'absence de variation de courant ou de tension

Etape	Port source	Port destinataire
1	Le moteur de politique envoie le message ***Accept*** au destinataire.	Le moteur de politique reçoit le message ***Accept*** et lance le ***PSTransitionTimer***.
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***Accept***.
3	Le moteur de politique attend ***tSrcTransition***, puis envoie le message ***PS_RDY*** au destinataire.	Le moteur de politique reçoit le message ***PS_RDY***.
4	Le moteur de politique reçoit le message ***GoodCRC*** du destinataire. Note: La décision qu'aucune transition de puissance n'est exigée peut être prise par le gestionnaire de politique d'utilisation des dispositifs ou l'alimentation en fonction de la mise en œuvre.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue le message ***PS_RDY***.

7.3.15 Permutation rapide des rôles

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectée lors d'une permutation rapide des rôles est représentée à la Figure 7-34. Les séquences parallèles qui ***Devoir/Doit/Doivent*** être suivies sont décrites dans le Tableau 7-15. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22 et le Tableau 7-23. Les négociations entre la nouvelle source et le nouveau destinataire ***Pouvoir/Peut/Peuvent*** se produire après envoi du message ***PS_RDY*** final par la nouvelle source. Note: A la Figure 7-34 et au Tableau 7-15, les nombres sont utilisés pour indiquer que les étapes et lettres associées au message sont utilisées pour indiquer d'autres événements.

Figure 7-34 Diagramme de transition pour la permutation rapide des rôles



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink Port Power Supply	Alimentation du port destinataire
Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Evaluate	Evaluer
(running)	(exécution)
Send	Envoyer

Send Accept	Envoyer Accept
Evaluate Accept	Evaluer Accept
Source	Source
Pps Transition Interval	Intervalle de transition Pps
Sink draws current continuously (not to exceed negotiated current)	Le destinataire appelle du courant en permanence (sans dépasser le courant négocié)
Port to Port Messaging	Messagerie port à port
Source Port Interaction	Interaction du port source
Sink Port Interaction	Interaction du port destinataire
Source V_{BUS} Voltage	Tension V_{BUS} source
Sink V_{BUS} Current	Courant V_{BUS} destinataire

Tableau 7-15 Description de séquence pour la permutation rapides de rôles

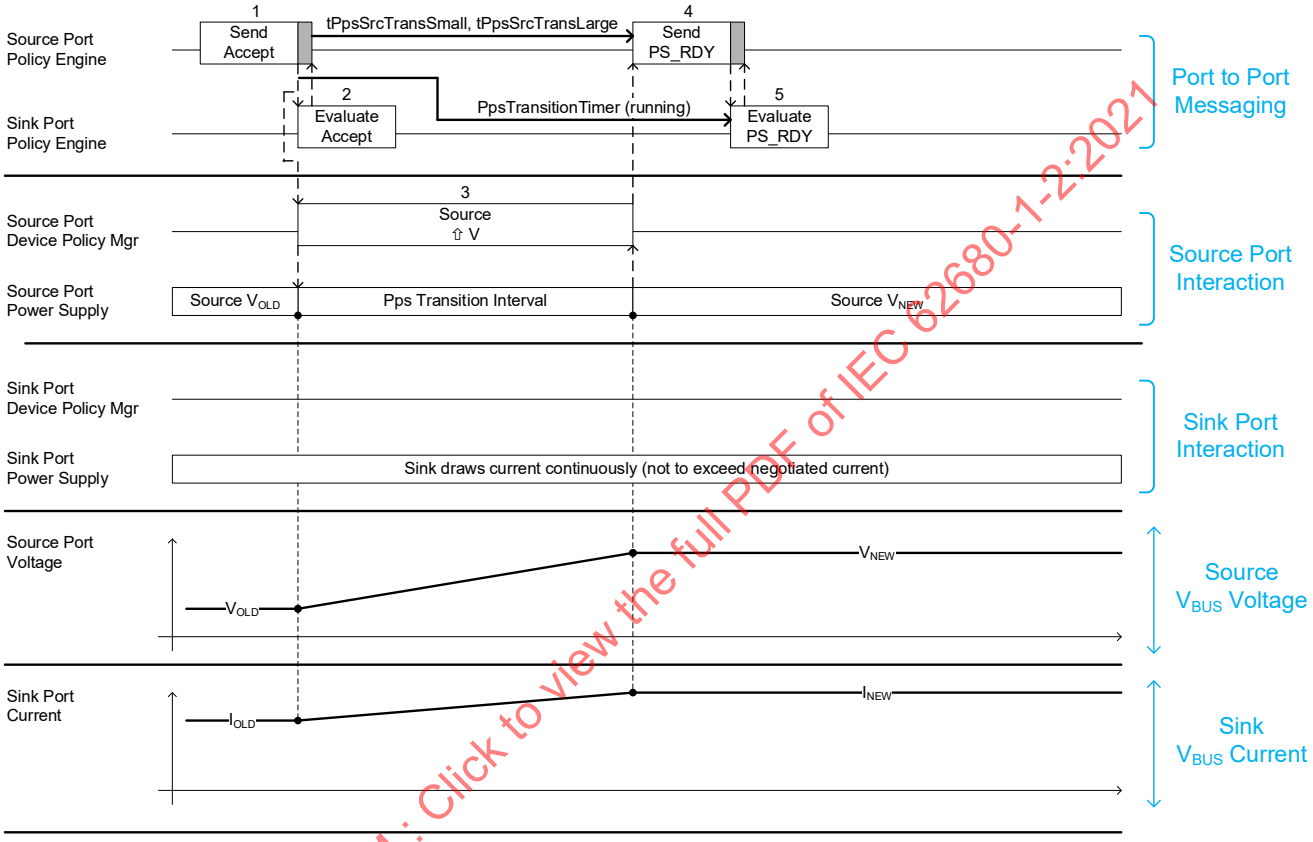
Etape	Port source initial → Nouveau port destinataire	Port destinataire initial → Nouveau port source
Signal de permutation rapide des rôles et transition de puissance		
A	La source connectée au Hub UFP (voir Figure 7-14) arrête d'alimenter V_{BUS}.	
B	Le moteur de politique signale la permutation rapide des rôles au destinataire initial sur le fil CC. Lorsque V_{BUS} < ***vSafe5V*** (min), cela indique au gestionnaire de politique d'utilisation des dispositifs de ne pas consommer plus que ***pSnkStdby***, jusqu'à ce que ***TSnkFRSwap*** se soit écoulé.	
C		Le moteur de politique détecte le signal de permutation rapide des rôles sur le fil CC provenant de la source initiale et ***Devoir/Doit/Doivent*** renvoyer le message ***FR_Swap*** à cette dernière (qui ne fournit plus V_{BUS}) dans un délai ***tFRSwapInit***.
D1	Le moteur de politique surveille V_{BUS} ≤ ***vSafe5V***, de manière à pouvoir envoyer un message ***PS_RDY*** à la nouvelle source à l'Etape 5 de la séquence de messages.	
D2		Le moteur de politique surveille V_{BUS} ≤ ***vSafe5V***, de sorte que le destinataire initial puisse endosser le rôle de la nouvelle source et commencer à fournir V_{BUS}.
E		Lorsque V_{BUS} = ***vSafe5V***, la nouvelle source ***Pouvoir/Peut/Peuvent*** alimenter V_{BUS}. Lorsque V_{BUS} < ***vSafe5V***, la nouvelle source ***Devoir/Doit/Doivent*** alimenter V_{BUS} dans un délai ***tSrcFRSwap***, et le message ***PS_RDY*** peut être envoyé au nouveau destinataire à l'Etape 7 de la séquence de messages.
F	La terminaison CC passe de Rp à Rd (voir ***[USB Type-C 2.0]***) avant que le nouveau destinataire n'envoie le message ***PS_RDY*** de l'Etape 5 à la nouvelle source.	
G		La terminaison CC passe de Rd à Rp (voir ***[USB Type-C 2.0]***) avant que la nouvelle source n'envoie le message ***PS_RDY*** de l'Etape 7 au nouveau destinataire.
Séquence de messages de permutation rapide des rôles		
1	Le moteur de politique reçoit le message ***FR_Swap*** du destinataire initial qui va devenir la nouvelle source.	Le moteur de politique envoie le message ***FR_Swap*** à la source initiale (qui ne fournit plus V_{BUS}) après la détection du signal de permutation rapide des rôles de l'Etape C.

Etape	Port source initial → Nouveau port destinataire	Port destinataire initial → Nouveau port source
2	La couche protocole envoie le message ***GoodCRC*** au destinataire initial. Le moteur de politique évalue ensuite le message ***FR_Swap***.	La couche protocole reçoit le message ***GoodCRC*** de la source initiale.
3	Le moteur de politique envoie un message ***Accept*** au destinataire initial qui va devenir la nouvelle source.	Le moteur de politique reçoit le message ***Accept*** de la source initiale qui va devenir le nouveau destinataire.
4	La couche protocole reçoit le message ***GoodCRC*** provenant du destinataire initial qui va devenir la nouvelle source.	La couche protocole envoie le message ***GoodCRC*** à la source initiale qui va devenir le nouveau destinataire.
5	Le moteur de politique envoie un message ***PS_RDY*** au destinataire initial qui va devenir la nouvelle source. Le moteur de politique ***Devoir/Doit/Doivent*** attendre l'étape D1 avant d'envoyer le message ***PS_RDY***, et il ***Devoir/Doit/Doivent*** envoyer le message ***PS_RDY*** dans un délai ***tFRSwap5V*** après l'envoi du message ***Accept***.	Le moteur de politique reçoit le message ***PS_RDY*** du nouveau destinataire.
6	La couche protocole reçoit le message ***GoodCRC*** de la nouvelle source.	La couche protocole envoie le message ***GoodCRC*** du destinataire initial qui est désormais la nouvelle source. Le moteur de politique évalue ensuite le message ***PS_RDY***.
7	Le moteur de politique reçoit le message ***PS_RDY*** de la nouvelle source.	Le moteur de politique envoie un message ***PS_RDY*** au nouveau destinataire. Le moteur de politique ***Devoir/Doit/Doivent*** attendre l'étape E avant d'envoyer le message ***PS_RDY***, et ***Devoir/Doit/Doivent*** envoyer le message ***PS_RDY*** dans un délai ***tFRSwapComplete*** après réception du message ***PS_RDY*** en provenance du port source initial.

7.3.16 Augmentation de la tension d'alimentation électrique programmable

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectée lors de l'augmentation de la tension est représentée à la Figure 7-35. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-16. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22 et le Tableau 7-23. Noter que sur cette figure, le destinataire a déjà envoyé un message ***Request*** à la source.

Figure 7-35 Diagramme de transition pour l'augmentation de la tension d'alimentation électrique programmable



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink Port Power Supply	Alimentation du port destinataire
Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Evaluate	Evaluer
(running)	(exécution)
Send	Envoyer
Send Accept	Envoyer Accept

Evaluate Accept	Evaluer Accept
Source	Source
Pps Transition Interval	Intervalle de transition Pps
Sink draws current continuously (not to exceed negotiated current)	Le destinataire appelle du courant en permanence (sans dépasser le courant négocié)
Port to Port Messaging	Messagerie port à port
Source Port Interaction	Interaction du port source
Sink Port Interaction	Interaction du port destinataire
Source V_{BUS} Voltage	Tension V_{BUS} source
Sink V_{BUS} Current	Courant V_{BUS} destinataire

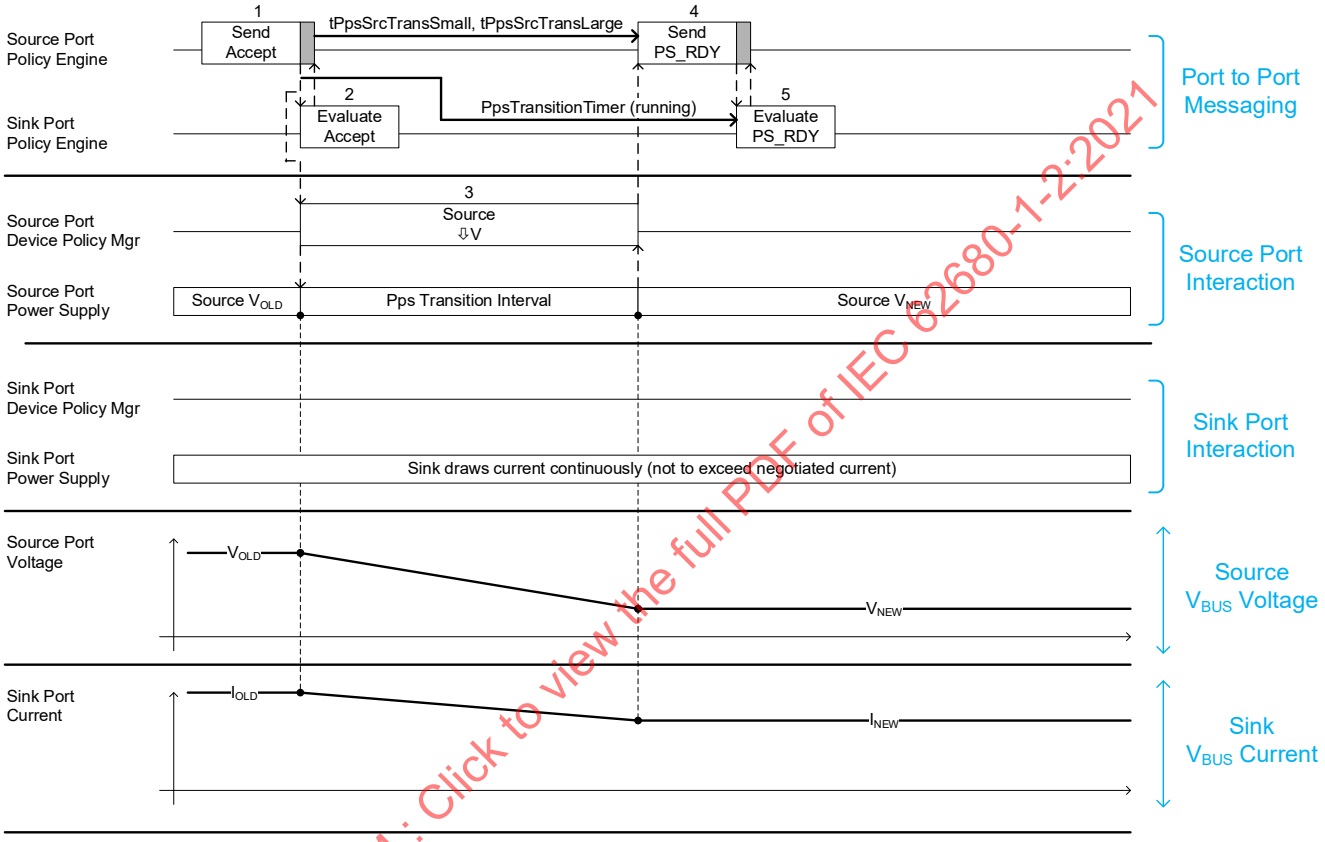
Tableau 7-16 Description de séquence d'augmentation de la tension d'alimentation électrique programmable

Etape	Port source	Port destinataire
1	Le moteur de politique envoie le message ***Accept*** au destinataire.	Le moteur de politique reçoit le message ***Accept*** et lance le ***PSTransitionTimer***.
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation d'augmenter sa tension de sortie.	La couche protocole envoie le message ***GoodCRC*** à la source. Moteur de politique. Le moteur de politique évalue ensuite le message ***Accept***.
3	Après l'envoi du message ***Accept***, l'alimentation électrique programmable commence à augmenter sa tension de sortie. La nouvelle tension de consigne de l'alimentation électrique programmable ***Devoir/Doit/Doivent*** être atteinte dans un délai ***TPpsSrcTransLarge*** pour des paliers supérieurs à ***VPpsSmallStep*** ou dans un délai ***TPpsSrcTransSmall***. L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle a atteint le nouveau point de consigne et si V_{BUS} est au nouveau niveau correspondant, ou si l'alimentation fonctionne en mode CL. Le statut de l'alimentation est transmis au moteur de politique.	
4	Le moteur de politique envoie le message ***PS_RDY*** au destinataire.	Le moteur de politique reçoit le message ***PS_RDY*** de la source.
5	La couche protocole reçoit le message ***GoodCRC*** du destinataire.	La couche protocole envoie le message ***GoodCRC*** à la source. Ensuite, le moteur de politique évalue le message ***PS_RDY*** provenant de la source et signale au gestionnaire de politique d'utilisation des dispositifs que l'alimentation programmable fonctionne à la nouvelle tension établie.

7.3.17 Diminution de la tension d'alimentation électrique programmable

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectée lors de la diminution de la tension est représentée à la Figure 7-36. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-17. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22 et le Tableau 7-23. Noter que sur cette figure, le destinataire a déjà envoyé un message ***Request*** à la source.

Figure 7-36 Diagramme de transition pour la diminution de la tension d'alimentation électrique programmable



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink Port Power Supply	Alimentation du port destinataire
Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Evaluate	Evaluer
(running)	(exécution)
Send	Envoyer
Send Accept	Envoyer Accept

Evaluate Accept	Evaluer Accept
Source	Source
Pps Transition Interval	Intervalle de transition Pps
Sink draws current continously (not to exceed negotiated curent)	Le destinataire appelle du courant en permanence (sans dépasser le courant négocié)
Port to Port Messaging	Messagerie port à port
Source Port Interaction	Interaction du port source
Sink Port Interaction	Interaction du port destinataire
Source V_{BUS} Voltage	Tension V_{BUS} source
Sink V_{BUS} Current	Courant V_{BUS} destinataire

Tableau 7-17 Description de séquence de diminution de la tension d'alimentation électrique programmable

Etape	Port source	Port destinataire
1	Le moteur de politique envoie le message ***Accept*** au destinataire.	Le moteur de politique reçoit le message ***Accept*** et lance le ***PSTransitionTimer***.
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de diminuer sa tension de sortie.	La couche protocole envoie le message ***GoodCRC*** à la source. Moteur de politique. Le moteur de politique évalue ensuite le message ***Accept***.
3	Après l'envoi du message ***Accept***, l'alimentation électrique programmable commence à diminuer sa tension de sortie. La nouvelle tension de consigne de l'alimentation électrique programmable (correspondant à la ***vPpsNew***) ***Devoir/Doit/Doivent*** être atteinte dans un délai ***TPpsSrcTransLarge*** pour des paliers supérieurs à ***VPpsSmallStep*** ou dans un délai ***TPpsSrcTransSmall***. L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle a atteint le nouveau niveau. Le statut de l'alimentation est transmis au moteur de politique.	
4	Le moteur de politique envoie le message ***PS_RDY*** au destinataire.	Le moteur de politique reçoit le message ***PS_RDY*** de la source.
5	La couche protocole reçoit le message ***GoodCRC*** du destinataire.	La couche protocole envoie le message ***GoodCRC*** à la source. Ensuite, le moteur de politique évalue le message ***PS_RDY*** provenant de la source et signale au gestionnaire de politique d'utilisation des dispositifs que l'alimentation programmable fonctionne à la nouvelle tension établie (qui correspond à ***vPpsNew***).

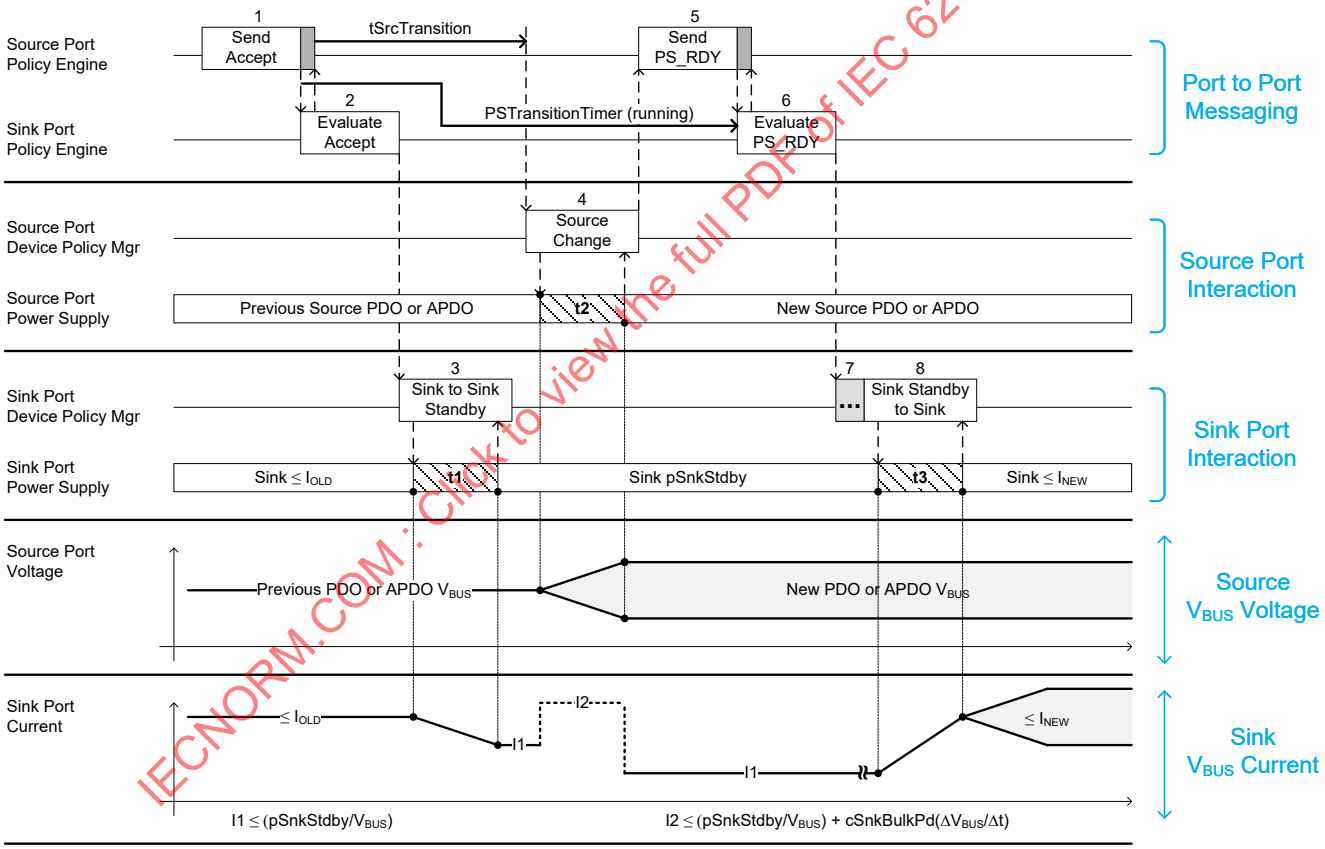
7.3.18 Modification du PDO ou de l'APDO source

L'interaction entre le gestionnaire de politique d'utilisation des dispositifs, le moteur de politique du port et l'alimentation lors du changement *entre* les PDO et APDO source (voir ci-dessous) est représentée à la Figure 7-37.

- PDO vers PDO
- PDO vers APDO
- APDO vers APDO
- APDO vers PDO

Lorsque la transition commence, la tension source ***Devoir/Doit/Doivent*** être dans les limites de la plage V_{BUS} ***Valide(s)*** du PDO ou de l'APDO source précédent. Lorsque la transition est terminée, la tension source ***Devoir/Doit/Doivent*** être dans les limites de la plage V_{BUS} ***Valide(s)*** du nouveau PDO ou nouvel APDO source. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-18. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22 et le Tableau 7-23. Noter que sur cette figure, le destinataire a déjà envoyé un message ***Request*** à la source.

Figure 7-37 Diagramme de transition pour la modification du PDO ou de l'APDO source



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire

Sink Port Power Supply	Alimentation du port destinataire
Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Evaluate	Evaluer
(running)	(exécution)
Send	Envoyer
Send Accept	Envoyer Accept
Evaluate Accept	Evaluer Accept
Source Change	Changement de source
Previous Source PDO or APDO	PDO ou APDO de l'ancienne source
New Source PDO or APDO	PDO ou APDO de la nouvelle source
Sink to Sink Standby	Destinataire à destinataire en attente
Sink	Destinataire
Sink Standby to Sink	Destinataire en attente à destinataire
Previous PDO or APDO V_{BUS}	V_{BUS} de l'ancien PDO ou APDO
New PDO or APDO V_{BUS}	V_{BUS} du nouveau PDO ou APDO
Sink V_{BUS} Current	Courant V_{BUS} destinataire

Tableau 7-18 Description de séquence de modification du PDO ou de l'APDO source

Etape	Port source	Port destinataire
1	Le moteur de politique envoie le message ***Accept*** au destinataire.	Le moteur de politique reçoit le message ***Accept*** et lance le ***PSTransitionTimer***.
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de passer au PDO ou à l'APDO de la nouvelle source.	La couche protocole envoie le message ***GoodCRC*** à la source. Moteur de politique. Le moteur de politique évalue ensuite le message ***Accept***.
3		Le moteur de politique indique au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de réduire sa consommation électrique à ***pSnkStdby*** dans un délai ***tSnkStdby*** (t1); t1 Doit s'achever avant ***tSrcTransition***. Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau.
4	***tSrcTransition*** après l'envoi du message ***GoodCRC***, la source commence à passer au nouveau PDO ou nouvel APDO. La source ***Devoir/Doit/Doivent*** être prête à fonctionner au nouveau niveau de puissance dans un délai ***tSrcReady*** (t2). L'alimentation informe le gestionnaire de politique d'utilisation des dispositifs qu'elle est prête à fonctionner au nouveau niveau de puissance. Le statut de l'alimentation est transmis au moteur de politique.	
5	Le moteur de politique envoie le message ***PS_RDY*** au destinataire.	Le moteur de politique reçoit le message ***PS_RDY*** de la source.
6	La couche protocole reçoit le message ***GoodCRC*** du destinataire.	La couche protocole envoie le message ***GoodCRC*** à la source. Ensuite, le moteur de politique évalue le message ***PS_RDY*** provenant de la source et signale au gestionnaire de politique d'utilisation des dispositifs que la source fonctionne au nouveau PDO ou APDO.

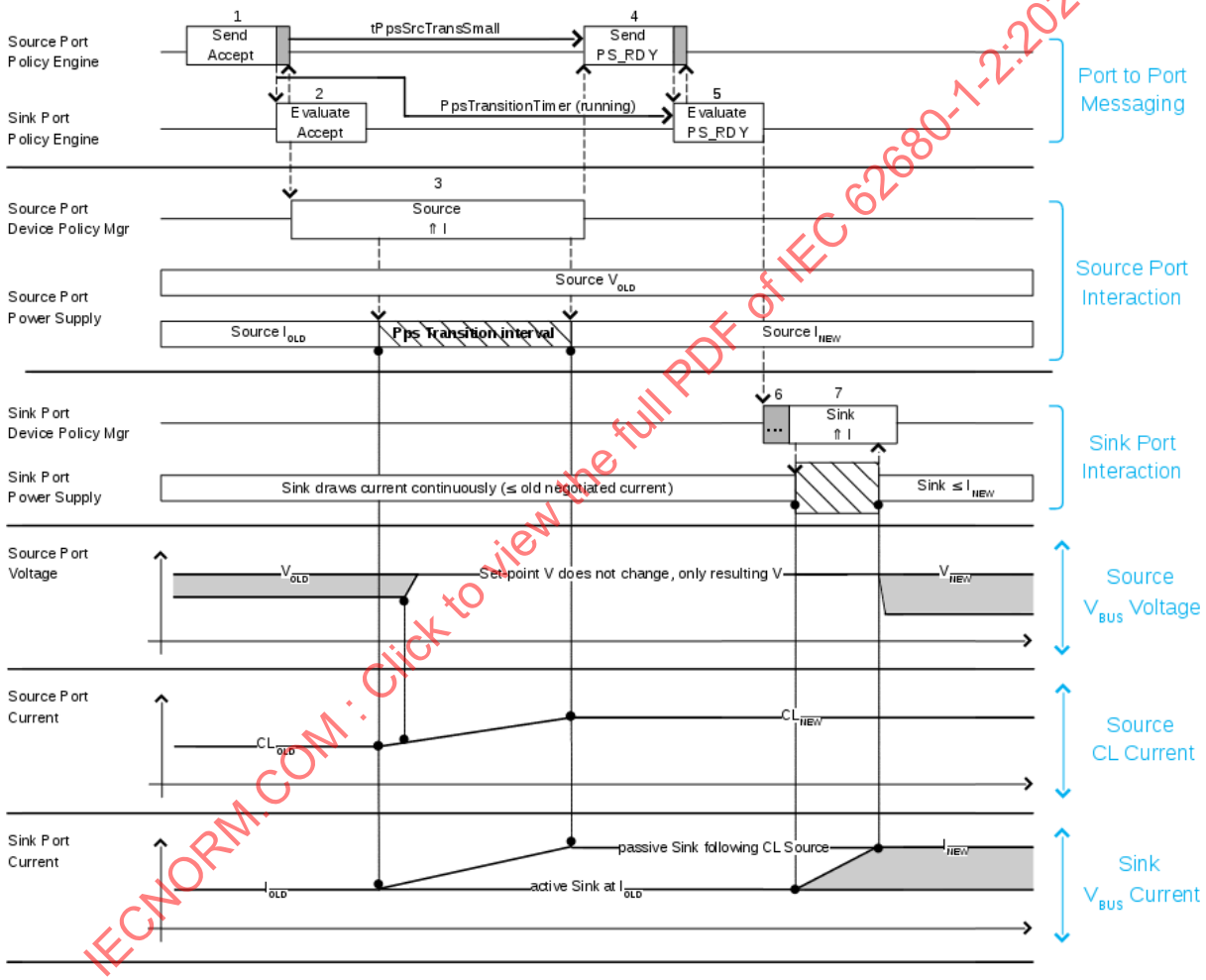
Etape	Port source	Port destinataire
7		Le destinataire ***Pouvoir/Peut/Peuvent*** commencer à fonctionner au nouveau niveau de puissance à tout moment après l'évaluation du message ***PS_RDY***. Cette durée est indéterminée.
8		Le destinataire ***Ne doit/doivent pas*** enfreindre le comportement de charge transitoire défini en 7.2.6 lors de la transition vers le nouveau niveau de puissance et lors du fonctionnement à ce nouveau niveau. La durée (t3) dépend de l'amplitude de la variation de charge.

7.3.19 Augmentation du courant d'alimentation électrique programmable

L'interaction de la politique système, de la politique d'utilisation des dispositifs et de l'alimentation qui ***Devoir/Doit/Doivent*** être respectée lors de l'augmentation de la limite de courant dans le même APDO, sans dépasser les valeurs maximales pour cet APDO et sans modifier la tension demandée, est représentée à la Figure 7-38. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-19. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22 et le Tableau 7-23. Noter que sur cette figure, le destinataire a déjà envoyé un message ***Request*** à la source.

Le destinataire ***Pouvoir/Peut/Peuvent*** appeler un courant égal à la limite de courant augmentée de la source avant de recevoir le message ***PS_RDY*** pour la nouvelle demande.

Figure 7-38 Diagramme de transition pour l'augmentation du courant en mode PPS



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink Port Power Supply	Alimentation du port destinataire

Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Sink Port Current	Courant du port destinataire
Port to Port Messaging	Echange de messages entre ports
Source Port Interaction	Interaction avec le port source
Sink Port Interaction	Interaction avec le port destinataire
Sink Vbus Current	Courant Vbus du destinataire
Source Vbus Voltage	Tension Vbus de la source
Source CLC Current	Courant CLC de la source
Send Accept	Envoyer Accept
Evaluate Accept	Evaluer Accept
(running)	(en cours)
Send	Envoyer
Evaluate	Evaluer
Source	Source
Sink	Destinataire
Set-point V does not change, only resulting V	La tension de consigne est invariable, seule la tension résultante varie
Passive Sink following CL Source	Destinataire passif après la source en CL
Active Sink at Iold	Destinataire actif à Iold

Tableau 7-19 Description de séquence d'augmentation du courant en mode PPS

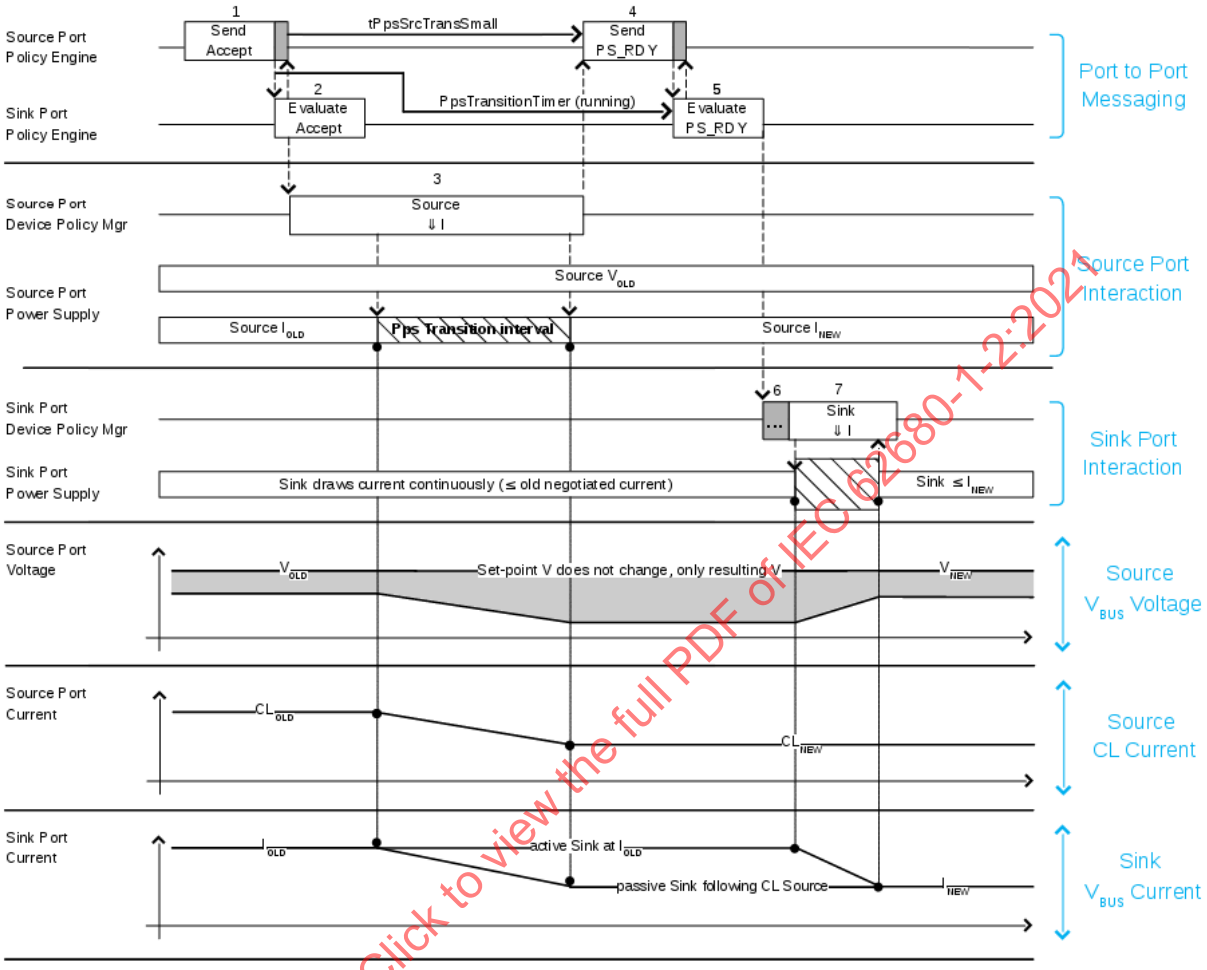
Etape	Port source	Port destinataire
1	Le moteur de politique envoie le message ***Accept*** au destinataire.	Le moteur de politique reçoit le message ***Accept*** et lance le ***PSTransitionTimer***.
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation d'augmenter sa limite de courant établie.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***Accept***.
3	L'alimentation augmente sa limite de courant de consigne jusqu'à la nouvelle valeur demandée.	Le destinataire appelle un courant conforme à la limite de courant augmentée de la source.
4	Le moteur de politique attend ***TPpsSrcTransSmall***, puis envoie le message ***PS_RDY*** au destinataire.	Le moteur de politique reçoit le message ***PS_RDY***.
5	Le moteur de politique reçoit le message ***GoodCRC*** du destinataire.	La couche protocole envoie le message ***GoodCRC*** à la source.
6		Le moteur de la politique évalue le message ***PS_RDY*** et indique au gestionnaire de politique d'utilisation des dispositifs qu'il peut augmenter le courant jusqu'à la valeur demandée sans que la source ne passe en mode CL.
7		Le destinataire augmente son courant.

7.3.20 Diminution du courant d'alimentation électrique programmable

L'interaction de la politique système, de la politique d'utilisation des dispositifs et de l'alimentation qui ***Devoir/Doit/Doivent*** être respectée lors de la diminution de la limite de courant dans le même APDO, sans dépasser les valeurs maximales pour cet APDO et sans modifier la tension demandée, est représentée à la Figure 7-39. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-20. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22

et le Tableau 7-23. Noter que sur cette figure, le destinataire a déjà envoyé un message ***Request*** à la source.

Figure 7-39 Diagramme de transition pour la diminution du courant en mode PPS



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink Port Power Supply	Alimentation du port destinataire
Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Sink Port Current	Courant du port destinataire
Port to Port Messaging	Echange de messages entre ports
Source Port Interaction	Interaction avec le port source
Sink Port Interaction	Interaction avec le port destinataire
Sink Vbus Current	Courant Vbus du destinataire

Source Vbus Voltage	Tension Vbus de la source
Source CLC Current	Courant CLC de la source
Send Accept	Envoyer Accept
Evaluate Accept	Evaluer Accept
(running)	(en cours)
Send	Envoyer
Evaluate	Evaluer
Source	Source
Sink	Destinataire
Sink draws current continuously (≤ old negotiated current)	Le destinataire appelle du courant en continu (≤ à l'ancien courant négocié)
Set-point V does not change, only resulting V	La tension de consigne est invariable, seule la tension résultante varie
Passive Sink following CL Source	Destinataire passif après la source en CL
Active Sink at Iold	Destinataire actif à Iold
Pps Transition interval	Intervalle de transition PPS

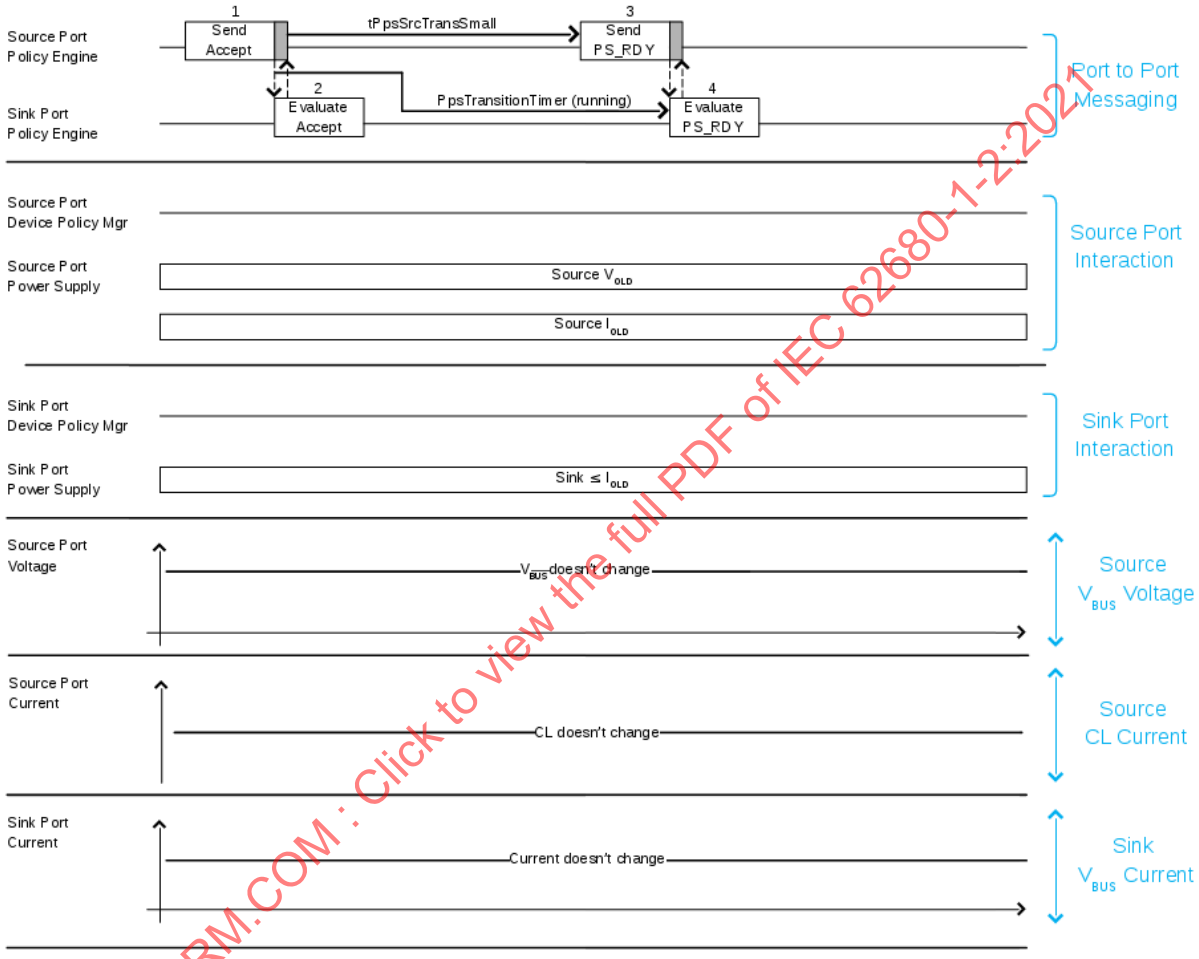
Tableau 7-20 Description de séquence de diminution du courant en mode PPS

Etape	Port source	Port destinataire
1	Le moteur de politique envoie le message ***Accept*** au destinataire.	Le moteur de politique reçoit le message ***Accept*** et lance le PSTransitionTimer.
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'ordonner à l'alimentation de diminuer sa limite de courant établie.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***Accept*** et demande au destinataire de réduire son courant au-dessous du nouveau niveau de courant négocié.
3	L'alimentation diminue sa limite de courant de consigne jusqu'à la nouvelle valeur négociée.	Le destinataire réduit son courant au-dessous du nouveau courant négocié afin d'empêcher la source d'entrer dans la limite de courant.
4	Le moteur de politique attend ***TPpsSrcTransSmall***, puis envoie le message ***PS_RDY*** au destinataire.	
5	Le moteur de politique reçoit le message ***GoodCRC*** du destinataire.	Le moteur de politique reçoit le message ***PS_RDY***.
6		La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue le message ***PS_RDY***.
7		Le destinataire est autorisé à consommer I_{NEW}, mais il doit savoir que la tension sur V_{BUS} peut alors chuter.

7.3.21 Demande identique d'alimentation électrique programmable

L'interaction entre la politique système, la politique d'utilisation des dispositifs et l'alimentation qui ***Devoir/Doit/Doivent*** être respectée lorsque le destinataire demande les mêmes niveaux de tension et de courant que les niveaux actuellement négociés est représentée à la Figure 7-40. La séquence qui ***Devoir/Doit/Doivent*** être suivie est décrite dans le Tableau 7-21. Les paramètres de temporisation qui ***Devoir/Doit/Doivent*** être suivis sont répertoriés dans le Tableau 7-22 et le Tableau 7-23. Noter que sur cette figure, le destinataire a déjà envoyé un message ***Request*** à la source.

Figure 7-40 Diagramme de transition en cas d'absence de variation de courant ou de tension en mode PPS



Anglais	Français
Source Port Policy Engine	Moteur de politique du port source
Sink Port Policy Engine	Moteur de politique du port destinataire
Source Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port source
Source Port Power Supply	Alimentation du port source
Sink Port Device Policy Mgr	Gestionnaire de politique d'utilisation des dispositifs du port destinataire
Sink Port Power Supply	Alimentation du port destinataire
Source Port Voltage	Tension du port source
Source Port Current	Courant du port source
Sink Port Current	Courant du port destinataire

Port to Port Messaging	Echange de messages entre ports
Source Port Interaction	Interaction avec le port source
Sink Port Interaction	Interaction avec le port destinataire
Sink Vbus Current	Courant Vbus du destinataire
Source Vbus Voltage	Tension Vbus de la source
Source CLC Current	Courant CLC de la source
Send Accept	Envoyer Accept
Evaluate Accept	Evaluer Accept
(running)	(en cours)
Send	Envoyer
Evaluate	Evaluer
Source	Source
Sink	Destinataire
Vbus doesn't change	Vbus est invariable
CL doesn't change	CL est invariable
Current doesn't change	Le courant est invariable

Tableau 7-21 Description de séquence d'augmentation du courant en mode PPS

Etape	Port source	Port destinataire
1	Le moteur de politique envoie le message ***Accept*** au destinataire.	Le moteur de politique reçoit le message ***Accept*** et lance le PSTransitionTimer.
2	La couche protocole reçoit le message ***GoodCRC*** du destinataire.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue ensuite le message ***Accept***.
3	Le moteur de politique envoie le message ***PS_RDY*** au destinataire dans un délai ***TPpsSrcTransSmall***.	Le moteur de politique reçoit le message ***PS_RDY***.
4	Le moteur de politique reçoit le message ***GoodCRC*** du destinataire. Note: La décision qu'aucune transition de puissance n'est exigée peut être prise par le gestionnaire de politique d'utilisation des dispositifs ou l'alimentation en fonction de la mise en œuvre.	La couche protocole envoie le message ***GoodCRC*** à la source. Le moteur de politique évalue le message ***PS_RDY***.

7.4 Paramètres électriques

7.4.1 Paramètres électriques de la source

Les paramètres électriques de la source qui ***Devoir/Doit/Doivent*** être respectés sont spécifiés dans le Tableau 7-22.

Tableau 7-22 Paramètres électriques de la source

Paramètre	Description	MIN	TYP	MAX	UNITS	Référence
cSrcBulk[1]	Capacité de masse de la source lorsqu'un port est alimenté à partir d'une alimentation dédiée.	10			µF	Section 7.1.2

Paramètre	Description	MIN	TYP	MAX	UNITS	Référence
cSrcBulkShared[1]	Capacité de masse de la source lorsqu'un port est alimenté à partir d'une alimentation partagée.	120			µF	Section 7.1.2
IPpsCLMin	Définition de la limite de courant minimale.	1			A	Section 7.1.4.4
IPpsCLNew	Précision de la limite de courant					Section 7.1.4.4
	1 A ≤ Courant de fonctionnement ≤ 3 A	-150		150	mA	
	Courant de fonctionnement > 3 A	-5		5	%	
IPpsCLOperating	Variation totale admise du courant de fonctionnement au point b de la Figure 7-7, la résistance de charge variant en mode Current Limit.	-25		100	mA	Figure 7-7
IPpsCLStep	Taille de palier de programmation de la limite de courant PPS.		50		mA	Section 7.1.4.4
iPpsCLTolerance	Ecart admissible du courant de fonctionnement le long de la ligne de charge entre le point b et e, comme le montre la Figure 7-8.	-25		25	mA	Figure 7-8
IPpsCLTransient	Sur-dépassement admis du courant de sortie lors d'une augmentation de charge en mode CL.			Nouvel-le charge + 100	mA	Section 7.1.4.4
	Sous-dépassement admis du courant de sortie lors d'une diminution de charge en mode CL.	Nouvel-le charge - 100				
IPpsCVCLTransient	Limites de courant transitoire CV à CF en prenant pour hypothèse la réduction de la tension de fonctionnement de la Section 7.2.3.1.	*IPpsCL New* - 100		Nouvel-le charge + 500	mA	Section 7.1.4.4
tNewSnk	Temps admis pour qu'une source initiale en veille de permutation passe au fonctionnement en tant que nouveau destinataire.			15	ms	Figure 7-28, Figure 7-29
TPpsCLCVTransient	Durée de stabilisation de la tension transitoire CF à CV.			25	ms	Section 7.1.4.4

Paramètre	Description	MIN	TYP	MAX	UNITS	Référence
TPpsCLProgramSettle	Durée de stabilisation de programmation de la limite de courant PPS.			250	ms	Section 7.1.4.4
TPpsCLSettle	Durée de stabilisation du courant transitoire de charge CF.			250	ms	Section 7.1.4.4
TPpsCVCLTransient	Durée de stabilisation transitoire CF à CV.			250	ms	Section 7.1.8.1
TPpsSrcTransLarge	Moment où le point de consigne de l'alimentation électrique programmable ***Devoir/Doit/Doivent*** effectuer une transition entre les tensions demandées pour des paliers supérieurs à *VPpsSmallStep*.	0		275	ms	Section 7.3.16 Section 7.3.17
TPpsSrcTransSmall	Moment où le point de consigne de l'alimentation électrique programmable ***Devoir/Doit/Doivent*** effectuer une transition entre les tensions demandées pour des paliers inférieurs ou égaux à *VPpsSmallStep*.	0		25	ms	Section 7.3.16 Section 7.3.17
tPpsTransient	Durée maximale que met l'alimentation programmable pour s'établir entre *vPpsNew* et *vPpsValid* en réponse à un transitoire de charge lorsque la charge cible est supérieure ou égale à 60 mA.			5	ms	Section 7.1.8.1
	Durée maximale que met l'alimentation programmable pour s'établir entre *vPpsNew* et *vPpsValid* en réponse à un transitoire de charge lorsque la charge cible est inférieure à 60 mA.			150	ms	Section 7.1.8.1
tSrcFRSwap	Durée entre le moment où le destinataire initial a détecté que V_{BUS} a chuté sous *vSafe5V* et le moment où le destinataire initial/la nouvelle source est capable de fournir le courant USB Type-C (voir *[USB Type-C 2.0]*)			150	µs	Section 7.1.13

Paramètre	Description	MIN	TYP	MAX	UNITS	Référence
tSrcReady	Durée entre le début de la transition positive/négative (t0) et le moment où la source est prête à fournir le niveau de puissance nouvellement négocié.			285	ms	Figure 7-2, Figure 7-3
tSrcRecover	Temps imparti pour le rétablissement de la source.	0,66		1	s	Section 7.1.5
tSrcSettle	Durée entre le début de la transition positive/négative (t0) et le moment où la tension de transition est dans les limites de la plage *vSrcNew.*			275	ms	Figure 7-2
tSrcSwapStdby	Durée maximale de passage de la source en veille de permutation			650	ms	Tableau 7-9 Tableau 7-10
tSrcTransient	Durée maximale que met la tension de sortie source pour s'établir entre *vSrcNew* et *vSrcValid* en réponse à un transitoire de charge lorsque la charge cible est supérieure ou égale à 60 mA.			5	ms	Section 7.1.8
	Durée maximale que met la tension de sortie source pour s'établir entre *vSrcNew* et *vSrcValid* en réponse à un transitoire de charge lorsque la charge cible est inférieure à 60 mA.			150	ms	Section 7.1.8
tSrcTransition	Durée pendant laquelle la source ***Devoir/Doit/Doivent*** attendre avant la transition de l'alimentation afin de s'assurer que le temps de préparation du destinataire est suffisant.	25		35	ms	Section 7.3
tSrcTurnOn	Temps de transition entre *vSafe0V* et *vSafe5V*.			275	ms	Tableau 7-12 Tableau 7-13
VPpsCLCVTransient	Limites de tension transitoire de charge CF à CV.	Tension d'exploi-tation * 0,95 – 0,1 V		Tension d'exploi-tation * 1,05 + 0,1 V	V	Section 7.1.4.4

Paramètre	Description	MIN	TYP	MAX	UNITS	Référence
VPpsCVCLTransient	Limites de tension transitoire CV à CF en prenant pour hypothèse la réduction de la tension de fonctionnement de la Section 7.2.3.1.	Tension d'exploi-tation – 1,0 V		Tension d'exploi-tation + 0,5 V	V	Section 7.1.8.1
vPpsMaxVoltage	Champ Maximum Voltage dans l'APDO de l'alimentation programmable	Tension APDO * 0,95		Tension APDO * 1,05	V	Section 7.1.4.3
vPpsMinVoltage	Champ Minimum Voltage dans l'APDO de l'alimentation programmable	Tension APDO * 0,95		Tension APDO * 1,05	V	Section 7.1.4.3
vPpsNew	Tension de sortie du RDO programmable mesurée au niveau de l'embase de la source.	Tension de sortie du RDO * 0,95	Tension de sortie du RDO	Tension de sortie du RDO * 1,05	V	Section 7.1.8.1
vPpsShutdown	Tension à laquelle la PPS s'arrête lorsqu'elle fonctionne en mode CL.	Tension APDO minimale * 0,85		Tension APDO minimale * 0,95	V	Section 7.1.4.4
vPpsSlewNeg	Vitesse de balayage maximale de l'alimentation électrique programmable pour des variations de tension négatives			-30	mV/µs	Section 7.1.8.1
vPpsSlewPos	Vitesse de balayage maximale de l'alimentation électrique programmable pour des variations de tension positives			30	mV/µs	Section 7.1.8.1
VPpsSmallStep	Taille de palier PPS définie comme un palier de faible ampleur par rapport à la précédente valeur *vPpsNew*.	-500		500	mV	Section 7.1.4.3
vPpsStep	Taille de palier de programmation de tension PPS.		20		mV	Section 7.1.8.1
vPpsValid	Plage en complément de *vPpsNew* dans laquelle la sortie de l'alimentation électrique programmable est estimée ***Valide(s)*** en réponse au palier de charge.	-0,1		0,1	V	Section 7.1.8.1
vSrcNeg	Tension la plus négative admise pendant la transition.			-0,3	V	Figure 7-10

Paramètre	Description	MIN	TYP	MAX	UNITS	Référence
vSrcNew	Sortie d'alimentation fixe mesurée au niveau de l'embase source.	Tension PDO *0,95	Tension PDO	Tension PDO *1,05	V	Figure 7-2 Figure 7-3
	Sortie d'alimentation variable mesurée au niveau de l'embase source.	Tension PDO minimale		Tension PDO maximale	V	
	Sortie d'alimentation par batterie mesurée au niveau de l'embase de la source.	Tension PDO minimale		Tension PDO maximale	V	
vSrcPeak	Plage admise d'une alimentation fixe en fonctionnement à courant de crête en cas de conditions de surcharge.	Tension PDO *0,90		Tension PDO *1,05	V	Tableau 6-10 Figure 7-12
vSrcSlewNeg	Vitesse de balayage maximale admise pour des transitions de tension négative. Courants limites fondés sur les caractéristiques assignées de connecteur 3 A et une capacité de masse maximale du destinataire de 100 µF.			-30	mV/µs	Section 7.1.4.2 Figure 7-3
vSrcSlewPos	Vitesse de balayage maximale admise pour des transitions de tension positive. Courants limites fondés sur les caractéristiques assignées de connecteur 3 A et une capacité de masse maximale du destinataire de 100 µF.			30	mV/µs	Section 7.1.4, Figure 7-2
vSrcValid	Plage en plus de *vSrcNew* dans laquelle une tension nouvellement négociée est jugée comme ***Valide(s)*** pendant et après une transition. Cette plage s'applique également à *vSafe5V*.	-0,5		0,5	V	Figure 7-2 Figure 7-3
Note 1: La source ***Devoir/Doit/Doivent*** charger et décharger la capacité de masse totale afin de respecter les exigences de temps de transition.						

7.4.2 Paramètres électriques du destinataire

Les paramètres électriques du destinataire qui ***Devoir/Doit/Doivent*** être respectés sont spécifiés dans le Tableau 7-23.

Tableau 7-23 Paramètres électriques du destinataire

Paramètre	Description	MIN	TYP	MAX	UNITS	Référence
cSnkBulk[1]	Capacité de masse du destinataire sur V_{BUS} lors du branchement et au cours d'une permutation rapide des rôles après l'arrêt de l'alimentation par l'ancienne source et avant l'établissement d'un contrat explicite (voir exemple à l'Annexe E).	1		10	µF	Section 7.2.2
cSnkBulkPd	Capacité de masse sur V_{BUS} admise pour un destinataire après une négociation réussie.	1		100	µF	Section 7.2.2
iLoadReleaseRate	Libération de charge di/dt. Voir ***[USB Type-C 2.0]***, 3.7.3.3.2, pour plus d'informations sur les câbles.	-150			mA/µs	Section 7.2.6
iLoadStepRate	Palier de charge di/dt. Voir ***[USB Type-C 2.0]***, 3.7.3.3.2, pour plus d'informations sur les câbles.			150	mA/µs	Section 7.2.6
INewFrsSink	Courant maximal que le nouveau destinataire peut appeler pendant une permutation rapide des rôles, jusqu'à ce que la source applique Rp. Correspond au champ USB Type-C Current exigé du PDO Fixed Supply du message ***Sink_Capabilities*** de l'ancienne source.			Courant USB par défaut ou 1,5 ou 3,0	A	Section 7.1.13
iOvershoot	Dépassement positif ou négatif lorsqu'une variation de charge se produit et qu'elle est inférieure ou égale à ***iLoadStepRate***, par rapport à la valeur définie après la variation de charge. Voir USB ***[USB Type-C 2.0]***, 3.7.3.3.2, pour plus d'informations sur les câbles.	-230		230	mA	Section 7.2.6
IPpsCLLoadRelease	Diminution de la libération de charge maximale en mode Current Limit.	-500			mA	Section 7.2.3.1
IPpsCLLoadReleaseRate	Vitesse de balayage de diminution de charge maximale en mode Current Limit.	-150			mA/µs	Section 7.2.3.1
IPpsCLLoadStep	Augmentation du palier de charge maximale en mode Current Limit.			500	mA	Section 7.2.3.1

Paramètre	Description	MIN	TYP	MAX	UNITS	Référence
IPpsCLLoadStepRate	Vitesse de balayage d'augmentation de charge maximale en mode Current Limit.			150	mA/µs	Section 7.2.3.1
iSafe0mA	Courant maximal qu'un destinataire est autorisé à consommer lorsque V_{BUS} passe à *vSafe0V*.			1,0	mA	Figure 7-31 Figure 7-32
iSnkSwapStdby	Courant maximal qu'un destinataire peut consommer en veille de permutation. Dans l'idéal, ce courant est très proche de 0 mA et dans une large mesure influencé par le courant de fuite du port.			2,5	mA	Section 7.2.7
pHubSusp	Consommation de courant interrompue pour un hub. 25 mW + 25 mW par port en aval pour 4 port au maximum.			125	mW	Section 7.2.3
pSnkStdby	Consommation de puissance maximale en mode veille du destinataire.			2,5	W	Section 7.2.3
pSnkSusp	Consommation de courant interrompue pour dispositif périphérique.			25	mW	Section 7.2.3
tNewSrc	Temps maximal admis pour qu'un destinataire initial en veille de permutation passe au fonctionnement en tant que nouvelle source.			275	ms	Section 7.2.7 Tableau 7-9 Tableau 7-10
TSnkFRSwap	Moment d'une permutation rapide des rôles où le nouveau destinataire ne peut pas consommer plus que *pSnkStdby*.			200	µs	Section 7.1.13
tSnkHardResetPrepare	Temps attribué à l'électronique de puissance du destinataire pour se préparer à une réinitialisation matérielle.			15	ms	Tableau 7-13
tSnkNewPower	Temps de transition maximal entre des niveaux de puissance.			15	ms	Section 7.2.3
tSnkRecover	Temps que met le destinataire pour reprendre un fonctionnement USB par défaut.			150	ms	Tableau 7-12
tSnkStdby	Temps de transition en mode veille du destinataire.			15	ms	Section 7.2.3
tSnkSwapStdby	Durée maximale de passage du destinataire en veille de permutation.			15	ms	Section 7.2.7

Note 1: Si une capacité de dérivation supérieure à *cSnkBulk* max ou *cSnkBulkPd* max est exigée dans le dispositif, ce dernier ***Devoir/Doit/Doivent*** intégrer une certaine forme de courant de surcharge V_{BUS}, comme indiqué dans *[USB 3.2]*, en 11.4.4.1.

7.4.3 Paramètres électriques communs

Les paramètres électriques communs à la source et au destinataire qui ***Devoir/Doit/Doivent*** être respectés sont spécifiés dans le Tableau 7-24.

Tableau 7-24 Paramètres électriques communs source/destinataire

Paramètre	Description	MIN	TYP	MAX	UNITS	Référence
tSafe0V	Temps pour atteindre ***vSafe0V*** max.			650	ms	Section 7.1.5 Figure 7-10 Tableau 7-12 Tableau 7-13
tSafe5V	Temps pour atteindre ***vSafe5V*** max.			275	ms	Section 7.1.4.2 Figure 7-10
tVconnReapplied	• Lorsque l'UFP est la source VCONN: durée après réception du dernier bit du ***GoodCRC*** qui acquitte le message ***PS_RDY*** avant de réappliquer VCONN. • Lorsque le DFP est la source VCONN: durée à partir de laquelle VCONN chute au-dessous de vRaReconnect.	10		20	ms	Figure 7-17 Figure 7-18
tVconnValid[1]	Temps écoulé entre ***tVconnReapplied*** et le moment où VCONN se situe dans la plage vVconnValid (voir ***[USB Type-C 2.0]***).	0		5	ms	Figure 7-17 Figure 7-18
tVconnZero	Temps écoulé entre la réception du dernier bit du ***GoodCRC*** qui acquitte le message ***Accept*** en réponse au message ***Data_Reset*** et le moment où VCONN se situe au-dessous de vRaReconnect (voir ***[USB Type-C 2.0]***)..			125	ms	Figure 7-17 Figure 7-18
vSafe0V	Tension de fonctionnement sûr à "zéro volt".	0		0,8	V	Section 7.1.5
vSafe5V	Tension de fonctionnement sûr à 5 V. Voir ***[USB 2.0]*** et ***[USB 3.2]*** pour la plage de tensions V_{BUS} admise.	4,75		5,5	V	Section 7.1.5
Note 1: tVconnStable (voir ***[USB Type-C 2.0]***) continue de s'appliquer.						

8. Politique d'utilisation des dispositifs

8.1 Vue d'ensemble

La présente section décrit la politique d'utilisation des dispositifs et le moteur de politique qui la met en œuvre. Pour avoir une vue d'ensemble de l'architecture et de la manière dont le gestionnaire de politique d'utilisation des dispositifs s'y intègre, voir 2.7.

8.2 Gestionnaire de politique d'utilisation des dispositifs

Le gestionnaire de politique d'utilisation des dispositifs est chargé de gérer la puissance utilisée par un ou plusieurs ports d'alimentation électrique USB. Pour disposer des informations suffisantes pour réaliser cette tâche, il a besoin d'informations pertinentes relatives au dispositif sur lequel il réside. En premier lieu, il a une connaissance préalable du dispositif, y compris des capacités de l'alimentation et des embases sur chaque port, ces derniers présentant, par exemple, des courants assignés spécifiques. Il doit également disposer d'informations provenant du module de contrôle de port USB-C en ce qui concerne l'insertion des câbles, le type et les caractéristiques assignées du câble, etc. De même, il doit disposer d'informations provenant de l'alimentation en ce qui concerne les modifications de ses capacités et être en mesure de demander des changements d'alimentation. Avec toutes ces informations, le gestionnaire de politique d'utilisation des dispositifs est en mesure de donner des informations à jour, relatives aux capacités disponibles à un port spécifique, et de gérer les ressources de puissance à l'intérieur du dispositif.

Lors de l'utilisation des capacités pour un port source donné, le gestionnaire de politique d'utilisation des dispositifs prend d'abord en compte le courant assigné de l'embase du port et détermine si le câble inséré est apte à l'alimentation USB ou non, puis détermine la capacité de la fiche. Cela permet de définir une limite supérieure pour les capacités qui peuvent être offertes. Ensuite, le gestionnaire de politique d'utilisation des dispositifs prend en considération les ressources d'alimentation disponibles, étant donné qu'elles limitent les tensions et courants qui peuvent être offerts. Enfin, le gestionnaire de politique d'utilisation des dispositifs prend en considération la puissance actuellement attribuée à d'autres ports, la puissance qui se trouve dans la réserve de puissance et tous les autres amendements à la politique du gestionnaire de politique système. Le gestionnaire de politique d'utilisation des dispositifs offre un ensemble de capacités dans les limites décrites ci-dessus.

Lors du choix d'une capacité pour un port destinataire donné, le gestionnaire de politique d'utilisation des dispositifs regarde les capacités offertes par la source. Cela permet de définir une limite supérieure pour les capacités qui peuvent être demandées. Le gestionnaire de politique d'utilisation des dispositifs tient également compte des capacités exigées par le destinataire pour fonctionner. Si une correspondance appropriée pour la tension et le courant peut être trouvée dans les limites de l'embase et du câble, elle est demandée de la part de la source. Si aucune correspondance appropriée ne peut être trouvée, une demande de la tension et de l'intensité de courant offertes est formulée, avec une indication de l'incohérence de capacité.

USB PD définit deux types de sources d'alimentation:

- les sources de tension prédéfinies (fixe, variable et par batterie);
- la source de tension programmable (PPS).

Les premières sont généralement utilisées pour la charge classique, lorsque l'électronique du chargeur électronique se trouve à l'intérieur du destinataire. Le gestionnaire de politique d'utilisation des dispositifs du destinataire demande une tension fixe à partir de la liste de PDO offerte par la source et qui est convertie en interne pour charger la batterie du destinataire et/ou alimenter sa fonction.

La deuxième déplace l'électronique du chargeur qui gère la commande de tension hors du destinataire, puis au sein de la source elle-même. Le gestionnaire de politique d'utilisation des dispositifs du destinataire demande une tension spécifique avec une précision de 20 mV précision et définit une limite de courant. Contrairement à l'interface USB traditionnelle dans laquelle les destinataires ont la responsabilité de limiter le courant qu'ils consomment, la source PPS limite le courant selon la demande du destinataire.

Le processus de demande de puissance est le même pour les deux types de sources d'alimentation, bien que le format et le contenu réels de la demande soient légèrement différents. La principale différence opérationnelle est qu'un destinataire qui utilise la PPS est tenu d'envoyer périodiquement des demandes pour informer la source qu'il est toujours actif et communique toujours. Lorsque cette communication échoue, cela entraîne une réinitialisation matérielle.

Pour les ports d'alimentation double fonction, le gestionnaire de politique d'utilisation des dispositifs gère la fonctionnalité à la fois d'une source et d'un destinataire. De plus, il est capable de gérer le processus de permutation des rôles d'alimentation entre les deux. En ce qui concerne la gestion de puissance, cela peut vouloir dire qu'un port qui, au départ, consomme de la puissance en tant que destinataire est capable de devenir une ressource de puissance en tant que source. Par conséquent, les sources branchées peuvent demander que de la puissance leur soit fournie.

Dans le gestionnaire de politique d'utilisation des dispositifs (et dans une certaine mesure le moteur de politique), la fonctionnalité est réglable en fonction de la complexité du dispositif, y compris le nombre de capacités d'alimentation différentes, le nombre de fonctionnalités différentes prises en charge (interface de gestionnaire de politique système ou incohérence de capacité, par exemple) et le nombre de ports gérés. Dans ces paramètres, il est possible que des dispositifs soient mis en œuvre, à partir d'alimentations très simples jusqu'à des alimentations plus complexes ou des dispositifs tels que des hubs USB ou des unités de disque dur. Dans les dispositifs à plusieurs ports, il est également admis d'avoir une combinaison de ports aptes à l'alimentation électrique USB et non aptes à l'alimentation électrique USB, qu'***Il convient d'/de/qu'/que*** le gestionnaire de politique d'utilisation des dispositifs gère.

Comme indiqué en 2.7, l'architecture logique utilisée dans la spécification de l'alimentation électrique par port USB varie en fonction de la mise en œuvre. Cela signifie que différentes mises en œuvre du gestionnaire de politique d'utilisation des dispositifs peuvent être relativement petites ou volumineuses en fonction de la complexité du dispositif, comme indiqué ci-dessus. Il est également possible de répartir différentes responsabilités entre le moteur de politique et le gestionnaire de politique d'utilisation des dispositifs, ce qui donne lieu à différents types d'architectures et d'interfaces.

Le gestionnaire de politique d'utilisation des dispositifs est responsable des tâches suivantes:

- maintien de la politique locale du dispositif;
- pour une source, surveillance des capacités en présence et déclenchement de notifications de changement;
- pour un destinataire, évaluation des capacités et réponse aux demandes les concernant de la part du moteur de politique pour un port donné;
- contrôle de la source/du destinataire dans le dispositif;
- commande du module de contrôle de port USB-C pour chaque port;
- interface avec le moteur de politique pour un port donné.

Le gestionnaire de politique d'utilisation des dispositifs est responsable des fonctions ***Facultatif(s/ve/ves),***suivantes lors de la mise en œuvre:

- communications avec la politique système sur USB;
- pour les sources à plusieurs ports, surveillance et équilibrage des exigences de puissance sur ces ports;
- surveillance des batteries et des alimentations en courant alternatif;
- gestion des modes dans son port partenaire et sa ou ses fiches de câble.

8.2.1 Capacités

Le gestionnaire de politique d'utilisation des dispositifs d'un fournisseur ***Devoir/Doit/Doivent*** connaître les alimentations disponibles dans le dispositif, ainsi que leurs capacités. De plus, il ***Devoir/Doit/Doivent*** connaître l'existence de toutes les autres sources d'alimentation électriques par port USB (les batteries et les entrées en courant alternatif, par exemple). Les sources de puissance disponibles et les demandes existantes sur le dispositif ***Devoir/Doit/Doivent*** être prises en compte lors de la présentation des capacités à un destinataire.

Le gestionnaire de politique d'utilisation des dispositifs d'un consommateur ***Devoir/Doit/Doivent*** connaître les exigences du destinataire et les utiliser pour évaluer les capacités offertes par une source. Il ***Devoir/Doit/Doivent*** connaître ses propres sources d'alimentation (des batteries ou des alimentations en courant alternatif, par exemple) si elles ont une incidence sur son fonctionnement en tant que destinataire.

Le gestionnaire de politique d'utilisation des dispositifs, pour un dispositif d'alimentation double fonction, ***Devoir/Doit/Doivent*** combiner les capacités ci-dessus et ***Devoir/Doit/Doivent*** également être capable de présenter la nature double fonction du dispositif à un dispositif apte à l'alimentation USB branché.

8.2.2 Politique système

Un dispositif apte à l'alimentation électrique par port USB donné peut ne présenter aucune capacité USB, ou l'alimentation peut avoir été ajoutée à un dispositif USB de sorte que cette alimentation n'ait pas de capacité USB intégrée. Dans ces deux cas, il ne ***Devoir/Doit/Doivent*** y avoir aucune exigence que le gestionnaire de politique d'utilisation des dispositifs interagisse avec l'interface USB du dispositif. Les exigences suivantes ***Devoir/Doit/Doivent*** uniquement s'appliquer aux dispositifs aptes à l'alimentation électriques par port USB qui présentent la fonctionnalité d'alimentation sur USB.

Le gestionnaire de politique d'utilisation des dispositifs ***Devoir/Doit/Doivent*** communiquer sur USB avec le gestionnaire de politique système en fonction des exigences spécifiées dans ***[USBTypeCBridge 1.0]***. A chaque fois que cela est demandé, le gestionnaire de politique d'utilisation des dispositifs ***Devoir/Doit/Doivent*** mettre en œuvre une politique locale en fonction de celle demandée par le gestionnaire de politique système. Par exemple, le gestionnaire de politique système peut demander qu'un dispositif alimenté par batterie arrête provisoirement la charge de manière à laisser une puissance suffisante à l'unité de disque dur pour se mettre en marche.

Noter qu'en raison des contraintes de temporisation, un dispositif apte à l'alimentation USB ***Devoir/Doit/Doivent*** être capable de répondre de manière autonome à toutes les demandes à criticité temporelle liées à l'alimentation.

8.2.3 Contrôle de la source/du destinataire

Le gestionnaire de politique d'utilisation des dispositifs d'un fournisseur ***Devoir/Doit/Doivent*** gérer l'alimentation de chaque port source d'alimentation USB et ***Devoir/Doit/Doivent*** savoir à tout moment quelle est la puissance négociée. Il ***Devoir/Doit/Doivent*** demander les transitions de l'alimentation et informer le moteur de politique qu'une transition est terminée.

Le gestionnaire de politique d'utilisation des dispositifs d'un consommateur ***Devoir/Doit/Doivent*** gérer le destinataire de chaque port destinataire d'alimentation USB et ***Devoir/Doit/Doivent*** savoir à tout moment quelle est la puissance négociée.

Le gestionnaire de politique d'utilisation des dispositifs, pour un dispositif d'alimentation double fonction, ***Devoir/Doit/Doivent*** gérer la transition entre les rôles de source et de destinataire de chaque port USB d'alimentation double fonction, et ***Devoir/Doit/Doivent*** savoir à tout instant donné dans quel rôle opérationnel se trouve le port.

8.2.4 Détection de câble

8.2.4.1 Gestionnaire de politique d'utilisation des dispositifs d'un fournisseur

Le gestionnaire de politique d'utilisation des dispositifs du fournisseur ***Devoir/Doit/Doivent*** commander le module de contrôle de port USB-C et ***Devoir/Doit/Doivent*** être en mesure d'utiliser le module de contrôle de port USB-C pour déterminer le statut de branchement.

Note: Il peut s'avérer nécessaire que le gestionnaire de politique d'utilisation des dispositifs initie également une découverte supplémentaire à l'aide de la commande ***Discover Identity*** afin de déterminer toutes les capacités du câblage (voir 6.4.4.2).

8.2.4.2 Gestionnaire de politique d'utilisation des dispositifs d'un consommateur

Le gestionnaire de politique d'utilisation des dispositifs d'un consommateur commande le module de contrôle de port USB-C et ***Devoir/Doit/Doivent*** être en mesure d'utiliser le module de contrôle de port USB-C pour déterminer le statut de branchement.

8.2.4.3 Gestionnaire de politique d'utilisation des dispositifs d'un consommateur/fournisseur

Le gestionnaire de politique d'utilisation des dispositifs d'un consommateur/fournisseur hérite des caractéristiques des consommateurs et des fournisseurs, et ***Devoir/Doit/Doivent*** commander le module de contrôle de port USB-C afin de prendre en charge l'alimentation de secours de la batterie déchargée et de déterminer ce qui suit pour un port donné:

- branchement d'un fournisseur/consommateur d'alimentation électrique par port USB qui prend en charge le retour de courant d'une batterie déchargée;
- présence de V_{BUS}.

8.2.4.4 Gestionnaire de politique d'utilisation des dispositifs d'un fournisseur/consommateur

Le gestionnaire de politique d'utilisation des dispositifs d'un fournisseur/consommateur hérite des caractéristiques des consommateurs et des fournisseurs, et ***Pouvoir/Peut/Peuvent*** commander le module de contrôle de port USB-C afin de prendre en charge l'alimentation de secours de la batterie déchargée et de déterminer ce qui suit pour un port donné:

- présence de V_{BUS}.

8.2.5 Gestion des exigences d'alimentation

Le gestionnaire de politique d'utilisation des dispositifs d'un fournisseur ***Devoir/Doit/Doivent*** connaître les exigences de tous les dispositifs connectés à ses ports sources. Il s'agit notamment de connaître la réserve de puissance qui peut être exigée par les dispositifs dans le futur et de s'assurer que cette puissance est partagée de manière optimale entre les dispositifs PD branchés. Il s'agit d'une fonction essentielle du gestionnaire de politique d'utilisation des dispositifs, dont la mise en œuvre est cruciale pour s'assurer que tous les dispositifs aptes à l'alimentation électrique par port USB obtiennent la puissance qu'ils exigent en temps voulu afin de faciliter le bon fonctionnement. Cela est compensé par le fait que le gestionnaire de politique d'utilisation des dispositifs est chargé de gérer les sources de puissance qui sont, par nature, finies.

Le gestionnaire de politique d'utilisation des dispositifs du consommateur ***Devoir/Doit/Doivent*** s'assurer qu'il ne prélève pas plus de puissance que nécessaire pour exécuter ses fonctions, et qu'il rend la puissance dont il n'a pas besoin, dans toute la mesure du possible (auxquels cas, le fournisseur ***Devoir/Doit/Doivent*** maintenir une réserve de puissance pour assurer le fonctionnement ultérieur).

8.2.5.1 Gestion de la réserve de puissance

Dans certains produits, un dispositif peut avoir une certaine fonctionnalité à un niveau de puissance, et une fonctionnalité plus importante à un autre niveau (une imprimante/scanner, par exemple, qui

fonctionne comme une imprimante à un niveau de puissance, et comme un scanner si elle peut disposer de plus de puissance). La visibilité du lien entre puissance et fonctionnalité n'est apparente qu'au niveau de l'hôte USB. Toutefois, le gestionnaire de politique d'utilisation des dispositifs fournit les mécanismes de gestion des exigences de puissance de tels dispositifs.

Les dispositifs dont le fanion GiveBack est effacé indiquent le courant de fonctionnement et le courant de fonctionnement maximal (voir 6.4.1.3.4). Pour de nombreux dispositifs, le courant de fonctionnement et le courant de fonctionnement maximal sont identiques. Les dispositifs dont les charges sont très variables, comme les unités de disque dur, sont susceptibles d'utiliser le courant de fonctionnement maximal.

Les dispositifs dont le fanion GiveBack est défini indiquent le courant de fonctionnement et le courant de fonctionnement minimal (voir 6.4.1.3.4). Pour de nombreux dispositifs, le courant de fonctionnement et le courant de fonctionnement minimal sont identiques. Les dispositifs qui chargent leurs propres batteries peuvent utiliser le courant de fonctionnement minimal et le fanion GiveBack.

Par exemple, dans le premier cas, il est possible qu'un dispositif mobile exige 500 mA pour fonctionner, mais souhaite 1 000 mA supplémentaires pour charger sa batterie. Le dispositif mobile définit le fanion GiveBack (voir 6.4.2.2) et demande 500 mA dans le champ Minimum Operating Current et 1 500 mA dans le champ Operating Current (à condition que 1 500 mA soient offerts par la source), indiquant au fournisseur qu'il peut temporairement récupérer les 1 000 mA pour répondre à une demande transitoire.

Dans le second cas, une unité de disque dur (UDD) peut exiger 2 A pour se mettre en marche, mais uniquement 1 A pour fonctionner. Au démarrage, l'unité de disque dur demande un courant de fonctionnement maximal de 2 A et un courant de fonctionnement de 2 A. Lorsque l'unité est mise en marche et prête à fonctionner, elle formule une autre demande, de 1 A pour son courant de fonctionnement et de 2 A pour son courant de fonctionnement maximal. Au fil du temps, ses temporisateurs d'inactivité peuvent expirer. L'unité de disque dur passe alors à un état de puissance inférieur. A la prochaine sollicitation, l'unité de disque dur doit de nouveau se mettre en marche. Elle demande donc un courant de fonctionnement de 2 A et un courant de fonctionnement maximal de 2 A. Le fournisseur peut disposer de la puissance supplémentaire disponible immédiatement, et il peut honorer la demande immédiatement. Si la puissance n'est pas disponible, le fournisseur est susceptible de devoir récupérer de la puissance, par exemple d'utiliser le message ***GotoMin*** pour récupérer de la puissance et honorer la demande de l'unité de disque dur. Dans ce cas, un message ***Wait*** demande à l'unité de disque dur de patienter. L'unité de disque dur continue de demander de la puissance supplémentaire tant que sa demande n'est pas satisfaite.

L'allocation de la puissance et le maintien d'une réserve de puissance ***Devoir/Doit/Doivent*** relever de la responsabilité du gestionnaire de politique d'utilisation des dispositifs, de manière à ne pas exagérément solliciter ses ressources de puissance disponibles. Un dispositif à plusieurs ports (un hub, par exemple) ***Devoir/Doit/Doivent*** toujours être capable de satisfaire aux demandes supplémentaires du port en exigeant la puissance la plus élevée de sa réserve de puissance.

Le message ***GotoMin*** permet au fournisseur de récupérer de la puissance auprès d'un port afin de prendre en charge un consommateur sur un autre port qui exige provisoirement de la puissance supplémentaire pour exécuter une opération à court terme. Dans l'exemple ci-dessus, le dispositif mobile en charge réduit son régime de charge pour permettre à un gestionnaire de politique d'utilisation des dispositifs de répondre à une demande de courant de démarrage formulée par une unité de disque dur, pour lui permettre de mettre en marche ses disques. Toute la puissance disponible récupérée à l'aide d'un message ***GotoMin*** ***Pouvoir/Peut/Peuvent*** être vue comme une réserve de puissance.

Un consommateur qui demande de la puissance ***Devoir/Doit/Doivent*** tenir compte de ses exigences opérationnelles lorsqu'il annonce son aptitude à renvoyer provisoirement de la puissance. Par exemple, ***Il convient d'/de/qu'/que*** un dispositif mobile avec une batterie déchargée utilisé pour procéder à un appel demande à garder une puissance suffisante pour poursuivre l'appel. Si les exigences du consommateur changent, il ***Devoir/Doit/Doivent*** renégocier sa puissance pour refléter les exigences modifiées.

8.2.5.2 Incohérence de capacité de puissance

Une incohérence de capacité se produit lorsqu'un consommateur ne peut pas obtenir la puissance exigée de la part d'un fournisseur (ou que la source n'est pas apte à l'alimentation USB) et que ce consommateur

exige de telles capacités pour fonctionner. Différentes actions sont réalisées par le gestionnaire de politique d'utilisation des dispositifs et le gestionnaire de politique système dans ce cas.

8.2.5.2.1 Gestion d'incohérence par le dispositif local

Le gestionnaire de politique d'utilisation des dispositifs du consommateur ***Devoir/Doit/Doivent*** être à l'origine de l'affichage d'un message adressé à l'utilisateur final indiquant qu'une incohérence de capacité de puissance s'est produite. Des exemples de ce type de rétroactions peuvent inclure:

- pour un dispositif simple, une LED ***Pouvoir/Peut/Peuvent*** être utilisée pour indiquer une défaillance. Par exemple, pendant la connexion, la LED peut être fixe et de couleur orange. Si la connexion a abouti, la LED peut passer au vert. Si la connexion échoue, elle peut passer au rouge ou clignoter orange;
- ***Il convient d'/de/qu'/que*** un dispositif plus sophistiqué avec une interface utilisateur (un dispositif mobile ou un moniteur, par exemple) fournisse une notification par l'intermédiaire de l'interface utilisateur sur le dispositif.

Le gestionnaire de politique d'utilisation des dispositifs ***Pouvoir/Peut/Peuvent*** être à l'origine de l'affichage d'un message adressé à l'utilisateur indiquant une incohérence de capacité de puissance.

Comme l'incohérence de capacité de puissance peut ne pas provoquer de défaillance d'exploitation, ***Il convient de ne pas*** que le gestionnaire de politique d'utilisation des dispositifs du fournisseur n'affiche pas de message à l'utilisateur si la puissance offerte au destinataire est conforme ou supérieure à la PDP minimale du destinataire annoncée dans le message ***Sink_Capabilities_Extended*** (voir 6.5.13). Si un message est affiché, il ***Il convient de ne pas*** le présenter comme une erreur, sauf si la puissance offerte au destinataire est inférieure à la PDP minimale du destinataire annoncée dans le message ***Sink_Capabilities_Extended***.

8.2.5.2.2 Communication du gestionnaire de politique d'utilisation des dispositifs avec la politique système

Dans un système apte à l'alimentation électrique par port USB avec un gestionnaire de politique d'utilisation des dispositifs actif (voir 8.2.2), le gestionnaire de politique d'utilisation des dispositifs ***Devoir/Doit/Doivent*** informer le gestionnaire de politique système de l'incohérence. Cette information ***Devoir/Doit/Doivent*** être retransmise au gestionnaire de politique système à l'aide des mécanismes décrits dans ***[USBBridge 1.1]***. ***Il convient d'/de/qu'/que*** le gestionnaire de politique système s'assure que l'utilisateur a été informé de la condition. Si un autre port du système peut satisfaire à l'exigence de puissance du consommateur, ***Il convient d'/de/qu'/que*** inviter l'utilisateur à déplacer le dispositif vers un autre port.

Pour identifier un port source plus approprié pour le consommateur, le gestionnaire de politique système ***Devoir/Doit/Doivent*** communiquer avec le gestionnaire de politique d'utilisation des dispositifs afin de déterminer les exigences du consommateur. Le gestionnaire de politique d'utilisation des dispositifs ***Devoir/Doit/Doivent*** utiliser un message ***Get_Sink_Cap*** (voir 6.3.8) pour découvrir les niveaux de puissance que le consommateur peut utiliser.

8.2.6 Utilisation du bit "Unconstrained Power" avec les batteries et les alimentations en courant alternatif

Le gestionnaire de politique d'utilisation des dispositifs d'un fournisseur ou d'un consommateur ***Pouvoir/Peut/Peuvent*** surveiller le statut de toute source de puissance variable pouvant avoir un impact sur ses capacités en tant que source (les batteries et les alimentations en courant alternatif, par exemple) et refléter cet impact dans le bit "Unconstrained Power" (voir 6.4.1.2.2.3 et 6.4.1.3.1.3) qui fait partie du message de capacités de la source ou du destinataire (voir 6.4.1). S'ils sont surveillés, et si une interface USB est prise en charge, le statut de l'alimentation externe (voir ***[USBTypeCBridge 1.0]***) et l'état de la batterie (voir 9.4.1) ***Devoir/Doit/Doivent*** également être indiqués au gestionnaire de politique système à l'aide de l'interface USB.

8.2.6.1 Alimentations en courant alternatif

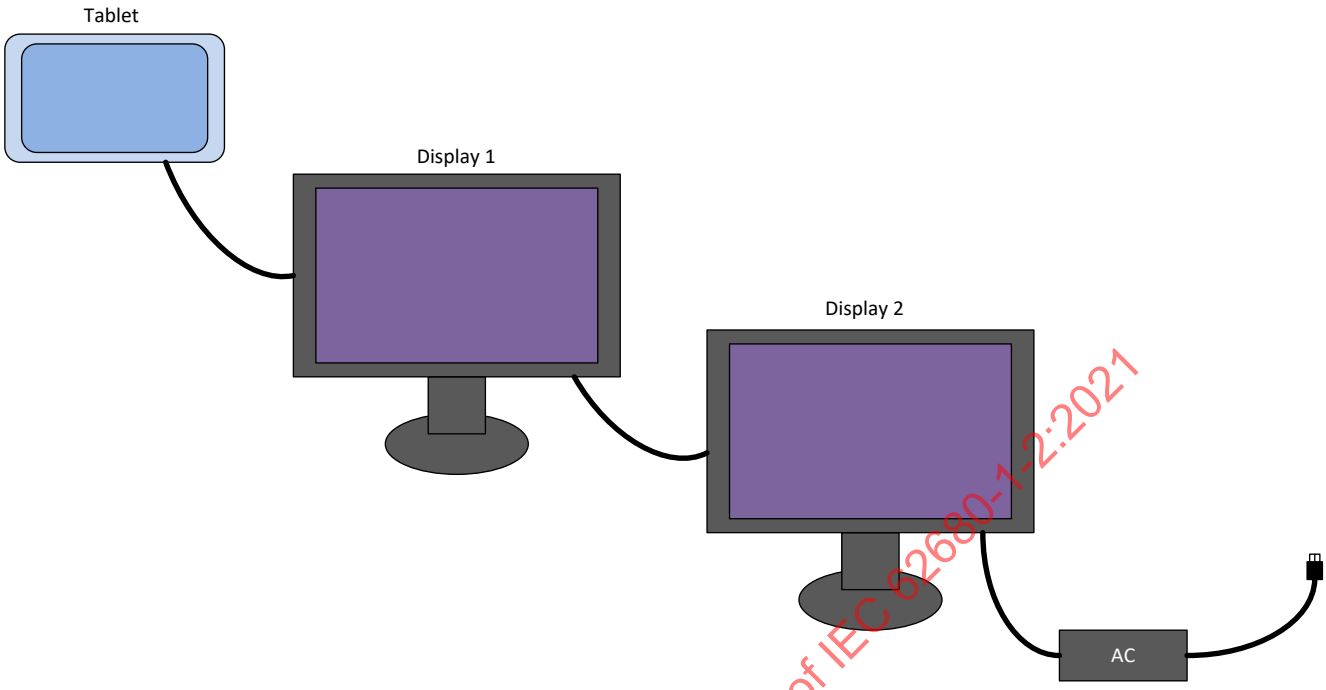
Le bit "Unconstrained Power" fourni par les sources et les destinataires (voir 6.4.1.2.2.3 et 6.4.1.3.1.3) indique un dispositif connecté acceptable pour utiliser la puissance annoncée pour la charge et pour tout ce qui s'avère nécessaire au fonctionnement normal. Un dispositif qui définit le bit "Unconstrained Power" dispose d'une source de puissance externe suffisante pour alimenter correctement le système tout en chargeant les dispositifs externes, ou bien prévoit de charger des dispositifs externes comme un état de fonction principal (un bloc de batteries, par exemple).

Dans le cas de la source de puissance externe, la puissance peut soit provenir de l'alimentation en courant alternatif connectée directement au dispositif, soit d'une alimentation en courant alternatif connectée à un dispositif branché, qui obtient également une puissance non contrainte auprès de son alimentation. De cette manière, le bit "Unconstrained Power" est communiqué par l'intermédiaire d'un système PD indiquant que la puissance provient d'une seule ou de plusieurs alimentations en courant alternatif, d'un bloc de batteries ou d'éléments similaires:

- si le bit "Unconstrained Power" est défini, cette puissance provient d'une alimentation en courant alternatif;
- les dispositifs capables de consommer sur plusieurs ports peuvent uniquement revendiquer qu'ils disposent d'une "puissance non contrainte" pour la puissance annoncée en tant que port fournisseur, en présence d'une puissance non contrainte supérieure à celle nécessaire au fonctionnement normal provenant d'alimentations externes (plusieurs alimentations en courant alternatif, par exemple);
- ce concept s'applique lorsque la puissance est acheminée par plusieurs niveaux de fournisseur et de consommateur, par exemple. La puissance fournie par un moniteur connecté à un moniteur qui tire sa puissance d'une alimentation en courant alternatif revendique une "puissance non contrainte" même s'il n'est pas directement connecté à l'alimentation en courant alternatif.

Une tablette numérique utilisée avec deux écrans A/V USB en guirlande (voir Figure 8-1) est un exemple de cas d'utilisation. La tablette et le premier écran ne sont pas alimentés en externe (ce qui signifie qu'ils ne disposent d'aucune source de puissance en dehors de l'alimentation USB). Le deuxième écran dispose d'une alimentation externe branchée, qui peut être une alimentation par dispositif USB ou une autre forme d'alimentation externe. Lorsque les écrans sont connectés comme indiqué, le bloc d'alimentation externe branché au deuxième écran est capable d'alimenter le premier écran et la tablette. Dans ce cas, le deuxième écran signale la présence d'un bloc d'alimentation de taille suffisante pour le premier écran en attribuant une valeur à son bit "Unconstrained Power". Le premier écran procède ensuite à son tour à une évaluation et indique la présence de puissance supplémentaire à la tablette en attribuant une valeur à son bit "Unconstrained Power". La puissance est transmise à tous les dispositifs par l'intermédiaire du système, à condition que la puissance disponible provenant de l'alimentation externe soit suffisante.

Figure 8-1 Exemple d'écrans en guirlande



Anglais	Français
Tablet	Tablette
Display	Ecran

Un autre exemple de cas d'utilisation est un ordinateur portable branché à une alimentation externe et à une tablette numérique. Dans cette situation, si l'alimentation externe est suffisamment importante pour alimenter l'ordinateur portable dans son état normal et charger un dispositif externe, l'ordinateur portable attribue une valeur à son bit "Unconstrained Power" et la tablette s'autorise elle-même à charger à son taux de crête. Toutefois, si la puissance externe est faible et n'empêche pas l'ordinateur portable de se décharger si sa puissance maximale est consommée par le dispositif externe, l'ordinateur portable n'attribue pas de valeur à son bit "Unconstrained Power" et la tablette peut choisir de consommer moins de puissance que ce dont elle dispose. Cette quantité peut être juste suffisante pour empêcher la tablette de se décharger, ou pas du tout. Par ailleurs, si la tablette détermine que l'ordinateur portable dispose d'une batterie bien plus importante avec plus de charge qu'elle-même n'en dispose, elle peut toujours choisir de se charger elle-même, bien que la vitesse risque de ne pas être maximale.

De cette manière, les destinataires qui ne reçoivent pas le bit "Unconstrained Power" de la source connectée peuvent toujours choisir de charger leurs batteries ou de se recharger à une vitesse réduite, si leur politique détermine que l'impact sur la source est minimal (comme dans le cas d'un téléphone équipé d'une petite batterie et qui se recharge sur un ordinateur portable équipé d'une grosse batterie, par exemple). Ces stratégies peuvent être décidées dans le cadre d'une autre communication d'alimentation USB.

8.2.6.2 Alimentations par batterie

S'il est surveillé, et si une interface USB est prise en charge, l'état de la batterie ***Devoir/Doit/Doivent*** être indiqué au gestionnaire de politique système à l'aide de l'interface USB.

Si le dispositif est alimenté par batterie, mais que son état lui permet principalement de charger les dispositifs externes, il est vu comme une source de puissance non contrainte et, de ce fait, ***Il convient d'/de/qu'/que*** définir le bit "Unconstrained Power".

Un algorithme simplifié est décrit ci-dessous afin de s'assurer que les dispositifs alimentés par batterie soient chargés à partir de dispositifs qui ne sont pas alimentés par batterie, dans la mesure du possible, et

de s'assurer également que les dispositifs ne procèdent pas constamment à une permutation des rôles d'alimentation.

Si deux dispositifs sont connectés et qu'ils ne disposent pas d'une puissance non contrainte, ***Il convient d'/de/qu'/que*** ils définissent leurs propres politiques de manière à éviter une permutation des rôles d'alimentation constante.

Cet algorithme utilisant le bit "Unconstrained Power" (voir 6.4.1.2.2.3 et 6.4.1.3.1.3), les décisions s'appuient sur la disponibilité et la suffisance d'une alimentation externe, et non sur les capacités totales d'un système, d'un dispositif ou d'un produit.

Recommandations:

1. ***Il convient d'/de/qu'/que*** le fournisseur/les consommateurs utilisant des sources externes volumineuses (bit "Unconstrained Power" défini) refusent toujours les demandes de permutation des rôles d'alimentation provenant d'un consommateur/de fournisseurs n'utilisant pas de source externe (bit "Unconstrained Power" effacé).
2. ***Il convient d'/de/qu'/que*** le fournisseur/les consommateurs n'utilisant pas de sources externes volumineuses (bit "Unconstrained Powered" effacé) acceptent systématiquement une demande de permutation des rôles d'alimentation provenant d'un consommateur/fournisseur utilisant des sources de puissance externes volumineuses (bit "Unconstrained Power" défini), sauf si le demandeur n'est pas en mesure de fournir les exigences du fournisseur/consommateur actuel.

8.2.7 Interface avec le moteur de politique

Le gestionnaire de politique d'utilisation des dispositifs ***Devoir/Doit/Doivent*** assurer une interface avec le moteur de politique pour chaque port du dispositif.

8.2.7.1 Gestionnaire de politique d'utilisation des dispositifs d'un fournisseur

Le gestionnaire de politique d'utilisation des dispositifs d'un fournisseur ***Devoir/Doit/Doivent*** également fournir les fonctions suivantes au moteur de politique:

- informer le moteur de politique des modifications du statut de branchement du câble/dispositif d'un câble donné;
- informer le moteur de politique de toute modification des capacités de source disponibles pour un port;
- évaluer les demandes d'un consommateur branché et apporter des réponses au moteur de politique;
- répondre aux demandes de transitions d'alimentation formulées par le moteur de politique;
- indiquer au moteur de politique que les transitions d'alimentation sont terminées;
- garder une réserve de puissance pour les dispositifs fonctionnant sur un port à une puissance inférieure à la puissance maximale.

8.2.7.2 Gestionnaire de politique d'utilisation des dispositifs d'un consommateur

Le gestionnaire de politique d'utilisation des dispositifs d'un consommateur ***Devoir/Doit/Doivent*** également fournir les fonctions suivantes au moteur de politique:

- informer le moteur de politique des modifications du statut de branchement du câble/dispositif;
- informer le moteur de politique de toute modification des exigences de puissance pour un port;
- évaluer les capacités de source et apporter des réponses adaptées:
  - gérer les demandes de capacités offertes;
  - indiquer si de la puissance supplémentaire est exigée;
- répondre aux demandes de transitions de destinataire formulées par le moteur de politique.

8.2.7.3 Gestionnaire de politique d'utilisation des dispositifs pour un dispositif d'alimentation double fonction

Le gestionnaire de politique d'utilisation des dispositifs, pour un dispositif d'alimentation double fonction, ***Devoir/Doit/Doivent*** fournir les fonctions suivantes au moteur de politique:

- gestionnaire de politique d'utilisation des dispositifs du fournisseur;
- gestionnaire de politique d'utilisation des dispositifs du consommateur;
- interface du moteur de politique pour demander des transitions d'alimentation entre la source et le destinataire et inversement;
- indications au moteur de politique pendant les transitions de permutation des rôles d'alimentation.

8.2.7.4 Gestionnaire de politique d'utilisation des dispositifs pour le traitement de la batterie déchargée d'un dispositif d'alimentation double fonction

Il convient d'/de/qu'/que le gestionnaire de politique d'utilisation des dispositifs, pour un dispositif d'alimentation double fonction dont la batterie est déchargée:

- commute les ports pour un fonctionnement du destinataire uniquement ou à destination du DFP pour obtenir la puissance auprès de la source branchée ensuite;
- utilise V_{BUS} à partir de la source branchée pour alimenter les communications par alimentation USB et la charge, afin de permettre la négociation d'une puissance d'entrée plus élevée.

8.3 Moteur de politique

8.3.1 Introduction

Il existe une instance de moteur de politique par port, qui interagit avec le gestionnaire de politique d'utilisation des dispositifs afin de mettre en œuvre la politique locale en cours pour ce port en particulier. La présente section inclut:

- des séquences de message correspondant à différentes opérations;
- des diagrammes d'état couvrant le fonctionnement des sources, des destinataires et des fiches de câbles.

8.3.2 Schémas de séquence atomique de messages

8.3.2.1 Introduction

Le moteur de politique d'utilisation des dispositifs achemine les séquences de messages et les réponses en fonction des séquences de message prévues et de la politique locale en cours.

Une AMS ***Devoir/Doit/Doivent*** être définie en tant que séquence de message qui commence et/ou se termine à l'état ***PE_SRC_Ready***, ***PE_SNK_Ready*** ou ***PE_CBL_Ready*** (voir 8.3.3.2, 8.3.3.3 et 8.3.3.24).

De plus, la séquence de découverte de la fiche de câble spécifiée en 8.3.3.24.3 ***Devoir/Doit/Doivent*** être définie en tant qu'AMS.

La source et le destinataire indiquent à la couche protocole à quel moment un AMS commence et se termine sur une entrée dans/une sortie de ***PE_SRC_Ready*** ou ***PE_SNK_Ready*** (voir 8.3.3.2 et 8.3.3.3).

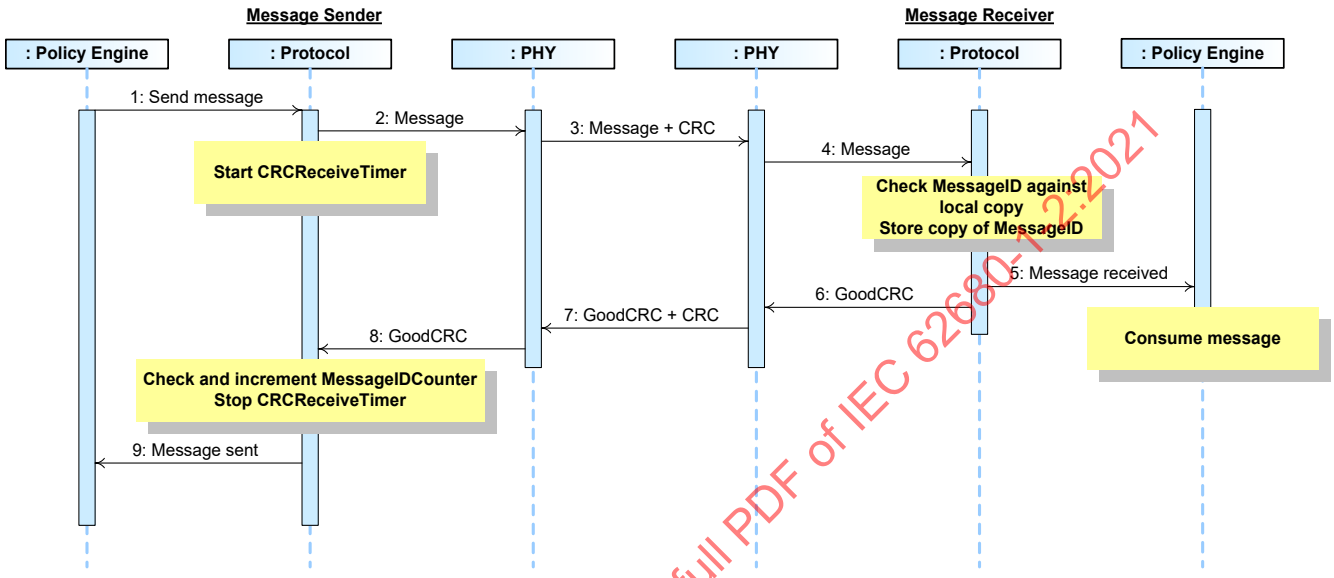
La Section 8.3.2.1.3 indique en détail les AMS interruptibles ou non interruptibles.

Cette section contient des diagrammes de séquence qui mettent en évidence certaines des transactions les plus intéressantes. Il ne s'agit en aucun cas d'un récapitulatif exhaustif de toutes les combinaisons possibles, mais est donné à titre de représentation.

8.3.2.1.1 Echange de message de base

La Figure 8-2 Echange de messages de base (réussi) représente la manière dont un message est envoyé. Noter que l'expéditeur peut être une source ou un destinataire tandis que le récepteur peut être un destinataire ou une source. La séquence de message de base est identique. Elle commence lorsque la couche protocole de l'expéditeur du message forme, sur l'ordre de son moteur de politique, un message qu'il transmet à la couche physique.

Figure 8-2 Echange de messages de base (réussi)



Anglais	Français
Policy Engine	Moteur de politique
Protocol	Protocole
Message Sender	Emetteur du message
PHY	PHY
Send message	Envoyer un message
Message	Message
Message + CRC	Message + CRC
Message Received	Message reçu
Message Sent	Message envoyé
Start CRCReceiveTimer	Lancer le CRCReceiveTimer
Check and increment MessageIDCounter	Vérifier et incrémenter le MessageIDCounter
Stop CRCReceiveTimer	Arrêter le CRCReceiveTimer
Check Message ID against local copy	Vérifier le Message ID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Consume message	Consommer le message

Tableau 8-1 Flux de messages de base

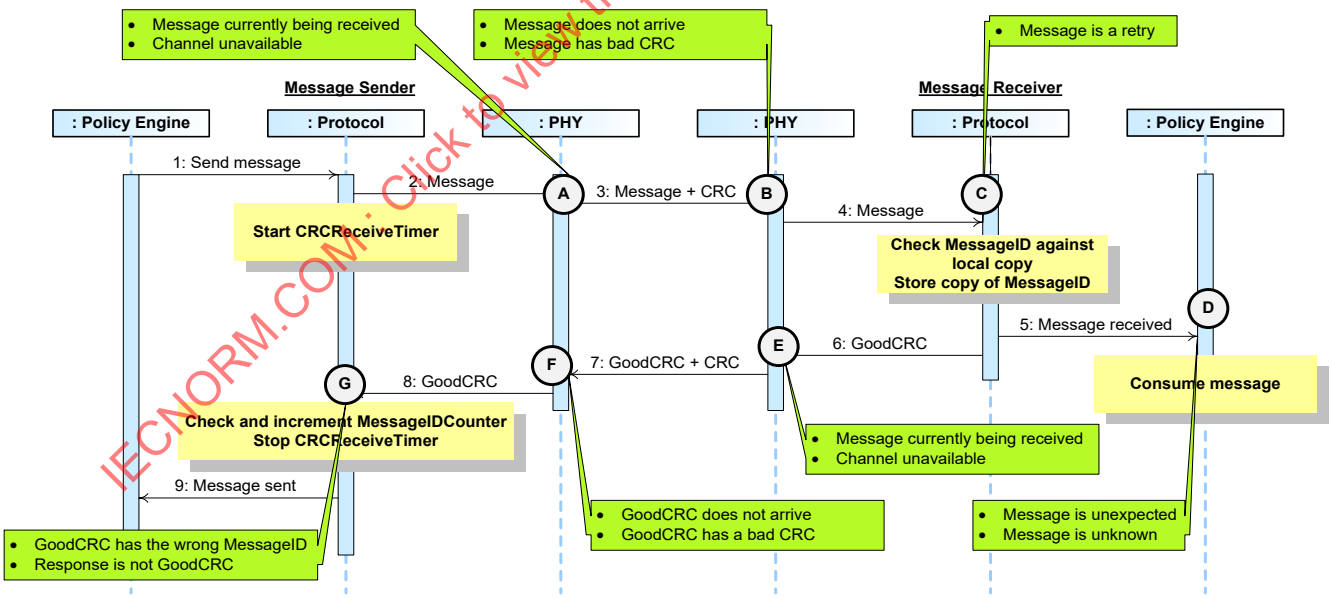
Etape	Expéditeur de message	Destinataire du message
1	Le moteur de politique demande à la couche protocole d'envoyer un message.	

Etape	Expéditeur de message	Destinataire du message
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message.	La couche physique reçoit le message et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole. La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***.	
9	La couche protocole informe le moteur de politique que l'envoi du message a abouti.	

8.3.2.1.2 Erreurs dans le flux de messages de base

Lors du flux de messages, il existe différents points au niveau desquels des défaillances de communication ou d'autres problèmes peuvent se produire. La Figure 8-3 est une version annotée de la Figure 8-2 indiquant à quels points des problèmes peuvent apparaître.

Figure 8-3 Flux de messages de base indiquant de possibles erreurs



Anglais	Français
Policy Engine	Moteur de politique
Protocol	Protocole
Message Sender	Emetteur du message
Message Receiver	Récepteur du message
PHY	PHY

Send message	Envoyer un message
Message	Message
Message Received	Message reçu
Message Sent	Message envoyé
Start CRCReceiveTimer	Lancer le CRCReceiveTimer
Check and increment MessageIDCounter	Vérifier et incrémenter le MessageIDCounter
Stop CRCReceiveTimer	Arrêter le CRCReceiveTimer
Check Message ID against local copy	Vérifier le Message ID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Consume message	Consommer le message
Message currently being received	Message en cours de réception
Channel unavailable	Canal non disponible
Message does not arrive	Le message n'arrive pas
Message has bad CRC	Le message a un mauvais CRC
Message is a retry	Le message est une relance
GoodCRC has the wrong MessageID	GoodCRC a le mauvais MessageID
Response is not GoodCRC	La réponse n'est pas GoodCRC
GoodCRC does not arrive	GoodCRC n'arrive pas
GoodCRC has a bad CRC	GoodCRC a un mauvais CRC
Message currently being received	Message en cours de réception
Channel unavailable	Canal non disponible
Message is unexpected	Le message est inattendu
Message is unknown	Le message est inconnu

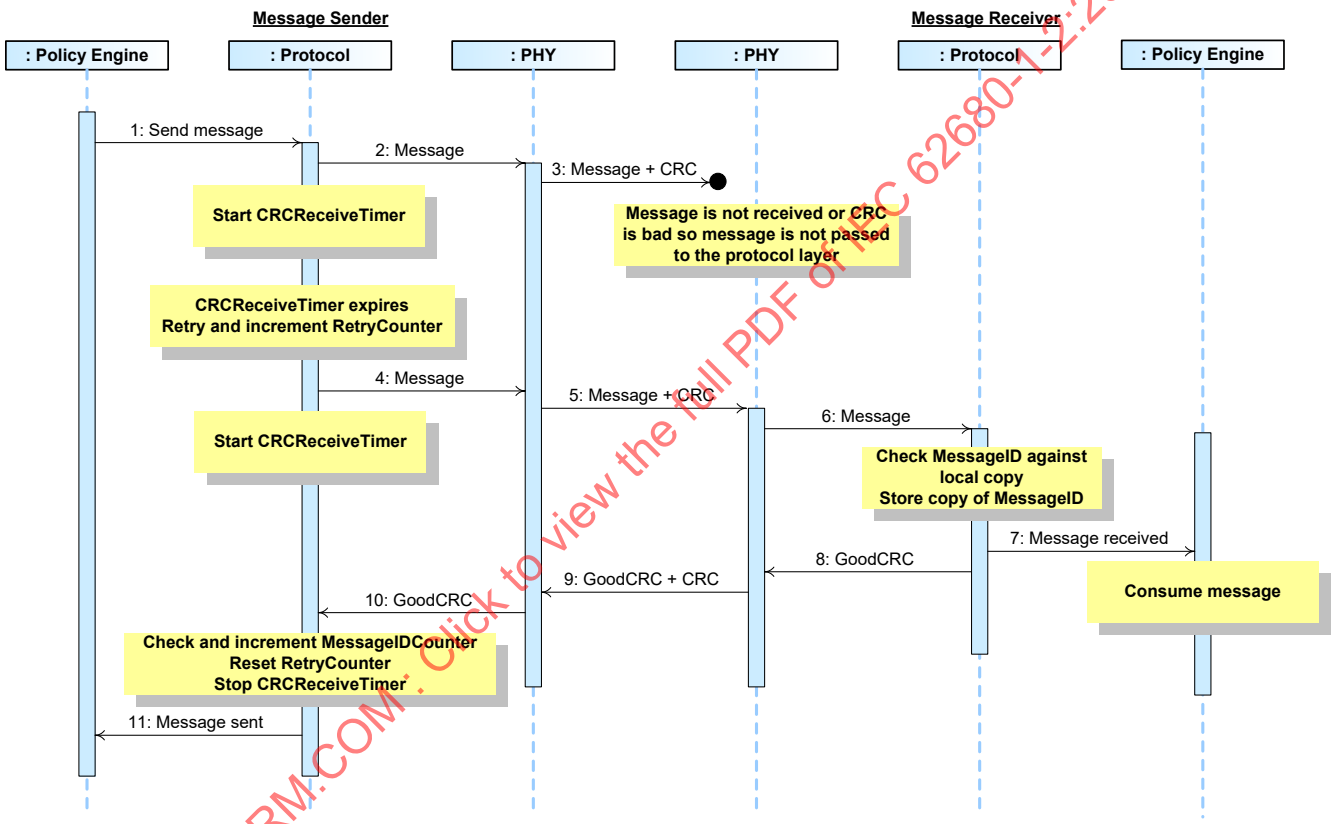
Tableau 8-2 Problèmes potentiels dans le flux de messages de base

Point	Problèmes possibles
A	1. Il y a un message entrant sur le canal, ce qui signifie que la couche PHY ne peut pas procéder à l'envoi. Dans ce cas, le message sortant est supprimé de la file d'attente et le message entrant est traité. 2. En raison de bruit sur la ligne, la transmission est impossible. Dans ce cas, le message sortant est ***Rejeté(s/ée/ées)*** par la couche PHY. La retransmission a lieu à l'aide du mécanisme normal de la couche protocole.
B	1. Le message n'arrive pas à la couche physique en raison de bruit sur le canal. 2. Le message arrive, mais a été corrompu et présente un CRC erroné. Il n'y a aucun message à transmettre à la couche protocole sur le récepteur, ce qui signifie qu'un message ***GoodCRC*** n'est pas envoyé. Cela donne lieu à une temporisation ***CRCReceiveTimer*** dans l'expéditeur de message.
C	1. Le ***MessageID*** du message reçu correspond au ***MessageID*** stocké. Il s'agit donc d'une relance. Le message n'est pas transmis au moteur de politique.
D	1. Le moteur de politique reçoit un message connu qui n'était pas attendu. 2. Le moteur de politique reçoit un message non reconnu. Ces cas sont des erreurs de protocole qui donnent lieu à la génération d'un message ***Soft_Reset***.
E	Comme pour le point A, mais côté récepteur du message.
F	1. La réponse au message ***GoodCRC*** n'arrive pas du côté de l'expéditeur du message en raison de bruit sur le canal. 2. La réponse au message ***GoodCRC*** arrive, mais son CRC est erroné. Un message ***GoodCRC*** n'est pas reçu par la couche protocole de l'expéditeur du message. Cela donne lieu à une temporisation ***CRCReceiveTimer*** dans l'expéditeur de message.

Point	Problèmes possibles
G	1. Le message ***GoodCRC*** est reçu, mais il contient le même ***MessageID*** que le message émis. 2. Un message est reçu, mais il ne s'agit pas d'un message ***GoodCRC*** (cas similaire à celui d'un message inattendu ou inconnu, mais cette fois détecté dans la couche protocole). Ces deux problèmes indiquent des erreurs de réception d'un message ***GoodCRC*** attendu, donnant lieu à une temporisation ***CRCReceiveTimer*** dans la couche protocole et à une nouvelle tentative subséquente (sauf pour les communications avec les fiches de câbles).

La Figure 8-4 représente l'un de ces cas: le flux de messages de base avec une relance en raison d'un CRC erroné au niveau du récepteur de message. Elle commence lorsque la couche protocole de l'expéditeur du message forme, sur l'ordre de son moteur de politique, un message qu'il transmet à la couche physique. La couche protocole est chargée des relances, sur la base du principe de "n essais et vous êtes éliminé" (***nRetryCount***).

Figure 8-4 Flux de messages de base avec CRC erroné suivi d'une relance



Anglais	Français
Policy Engine	Moteur de politique
Protocol	Protocole
Message Sender	Emetteur du message
Message Receiver	Récepteur du message
PHY	PHY
Send message	Envoyer un message
Message	Message
Message Received	Message reçu
Message Sent	Message envoyé
Start CRCReceiveTimer	Lancer le CRCReceiveTimer

Check and increment MessageIDCounter	Vérifier et incrémenter le MessageIDCounter
Stop CRCReceiveTimer	Arrêter le CRCReceiveTimer
Check Message ID against local copy	Vérifier le Message ID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Consume message	Consommer le message
Message is not received or CRC is bad so message is not passed to the protocol layer	Le message n'est pas reçu ou le CRC est mauvais, donc le message n'est pas transmis à la couche protocole
Start CRCReceiveTimer	Lancer le CRCReceiveTimer
CRCReceiveTimer expires	Le CRCReceiveTimer expire
Retry and increment RetryCounter	Relancer et incrémenter le RetryCounter

Tableau 8-3 Flux de messages de base avec défaillance CRC

Etape	Expéditeur de message	Destinataire du message
1	Le moteur de politique demande à la couche protocole d'envoyer un message.	
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message.	La couche physique ne reçoit pas de message ou reçoit un message avec un CRC incorrect. Rien n'a été transmis à la couche protocole.
4	Etant donné qu'aucune réponse n'est reçue, ***CRCReceiveTimer*** expire et déclenche la première relance par la couche protocole. ***RetryCounter*** est incrémenté. La couche protocole transmet le message à la couche physique. Elle lance le ***CRCReceiveTimer***.	
5	La couche physique ajoute un CRC et envoie le message.	La couche physique reçoit le message et contrôle le CRC pour vérifier le message.
6		La couche physique supprime le CRC et transfère le message vers la couche protocole.
7		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message reçu au moteur de politique qui le consomme.
8		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
9	La couche physique reçoit le message et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
10	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
11	La couche protocole vérifie le ***MessageID***, arrête ***CRCReceiveTimer*** et réinitialise ***RetryCounter***. La couche protocole informe le moteur de politique que l'envoi du message a abouti.	

8.3.2.1.3 Séquences atomiques de messages interruptibles et non interruptibles

Le Tableau 8-4 présente en détail les AMS (définies en 8.3.2) qui ***Devoir/Doit/Doivent*** être traitées comme étant interruptibles ou non interruptibles pendant la séquence. Chaque AMS qui commence par le même message ***Devoir/Doit/Doivent*** obéir à l'exigence interruptible/non interruptible. Noter que chaque AMS est interruptible jusqu'à ce que le premier message de la séquence soit envoyé avec succès (message ***GoodCRC*** reçu). Une séquence de VDM ***Devoir/Doit/Doivent*** être interruptible. A l'issue de l'AMS à l'origine de l'interruption, si l'AMS originale est toujours nécessaire, l'AMS interrompue ***Devoir/Doit/Doivent*** être de nouveau exécutée.

Tableau 8-4 AMS interruptible et non interruptible

AMS	Interruptible	Référence
Négociation de puissance	Non	Sections 8.3.3.2, 8.3.3.3
GotoMin	Non	Section 8.3.3.2, Section 8.3.3.3
Réinitialisation logicielle	Non	Section 8.3.3.4
Réinitialisation des données	Non	Section 8.3.2.4
Réinitialisation matérielle	Non	Section 8.3.3.2, Section 8.3.3.3
Réinitialisation de câble	Non	Section 8.3.3.24.2.3
Obtention des capacités de source	Non	Section 8.3.3.2, Section 8.3.3.3
Obtention des capacités de destinataire	Non	Section 8.3.3.2, Section 8.3.3.3
Permutation des rôles d'alimentation	Non	Section 8.3.3.18.3, Section 8.3.3.18.4
Permutation rapide des rôles	Non	Section 8.3.3.18.5, Section 8.3.3.18.6
Permutation des rôles de transmission de données	Non	Section 8.3.3.18.1, Section 8.3.3.18.2
Permutation de VCONN	Non	Section 8.3.3.19
Alerte de la source	N/A	Section 8.3.3.8
Obtention des capacités étendues de source	Non	Section 8.3.3.9
Obtention de statut de la source/du destinataire	Non	Section 8.3.3.10
Obtention des capacités de la batterie	Non	Section 8.3.3.11
Obtention du statut de la batterie	Non	Section 8.3.3.12
Obtention des informations du fabricant	Non	Section 8.3.3.13
Sécurité	Oui	Section 8.3.3.14
Mise à jour du micrologiciel	Oui	Section 8.3.3.17
Découverte d'identité	Oui	Section 8.3.3.20.1, Section 8.3.3.21.1
Découverte d'identité de la fiche de câble au démarrage de la source	Oui	Paragraphe 8.3.3.20.1, paragraphe 8.3.3.24.3
Découverte de SVID	Oui	Section 8.3.3.20.2, Section 8.3.3.21.2
Découverte de modes	Oui	Section 8.3.3.20.3, Section 8.3.3.21.3
Mode d'entrée DFP vers UFP	Oui	Section 8.3.3.22.1, Section 8.3.3.23.1
Mode de sortie DFP vers UFP	Oui	Section 8.3.3.22.2, Section 8.3.3.23.2
Mode d'entrée DFP vers fiche de câble	Oui	Section 8.3.3.22.1, Section 8.3.3.24.4.1
Mode de sortie DFP vers fiche de câble	Oui	Section 8.3.3.22.1, Section 8.3.3.24.4.2
Attention	N/A	Section 8.3.3.20.4
Autotest intégré (BIST)	Non	Section 8.3.2.14
Séquence de VDM non structurés	Oui	Section 6.4.4.1
Séquence de VDM structurés utilisant des commandes de fournisseur	Oui	Section 6.4.4.2
Informations de pays	Oui	Section 8.3.2.10.8
Entrée en mode USB	Non	Section 8.3.2.15

AMS	Interruptible	Référence
Codes pays	Oui	Section 8.3.2.10.7

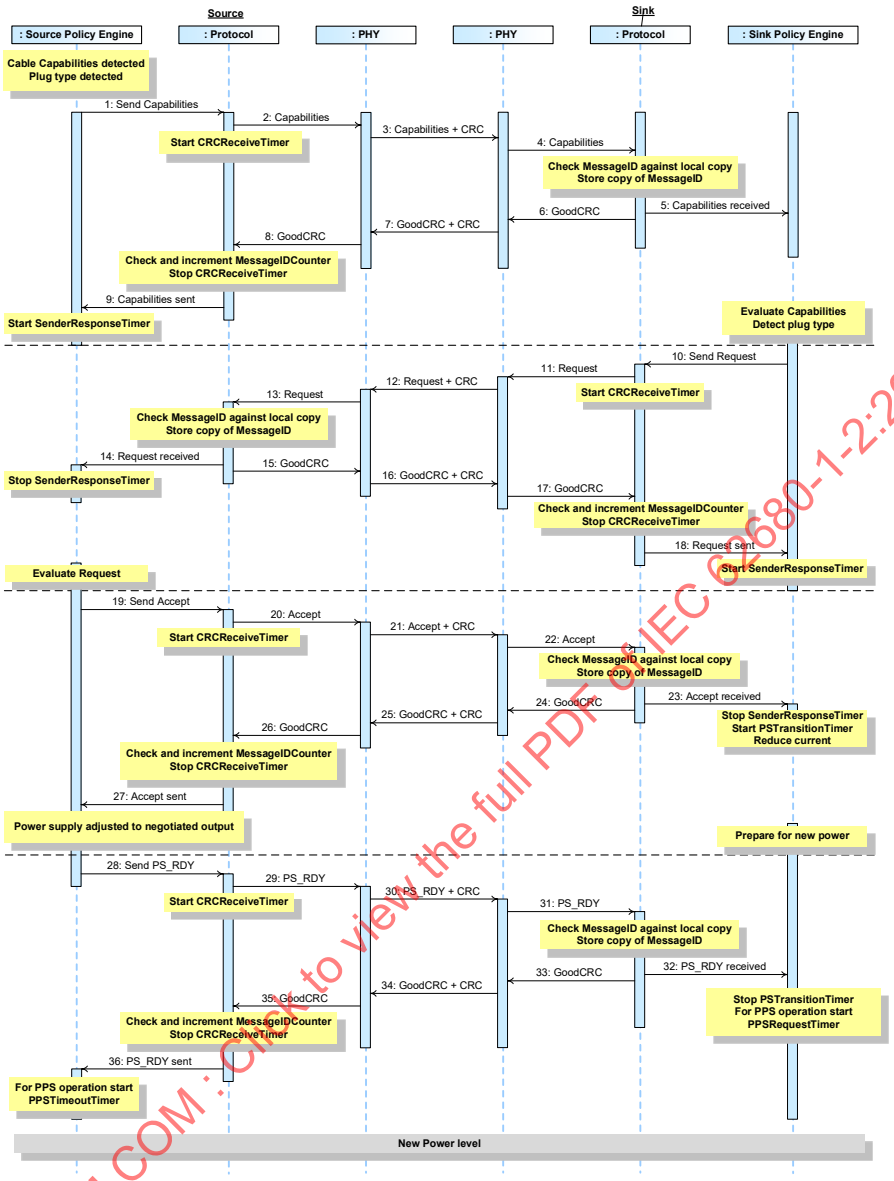
8.3.2.2 Négociation de puissance

8.3.2.2.1 Négociation de contrat explicite

La Figure 8-5 donne un exemple de flux de messages ayant abouti pendant la négociation de puissance. La négociation passe par 5 phases distinctes:

- la source envoie ses capacités de puissance dans un message ***Source_Capabilities***;
- le destinataire évalue ces capacités et choisit, dans la phase de demande, un niveau de puissance en envoyant un message ***Request***;
- la source évalue la demande et l'accepte avec un message ***Accept***;
- la source passe au nouveau niveau de puissance, puis informe le destinataire en envoyant un message ***PS_RDY***;
- le destinataire commence à utiliser le nouveau niveau de puissance.
- Pour le fonctionnement PPS:
  - la source lance son temporisateur de maintien;
  - le destinataire lance son temporisateur de demande pour l'envoi de messages ***Request*** périodiques.

Figure 8-5 Négociation réussie d'alimentation fixe, variable ou par batterie



Anglais	Français
Cable Capabilities detected	Capacités de câble détectées
Capabilities	Capacités
Capabilities + CRC	Capacités + CRC
Capabilities Received	Capacités reçues
Capabilities Sent	Capacités envoyées
Check and increment MessageIDCounter	Vérifier et incrémenter le MessageIDCounter
Check Message ID against local copy	Vérifier le Message ID par rapport à la copie locale
Detect plug type	Détecter le type de fiche
Evaluate Capabilities	Evaluer les capacités
Evaluate Request	Evaluer la demande
PHY	PHY

Plug type detected	Type de fiche détecté
Protocol	Protocole
Request	Demande
Request Received	Demande reçue
Send Capabilities	Envoyer les capacités
Send Request	Envoyer une demande
Sink	Destinataire
Sink Policy Engine	Moteur de politique du destinataire
Source	Source
Source Policy Engine	Moteur de politique de la source
Start CRCReceiveTimer	Lancer le CRCReceiveTimer
Start SenderResponseTimer	Lancer le SenderResponseTimer
Start SenderResponseTimer	Lancer le SenderResponseTimer
Stop CRCReceiveTimer	Arrêter le CRCReceiveTimer
Stop SenderResponseTimer	Arrêter le SenderResponseTimer
Store copy of MessageID	Stocker une copie du MessageID
Power supply adjusted to negotiated output	Alimentation réglée à la sortie négociée
Send Ping if required to maintain activity	Envoyer un ping si nécessaire pour maintenir l'activité
Accept	Accept
Send Acccept	Envoyer Accept
Accept received	Accept reçu
Start PSTransitionTimer	Lancer le PSTransitionTimer
Reduce Current	Réduire le courant
Prepare for new power	Préparer pour la nouvelle puissance
Send	Envoyer
Sent	Envoyé
Stop PSTransitionTimer	Arrêter le PSTransitionTimer
New Power Level	Nouveau niveau de puissance

Le Tableau 8-5 ci-dessous donne une explication précise de ce qu'il se passe à chaque étape indiquée sur la Figure 8-5 ci-dessus.

Tableau 8-5 Etapes d'une négociation de puissance réussie

Etape	Source	Destinataire
1	Les capacités du câble ou le type de fiche sont détectés s'ils ne sont pas déjà connus (voir 4.4). Le moteur de politique demande à la couche protocole d'envoyer un message ***Source_Capabilities*** qui donne les capacités présentes de l'alimentation.	
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Source_Capabilities***.	La couche physique reçoit le message ***Source_Capabilities*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Source_Capabilities*** vers la couche protocole.

Etape	Source	Destinataire
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Source_Capabilities*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Source_Capabilities*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique évalue le message ***Source_Capabilities*** envoyé par la source, détecte le type de fiche si cela s'avère nécessaire (voir 4.4) et choisit la puissance souhaitée. Il demande à la couche protocole de former les données (objet de données d'alimentation, par exemple) qui représentent sa demande dans un message.
11		La couche protocole crée le message ***Request*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message ***Request*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***Request***.
13	La couche physique supprime le CRC et transfère le message ***Request*** vers la couche protocole.	
14	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transmet les informations de demande au moteur de politique. Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message.	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique transfère le message ***GoodCRC*** à la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***. Elle informe le moteur de politique que l'envoi du message ***Request*** a abouti. La couche protocole arrête le ***CRCReceiveTimer***. Le moteur de politique lance le ***SenderResponseTimer***.
19	Le moteur de politique évalue le message ***Request*** envoyé par le destinataire et décide s'il peut satisfaire à la demande. Il demande à la couche protocole de former un message ***Accept***.	

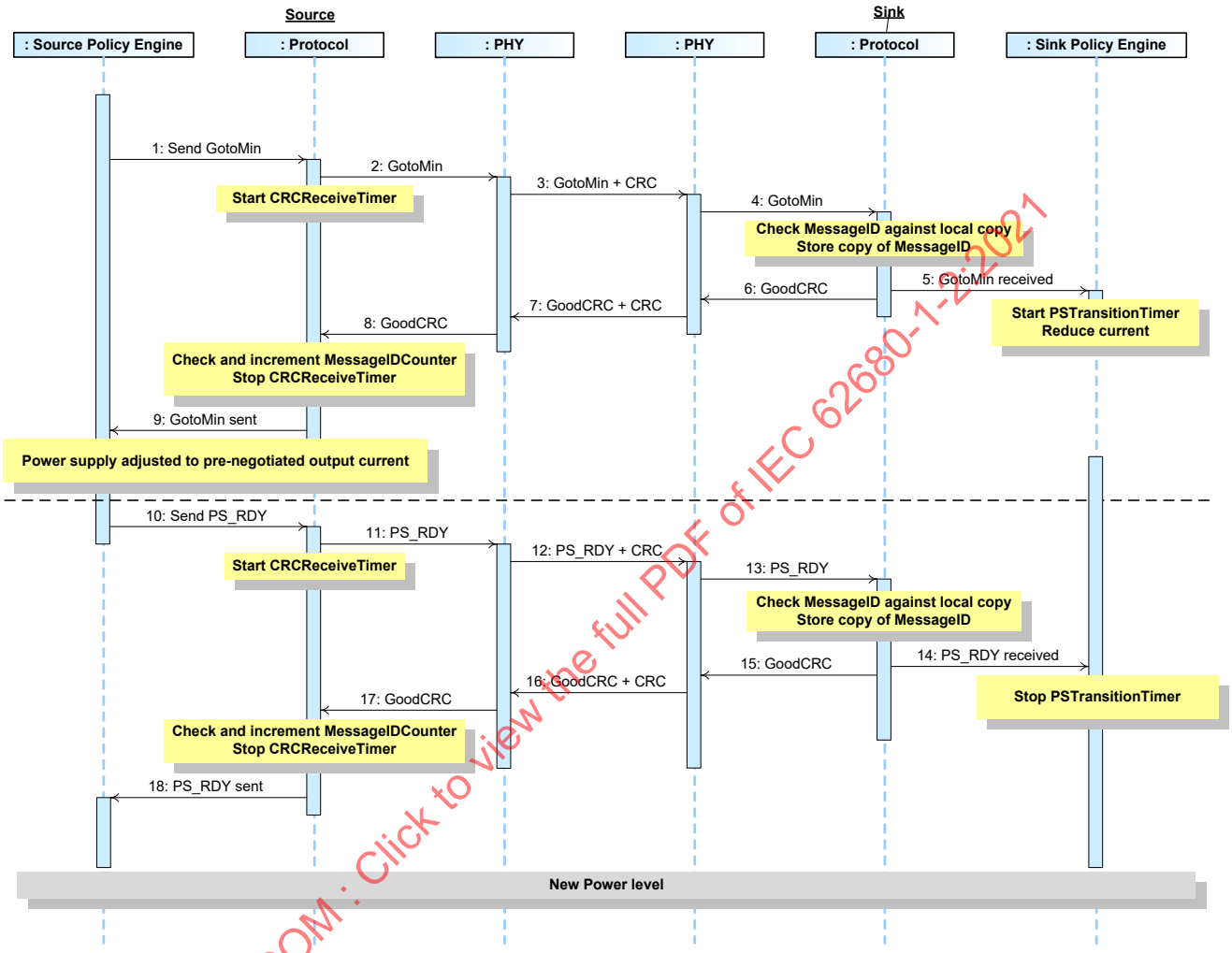
Etape	Source	Destinataire
20	La couche protocole forme le message ***Accept*** transmis à la couche physique et lance le ***CRCReceiveTimer***.	
21	La couche physique ajoute un CRC et envoie le message ***Accept***.	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
22		La couche physique transfère le message ***Accept*** à la couche protocole.
23		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole informe le moteur de politique que la réception d'un message ***Accept*** a abouti. Le moteur de politique arrête le ***SenderResponseTimer***, lance le ***PSTransitionTimer*** et réduit sa consommation de courant. Le gestionnaire de politique d'utilisation des dispositifs prépare l'alimentation à la transition vers le nouveau niveau de puissance.
24		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
25	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message.
26	La couche physique transfère le message ***GoodCRC*** à la couche protocole. La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***.	
27	La couche protocole informe le moteur de politique que l'envoi d'un message ***Accept*** a abouti.	
L'alimentation ajuste sa sortie à la valeur négociée		
28	Le gestionnaire de politique d'utilisation des dispositifs informe le moteur de politique que l'alimentation est réglée à la nouvelle condition de fonctionnement et demande à la couche protocole d'envoyer un message ***PS_RDY***.	
29	La couche protocole forme le message ***PS_RDY*** et lance le ***CRCReceiveTimer***.	
30	La couche physique ajoute un CRC et envoie le message ***PS_RDY***.	La couche physique reçoit le message ***PS_RDY*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
31		La couche physique transfère le message ***PS_RDY*** à la couche protocole.
32		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole informe le moteur de politique qu'un message RS_RDY a été reçu. Le moteur de politique arrête le ***PSTransitionTimer***. Lors du fonctionnement PPS, le moteur de politique lance le ***SinkPPSPeriodicTimer***.
33		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
34	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message.

Etape	Source	Destinataire
35	La couche physique transfère le message ***GoodCRC*** à la couche protocole. La couche protocole vérifie et incrémente le ***MessageIDCounter***. Elle arrête le ***CRCReceiveTimer***.	
36	La couche protocole informe le moteur de politique que l'envoi du message ***PS_RDY*** a abouti.	
37	Lors du fonctionnement PPS, le moteur de politique lance le ***SourcePPSCommTimer***.	

8.3.2.2.2 Récupération de puissance avec le message GotoMin

Il s'agit d'un exemple d'opération GotoMin. La Figure 8-6 représente les messages au fur et à mesure de leur circulation à travers le bus et dans les dispositifs afin d'accomplir GotoMin.

Figure 8-6 Opération GotoMin réussie



Anglais	Français
Check and increment MessageIDCounter	Vérifier et incrémenter le MessageIDCounter
Check Message ID against local copy	Vérifier le Message ID par rapport à la copie locale
New Power Level	Nouveau niveau de puissance
PHY	PHY
Power supply adjusted to pre-negotiated output current	Alimentation réglée au courant de sortie préalablement négocié
Protocol	Protocole
Reduce Current	Réduire le courant
Send	Envoyer
Sent	Envoyé
Sink	Destinataire
Sink Policy Engine	Moteur de politique du destinataire
Source	Source

Source Policy Engine	Moteur de politique de la source
Start CRCReceiveTimer	Lancer le CRCReceiveTimer
Start PSTransitionTimer	Lancer le PSTransitionTimer
Stop CRCReceiveTimer	Arrêter le CRCReceiveTimer
Stop PSTransitionTimer	Arrêter le PSTransitionTimer
Store copy of MessageID	Stocker une copie du MessageID
Received	Reçu

Le Tableau 8-6 donne une explication précise de ce qu'il se passe à chaque étape indiquée sur la Figure 8-6 ci-dessus.

Tableau 8-6 Etapes d'une négociation GotoMin

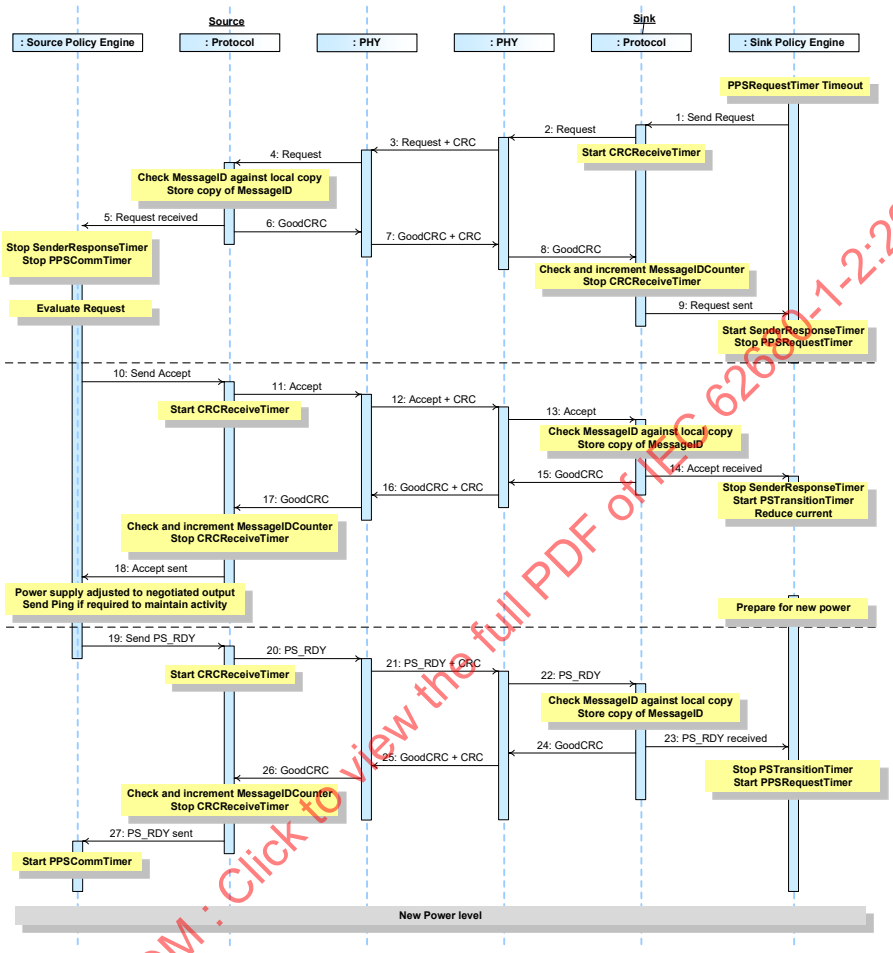
Etape	Source	Destinataire
1	Le moteur de politique demande à la couche protocole de former un message ***GotoMin***.	
2	La couche protocole forme le message ***GotoMin*** transmis à la couche physique et lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***GotoMin***.	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
4		La couche physique transfère le message ***GotoMin*** à la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole informe le moteur de politique que la réception d'un message ***GotoMin*** a abouti. La politique lance le ***PSTransitionTimer*** et réduit sa consommation de courant. Le moteur de politique prépare l'alimentation à la transition vers le nouveau niveau de puissance.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message.
8	La couche physique transfère le message ***GoodCRC*** à la couche protocole. La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***.	
9	La couche protocole informe le moteur de politique que l'envoi d'un message ***GotoMin*** a abouti.	
	L'alimentation ajuste sa sortie à la valeur négociée	
10	Le moteur de politique constate que l'alimentation a été réglée à la nouvelle condition de fonctionnement et demande à la couche protocole d'envoyer un message ***PS_RDY***.	
11	La couche protocole forme le message ***PS_RDY*** et lance le ***CRCReceiveTimer***.	
12	La couche physique ajoute un CRC et envoie le message ***PS_RDY***.	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
13		La couche physique transfère le message ***PS_RDY*** à la couche protocole.

Etape	Source	Destinataire
14		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole informe le moteur de politique que la réception d'un message ***PS_RDY*** a abouti. Le moteur de politique arrête le ***PSTransitionTimer***.
15		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
16	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message.
17	La couche physique transfère le message ***GoodCRC*** à la couche protocole. La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***.	
18	La couche protocole informe le moteur de politique que l'envoi du message ***PS_RDY*** a abouti.	

8.3.2.2.3 Maintien de PPS

Voici un exemple d'opération de maintien de PPS au cours d'un contrat explicite, la PPS étant l'APDO. La Figure 8-7 représente les messages au fur et à mesure de leur circulation à travers le bus et dans les dispositifs afin de procéder au maintien.

Figure 8-7 Maintien de PPS



Anglais	Français
Policy Engine	Moteur de politique
Protocol	Protocole
Send Soft Reset	Envoyer Soft Reset
Reset	Réinitialiser
stored	enregistré
Check and increment	Vérifier et incrémenter
Stop	Arrêter
Start	Démarrer
Store copy of	Enregistrer une copie
Reset Complete, Explicit Contract negotiation	Réinitialisation terminée, négociation de contrat explicite
Soft Reset received	Soft Reset reçu
Soft Resent Sent	Soft Resent envoyé
Send Accept	Envoyer Accept

Accept sent	Accept envoyé
Accept received	Accept reçu

Le Tableau 8-7 ci-dessous donne une explication précise de ce qu'il se passe à chaque étape indiquée sur la Figure 8-5 ci-dessus.

Tableau 8-7 Etapes d'une négociation de puissance réussie

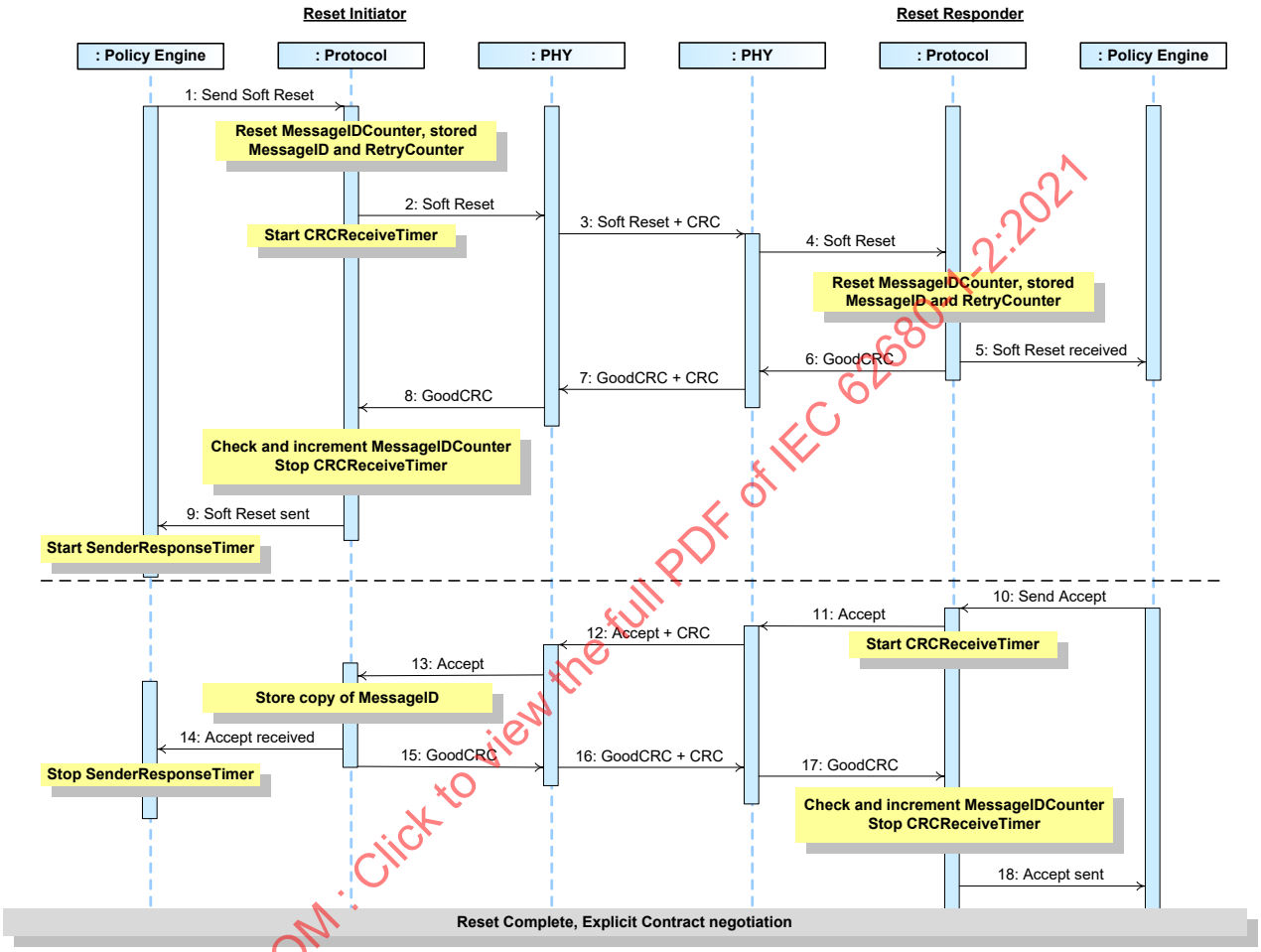
Etape	Source	Destinataire
1		Le ***SinkPPSPeriodicTimer*** expire dans le moteur de politique. Le moteur de politique demande à la couche protocole de former un message ***Request***. La couche protocole crée le message ***Request*** et le transmet à la couche physique. La couche protocole lance le ***CRCReceiveTimer***.
2	La couche physique reçoit le message ***Request*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***Request***.
3	La couche physique supprime le CRC et transfère le message ***Request*** vers la couche protocole.	
4	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transmet les informations de demande au moteur de politique. Le moteur de politique arrête le ***SourcePPSCommTimer.***	
5	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
6	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
7		La couche physique transfère le message ***GoodCRC*** à la couche protocole.
8		La couche protocole vérifie et incrémente le ***MessageIDCounter***. Elle informe le moteur de politique que l'envoi du message ***Request*** a abouti. La couche protocole arrête le ***CRCReceiveTimer***. Le moteur de politique lance le ***SenderResponseTimer***.
9	Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'évaluer le message ***Request*** envoyé par le destinataire et décide si la source peut répondre à la demande. Le moteur de politique demande à la couche protocole de former un message ***Accept***.	
10	La couche protocole forme le message ***Accept*** transmis à la couche physique et lance le ***CRCReceiveTimer***.	
11	La couche physique ajoute un CRC et envoie le message ***Accept***.	La couche physique reçoit le message ***Accept*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
12		La couche physique transfère le message ***Accept*** à la couche protocole.

Etape	Source	Destinataire
13		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole informe le moteur de politique que la réception d'un message ***Accept*** a abouti. Le moteur de politique arrête le ***SenderResponseTimer***, lance le ***PSTransitionTimer*** et réduit sa consommation de courant. Le gestionnaire de politique d'utilisation des dispositifs prépare l'alimentation à la transition vers le nouveau niveau de puissance.
14		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
15	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
16	La couche physique transfère le message ***GoodCRC*** à la couche protocole. La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***.	
17	La couche protocole informe le moteur de politique que l'envoi d'un message ***Accept*** a abouti.	
L'alimentation ajuste sa sortie à la valeur négociée		
18	Le gestionnaire de politique d'utilisation des dispositifs informe le moteur de politique que l'alimentation est réglée à la nouvelle condition de fonctionnement et demande à la couche protocole d'envoyer un message ***PS_RDY***.	
19	La couche protocole forme le message ***PS_RDY*** et lance le ***CRCReceiveTimer***.	
20	La couche physique ajoute un CRC et envoie le message ***PS_RDY***.	La couche physique reçoit le message ***PS_RDY*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
21		La couche physique transfère le message ***PS_RDY*** à la couche protocole.
22		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole informe le moteur de politique qu'un message RS_RDY a été reçu. Le moteur de politique arrête le ***PSTransitionTimer***. Lors du fonctionnement PPS, le moteur de politique lance le ***SinkPPSPeriodicTimer***.
23		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
24	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
25	La couche physique transfère le message ***GoodCRC*** à la couche protocole. La couche protocole vérifie et incrémente le ***MessageIDCounter***. Elle arrête le ***CRCReceiveTimer***.	
26	La couche protocole informe le moteur de politique que l'envoi du message ***PS_RDY*** a abouti.	
27	Lors du fonctionnement PPS, le moteur de politique lance le ***SourcePPSCommTimer***.	

8.3.2.3 Réinitialisation logicielle

Il s'agit d'un exemple d'opération de réinitialisation logicielle. La Figure 8-8 représente les messages au fur et à mesure de leur circulation à travers le bus et dans les dispositifs afin de procéder à la réinitialisation logicielle.

Figure 8-8 Réinitialisation logicielle



Anglais	Français
Accept	Accept
Accept received	Accept reçu
Accept sent	Accept envoyé
Check and increment MessageIDCounter	Vérifier et incrémenter le MessageIDCounter
Explicit Contract negotiation	Négociation de contrat explicite
PHY	PHY
Policy Engine	Moteur de politique
Protocol	Protocole
Reset Complete	Réinitialisation terminée
Reset Initiator	Initiateur de la réinitialisation
Reset MessageIDCounter	Réinitialiser le MessageIDCounter
Send Accept	Envoyer Accept
Send Soft Reset	Envoyer Soft Reset
Soft Reset	Réinitialisation logicielle

Soft Reset received	Soft Reset reçu
Soft Reset Sent	Soft Reset envoyé
Start SenderResponseTimer	Lancer le SenderResponseTimer
Stop CRCReceiveTimer	Arrêter le CRCReceiveTimer
Stored Message ID and RetryCounter	Message ID et RetryCounter stockés
Start CRCReceiveTimer	Lancer le CRCReceiveTimer

Le Tableau 8-8 ci-dessous donne une explication précise de ce qu'il se passe à chaque étape indiquée sur la Figure 8-8 ci-dessus.

Tableau 8-8 Etapes d'une réinitialisation logicielle

Etape	Initiateur de réinitialisation	Répondeur de réinitialisation
1	Le moteur de politique demande à la couche protocole de générer un message ***Soft_Reset*** pour demander une réinitialisation logicielle.	
2	La couche protocole réinitialise le ***MessageIDCounter***, le ***MessageID*** stocké et le ***RetryCounter***. La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Soft_Reset***.	La couche physique reçoit le message ***Soft_Reset*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Soft_Reset*** vers la couche protocole.
5		La couche protocole ne vérifie pas le ***MessageID*** dans le message entrant et réinitialise le ***MessageIDCounter***, le ***MessageID*** stocké et le ***RetryCounter***. La couche protocole transfère les informations du message ***Soft_Reset*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Soft_Reset*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande à la couche protocole de former un message ***Accept***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message.
13	La couche protocole stocke le ***MessageID*** du message entrant.	
14	La couche protocole transfère les informations du message ***Accept*** reçu au moteur de politique qui le consomme.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	

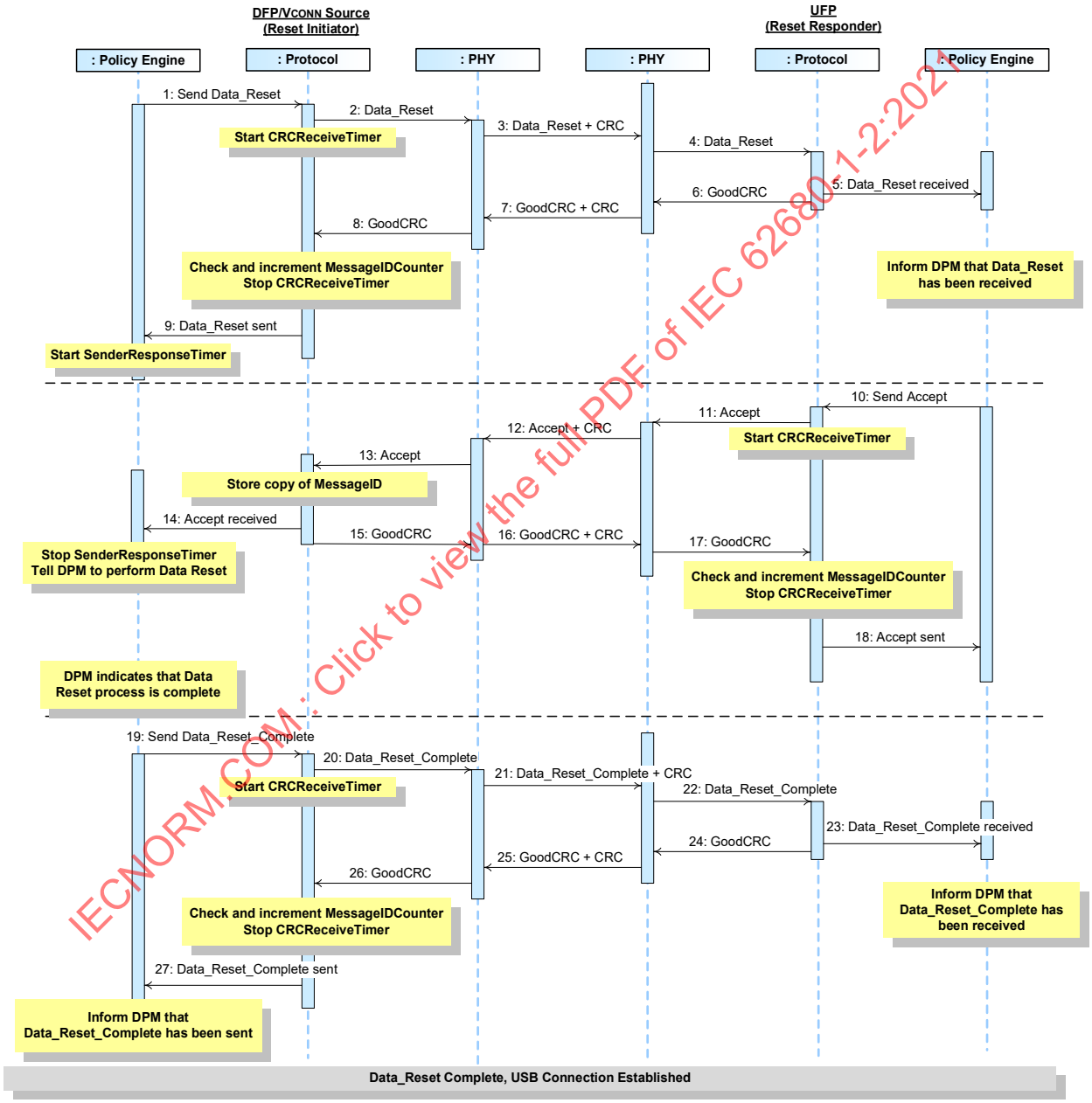
Etape	Initiateur de réinitialisation	Répondeur de réinitialisation
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Accept*** a abouti.
	La réinitialisation est terminée et la communication de protocole peut redémarrer. Les ports partenaires procèdent à une négociation du contrat explicite pour resynchroniser leurs diagrammes d'états.	

8.3.2.4 Réinitialisation des données

8.3.2.4.1 Réinitialisation des données initiée par le DFP, où le DFP est la source VCONN

Il s'agit d'un exemple d'opération de réinitialisation des données où le DFP est également la source VCONN et initie une réinitialisation des données. La Figure 8-9 représente les messages au fur et à mesure de leur circulation à travers le bus et dans les dispositifs afin de procéder à la réinitialisation des données.

Figure 8-9 Réinitialisation des données initiée par le DFP, où le DFP est la source Vconn



Anglais	Français
DFP/Vconn Source (Reset Initiator)	DFP/Source Vconn (Initiateur de réinitialisation)
UFP (Reset Responder)	UFP (Répondeur de réinitialisation)
Policy Engine	Moteur de politique

Protocol	Protocole
Send Data_Reset	Envoyer Data_Reset
Data_Reset received	Data_Reset reçu
Data_Reset sent	Data_Reset envoyé
Start CRCReceiveTimer	Lancer le CRCReceiveTimer
Check and increment MessageIDCounter	Vérifier et incrémenter le MessageIDCounter
Stop CRCReceiveTimer	Arrêter le CRCReceiveTimer
Inform DPM that Data_Reset has been received	Informer le DPM de la reception de Data_Reset
Start SenderResponseTimer	Lancer le SenderResponseTimer
Send Accept	Envoyer Accept
Accept received	Accept reçu
Accept sent	Accept envoyé
Store Copy of Message ID	Stocker une copie du Message ID
Stop SenderResponseTimer	Arrêter le SenderResponseTimer
Tell DPM to perform Data Reset	Demander au DPM de procéder à une réinitialisation des données
DPM indicates that Data Reset process is complete	Le DPM indique que le processus de réinitialisation des données est terminé
Send Data_Reset_Complete	Envoyer Data_Reset_Complete
Data_Reset_Complete received	Data_Reset_Complete reçu
Data_Reset_Complete sent	Data_Reset_Complete envoyé
Inform DPM that Data_Reset_Complete has been received	Informer le DPM de la reception de Data_Reset_Complete
Inform DPM that Data_Reset_Complete has been sent	Informer le DPM de l'envoi de Data_Reset_Complete
Data_Reset Complete, USB Connection Established	Data_Reset terminée, connexion USB établie

Le Tableau 8-9 ci-dessous donne une explication précise de ce qu'il se passe à chaque étape indiquée sur la Figure 8-9 ci-dessus.

Tableau 8-9 Etapes d'une réinitialisation des données initiée par le DFP, où le DFP est la source Vconn

Etape	DFP/Source VCONN (initiateur de réinitialisation)	UFP (répondeur de réinitialisation)
1	Le moteur de politique demande à la couche protocole de générer un message ***Data_Reset*** pour demander une réinitialisation des données.	
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Data_Reset***.	La couche physique reçoit le message ***Data_Reset*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Data_Reset*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Data_Reset*** reçu au moteur de politique qui le consomme. Le moteur de politique informe le gestionnaire de politique d'utilisation des dispositifs que la réception d'un message ***Data_Reset*** a abouti.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.

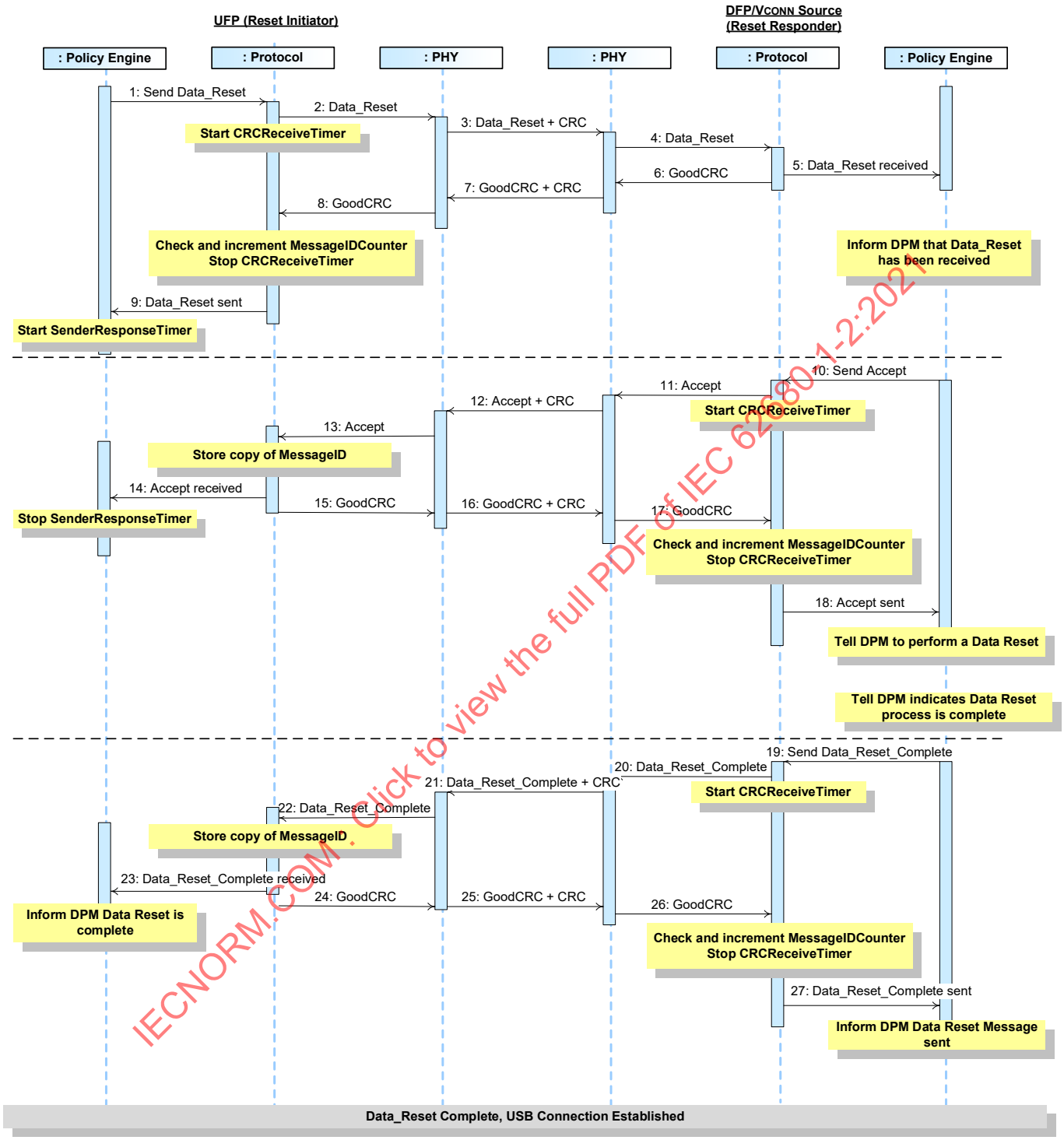
Etape	DFP/Source VCONN (initiateur de réinitialisation)	UFP (répondeur de réinitialisation)
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Data_Reset*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande à la couche protocole de former un message ***Accept***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message.
13	La couche protocole stocke le ***MessageID*** du message entrant.	
14	La couche protocole transfère les informations du message ***Accept*** reçu au moteur de politique qui le consomme. Le moteur de politique arrête le ***SenderResponseTimer*** et indique au gestionnaire de politique d'utilisation des dispositifs d'effectuer une réinitialisation des données. Le gestionnaire de politique d'utilisation des dispositifs procède à un cycle VCONN avant de réinitialiser la connexion de données.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Accept*** a abouti.
19	Le gestionnaire de politique d'utilisation des dispositifs indique que le processus de réinitialisation des données est terminé. Le moteur de politique demande à la couche protocole de générer un message ***Data_Reset_Complete***.	
20	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
21	La couche physique ajoute un CRC et envoie le message ***Data_Reset_Complete***.	La couche physique reçoit le message ***Data_Reset_Complete*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
22		La couche physique supprime le CRC et transfère le message ***Data_Reset_Complete*** vers la couche protocole.

Etape	DFP/Source VCONN (initiateur de réinitialisation)	UFP (répondeur de réinitialisation)
23		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Data_Reset_Complete*** reçu au moteur de politique qui le consomme. Le moteur de politique informe le gestionnaire de politique d'utilisation des dispositifs que la réception d'un message ***Data_Reset_Complete*** a abouti.
24		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
25	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
26	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
27	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Data_Reset_Complete*** a abouti. Le moteur de politique informe le gestionnaire de politique d'utilisation des dispositifs que l'envoi du message ***Data_Reset_Complete*** a abouti.	
La réinitialisation des données est terminée, comme défini en 6.3.14, Etape 5. Les ports partenaires rétablissent une connexion de données USB.		

8.3.2.4.2 Réception d'une réinitialisation des données par le DFP, où le DFP est la source VCONN

Il s'agit d'un exemple d'opération de réinitialisation des données où le DFP reçoit un message de réinitialisation des données et est la source VCONN. La Figure 8-10 représente les messages au fur et à mesure de leur circulation à travers le bus et dans les dispositifs afin de procéder à la réinitialisation des données.

Figure 8-10 Réception d'une réinitialisation par le DFP, où le DFP est la source Vconn



Anglais	Français
UFP (Reset Initiator)	UFP (Initiateur de réinitialisation)
DFP/Vconn Source (Reset Responder)	DFP/Source Vconn (Répondeur de réinitialisation)
Policy Engine	Moteur de politique
Protocol	Protocole

Send Data_Reset	Envoyer Data_Reset
Data_Reset received	Data_Reset reçu
Data_Reset sent	Data_Reset envoyé
Start CRCReceiveTimer	Lancer le CRCReceiveTimer
Check and increment MessageIDCounter	Vérifier et incrémenter le MessageIDCounter
Stop CRCReceiveTimer	Arrêter le CRCReceiveTimer
Inform DPM that Data_Reset has been received	Informer le DPM de la reception de Data_Reset
Start SenderResponseTimer	Lancer le SenderResponseTimer
Send Accept	Envoyer Accept
Accept received	Accept reçu
Accept sent	Accept envoyé
Store Copy of Message ID	Stocker une copie du Message ID
Stop SenderResponseTimer	Arrêter le SenderResponseTimer
Tell DPM to perform a Data Reset	Demander au DPM de procéder à une réinitialisation des données
Tell DPM indicates that Data Reset process is complete	Demander au DPM d'indiquer que le processus de réinitialisation des données est terminé
Send Data_Reset_Complete	Envoyer Data_Reset_Complete
Data_Reset_Complete received	Data_Reset_Complete reçu
Data_Reset_Complete sent	Data_Reset_Complete envoyé
Inform DPM Data Reset is complete	Informer le DPM de l'achèvement de la réinitialisation des données
Inform DPM Data Reset Message sent	Informer le DPM de l'envoi du message de réinitialisation des données
Data_Reset Complete, USB Connection Established	Data_Reset terminée, connexion USB établie

La Figure 8-10 ci-dessous donne une explication précise de ce qu'il se passe à chaque étape indiquée sur la Figure 8-10 ci-dessus.

Tableau 8-10 Etapes de la réception d'une réinitialisation des données par le DFP, où le DFP est la source Vconn

Etape	UFP (initiateur de réinitialisation)	DFP/Source VCONN (répondeur de réinitialisation)
1	Le moteur de politique demande à la couche protocole de générer un message ***Data_Reset*** pour demander une réinitialisation des données.	
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Data_Reset***.	La couche physique reçoit le message ***Data_Reset*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Data_Reset*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Data_Reset*** reçu au moteur de politique qui le consomme. Le moteur de politique informe le gestionnaire de politique d'utilisation des dispositifs que la réception d'un message ***Data_Reset*** a abouti.

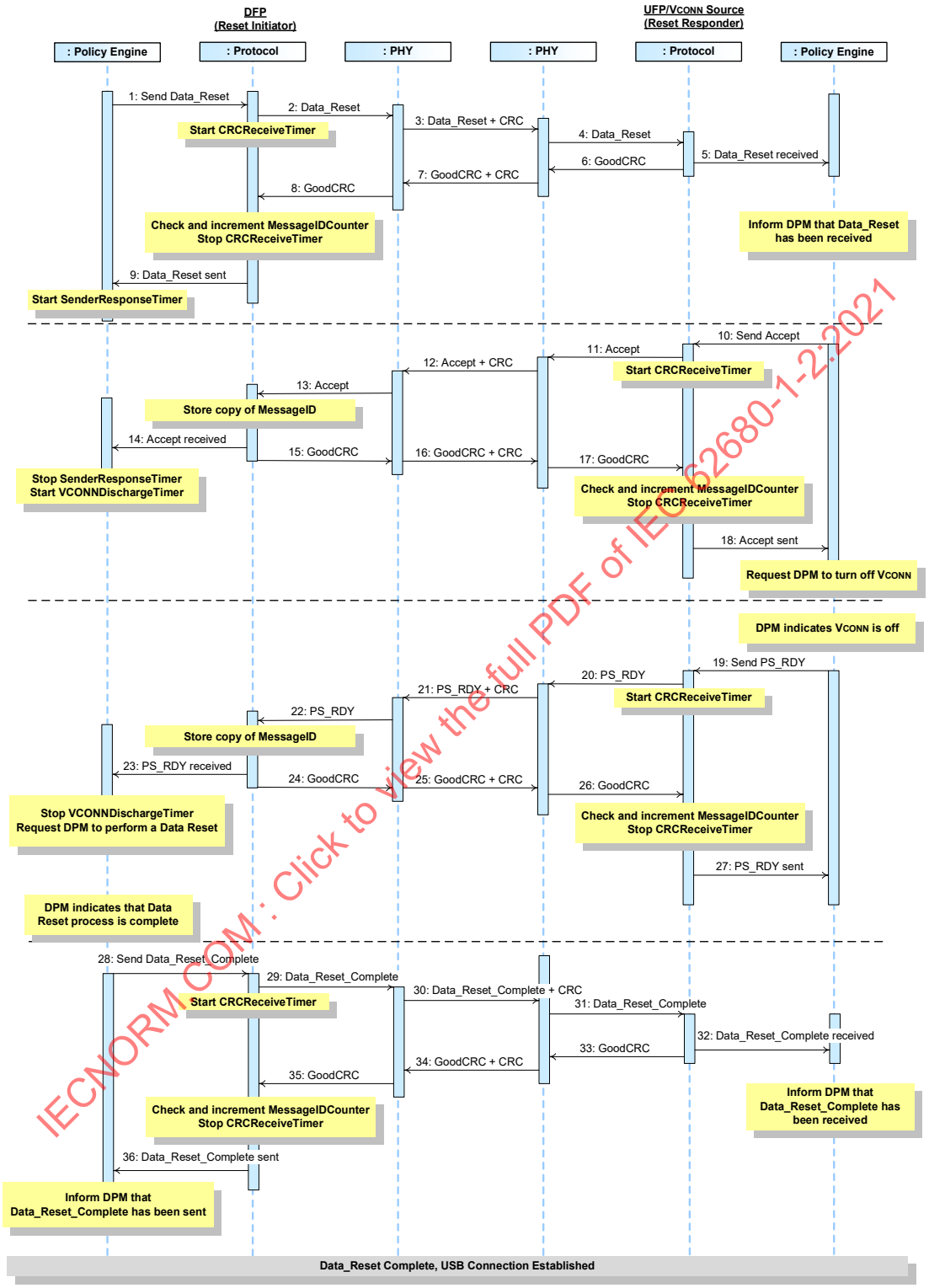
Etape	UFP (initiateur de réinitialisation)	DFP/Source VCONN (répondeur de réinitialisation)
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Data_Reset*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande à la couche protocole de former un message ***Accept***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message.
13	La couche protocole stocke le ***MessageID*** du message entrant.	
14	La couche protocole transfère les informations du message ***Accept*** reçu au moteur de politique qui le consomme. Le moteur de politique arrête le ***SenderResponseTimer***. Le gestionnaire de politique d'utilisation des dispositifs procède à un cycle VCONN avant de réinitialiser la connexion de données.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Accept*** a abouti. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'effectuer une réinitialisation des données.
19		Le gestionnaire de politique d'utilisation des dispositifs indique que le processus de réinitialisation des données est terminé. Le moteur de politique demande à la couche protocole de générer un message ***Data_Reset_Complete***.
20		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
21	La couche physique reçoit le message ***Data_Reset_Complete*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***Data_Reset_Complete***.
22	La couche physique supprime le CRC et transfère le message ***Data_Reset_Complete*** vers la couche protocole.	

Etape	UFP (initiateur de réinitialisation)	DFP/Source VCONN (répondeur de réinitialisation)
23	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Data_Reset_Complete*** reçu au moteur de politique qui le consomme. Le moteur de politique informe le gestionnaire de politique d'utilisation des dispositifs que la réception d'un message ***Data_Reset_Complete*** a abouti.	
24	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
25	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.
26		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
27		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Data_Reset_Complete*** a abouti. Le moteur de politique informe le gestionnaire de politique d'utilisation des dispositifs que l'envoi du message ***Data_Reset_Complete*** a abouti.
	La réinitialisation est terminée, comme défini en 6.3.14, étape 5. Les ports partenaires rétablissent une connexion de données USB.	

8.3.2.4.3 Réinitialisation des données initiée par le DFP, où l'UFP est la source VCONN

Il s'agit d'un exemple d'opération de réinitialisation des données où le DFP initie une réinitialisation des données et où l'UFP est la source VCONN. La Figure 8-11 représente les messages au fur et à mesure de leur circulation à travers le bus et dans les dispositifs afin de procéder à la réinitialisation des données.

Figure 8-11 Réinitialisation des données initiée par le DFP, où l'UFP est la source Vconn



Anglais	Français
DFP (Reset Initiator)	DFP (Initiateur de réinitialisation)
UFP/Vconn Source (Reset Responder)	UFP/Source Vconn (Répondeur de réinitialisation)
Policy Engine	Moteur de politique

Protocol	Protocole
Send Data_Reset	Envoyer Data_Reset
Data_Reset received	Data_Reset reçu
Data_Reset sent	Data_Reset envoyé
Start CRCReceiveTimer	Lancer le CRCReceiveTimer
Check and increment MessageIDCounter	Vérifier et incrémenter le MessageIDCounter
Stop CRCReceiveTimer	Arrêter le CRCReceiveTimer
Inform DPM that Data_Reset has been received	Informer le DPM de la reception de Data_Reset
Start SenderResponseTimer	Lancer le SenderResponseTimer
Send Accept	Envoyer Accept
Accept received	Accept reçu
Accept sent	Accept envoyé
Store Copy of Message ID	Stocker une copie du Message ID
Stop SenderResponseTimer	Arrêter le SenderResponseTimer
Start VconnDischargeTimer	Lancer le VconnDischargeTimer
Request DPM to turn off Vconn	Demander au DPM de désactiver Vconn
DPM indicates Vconn is off	Le DPM indique que Vconn est désactivée
Stop VconnDischargeTimer	Arrêter le VconnDischargeTimer
Request DPM to perform a Data Reset	Demander au DPM de procéder à une réinitialisation des données
DPM indicates that Data Reset process is complete	Le DPM indique que le processus de réinitialisation des données est terminé
Send Data_Reset_Complete	Envoyer Data_Reset_Complete
Data_Reset_Complete received	Data_Reset_Complete reçu
Data_Reset_Complete sent	Data_Reset_Complete envoyé
Inform DPM that Data_Reset_Complete has been received	Informer le DPM de la réception de Data_Reset_Complete
Inform DPM that Data_Reset_Complete has been sent	Informer le DPM de l'envoi de Data_Reset_Complete
Data_Reset Complete, USB Connection Established	Data_Reset terminée, connexion USB établie

Le Tableau 8-11 Etapes d'une réinitialisation des données initiée par le DFP, où l'UFP est la source Vconn ci-dessous donne une explication précise de ce qu'il se passe à chaque étape indiquée sur la Figure 8-11 ci-dessus.

Tableau 8-11 Etapes d'une réinitialisation des données initiée par le DFP, où l'UFP est la source Vconn

Etape	DFP (initiateur de réinitialisation)	UFP/Source VCONN (répondeur de réinitialisation)
1	Le moteur de politique demande à la couche protocole de générer un message ***Data_Reset*** pour demander une réinitialisation logicielle.	
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Data_Reset***.	La couche physique reçoit le message ***Data_Reset*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Data_Reset*** vers la couche protocole.

Etape	DFP (initiateur de réinitialisation)	UFP/Source VCONN (répondeur de réinitialisation)
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Data_Reset*** reçu au moteur de politique qui le consomme. Le moteur de politique informe le gestionnaire de politique d'utilisation des dispositifs que la réception d'un message ***Data_Reset*** a abouti.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Data_Reset*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande à la couche protocole de former un message ***Accept***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message.
13	La couche protocole stocke le ***MessageID*** du message entrant.	
14	La couche protocole transfère les informations du message ***Accept*** reçu au moteur de politique qui le consomme. Le moteur de politique arrête le ***SenderResponseTimer*** et lance le ***VCONNDischargeTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Accept*** a abouti. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs de désactiver VCONN.
19		Lorsque le gestionnaire de politique d'utilisation des dispositifs indique que VCONN a été désactivée, le moteur de politique demande à la couche protocole de former un message ***PS_RDY***.
20		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
21	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message.

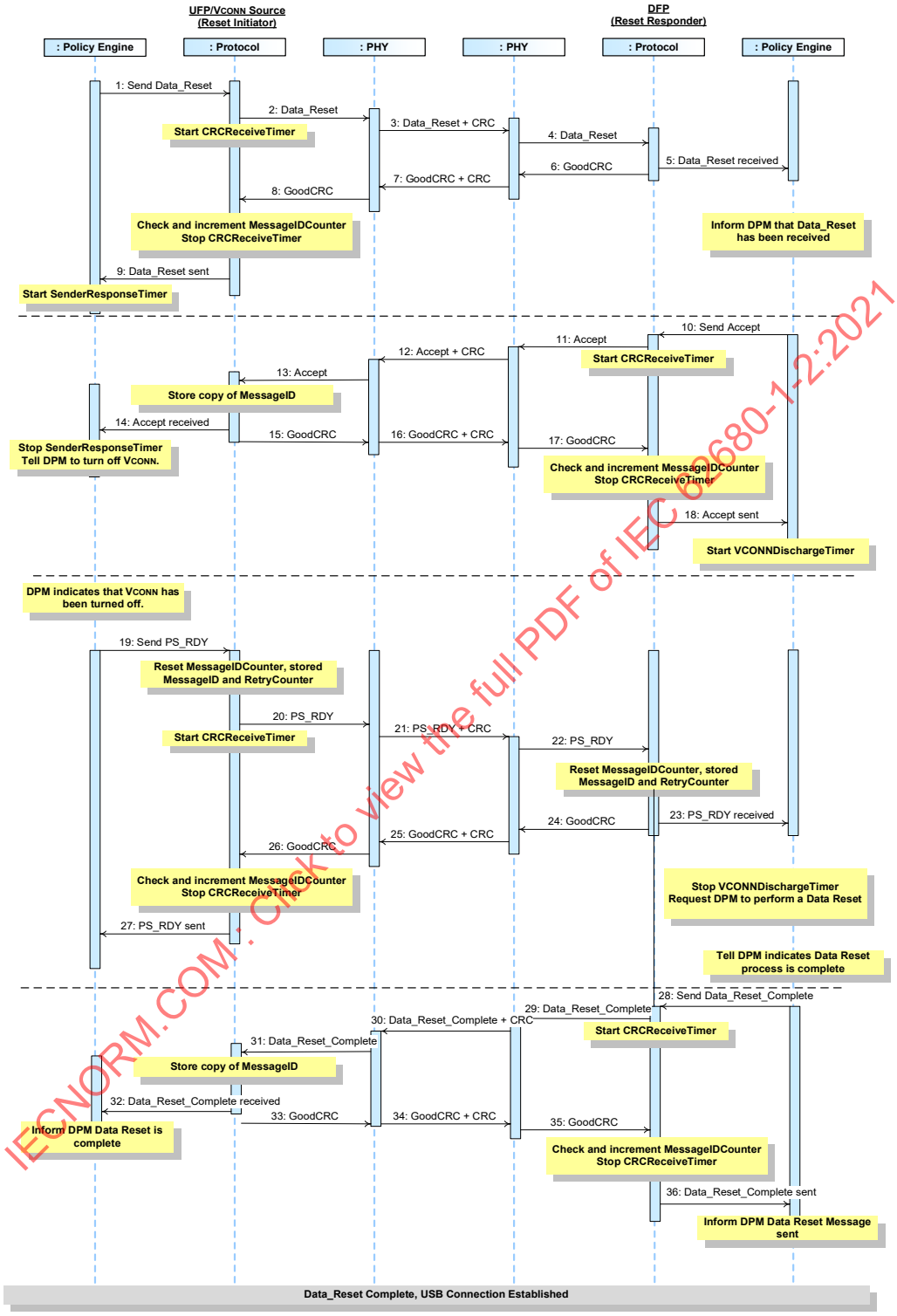
Etape	DFP (initiateur de réinitialisation)	UFP/Source VCONN (répondeur de réinitialisation)
22	La couche protocole stocke le ***MessageID*** du message entrant.	
23	La couche protocole transfère les informations du message ***PS_RDY*** reçu au moteur de politique qui le consomme. Le moteur de politique arrête le ***VCONNDischargeTimer*** et indique au gestionnaire de politique d'utilisation des dispositifs d'effectuer une réinitialisation des données. Le gestionnaire de politique d'utilisation des dispositifs active VCONN avant de réinitialiser la connexion de données.	
24	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
25	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
26		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
27		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***PS_RDY*** a abouti.
28	Le gestionnaire de politique d'utilisation des dispositifs indique que le processus de réinitialisation des données est terminé. Le moteur de politique demande à la couche protocole de générer un message ***Data_Reset_Complete***.	
29	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
30	La couche physique ajoute un CRC et envoie le message ***Data_Reset_Complete***.	La couche physique reçoit le message ***Data_Reset_Complete*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
31		La couche physique supprime le CRC et transfère le message ***Data_Reset_Complete*** vers la couche protocole.
32		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Data_Reset_Complete*** reçu au moteur de politique qui le consomme. Le moteur de politique informe le gestionnaire de politique d'utilisation des dispositifs que la réception d'un message ***Data_Reset_Complete*** a abouti.
33		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
34	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
35	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
36	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Data_Reset_Complete*** a abouti. Le moteur de politique informe le gestionnaire de politique d'utilisation des dispositifs que l'envoi du message ***Data_Reset_Complete*** a abouti.	

Etape	DFP (initiateur de réinitialisation)	UFP/Source VCONN (répondeur de réinitialisation)
La réinitialisation est terminée, comme défini en 6.3.14, Etape 5. Les ports partenaires rétablissent une connexion de données USB.		

8.3.2.4.4 Réception d'une réinitialisation des données par le DFP, où l'UFP est la source VCONN

Il s'agit d'un exemple d'opération de réinitialisation des données où le DFP reçoit un message de réinitialisation des données et où l'UFP est la source VCONN. La Figure 8-12 représente les messages au fur et à mesure de leur circulation à travers le bus et dans les dispositifs afin de procéder à la réinitialisation des données.

Figure 8-12 Réception d'une réinitialisation par le DFP, où l'UFP est la source Vconn



Anglais	Français
UFP/Vconn Source (Reset Initiator)	UFP/Source Vconn (Initiateur de réinitialisation)
DFP (Reset Responder)	DFP (Répondeur de réinitialisation)
Policy Engine	Moteur de politique

Protocol	Protocole
Send Data_Reset	Envoyer Data_Reset
Data_Reset received	Data_Reset reçu
Data_Reset sent	Data_Reset envoyé
Start CRCReceiveTimer	Lancer le CRCReceiveTimer
Check and increment MessageIDCounter	Vérifier et incrémenter le MessageIDCounter
Stop CRCReceiveTimer	Arrêter le CRCReceiveTimer
Inform DPM that Data_Reset has been received	Informer le DPM de la reception de Data_Reset
Start SenderResponseTimer	Lancer le SenderResponseTimer
Send Accept	Envoyer Accept
Accept received	Accept reçu
Accept sent	Accept envoyé
Store Copy of Message ID	Stocker une copie du Message ID
Stop SenderResponseTimer	Arrêter le SenderResponseTimer
Tell DPM to turn off Vconn	Demander au DPM de désactiver Vconn
Start VconnDischargeTimer	Lancer le VconnDischargeTimer
DPM indicates that Vconn has been off	Le DPM indique que Vconn a été désactivée
Reset MessageIDCounter, sored MessageID and RetryCounter	Réinitialiser le MessageIDCounter, le MessageID envoyé et le RetryCounter
Stop VconnDischargeTimer	Arrêter le VconnDischargeTimer
Request DPM to perform a Data Reset	Demander au DPM de procéder à une réinitialisation des données
Tell DPM indicates that Data Reset process is complete	Demander au DPM d'indiquer que le processus de réinitialisation des données est terminé
Send Data_Reset_Complete	Envoyer Data_Reset_Complete
Data_Reset_Complete received	Data_Reset_Complete reçu
Data_Reset_Complete sent	Data_Reset_Complete envoyé
Inform DPM Data Reset is complete	Informer le DPM de l'achèvement de la réinitialisation des données
Inform DPM Data Reset Message sent	Informer le DPM de l'envoi du message de réinitialisation des données
Data_Reset Complete, USB Connection Established	Data_Reset terminée, connexion USB établie

Le Tableau 8-12 ci-dessous donne une explication précise de ce qu'il se passe à chaque étape indiquée sur la Figure 8-12 ci-dessus.

Tableau 8-12 Etapes de la réception d'une réinitialisation des données par le DFP, où l'UFP est la source Vconn

Etape	UFP/Source VCONN (initiateur de réinitialisation)	DFP (répondeur de réinitialisation)
1	Le moteur de politique demande à la couche protocole de générer un message ***Data_Reset*** pour demander une réinitialisation logicielle.	
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Data_Reset***.	La couche physique reçoit le message ***Data_Reset*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Data_Reset*** vers la couche protocole.

Etape	UFP/Source VCONN (initiateur de réinitialisation)	DFP (répondeur de réinitialisation)
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Data_Reset*** reçu au moteur de politique qui le consomme. Le moteur de politique informe le gestionnaire de politique d'utilisation des dispositifs que la réception d'un message ***Data_Reset*** a abouti.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Data_Reset*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande à la couche protocole de former un message ***Accept***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message.
13	La couche protocole stocke le ***MessageID*** du message entrant.	
14	La couche protocole transfère les informations du message ***Accept*** reçu au moteur de politique qui le consomme. Le moteur de politique arrête le SenderResponseTimer et demande au gestionnaire de politique d'utilisation des dispositifs de désactiver VCONN.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Accept*** a abouti. Le moteur de politique lance le VCONNDischargeTimer.
19	Lorsque le gestionnaire de politique d'utilisation des dispositifs indique que VCONN a été désactivée, le moteur de politique demande à la couche protocole de générer un message ***PS_RDY*** pour demander une réinitialisation logicielle.	
20	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	

Etape	UFP/Source VCONN (initiateur de réinitialisation)	DFP (répondeur de réinitialisation)
21	La couche physique ajoute un CRC et envoie le message ***PS_RDY***.	La couche physique reçoit le message ***PS_RDY*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
22		La couche physique supprime le CRC et transfère le message ***PS_RDY*** vers la couche protocole.
23		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***PS_RDY*** reçu au moteur de politique qui le consomme. Le moteur de politique arrête le VCONNDischargeTimer et demande au gestionnaire de politique d'utilisation des dispositifs d'effectuer une réinitialisation des données. Le gestionnaire de politique d'utilisation des dispositifs active VCONN avant de réinitialiser la connexion de données.
24		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
25	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
26	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
27	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***PS_RDY*** a abouti.	
28		Le gestionnaire de politique d'utilisation des dispositifs indique que le processus de réinitialisation des données est terminé. Le moteur de politique demande à la couche protocole de générer un message ***Data_Reset_Complete***.
29		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
30	La couche physique reçoit le message ***Data_Reset_Complete*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***Data_Reset_Complete***.
31	La couche physique supprime le CRC et transfère le message ***Data_Reset_Complete*** vers la couche protocole.	
32	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Data_Reset_Complete*** reçu au moteur de politique qui le consomme. Le moteur de politique informe le gestionnaire de politique d'utilisation des dispositifs que la réception d'un message ***Data_Reset_Complete*** a abouti.	
33	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
34	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.

Etape	UFP/Source VCONN (initiateur de réinitialisation)	DFP (répondeur de réinitialisation)
35		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
36		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Data_Reset_Complete*** a abouti. Le moteur de politique informe le gestionnaire de politique d'utilisation des dispositifs que l'envoi du message ***Data_Reset_Complete*** a abouti.
La réinitialisation est terminée, comme défini en 6.3.14, Etape 5. Les ports partenaires rétablissent une connexion de données USB.		

8.3.2.5 Réinitialisation matérielle

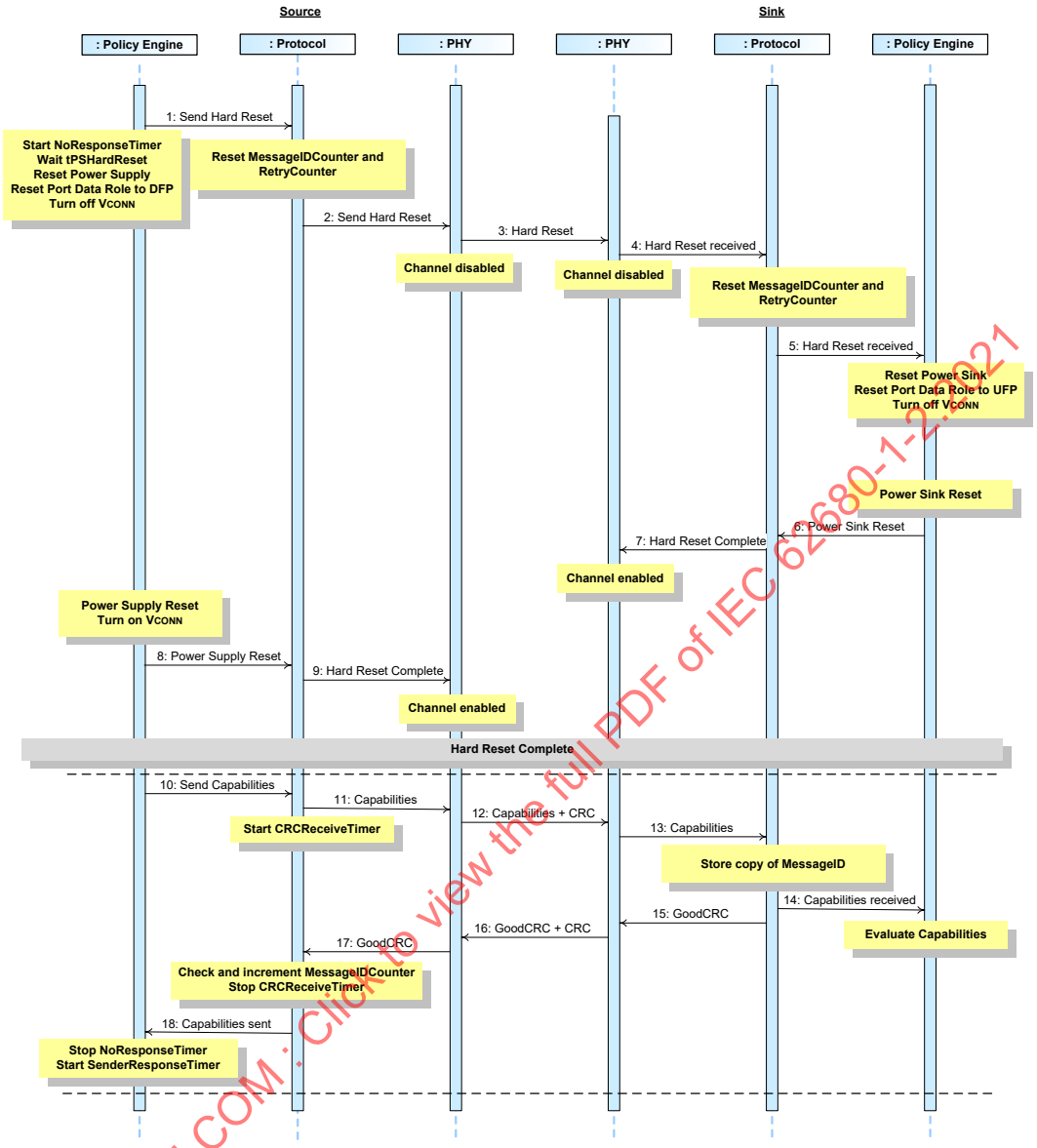
Les sections suivantes décrivent la procédure de réinitialisation matérielle de l'alimentation électrique par port USB. La réinitialisation matérielle rétablit le rôle par défaut et la tension/le courant de fonctionnement de l'alimentation électrique par port USB. Lors de la réinitialisation logicielle, les communications de couche PHY de l'alimentation par port USB ***Devoir/Doit/Doivent*** être désactivées en empêchant la communication entre les ports partenaires.

Note: Dans ce cas, la réinitialisation matérielle est appliquée à la capacité d'alimentation électrique par port USB d'un port individuel sur lequel la réinitialisation matérielle est demandée. Un effet secondaire de la réinitialisation matérielle est qu'elle peut réinitialiser d'autres fonctions sur le port (USB, par exemple).

8.3.2.5.1 Réinitialisation matérielle déclenchée par la source

Voici un exemple d'opération de réinitialisation matérielle déclenchée par une source. La Figure 8-13 représente les messages au fur et à mesure de leur circulation à travers le bus et dans les dispositifs afin de procéder à la réinitialisation matérielle.

Figure 8-13 Réinitialisation matérielle déclenchée par la source



Anglais	Français
Check and increment MessageIDCounter	Vérifier et incrémenter le MessageIDCounter
PHY	PHY
Policy Engine	Moteur de politique
Protocol	Protocole
Reset Power Supply	Réinitialiser l'alimentation
Reset Power Sink	Réinitialiser le destinataire de l'alimentation
Reset MessageIDCounter and RetryCounter	Réinitialiser le MessageIDCounter et le RetryCounter
Reset Port Data Role to DFP	Réinitialiser le rôle de transmission de données à DFP
Reset Port Data Role to UFP	Réinitialiser le rôle de transmission de données à UFP
Turn off V_{CONN}	Désactiver V_{CONN}
Send Capabilities	Envoyer les capacités

Capabilities	Capacités
Capabilities received	Capacités reçues
Capabilities sent	Capacités envoyées
Evaluate Capabilities	Evaluer les capacités
Stop NoResponseTimer	Arrêter le NoResponseTimer
Start NoResponseTimer	Lancer le NoResponseTimer
Stop CRCReceiveTimer	Arrêter le CRCReceiveTimer
Store Copy of Message ID	Stocker une copie du Message ID
Start CRCReceiveTimer	Lancer le CRCReceiveTimer
Start SenderResponseTimer	Lancer le SenderResponseTimer
Source	Source
Sink	Destinataire
Send Hard Reset	Envoyer Hard Reset
Hard Reset	Réinitialisation matérielle
Hard Reset Received	Hard Reset reçu
Hard Reset Complete	Réinitialisation matérielle terminée
Power Sink Reset	Réinitialisation du destinataire de l'alimentation
Wait tPSHardReset	Attendre tPSHardReset
Channel Disabbled	Canal désactivé
Channel Enabled	Canal activé
Power Supply Reset	Réinitialisation de l'alimentation

Tableau 8-13 Etapes de la réinitialisation matérielle déclenchée par la source

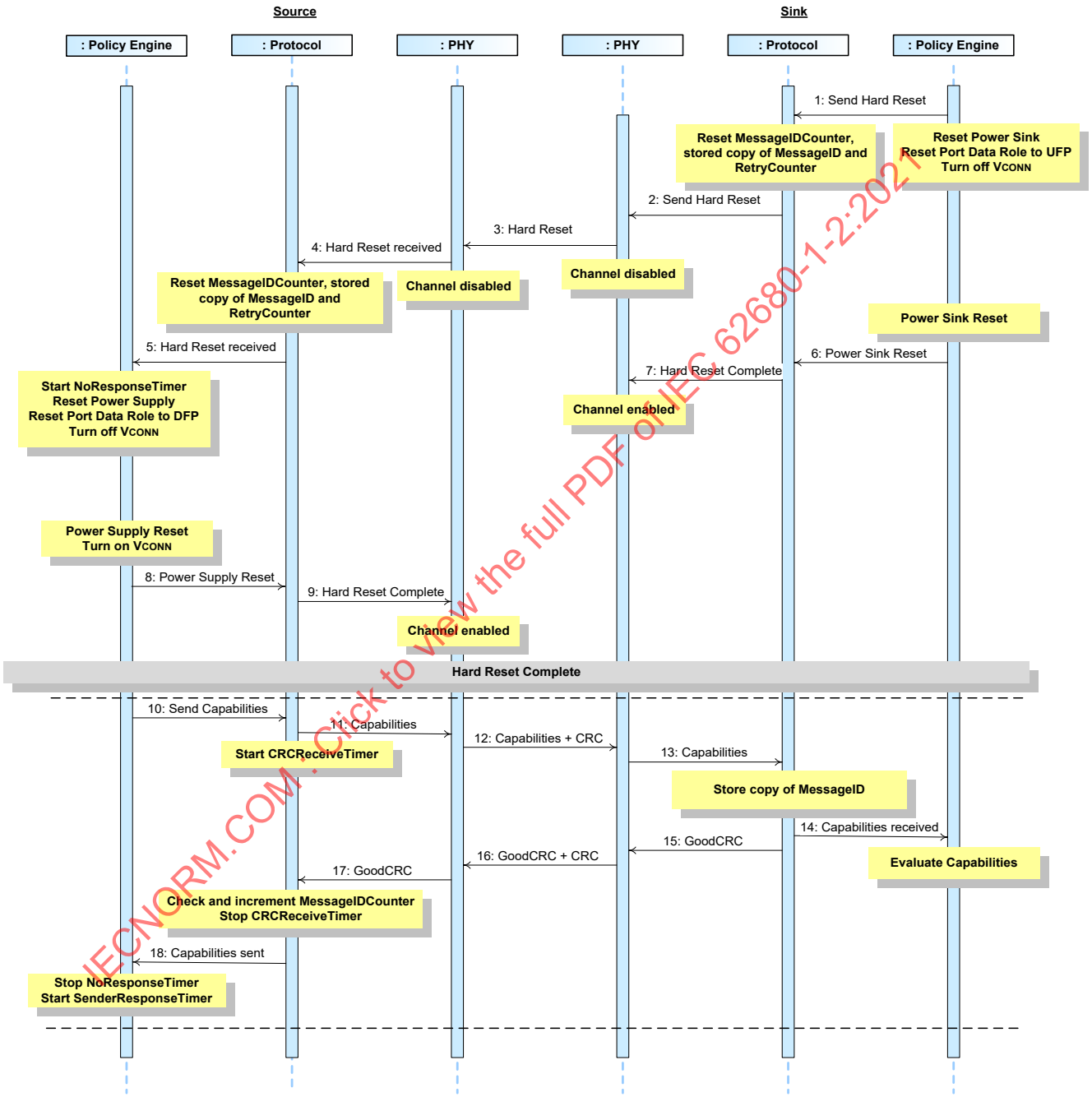
Etape	Source	Destinataire
1	Le moteur de politique demande à la couche protocole de générer un signal ***Hard Reset***. Le moteur de politique lance le ***NoResponseTimer*** et demande au gestionnaire de politique d'utilisation des dispositifs de réinitialiser l'alimentation sur le fonctionnement USB par défaut. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs de réinitialiser le rôle de transmission de données du port sur DFP et de désactiver VCONN s'il est activé.	
2	La couche protocole réinitialise le ***MessageIDCounter*** et le ***RetryCounter.*** La couche protocole demande à la couche physique d'envoyer un signal ***Hard Reset***.	
3	La couche physique envoie le signal ***Hard Reset***, puis désactive le canal de communication de la couche PHY pour la transmission et la réception.	La couche physique reçoit le signal ***Hard Reset***, puis désactive le canal de communication de la couche PHY pour la transmission et la réception.
4		La couche physique informe la couche protocole de la réinitialisation matérielle. La couche protocole réinitialise le ***MessageIDCounter*** et le ***RetryCounter***.
5		La couche protocole informe le moteur de politique de la réinitialisation matérielle. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs de réinitialiser le destinataire de l'alimentation sur le fonctionnement par défaut. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs de réinitialiser le rôle de transmission de données du port sur UFP et de désactiver VCONN s'il est activé.
6		Le destinataire de l'alimentation retourne au fonctionnement par défaut. Le moteur de politique informe la couche protocole que le destinataire de l'alimentation a été réinitialisé.
7		La couche protocole informe la couche PHY que la réinitialisation matérielle est terminée. La couche PHY permet au canal de communication de la couche PHY de transmettre et de recevoir.
8	L'alimentation est réinitialisée aux valeurs par défaut et VCONN est activée. Le moteur de politique informe la couche protocole que l'alimentation a été réinitialisée.	
9	La couche protocole informe la couche PHY que la réinitialisation matérielle est terminée. La couche PHY permet au canal de communication de la couche PHY de transmettre et de recevoir.	
	La réinitialisation est terminée et la communication de protocole peut redémarrer.	
10	Le moteur de politique demande à la couche protocole d'envoyer un message ***Source_Capabilities*** qui donne les capacités présentes de l'alimentation.	
11	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
12	La couche physique ajoute un CRC et envoie le message ***Source_Capabilities***.	La couche physique reçoit le message ***Source_Capabilities*** et contrôle le CRC pour vérifier le message.

Etape	Source	Destinataire
13		La couche physique supprime le CRC et transfère le message ***Source_Capabilities*** vers la couche protocole.
14		La couche protocole stocke le ***MessageID*** du message entrant. La couche protocole transfère les informations du message ***Source_Capabilities*** reçu au moteur de politique qui le consomme.
15		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
16	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
17	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
18	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Source_Capabilities*** a abouti. Le moteur de politique arrête le ***NoResponseTimer*** et lance le ***SenderResponseTimer***.	
	La communication d'alimentation USB est rétablie.	

8.3.2.5.2 Réinitialisation matérielle déclenchée par le destinataire

Voici un exemple d'opération de réinitialisation matérielle déclenchée par un destinataire. La Figure 8-14 représente les messages au fur et à mesure de leur circulation à travers le bus et dans les dispositifs afin de procéder à la réinitialisation matérielle.

Figure 8-14 Réinitialisation matérielle déclenchée par le destinataire



Anglais	Français
Capabilities	Capacités
Capabilities received	Capacités reçues
Capabilities sent	Capacités envoyées
Channel Disabbled	Canal désactivé

Channel Enabled	Canal activé
Check and increment MessageIDCounter	Vérifier et incrémenter le MessageIDCounter
Evaluate Capabilities	Evaluer les capacités
Hard Reset	Réinitialisation matérielle
Hard Reset Complete	Réinitialisation matérielle terminée
Hard Reset Received	Hard Reset reçu
PHY	PHY
Policy Engine	Moteur de politique
Power Sink Reset	Réinitialisation du destinataire de l'alimentation
Power Supply Reset	Réinitialisation de l'alimentation
Protocol	Protocole
Reset MessageIDCounter and RetryCounter	Réinitialiser le MessageIDCounter et le RetryCounter
Reset Port Data Role to DFP	Réinitialiser le rôle de transmission de données à DFP
Reset Port Data Role to UFP	Réinitialiser le rôle de transmission de données à UFP
Reset Power Sink	Réinitialiser le destinataire de l'alimentation
Reset Power Supply	Réinitialiser l'alimentation
Send Capabilities	Envoyer les capacités
Send Hard Reset	Envoyer Hard Reset
Sink	Destinataire
Source	Source
Start CRCReceiveTimer	Lancer le CRCReceiveTimer
Start NoResponseTimer	Lancer le NoResponseTimer
Start SenderResponseTimer	Lancer le SenderResponseTimer
Stop CRCReceiveTimer	Arrêter le CRCReceiveTimer
Stop NoResponseTimer	Arrêter le NoResponseTimer
Store Copy of Message ID and RetryCounter	Stocker une copie du Message ID et du RetryCounter
Store Copy of Message ID	Stocker une copie du Message ID
Turn off V_{CONN}	Désactiver V_{CONN}

Tableau 8-14 Etapes de la réinitialisation matérielle déclenchée par le destinataire

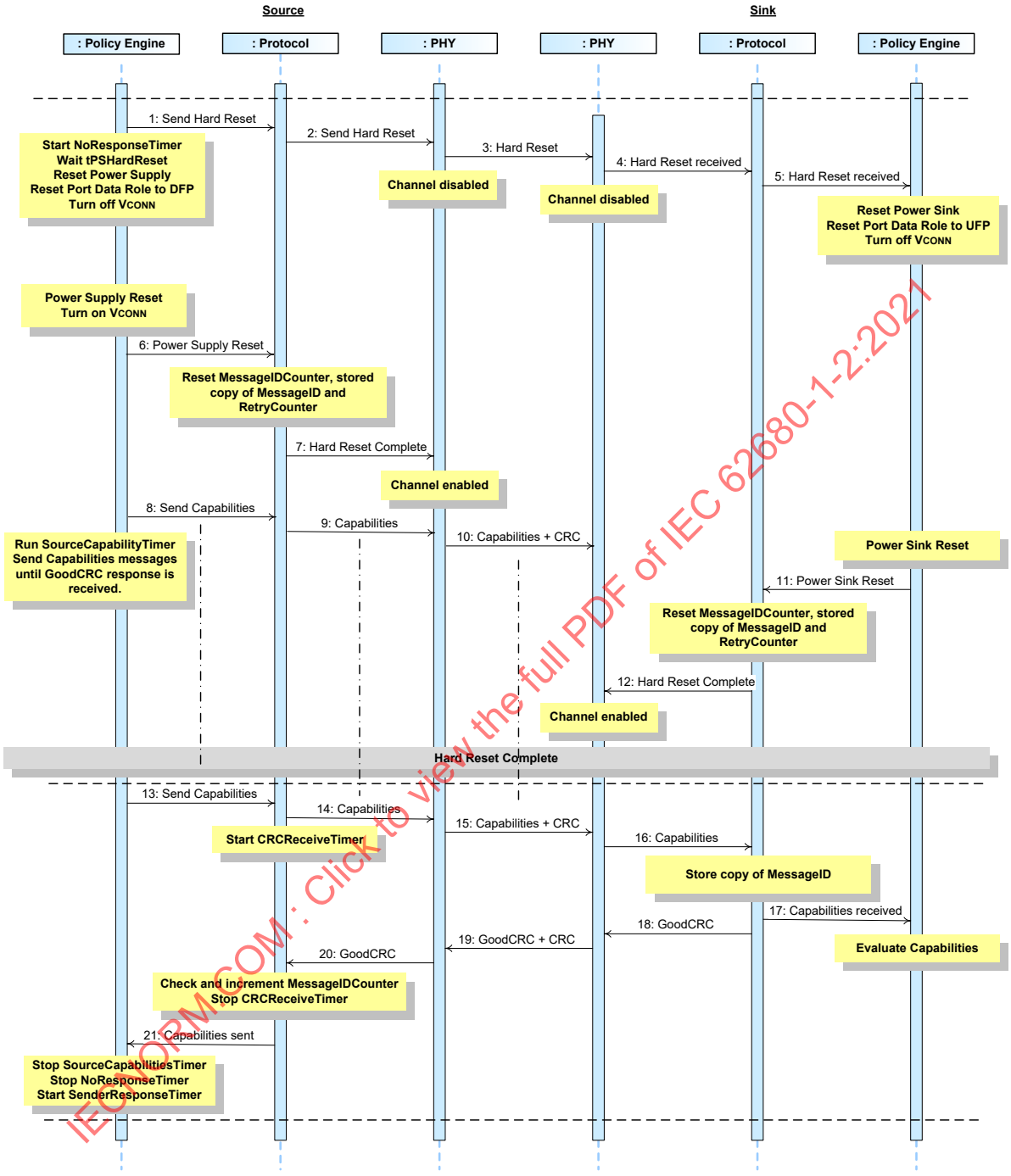
Etape	Source	Destinataire
1		Le moteur de politique demande à la couche protocole de générer un signal ***Hard Reset***. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs de réinitialiser l'alimentation sur le fonctionnement USB par défaut. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs de réinitialiser le rôle de transmission de données du port sur UFP et de désactiver VCONN s'il est activé.
2		La couche protocole réinitialise le ***MessageIDCounter***, la copie stockée du ***MessageID*** et le ***RetryCounter***. La couche protocole demande à la couche physique d'envoyer un signal ***Hard Reset***.
3	La couche physique reçoit le signal ***Hard Reset***, puis désactive le canal de communication de la couche PHY pour la transmission et la réception.	La couche physique envoie le signal ***Hard Reset***, puis désactive le canal de communication de la couche PHY pour la transmission et la réception.
4	La couche physique informe la couche protocole de la réinitialisation matérielle. La couche protocole réinitialise le ***MessageIDCounter***, la copie stockée du ***MessageID*** et le ***RetryCounter***.	
5	La couche protocole informe le moteur de politique de la réinitialisation matérielle. Le moteur de politique lance le ***NoResponseTimer*** et demande au gestionnaire de politique d'utilisation des dispositifs de réinitialiser le destinataire de l'alimentation sur le fonctionnement par défaut. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs de réinitialiser le rôle de transmission de données du port sur DFP et de désactiver VCONN s'il est activé.	
6		Le destinataire de l'alimentation retourne au fonctionnement USB par défaut. Le moteur de politique informe la couche protocole que le destinataire de l'alimentation a été réinitialisé.
7		La couche protocole informe la couche PHY que la réinitialisation matérielle est terminée. La couche PHY permet au canal de communication de la couche PHY de transmettre et de recevoir.
8	L'alimentation est réinitialisée au fonctionnement USB par défaut et VCONN est activée. Le moteur de politique informe la couche protocole que l'alimentation a été réinitialisée.	
9	La couche protocole informe la couche PHY que la réinitialisation matérielle est terminée. La couche PHY permet au canal de communication de la couche PHY de transmettre et de recevoir.	
	La réinitialisation est terminée et la communication de protocole peut redémarrer.	
10	Le moteur de politique demande à la couche protocole d'envoyer un message ***Source_Capabilities*** qui donne les capacités présentes de l'alimentation.	
11	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
12	La couche physique ajoute un CRC et envoie le message ***Source_Capabilities***.	La couche physique reçoit le message ***Source_Capabilities*** et contrôle le CRC pour vérifier le message.

Etape	Source	Destinataire
13		La couche physique supprime le CRC et transfère le message ***Source_Capabilities*** vers la couche protocole.
14		La couche protocole stocke le ***MessageID*** du message entrant. La couche protocole transfère les informations du message ***Source_Capabilities*** reçu au moteur de politique qui le consomme.
15		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
16	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
17	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
18	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Source_Capabilities*** a abouti. Le moteur de politique arrête le ***NoResponseTimer*** et lance le ***SenderResponseTimer***.	
	La communication d'alimentation USB est rétablie.	

8.3.2.5.3 Réinitialisation matérielle déclenchée par la source – Longue réinitialisation du destinataire

Voici un exemple d'opération de réinitialisation matérielle déclenchée par une source. Dans cet exemple, le destinataire est lent à répondre à la demande, la source envoyant plusieurs messages ***Source_Capabilities*** avant de recevoir un message ***GoodCRC*** en réponse. La Figure 8-15 représente les messages au fur et à mesure de leur circulation à travers le bus et dans les dispositifs afin de procéder à la réinitialisation matérielle.

Figure 8-15 Réinitialisation déclenchée par la source - Réinitialisation longue du destinataire



Anglais	Français
Source	Source
Sink	Destinataire
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Send Hard Reset	Envoyer Hard Reset

Hard Reset	Réinitialisation matérielle
Hard Reset received	Hard Reset reçu
Start	Début
Wait	Attendre
Reset Power Supply	Réinitialiser l'alimentation
Reset Port Data Role to DFP	Réinitialiser le rôle de transmission de données à DFP
Turn off V_{CONN}	Désactiver V_{CONN}
Channel disabled	Canal désactivé
Reset Power Sink	Réinitialiser le destinataire de l'alimentation
Reset Port Data Role to UFP	Réinitialiser le rôle de transmission de données à UFP
Power Supply Reset	Réinitialisation de l'alimentation
Turn on V_{CONN}	Désactiver V_{CONN}
Power Supply Reset	Réinitialisation de l'alimentation
Reset MessageIDCounter, stored copy of MessageID and RetryCounter	Réinitialiser le MessageIDCounter, stocker une copie du MessageID et du RetryCounter
Hard Reset Complete	Réinitialisation matérielle terminée
Channel enabled	Canal activé
Send Capabilities	Envoyer les capacités
Capabilities	Capacités
Capabilities + CRC	Capacités + CRC
Run	Exécution
Send Capabilities messages until GoodCRC response is received.	Envoyer des messages de capacités jusqu'à réception d'une réponse GoodCRC
Power Sink Reset	Réinitialisation du destinataire de l'alimentation
Store copy of MessageID	Stocker une copie du MessageID
Capabilities received	Capacités reçues
CRC	CRC
Evaluate Capabilities	Evaluer les capacités
Check and increment	Vérifier et incrémenter
Stop	Arrêt
Capabilities sent	Capacités envoyées

Tableau 8-15 Etapes de la réinitialisation matérielle déclenchée par la source – Réinitialisation longue du destinataire

Etape	Source	Destinataire
1	Le moteur de politique demande à la couche protocole de générer un signal ***Hard Reset***. Le moteur de politique lance le ***NoResponseTimer*** et demande au gestionnaire de politique d'utilisation des dispositifs de réinitialiser l'alimentation sur le fonctionnement USB par défaut. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs de réinitialiser le rôle de transmission de données du port sur DFP et de désactiver V_{CONN} s'il est activé.	

Etape	Source	Destinataire
2	La couche protocole réinitialise le ***MessageIDCounter***, la copie stockée du ***MessageID*** et le ***RetryCounter***. La couche protocole demande à la couche physique d'envoyer un signal ***Hard Reset***.	
3	La couche physique envoie le signal ***Hard Reset***, puis désactive le canal de communication de la couche PHY pour la transmission et la réception.	La couche physique reçoit le signal ***Hard Reset***, puis désactive le canal de communication de la couche PHY pour la transmission et la réception.
4		La couche physique informe la couche protocole de la réinitialisation matérielle. La couche protocole réinitialise le ***MessageIDCounter***, la copie stockée du ***MessageID*** et le ***RetryCounter***.
5		La couche protocole informe le moteur de politique de la réinitialisation matérielle. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs de réinitialiser le destinataire de l'alimentation sur le fonctionnement par défaut. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs de réinitialiser le rôle de transmission de données du port sur UFP et de désactiver VCONN s'il est activé.
6	L'alimentation est réinitialisée au fonctionnement USB par défaut et VCONN est activée. Le moteur de politique informe la couche protocole que l'alimentation a été réinitialisée.	
7	La couche protocole informe la couche PHY que la réinitialisation matérielle est terminée. La couche PHY permet au canal de communication de la couche PHY de transmettre et de recevoir.	
	La réinitialisation est terminée et la communication de protocole peut redémarrer.	
8	Le moteur de politique demande à la couche protocole d'envoyer un message ***Source_Capabilities*** qui donne les capacités présentes de l'alimentation. Le moteur de politique lance le ***SourceCapabilityTimer***. ***SourceCapabilityTimer*** atteint le délai d'attente une ou plusieurs fois tant qu'un message ***GoodCRC*** n'a pas été reçu en réponse.	
9	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
10	La couche physique ajoute un CRC et envoie le message ***Source_Capabilities***.	Note: Le message ***Source_Capabilities*** n'est pas reçu étant donné que le canal est désactivé.
11		Le destinataire de l'alimentation retourne au fonctionnement USB par défaut. Le moteur de politique informe la couche protocole que le destinataire de l'alimentation a été réinitialisé.
12		La couche protocole informe la couche PHY que la réinitialisation matérielle est terminée. La couche PHY permet au canal de communication de la couche PHY de transmettre et de recevoir.
	La réinitialisation est terminée et la communication de protocole peut redémarrer.	
13	Le moteur de politique demande à la couche protocole d'envoyer un message ***Source_Capabilities*** qui donne les capacités présentes de l'alimentation. Il lance le ***SourceCapabilityTimer***.	
14	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	

Etape	Source	Destinataire
15	La couche physique ajoute un CRC et envoie le message ***Source_Capabilities***.	La couche physique reçoit le message ***Source_Capabilities*** et contrôle le CRC pour vérifier le message.
16		La couche physique supprime le CRC et transfère le message ***Source_Capabilities*** vers la couche protocole.
17		La couche protocole stocke le ***MessageID*** du message entrant. La couche protocole transfère les informations du message ***Source_Capabilities*** reçu au moteur de politique qui le consomme.
18		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
19	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
20	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
21	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Source_Capabilities*** a abouti. Le moteur de politique arrête le ***SourceCapabilityTimer***, arrête le ***NoResponseTimer*** et lance le ***SenderResponseTimer***.	
	La communication d'alimentation USB est rétablie.	

8.3.2.6 Permutation des rôles d'alimentation

8.3.2.6.1 Permutation des rôles d'alimentation déclenchée par la source sans négociation de puissance subséquente

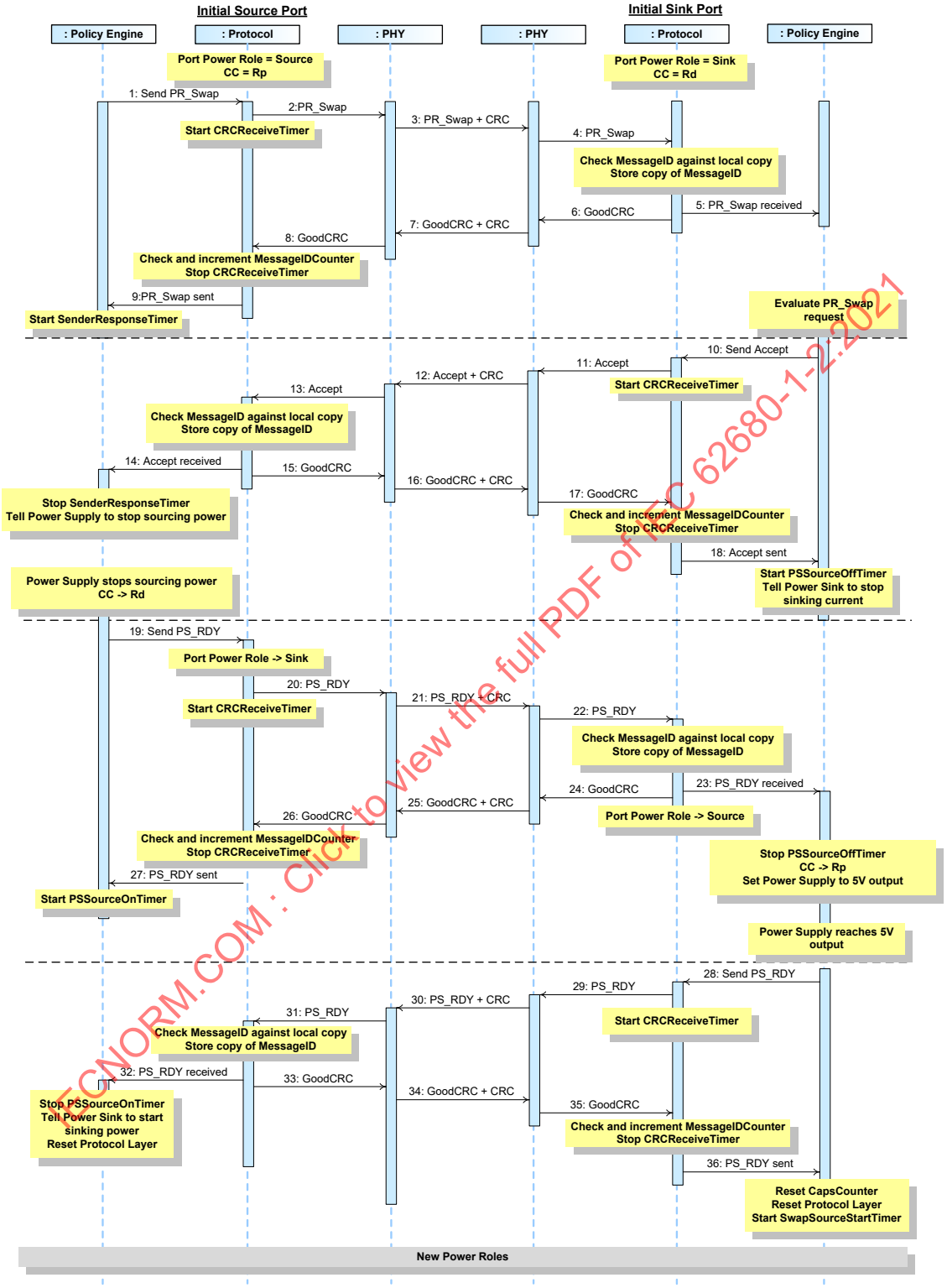
Il s'agit d'un exemple d'opération réussie de permutation des rôles d'alimentation initiée par un port qui, initialement, au début de cette séquence de messages, agit en tant que source, et qui a donc une polarisation Rp sur son fil CC. Il n'inclut pas de négociation de puissance subséquente exigée afin d'établir un contrat explicite (voir 8.3.2.2).

Quatre phases distinctes caractérisent la négociation de permutation des rôles d'alimentation:

1. un message ***PR_Swap*** est envoyé;
2. un message ***Accept*** est reçu en réponse au message ***PR_Swap***;
3. le nouveau destinataire définit sa puissance restituée sur ***vSafe0V***, puis affirme Rd et envoie un message ***PS_RDY*** à l'issue de ce processus;
4. la nouvelle source affirme Rp, puis définit sa puissance restituée sur ***vSafe5V*** et envoie un message ***PS_RDY*** lorsqu'elle est prête à assurer l'alimentation.

La Figure 8-16 représente les messages au fur et à mesure de leur circulation à travers le bus et dans les dispositifs afin de procéder à la séquence de permutation rapide des rôles.

Figure 8-16 Séquence réussie de permutation des rôles d'alimentation initiée par la source



Anglais	Français
Initial Source Port	Port source initial
Initial Sink Port	Port destinataire initial

Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Source	Rôle d'alimentation du port = source
CC = Rp	CC = Rp
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Send	Envoyer
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
PR Swap received	PR_Swap reçu
CRC	CRC
Check and increment	Vérifier et incrémenter
Stop	Arrêt
RP_Swap sent	RP_Swap envoyé
Evauate PR_Swap request	Evaluer la demande PR_Swap
Send Accept	Envoyer Accept
Accept	Accept
Accept + CRC	Accept + CRC
Accept received	Accept reçu
Tell Power Supply stops sourcing power	Ordonner à l'alimentation d'arrêter de fournir de la puissance
CC -> Rd	CC -> Rd
Tell Power Sink to stop sinking current	Ordonner au destinataire de l'alimentation d'arrêter de consommer du courant
Port Power Role -> Sink	Rôle d'alimentation du port -> destinataire
PS_RDY received	PS_RDY reçu
Port Power Role -> Source	Rôle d'alimentation du port -> source
PS_RDY sent	PS_RDY envoyé
CC -> Rp	CC -> Rp
Set Power Supply to 5V output	Définir l'alimentation sur 5V en sortie
Power Supply reaches 5V output	L'alimentation atteint 5V en sortie
Tell Power Sink to start sinking current	Ordonner au destinataire de l'alimentation de commencer à consommer du courant
Reset Protocol Layer	Réinitialiser la couche protocole
Reset	Réinitialisation
New Power Roles	Nouveaux rôles d'alimentation

Le Tableau 8-16 ci-dessous donne une explication précise de ce qu'il se passe à chaque étape indiquée sur la Figure 8-16 ci-dessus.

Tableau 8-16 Etapes d'une séquence réussie de permutation des rôles d'alimentation initiée par la source

Etape	Port source initial	Port destinataire initial
1	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***PR_Swap***.	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***PR_Swap***.	La couche physique reçoit le message ***PR_Swap*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***PR_Swap*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***PR_Swap*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***PR_Swap*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique évalue le message ***PR_Swap*** envoyé par la source et décide qu'il est en mesure de procéder à la permutation des rôles d'alimentation et qu'il le souhaite. Il demande à la couche protocole de former un message ***Accept***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Accept***.	La couche physique ajoute un CRC et envoie le message ***Accept***.
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***PR_Swap*** reçu au moteur de politique qui le consomme.	
14	Le moteur de politique demande à son alimentation de ne plus fournir la puissance, et arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.

Etape	Port source initial	Port destinataire initial
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Accept*** a abouti. Le moteur de politique lance le ***PSSourceOffTimer*** et demande à l'alimentation d'arrêter de consommer le courant.
19	Le moteur de politique détermine que son alimentation ne fournit plus V_{BUS}. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'affirmer la dépolarisation Rd sur le fil CC. Le moteur de politique ordonne ensuite à la couche protocole de générer un message ***PS_RDY***, le bit ***Port Power Role*** de l'en-tête de message étant défini sur "Sink", afin de signaler à son port partenaire qu'il peut commencer à fournir V_{BUS}.	
20	La couche protocole définit le bit ***Port Power Role*** de l'en-tête de message sur "Sink", crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
21	La couche physique ajoute un CRC et envoie le message ***PS_RDY***.	La couche physique reçoit le message ***PS_RDY*** et contrôle le CRC pour vérifier le message.
22		La couche physique supprime le CRC et transfère le message ***PS_RDY*** vers la couche protocole.
23		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***PS_RDY*** reçu au moteur de politique qui le consomme. Le moteur de politique arrête le ***PSSourceOffTimer***, demande au gestionnaire de politique d'utilisation des dispositifs d'appliquer la polarisation Rp, puis commence à commuter l'alimentation au fonctionnement source ***vSafe5V***.
24		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
25	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
26	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
27	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***PS_RDY*** a abouti. Le moteur de politique lance le ***PSSourceOnTimer***.	
28		Le moteur de politique, lorsque son alimentation est prête à fournir la puissance, demande à la couche protocole de former un message ***PS_RDY***. Le bit ***Port Power Role*** utilisé dans cet en-tête de message et dans les en-têtes de message suivants est à présent défini sur "Source".
29		La couche protocole crée le message ***PS_RDY*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
30	La couche physique reçoit le message ***PS_RDY*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***PS_RDY***.
31	La couche physique supprime le CRC et transfère le message ***PS_RDY*** vers la couche protocole.	

Etape	Port source initial	Port destinataire initial
32	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***PS_RDY*** reçu au moteur de politique qui le consomme.	
33	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
34	La couche physique ajoute un CRC et envoie le message ***GoodCRC***. Le moteur de politique arrête le ***PSSourceOnTimer***, informe l'alimentation qu'elle peut désormais alimenter le destinataire, puis réinitialise la couche protocole.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
35		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
36		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***PS_RDY*** a abouti. Le moteur de politique réinitialise ***CapsCounter***, réinitialise la couche de protocole et lance le ***SwapSourceStartTimer***, qui doit expirer avant d'envoyer un message ***Source_Capabilities*** .
	La permutation des rôles d'alimentation est terminée, les rôles ont été inversés et les ports partenaires sont libres de négocier plus de puissance.	

8.3.2.6.2 Permutation des rôles d'alimentation déclenchée par le destinataire sans négociation de puissance subséquente

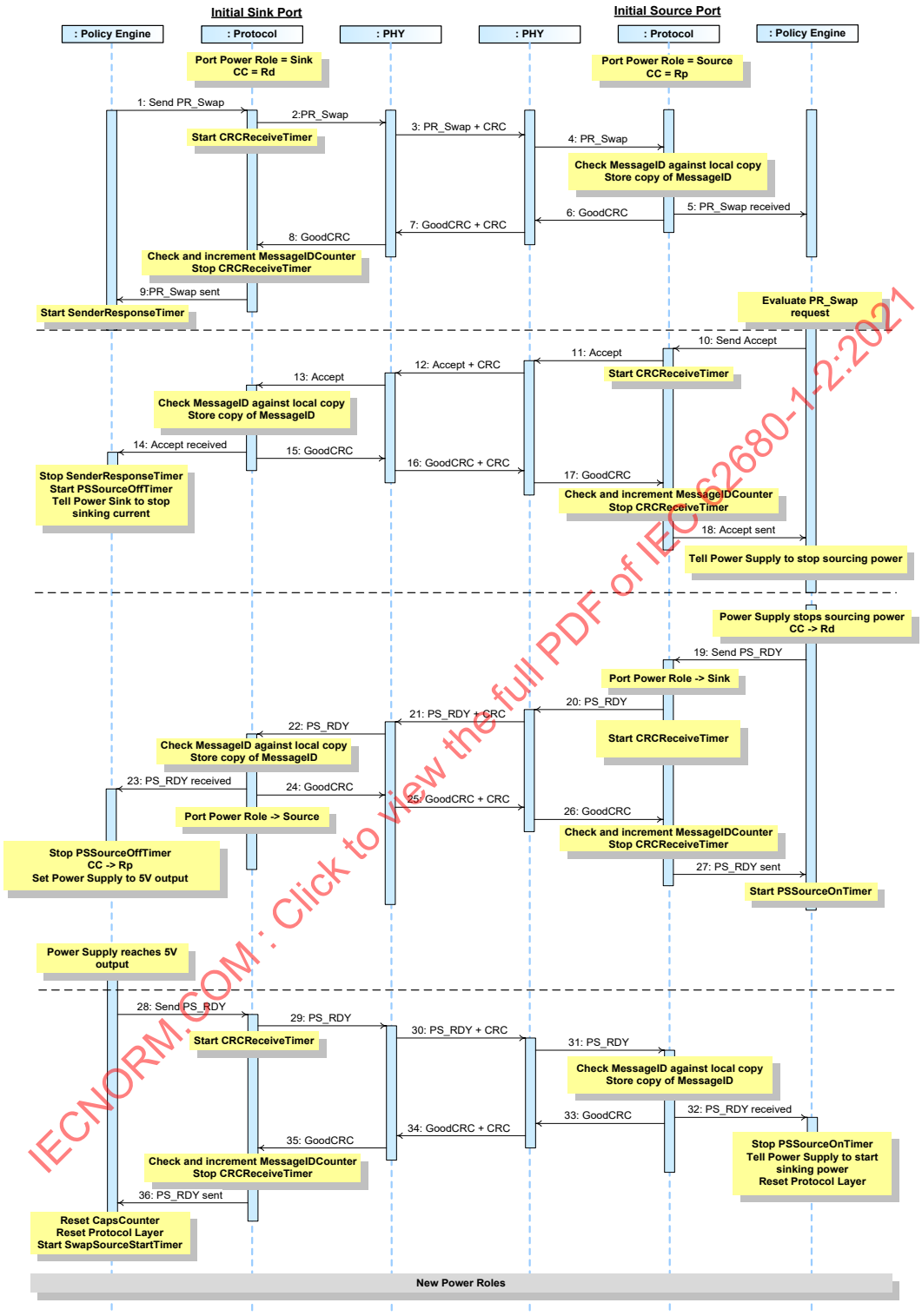
Il s'agit d'un exemple d'opération réussie de permutation des rôles d'alimentation initiée par un port qui, initialement, au début de cette séquence de messages, agit en tant que destinataire, et qui a donc une dépolarisation Rd sur son fil CC. Il n'inclut pas de négociation de puissance subséquente exigée afin d'établir un contrat explicite (voir 8.3.2.2).

Quatre phases distinctes caractérisent la négociation de permutation des rôles d'alimentation:

1. un message ***PR_Swap*** est envoyé;
2. un message ***Accept*** est reçu en réponse au message ***PR_Swap***;
3. le nouveau destinataire définit sa puissance restituée sur ***vSafe0V***, puis affirme Rd et envoie un message ***PS_RDY*** à l'issue de ce processus;
4. la nouvelle source affirme Rp, puis définit sa puissance restituée sur ***vSafe5V*** et envoie un message ***PS_RDY*** lorsqu'elle est prête à assurer l'alimentation.

La Figure 8-17 représente les messages au fur et à mesure de leur circulation à travers le bus et dans les dispositifs afin de procéder à la séquence de permutation des rôles d'alimentation.

Figure 8-17 Séquence réussie de permutation des rôles d'alimentation initiée par le destinataire



Anglais	Français
Initial Source Port	Port source initial
Initial Sink Port	Port destinataire initial
Policy Engine	Moteur de politique

Protocol	Protocole
PHY	PHY
Port Power Role = Source	Rôle d'alimentation du port = source
CC = Rp	CC = Rp
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Send	Envoyer
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
PR_Swap received	PR_Swap reçu
CRC	CRC
Check and increment	Vérifier et incrémenter
Stop	Arrêt
RP_Swap sent	RP_Swap envoyé
Evauate PR_Swap request	Evaluer la demande PR_Swap
Send Accept	Envoyer Accept
Accept	Accept
Accept + CRC	Accept + CRC
Accept received	Accept reçu
Tell Power Supply to stop sourcing power	Ordonner à l'alimentation d'arrêter de fournir de la puissance
Tell Power Supply stops sourcing power	Ordonner à l'alimentation d'arrêter de fournir de la puissance
CC -> Rd	CC -> Rd
Tell Power Sink to stop sinking current	Ordonner au destinataire de l'alimentation d'arrêter de consommer du courant
Port Power Role -> Sink	Rôle d'alimentation du port -> destinataire
PS_RDY received	PS_RDY reçu
Port Power Role -> Source	Rôle d'alimentation du port -> source
PS_RDY sent	PS_RDY envoyé
CC -> Rp	CC -> Rp
Set Power Supply to 5V output	Définir l'alimentation sur 5V en sortie
Power Supply reaches 5V output	L'alimentation atteint 5V en sortie
Tell Power Supply to start sinking current	Ordonner à l'alimentation de commencer à consommer du courant
Reset Protocol Layer	Réinitialiser la couche protocole
Reset	Réinitialisation
New Power Roles	Nouveaux rôles d'alimentation

Tableau 8-17 Etapes d'une séquence réussie de permutation des rôles d'alimentation initiée par le destinataire

Etape	Port destinataire initial	Port source initial
1	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***PR_Swap***.	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***PR_Swap***.	La couche physique reçoit le message ***PR_Swap*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***PR_Swap*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***PR_Swap*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***PR_Swap*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique évalue le message ***PR_Swap*** envoyé par le destinataire et décide qu'il est en mesure de procéder à la permutation des rôles d'alimentation et qu'il le souhaite. Il demande à la couche protocole de former un message ***Accept***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Accept***.	La couche physique ajoute un CRC et envoie le message ***Accept***.
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***PR_Swap*** reçu au moteur de politique qui le consomme.	
14	Le moteur de politique arrête le ***SenderResponseTimer***, lance le ***PSSourceOffTimer*** et demande à l'alimentation d'arrêter de consommer le courant.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.

Etape	Port destinataire initial	Port source initial
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Accept*** a abouti. Le moteur de politique demande à l'alimentation d'arrêter de fournir la puissance.
19		Le moteur de politique détermine que son alimentation ne fournit plus V_{BUS}. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'affirmer la dépolarisation Rd sur le fil CC. Le moteur de politique ordonne ensuite à la couche protocole de générer un message ***PS_RDY***, le bit ***Port Power Role*** de l'en-tête de message étant défini sur "Sink", afin de signaler à son port partenaire qu'il peut commencer à fournir V_{BUS}.
20		La couche protocole définit le bit ***Port Power Role*** de l'en-tête de message sur "Sink", crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
21	La couche physique reçoit le message ***PS_RDY*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***PS_RDY***.
22	La couche physique supprime le CRC et transfère le message ***PS_RDY*** vers la couche protocole.	
23	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***PS_RDY*** reçu au moteur de politique qui le consomme. Le moteur de politique arrête le ***PSSourceOffTimer***, demande au gestionnaire de politique d'utilisation des dispositifs d'appliquer la polarisation Rp, puis commence à commuter l'alimentation au fonctionnement source ***vSafe5V***.	
24	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
25	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.
26		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
27		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***PS_RDY*** a abouti. Le moteur de politique lance le ***PSSourceOnTimer***.
28	Le moteur de politique, lorsque son alimentation est prête à fournir la puissance, demande à la couche protocole de former un message ***PS_RDY***. Le bit ***Port Power Role*** utilisé dans cet en-tête de message et dans les en-têtes de message suivants est à présent défini sur "Source".	
29	La couche protocole crée le message ***PS_RDY*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
30	La couche physique ajoute un CRC et envoie le message ***PS_RDY***.	La couche physique reçoit le message ***PS_RDY*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
31		La couche physique supprime le CRC et transfère le message ***PS_RDY*** vers la couche protocole.

Etape	Port destinataire initial	Port source initial
32		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***PS_RDY*** reçu au moteur de politique qui le consomme. Le moteur de politique arrête le ***PSSourceOnTimer*** et informe l'alimentation qu'elle peut commencer à consommer la puissance.
33		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
34	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***. Le moteur de politique arrête le ***PSSourceOnTimer***, informe l'alimentation qu'elle peut désormais alimenter le destinataire, puis réinitialise la couche protocole.
35	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
36	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***PS_RDY*** a abouti. Le moteur de politique réinitialise ***CapsCounter***, réinitialise la couche de protocole et lance le ***SwapSourceStartTimer***, qui doit expirer avant d'envoyer un message ***Source_Capabilities*** .	
	La permutation des rôles d'alimentation est terminée, les rôles ont été inversés et les ports partenaires sont libres de négocier plus de puissance.	

8.3.2.7 Permutation rapide des rôles

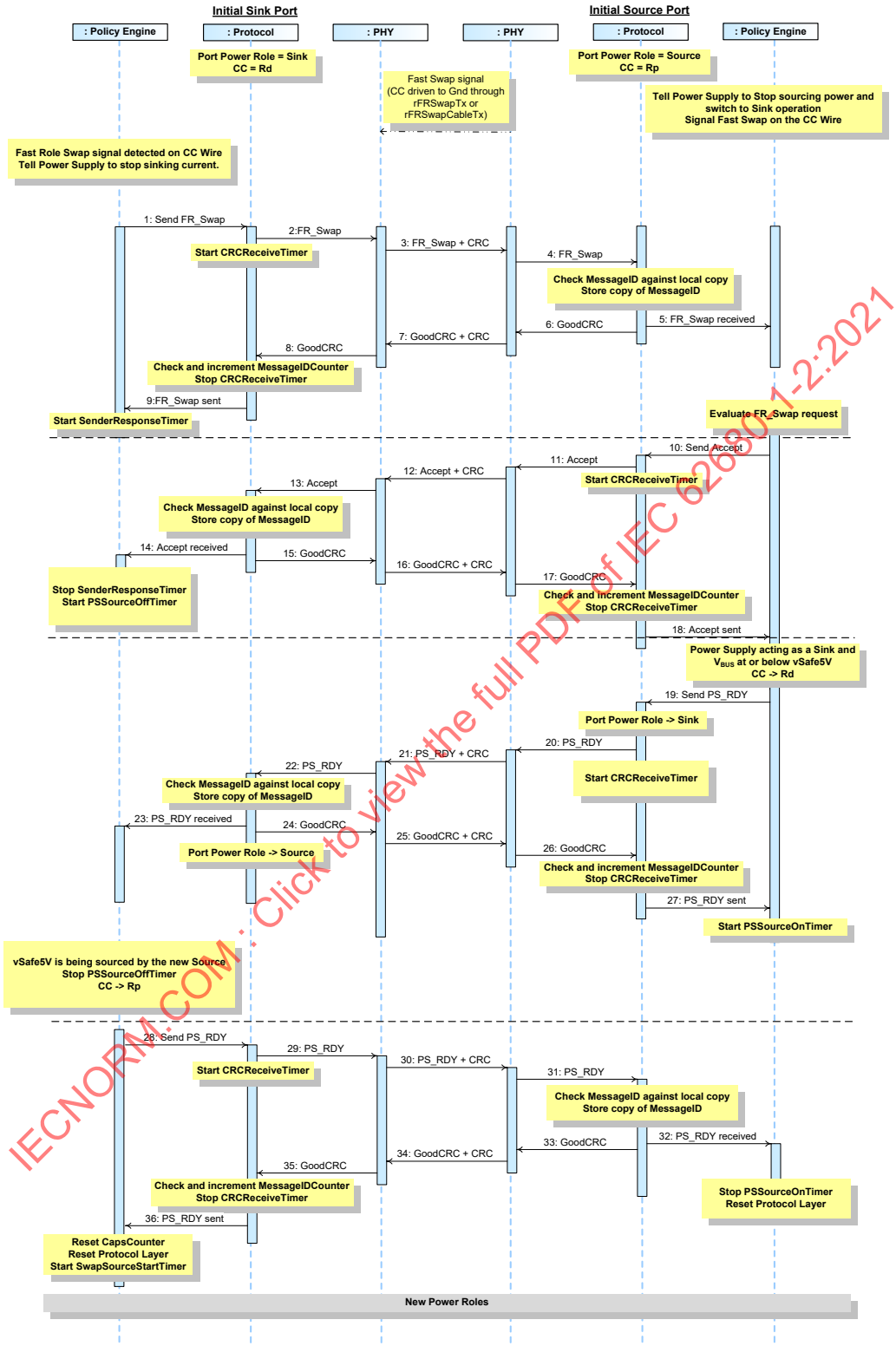
Il s'agit d'un exemple d'opération réussie de permutation rapide des rôles initiée par un port qui est initialement une source et a donc une polarisation Rp sur son fil CC, qui a perdu de la puissance et a besoin d'obtenir rapidement ***vSafe5V***. Il n'inclut pas de négociation de puissance subséquente exigée afin d'établir un contrat explicite (voir 8.3.2.2).

Plusieurs phases distinctes caractérisent la négociation de permutation rapide des rôles:

1. la source initiale cesse de fournir sa puissance restituée qui commence à passer à ***vSafe0V*** et signale la permutation rapide des rôles sur le fil CC; ces deux événements pourraient survenir simultanément ou dans n'importe quel ordre;
2. le destinataire initial arrête de consommer la puissance. A ce stade, la nouvelle source a toujours Rd affirmée et le nouveau destinataire a toujours Rp affirmée;
3. un message ***FR_Swap*** est envoyé par la nouvelle source dans un délai ***tFRSwapInit*** de détection du signal de permutation rapide;
4. un message ***Accept*** est envoyé par le nouveau destinataire en réponse au message ***FR_Swap***;
5. le nouveau destinataire affirme Rd et envoie un message ***PS_RDY*** indiquant que la tension sur V_{BUS} est inférieure ou égale à ***vSafe5V***;
6. la nouvelle source affirme Rp et envoie un message ***PS_RDY*** indiquant qu'elle agit en tant que source et qu'elle fournit ***vSafe5V***. Note: La nouvelle source peut commencer à appliquer V_{BUS} à tout moment où V_{BUS} se trouve au niveau ou au-dessous de ***vSafe5V*** (max), mais va commencer à amener V_{BUS} à ***vSafe5V*** au plus tard ***tSrcFRSwap*** après avoir détecté à la fois un signal FRS et que V_{BUS} est passée sous ***vSafe5V*** (min).

La Figure 8-18 représente les messages au fur et à mesure de leur circulation à travers le bus et dans les dispositifs afin de procéder à la permutation rapide des rôles.

Figure 8-18 Séquence réussie de permutation rapide des rôles



Anglais	Français
Initial Source Port	Port source initial
Initial Sink Port	Port destinataire initial

Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Source	Rôle d'alimentation du port = source
CC = Rp	CC = Rp
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Fast Swap signal (CC driven to Gnd through rFRSwapTx or rFRSwapCableTx)	Signal Fast Swap (conduit par CC vers Gnd à travers rFRSwapTx ou rFRSwapCableTx)
Tell Power Suppy to Stop sourcing power and switch to Sink operation	Ordonner à l'alimentation d'arrêter de fournir de la puissance et de passer à un fonctionnement en tant que destinataire
Signal Fast Swap on the CC Wire	Signal Fast Swap sur le fil CC
Fast Role Swap signal detected on CC Wire	Signal Fast Swap détecté sur le fil CC
Tell Power Supply to stop sinking current.	Ordonner à l'alimentation d'arrêter de consommer du courant
Send	Envoyer
CRC	CRC
Start	Début
Check	Vérifier
Store	Stocker
FR_Swap received	FR_Swap reçu
Check and increment	Vérifier et incrémenter
Stop	Arrêt
FR_Swap sent	FR_Swap envoyé
Evaluate FR_Swap request	Evaluer la demande FR_Swap
Port Power Role -> Sink	Rôle d'alimentation du port -> destinataire
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
PS_RDY received	PS_RDY reçu
Port Power Role -> Source	Rôle d'alimentation du port -> source
PS_RDY sent	PS_RDY envoyé
vSafe5V is being sourced by the new Source	vSafe5V est alimentée par la nouvelle source
CC -> Rp	CC -> Rp
Reset	Réinitialisation
New Power Roles	Nouveaux rôles d'alimentation

Tableau 8-18 Etapes d'une séquence réussie de permutation rapide des rôles

Etape	Port destinataire initial	Port source initial
1	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC. Le gestionnaire de politique d'utilisation des dispositifs détecte une permutation rapide sur le fil CC et demande à l'alimentation d'arrêter de consommer le courant. Le moteur de politique demande à la couche protocole d'envoyer un message ***FR_Swap*** dans un délai ***tFRSwapInit*** après détection du signal de permutation rapide.	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC. Le gestionnaire de politique d'utilisation des dispositifs demande à l'alimentation d'arrêter de fournir la puissance et de basculer vers un fonctionnement en tant que destinataire. Le gestionnaire de politique d'utilisation des dispositifs signale une permutation rapide sur le fil CC en connectant CC à la masse avec une résistance inférieure à ***rFRSwapTx*** pendant au moins ***tFRSwapTx***.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***FR_Swap***.	La couche physique reçoit le message ***FR_Swap*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***PR_Swap*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***FR_Swap*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***FR_Swap*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique évalue le message ***PR_Swap*** envoyé par le destinataire et décide qu'il est en mesure de procéder à la permutation des rôles d'alimentation et qu'il le souhaite. Il demande à la couche protocole de former un message ***Accept***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Accept***.	La couche physique ajoute un CRC et envoie le message ***Accept***.
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***PR_Swap*** reçu au moteur de politique qui le consomme.	
14	Le moteur de politique arrête le ***SenderResponseTimer*** et lance le ***PSSourceOffTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	

Etape	Port destinataire initial	Port source initial
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Accept*** a abouti.
19		Le moteur de politique détermine que son alimentation ne fournit plus V_{BUS} et agit en tant que destinataire. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'affirmer la dépolarisation Rd sur le fil CC. Le moteur de politique ordonne ensuite à la couche protocole de générer un message ***PS_RDY***, le bit ***Port Power Role*** de l'en-tête de message étant défini sur "Sink", afin de signaler à son port partenaire qu'il peut commencer à fournir V_{BUS}.
20		La couche protocole définit le bit ***Port Power Role*** de l'en-tête de message sur "Sink", crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
21	La couche physique reçoit le message ***PS_RDY*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***PS_RDY***.
22	La couche physique supprime le CRC et transfère le message ***PS_RDY*** vers la couche protocole.	
23	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***PS_RDY*** reçu au moteur de politique qui le consomme. Le moteur de politique arrête le ***PSSourceOffTimer***.	
24	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
25	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.
26		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
27		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***PS_RDY*** a abouti. Le moteur de politique lance le ***PSSourceOnTimer***.
28	Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'appliquer la polarisation Rp. Note: A un certain stade (avant ou après réception du message ***PS_RDY***), la nouvelle source a appliqué ***vSafe5V***, au plus tard ***tSrcFRSwap*** après avoir détecté le signal FRS et que V_{BUS} est passé sous ***vSafe5V***. Le moteur de politique, lorsque son alimentation est prête à fournir la puissance, demande à la couche protocole de former un message ***PS_RDY***. Le bit ***Port Power Role*** utilisé dans cet en-tête de message et dans les en-têtes de message suivants est à présent défini sur "Source".	

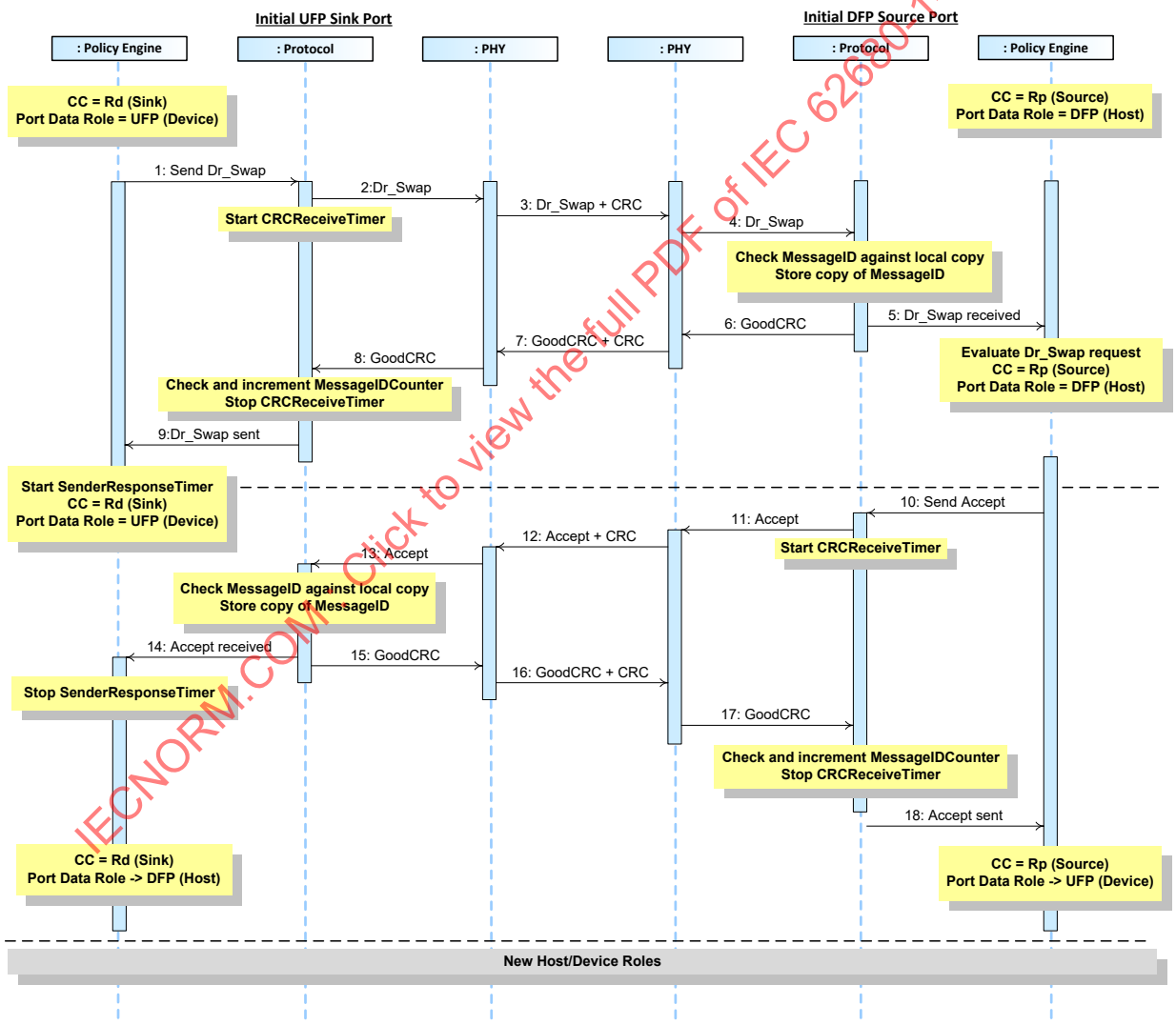
Etape	Port destinataire initial	Port source initial
29	La couche protocole crée le message ***PS_RDY*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
30	La couche physique ajoute un CRC et envoie le message ***PS_RDY***.	La couche physique reçoit le message ***PS_RDY*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
31		La couche physique supprime le CRC et transfère le message ***PS_RDY*** vers la couche protocole.
32		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***PS_RDY*** reçu au moteur de politique qui le consomme. Le moteur de politique arrête le ***PSSourceOnTimer***.
33		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
34	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***. Le moteur de politique réinitialise la couche protocole.
35	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
36	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***PS_RDY*** a abouti. Le moteur de politique réinitialise ***CapsCounter***, réinitialise la couche de protocole et lance le ***SwapSourceStartTimer***, qui doit expirer avant d'envoyer un message ***Source_Capabilities*** .	
	La permutation rapide des rôles est terminée, les rôles ont été inversés et les ports partenaires sont libres de négocier plus de puissance.	

8.3.2.8 Permutation des rôles de transmission de données

8.3.2.8.1 Permutation des rôles de transmission de données, déclenchée par un UFP fonctionnant en tant que destinataire

La Figure 8-19 donne un exemple de séquence entre un port, qui est au départ un UFP (dispositif) et un destinataire (Rd affirmée), et un port qui est au départ un DFP (hôte) et une source (Rp affirmée). Une permutation des rôles de transmission de données est déclenchée par l'UFP. Au cours de ce processus, les ports partenaires continuent de fonctionner en tant que source ou que destinataire (la puissance et Rp/Rd restent constants), mais échangent les rôles de transmission de données entre DFP (hôte) et UFP (dispositif).

Figure 8-19 Permutation des rôles de transmission de données, déclenchée par un UFP fonctionnant en tant que destinataire



Anglais	Français
Initial UFP Sink Port	Port destinataire UFP initial
Initial DFP Source Port	Port source DFP initial

Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
CC = Rd (Sink)	CC = Rd (destinataire)
Port Data Role = UFP (Device)	Rôle de transmission de données du port = UFP (dispositif)
CC = Rp (Source)	CC = Rp (Source)
Port Data Role = DFP (Host)	Rôle de transmission de données du port = DFP (hôte)
Send	Envoyer
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Dr_Swap received	Dr_Swap reçu
CRC	CRC
Evaluate Dr_Swap request	Evaluer la demande Dr_Swap
Check and increment	Vérifier et incrémenter
Dr_Swap sent	Dr_Swap envoyé
Stop	Arrêt
Send Accept	Envoyer Accept
Accept	Accept
Accept + CRC	Accept + CRC
Accept received	Accept reçu
Accept sent	Accept envoyé
New host/Device Roles	Nouveaux rôles hôte/dispositif

Le Tableau 8-19 ci-dessous donne une explication précise de ce qu'il se passe à chaque étape indiquée sur la Figure 8-19 ci-dessus.

Tableau 8-19 Etapes de la permutation des rôles de transmission de données, déclenchée par un UFP fonctionnant en tant que destinataire

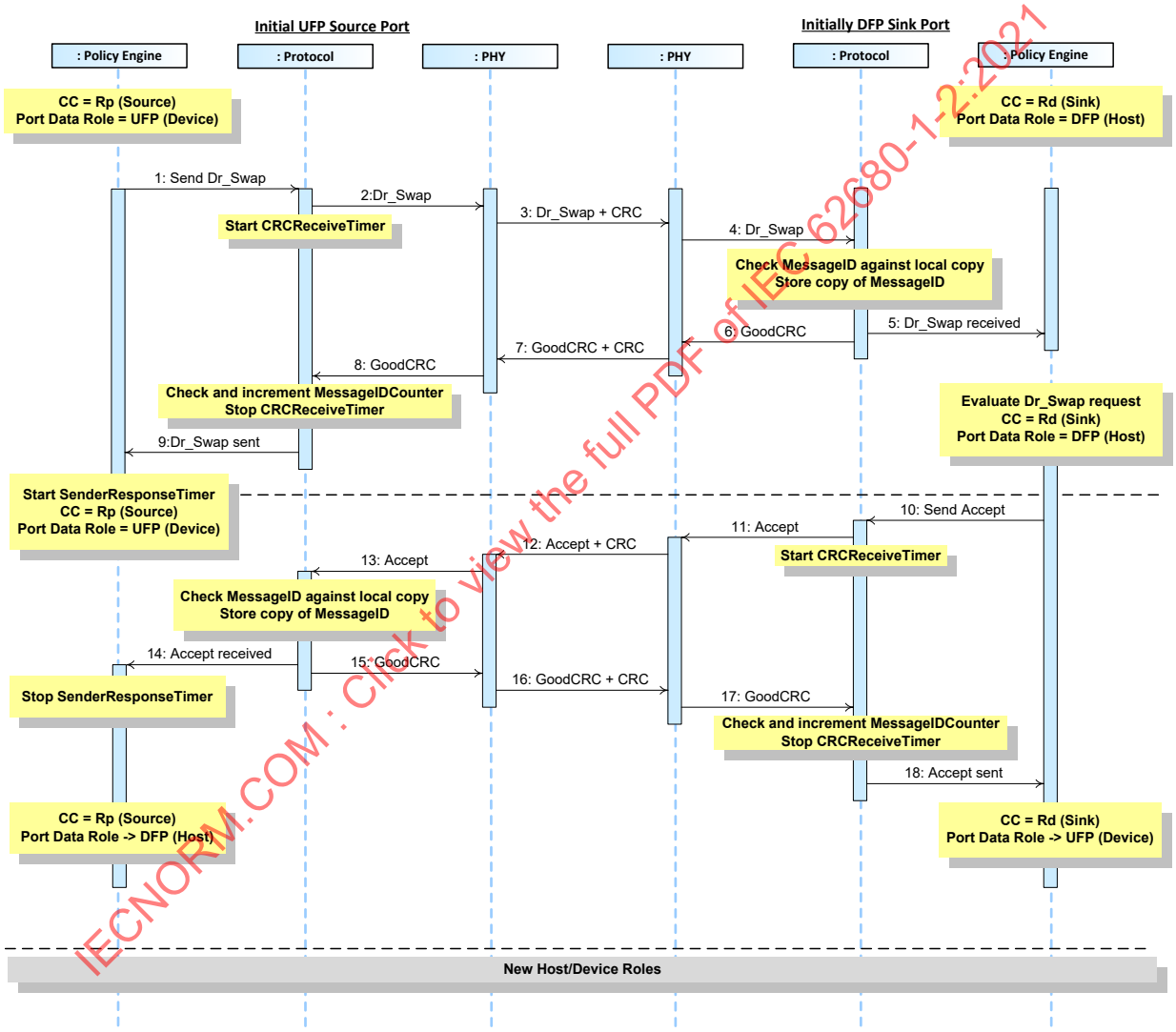
Etape	Port destinataire UFP initial	Port source DFP initial
1	Le port commence en tant qu'UFP (dispositif) qui fonctionne en tant que destinataire, avec Rd affirmée et ***Port Data Role*** défini sur UFP. Le moteur de politique demande à la couche protocole d'envoyer un message ***DR_Swap***.	Le port commence en tant que DFP (hôte) qui fonctionne en tant que source, avec Rp affirmée et ***Port Data Role*** défini sur DFP.
2	La couche protocole crée le message ***DR_Swap*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***DR_Swap***.	La couche physique reçoit le message ***DR_Swap*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***DR_Swap*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***DR_Swap*** reçu au moteur de politique qui le consomme.

Etape	Port destinataire UFP initial	Port source DFP initial
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***DR_Swap*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique évalue le message ***DR_Swap*** et décide qu'il est en mesure de procéder à la permutation des rôles de transmission de données et qu'il le souhaite. Il demande à la couche protocole de former un message ***Accept***.
11		La couche protocole crée le message ***Accept*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message ***Accept*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***Accept***.
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Accept*** reçu au moteur de politique qui le consomme.	
14	Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18	Le moteur de politique demande que le rôle de transmission de données passe d'UFP (dispositif) à DFP (hôte). Le rôle d'alimentation est à présent DFP (hôte), avec ***Port Data Role*** défini sur DFP, fonctionnant toujours en tant que destinataire (Rd affirmée).	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Accept*** a abouti. Le moteur de politique demande que le rôle de transmission de données passe à UFP (dispositif), avec ***Port Data Role*** défini sur UFP, et continue de fournir l'alimentation en tant que source (Rp affirmée).
	La permutation des rôles de transmission de données est terminée; les rôles de transmission de données ont été inversés tout en conservant le sens du débit de puissance.	

8.3.2.8.2 Permutation des rôles de transmission de données, déclenchée par un UFP fonctionnant en tant que source

La Figure 8-20 donne un exemple de séquence entre un port, qui est au départ un UFP (dispositif) et une source (Rp affirmée), et un port qui est au départ un DFP (hôte) et un destinataire (Rd affirmée). Une permutation des rôles de transmission de données est déclenchée par l'UFP. Au cours de ce processus, les ports partenaires continuent de fonctionner en tant que source ou que destinataire (la puissance et Rp/Rd restent constants), mais échangent les rôles de transmission de données entre DFP (hôte) et UFP (dispositif).

Figure 8-20 Permutation des rôles de transmission de données, déclenchée par un UFP fonctionnant en tant que source



Anglais	Français
Initial UFP Sink Port	Port destinataire UFP initial
Initial DFP Source Port	Port source DFP initial
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY

CC = Rd (Sink)	CC = Rd (destinataire)
Port Data Role = UFP (Device)	Rôle de transmission de données du port = UFP (dispositif)
CC = Rp (Source)	CC = Rp (Source)
Port Data Role = DFP (Host)	Rôle de transmission de données du port = DFP (hôte)
Send	Envoyer
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Dr_Swap received	Dr_Swap reçu
CRC	CRC
Evaluate Dr_Swap request	Evaluer la demande Dr_Swap
Check and increment	Vérifier et incrémenter
Dr_Swap sent	Dr_Swap envoyé
Stop	Arrêt
Send Accept	Envoyer Accept
Accept	Accept
Accept + CRC	Accept + CRC
Accept received	Accept reçu
Accept sent	Accept envoyé
New host/Device Roles	Nouveaux rôles hôte/dispositif

Le Tableau 8-20 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-20 ci-dessus.

Tableau 8-20 Etapes de la permutation des rôles de transmission de données, déclenchée par un UFP fonctionnant en tant que source

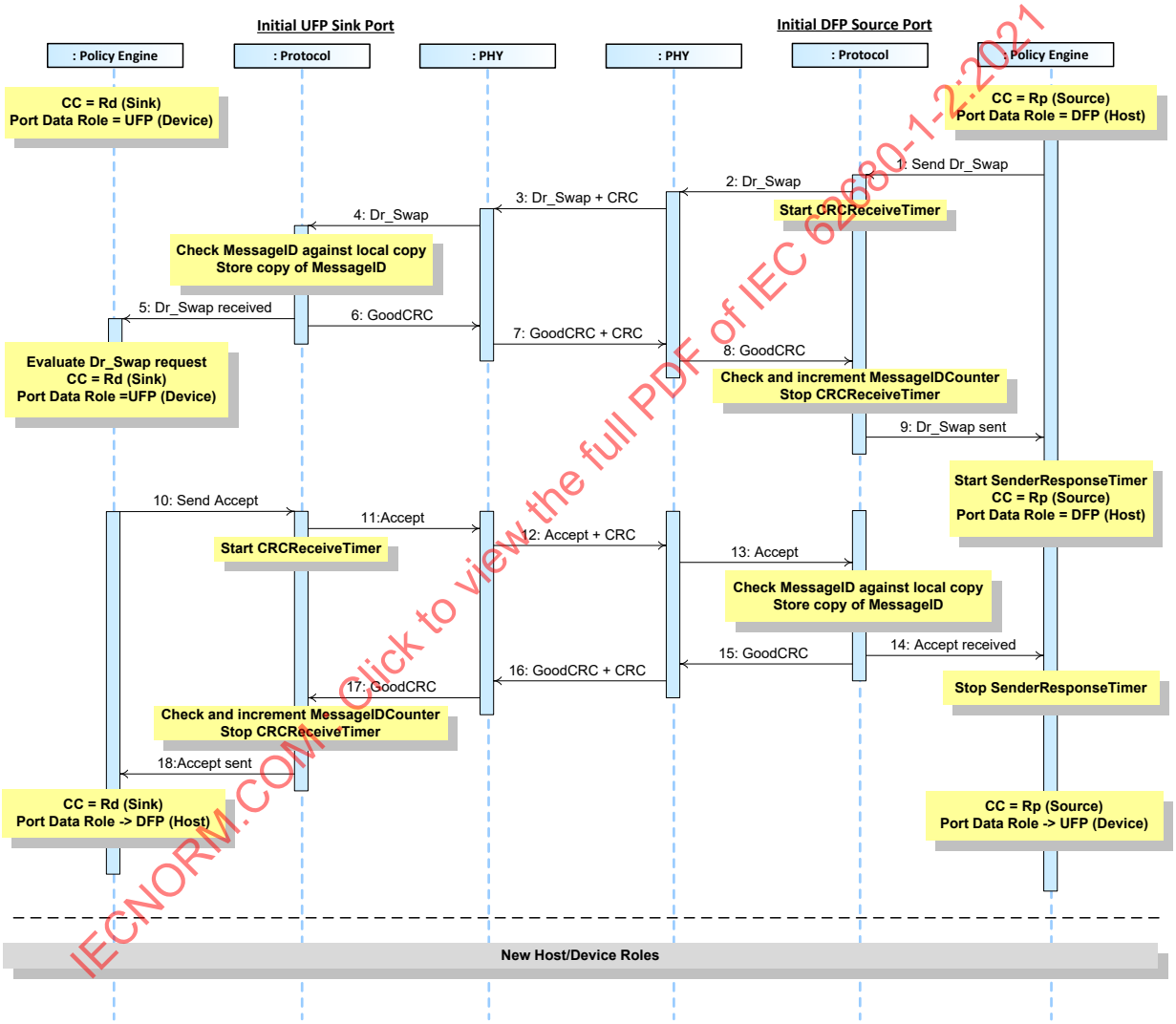
Etape	Port source UFP initial	Port destinataire DFP initial
1	Le port commence en tant qu'UFP (dispositif) qui fonctionne en tant que source, avec Rp affirmée et ***Port Data Role*** défini sur UFP. Le moteur de politique demande à la couche protocole d'envoyer un message ***DR_Swap***.	Le port commence en tant que DFP (hôte) qui fonctionne en tant que destinataire, avec Rd affirmée et ***Port Data Role*** défini sur DFP.
2	La couche protocole crée le message ***DR_Swap*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***DR_Swap***.	La couche physique reçoit le message ***DR_Swap*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***DR_Swap*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***DR_Swap*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.

Etape	Port source UFP initial	Port destinataire DFP initial
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***DR_Swap*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique évalue le message ***DR_Swap*** et décide qu'il est en mesure de procéder à la permutation des rôles de transmission de données et qu'il le souhaite. Il demande à la couche protocole de former un message ***Accept***.
11		La couche protocole crée le message ***Accept*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message ***Accept*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***Accept***.
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Accept*** reçu au moteur de politique qui le consomme.	
14	Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18	Le moteur de politique demande que le rôle de transmission de données passe d'UFP (dispositif) à DFP (hôte). Le rôle d'alimentation est à présent DFP (hôte), avec ***Port Data Role*** défini sur DFP, et continue de fournir la puissance en tant que source (Rp affirmée).	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Accept*** a abouti. Le moteur de politique demande que le rôle de transmission de données passe à UFP (dispositif), avec ***Port Data Role*** défini sur UFP, et qu'il continue de fonctionner en tant que destinataire (Rp affirmée).
	La permutation des rôles de transmission de données est terminée; les rôles de transmission de données ont été inversés tout en conservant le sens du débit de puissance.	

8.3.2.8.3 Permutation des rôles de transmission de données, déclenchée par un DFP fonctionnant en tant que source

La Figure 8-21 donne un exemple de séquence entre un port, qui est au départ un UFP (dispositif) et un destinataire (Rd affirmée), et un accès qui est au départ un DFP et une source (Rp affirmée). Une permutation des rôles de transmission de données est déclenchée par le DFP. Au cours de ce processus, les ports partenaires continuent de fonctionner en tant que source ou que destinataire (la puissance et Rp/Rd restent constants), mais échangent les rôles de transmission de données entre DFP (hôte) et UFP (dispositif).

Figure 8-21 Permutation des rôles de transmission de données, déclenchée par un DFP fonctionnant en tant que source



Anglais	Français
Initial UFP Sink Port	Port destinataire UFP initial
Initial DFP Source Port	Port source DFP initial
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
CC = Rd (Sink)	CC = Rd (destinataire)

Port Data Role = UFP (Device)	Rôle de transmission de données du port = UFP (dispositif)
CC = Rp (Source)	CC = Rp (Source)
Port Data Role = DFP (Host)	Rôle de transmission de données du port = DFP (hôte)
Send	Envoyer
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Dr_Swap received	Dr_Swap reçu
CRC	CRC
Evaluate Dr_Swap request	Evaluer la demande Dr_Swap
Check and increment	Vérifier et incrémenter
Dr_Swap sent	Dr_Swap envoyé
Stop	Arrêt
Send Accept	Envoyer Accept
Accept	Accept
Accept + CRC	Accept + CRC
Accept received	Accept reçu
Accept sent	Accept envoyé
New host/Device Roles	Nouveaux rôles hôte/dispositif

Le Tableau 8-21 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-21 ci-dessus.

Tableau 8-21 Etapes de la permutation des rôles de transmission de données, déclenchée par un DFP fonctionnant en tant que source

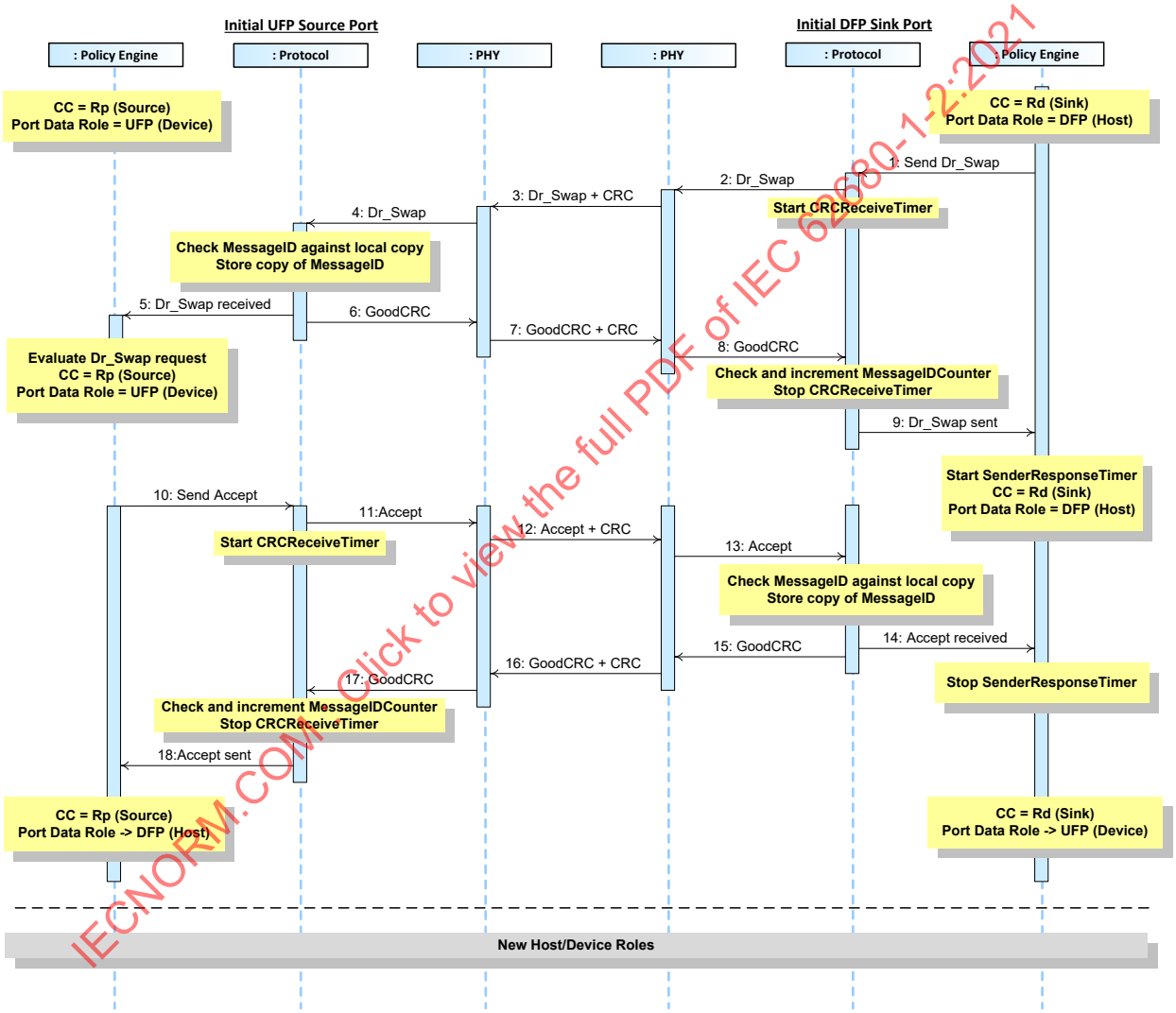
Etape	Port destinataire UFP initial	Port source DFP initial
1	Le port commence en tant qu'UFP (dispositif) qui fonctionne en tant que destinataire, avec Rd affirmée et ***Port Data Role*** défini sur UFP.	Le port commence en tant que DFP (hôte) qui fonctionne en tant que source, avec Rp affirmée et ***Port Data Role*** défini sur DFP. Le moteur de politique demande à la couche protocole d'envoyer un message ***DR_Swap***.
2		La couche protocole crée le message ***DR_Swap*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
3	La couche physique reçoit le message ***DR_Swap*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***DR_Swap***.
4	La couche physique supprime le CRC et transfère le message ***DR_Swap*** vers la couche protocole.	
5	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***DR_Swap*** reçu au moteur de politique qui le consomme.	
6	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
7	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.
8		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.

Etape	Port destinataire UFP initial	Port source DFP initial
9		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***DR_Swap*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.
10	Le moteur de politique évalue le message ***DR_Swap*** et décide qu'il est en mesure de procéder à la permutation des rôles de transmission de données et qu'il le souhaite. Il demande à la couche protocole de former un message ***Accept***.	
11	La couche protocole crée le message ***Accept*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
12	La couche physique ajoute un CRC et envoie le message ***Accept***.	La couche physique reçoit le message ***Accept*** et contrôle le CRC pour vérifier le message.
13		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Accept*** reçu au moteur de politique qui le consomme.
14		Le moteur de politique arrête le ***SenderResponseTimer***.
15		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
16	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
17	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
18	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Accept*** a abouti. Le moteur de politique demande que le rôle de transmission de données passe à DFP (hôte), avec ***Port Data Role*** défini sur DFP, et qu'il continue de fonctionner en tant que destinataire (Rp affirmée).	Le moteur de politique demande que le rôle de transmission de données passe de DFP (hôte) à UFP (dispositif). Le rôle d'alimentation est à présent UFP (dispositif), avec ***Port Data Role*** défini sur UFP, et continue de fournir la puissance en tant que source (Rp affirmée).
	La permutation des rôles de transmission de données est terminée; les rôles de transmission de données ont été inversés tout en conservant le sens du débit de puissance.	

8.3.2.8.4 Permutation des rôles de transmission de données, déclenchée par un DFP fonctionnant en tant que destinataire

La Figure 8-22 donne un exemple de séquence entre un port, qui est au départ un UFP (dispositif) et une source (Rp affirmée), et un port qui est au départ un DFP (hôte) et un destinataire (Rd affirmée). Une permutation des rôles de transmission de données est déclenchée par le DFP. Au cours de ce processus, les ports partenaires continuent de fonctionner en tant que source ou que destinataire (la puissance et Rp/Rd restent constants), mais échangent les rôles de transmission de données entre DFP (hôte) et UFP (dispositif).

Figure 8-22 Permutation des rôles de transmission de données, déclenchée par un DFP fonctionnant en tant que destinataire



Anglais	Français
Initial UFP Sink Port	Port destinataire UFP initial
Initial DFP Source Port	Port source DFP initial
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
CC = Rd (Sink)	CC = Rd (destinataire)

Port Data Role = UFP (Device)	Rôle de transmission de données du port = UFP (dispositif)
CC = Rp (Source)	CC = Rp (Source)
Port Data Role = DFP (Host)	Rôle de transmission de données du port = DFP (hôte)
Send	Envoyer
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Dr_Swap received	Dr_Swap reçu
CRC	CRC
Evaluate Dr_Swap request	Evaluer la demande Dr_Swap
Check and increment	Vérifier et incrémenter
Dr_Swap sent	Dr_Swap envoyé
Stop	Arrêt
Send Accept	Envoyer Accept
Accept	Accept
Accept + CRC	Accept + CRC
Accept received	Accept reçu
Accept sent	Accept envoyé
New host/Device Roles	Nouveaux rôles hôte/dispositif

Le Tableau 8-22 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-22 ci-dessus.

Tableau 8-22 Etapes de la permutation des rôles de transmission de données, déclenchée par un DFP fonctionnant en tant que destinataire

Etape	Port source UFP initial	Port destinataire DFP initial
1	Le port commence en tant qu'UFP (dispositif) qui fonctionne en tant que source, avec Rp affirmée et ***Port Data Role*** défini sur UFP.	Le port commence en tant que DFP (hôte) qui fonctionne en tant que destinataire, avec Rd affirmée et ***Port Data Role*** défini sur DFP. Le moteur de politique demande à la couche protocole d'envoyer un message ***DR_Swap***.
2		La couche protocole crée le message ***DR_Swap*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
3	La couche physique reçoit le message ***DR_Swap*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***DR_Swap***.
4	La couche physique supprime le CRC et transfère le message ***DR_Swap*** vers la couche protocole.	
5	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***DR_Swap*** reçu au moteur de politique qui le consomme.	
6	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
7	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.
8		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.

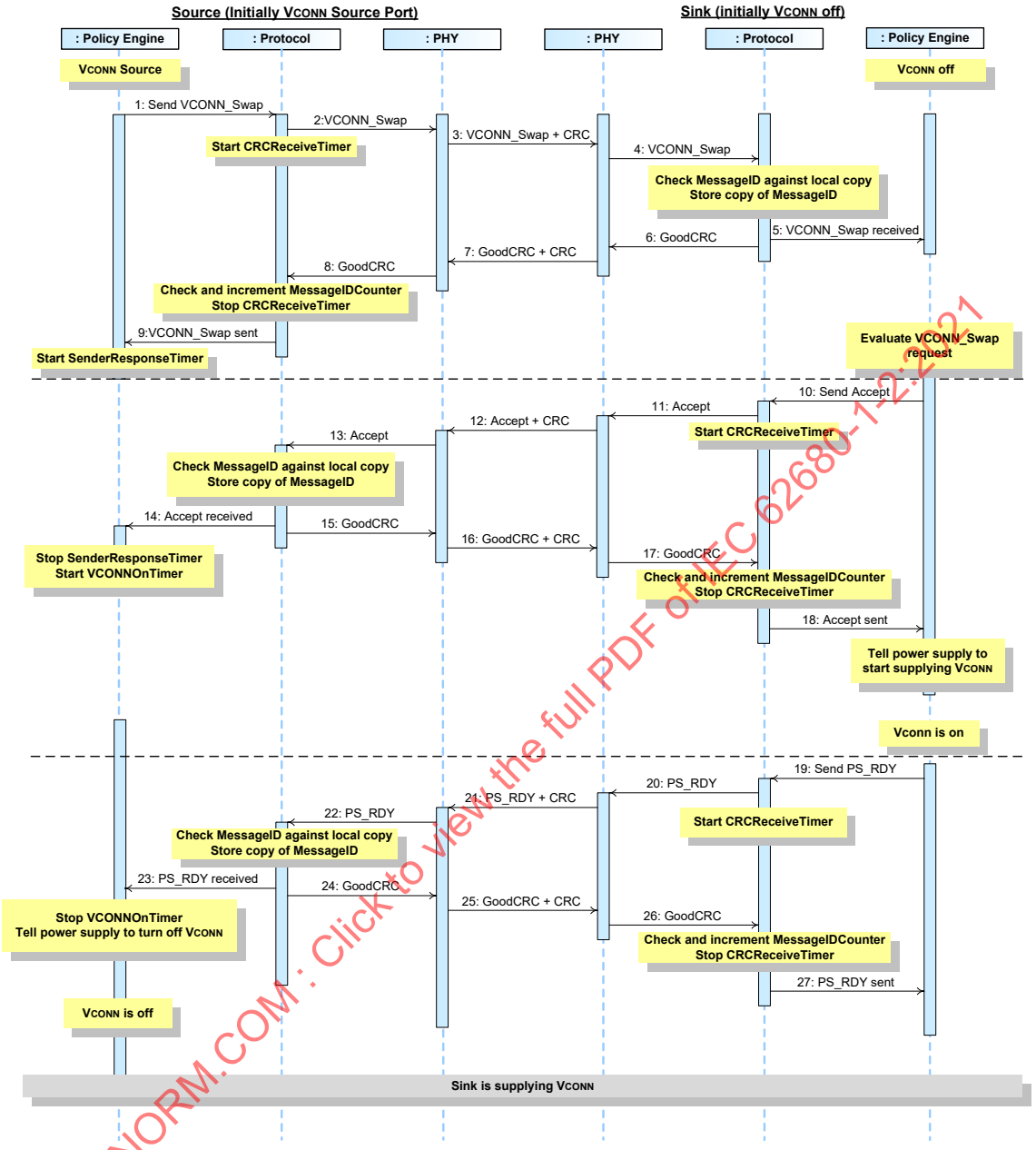
Etape	Port source UFP initial	Port destinataire DFP initial
9		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***DR_Swap*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.
10	Le moteur de politique évalue le message ***DR_Swap*** et décide qu'il est en mesure de procéder à la permutation des rôles de transmission de données et qu'il le souhaite. Il demande à la couche protocole de former un message ***Accept***.	
11	La couche protocole crée le message ***Accept*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
12	La couche physique ajoute un CRC et envoie le message ***Accept***.	La couche physique reçoit le message ***Accept*** et contrôle le CRC pour vérifier le message.
13		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Accept*** reçu au moteur de politique qui le consomme.
14		Le moteur de politique arrête le ***SenderResponseTimer***.
15		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
16	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
17	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
18	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Accept*** a abouti. Le moteur de politique demande que le rôle de transmission de données passe à DFP (hôte), avec ***Port Data Role*** défini sur DFP, et continue de fournir l'alimentation en tant que source (Rp affirmée).	Le moteur de politique demande que le rôle de transmission de données passe de DFP (hôte) à UFP (dispositif). Le rôle d'alimentation est à présent UFP (hôte), avec ***Port Data Role*** défini sur UFP, fonctionnant toujours en tant que destinataire (Rd affirmée).
	La permutation des rôles de transmission de données est terminée; les rôles de transmission de données ont été inversés tout en conservant le sens du débit de puissance.	

8.3.2.9 Permutation de VCONN

8.3.2.9.1 Permutation de la source VCONN de source à destinataire

La Figure 8-23 donne un exemple de séquence entre une source et un destinataire, où la source fournit au départ VCONN, puis demande au destinataire de s'en charger. Au cours du processus, les ports partenaires, qui conservent leur rôle de source ou de destinataire, continuent de fonctionner en tant que tels (la puissance reste constante), mais échangent la source VCONN de la source au destinataire.

Figure 8-23 Permutation de la source VCONN de source à destinataire



Anglais	Français
Source (Initially VCONN Source Port)	Source (au départ le port source VCONN)
Sink (Initially VCONN off)	Destinataire (au départ VCONN désactivée)
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
VCONN Source	Source VCONN
VCONN off	VCONN désactivée
Send	Envoyer
CRC	CRC

Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
VCONN_Swap received	VCONN_Swap reçu
Check and increment	Vérifier et incrémenter
Stop	Arrêt
VCONN_Swap sent	VCONN_Swap envoyé
Evaluate VCONN_Swap request	Evaluer la demande VCONN_Swap
Send Accept	Envoyer Accept
Accept	Accept
Accept + CRC	Accept + CRC
Accept received	Accept reçu
Tell power supply to start supplying V_{CONN}	Ordonner à l'alimentation de commencer à fournir V_{CONN}
Vconn is on	Vconn est activée
PS_RDY received	PS_RDY reçu
Tell power supply to turn off V_{CONN}	Ordonner à l'alimentation de désactiver V_{CONN}
PS_RDY sent	PS_RDY envoyé
V_{CONN} is off	V_{CONN} est désactivée
Sink is supplying V_{CONN}	Le destinataire fournit V_{CONN}

Le Tableau 8-23 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-23 ci-dessus.

Tableau 8-23 Etapes de la permutation de la source V_{CONN} de source à destinataire

Etape	Source (initialement la source V_{CONN})	Destinataire (au départ V_{CONN} désactivée)
1	La source démarre en tant que source V_{CONN}. Le moteur de politique demande à la couche protocole d'envoyer un message ***VCONN_Swap***.	Le destinataire démarre avec V_{CONN} désactivée.
2	La couche protocole crée le message ***VCONN_Swap*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***VCONN_Swap***.	La couche physique reçoit le message ***VCONN_Swap*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***VCONN_Swap*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***VCONN_Swap*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	

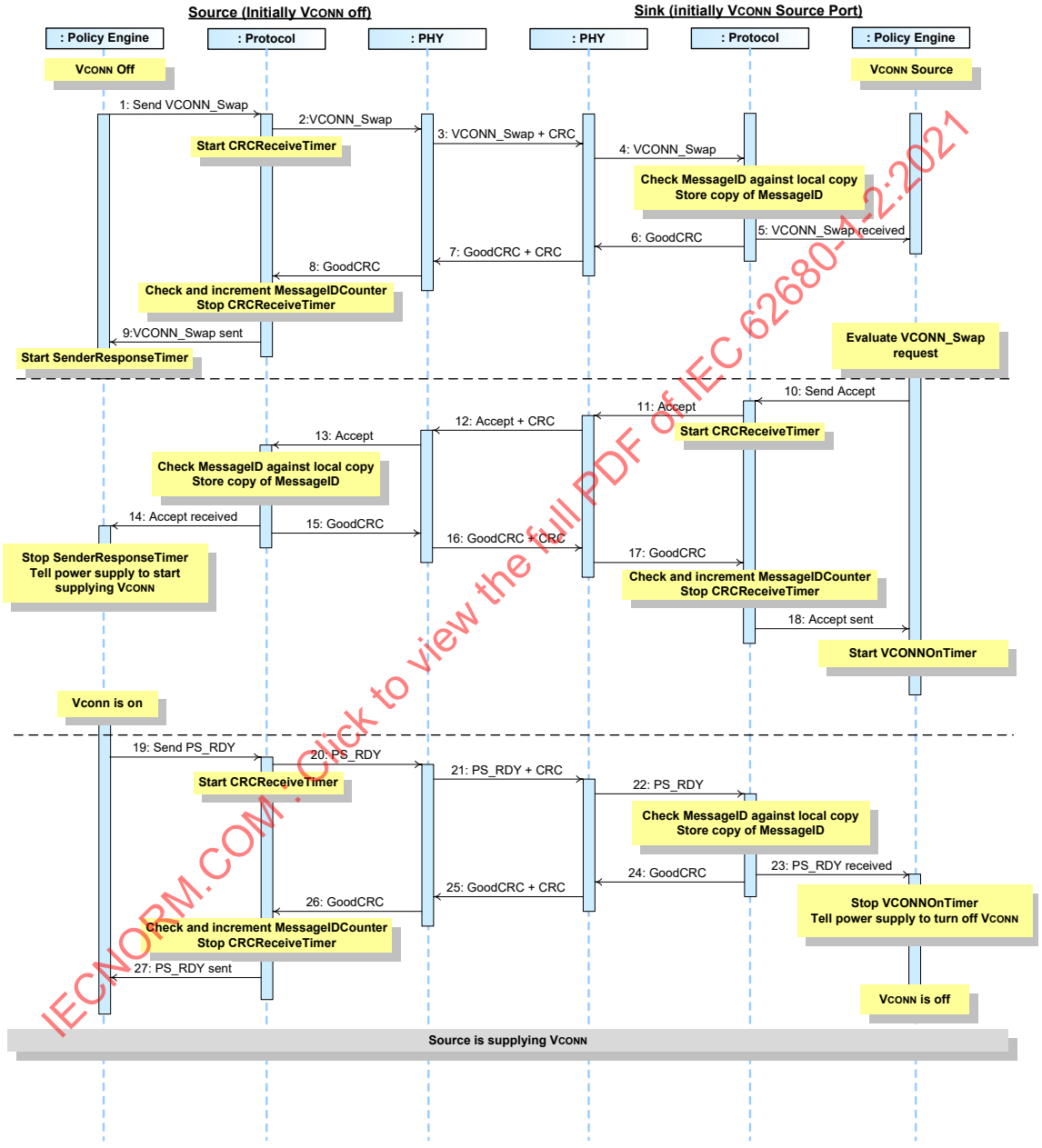
Etape	Source (initialement la source VCONN)	Destinataire (au départ VCONN désactivée)
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***VCONN_Swap*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique évalue le message ***VCONN_Swap*** envoyé par la source et décide qu'il est en mesure de procéder à la permutation de VCONN et qu'il le souhaite. Il demande à la couche protocole de former un message ***Accept***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message ***Accept*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***Accept***.
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Accept*** reçu au moteur de politique qui le consomme.	
14	Le moteur de politique arrête le ***SenderResponseTimer*** et lance le ***VCONNOnTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Accept*** a abouti. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'activer VCONN.
19		Le gestionnaire de politique d'utilisation des dispositifs informe le moteur de politique que son alimentation fournit VCONN. Le moteur de politique demande à la couche protocole de générer un message ***PS_RDY*** pour signaler à la source qu'elle peut désactiver VCONN.
20		La couche protocole crée le message ***PS_RDY*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
21	La couche physique reçoit le message ***PS_RDY*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***PS_RDY***.
22	La couche physique supprime le CRC et transfère le message ***PS_RDY*** vers la couche protocole.	
23	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***PS_RDY*** reçu au moteur de politique qui le consomme.	

Etape	Source (initialement la source VCONN)	Destinataire (au départ VCONN désactivée)
24	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
25	La couche physique ajoute un CRC et envoie le message ***GoodCRC***. Le moteur de politique arrête le ***VCONNOnTimer*** et demande à l'alimentation d'arrêter de fournir VCONN.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
26		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
27	VCONN est désactivée.	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***PS_RDY*** a abouti.
	Le destinataire est à présent la source VCONN, et la source a VCONN désactivée.	

8.3.2.9.2 Permutation de la source VCONN de destinataire à source

La Figure 8-24 donne un exemple de séquence entre une source et un destinataire, où la source commence par fournir VCONN avant de demander au destinataire de devenir la source VCONN. Au cours du processus, les ports partenaires, qui conservent leur rôle de source ou de destinataire, continuent de fonctionner en tant que tels (la puissance reste constante), mais échangent la source VCONN du destinataire vers la source.

Figure 8-24 Permutation de la source VCONN de destinataire à source



Anglais	Français
Source (Initially V_{CONN} off)	Source (au départ V_{CONN} désactivée)
Sink (Initially V_{CONN} Source Port)	Destinataire (au départ port source V_{CONN})
Policy Engine	Moteur de politique
Protocol	Protocole

PHY	PHY
V_{CONN} Source	Source V_{CONN}
V_{CONN} off	V_{CONN} désactivée
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
VCONN_Swap received	VCONN_Swap reçu
Check and increment	Vérifier et incrémenter
Stop	Arrêt
VCONN_Swap sent	VCONN_Swap envoyé
Evaluate VCONN_Swap request	Evaluer la demande VCONN_Swap
Send Accept	Envoyer Accept
Accept	Accept
Accept + CRC	Accept + CRC
Accept received	Accept reçu
Tell power supply to start supplying V_{CONN}	Ordonner à l'alimentation de commencer à fournir V_{CONN}
Vconn is on	Vconn est activée
PS_RDY received	PS_RDY reçu
Tell power supply to turn off V_{CONN}	Ordonner à l'alimentation de désactiver V_{CONN}
PS_RDY sent	PS_RDY envoyé
V_{CONN} is off	V_{CONN} est désactivée
Sink is supplying V_{CONN}	Le destinataire fournit V_{CONN}

Le Tableau 8-24 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-24 ci-dessus.

Tableau 8-24 Etapes de la permutation de la source V_{CONN} de destinataire à source

Etape	Source	Destinataire
1	La source démarre avec V_{CONN} désactivée. Le moteur de politique demande à la couche protocole d'envoyer un message ***VCONN_Swap***.	Le destinataire démarre en tant que source V_{CONN}.
2	La couche protocole crée le message ***VCONN_Swap*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***VCONN_Swap***.	La couche physique reçoit le message ***VCONN_Swap*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***VCONN_Swap*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***VCONN_Swap*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.

Etape	Source	Destinataire
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***VCONN_Swap*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique évalue le message ***VCONN_Swap*** envoyé par la source et décide qu'il est en mesure de procéder à la permutation de VCONN et qu'il le souhaite. Il demande à la couche protocole de former un message ***Accept***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message ***Accept*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***Accept***.
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Accept*** reçu au moteur de politique qui le consomme.	
14	Le moteur de politique arrête le ***SenderResponseTimer***. Le moteur de politique demande au gestionnaire de politique d'utilisation des dispositifs d'activer VCONN.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Accept*** a abouti. Le moteur de politique lance le ***VCONNOnTimer***.
19	Le gestionnaire de politique d'utilisation des dispositifs signale au moteur de politique que son alimentation fournit VCONN. Le moteur demande à la couche protocole de générer un message ***PS_RDY*** pour signaler au destinataire qu'il peut désactiver VCONN.	
20	La couche protocole crée le message ***PS_RDY*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
21	La couche physique ajoute un CRC et envoie le message ***PS_RDY***.	La couche physique reçoit le message ***PS_RDY*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
22		La couche physique supprime le CRC et transfère le message ***PS_RDY*** vers la couche protocole.

Etape	Source	Destinataire
23		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***PS_RDY*** reçu au moteur de politique qui le consomme.
24		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
25	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***. Le moteur de politique arrête le ***VCONNOnTimer*** et demande à l'alimentation d'arrêter de fournir VCONN.
26	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
27	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***PS_RDY*** a abouti.	VCONN est désactivée.
	La source est à présent la source VCONN, et le destinataire a VCONN désactivée.	

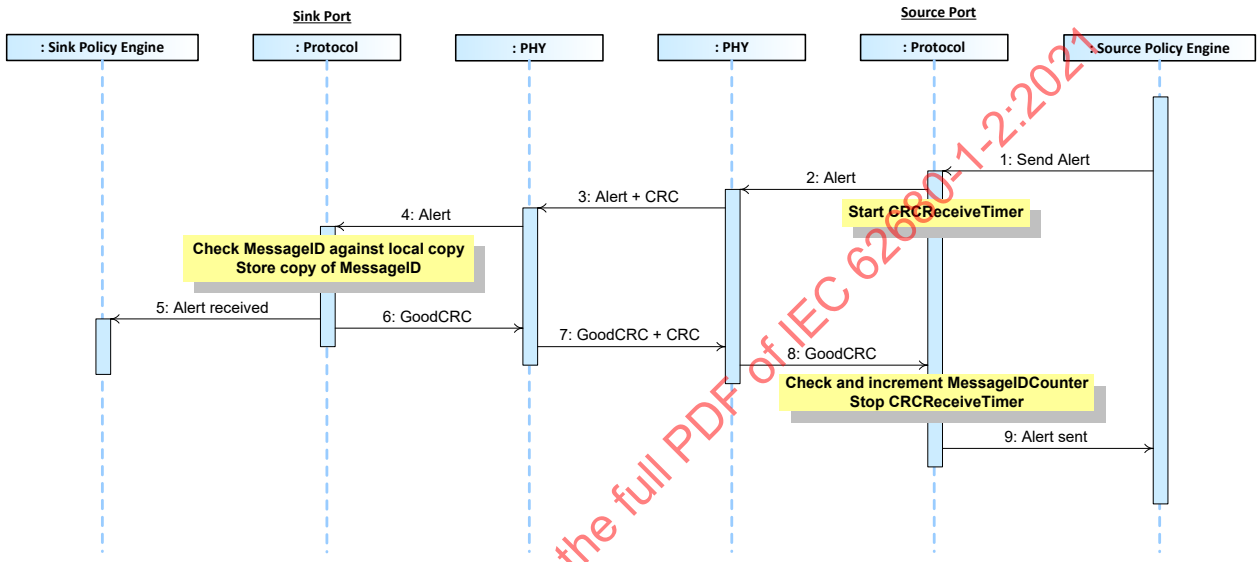
8.3.2.10 Capacités, statuts et informations supplémentaires

8.3.2.10.1 Alert

8.3.2.10.1.1 La source envoie une alerte au destinataire

La Figure 8-25 donne un exemple de séquence entre une source et un destinataire, où la source alerte le destinataire d'un changement de statut. Cet AMS est suivi par l'obtention du statut de la source afin de déterminer d'autres détails de l'alerte (voir 8.3.2.10.2).

Figure 8-25 Alerte de la source au destinataire



Anglais	Français
Sink Port	Port destinataire
Source Port	Port source
Sink Policy Engine	Moteur de politique du destinataire
Source Policy Engine	Moteur de politique de la source
Protocol	Protocole
PHY	PHY
Send Alert	Envoyer Alert
Alert	Alert
Alert + CRC	Alert + CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Alert received	Alert reçu
CRC	CRC
Check and increment	Vérifier et incrémenter
Stop	Arrêt
Alert sent	Alert envoyé

Le Figure 8-25 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-25 ci-dessus.

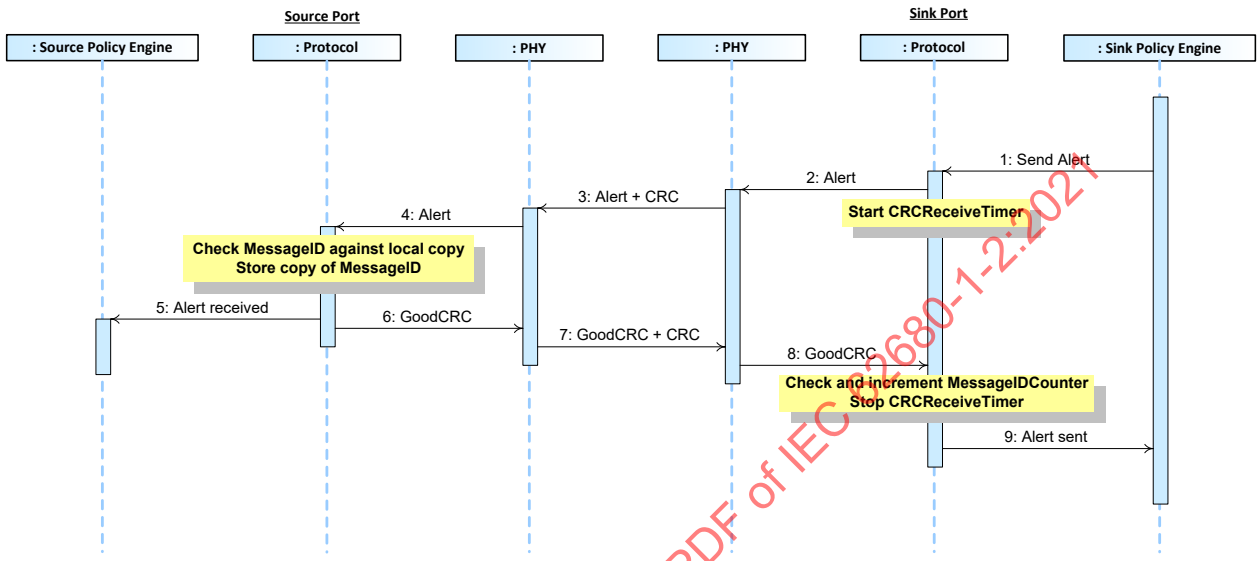
Tableau 8-25 Etapes de l'alerte de la source au destinataire

Etape	Destinataire	Source
1		Le gestionnaire de politique d'utilisation des dispositifs indique une condition d'alerte de la source. Le moteur de politique demande à la couche protocole de former un message ***Alert***.
2		La couche protocole crée le message ***Alert*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
3	La couche physique reçoit le message ***Alert*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***Alert***.
4	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère le message ***Alert*** reçu au moteur de politique qui le consomme.	
5	Le moteur de politique informe le gestionnaire de politique d'utilisation des dispositifs.	
6	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
7	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
8		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
9		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Alert*** a abouti.

8.3.2.10.1.1 Le destinataire envoie une alerte à la source

La Figure 8-26 donne un exemple de séquence entre une source et un destinataire, où le destinataire alerte la source d'un changement de statut. Cet AMS est suivi par l'obtention du statut du destinataire afin de déterminer d'autres détails de l'alerte (voir 8.3.2.10.2).

Figure 8-26 Alerte du destinataire à la source



Anglais	Français
Sink Port	Port destinataire
Source Port	Port source
Sink Policy Engine	Moteur de politique du destinataire
Source Policy Engine	Moteur de politique de la source
Protocol	Protocole
PHY	PHY
Send Alert	Envoyer Alert
Alert	Alert
Alert + CRC	Alert + CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Alert received	Alert reçu
CRC	CRC
Check and increment	Vérifier et incrémenter
Stop	Arrêt
Alert sent	Alert envoyé

Le Tableau 8-26 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-26 ci-dessus.

Tableau 8-26 Etapes de l'alerte du destinataire à la source

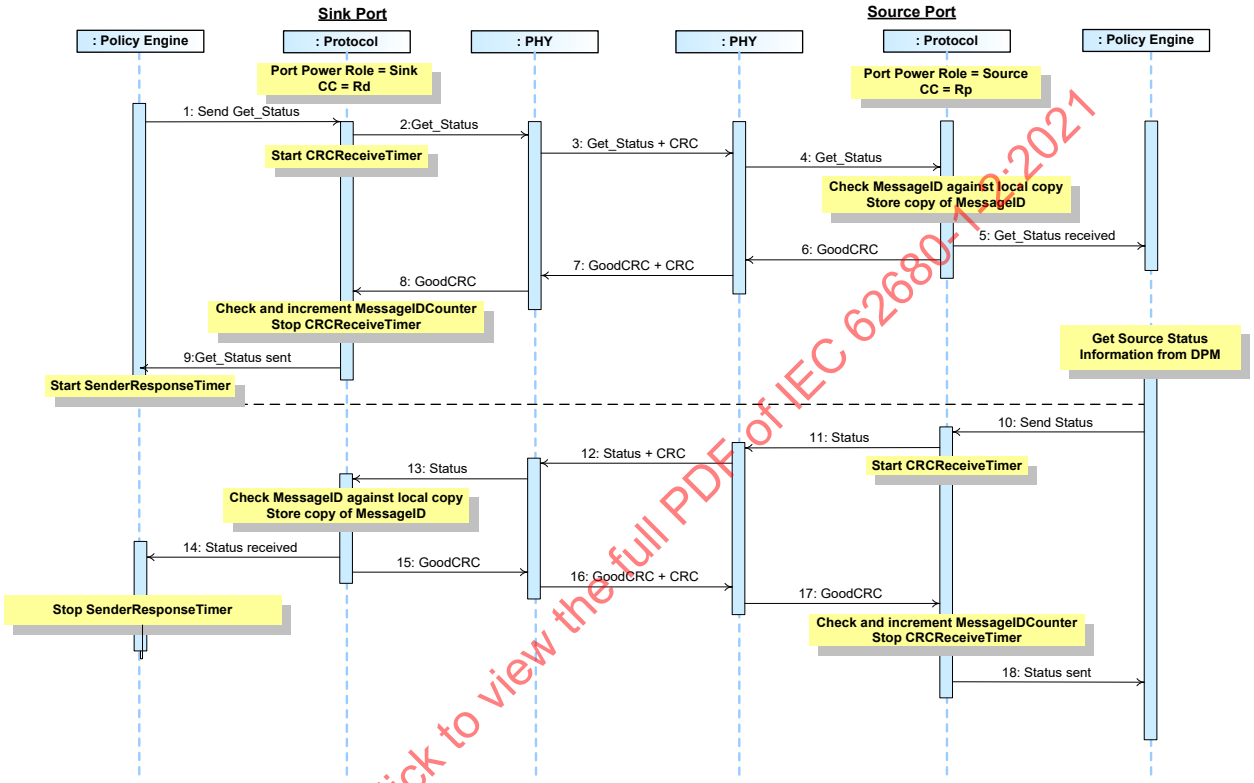
Etape	Source	Destinataire
1		Le gestionnaire de politique d'utilisation des dispositifs indique une condition d'alerte du destinataire. Le moteur de politique demande à la couche protocole de former un message ***Alert***.
2		La couche protocole crée le message ***Alert*** et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
3	La couche physique reçoit le message ***Alert*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***Alert***.
4	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère le message ***Alert*** reçu au moteur de politique qui le consomme.	
5	Le moteur de politique informe le gestionnaire de politique d'utilisation des dispositifs.	
6	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
7	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
8		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
9		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Alert*** a abouti.

8.3.2.10.2 Statut

8.3.2.10.2.1 Destinataire obtenant le statut de la source

La Figure 8-27 donne un exemple de séquence entre une source et un destinataire où, lorsque le destinataire a reçu une alerte (voir 8.3.2.10.1) signalant qu'un changement de statut a eu lieu, le destinataire obtient plus d'informations sur ce changement.

Figure 8-27 Destinataire obtenant le statut de la source



Anglais	Français
Sink Port	Port destinataire
Source Port	Port source
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Port Power Role = Source	Rôle d'alimentation du port = source
CC = Rp	CC = Rp
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Get_Status received	Get_Status reçu

Check and increment	Vérifier et incrémenter
Stop	Arrêt
Get_Status sent	Get_Status envoyé
Get Source Status information from DPM	Obtenir les informations relatives au statut de la source auprès du DPM
Send Status	Envoyer statut
Status	Statut
Status + CRC	Statut + CRC
Status received	Statut reçu
Status sent	Statut envoyé

Le Tableau 8-27 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-27 ci-dessus.

Tableau 8-27 Etapes d'une séquence où le destinataire obtient le statut de la source

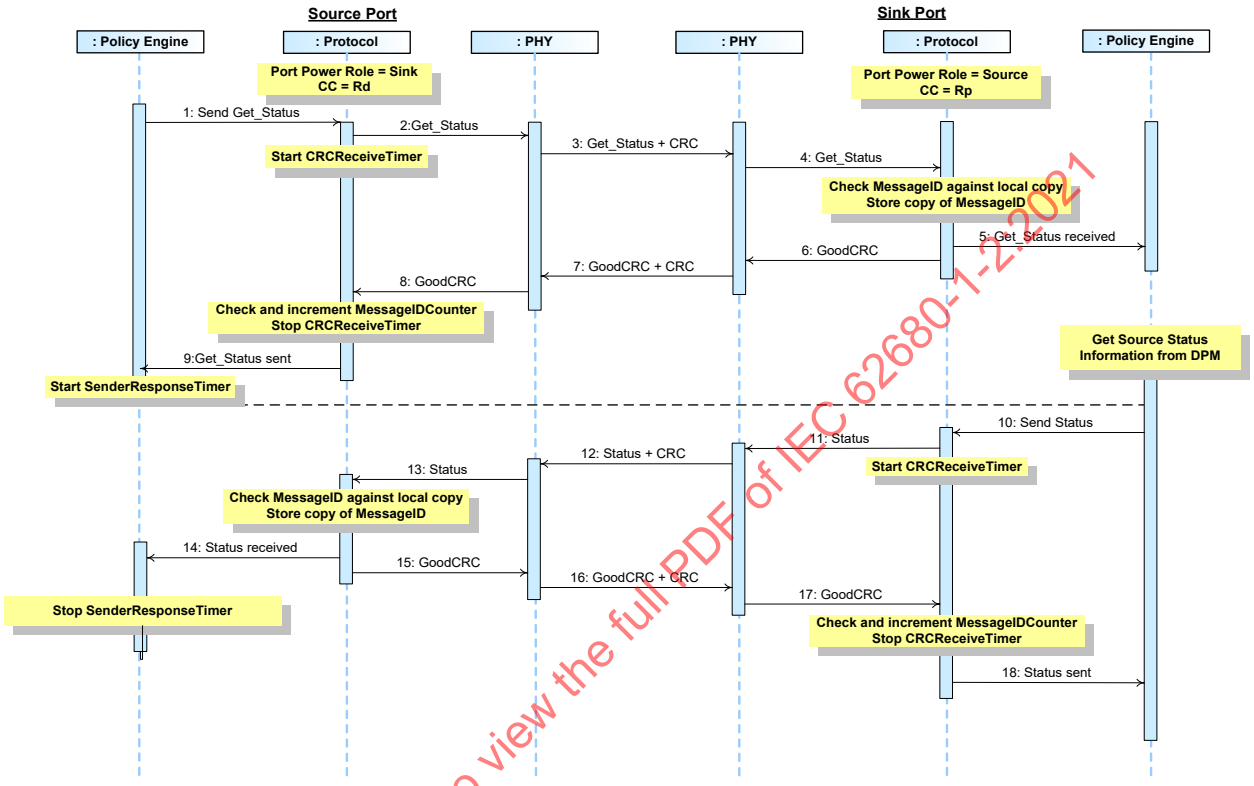
Etape	Port destinataire	Port source
1	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***Get_Status***.	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Get_Status***.	La couche physique reçoit le message ***Get_Status*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Get_Status*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Get_Status*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Get_Status*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande au DPM le statut de la source actuelle qui est fourni. Le moteur de politique demande à la couche protocole de former un message ***Statut***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Statut***.	La couche physique ajoute un CRC et envoie le message ***Statut***.

Etape	Port destinataire	Port source
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Statut*** reçu au moteur de politique qui le consomme.	
14	Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Statut*** a abouti.
	La source a informé le destinataire de son statut actuel.	

8.3.2.10.2.2 Source obtenant le statut du destinataire

La Figure 8-27 donne un exemple de séquence entre une source et un destinataire où, lorsque la source a reçu une alerte (voir 8.3.2.10.1) signalant qu'un changement de statut a eu lieu, la source obtient plus d'informations sur ce changement.

Figure 8-28 Source obtenant le statut du destinataire



Anglais	Français
Sink Port	Port destinataire
Source Port	Port source
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Port Power Role = Source	Rôle d'alimentation du port = source
CC = Rp	CC = Rp
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Get_Status received	Get_Status reçu
Check and increment	Vérifier et incrémenter

Stop	Arrêt
Get_Status sent	Get_Status envoyé
Get Source Status information from DPM	Obtenir les informations relatives au statut de la source auprès du DPM
Send Status	Envoyer statut
Status	Statut
Status + CRC	Statut + CRC
Status received	Statut reçu
Status sent	Statut envoyé

Le Tableau 8-27 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-27 ci-dessus.

Tableau 8-28 Etapes d'une séquence où la source obtient le statut du destinataire

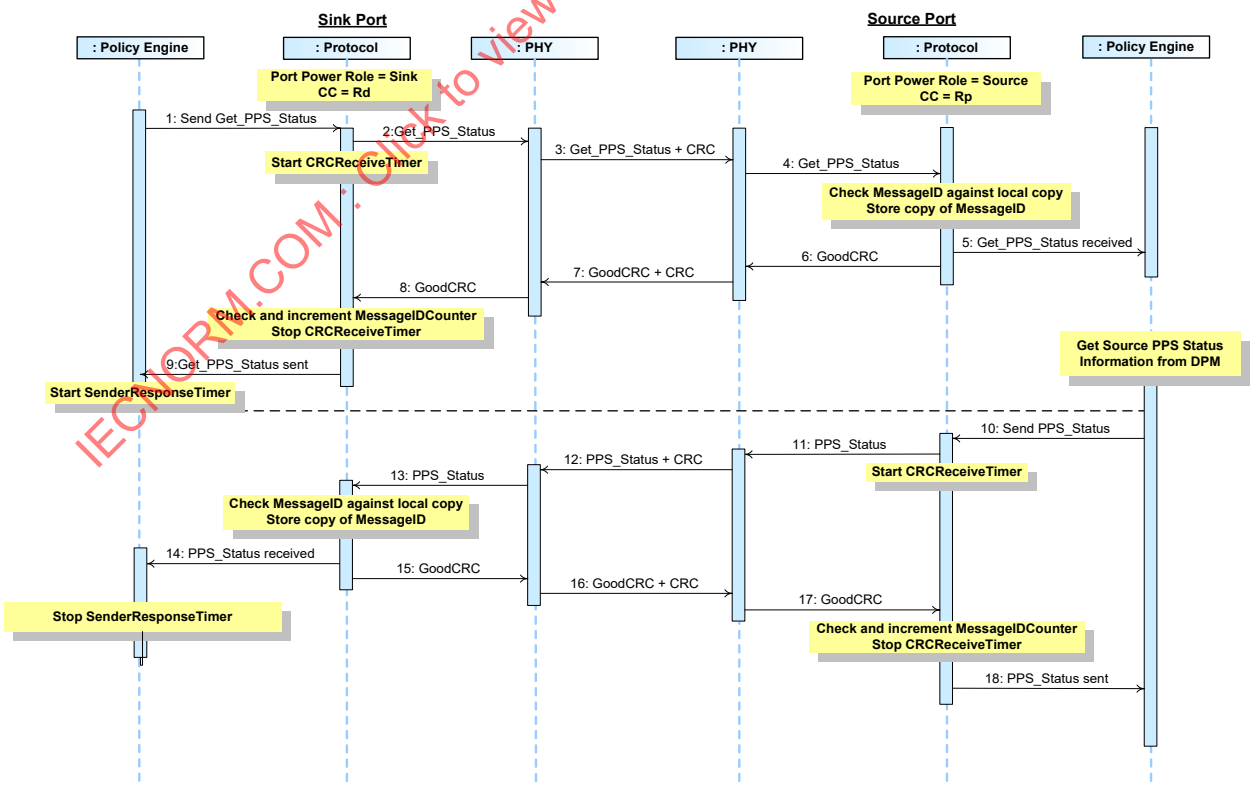
Etape	Port source	Port destinataire
1	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***Get_Status***.	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Get_Status***.	La couche physique reçoit le message ***Get_Status*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Get_Status*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Get_Status*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Get_Status*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande au DPM le statut de la source actuelle qui est fourni. Le moteur de politique demande à la couche protocole de former un message ***Statut***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Statut***.	La couche physique ajoute un CRC et envoie le message ***Statut***.

Etape	Port source	Port destinataire
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Statut*** reçu au moteur de politique qui le consomme.	
14	Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Statut*** a abouti.
	Le destinataire a informé la source de son statut actuel.	

8.3.2.10.2.3 Destinataire obtenant le statut PPS de la source

La Figure 8-29 donne un exemple de séquence entre une source et un destinataire où, lorsque le destinataire a reçu une alerte (voir 8.3.2.10.1) signalant qu'un changement de statut PPS a eu lieu, le destinataire obtient plus d'informations sur ce changement.

Figure 8-29 Destinataire obtenant le statut PPS de la source



Anglais	Français
Sink Port	Port destinataire
Source Port	Port source
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Port Power Role = Source	Rôle d'alimentation du port = source
CC = Rp	CC = Rp
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Get_PPS_Status received	Get_PPS_Status reçu
Check and increment	Vérifier et incrémenter
Stop	Arrêt
Get_PPS_Status sent	Get_PPS_Status envoyé
Get Source PPS Status information from DPM	Obtenir les informations relatives au statut PPS de la source auprès du DPM
PPS_Status received	PPS_Status reçu
PPS_Status sent	PPS_Status envoyé

Le Tableau 8-29 ci-dessous donne une explication précise de ce qu'il se passe à chaque étape indiquée sur la Figure 8-29 ci-dessus.

Tableau 8-29 Etapes d'une séquence où le destinataire obtient le statut PPS de la source

Etape	Port destinataire	Port source
1	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***Get_PPS_Status***.	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Get_PPS_Status***.	La couche physique reçoit le message ***Get_PPS_Status*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Get_Status*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Get_PPS_Status*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.

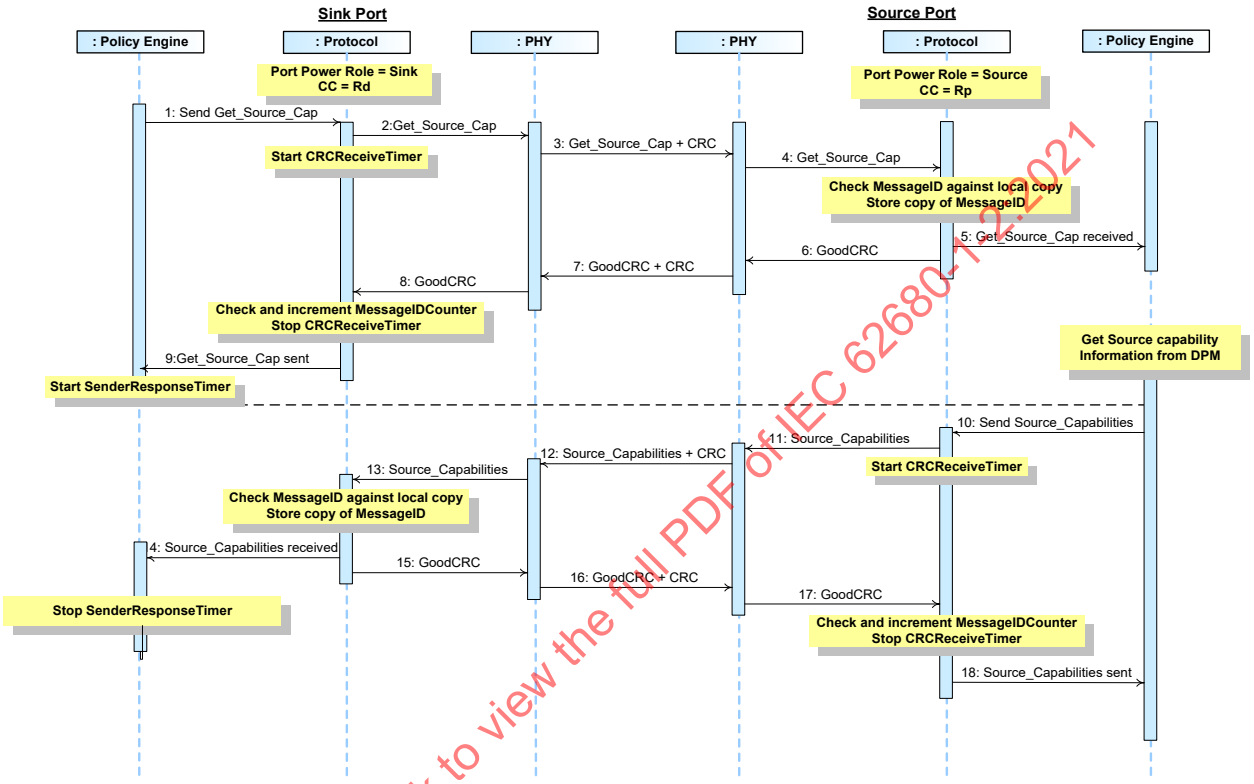
Etape	Port destinataire	Port source
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Get_PPS_Status*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande au DPM le statut de la source actuelle qui est fourni. Le moteur de politique demande à la couche protocole de former un message ***PPS_Status***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***PPS_Status***.	La couche physique ajoute un CRC et envoie le message ***PPS_Status***.
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***PPS_Status*** reçu au moteur de politique qui le consomme.	
14	Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***PPS_Status*** a abouti.
	La source a informé le destinataire de son statut PPS actuel.	

8.3.2.10.3 Capacités de source/destinataire

8.3.2.10.3.1 Destinataire obtenant les capacités de la source

La Figure 8-30 donne un exemple de séquence entre une source et un destinataire, où le destinataire obtient les capacités de la source.

Figure 8-30 Destinataire obtenant les capacités de la source



Anglais	Français
Sink Port	Port destinataire
Source Port	Port source
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Port Power Role = Source	Rôle d'alimentation du port = source
CC = Rp	CC = Rp
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Get_Source_Cap received	Get_Source_Cap reçu

Check and increment	Vérifier et incrémenter
Stop	Arrêt
Get_Source_Cap sent	Get_Source_Cap envoyé
Get Source capability information from DPM	Obtenir les informations relatives aux capacités de la source auprès du DPM
Source_Capabilities received	Source_Capabilities reçu
Source_Capabilities sent	Source_Capabilities envoyé

Le Tableau 8-30 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-30 ci-dessus.

Tableau 8-30 Etapes d'une séquence où le destinataire obtient les capacités de la source

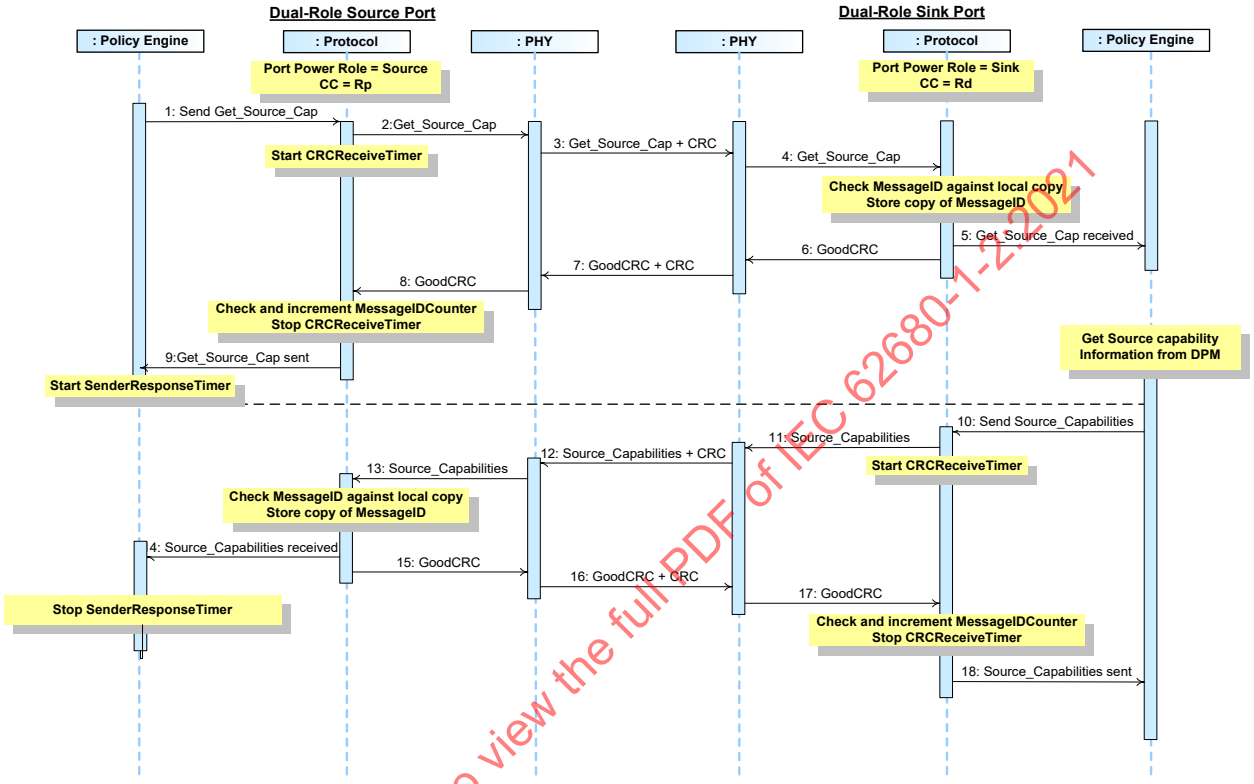
Etape	Port destinataire	Port source
1	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***Get_Source_Cap***.	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Get_Source_Cap***.	La couche physique reçoit le message ***Get_Source_Cap*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Get_Source_Cap*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Get_Source_Cap*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Get_Source_Cap*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande au DPM les capacités de la source actuelle qui sont fournies. Le moteur de politique demande à la couche protocole de former un message ***Source_Capabilities***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Source_Capabilities***.	La couche physique ajoute un CRC et envoie le message ***Source_Capabilities***.
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Source_Capabilities*** reçu au moteur de politique qui le consomme.	

Etape	Port destinataire	Port source
14	Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Source_Capabilities*** a abouti.
	La source a informé le destinataire de ses capacités.	

8.3.2.10.3.2 Source double fonction obtenant les capacités de source d'un destinataire double fonction

La Figure 8-31 donne un exemple de séquence entre une source double fonction et un destinataire double fonction, où la source obtient les capacités du destinataire en tant que source.

Figure 8-31 Source double fonction obtenant les capacités du destinataire double fonction en tant que source



Anglais	Français
Dual-Role Source Port	Port source double fonction
Dual-Role Sink Port	Port destinataire double fonction
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Source	Rôle d'alimentation du port = source
CC = Rp	CC = Rp
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Get_Source_Cap received	Get_Source_Cap reçu
Check and increment	Vérifier et incrémenter

Stop	Arrêt
Get_Source_Cap sent	Get_Source_Cap envoyé
Get Source capability information from DPM	Obtenir les informations relatives aux capacités de la source auprès du DPM
Source_Capabilities received	Source_Capabilities reçu
Source_Capabilities_sent	Source_Capabilities envoyé

Le Tableau 8-31 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-31 ci-dessus.

Tableau 8-31 Etapes d'une séquence où la source double fonction obtient les capacités du destinataire double fonction en tant que source

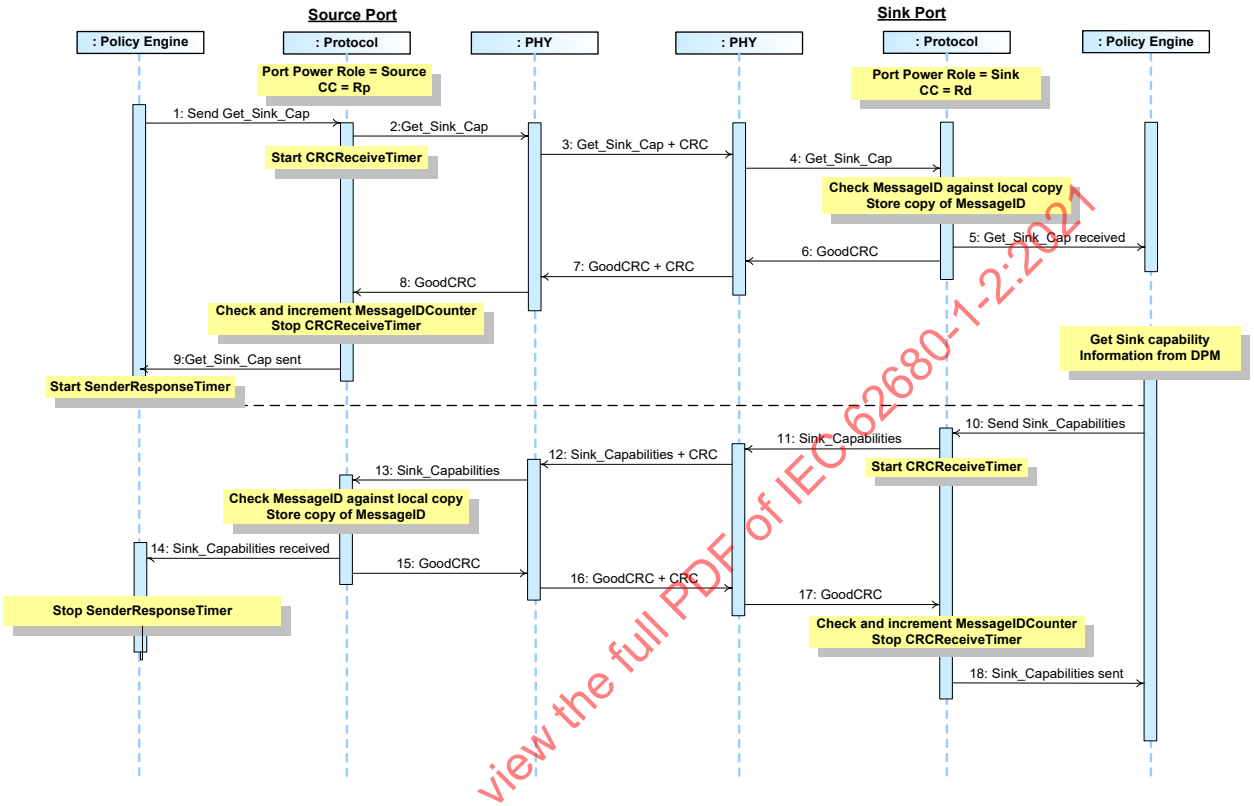
Etape	Port source double fonction	Port destinataire double fonction
1	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***Get_Source_Cap***.	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Get_Source_Cap***.	La couche physique reçoit le message ***Get_Source_Cap*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Get_Source_Cap*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Get_Source_Cap*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Get_Source_Cap*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande au DPM les capacités de la source actuelle qui sont fournies. Le moteur de politique demande à la couche protocole de former un message ***Source_Capabilities***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Source_Capabilities***.	La couche physique ajoute un CRC et envoie le message ***Source_Capabilities***.
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Source_Capabilities*** reçu au moteur de politique qui le consomme.	
14	Le moteur de politique arrête le ***SenderResponseTimer***.	

Etape	Port source double fonction	Port destinataire double fonction
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Source_Capabilities*** a abouti.
	Le destinataire double fonction a informé la source double fonction de ses capacités.	

8.3.2.10.3.3 Source obtenant les capacités du destinataire

La Figure 8-32 donne un exemple de séquence entre une source et un destinataire, où la source obtient les capacités du destinataire.

Figure 8-32 Source obtenant les capacités du destinataire



Anglais	Français
Sink Port	Port destinataire
Source Port	Port source
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Source	Rôle d'alimentation du port = source
CC = Rp	CC = Rp
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Get_Sink_Cap received	Get_Sink_Cap reçu
Check and increment	Vérifier et incrémenter
Stop	Arrêt

Get_Sink_Cap sent	Get_Sink_Cap envoyé
Get Sink capability information from DPM	Obtenir les informations relatives aux capacités du destinataire auprès du DPM
Sink_Capabilities received	Sink_Capabilities reçu
Sink_Capabilities_sent	Sink_Capabilities envoyé

Le Tableau 8-32 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-32 ci-dessus.

Tableau 8-32 Etapes d'une séquence où la source obtient les capacités du destinataire

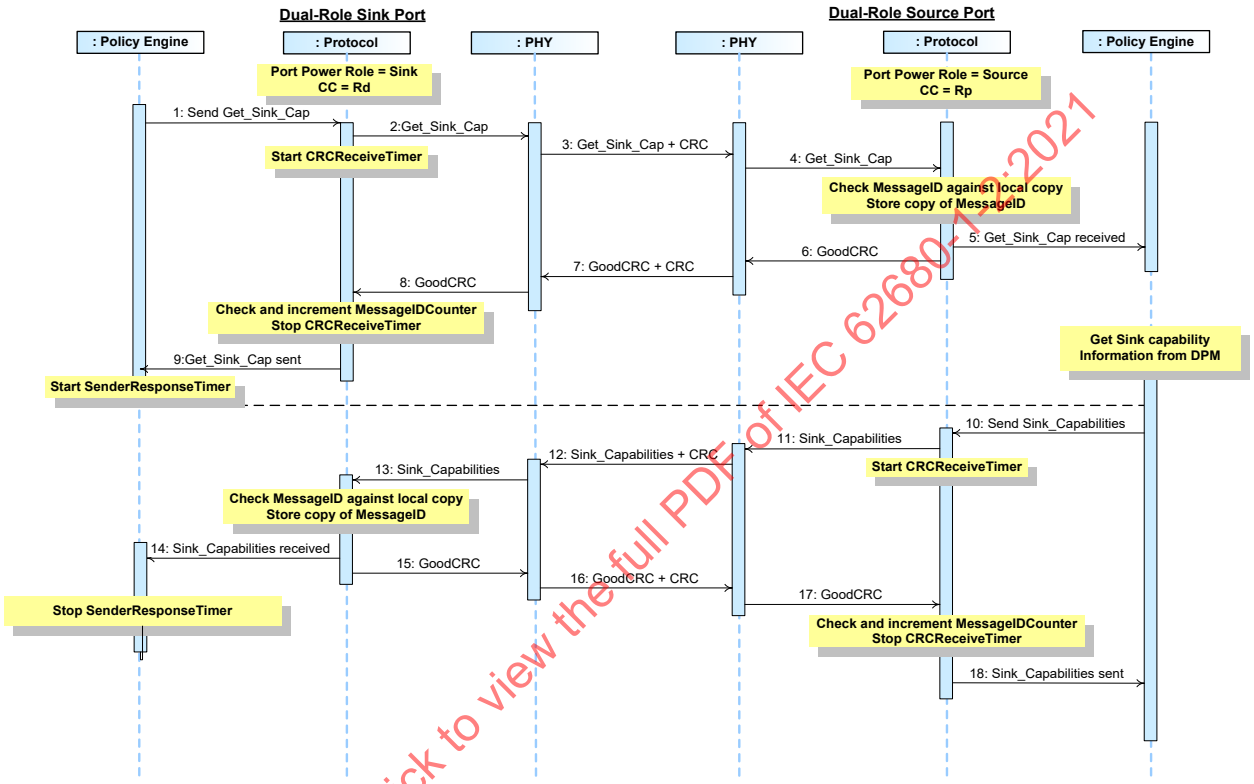
Etape	Port source	Port destinataire
1	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***Get_Sink_Cap***.	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Get_Sink_Cap***.	La couche physique reçoit le message ***Get_Sink_Cap*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Get_Sink_Cap*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Get_Sink_Cap*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Get_Sink_Cap*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande au DPM les capacités du destinataire actuel qui sont fournies. Le moteur de politique demande à la couche protocole de former un message ***Sink_Capabilities***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Sink_Capabilities***.	La couche physique ajoute un CRC et envoie le message ***Sink_Capabilities***.
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Sink_Capabilities*** reçu au moteur de politique qui le consomme.	
14	Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	

Etape	Port source	Port destinataire
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Sink_Capabilities*** a abouti.
	Le destinataire a informé la source de ses capacités.	

8.3.2.10.3.4 Destinataire double fonction obtenant les capacités de destinataire d'une source double fonction

La Figure 8-33 donne un exemple de séquence entre une source double fonction et un destinataire double fonction, où le destinataire double fonction obtient les capacités de la source double en tant que destinataire.

Figure 8-33 Destinataire double fonction obtenant les capacités de la source double fonction en tant que destinataire



Anglais	Français
Dual-Role Source Port	Port source double fonction
Dual-Role Sink Port	Port destinataire double fonction
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Source	Rôle d'alimentation du port = source
CC = Rp	CC = Rp
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Get_Sink_Cap received	Get_Sink_Cap reçu

Check and increment	Vérifier et incrémenter
Stop	Arrêt
Get_Sink_Cap sent	Get_Sink_Cap envoyé
Get Sink capability information from DPM	Obtenir les informations relatives aux capacités du destinataire auprès du DPM
Sink_Capabilities received	Sink_Capabilities reçu
Sink_Capabilities_sent	Sink_Capabilities envoyé

Le Tableau 8-33 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-33 ci-dessus.

Tableau 8-33 Etapes d'une séquence où le destinataire double fonction obtient les capacités de la source double fonction en tant que destinataire

Etape	Port destinataire double fonction	Port source double fonction
1	Le ***Port Power Role*** du port est défini sur "Dual-Role Sink" avec la dépolarisation Rd sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***Get_Sink_Cap***.	Le ***Port Power Role*** du port est défini sur "Dual-Role Source" avec la polarisation Rp sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Get_Sink_Cap***.	La couche physique reçoit le message ***Get_Sink_Cap*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Get_Sink_Cap*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Get_Sink_Cap*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Get_Sink_Cap*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande au DPM les capacités de la source double fonction actuelle qui sont fournies. Le moteur de politique demande à la couche protocole de former un message ***Sink_Capabilities***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Sink_Capabilities***.	La couche physique ajoute un CRC et envoie le message ***Sink_Capabilities***.
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Sink_Capabilities*** reçu au moteur de politique qui le consomme.	

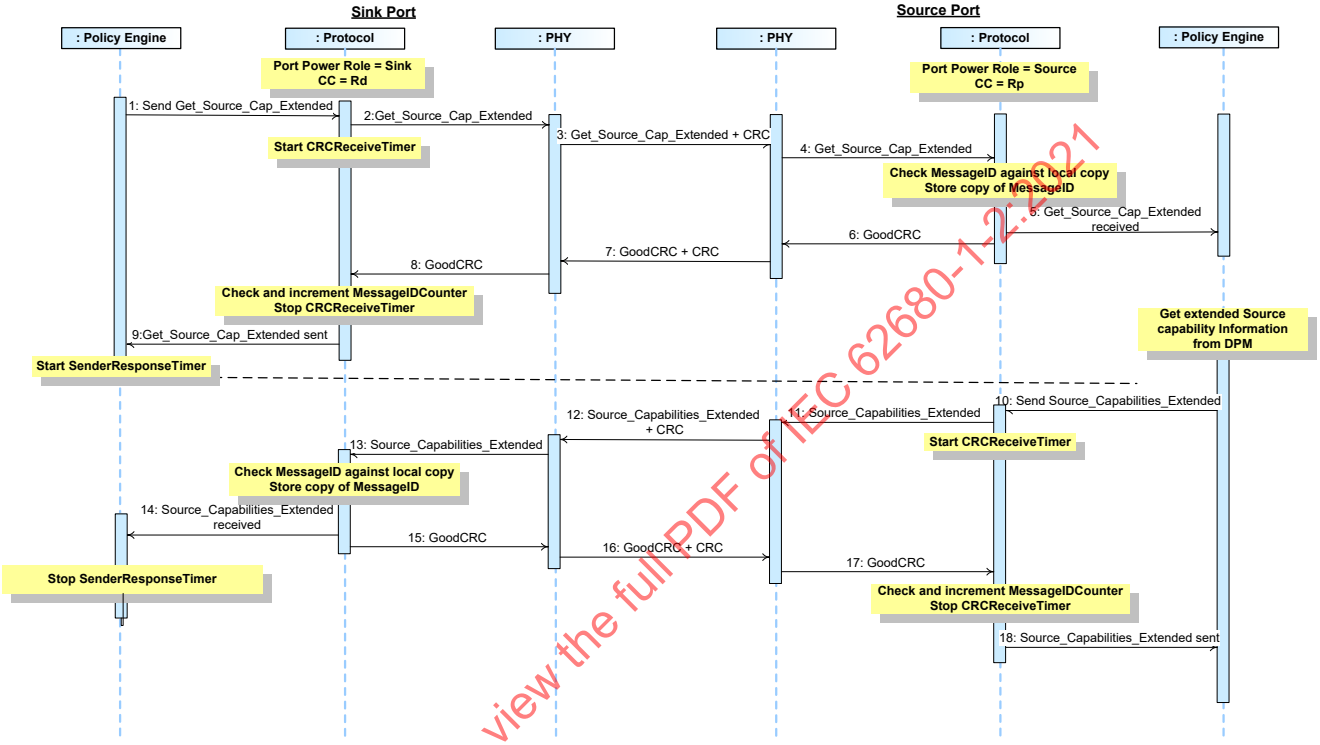
Etape	Port destinataire double fonction	Port source double fonction
14	Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Sink_Capabilities*** a abouti.
	La source double fonction a informé le destinataire double fonction de ses capacités en tant que destinataire.	

8.3.2.10.4 Capacités étendues

8.3.2.10.4.1 Destinataire obtenant les capacités étendues de la source

La Figure 8-34 donne un exemple de séquence entre une source et un destinataire, où le destinataire obtient les capacités étendues de la source.

Figure 8-34 Destinataire obtenant les capacités étendues de la source



Anglais	Français
Sink Port	Port destinataire
Source Port	Port source
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Source	Rôle d'alimentation du port = source
CC = Rp	CC = Rp
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Get_Source_Cap_Extended received	Get_Source_Cap_Extended reçu
Check and increment	Vérifier et incrémenter

Stop	Arrêt
Get_Source_Cap_Extended sent	Get_Source_Cap_Extended envoyé
Get extended Source capability information from DPM	Obtenir les informations relatives aux capacités étendues de la source auprès du DPM
Source_Capabilities_Extended received	Source_Capabilities_Extended reçu
Source_Capabilities_Extended_sent	Source_Capabilities_Extended envoyé

Le Tableau 8-34 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-34 ci-dessus.

Tableau 8-34 Etapes d'une séquence où le destinataire obtient les capacités étendues de la source

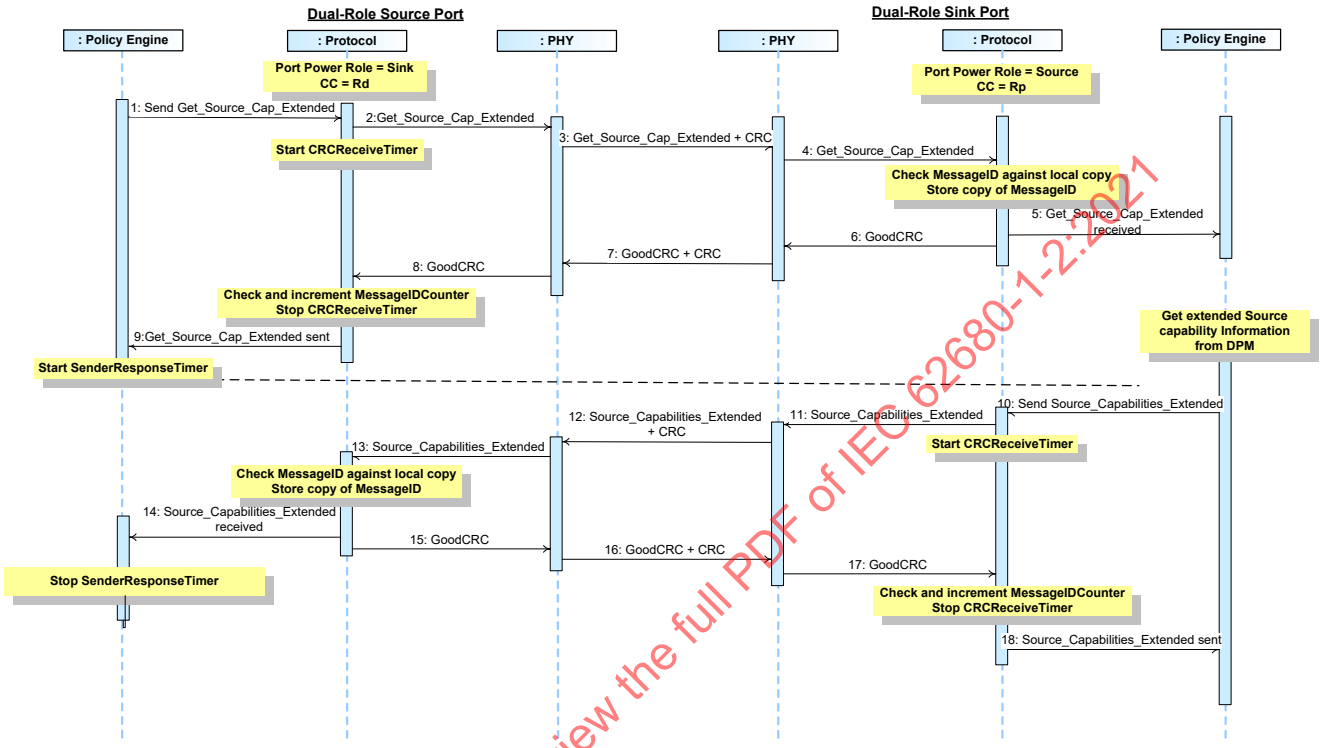
Etape	Port destinataire	Port source
1	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***Get_Source_Cap_Extended***.	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Get_Source_Cap_Extended***.	La couche physique reçoit le message ***Get_Source_Cap_Extended*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Get_Source_Cap_Extended*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Get_Source_Cap_Extended*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Get_Source_Cap_Extended*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande au DPM les capacités étendues de la source actuelle qui sont fournies. Le moteur de politique demande à la couche protocole de former un message ***Source_Capabilities_Extended***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Source_Capabilities_Extended***.	La couche physique ajoute un CRC et envoie le message ***Source_Capabilities_Extended***.

Etape	Port destinataire	Port source
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Source_Capabilities_Extended*** reçu au moteur de politique qui le consomme.	
14	Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Source_Capabilities_Extended*** a abouti.
	La source a informé le destinataire de ses capacités étendues.	

8.3.2.10.4.2 Source double fonction obtenant les capacités étendues de source d'un destinataire double fonction

La Figure 8-35 donne un exemple de séquence entre une source et un destinataire, où la source double fonction obtient les capacités étendues du destinataire double fonction en tant que source.

Figure 8-35 Source double fonction obtenant les capacités étendues du destinataire double fonction



Anglais	Français
Dual-Role Source Port	Port source double fonction
Dual-Role Sink Port	Port destinataire double fonction
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Source	Rôle d'alimentation du port = source
CC = Rp	CC = Rp
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Get_Source_Cap_Extended received	Get_Source_Cap_Extended reçu
Check and increment	Vérifier et incrémenter
Stop	Arrêt

Get_Source_Cap_Extended sent	Get_Source_Cap_Extended envoyé
Get extended Source capability information from DPM	Obtenir les informations relatives aux capacités étendues de la source auprès du DPM
Source_Capabilities_Extended received	Source_Capabilities_Extended reçu
Source_Capabilities_Extended_sent	Source_Capabilities_Extended envoyé

Le Tableau 8-35 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-35 ci-dessus.

Tableau 8-35 Etapes d'une séquence où la source double fonction obtient les capacités étendues du destinataire double fonction

Etape	Port source double fonction	Port destinataire double fonction
1	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***Get_Source_Cap_Extended***.	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Get_Source_Cap_Extended***.	La couche physique reçoit le message ***Get_Source_Cap_Extended*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Get_Source_Cap_Extended*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Get_Source_Cap_Extended*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Get_Source_Cap_Extended*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande au DPM les capacités étendues de la source actuelle qui sont fournies. Le moteur de politique demande à la couche protocole de former un message ***Source_Capabilities_Extended***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Source_Capabilities_Extended***.	La couche physique ajoute un CRC et envoie le message ***Source_Capabilities_Extended***.

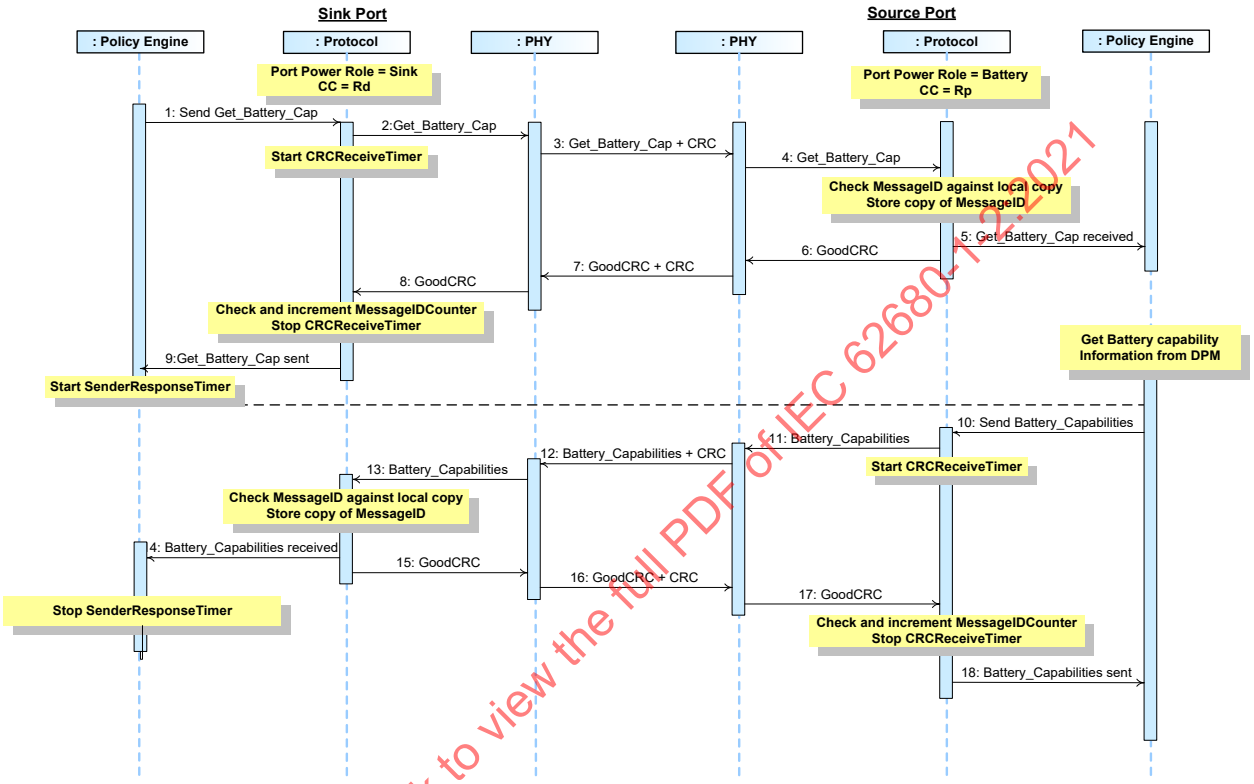
Etape	Port source double fonction	Port destinataire double fonction
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Source_Capabilities_Extended*** reçu au moteur de politique qui le consomme.	
14	Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Source_Capabilities_Extended*** a abouti.
	Le destinataire double fonction a informé la source double fonction de ses capacités étendues en tant que source.	

8.3.2.10.5 Capacités et statut de la batterie

8.3.2.10.5.1 Destinataire obtenant les capacités de la batterie

La Figure 8-36 donne un exemple de séquence entre une source et un destinataire, où le destinataire obtient les capacités de batterie de la source, pour une batterie donnée.

Figure 8-36 Destinataire obtenant les capacités de la batterie de la source



Anglais	Français
Sink Port	Port destinataire
Source Port	Port source
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Battery	Rôle d'alimentation du port = batterie
CC = Rp	CC = Rp
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Get_Battery_Cap received	Get_Battery_Cap reçu

Check and increment	Vérifier et incrémenter
Stop	Arrêt
Get_Battery_Cap sent	Get_Battery_Cap envoyé
Get Battery capability information from DPM	Obtenir les informations relatives aux capacités de la batterie auprès du DPM
Battery_Capabilities received	Battery_Capabilities reçu
Battery_Capabilities_sent	Battery_Capabilities envoyé

Le Tableau 8-36 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-36 ci-dessus.

Tableau 8-36 Etapes d'une séquence où le destinataire obtient les capacités de la batterie de la source

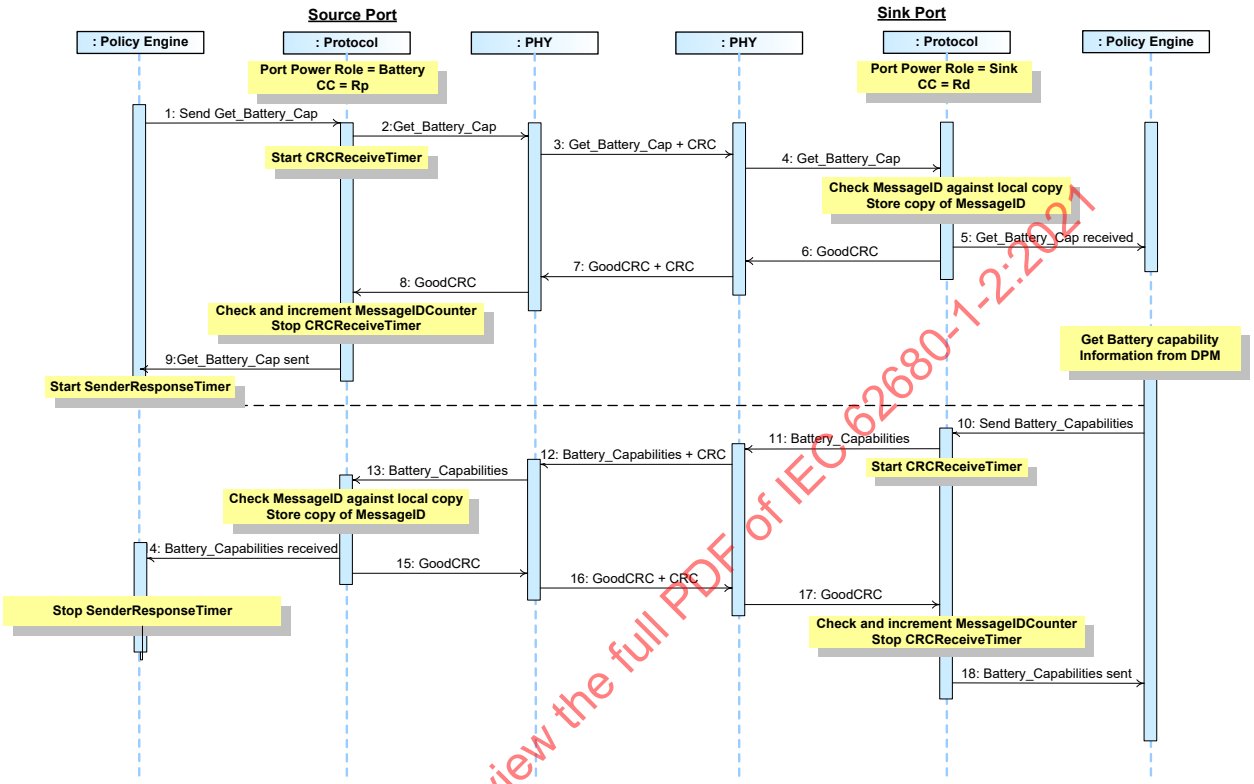
Etape	Port destinataire	Port source
1	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***Get_Battery_Cap*** contenant le numéro de la batterie dont les capacités sont demandées.	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Get_Battery_Cap***.	La couche physique reçoit le message ***Get_Battery_Cap*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Get_Battery_Cap*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Get_Battery_Cap*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Get_Battery_Cap*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande au DPM les capacités de batterie de la source actuelle, correspondant au numéro de batterie demandé, qui sont fournies. Le moteur de politique demande à la couche protocole de former un message ***Battery_Capabilities***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Battery_Capabilities***.	La couche physique ajoute un CRC et envoie le message ***Battery_Capabilities***.

Etape	Port destinataire	Port source
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Battery_Capabilities*** reçu au moteur de politique qui le consomme.	
14	Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Battery_Capabilities*** a abouti.
	La source a informé le destinataire des capacités de batterie de la batterie demandée.	

8.3.2.10.5.2 Source obtenant les capacités de la batterie

La Figure 8-37 donne un exemple de séquence entre une source et un destinataire, où la source obtient les capacités de batterie du destinataire, pour une batterie donnée.

Figure 8-37 Source obtenant les capacités de la batterie du destinataire



Anglais	Français
Sink Port	Port destinataire
Source Port	Port source
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Battery	Rôle d'alimentation du port = batterie
CC = Rp	CC = Rp
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Get_Battery_Cap received	Get_Battery_Cap reçu
Check and increment	Vérifier et incrémenter
Stop	Arrêt

Get_Battery_Cap sent	Get_Battery_Cap envoyé
Get Battery capability information from DPM	Obtenir les informations relatives aux capacités de la batterie auprès du DPM
Battery_Capabilities received	Battery_Capabilities reçu
Battery_Capabilities_sent	Battery_Capabilities envoyé

Le Tableau 8-37 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-37 ci-dessus.

Tableau 8-37 Etapes d'une séquence où la source obtient les capacités de la batterie du destinataire

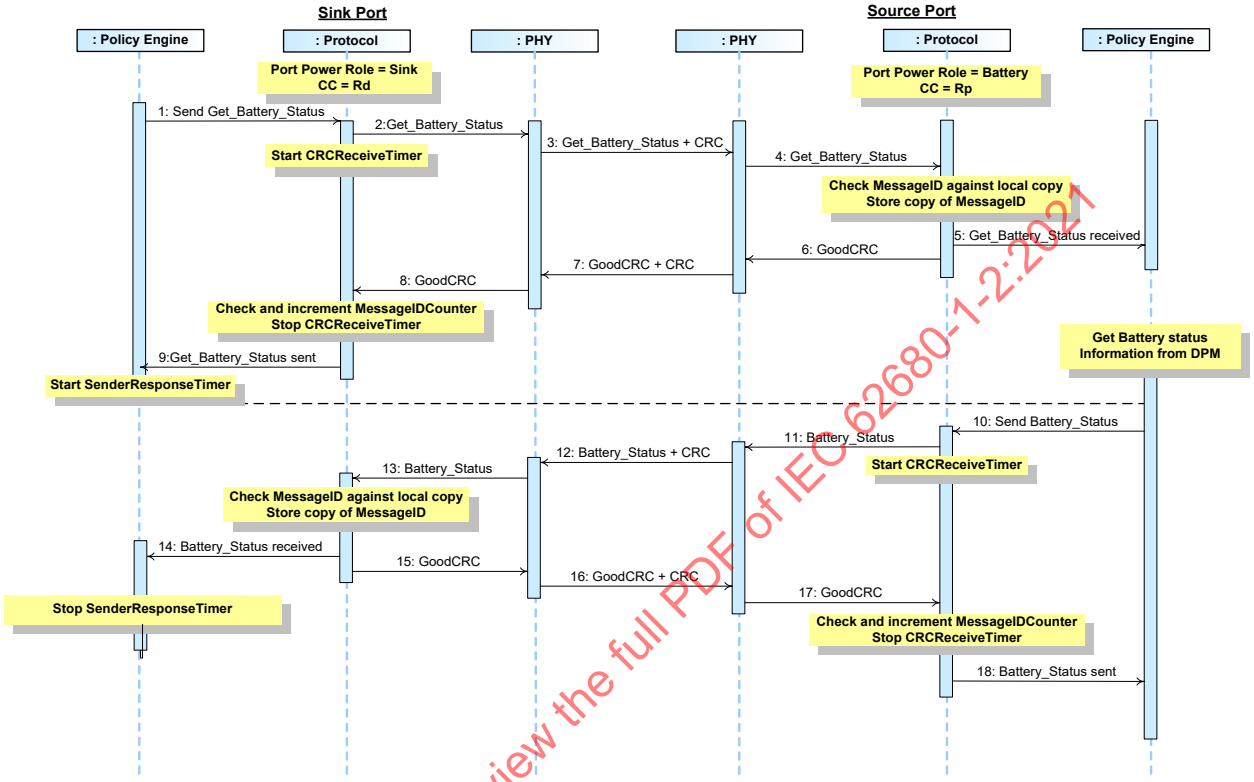
Etape	Port source	Port destinataire
1	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***Get_Battery_Cap*** contenant le numéro de la batterie dont les capacités sont demandées.	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Get_Battery_Cap***.	La couche physique reçoit le message ***Get_Battery_Cap*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Get_Battery_Cap*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Get_Battery_Cap*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Get_Battery_Cap*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande au DPM les capacités de batterie de la source actuelle, correspondant au numéro de batterie demandé, qui sont fournies. Le moteur de politique demande à la couche protocole de former un message ***Battery_Capabilities***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Battery_Capabilities***.	La couche physique ajoute un CRC et envoie le message ***Battery_Capabilities***.
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Battery_Capabilities*** reçu au moteur de politique qui le consomme.	

Etape	Port source	Port destinataire
14	Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Battery_Capabilities*** a abouti.
	Le destinataire a informé la source des capacités de batterie pour la batterie demandée.	

8.3.2.10.5.3 Destinataire obtenant le statut de la batterie

La Figure 8-38 donne un exemple de séquence entre une source et un destinataire, où le destinataire obtient le statut de batterie de la source, pour une batterie donnée.

Figure 8-38 Destinataire obtenant le statut de la batterie de la source



Anglais	Français
Sink Port	Port destinataire
Source Port	Port source
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Battery	Rôle d'alimentation du port = batterie
CC = Rp	CC = Rp
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Get_Battery_Status received	Get_Battery_Status reçu
Check and increment	Vérifier et incrémenter
Stop	Arrêt

Get_Battery_Status sent	Get_Battery_Status envoyé
Get Battery status information from DPM	Obtenir les informations relatives au statut de la batterie auprès du DPM
Battery_Status received	Battery_Status reçu
Battery_Status_sent	Battery_Status envoyé

Le Tableau 8-38 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-38 ci-dessus.

Tableau 8-38 Etapes d'une séquence où le destinataire obtient le statut de la batterie de la source

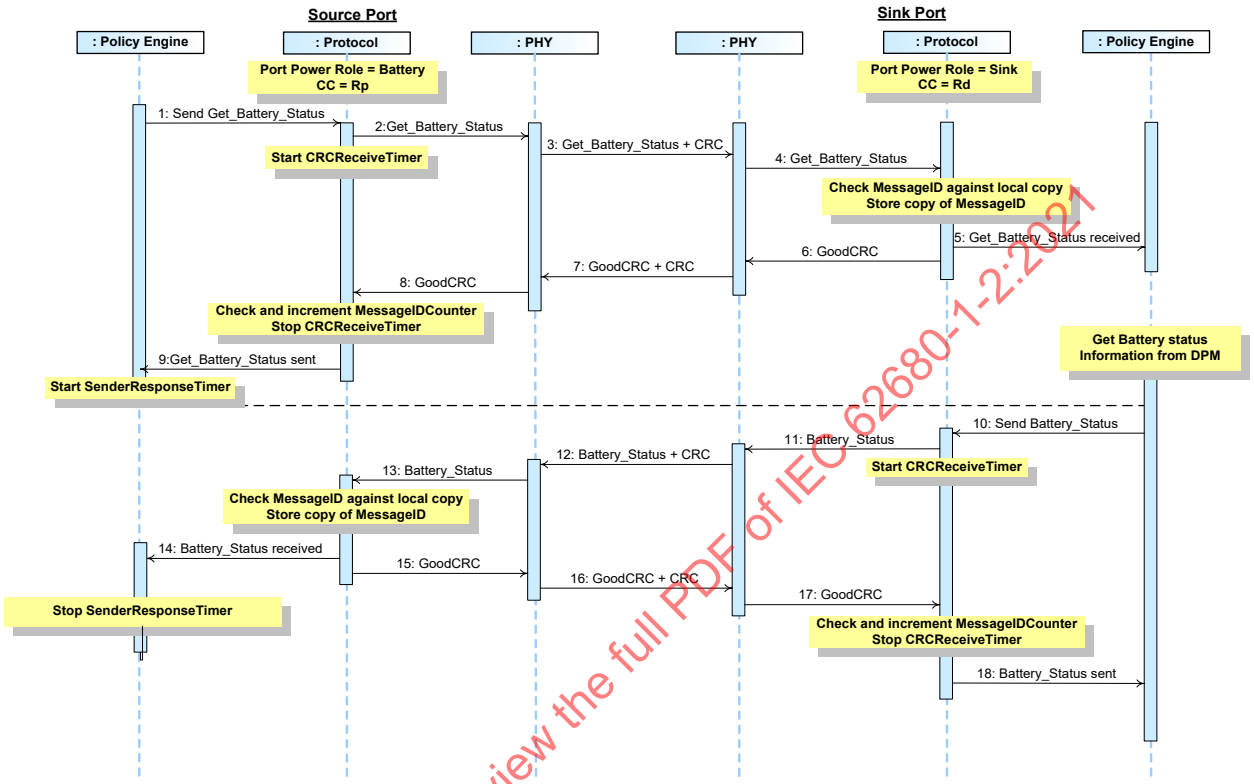
Etape	Port destinataire	Port source
1	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***Get_Battery_Status*** contenant le numéro de la batterie dont le statut est demandé.	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Get_Battery_Status***.	La couche physique reçoit le message ***Get_Battery_Status*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Get_Battery_Status*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Get_Battery_Status*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Get_Battery_Status*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande au DPM le statut de batterie source actuelle, correspondant au numéro de batterie demandé, qui est fourni. Le moteur de politique demande à la couche protocole de former un message ***Battery_Status***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Battery_Status***.	La couche physique ajoute un CRC et envoie le message ***Battery_Status***.
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Battery_Status*** reçu au moteur de politique qui le consomme.	

Etape	Port destinataire	Port source
14	Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Battery_Status*** a abouti.
	La source a informé le destinataire du statut de batterie de la batterie demandée.	

8.3.2.10.5.4 Source obtenant le statut de la batterie

La Figure 8-39 donne un exemple de séquence entre une source et un destinataire, où la source obtient le statut de batterie du destinataire, pour une batterie donnée.

Figure 8-39 Source obtenant le statut de la batterie du destinataire



Anglais	Français
Sink Port	Port destinataire
Source Port	Port source
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Battery	Rôle d'alimentation du port = batterie
CC = Rp	CC = Rp
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Get_Battery_Status received	Get_Battery_Status reçu
Check and increment	Vérifier et incrémenter
Stop	Arrêt

Get_Battery_Status sent	Get_Battery_Status envoyé
Get Battery status information from DPM	Obtenir les informations relatives au statut de la batterie auprès du DPM
Battery_Status received	Battery_Status reçu
Battery_Status_sent	Battery_Status envoyé

Le Tableau 8-39 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-39 ci-dessus.

Tableau 8-39 Etapes d'une séquence où la source obtient le statut de la batterie du destinataire

Etape	Port source	Port destinataire
1	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***Get_Battery_Status*** contenant le numéro de la batterie dont le statut est demandé.	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Get_Battery_Status***.	La couche physique reçoit le message ***Get_Battery_Status*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Get_Battery_Status*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Get_Battery_Status*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Get_Battery_Status*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande au DPM le statut de batterie source actuelle, correspondant au numéro de batterie demandé, qui est fourni. Le moteur de politique demande à la couche protocole de former un message ***Battery_Status***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Battery_Status***.	La couche physique ajoute un CRC et envoie le message ***Battery_Status***.
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Battery_Status*** reçu au moteur de politique qui le consomme.	

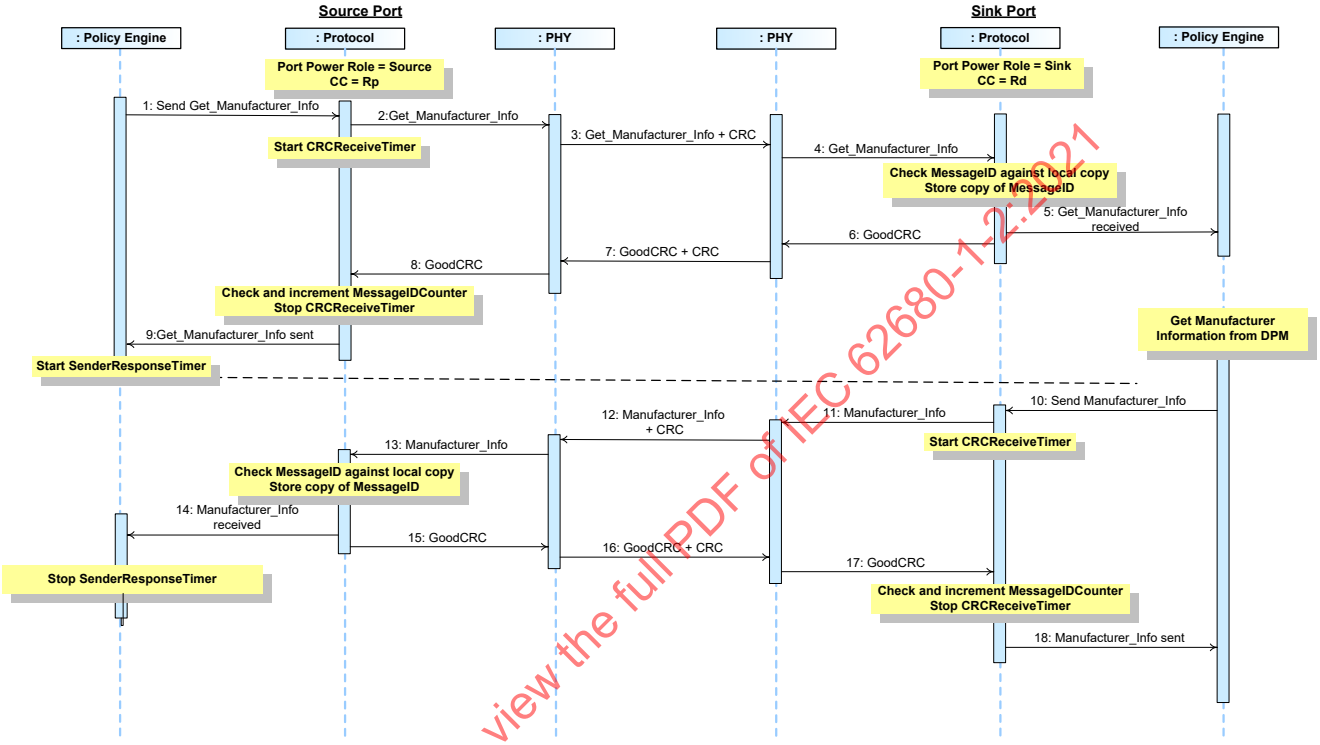
Etape	Port source	Port destinataire
14	Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Battery_Status*** a abouti.
	Le destinataire a informé la source du statut de batterie pour la batterie demandée.	

8.3.2.10.6 Informations du fabricant

8.3.2.10.6.1 Source obtenant les informations du fabricant du port d'un destinataire

La Figure 8-40 donne un exemple de séquence entre une source et un destinataire, où la source obtient les informations du fabricant pour le port.

Figure 8-40 Source obtenant les informations du fabricant du port du destinataire



Anglais	Français
Sink Port	Port destinataire
Source Port	Port source
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Source	Rôle d'alimentation du port = source
CC = Rp	CC = Rp
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Get_Manufacturer_Info received	Get_Manufacturer_Info reçu
Check and increment	Vérifier et incrémenter

Stop	Arrêt
Get_Manufacturer_Info sent	Get_Manufacturer_Info envoyé
Get Manufacturer information from DPM	Obtenir les informations du fabricant auprès du DPM
Manufacturer_Info received	Manufacturer_Info reçu
Manufacturer_Info_sent	Manufacturer_Info envoyé

Le Tableau 8-40 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-40 ci-dessus.

Tableau 8-40 Etapes d'une séquence où la source obtient les informations du fabricant du port du destinataire

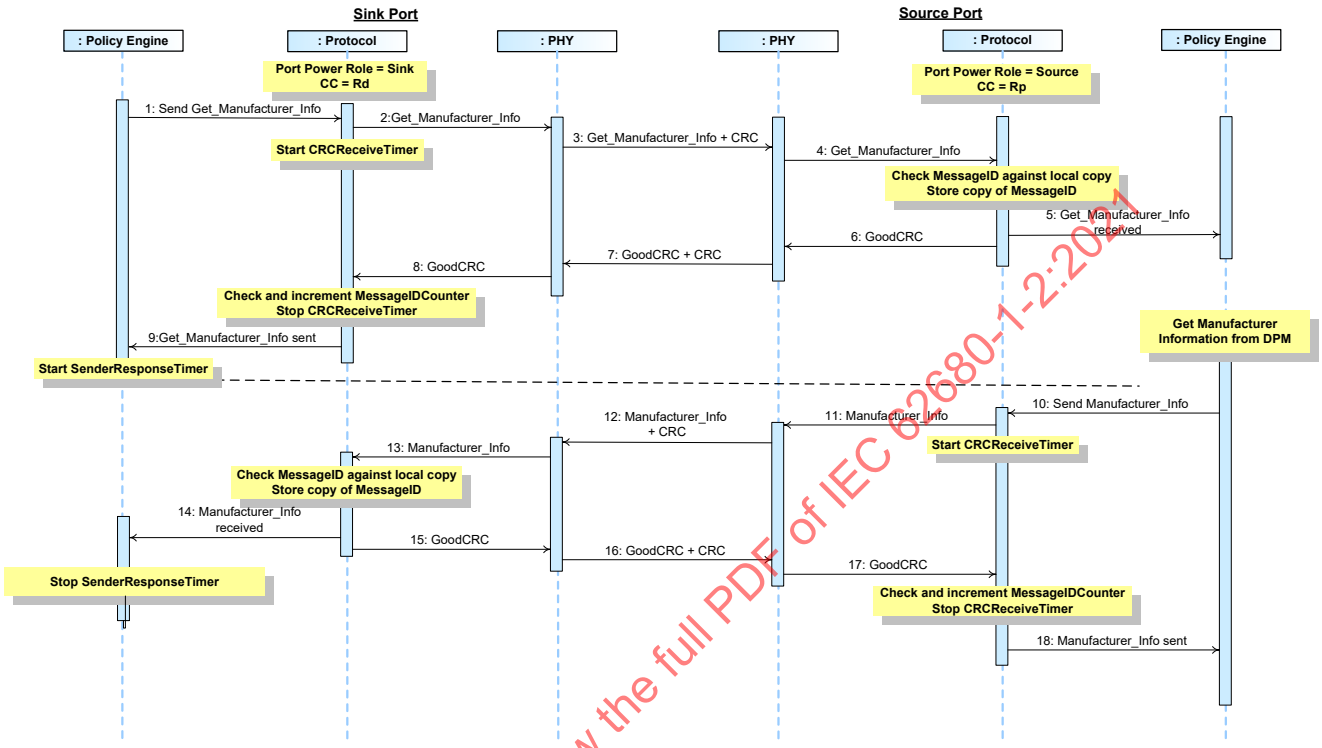
Etape	Port source	Port destinataire
1	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***Get_Manufacturer_Info*** avec une demande d'informations sur le port.	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Get_Manufacturer_Info***.	La couche physique reçoit le message ***Get_Manufacturer_Info*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Get_Manufacturer_Info*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Get_Manufacturer_Info*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Get_Manufacturer_Info*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande au DPM les informations du fabricant du port qui sont fournies. Le moteur de politique demande à la couche protocole de former un message ***Manufacturer_Info***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Manufacturer_Info***.	La couche physique ajoute un CRC et envoie le message ***Manufacturer_Info***.

Etape	Port source	Port destinataire
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Manufacturer_Info*** reçu au moteur de politique qui le consomme.	
14	Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Manufacturer_Info*** a abouti.
	Le destinataire a informé la source des informations du fabricant pour le port.	

8.3.2.10.6.2 Destinataire obtenant les informations du fabricant du port d'une source

La Figure 8-41 donne un exemple de séquence entre une source et un destinataire, où la source obtient les informations du fabricant pour le port.

Figure 8-41 Destinataire obtenant les informations du fabricant du port de la source



Anglais	Français
Sink Port	Port destinataire
Source Port	Port source
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Source	Rôle d'alimentation du port = source
CC = Rp	CC = Rp
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Get_Manufacturer_Info received	Get_Manufacturer_Info reçu
Check and increment	Vérifier et incrémenter
Stop	Arrêt
Get_Manufacturer_Info sent	Get_Manufacturer_Info envoyé

Get Manufacturer information from DPM	Obtenir les informations du fabricant auprès du DPM
Manufacturer_Info received	Manufacturer_Info reçu
Manufacturer_Info_sent	Manufacturer_Info envoyé

Le Tableau 8-41 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-41 ci-dessus.

Tableau 8-41 Etapes d'une séquence où la source obtient les informations du fabricant du port du destinataire

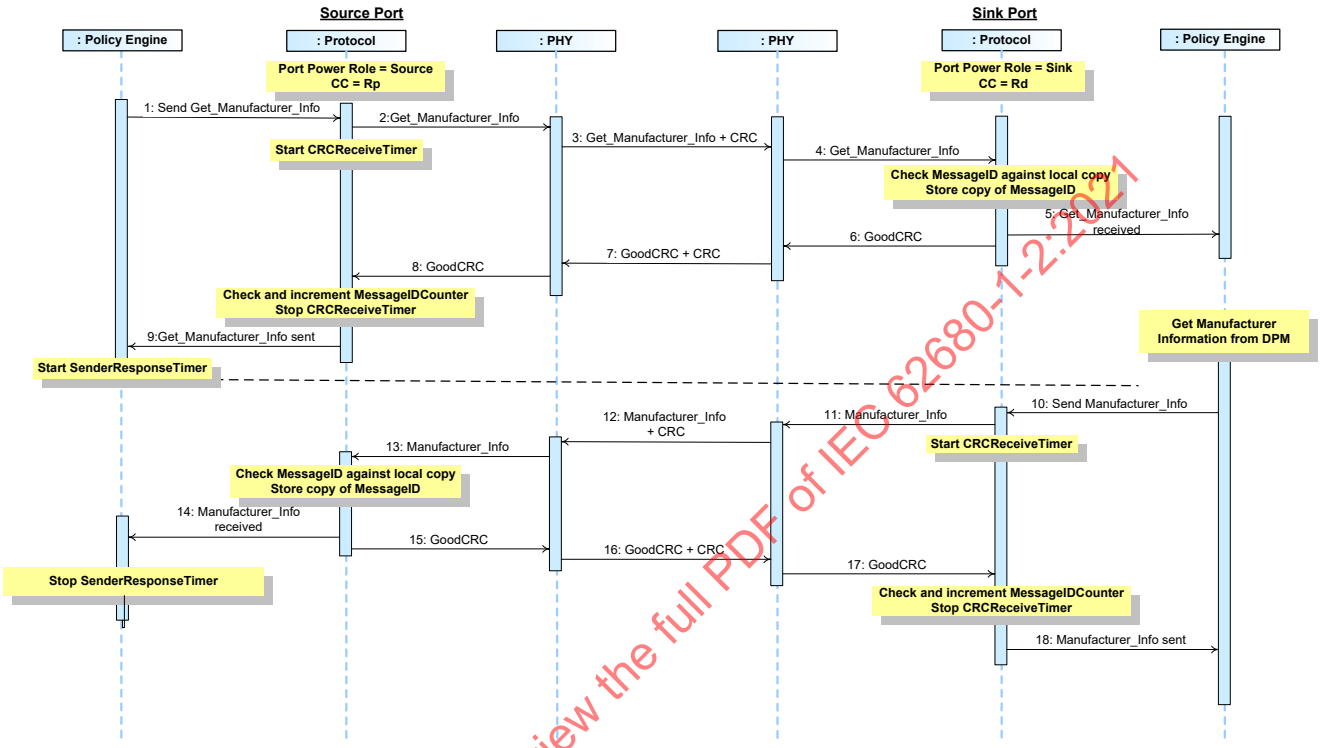
Etape	Port destinataire	Port source
1	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***Get_Manufacturer_Info*** avec une demande d'informations sur le port.	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Get_Manufacturer_Info***.	La couche physique reçoit le message ***Get_Manufacturer_Info*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Get_Manufacturer_Info*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Get_Manufacturer_Info*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter*** et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Get_Manufacturer_Info*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande au DPM les informations du fabricant du port qui sont fournies. Le moteur de politique demande à la couche protocole de former un message ***Manufacturer_Info***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Manufacturer_Info***.	La couche physique ajoute un CRC et envoie le message ***Manufacturer_Info***.
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Manufacturer_Info*** reçu au moteur de politique qui le consomme.	

Etape	Port destinataire	Port source
14	Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Manufacturer_Info*** a abouti.
	Le destinataire a informé la source des informations du fabricant pour le port.	

8.3.2.10.6.3 Source obtenant les informations du fabricant de la batterie auprès d'un destinataire

La Figure 8-42 donne un exemple de séquence entre une source et un destinataire, où la source obtient les informations du fabricant du destinataire pour une de ses batteries.

Figure 8-42 Source obtenant les informations du fabricant de la batterie du destinataire



Anglais	Français
Sink Port	Port destinataire
Source Port	Port source
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Source	Rôle d'alimentation du port = source
CC = Rp	CC = Rp
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Get_Manufacturer_Info received	Get_Manufacturer_Info reçu
Check and increment	Vérifier et incrémenter
Stop	Arrêt

Get_Manufacturer_Info sent	Get_Manufacturer_Info envoyé
Get Manufacturer information from DPM	Obtenir les informations du fabricant auprès du DPM
Manufacturer_Info received	Manufacturer_Info reçu
Manufacturer_Info_sent	Manufacturer_Info envoyé

Le Tableau 8-42 ci-dessous donne une explication précise de ce qu'il se passe à chacune des étapes indiquées sur la Figure 8-42 ci-dessus.

Tableau 8-42 Etapes d'une séquence où la source obtient les informations du fabricant de la batterie du destinataire

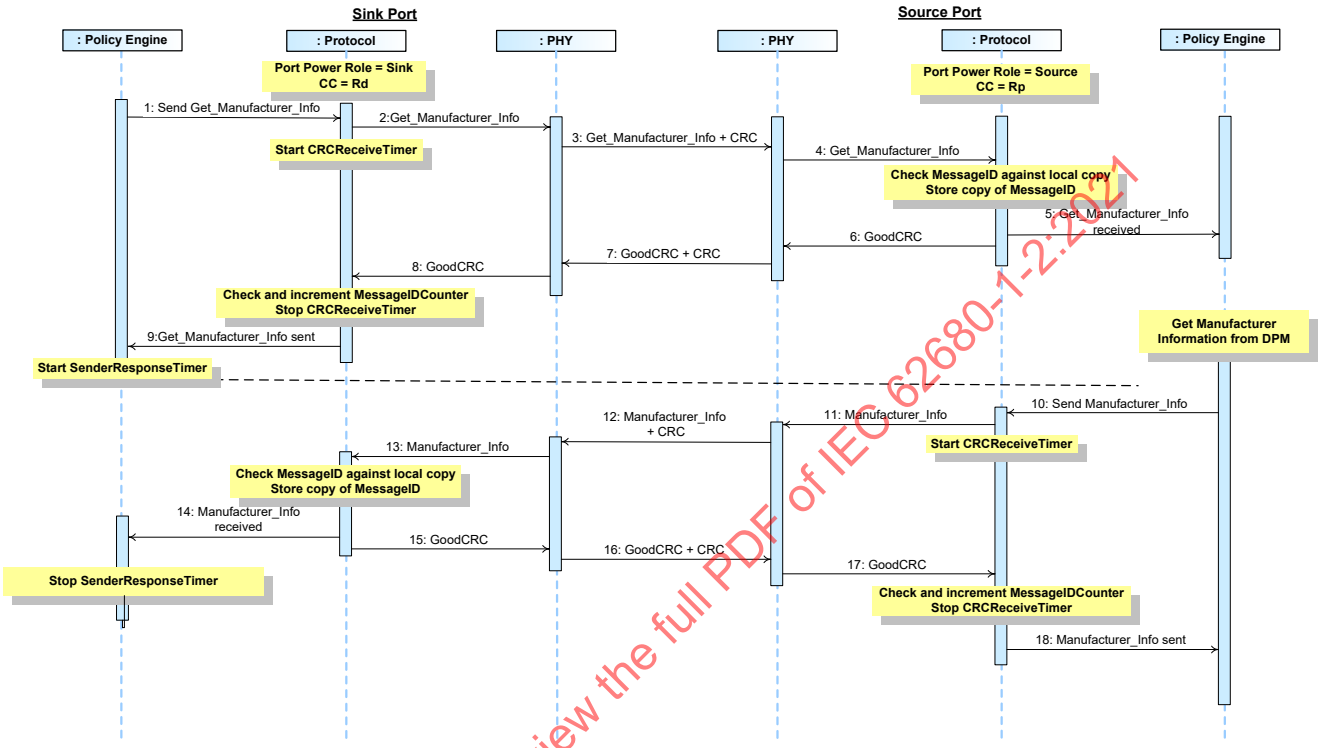
Etape	Port source	Port destinataire
1	Le ***Port Power Role*** du port est défini sur "Source" avec la polarisation Rp sur son fil CC. Le moteur de politique demande à la couche protocole d'envoyer un message ***Get_Manufacturer_Info*** avec une demande d'informations de batterie, pour une batterie donnée.	Le ***Port Power Role*** du port est défini sur "Sink" avec la dépolarisation Rd sur son fil CC.
2	La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.	
3	La couche physique ajoute un CRC et envoie le message ***Get_Manufacturer_Info***.	La couche physique reçoit le message ***Get_Manufacturer_Info*** et contrôle le CRC pour vérifier le message.
4		La couche physique supprime le CRC et transfère le message ***Get_Manufacturer_Info*** vers la couche protocole.
5		La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Get_Manufacturer_Info*** reçu au moteur de politique qui le consomme.
6		La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.
7	La couche physique reçoit le message ***GoodCRC*** et contrôle le CRC pour vérifier le message.	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.
8	La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.	
9	La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Get_Manufacturer_Info*** a abouti. Le moteur de politique lance le ***SenderResponseTimer***.	
10		Le moteur de politique demande au DPM les informations du fabricant de la batterie donnée qui sont fournies. Le moteur de politique demande à la couche protocole de former un message ***Manufacturer_Info***.
11		La couche protocole crée le message et le transmet à la couche physique. Elle lance le ***CRCReceiveTimer***.
12	La couche physique reçoit le message et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message ***Manufacturer_Info***.	La couche physique ajoute un CRC et envoie le message ***Manufacturer_Info***.

Etape	Port source	Port destinataire
13	La couche protocole vérifie que le ***MessageID*** du message entrant est différent de la valeur déjà stockée, puis stocke une copie de la nouvelle valeur. La couche protocole transfère les informations du message ***Manufacturer_Info*** reçu au moteur de politique qui le consomme.	
14	Le moteur de politique arrête le ***SenderResponseTimer***.	
15	La couche protocole génère un message ***GoodCRC*** et le transmet à la couche physique.	
16	La couche physique ajoute un CRC et envoie le message ***GoodCRC***.	La couche physique reçoit le message ***GoodCRC*** et compare le CRC qu'elle a calculé à celui envoyé pour vérifier le message.
17		La couche physique supprime le CRC et transfère le message ***GoodCRC*** vers la couche protocole.
18		La couche protocole vérifie et incrémente le ***MessageIDCounter***, et arrête le ***CRCReceiveTimer***. La couche protocole informe le moteur de politique que l'envoi du message ***Manufacturer_Info*** a abouti.
	Le destinataire a informé la source des informations du fabricant pour la batterie demandée.	

8.3.2.10.6.4 Destinataire obtenant les informations du fabricant de la batterie auprès d'une source

La Figure 8-43 donne un exemple de séquence entre une source et un destinataire, où la source obtient les informations du fabricant pour le port.

Figure 8-43 Destinataire obtenant les informations du fabricant de la batterie de la source



Anglais	Français
Sink Port	Port destinataire
Source Port	Port source
Policy Engine	Moteur de politique
Protocol	Protocole
PHY	PHY
Port Power Role = Source	Rôle d'alimentation du port = source
CC = Rp	CC = Rp
Port Power Role = Sink	Rôle d'alimentation du port = destinataire
CC = Rd	CC = Rd
Send	Envoyer
CRC	CRC
Start	Début
Check MessageID against local copy	Vérifier le MessageID par rapport à la copie locale
Store copy of MessageID	Stocker une copie du MessageID
Get_Manufacturer_Info received	Get_Manufacturer_Info reçu
Check and increment	Vérifier et incrémenter
Stop	Arrêt