Power Delivery Tests

Version: 1.02
Release date: May 7, 2018
# Table of Contents

**Editor** ........................................................................................................................................ 2

**Contributors** .................................................................................................................................. 2

**Table of Contents** ............................................................................................................................... 3

**Copyright and Intellectual Property** .................................................................................................. 6

**Revisions History** .............................................................................................................................. 7

**Terms and Abbreviations** .................................................................................................................. 9

**Test Results** ....................................................................................................................................... 10

**Related Documents** .......................................................................................................................... 11

**Test Procedures** ................................................................................................................................. 12

PROC.PD.E1. Bring-up procedure .......................................................... 12
PROC.PD.E2. Bring-up procedure for PPS Tests ..................................... 13
PROC.PD.E3. Wait to Start AMS ............................................................ 14

**Assertions** .......................................................................................................................................... 15

New or changed assertions in PD3 .......................................................... 15
Common assertions to all tests ............................................................... 44

**Tests Applicability** ................................................................................................................................ 45

**Link Layer Tests** .................................................................................................................................. 47

TD. PD. LL3. E2. Retransmission ............................................................ 47

**Source Tests** ....................................................................................................................................... 49

TD. PD. SRC3. E1. Source Capabilities Fields Checks ......................... 49
TD. PD. SRC3. E2. Accept Fields Checks ............................................. 50
TD. PD. SRC3. E3. PS_RDY Fields Checks ........................................... 50
TD. PD. SRC3. E4. Specification Revision Check after Contract ............ 51
TD. PD. SRC3. E5. Source_Capabilities_Extended sent timely ............. 52
TD. PD. SRC3. E7. Battery Status sent timely ...................................... 54
TD. PD. SRC3. E8. Battery Status Fields Checks .................................. 54
TD. PD. SRC3. E10. Unrecognized Message Received in Ready State .... 56
TD. PD. SRC3. E11. Get_Status Fields Checks ...................................... 57
Power Delivery 3.0 Tests

TD. PD. SRC3. E12.  Get_Battery_Status Fields Checks 58
TD. PD. SRC3. E15.  Battery_Capabilities sent timely 61
TD. PD. SRC3. E16.  Battery_Capabilities Fields Checks 61
TD. PD. SRC3. E17.  Battery_Capabilities Fields Checks - Invalid Battery Reference 63
TD. PD. SRC3. E18.  Manufacturer_Info Sent Timely 64
TD. PD. SRC3. E19.  Manufacturer_Info Fields Checks 65
TD. PD. SRC3. E20.  Manufacturer_Info Fields Checks - Invalid Manufacturer Info Target 66
TD. PD. SRC3. E21.  Manufacturer_Info Fields Checks - Invalid Manufacturer Info Ref 67
TD. PD. SRC3. E22.  Cable Type Detection 68
TD. PD. SRC3. E23.  Vconn Swap 68
TD. PD. SRC3. E24.  Unexpected Message Received in Ready State 69
TD. PD. SRC3. E25.  Receiving chunked extended message 70
TD. PD. SRC3. E26.  Soft_Reset sent regardless of Rp value 71
TD. PD. SRC3. E27.  PPS_Status Sent Timely 71
TD. PD. SRC3. E28.  PPS_Status Fields Check 72
TD. PD. SRC3. E29.  SourcePPSCommTimer Deadline 73
TD. PD. SRC3. E30.  SourcePPSCommTimer Timeout 74
TD. PD. SRC3. E31.  SourcePPSCommTimer Stopped 74
TD. PD. SRC3. E32.  ChunkSenderResponseTimer Timeout 75
TD. PD. SRC3. E33.  Country_Codes Sent Timely 76
TD. PD. SRC3. E34.  Country_Codes Fields Checks 76
TD. PD. SRC3. E35.  Country_Info Sent Timely 78
TD. PD. SRC3. E36.  Country_Info Fields Checks 78

Sink Tests .................................................................................................................................................. 80

TD. PD. SNK3. E1.  Request Fields Checks 80
TD. PD. SNK3. E2.  Unrecognized Message Received in Ready State 81
TD. PD. SNK3. E3.  Get_Source_Cap_Extended Fields Checks 81
TD. PD. SNK3. E4.  SenderResponseTimer Deadline - Source_Capabilities_Extended 82
TD. PD. SNK3. E5.  SenderResponseTimer Timeout - Source_Capabilities_Extended 83
TD. PD. SNK3. E6.  Get_Status Fields Checks 83
TD. PD. SNK3. E7.  Get_Battery_Status Fields Checks 84
TD. PD. SNK3. E8.  Status sent timely 85
TD. PD. SNK3. E9.  Manufacturer_Info Sent Timely 86
TD. PD. SNK3. E10.  Source_Capabilities_Extended sent timely 86
TD. PD. SNK3. E11.  Receiving chunked extended message 87
Power Delivery 3.0 Tests

TD. PD. SNK3. E12.  Soft_Reset sent regardless of Rp value  88
TD. PD. SNK3. E13.  SinkPPSPeriodicTimer Timeout  88
TD. PD. SNK3. E14.  Request Fields Checks - PPS  89
TD. PD. SNK3. E15.  Status Fields Checks  90
TD. PD. SNK3. E16.  Manufacturer_Info Fields Checks  91
TD. PD. SNK3. E17.  Manufacturer_Info Fields Checks - Invalid Manufacturer Info Target  92
TD. PD. SNK3. E18.  Manufacturer_Info Fields Checks - Invalid Manufacturer Info Ref  93
TD. PD. SNK3. E19.  ChunkSenderResponseTimer Timeout  94
TD. PD. SNK3. E20.  Collision Avoidance after PR_Swap  95

Cable Tests ................................................................. 97
TD.PD.CBL3.E1.  Receiving Chunked Extended Message  97
TD.PD.CBL3.E2.  ChunkSenderResponseTimer Timeout  98
TD.PD.CBL3.E3.  Manufacturer_Info Fields Checks  99
TD.PD.CBL3.E4.  Manufacturer_Info Fields Checks - Invalid Manufacturer Info Target  100
TD.PD.CBL3.E5.  Unrecognized Message Received  101
TD.PD.CBL3.E7.  Status sent timely  102
TD.PD.CBL3.E8.  Status Fields Checks  102

VDM Tests ........................................................................ 104
TD. PD. VDM3. E1.  Fields Checks - Discover Identity  104
TD. PD. VDM3. E2.  Unrecognized VID in Unstructured VDM  105

Consistency Tests ............................................................. 107
TD. PD. VNDI3. E1.  Source Capabilities  107
TD. PD. VNDI3. E2.  Request  108
TD. PD. VNDI3. E3.  VDM Identity  108
TD. PD. VNDI3. E4.  Manufacturer Info  110
TD. PD. VNDI3. E5.  Chunking Implemented  110
TD. PD. VNDI3. E6.  Unchunked_Extended_Messages_Supported  111
TD. PD. VNDI3. E7.  Security_Msgs_Supported  112
TD. PD. VNDI3. E8.  Sink Capabilities  112
TD. PD. VNDI3. E9.  Source_Capabilities_Extended  113
Copyright and Intellectual Property

Copyright © 2011 - 2016, USB Implementers Forum, Inc.

All rights reserved.

A LICENSE IS HEREBY GRANTED TO REPRODUCE THIS SPECIFICATION FOR INTERNAL USE ONLY. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, IS GRANTED OR INTENDED HEREBY.

USB-IF AND THE AUTHORS OF THIS SPECIFICATION EXPRESSLY DISCLAIM ALL LIABILITY FOR INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, RELATING TO IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. USB-IF AND THE AUTHORS OF THIS SPECIFICATION ALSO DO NOT WARRANT OR REPRESENT THAT SUCH IMPLEMENTATION(S) WILL NOT INFRINGE THE INTELLECTUAL PROPERTY RIGHTS OF OTHERS.

THIS SPECIFICATION IS PROVIDED "AS IS" AND WITH NO WARRANTIES, EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE. ALL WARRANTIES ARE EXPRESSLY DISCLAIMED. NO WARRANTY OF MERCHANTABILITY, NO WARRANTY OF NON-INFRINGEMENT, NO WARRANTY OF FITNESS FOR ANY PARTICULAR PURPOSE, AND NO WARRANTY ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE.

IN NO EVENT WILL USB-IF OR USB-IF MEMBERS BE LIABLE TO ANOTHER FOR THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA OR ANY INCIDENTAL, CONSEQUENTIAL, INDIRECT, OR SPECIAL DAMAGES, WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THE USE OF THIS SPECIFICATION, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

All product names are trademarks, registered trademarks, or servicemarks of their respective owners.
<table>
<thead>
<tr>
<th>Date</th>
<th>Rev</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aug. 30, 2016</td>
<td>0.10</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>
| Oct. 11, 2016 | 0.11 | Updated legal aspects  
Added section for Authors and Editor |
| Oct. 26, 2016 | 0.12 | Update Source Tests section with the following:  
- Minor updates to many tests  
- Change font for message names and parameter values  
- Parameters are referred to only by name, not by value |
| Nov. 2, 2016 | 0.13 | Update Sink Tests section with the following:  
- Minor updates to many tests  
- Change font for message names and parameter values  
- Parameters are referred to only by name, not by value |
| Jan. 9, 2017 | 0.3  | Added Cable Tests  
Added Tests Applicability Section |
| Jan. 25, 2017 | 0.4  | Minor updates to existing Cable Tests and added more Cable Tests |
| Mar. 2, 2017 | 0.45 | Update assertions and test spec per PD_R3_0 V1.1 20170112 |
| April 17, 2017 | 0.5  | Added SRC3.E24 and E25  
Deleted SNK3.E12  
Updated VNDI3.E5 to support all UUT Types  
Listed all tests in Tests Applicability Section  
Minor updates to assertions of other tests |
| June 30, 2017 | 0.7  | Added LL3.E1  
Added SRC3.E26 – E30  
Added SNK3.E12  
Added CBL3.E6  
Added VNDI3.E6 – E8 |
| July 10, 2017 | 0.8  | Added a new SRC3.E26 and renumbering SRC3.E26 – E30 to E27 – E31  
Added a new SNK3.E12 and renumbering SNK3.E12 to E13  
Added SNK3.E14  
Added VDM3.E2  
Updated SRC3.E9 and E17  
Updated VDM3.E1 to include NAK and BUSY cases  
| July 12, 2017 | 0.81 | Updated SNK3.E4 and E5 to resolve conflicts with the main spec |
| July 25, 2017 | 0.82 | Updated SRC3.E16, E17 to resolve conflicts with the main spec  
Fixed a typo in SRC3.E28 |
| Aug. 2, 2017  | 0.83 | LL3.E1 doesn’t apply to cable  
VDM3.E2 doesn’t apply to Provider / Consumer, Provider Only |
| Sept. 13, 2017 | 0.88 | Updated SRC3.E5, SRC3.E13, SRC3.E18, SNK3.E8, SNK3.E9 and SNK3.E10, if a Not_Supported message is received, the result is changed from N/A to Pass  
Revised SRC3.E8  
Changed PPS RDO current from 0.5A to 1A, because 1A is minimum |
### Power Delivery 3.0 Tests

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sept. 14, 2017</td>
<td>0.89</td>
<td>Updated PD.E1, if the UUT is a PD2 UUT, reports N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated SRC3.E6, SRC3.E12, SRC3.E14, SRC3.16-17, SRC3.E19-21, SRC3.E28 and SNK3.E7, test the case the Tester doesn’t support unchunked extended messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added SRC3.E32-33 and SNK3.E15-20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added LL3.E2</td>
</tr>
<tr>
<td>Sept. 20, 2017</td>
<td>0.90</td>
<td>Deleted SRC3.E33 and moved that to PROC.PD.E3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fixed a timer in VNDI3.E7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated PD.E2, allowing a 5s for a Sink to request PPS APDO</td>
</tr>
<tr>
<td>Oct. 12, 2017</td>
<td>0.91</td>
<td>Fixed a copy/paste error in SRC3.E15-E18</td>
</tr>
<tr>
<td>Oct. 18, 2017</td>
<td>0.92</td>
<td>Changed step d).2 of SRC.E1 to be compatible with USB PD R3 0 V1 1 PPS Current Requirements above Nominal Prog Voltage ECR</td>
</tr>
<tr>
<td>Oct. 20, 2017</td>
<td>0.93</td>
<td>Added assertions list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated the assertions tested of some tests</td>
</tr>
<tr>
<td>Nov. 27, 2017</td>
<td>0.94</td>
<td>Updated SRC3.E32 and SNK3.E19, calling out the Tester set Unchunked Extended Messages Supported to 0</td>
</tr>
<tr>
<td>Nov. 28, 2017</td>
<td>0.95</td>
<td>Removed not applicable from test descriptions</td>
</tr>
<tr>
<td>Dec. 12, 2017</td>
<td>0.96</td>
<td>Incorporated USB PD R3.0 V1.1 ECR Battery Numbering 20170405.docx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added checking of the total number of data bytes is consistent with the Number of Data Objects field for chunked messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added support for Num_Fixed_Batteries and Num_Swappable_Battery_Slots in VIF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added SRC3.E33-E36</td>
</tr>
<tr>
<td>Jan. 10, 2018</td>
<td>0.97</td>
<td>Updated SRC3.E35 and E36, if an UUT supports Country_Codes, it shall support Country_Info</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In SRC3.E36, only request Country_Info for the first country code</td>
</tr>
<tr>
<td>Feb. 12, 2018</td>
<td>0.98</td>
<td>Updated SRC3.E22, always pass if the UUT has a captive cable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated SRC3.E28 according to USB PD R3.0 V1.1 ECN PPS Status measurements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated SRC3.E30 according to USB PD R3.0 V1.1 ECN Clarify PPS Periodic RDO requirements</td>
</tr>
<tr>
<td>Feb. 14, 2018</td>
<td>0.99</td>
<td>Revised SRC3.E28 after working group discussion</td>
</tr>
<tr>
<td>Feb. 22, 2018</td>
<td>1.0</td>
<td>Revised VDM3.E1 according to USB PD R3.0 V1.1 ECN Active Cable Status 20170919</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added LL3.E3 according to USB PD R3.0 V1.1 ECN GoodCRC spec rev - 20160602</td>
</tr>
<tr>
<td>March 5, 2018</td>
<td>1.01</td>
<td>Added CBL3.E7 and E8 according to USB PD R3.0 V1.1 ECN Active Cable Status 20170919</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Revised VNDI3.E1 since the USB PD ECR Revision 3.0 Version 1.1 PDP Rounding Rules 20180129 was approved in the working group</td>
</tr>
<tr>
<td>May 7, 2018</td>
<td>1.02</td>
<td>Added 2.75V and 5.75V Vconn voltage requirement for cable tests</td>
</tr>
</tbody>
</table>
**Terms and Abbreviations**

The terms and abbreviations specific to compliance testing are listed in the table below. All other used terms and abbreviations are from the relevant specifications.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tester</td>
<td>The Tester is a piece of test equipment capable of running all tests described in this specification against an UUT.</td>
</tr>
<tr>
<td>Unit Under Test (UUT)</td>
<td>The PD device that is being tested by the Tester.</td>
</tr>
</tbody>
</table>
Test Results

The result of a test may be one of the following:

- **Pass**
  - The UUT meets all assertions tested in the test

- **Fail**
  - The UUT doesn’t meet at least one assertion tested in the test

- **Not Applicable (N/A)**
  - The test is not applicable to the UUT because one of the following reasons
    - The test is not applicable to this UUT Type
    - The UUT doesn’t meet a prerequisite
Related Documents

USB_PD_R3_0 V1.1 20170112
USB PD R3 0 V1.1 ECN PPS Current Requirements above Nominal Prog Voltage 20170918
USB PD R3.0 V1.1 ECN Battery Numbering 20170405
USB PD R3.0 V1.1 ECN Chunking Clarification 20170405
USB PD R3.0 V1.1 ECN VCONN_Swap Clarification 20170420
USB PD R3.0 V1.1 ECN Clarify PPS Periodic RDO requirements
USB PD R3.0 V1.1 ECN PDP Clarifications 20170922
USB PD R3.0 V1.1 ECN PDP Rounding 20170918
USB PD R3.0 V1.1 ECN Active Cable Status 20170919
USB PD R3.0 V1.1 ECN GoodCRC spec rev - 20160602
USB PD R3.0 V1.1 ECN APDO Min Voltage
Test Procedures

PROC.PD.E1. Bring-up procedure

For UFP(Sink) UUT:

a) The test starts in a disconnected state.
b) The Tester applies Rp (1.5 A) and waits for the UUT attachment.
c) If Ra is detected, the Tester applies Vconn.
d) The Tester applies Vbus and waits 50 ms.
e) The Tester transmits Source_Capabilities until reception of GoodCrc for tNoResponse max. The Source_Capabilities includes Fixed 5V 3A PDO.
f) The Tester waits for the Request from the UUT for tSenderResponse max. If the Specification Revision field of the Request message is 01b (Rev 2.0), the test is not applicable and stops here.
g) The Tester sends Accept, and when Vbus is stable at the target voltage, sends PS_RDY.

For DFP(Source) UUT:

a) The test starts in a disconnected state.
b) The Tester applies Rd and waits for Vbus for tNoResponse max.
c) The Tester waits for Source_Capabilities for tNoResponse max. If the Specification Revision field of the Source_Capabilities message is 01b (Rev 2.0), the test is not applicable and stops here.
d) The Tester replies GoodCrc on reception of the Source_Capabilities.
e) The Tester requests 5V 0.5A.
f) The Tester waits for PS_RDY for tPSSourceOn max.

For Cable UUT:

a) The test starts in a disconnected state.
b) The Tester applies Rp (3A) and waits for the UUT attachment (Ra Detected).
c) The Tester applies Rp (3A) and Rd on CC.
d) The Tester applies Vconn.
e) The Tester sends a Discover Identity message to the UUT.
f) If the Specification Revision field of the message received is 01b (Rev 2.0), the test is not applicable and stops here.

Note:
1. Cable tests need to be run four times:
   a. At both ends of the cable
   b. At two VConn voltage levels: 2.75V and 5.75V

2. The Tester uses SOP' message in cable tests
For DRP UUT:

   a) If a test applies to Source or Provider/Consumer, test the UUT as a Source.
   b) If a test applies to Sink or Consumer/Provider, test the UUT as a Sink.

PROC.PD.E2. Bring-up procedure for PPS Tests

For UFP(Sink) UUT:

   a) The test starts in a disconnected state.
   b) The Tester applies Rp (1.5 A) and waits for the UUT attachment.
   c) If Ra is detected, the Tester applies Vconn.
   d) The Tester applies Vbus and waits 50 ms.
   e) The Tester transmits Source Capabilities until reception of GoodCrc for tNoResponse max. The Source Capabilities includes Fixed 5V 3A PDO and PPS 5V 3A APDO.
   f) The Tester waits for the Request from the UUT for tSenderResponse max.
   g) Upon receipt of the Request message from the UUT, the Tester sends Accept, and when Vbus is stable at the target voltage, sends PS_RDY. If the Sink requests the PPS APDO, the bring-up procedure ends up here.
   h) The Tester change the Rp value to SinkTXOK.
   i) If the Sink doesn’t request the PPS APDO in 5s from the time the Rp value changed to SinkTXOK, the test is not applicable. If the Request message is received, The Tester sends Accept, and when Vbus is stable at the target voltage, sends PS_RDY.

For DFP(Source) UUT:

   a) The test starts in a disconnected state.
   b) The Tester applies Rd and waits for Vbus for tNoResponse max.
   c) The Tester waits for Source Capabilities for tNoResponse max.
   d) The Tester replies GoodCrc on reception of the Source Capabilities.
   e) If there is no PPS APDO in the Source Capabilities, the test is not applicable and stops here.
   f) The Tester requests PPS APDO at 4V 1A.
   g) The Tester waits for PS_RDY for tPSSourceOn max.

For DRP UUT:

   a) If a test applies to Source or Provider/Consumer, test the UUT as a Source.
   b) If a test applies to Sink or Consumer/Provider, test the UUT as a Sink.
PROC.PD.E3.  **Wait to Start AMS**

For DFP(Source) UUT:

a) The Tester keeps monitoring the Rp value and if the UUT doesn't set the value to SinkTXOK if it doesn't have anything to send in 1s, the test fails. During this period, the Tester replies any message sent from the UUT with a proper response.
## Assertions

New or changed assertions in PD3

<table>
<thead>
<tr>
<th>Assertion No.</th>
<th>Covered</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4#1</td>
<td>y</td>
<td>The Source Shall limit maximum capabilities it offers so as not to exceed the capabilities of the type of cabling detected.</td>
</tr>
<tr>
<td>4.4#2</td>
<td>y</td>
<td>Sources Shall detect the type of Attached cable and limit the Capabilities they offer based on the current carrying capability of the cable determined by the Cable capabilities determined using the Discover Identity Command (see Section 6.4.4.2) sent using SOP' Communication (see Section 2.5) to the Cable Plug.</td>
</tr>
<tr>
<td>4.4#5</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>5.4#2</td>
<td></td>
<td>The receiver Shall search for all four K-codes and when it finds either three out of four or all four in the correct place, it May interpret it as a Valid ordered set (see Table 5-3).</td>
</tr>
<tr>
<td>5.7#6</td>
<td>y</td>
<td>In addition the PHY Layer Shall control the Rp resistor value to avoid collisions between Source and Sink transmissions.</td>
</tr>
<tr>
<td>5.7#7</td>
<td>y</td>
<td>The Source Shall set an Rp value corresponding to a current of 3A to indicate to the Sink that it May initiate an AMS.</td>
</tr>
<tr>
<td>5.7#8</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>5.7#9</td>
<td>y</td>
<td>Table 5-13 details the Rp values that Shall be used by the Source to control Sink initiation of an AMS.</td>
</tr>
<tr>
<td>5.8.5.5#1</td>
<td></td>
<td>A Transmitter in a Port or Cable Plug Shall tolerate having its output be shorted to ground for tFRSwapTx max.</td>
</tr>
<tr>
<td>6.2.1.1.1#1</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>6.2.1.1.1#2</td>
<td>y</td>
<td>The Extended field Shall apply to all SOP* Packet types.</td>
</tr>
<tr>
<td>6.2.1.1.2#3</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>6.2.1.1.2#4</td>
<td>y</td>
<td>When the Extended bit is set to one and Chunked bit is set to zero, the Number of Data Objects field Shall be Reserved.</td>
</tr>
<tr>
<td>6.2.1.1.5#1</td>
<td>y</td>
<td>The Specification Revision field Shall be one of the following values (except 11b):</td>
</tr>
<tr>
<td>Test Case</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>6.2.1.5#3</td>
<td>To ensure interoperability with existing USBPD Products, USBPD Products Shall support every PD Specification Revision starting from [USBPD 2.0].</td>
<td></td>
</tr>
<tr>
<td>6.2.1.5#4</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.5#6</td>
<td>When the Source Port first communicates with the Sink Port the Specification Revision field Shall be used as described by the following steps:</td>
<td></td>
</tr>
<tr>
<td>6.2.1.5#7</td>
<td>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 they are Detached.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.5#8</td>
<td>When a Vconn Source first communicates with a Cable Plug the Specification Revision field Shall be used as described by the following steps:</td>
<td></td>
</tr>
<tr>
<td>6.2.1.5#9</td>
<td>The Cable Plug and Vconn Source Shall communicate using the lower of the two revisions until an Explicit Contract has been established.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.5#12</td>
<td>All data in all Messages Shall be consistent with the Specification Revision field in the Message Header for that particular Message.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.5#13</td>
<td>A Cable Plug Shall Not save the state of the agreed Specification Revision.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.5#14</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.5#15</td>
<td>Cable Plugs Shall operate using the same Specification Revision for both SOP’ and SOP’”.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.5#16</td>
<td>Cable assemblies with two Cable Plugs Shall operate using the same Specification Revision for both Cable Plugs.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.8#1</td>
<td>The 5-bit Message Type field Shall indicate the type of Message being sent.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.2#1</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.2#2</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.2#3</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.2.1#1</td>
<td>y</td>
<td>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 &gt; MaxExtendedMsgLegacyLen bytes in a single Unchunked Extended Message (see Section 6.4.1.2.2.6 and Section 6.4.2.6).</td>
</tr>
<tr>
<td>6.2.1.2.1#2</td>
<td>y</td>
<td>The Chunked bit in every Extended Message Shall be set to one</td>
</tr>
<tr>
<td>6.2.1.2.1#3</td>
<td></td>
<td>Every Extended Message of Data Size &gt; MaxExtendedMsgLegacyLen Shall be transmitted between the Port Partners in Chunks</td>
</tr>
<tr>
<td>6.2.1.2.1#4</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>6.2.1.2.1#5</td>
<td></td>
<td>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).</td>
</tr>
<tr>
<td>6.2.1.2.1#6</td>
<td>y</td>
<td>The Chunked bit in every Extended Message Shall be set to zero.</td>
</tr>
<tr>
<td>6.2.1.2.1#7</td>
<td>y</td>
<td>Every Extended Message Shall be transmitted between the Port Partners Unchunked</td>
</tr>
<tr>
<td>6.2.1.2.1#8</td>
<td></td>
<td>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).</td>
</tr>
<tr>
<td>6.2.1.2.1#9</td>
<td></td>
<td>When sending Extended Messages to the Cable Plug the Vconn Source Shall only send Chunked Messages.</td>
</tr>
<tr>
<td>6.2.1.2.1#10</td>
<td></td>
<td>Cable Plugs Shall always send Extended Messages of Data Size &gt; MaxExtendedMsgLegacyLen Chunked and Shall set the Chunked bit in every Extended Message to one.</td>
</tr>
<tr>
<td>6.2.1.2.1#11</td>
<td></td>
<td>When Extended Messages are supported Chunking Shall be supported.</td>
</tr>
<tr>
<td>6.2.1.2.2#1</td>
<td>y</td>
<td>The Chunk Number field Shall only be Valid in a Message if the Chunked flag is set to one.</td>
</tr>
<tr>
<td>6.2.1.2.2#2</td>
<td>y</td>
<td>if the Chunked flag is set to zero the Chunk Number field Shall also be set to zero.</td>
</tr>
<tr>
<td>6.2.1.2.2#3</td>
<td>y</td>
<td>The requestor Shall only set this field to one of the following values:</td>
</tr>
<tr>
<td>6.2.1.2.2#4</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>6.2.1.2.3#1</td>
<td>y</td>
<td>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).</td>
</tr>
<tr>
<td>Assertions</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>6.2.1.2.3#2</td>
<td>For Unchunked Extended Message transfers Messages Shall be sent and received without the request/response mechanism (see Figure 6-4).</td>
<td></td>
</tr>
<tr>
<td>6.2.1.2.3#3</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.2.3#4</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.2.3#5</td>
<td>Both the Chunk request and the Chunk response Shall contain the same value in the Message Type field.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.2.3#6</td>
<td>When the Request Chunk bit is set to one the Data Size field Shall be zero.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.2.4#1</td>
<td>The Data Size field Shall indicate how many bytes of data in total are in Data Block being returned.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.2.4#2</td>
<td>The total number of data bytes in the Message Shall Not exceed MaxExtendedMsgLen.</td>
<td></td>
</tr>
<tr>
<td>6.2.1.2.4#3</td>
<td>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).</td>
<td></td>
</tr>
<tr>
<td>6.2.1.2.4#4</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>6.3.4#4</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>6.3.6#1</td>
<td>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).</td>
<td></td>
</tr>
<tr>
<td>6.3.11#7</td>
<td>A Reject Message Shall only be sent by the Port that is not presently the Vconn Source in response to a VCONN_Swap Message.</td>
<td></td>
</tr>
<tr>
<td>6.3.11#8</td>
<td>The Port that is presently the Vconn Source Shall Not send a Reject Message in response to VCONN_Swap Message.</td>
<td></td>
</tr>
<tr>
<td>6.3.11#10</td>
<td>A Wait Message Shall only be sent by the Port that is not presently the Vconn Source in response to a VCONN_Swap Message.</td>
<td></td>
</tr>
<tr>
<td>6.3.11#11</td>
<td>The Port that is presently the Vconn Source Shall Not send a Wait Message in response to VCONN_Swap Message.</td>
<td></td>
</tr>
<tr>
<td>6.3.11#14</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>Assertion</td>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>6.3.12.4#1</td>
<td>y</td>
<td>A Wait Message Shall only be sent by the Port that is not presently the Vconn Source in response to a VCONN_Swap Message.</td>
</tr>
<tr>
<td>6.3.12.4#2</td>
<td>y</td>
<td>The Port that is presently the Vconn Source Shall Not send a Wait Message in response to VCONN_Swap Message.</td>
</tr>
<tr>
<td>6.3.14#1</td>
<td>y</td>
<td>The Not_Supported Message Shall be sent by a Port in response to any Message it does not support.</td>
</tr>
<tr>
<td>6.3.16#1</td>
<td>y</td>
<td>The Source or Sink Shall respond by returning a Status Message (see Section 6.5.2).</td>
</tr>
<tr>
<td>6.3.18#1</td>
<td>y</td>
<td>The Port Shall respond by returning a PPS_Status Message (see Section 6.5.10).</td>
</tr>
<tr>
<td>6.3.19#1</td>
<td>y</td>
<td>The Port Partner Shall respond by returning a Country_Codes Message (see Section 6.5.11).</td>
</tr>
<tr>
<td>6.4.1#3</td>
<td>y</td>
<td>Power Data Objects in a Capabilities Message Shall be sent in the following order:</td>
</tr>
<tr>
<td>6.4.1#8</td>
<td>y</td>
<td>The Programmable Power Supply Objects, if present, Shall be sent in Maximum Voltage order, lowest to highest.</td>
</tr>
<tr>
<td>6.4.1.2#6</td>
<td>y</td>
<td>Sinks with Accessory Support do not source VBUS (see [USB Type-C 1.2]) however when sourcing Vconn they Shall advertise vSafe5V with the Maximum Current set to 0mA in the first Power Data Object.</td>
</tr>
<tr>
<td>6.4.1.2#9</td>
<td>y</td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· The Sink requests something other than PPS APDO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· There is a Power Role Swap.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· There is a Hard Reset.</td>
</tr>
<tr>
<td>6.4.1.2#10</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>6.4.1.2#11</td>
<td>y</td>
<td>The Source Shall discontinue this behavior after:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Receiving a Request Message with a Fixed, Variable or Battery RDO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· There is a Power Role Swap.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· There is a Hard Reset.</td>
</tr>
<tr>
<td>6.4.1.2.2.6#1</td>
<td>y</td>
<td>The Unchunked Extended Messages Supported bit Shall be set when the Port can send and receive Extended Messages with Data Size &gt; MaxExtendedMsgLegacyLen bytes in a single, Unchunked Message.</td>
</tr>
<tr>
<td>6.4.1.2.2.7#5</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>6.4.1.2.2.7#6</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>Section</td>
<td>Assertion</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>6.4.1.2.5#1</td>
<td>SPT</td>
<td>The voltage fields define the output voltage range over which the power supply Shall be adjustable in 20mV steps.</td>
</tr>
<tr>
<td>6.4.1.2.5#2</td>
<td>SPT</td>
<td>The Maximum Current field contains the current the Programmable Power Supply Shall be capable of delivering over the advertised voltage range.</td>
</tr>
<tr>
<td>6.4.1.3.1.6#1</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>6.4.1.3.4#1</td>
<td>y</td>
<td>The Maximum and Minimum Voltage fields Shall be set to the output voltage range that the Sink requires to operate.</td>
</tr>
<tr>
<td>6.4.1.3.4#2</td>
<td>y</td>
<td>The Operational Current field Shall be set to the maximum current the Sink requires over the voltage range.</td>
</tr>
<tr>
<td>6.4.2.6#1</td>
<td>y</td>
<td>The Unchunked Extended Messages Supported bit Shall be set when the Port can send and receive Extended Messages with Data Size &gt; MaxExtendedMsgLegacyLen bytes in a single, Unchunked Message.</td>
</tr>
<tr>
<td>6.4.2.7#3</td>
<td>SPT</td>
<td>When the request is accepted the Source’s output current supplied into any load Shall be less than or equal to the Operating Current.</td>
</tr>
<tr>
<td>6.4.2.7#4</td>
<td>SPT</td>
<td>When the Sink attempts to consume more current, the Source Shall reduce the output voltage so as not to exceed the Operating Current value.</td>
</tr>
<tr>
<td>6.4.2.7#6</td>
<td>Y</td>
<td>This field Shall apply to the Fixed, Variable and Programmable RDO.</td>
</tr>
<tr>
<td>6.4.2.13#1</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>6.4.2.13#2</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>6.4.2.13#3</td>
<td>y</td>
<td>This field Shall apply to the Programmable RDO.</td>
</tr>
<tr>
<td>6.4.4.1#9</td>
<td>y</td>
<td>When a DFP or UFP does not support Unstructured VDMs or does not recognize the VID it Shall return a Not_Supported Message.</td>
</tr>
<tr>
<td>6.4.4.2#6</td>
<td>y</td>
<td>When a DFP or UFP does not support Structured VDMs any Structured VDMs received Shall return a Not_Supported Message.</td>
</tr>
<tr>
<td>6.4.4.2#10</td>
<td>y</td>
<td>Version 1.0 = 00b (Shall Not be used)</td>
</tr>
<tr>
<td>6.4.4.2.3#2</td>
<td>y</td>
<td>To ensure interoperability with existing USBPD Products, USBPD Products Shall support every Structured VDM Version number starting from Version 1.0.</td>
</tr>
<tr>
<td>6.4.4.2.3#4</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>6.4.4.3.1.3#1</td>
<td>y</td>
<td>For DRD Products this field Shall indicate the capability regardless of the present Data Role.</td>
</tr>
<tr>
<td>Specification</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>6.4.4.3.1.6#1</td>
<td>For DRD Products this field Shall indicate the capability regardless of the present Data Role.</td>
<td></td>
</tr>
<tr>
<td>6.4.4.3.1.6#2</td>
<td>Table 6-32 defines the Product Type VDOs which Shall be returned.</td>
<td></td>
</tr>
<tr>
<td>6.4.4.3.1.6#3</td>
<td>Shall be used where no other Product Type value is appropriate.</td>
<td></td>
</tr>
<tr>
<td>6.4.4.3.1.6#4</td>
<td>Shall be used when the Product is a PDUSB Hub.</td>
<td></td>
</tr>
<tr>
<td>6.4.4.3.1.6#5</td>
<td>Shall be used when the Product is a PDUSB Host.</td>
<td></td>
</tr>
<tr>
<td>6.4.4.3.1.6#6</td>
<td>Shall be used when the Product is a Power Brick/Wall Wart.</td>
<td></td>
</tr>
<tr>
<td>6.4.4.3.1.6#7</td>
<td>Shall be used when the Product is a PDUSB Host or DFP that supports one or more Alternate Modes.</td>
<td></td>
</tr>
<tr>
<td>6.4.3.1.4.1#3</td>
<td>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 6-35.</td>
<td></td>
</tr>
<tr>
<td>6.4.3.1.4.1#4</td>
<td>00b = Reserved, Shall Not be used</td>
<td></td>
</tr>
<tr>
<td>6.4.3.1.4.1#5</td>
<td>01b = Reserved, Shall Not be used</td>
<td></td>
</tr>
<tr>
<td>6.4.3.1.4.1#14</td>
<td>The Maximum VBUS 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.</td>
<td></td>
</tr>
<tr>
<td>6.4.3.1.4.2#3</td>
<td>Active Cables Shall support the Structured VDM Discover Identity Command and Shall return the Active Cable VDO in a Discover Identity Command ACK as shown in Table 6-36.</td>
<td></td>
</tr>
<tr>
<td>6.4.3.1.4.2#4</td>
<td>00b = Reserved, Shall Not be used</td>
<td></td>
</tr>
<tr>
<td>6.4.3.1.4.2#5</td>
<td>01b = Reserved, Shall Not be used</td>
<td></td>
</tr>
<tr>
<td>6.4.3.1.4.2#14</td>
<td>The Maximum VBUS 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.</td>
<td></td>
</tr>
<tr>
<td>6.4.5#1</td>
<td>The Battery_Status Message Shall be sent in response to a Get_Battery_Status Message.</td>
<td></td>
</tr>
<tr>
<td>6.4.5#3</td>
<td>The Battery_Status Message returns a BSDO whose format Shall be as shown in Figure 6-27 and Table 6-39.</td>
<td></td>
</tr>
<tr>
<td>6.4.5#4</td>
<td>The Number of Data Objects field in the Battery_Status Message Shall be set to 1.</td>
<td></td>
</tr>
<tr>
<td>6.4.5#5</td>
<td>When Battery is present Shall contain the Battery charging status:</td>
<td></td>
</tr>
<tr>
<td>6.4.5#6</td>
<td>11b: Reserved, Shall Not be used</td>
<td></td>
</tr>
<tr>
<td>6.4.5.2#1</td>
<td>The Battery Info field Shall be used to report additional information about the Battery’s present status.</td>
<td></td>
</tr>
<tr>
<td>6.4.5.2#2</td>
<td>The Battery Info field’s bits Shall reflect the present conditions under which the Battery is operating in the systems.</td>
<td></td>
</tr>
<tr>
<td>6.4.5.2#1</td>
<td>The Invalid Battery Reference bit Shall be set when the Get_Battery_Status Message contains a reference to a Battery that does not exist.</td>
<td></td>
</tr>
<tr>
<td><strong>6.4.5.2.2#1</strong></td>
<td>y</td>
<td>The Battery is Present bit Shall be set whenever the Battery is present.</td>
</tr>
<tr>
<td><strong>6.4.5.2.2#2</strong></td>
<td>y</td>
<td>It Shall always be set for Batteries that are not Hot Swappable Batteries.</td>
</tr>
<tr>
<td><strong>6.4.5.2.3#1</strong></td>
<td>y</td>
<td>These bits Shall be set when the Battery is present bit is set.</td>
</tr>
<tr>
<td><strong>6.4.5.2.3#2</strong></td>
<td>y</td>
<td>Otherwise when the Battery is present bit is zero the Battery charging status bits Shall also be zero.</td>
</tr>
<tr>
<td><strong>6.4.6#1</strong></td>
<td>y</td>
<td>The Alert Message Shall only be sent when the Source or Sink detects a status change.</td>
</tr>
<tr>
<td><strong>6.4.6#2</strong></td>
<td>y</td>
<td>The Alert Message Shall contain exactly one Alert Data Object (ADO) and the format Shall be as shown in Figure 6-28 and Table 6-40.</td>
</tr>
<tr>
<td><strong>6.4.6.1#1</strong></td>
<td>y</td>
<td>The Type of Alert field Shall be used to report Source or Sink status changes.</td>
</tr>
<tr>
<td><strong>6.4.6.1#2</strong></td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td><strong>6.4.6.1#3</strong></td>
<td>y</td>
<td>Once the Alert Message has been sent the Type of Alert field Shall be cleared.</td>
</tr>
<tr>
<td><strong>6.4.6.1.1#1</strong></td>
<td>y</td>
<td>The Battery Status Change bit Shall be set when any Battery’s power state changes between charging, discharging, neither.</td>
</tr>
<tr>
<td><strong>6.4.6.1.1#2</strong></td>
<td>y</td>
<td>For Hot Swappable Batteries, it Shall also be set when a Battery is Attached or Detached.</td>
</tr>
<tr>
<td><strong>6.4.6.1.2#1</strong></td>
<td>y</td>
<td>The Over-Current Protection Event bit Shall be set when a Source detects its output current exceeds its limits triggering its protection circuitry.</td>
</tr>
<tr>
<td><strong>6.4.6.1.3#1</strong></td>
<td>y</td>
<td>The Over-Temperature Protection Event bit Shall be set when a Source or Sink shuts down due to over-temperature triggering its protection circuitry.</td>
</tr>
<tr>
<td><strong>6.4.6.1.4#1</strong></td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td><strong>6.4.6.1.4#2</strong></td>
<td>y</td>
<td>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 Foldback (CF).</td>
</tr>
<tr>
<td><strong>6.4.6.1.5#1</strong></td>
<td>y</td>
<td>The Source Input Event bit Shall be set when the Source/Sink’s input changes.</td>
</tr>
<tr>
<td><strong>6.4.6.1.6#1</strong></td>
<td>y</td>
<td>The Over-Voltage Protection Event bit Shall be set when the Sink detects its output voltage exceeds its limits triggering its protection circuitry.</td>
</tr>
<tr>
<td><strong>6.4.6.2#1</strong></td>
<td>y</td>
<td>Once the Alert Message has been sent the Fixed Batteries field Shall be cleared.</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>6.4.6.3#1</td>
<td>Once the Alert Message has been sent the Hot Swappable Batteries field Shall be cleared.</td>
<td></td>
</tr>
<tr>
<td>6.4.7#1</td>
<td>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].</td>
<td></td>
</tr>
<tr>
<td>6.4.7#2</td>
<td>The Get_Country_Info Message Shall be as shown in Figure 6-29 and Table 6-41.</td>
<td></td>
</tr>
<tr>
<td>6.5#1</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>6.5.1#1</td>
<td>The Source_Capabilities_Extended Message Shall return a 24-byte Source Capabilities Extended Data Block (SCEDB) whose format Shall be as shown in Figure 6-30 and Table 6-43.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.1#1</td>
<td>The Vendor ID field Shall contain the 16-bit Vendor ID (VID) assigned to the Source’s vendor by the USB-IF.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.1#2</td>
<td>If the vendor does not have a VID, the Vendor ID field Shall be set to zero.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.1#3</td>
<td>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 3.1]).</td>
<td></td>
</tr>
<tr>
<td>6.5.1.2#1</td>
<td>The Product ID field Shall contain the 16-bit Product ID (PID) assigned by the Source’s vendor.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.2#2</td>
<td>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 3.1]).</td>
<td></td>
</tr>
<tr>
<td>6.5.1.3#1</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.3#2</td>
<td>If the vendor does not have an XID, then it Shall return zero in this field (see [USB 2.0] and [USB 3.1]).</td>
<td></td>
</tr>
<tr>
<td>6.5.1.4#1</td>
<td>The Firmware Version field Shall contain an 8-bit firmware version number assigned to the device by the vendor.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.5#1</td>
<td>The Hardware Version field Shall contain an 8-bit hardware version number assigned to the device by the vendor.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.6.1#1</td>
<td>The Source Shall report its load step response capability in bits 0…1 of the Voltage Regulation bit field.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.6.2#1</td>
<td>The Source Shall report its load step magnitude rate as a percentage of IoC in bit 2 of the Voltage Regulation field.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.7#1</td>
<td>The Holdup Time field Shall contain the Source’s holdup time (see Section 7.1.12.2).</td>
<td></td>
</tr>
<tr>
<td>6.5.1.8#1</td>
<td>The Compliance field Shall contain the standards the Source is compliant with (see Section 7.1.12.3).</td>
<td></td>
</tr>
<tr>
<td>6.5.1.9#1</td>
<td>A Source Shall set the Touch Current bit (bit-0) when their leakage current is less than 65(\mu)A rms when Source’s maximum capability is less than or equal to 30W, or when their leakage current is less than 100 (\mu)A rms when its power capability is between 30W and 100W.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.9#2</td>
<td>The total combined leakage current Shall be measured in accordance with [IEC 60950-1] when tested at 250VAC rms at 50 Hz.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.9#3</td>
<td>A Source with a ground pin Shall set the Ground pin bit (bit-1).</td>
<td></td>
</tr>
<tr>
<td>6.5.1.9#4</td>
<td>A Source whose Ground pin is intended to be connected to a protective earth Shall set both bit1 and bit 2.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.10#1</td>
<td>The Peak Current field Shall contain the combinations of Peak Current that the Source supports (see Section 7.1.12.4).</td>
<td></td>
</tr>
<tr>
<td>6.5.1.10#2</td>
<td>A Source Shall populate unused Peak Current bit fields with zero.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.10#3</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.10#4</td>
<td>Overload Period Shall be the minimum rolling average time window in 20ms increments, where a value of 20ms is recommended.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.10#5</td>
<td>Duty Cycle Shall be the maximum percentage of overload period reported in 5% increments.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.10#6</td>
<td>VBUS Droop Shall be set to one to indicate there is an additional 5% voltage droop on VBUS when the overload conditions occur.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.11#1</td>
<td>The Touch Temp field Shall report the IEC standard used to determine the surface temperature of the Source’s enclosure.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.12#1</td>
<td>The Source Inputs field Shall identify the possible inputs that provide power to the Source.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.13#1</td>
<td>The Batteries field Shall report the number of batteries the source supports.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.13#2</td>
<td>It Shall independently report the number of Hot Swappable Batteries and the number of Fixed batteries.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.13#3</td>
<td>The maximum number of each type of Battery Shall be no more than 4.</td>
<td></td>
</tr>
<tr>
<td>6.5.1.14#1</td>
<td>The Source PDP field Shall report the Source’s rated PDP as defined in Table 10-2.</td>
<td></td>
</tr>
<tr>
<td>6.5.2#1</td>
<td>The Status Message Shall be sent in response to a Get_Status Message.</td>
<td></td>
</tr>
<tr>
<td>6.5.2#2</td>
<td>The Status Message returns a 5-byte Status Data Block (SDB) whose format Shall be as shown in Figure 6-31 and Table 6-44.</td>
<td></td>
</tr>
<tr>
<td>6.5.2#4</td>
<td>When Present Source Input field bit 3 is not set this field is Reserved and shall be set to zero.</td>
<td></td>
</tr>
<tr>
<td>6.5.2.4#2</td>
<td>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_PPS_Status Message.</td>
<td></td>
</tr>
<tr>
<td>Section</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>6.5.2.4#2</td>
<td>y</td>
<td>When the OTP event flag is set the Temperature Status field Shall also be set to over temperature.</td>
</tr>
<tr>
<td>6.5.2.4#3</td>
<td></td>
<td>The CF/CV mode bit is only Valid when operating as a Programmable Power Supply and Shall be Ignored otherwise.</td>
</tr>
<tr>
<td>6.5.2.4#4</td>
<td>SPT</td>
<td>When the Source is operating as a Programmable Power Supply the CF/CV mode bit Shall be set when operating in Current Foldback mode (CF mode) and Shall be cleared when operating in Constant Voltage mode (CV mode).</td>
</tr>
<tr>
<td>6.5.2.5#1</td>
<td>y</td>
<td>When the Temperature Status field is set to over temperature the OTP event flag Shall also be set.</td>
</tr>
<tr>
<td>6.5.3#1</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>6.5.3#2</td>
<td></td>
<td>The Get_Battery_Cap Message contains a 1 byte Get Battery Cap Data Block (GBCDB), whose format Shall be as shown in Figure 6-32 and Table 6-45.</td>
</tr>
<tr>
<td>6.5.3#3</td>
<td></td>
<td>The Data Size field in the Get_Battery_Cap Message Shall be set to 1.</td>
</tr>
<tr>
<td>6.5.4#2</td>
<td>y</td>
<td>The Get_Battery_Status Message contains a 1 byte Get Battery Status Data Block (GBSDB) whose format Shall be as shown in Figure 6-33 and Table 6-46.</td>
</tr>
<tr>
<td>6.5.4#3</td>
<td>y</td>
<td>The Data Size field in the Get_Battery_Status Message Shall be set to 1.</td>
</tr>
<tr>
<td>6.5.5#2</td>
<td>y</td>
<td>The Battery_Capabilities Message returns a 9-byte BCDB whose format Shall be as shown in Figure 6-34 and Table 6-44.</td>
</tr>
<tr>
<td>6.5.5.1#1</td>
<td>y</td>
<td>The Battery Design Capacity field Shall return the Battery’s design capacity in tenths of WH.</td>
</tr>
<tr>
<td>6.5.5.3.1#1</td>
<td>y</td>
<td>The Invalid Battery Reference bit Shall be set when the Get_Battery_Cap Message contains a reference to a Battery that does not exist.</td>
</tr>
<tr>
<td>6.5.6#1</td>
<td>y</td>
<td>The Port or Cable Plug Shall respond by returning a Manufacturer_Info Message (Section 6.5.7) containing a Manufacturer Info Data Block (MIDB).</td>
</tr>
<tr>
<td>6.5.6#2</td>
<td></td>
<td>The Get_Manufacturer_Info Message returns a GMIDB whose format Shall be as shown in Figure 6-33 and Table 6-48.</td>
</tr>
<tr>
<td>6.5.6#3</td>
<td></td>
<td>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:</td>
</tr>
<tr>
<td>6.5.6#4</td>
<td></td>
<td>Otherwise this field is Reserved and Shall be set to zero.</td>
</tr>
<tr>
<td>6.5.7#1</td>
<td>y</td>
<td>The Manufacturer_Info Message Shall be sent in response to a Get_Manufacturer_Info Message.</td>
</tr>
</tbody>
</table>
### 6.5.7#2
- **y**
- The Manufacturer_Info Message returns a Manufacturer Info Data Block (MIDB) whose format Shall be as shown in Figure 6-34 and Table 6-44.

### 6.5.7.1#1
- **y**
- This field Shall contain the device’s or Battery’s 16-bit vendor ID assigned by the USB.

### 6.5.7.2#1
- **y**
- This field Shall contain the device’s or Battery’s 16-bit product identifier designated by the vendor.

### 6.5.7.3#1
- This field Shall contain the device’s or Battery’s manufacturer string as defined by the vendor.

### 6.5.7.3#2
- **y**
- If the Manufacturer Info Target field or Manufacturer Info Ref field in the Get.Manufacturer_Info Message is unrecognized the field Shall return zero bytes.

### 6.5.8.1#1
- The Security_Request Message contains a Security Request Data Block (SRQDB) whose format Shall be as shown in Figure 6-37.

### 6.5.8.2#1
- The Security_Response Message contains a Security Response Data Block (SRPDB) whose format Shall be as shown in Figure 6-38.

### 6.5.9.1#1
- The Firmware_Update_Request Message contains a Firmware Update Request Data Block (FRQDB) whose format Shall be as shown in Figure 6-39.

### 6.5.9.2#1
- The Firmware_Update_Response Message contains a Firmware Update Response Data Block (FRPDB) whose format Shall be as shown in Figure 6-40.

### 6.5.10#1
- **y**
- The PPS_Status Message Shall be sent in response to a Get_PPS_Status Message.

### 6.5.10#3
- **y**
- The PPS_Status Message Shall return a 4-byte PPS Status Data Block (PPSSDB) whose format Shall be as shown in Figure 6-41 and Table 6-50.

### 6.5.10.1#1
- **y**
- The Output Voltage field Shall return the Source’s output voltage at the time of the request.

### 6.5.10.1#2
- **y**
- If the Source does not support this field, it Shall be set to 0xFFFF.

### 6.5.10.2#1
- **y**
- The Output Current field Shall return the Source’s output current at the time of the request.

### 6.5.10.2#2
- **y**
- If the Source does not support this field, it Shall be set to 0xFF.

### 6.5.10.3#1
- **y**
- The PTF (Present Temperature Flag) Shall provide a real time indication of the Source’s internal thermal status.

### 6.5.10.3#2
- **y**
- The OMF (Operating Mode Flag) Shall provide a real time indication of the Source’s operating mode.

### 6.5.11#1
- **y**
- The Country_Codes Message Shall be sent in response to a Get_Country_Codes Message.

### 6.5.11#2
- **y**
- The Country_Codes Message Shall contain a 4-260 byte Country Code Data Block (CCDB) whose format Shall be as shown in Figure 6-42 and Table 6-51.
<table>
<thead>
<tr>
<th>Specification</th>
<th>Assertion</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5.11.1#1</td>
<td>The Country Code field Shall contain the Alpha-2 Country Code defined by [ISO 3166].</td>
</tr>
<tr>
<td>6.5.12#1</td>
<td>The Country_Info Message Shall be sent in response to a Get_Country_Info Message.</td>
</tr>
<tr>
<td>6.5.12#2</td>
<td>The Country_Info Message Shall contain a 4-260 byte Country Info Data Block (CIDB) whose format Shall be as shown in Figure 6-43 and Table 6-52.</td>
</tr>
<tr>
<td>6.5.12.1#1</td>
<td>The Country Code field Shall contain the Alpha-2 Country Code defined by [ISO 3166].</td>
</tr>
<tr>
<td>6.5.12.2#1</td>
<td>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.</td>
</tr>
<tr>
<td>6.6.17.1#1</td>
<td>The ChunkingNotSupportedTimer Shall be started when:</td>
</tr>
<tr>
<td>6.6.17.1#2</td>
<td>The Policy Engine Shall Not send its Not_Supported Message before the ChunkingNotSupportedTimer expires.</td>
</tr>
<tr>
<td>6.6.17.2#1</td>
<td>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.</td>
</tr>
<tr>
<td>6.6.17.2#2</td>
<td>The ChunkSenderRequestTimer Shall be started when:</td>
</tr>
<tr>
<td>6.6.17.2#3</td>
<td>The ChunkSenderRequestTimer Shall be stopped when:</td>
</tr>
<tr>
<td>6.6.17.2#4</td>
<td>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.</td>
</tr>
<tr>
<td>6.6.17.2#5</td>
<td>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.</td>
</tr>
<tr>
<td>6.6.17.3#1</td>
<td>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.</td>
</tr>
<tr>
<td>6.6.17.3#2</td>
<td>The ChunkSenderResponseTimer Shall be started when:</td>
</tr>
<tr>
<td>6.6.17.3#3</td>
<td>The ChunkSenderResponseTimer Shall be stopped when:</td>
</tr>
<tr>
<td>6.6.17.3#4</td>
<td>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.</td>
</tr>
<tr>
<td>6.6.17.3#5</td>
<td>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.</td>
</tr>
<tr>
<td>Assertion</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>6.6.18.1#1</td>
<td>The SinkPPSPeriodicTimer Shall be used by the Sink’s Policy Engine to ensure that a Request Message requesting a PPS APDO is sent periodically within a bounded time of tPPSRequest.</td>
</tr>
<tr>
<td>6.6.18.1#2</td>
<td>The tPPSRequest time Shall be measured from the time the last bit of the EOP of the GoodCRC Message sent by the Source in response to the previous Request Message.</td>
</tr>
<tr>
<td>6.6.18.1#3</td>
<td>SinkPPSPeriodicTimer Shall be re-initialized and restarted when the last bit of the EOP of the GoodCRC Message sent in response to a Request Message for a PPS APDO has been received by the Physical Layer.</td>
</tr>
<tr>
<td>6.6.18.1#4</td>
<td>The Sink Shall stop the SinkPPSPeriodicTimer when: · The Sink requests something other than PPS APDO. · There is a Power Role Swap. · There is a Hard Reset.</td>
</tr>
<tr>
<td>6.6.18.2#1</td>
<td>The SourcePPSCommTimer Shall be used by the Source’s Policy Engine to ensure that a Request Message requesting a PPS APDO is received periodically within a bounded time of tPPSTimeout.</td>
</tr>
<tr>
<td>6.6.18.2#2</td>
<td>The tPPSTimeout time Shall be measured from the time the last bit of the EOP of the GoodCRC Message sent in response to the previous Request Message for a PPS APDO has been sent by the Physical Layer.</td>
</tr>
<tr>
<td>6.6.18.2#3</td>
<td>The SourcePPSCommTimer Shall be re-initialized and restarted when the last bit of the EOP of the GoodCRC Message sent in response to a Request Message for a PPS APDO has been sent by the Physical Layer.</td>
</tr>
<tr>
<td>6.6.18.2#4</td>
<td>The Source Shall stop the SourcePPSCommTimer when: · A Request Message has been received. · There is a Power Role Swap. · There is a Hard Reset.</td>
</tr>
<tr>
<td>6.6.18.2#5</td>
<td>When the SourcePPSCommTimer times out the Source Shall issue Hard Reset Signaling.</td>
</tr>
<tr>
<td>6.7.2#3</td>
<td>Extended Messages of Data Size &gt; MaxExtendedMsgLegacyLen that are not Chunked (Chunked flag set to zero) Shall Not be retried.</td>
</tr>
<tr>
<td>6.7.2#4</td>
<td>Extended Messages of Data Size ≤ MaxExtendedMsgLegacyLen (Chunked flag set to zero or one) Shall be retried.</td>
</tr>
<tr>
<td>6.7.2#5</td>
<td>Extended Messages of Data Size &gt; MaxExtendedMsgLegacyLen that are Chunked (Chunked flag set to one) individual Chunks Shall be retried.</td>
</tr>
<tr>
<td>6.8.1#4</td>
<td>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.</td>
</tr>
<tr>
<td>6.8.1#6</td>
<td>y</td>
</tr>
<tr>
<td>6.8.1#7</td>
<td>y</td>
</tr>
<tr>
<td>6.8.2#6</td>
<td>y</td>
</tr>
<tr>
<td>6.11.2#2</td>
<td></td>
</tr>
<tr>
<td>6.11.2.1.1#1</td>
<td>y</td>
</tr>
<tr>
<td>6.11.2.1.2#1</td>
<td></td>
</tr>
<tr>
<td>6.11.2.1.1.1#1</td>
<td>y</td>
</tr>
<tr>
<td>6.11.2.1.2.1#2</td>
<td></td>
</tr>
<tr>
<td>6.11.2.1.2.1#3</td>
<td>y</td>
</tr>
<tr>
<td>6.11.2.1.2.2#1</td>
<td></td>
</tr>
<tr>
<td>6.11.2.1.2.2#2</td>
<td></td>
</tr>
<tr>
<td>6.11.2.1.2.3#1</td>
<td>y</td>
</tr>
<tr>
<td>6.11.2.1.2.4#1</td>
<td>y</td>
</tr>
<tr>
<td>6.11.2.1.2.4#2</td>
<td>y</td>
</tr>
<tr>
<td>6.11.2.1.2.4#3</td>
<td></td>
</tr>
<tr>
<td>6.11.2.1.2.5#1</td>
<td>y</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>6.11.2.1.2.5#2</td>
<td>y</td>
</tr>
<tr>
<td>6.11.2.1.2.5#3</td>
<td>y</td>
</tr>
<tr>
<td>6.11.2.1.2.5#4</td>
<td>y</td>
</tr>
<tr>
<td>6.11.2.1.3.1#1</td>
<td></td>
</tr>
<tr>
<td>6.11.2.1.3.1#2</td>
<td></td>
</tr>
<tr>
<td>6.11.2.1.3.1#3</td>
<td></td>
</tr>
<tr>
<td>6.11.2.1.3.1#4</td>
<td></td>
</tr>
<tr>
<td>6.11.2.1.3.1#5</td>
<td></td>
</tr>
<tr>
<td>6.11.2.1.3.2#1</td>
<td></td>
</tr>
<tr>
<td>6.11.2.1.3.2#2</td>
<td></td>
</tr>
<tr>
<td>6.11.2.1.3.3#1</td>
<td></td>
</tr>
<tr>
<td>6.11.2.1.3.3#2</td>
<td></td>
</tr>
<tr>
<td>6.11.2.1.3.4#1</td>
<td></td>
</tr>
<tr>
<td>6.11.2.1.3.4#2</td>
<td>The Chunked Tx State Machine Shall transition to the TCH_Wait_For_Message_Request_From_Policy_Engine state when:</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>6.11.2.1.3.5#1</td>
<td>On entry to the TCH_Prepare_To_Send_Chunked_Message state the Chunked Tx State Machine Shall:</td>
</tr>
<tr>
<td>6.11.2.1.3.5#2</td>
<td>The Chunked Tx State Machine Shall transition to the TCH_Construct_Chunked_Message state when:</td>
</tr>
<tr>
<td>6.11.2.1.3.6#1</td>
<td>On entry to the TCH_Construct_Chunked_Message state the Chunked Tx State Machine Shall:</td>
</tr>
<tr>
<td>6.11.2.1.3.6#2</td>
<td>The Chunked Tx State Machine Shall transition to the TCH_Sending_Chunked_Message state when:</td>
</tr>
<tr>
<td>6.11.2.1.3.6#3</td>
<td>The Chunked Tx State Machine Shall transition to the TCH_Wait_For_Message_Request_From_Policy_Engine state when:</td>
</tr>
<tr>
<td>6.11.2.1.3.7#1</td>
<td>On entry to the TCH_Wait_Chunk_Request state the Chunked Tx State Machine Shall:</td>
</tr>
<tr>
<td>6.11.2.1.3.7#2</td>
<td>The Chunked Tx State Machine Shall transition to the TCH_Message_Sent state when:</td>
</tr>
<tr>
<td>6.11.2.1.3.7#3</td>
<td>The Chunked Tx State Machine Shall transition to the TCH_Report_Error state when:</td>
</tr>
<tr>
<td>6.11.2.1.3.8#1</td>
<td>On entry to the TCH_Wait_Chunk_Request state the Chunked Tx State Machine Shall:</td>
</tr>
<tr>
<td>6.11.2.1.3.8#2</td>
<td>The Chunked Tx State Machine Shall transition to the TCH_Report_Error state when:</td>
</tr>
<tr>
<td>6.11.2.1.3.8#3</td>
<td>The Chunked Tx State Machine Shall transition to the TCH_Message_Sent state when:</td>
</tr>
<tr>
<td>6.11.2.1.3.8#4</td>
<td>The Chunked Tx State Machine Shall transition to the TCH_Message_Received state when:</td>
</tr>
<tr>
<td>6.11.2.1.3.9#1</td>
<td>The Chunked Tx State Machine Shall enter the TCH_Message_Received state:</td>
</tr>
<tr>
<td>6.11.2.1.3.9#2</td>
<td>On entry to the TCH_Message_Received state the Chunked Tx State Machine Shall:</td>
</tr>
<tr>
<td>6.11.2.1.3.9#3</td>
<td>The Chunked Message Router Shall transition to the TCH_Wait_For_Message_Request_From_Policy_Engine state when:</td>
</tr>
<tr>
<td>6.11.2.1.3.10#1</td>
<td>On entry to the TCH_Report_Error state the Chunked Tx State Machine Shall:</td>
</tr>
<tr>
<td>6.11.2.1.3.10#2</td>
<td>The Chunked Tx State Machine Shall transition to the TCH_Wait_For_Message_Request_From_Policy_Engine state when:</td>
</tr>
<tr>
<td>6.11.2.1.4.1#1</td>
<td>The Chunked Message Router Shall transition to the RTR_Rx_Chunks state when:</td>
</tr>
<tr>
<td>6.11.2.1.4.1#2</td>
<td>The Chunked Message Router Shall transition to the RTR_Tx_Chunks state when:</td>
</tr>
<tr>
<td>6.11.2.1.4.1#3</td>
<td>The Chunked Message Router Shall transition to the RTR_Ping state when:</td>
</tr>
<tr>
<td>Specification</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>6.11.2.1.4.2#1</td>
<td>On entry to the RTR_Rx_Chunks state the Chunked Message Router Shall:</td>
</tr>
<tr>
<td>6.11.2.1.4.3#1</td>
<td>On entry to the RTR_Ping state the Chunked Message Router Shall:</td>
</tr>
<tr>
<td>6.11.2.1.4.4#1</td>
<td>On entry to the RTR.Tx_Chunks state the Chunked Message Router Shall:</td>
</tr>
<tr>
<td>6.11.2.2.1#1</td>
<td>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:</td>
</tr>
<tr>
<td>6.11.2.2.1#2</td>
<td>On entry to the PRL.Tx_Src_Sink.Tx state the Protocol Layer Shall request the PHY Layer to Rp to SinkTxOk.</td>
</tr>
<tr>
<td>6.11.2.1#3</td>
<td>The Protocol Layer Shall transition to the PRL.Tx_Wait_for_Message_Request state when:</td>
</tr>
<tr>
<td>6.11.2.2.2#1</td>
<td>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:</td>
</tr>
<tr>
<td>6.11.2.2.2#2</td>
<td>On entry to the PRL.Tx_Src_Source.Tx state the Protocol Layer Shall set Rp to SinkTxNG.</td>
</tr>
<tr>
<td>6.11.2.2.2#3</td>
<td>The Protocol Layer Shall transition to the PRL.Tx_Src_Pending state when:</td>
</tr>
<tr>
<td>6.11.2.2.3#1</td>
<td>On entry to the PRL.Tx_Src_Pending state the SinkTxTimer Shall be initialized and run.</td>
</tr>
<tr>
<td>6.11.2.2.3#2</td>
<td>The Protocol Layer Shall transition to the PRL.Tx_Construct_Message state when:</td>
</tr>
<tr>
<td>6.11.2.2.3#3</td>
<td>The Protocol Layer Shall transition to the PRL.Tx_Layer_Reset_for_Transmit state when:</td>
</tr>
<tr>
<td>6.11.2.3.1#1</td>
<td>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:</td>
</tr>
<tr>
<td>6.11.2.3.1#2</td>
<td>The Protocol Layer Shall transition to the PRL.Tx_Snk_Pending state when:</td>
</tr>
<tr>
<td>6.11.2.3.2#1</td>
<td>The Protocol Layer Shall transition to the PRL.Tx_Construct_Message state when:</td>
</tr>
<tr>
<td>6.11.2.3.2#2</td>
<td>The Protocol Layer Shall transition to the PRL.Tx_Layer_Reset_for_Transmit state when:</td>
</tr>
<tr>
<td>6.12.1#6</td>
<td>Note 5: Shall be supported products that support the Source_Capabilities.Extended.Message.</td>
</tr>
<tr>
<td>6.12.1#10</td>
<td>Note 9: Shall be supported when PPS is supported.</td>
</tr>
<tr>
<td>6.12.2#4</td>
<td>Note 2: Shall be supported by products that contain batteries.</td>
</tr>
<tr>
<td>6.12.2#5</td>
<td>Note 3: Shall be supported by products that support the Get_Battery_Status.Message.</td>
</tr>
<tr>
<td>6.12.3#3</td>
<td>Note 1: Shall be supported by products that contain batteries.</td>
</tr>
</tbody>
</table>
### Power Delivery 3.0 Tests

<table>
<thead>
<tr>
<th>Note</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.12.3#8</td>
<td>Note 6: Shall be supported by products that support USB security communication as defined in [USBTypeCAuthentication 1.0]</td>
</tr>
<tr>
<td>6.12.3#9</td>
<td>Note 7: Shall be supported by products that support USB firmware update communication as defined in [USBPD FirmwareUpdate 1.0]</td>
</tr>
</tbody>
</table>
| 6.12.3#10 | y  
Note 8: Shall be supported when PPS is supported. |
| 6.12.4#2 | y  
If Structured VDMs are not supported, the DFP or UFP receiving a VDM Command Shall send a Not_Supported Message in response. |
| 7.1.3#5 | SPT  
The output voltage of the Programmable Power Supply Shall remain within a range defined by the relative tolerance vPpsNew and the absolute band vPpsValid. |
| 7.1.4.3#2 | SPT  
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 tPpsSrcTransition. |
| 7.1.4.3#3 | SPT  
Any undershoot or overshoot beyond vPpsNew Shall Not exceed vPpsValid at any time. |
| 7.1.4.3#4 | SPT  
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. |
| 7.1.4.3#5 | SPT  
The nominal requested voltage of all linear voltage changes Shall equate to an integer number of LSB changes. |
| 7.1.4.3#6 | SPT  
A PPS Shall be able to supply the negotiated current level as it change its output voltage to the requested level. |
| 7.1.4.3#7 | SPT  
All PPS voltage increases Shall result in a voltage that is greater than the previous PPS output voltage. |
| 7.1.4.3#8 | SPT  
Likewise, all PPS voltage decreases Shall result in a voltage that is less than the previous PPS output voltage. |
| 7.1.4.3#9 | SPT  
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. |
| 7.1.4.4#1 | SPT  
The Programmable Power Supply Shall foldback 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. |
| 7.1.4.4#2 | SPT  
All programming changes of the Operating Current Shall settle to the new Operating Current value within tPpsCfProgramSettle. |
| 7.1.4.4#3 | SPT  
A Source that supports PPS Shall support foldback programmability between iPpsCfMin and the Maximum Current value in the PPS APDO. |
| 7.1.4.4#4 | SPT  
Any current overshoot or undershoot that occurs due to a load change during Current Foldback Shall Not exceed iPpsCfTransient and Shall settle to the Operating Current value within tPpsCfSettle. |
<table>
<thead>
<tr>
<th>Test Case</th>
<th>Assertion</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1.4.4#5</td>
<td>Voltage overshoot or undershoot caused by a transition from Current Foldback mode to Constant Voltage mode Shall Not exceed $v_{PpsCfCvTransient}$ and Shall settle to the Operating Voltage value within $t_{PpsCfCvTransient}$.</td>
</tr>
<tr>
<td>7.1.4.4#6</td>
<td>Likewise, current overshoot or undershoot caused by a transition from Constant Voltage mode to Current Foldback mode Shall Not exceed $i_{PpsCvCfTransient}$ and Shall settle to the Operating Current value within $t_{PpsCvCfTransient}$.</td>
</tr>
<tr>
<td>7.1.4.4#7</td>
<td>The PPS Shall maintain its output voltage within the Minimum Voltage and Maximum Voltage values advertised in the PPS APDO for all static and dynamic load conditions during Current Foldback operation.</td>
</tr>
<tr>
<td>7.1.4.4#8</td>
<td>Rather, the Source Shall send Hard Reset Signaling and discharge VBUS to $v_{Safe0V}$ then resume default operation at $v_{Safe5V}$.</td>
</tr>
<tr>
<td>7.1.8.1#1</td>
<td>after $t_{PpsSrcTransition}$ and during static load conditions the Source output voltage Shall remain within the $v_{PpsNew}$ limits.</td>
</tr>
<tr>
<td>7.1.8.1#2</td>
<td>after $t_{PpsSrcTransition}$ and during transient load conditions the Source output voltage Shall Not go beyond the range specified by $v_{PpsValid}$.</td>
</tr>
<tr>
<td>7.1.8.1#3</td>
<td>The amount of time the Source output voltage can be in the band between $v_{PpsNew}$ and $v_{PpsValid}$ Shall Not exceed $t_{PpsTransient}$.</td>
</tr>
<tr>
<td>7.1.12.1.1#1</td>
<td>A Source Shall meet the following requirements under the load step reported in the Extended Source Capabilities:</td>
</tr>
<tr>
<td>7.1.12.1.1#2</td>
<td>The Source Shall maintain VBUS regulation within the $v_{SrcValid}$ range.</td>
</tr>
<tr>
<td>7.1.12.1.1#3</td>
<td>The noise on the CC line Shall remain below $v_{NoiseIdle}$ and $v_{NoiseActive}$.</td>
</tr>
<tr>
<td>7.1.12.1.1#4</td>
<td>The Source Shall ensure that PD Communications meet the transmit and receive masks as specified in Section 5.8.2 under all load conditions.</td>
</tr>
<tr>
<td>7.1.12.2#1</td>
<td>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.</td>
</tr>
<tr>
<td>7.1.12.2#2</td>
<td>A mains supplied Source Shall report its holdup time in this field.</td>
</tr>
<tr>
<td>7.1.12.3#1</td>
<td>A Source claiming LPS, PS1 or PS2 compliance (see [IEC 62368-1]) Shall report its capabilities in the Compliance field.</td>
</tr>
<tr>
<td>7.1.12.3#2</td>
<td>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.</td>
</tr>
<tr>
<td>7.1.12.4#1</td>
<td>The duration of peak current Shall be followed by a current consumption below the Operating Current ($I_{OC}$) in order to maintain average power delivery below the $I_{OC}$ current.</td>
</tr>
<tr>
<td>7.1.12.4#2</td>
<td>In this case the Source Shall report its additional capability in the Peak Current field in the Source_Capabilities_Extended Message.</td>
</tr>
<tr>
<td>7.1.12.4#3</td>
<td>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.</td>
</tr>
<tr>
<td>7.1.12.6#1</td>
<td>The Batteries field Shall report the number of Batteries the Source supports.</td>
</tr>
<tr>
<td>7.1.12.6#2</td>
<td>The Source Shall independently report the number of Hot Swappable Batteries and the number of Fixed batteries.</td>
</tr>
<tr>
<td>7.2.3.1#1</td>
<td>However, prior to operating the PPS in current foldback, the Sink Shall program the PPS Operating Voltage to the lowest practical level that satisfies the Sink load requirement.</td>
</tr>
<tr>
<td>7.2.3.1#2</td>
<td>When operating with a PPS that is in current foldback, the Sink Shall Not change its load in a manner that exceeds iPpsCfLoadStepRate or iPpsCfLoadReleaseRate.</td>
</tr>
<tr>
<td>7.2.3.1#3</td>
<td>The load change magnitude Shall Not exceed iPpsCfLoadStep or iPpsCfLoadRelease.</td>
</tr>
<tr>
<td>7.2.3.1#4</td>
<td>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.</td>
</tr>
<tr>
<td>7.2.6#3</td>
<td>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.</td>
</tr>
<tr>
<td>7.2.9.4#2</td>
<td>The Sink Shall never draw higher current than the Maximum Current value in the PPS APDO.</td>
</tr>
<tr>
<td>7.3.16#4</td>
<td>The Programmable Power Supply new voltage Shall be reached by tPpsSrcTransition.</td>
</tr>
<tr>
<td>7.3.17#4</td>
<td>The Programmable Power Supply new voltage Shall be reached by tPpsSrcTransition.</td>
</tr>
<tr>
<td>7.3.18#1</td>
<td>The Source voltage as the transition starts Shall be any voltage within the Valid VBUS range of the previous Source PDO or APDO.</td>
</tr>
<tr>
<td>7.3.18#2</td>
<td>The Source voltage after the transition is complete Shall be any voltage within the Valid VBUS range of the new Source PDO or APDO.</td>
</tr>
<tr>
<td>7.3.18#5</td>
<td>The Source transition to the new PDO or APDO VBUS voltage Shall be completed by tSrcTransition.</td>
</tr>
<tr>
<td>7.4.1#2</td>
<td>The time the Programmable Power Supply Shall transition between requested voltages.</td>
</tr>
</tbody>
</table>
| 8.3.3.2.6#6 | 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 SourcePPSCommTimer |
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3.3.2.6#7</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.7#4</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.5.1.1#1</td>
<td>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).</td>
</tr>
<tr>
<td>8.3.3.5.1.2#1</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.5.1.3#1</td>
<td>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 6.6.17.1 and Section 8.3.3.4.1).</td>
</tr>
<tr>
<td>8.3.3.5.1.3#2</td>
<td>On entry to the PE_SRC_Chunk_Received state (from the PE_SRC_Ready state) the Policy Engine Shall initialize and run the ChunkingNotSupportedTimer.</td>
</tr>
<tr>
<td>8.3.3.5.1.3#3</td>
<td>The Policy Engine Shall transition to PE_SRC_Send_Not_Supported when:</td>
</tr>
<tr>
<td>8.3.3.5.2.1#1</td>
<td>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).</td>
</tr>
<tr>
<td>8.3.3.5.2.1#2</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.5.2.3#1</td>
<td>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 6.6.17.1 and Section 8.3.3.4.1).</td>
</tr>
<tr>
<td>Assertion</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>8.3.3.5.2.3#2</td>
<td>On entry to the PE_SNK_Chunk_Received state (from the PE_SNK_Ready state) the Policy Engine Shall initialize and run the ChunkingNotSupportedTimer.</td>
</tr>
<tr>
<td>8.3.3.5.2.3#3</td>
<td>The Policy Engine Shall transition to PE_SNK_Send_Not_Supported when:</td>
</tr>
<tr>
<td>8.3.3.7.1.1#1</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.7.1.1#2</td>
<td>On entry to the PE_SRC_Send_Source_Alert state the Policy Engine Shall request the Protocol Layer to send an Alert Message.</td>
</tr>
<tr>
<td>8.3.3.7.1.1#3</td>
<td>The Policy Engine Shall transition back to PE_SRC_Ready (see Figure 8-66) when:</td>
</tr>
<tr>
<td>8.3.3.7.2.1#1</td>
<td>The PE_SNK_Source_Alert_Received state Shall be entered from the PE_SNK_Ready state when an Alert Message is received.</td>
</tr>
<tr>
<td>8.3.3.7.2.1#2</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.7.2.1#3</td>
<td>The Policy Engine Shall transition back to PE_SNK_Ready (see Figure 8-67) when:</td>
</tr>
<tr>
<td>8.3.3.7.3.1#1</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.7.3.1#2</td>
<td>On entry to the PE_SNK_Send_Sink_Alert state the Policy Engine Shall request the Protocol Layer to send an Alert Message.</td>
</tr>
<tr>
<td>8.3.3.7.3.1#3</td>
<td>The Policy Engine Shall transition back to PE_SNK_Ready (see Figure 8-67) when:</td>
</tr>
<tr>
<td>8.3.3.7.4.1#1</td>
<td>The PE_SRC_Sink_Alert_Received state Shall be entered from the PE_SRC_Ready state when an Alert Message is received.</td>
</tr>
<tr>
<td>8.3.3.7.4.1#2</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.7.4.1#3</td>
<td>The Policy Engine Shall transition back to PE_SRC_Ready (see Figure 8-66) when:</td>
</tr>
<tr>
<td>8.3.3.8.1.1#1</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.8.1.1#2</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.8.1.1#3</td>
<td>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).</td>
</tr>
<tr>
<td>8.3.3.8.1.1#4</td>
<td>The Policy Engine Shall transition back to the PE_SNK_Ready state (see Figure 8-67) when:</td>
</tr>
<tr>
<td>8.3.3.8.2.1#1</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.8.2.1#2</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.8.2.1#3</td>
<td>The Policy Engine Shall transition back to the PE_SRC_Ready state (see Figure 8-66) when:</td>
</tr>
<tr>
<td>8.3.3.9.1.1#1</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.9.1.1#2</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.9.1.1#3</td>
<td>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).</td>
</tr>
<tr>
<td>8.3.3.9.1.1#4</td>
<td>The Policy Engine Shall transition back to the PE_SNK_Ready state (see Figure 8-67) when:</td>
</tr>
<tr>
<td>8.3.3.9.2.1#1</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.9.2.1#2</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.9.3.1#1</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.9.3.1#2</td>
<td>On entry to the PE_SRC_Get_Sink_Status state the Policy Engine Shall send a Status Message and initialize and run the SenderResponseTimer.</td>
</tr>
<tr>
<td>8.3.3.9.3.1#3</td>
<td>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).</td>
</tr>
<tr>
<td>8.3.3.9.3.1#4</td>
<td>The Policy Engine Shall transition back to the PE_SRC_Ready state (see Figure 8-66) when:</td>
</tr>
</tbody>
</table>
| 8.3.3.9.4.1#1 | 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.
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Outcome</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3.3.9.4.1#2</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.9.5.1#1</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.9.5.1#2</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.9.5.1#3</td>
<td></td>
<td>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).</td>
</tr>
<tr>
<td>8.3.3.9.5.1#4</td>
<td></td>
<td>The Policy Engine Shall transition back to the PE_SNK_Ready state (see Figure 8-67) when:</td>
</tr>
<tr>
<td>8.3.3.9.6.1#1</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.9.6.1#2</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.9.6.1#3</td>
<td></td>
<td>The Policy Engine Shall transition back to the PE_SRC_Ready state (see Figure 8-66) when:</td>
</tr>
<tr>
<td>8.3.3.10.1.1#1</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.10.1.1#2</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.10.1.1#3</td>
<td></td>
<td>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).</td>
</tr>
<tr>
<td>8.3.3.10.1.1#4</td>
<td></td>
<td>The Policy Engine Shall transition back to either the PE_SRC_Ready or PE_SNK_Ready state as appropriate (see Figure 8-66 and Figure 8-67) when:</td>
</tr>
<tr>
<td>8.3.3.10.2.1#1</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>Assertion</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>8.3.3.10.2.1#2</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>8.3.3.11.1.1#1</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>8.3.3.11.1.1#2</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>8.3.3.11.1.1#3</td>
<td>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).</td>
<td></td>
</tr>
<tr>
<td>8.3.3.11.1.1#4</td>
<td>The Policy Engine Shall transition back to either the PE_SRC_Ready or PE_SNK_Ready state as appropriate (see Figure 8-66 and Figure 8-67) when:</td>
<td></td>
</tr>
<tr>
<td>8.3.3.12.1.1#1</td>
<td>The Policy Engine Shall transition to the PE_Get_Manufacturer_Info state, from either the PE_SRC_Ready or PE_SNK_Ready state, when a Get_Battery_Status Message is received.</td>
<td></td>
</tr>
<tr>
<td>8.3.3.12.1.1#2</td>
<td>On entry to the PE_Get_Manufacturer_Info 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.</td>
<td></td>
</tr>
<tr>
<td>8.3.3.12.1.1#3</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>8.3.3.12.1.1#4</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>8.3.3.3.11.1#3</td>
<td>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).</td>
<td></td>
</tr>
<tr>
<td>8.3.3.3.12.1.1#4</td>
<td>The Policy Engine Shall transition back to either the PE_SRC_Ready or PE_SNK_Ready state as appropriate (see Figure 8-66 and Figure 8-67) when:</td>
<td></td>
</tr>
<tr>
<td>8.3.3.3.12.2.1#1</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>assertion_id</td>
<td>y</td>
<td>description</td>
</tr>
<tr>
<td>--------------</td>
<td>---</td>
<td>-------------</td>
</tr>
<tr>
<td>8.3.3.12.2.1#2</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.13.1.1#1</td>
<td></td>
<td>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 Manufacturer Information from the Device Policy Manager.</td>
</tr>
<tr>
<td>8.3.3.13.1.1#2</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.13.1.1#3</td>
<td></td>
<td>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).</td>
</tr>
<tr>
<td>8.3.3.13.1.1#4</td>
<td></td>
<td>The Policy Engine Shall transition back to either the PE_SRC.Ready or PE_SNK.Ready state as appropriate (see Figure 8-66 and Figure 8-67) when:</td>
</tr>
<tr>
<td>8.3.3.13.2.1#1</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.13.2.1#2</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.13.2.1#3</td>
<td></td>
<td>The Policy Engine Shall transition back to either the PE_SRC.Ready, PE_SNK.Ready or PE_CBL.Ready state as appropriate (see Figure 8-66, Figure 8-67 and Figure 8-126) when:</td>
</tr>
<tr>
<td>8.3.3.13.3.1#1</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.13.3.1#2</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.13.3.1#3</td>
<td></td>
<td>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).</td>
</tr>
<tr>
<td>8.3.3.13.3.1#4</td>
<td></td>
<td>The Policy Engine Shall transition back to either the PE_SRC.Ready or PE_SNK.Ready state as appropriate (see Figure 8-66 and Figure 8-67) when:</td>
</tr>
<tr>
<td>8.3.3.13.4.1#1</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>----------------</td>
<td>----</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>8.3.3.13.4.1#2</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.13.4.1#3</td>
<td></td>
<td>The Policy Engine Shall transition back to either the PE_SRC_Ready, PE_SNK_Ready or PE_CBL_Ready state as appropriate (see Figure 8-66, Figure 8-67 and Figure 8-126) when:</td>
</tr>
<tr>
<td>8.3.3.14.2.1#1</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.14.2.1#2</td>
<td>y</td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.14.2.1#3</td>
<td></td>
<td>The Policy Engine Shall transition back to either the PE_SRC_Ready, PE_SNK_Ready or PE_CBL_Ready state as appropriate (see Figure 8-66, Figure 8-67 and Figure 8-126) when:</td>
</tr>
<tr>
<td>8.3.3.15.1.1#1</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.15.1.1#2</td>
<td></td>
<td>On entry to the PE_Send_Firmware_Update_Request state the Policy Engine Shall send a Firmware_Update_Request Message.</td>
</tr>
<tr>
<td>8.3.3.15.1.1#3</td>
<td></td>
<td>The Policy Engine Shall transition back to either the PE_SRC_Ready or PE_SNK_Ready state as appropriate (see Figure 8-66 and Figure 8-67) when:</td>
</tr>
<tr>
<td>8.3.3.15.2.1#1</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.15.2.1#2</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>8.3.3.15.2.1#3</td>
<td></td>
<td>The Policy Engine Shall transition back to either the PE_SRC_Ready, PE_SNK_Ready or PE_CBL_Ready state as appropriate (see Figure 8-66, Figure 8-67 and Figure 8-126) when:</td>
</tr>
<tr>
<td>8.3.3.15.3.1#1</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>8.3.3.15.3.1#2</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>8.3.3.15.3.1#3</td>
<td>The Policy Engine Shall transition back to either the PE_SRC_Ready or PE_SNK_Ready state as appropriate (see Figure 8-66, Figure 8-67 and Figure 8-126) when:</td>
<td></td>
</tr>
<tr>
<td>8.3.3.16.11.1#1</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>8.3.3.16.11.1#2</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>8.3.3.16.11.1#3</td>
<td>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).</td>
<td></td>
</tr>
<tr>
<td>8.3.3.16.11.1#4</td>
<td>The Policy Engine Shall transition back to the PE_SRC_Ready state (see Figure 8-66) when:</td>
<td></td>
</tr>
<tr>
<td>8.3.3.16.12.1#1</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>8.3.3.16.12.1#2</td>
<td>The Policy Engine Shall request the present extended Source capabilities from the Device Policy Manager and then send a SourceCapabilitiesExtended Message based on these capabilities.</td>
<td></td>
</tr>
<tr>
<td>8.3.3.23.1.1#2</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>8.3.3.23.1.1#3</td>
<td>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:</td>
<td></td>
</tr>
<tr>
<td>10.2.3.2#1</td>
<td>The Programmable Power Supply’s PDP Shall be calculated as the product of the Nominal Voltage times the Maximum Current.</td>
<td></td>
</tr>
<tr>
<td>10.2.3.2#3</td>
<td>A Source that advertises Optional Programmable Power Supply APDOs Shall advertise the PDOs and APDOs shown in Table 10-7.</td>
<td></td>
</tr>
<tr>
<td>10.2.3.2#4</td>
<td>A Source Shall advertise Optional Programmable Power Supply APDOs with Maximum Voltage and Minimum Voltages for nominal voltage as defined in Table 10-8.</td>
<td></td>
</tr>
</tbody>
</table>
A Source that advertises Programmable Power Supply APDOs other than the ones listed in Table 10-8 Shall Not advertise additional APDO’s with a Maximum Voltage * Maximum Current that exceeds the adapter's PDP.

In no case Shall a Source advertise a current that exceeds the attached cable’s current rating.

Table 10-7 shows the Programmable Power Supply APDOs that Shall be offered for a given PDP.

The voltage output at the Source’s connector Shall be +/-5% for both the Maximum Voltage and the Minimum Voltage.

Common assertions to all tests

6.2.1.1.8#1,
### Tests Applicability

<table>
<thead>
<tr>
<th>Test</th>
<th>Consumer Only</th>
<th>Consumer/Provider</th>
<th>Provider Only</th>
<th>Provider/Consumer</th>
<th>Dual-role Port</th>
<th>Cable</th>
</tr>
</thead>
<tbody>
<tr>
<td>TD.PD.LL3.E1</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.LL3.E2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.LL3.E3</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E1</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E2</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E3</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E4</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E5</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E6</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E7</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E8</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E9</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E10</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E11</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E12</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E13</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E14</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E15</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E16</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E17</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E18</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E19</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E20</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E21</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E22</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E23</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E24</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E25</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E26</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E27</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E28</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E29</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E30</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E31</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E32</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E33</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E34</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E35</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SRC3.E36</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E1</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E2</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E3</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E4</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E5</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E6</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E7</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E8</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E9</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E10</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E11</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E12</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E13</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E14</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E15</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E16</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E17</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E18</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E19</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.SNK3.E20</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.CBL3.E1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>Consumer Only</td>
<td>Consumer/Provider</td>
<td>Provider Only</td>
<td>Provider/Consumer</td>
<td>Dual-role Port</td>
<td>Cable</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>-------------------</td>
<td>---------------</td>
<td>-------------------</td>
<td>----------------</td>
<td>-------</td>
</tr>
<tr>
<td>TD.PD.CBL3.E2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.CBL3.E3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.CBL3.E4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.CBL3.E5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.CBL3.E6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.CBL3.E7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.CBL3.E8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.VDM3.E1</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.VDM3.E2</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.VNDI3.E1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.VNDI3.E2</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.VNDI3.E3</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.VNDI3.E4</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.VNDI3.E5</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.VNDI3.E6</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.VNDI3.E7</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.VNDI3.E8</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>TD.PD.VNDI3.E9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
</tbody>
</table>
Link Layer Tests

TD. PD. LL3. E1. GoodCRC Specification Revision compatibility

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port, Consumer Only, Consumer / Provider

Assertions Tested:
6.2.1.5#4

Description:
The Tester verifies the UUT accepts GoodCRC with Specification Revision set to 00b, 01b, or 10b.

Steps:
a) The Tester is configured to reply GoodCRC with Specification Revision set to 00b.
b) The Tester runs PROC.PD.E1 Bring-up according to the UUT role until the Tester has replied GoodCRC to the first message from the UUT.
c) The test fails if the UUT retransmits the last message.
d) Repeat the test with Specification Revision set to 01b and 10b in GoodCRC.

TD. PD. LL3. E2. Retransmission

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port, Consumer Only, Consumer / Provider

Assertions Tested:
6.6.1#2

Description:
The Tester verifies the UUT retransmit messages correctly.

Steps:
a) Run PROC.PD.E1 Bring-up according to the UUT role.
b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a Get_Source_Cap message to the UUT.
c) Upon receipt of the message in reply, the Tester doesn’t reply GoodCRC, and verifies the UUT retransmit the message nRetryCount times.
TD. PD. LL3. E3.  GoodCRC Compatibility with PD2

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port, Consumer Only, Consumer / Provider, Cable

Assertions Tested:

Description:
The Tester verifies the UUT set Specification Revision to 01b in GoodCRC when the Tester is in PD2 mode.

Steps:

a) Put the Tester to PD2 mode, i.e. always set Specification Revision to 01b in PD messages.

b) Run PROC.PD.E1 Bring-up according to the UUT role until the Tester has received a GoodCRC from the UUT.

   c) The Tester verifies the Specification Revision is 01b in the GoodCRC message.
Source Tests

TD. PD. SRC3. E1. Source Capabilities Fields Checks

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:
6.2.1.1.1#1, 6.2.1.1.5#6, 6.2.1.1.8#1, 6.4.1#3, 6.4.1#8

Description:
As Consumer (UFP), the Tester waits for a Source Capabilities message from the Provider (DFP, UUT) and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up For DFP UUT steps a and b.

b) Upon receipt of the Source Capabilities message from the Provider, the Tester verifies:
   1. **Number of Data Objects** field equals the number of Src_PDOs in the message and is not 000b.
   2. **Port Power Role** field = 1b (Source)
   3. **Specification Revision** field = 10b (Rev 3.0)
   4. **Port Data Role** field = 1b (DFP)
   5. **Message Type** field = 00001b (Source Capabilities)
   6. **Extended** field = 0b

c) For the first PDO, the Tester verifies:
   1. Bits 31..30 (PDO type) are 00b (Fixed Supply).
   2. Voltage field = 100 (5 V)
   3. Bits 23..22 = 000b (**Reserved**)

d) For the other PDOs (if any), the Tester verifies:
   1. If Bits 31..30 are 00b, Bits 29..22 are set to 0.
   2. If Bits 31..30 are 11b
      - Bits 29..28 are 00b (**Reserved**)
      - Bits 26..25 are 00b (**Reserved**)
      - Bit 16 is 0b (**Reserved**)
      - Bit 7 is 0b (**Reserved**)
   3. PDOs are in the order of Fixed Supply Objects (if present), Battery Supply Objects (if present), Variable Supply Objects (if present) and then Programmable Power Supply Objects (if present).
4. Fixed Supply Objects (if present) are in voltage order; lowest to highest.
5. Battery Supply Objects (if present) are in Minimum Voltage order; lowest to highest.
6. Variable Supply Objects (if present) are in Minimum Voltage order; lowest to highest.
7. Programmable Power Supply Objects (if present) are in Maximum Voltage order; lowest to highest.

e) The test fails if any verification is not satisfied.

TD. PD. SRC3. E2. Accept Fields Checks

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.2.1.1.1#1, 6.2.1.1.5#1, 6.2.1.1.5#7, 6.2.1.1.5#12, 6.2.1.1.8#1

Description:
As Consumer (UFP), the Tester requests 100% of the offered current or power under the terms of the last PDO on the UUT, waits for an Accept message from the Provider (DFP, UUT) and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up steps a-d according to the UUT role.

b) The Tester sends a Request, with Specification Revision in the message header set to 10b, for power under the conditions of the last Power Data Object, requesting 100% of the current or power offered depending on the type of the PDO.

c) Upon receipt of the Accept message, the Tester verifies:
   1. Number of Data Objects field = 000b
   2. Port Power Role field = 1b (Source)
   3. Specification Revision field = 10b (Rev 3.0)
   4. Port Data Role field = 1b (DFP)
   5. Extended = 0b

d) The test fails if any verification is not satisfied.

TD. PD. SRC3. E3. PS_RDY Fields Checks

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.2.1.1.1#1, 6.2.1.1.5#1, 6.2.1.1.5#7, 6.2.1.1.5#12, 6.2.1.1.8#1
Description:

As Consumer (UFP), the Tester requests 100% of the offered current or power under the terms of the last PDO on the UUT, waits for a PS_RDY message from the Provider (DFP, UUT) and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester sends a Request, with Specification Revision in the message header set to 10b, for power under the conditions of the last Power Data Object, requesting 100% of the current or power offered depending on the type of the PDO.

c) Upon receipt of the Accept message from the Provider, the Tester replies with a GoodCRC message.

d) Upon receipt of the PS_RDY message, the Tester verifies:
   1. Number of Data Objects field = 000b
   2. Port Power Role field = 1b (Source)
   3. Specification Revision field = 10b (Rev 3.0)
   4. Port Data Role field = 1b (DFP)
   5. Extended = 0b

e) The test fails if any verification is not satisfied.

TD. PD. SRC3. E4. Specification Revision Check after Contract

Applicable to:

- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

5.7#6, 5.7#7, 5.7#8, 5.7#9, 6.2.1.1.5#1, 6.2.1.1.5#7

Description:

As Consumer (UFP), the Tester verifies that the UUT continues to use Rev 3.0 after setting up a contract with the UUT using Rev 3.0.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a Get_Source_Cap message to the UUT.

c) Upon receipt of the Source_Capabilities Message, the Tester verifies the Specification Revision in the message header is 10b (Rev 3.0).
TD. PD. SRC3. E5. Source_Capabilities_Extended sent timely

Applicable to:

- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.12.1#6, 8.3.3.8.2.1#1, 8.3.3.8.2.1#2

Description:

As Consumer (UFP), the Tester verifies that the UUT replies Get_Source_Cap_Extended message with a Source_Capabilities_Extended message timely.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.
b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a Get_Source_Cap_Extended message to the UUT.
c) If a Source_Capabilities_Extended or Not_Supported message is not received within tReceiverResponse max, the test fails. This delay is measured from the time the last bit of Get_Source_Cap_Extended message EOP has been transmitted to the time the first bit of the Source_Capabilities_Extended message preamble has been received.

TD. PD. SRC3. E6. Source_Capabilities_Extended Fields Checks

Applicable to:

- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.2.1.1.1#1, 6.2.1.1.1#2, 6.2.1.1.1#4, 6.2.1.1.2#3, 6.2.1.1.2#4, 6.2.1.1.5#7, 6.2.1.1.8#1, 6.2.1.2#1, 6.2.1.2#3, 6.2.1.2.1#1, 6.2.1.2.1#2, 6.2.1.2.1#4, 6.2.1.2.1#6, 6.2.1.2.1#7, 6.2.1.2.2#1, 6.2.1.2.2#2, 6.2.1.2.3#1, 6.2.1.2.3#3, 6.2.1.2.4#1, 6.2.1.2.4#3, 6.4.1.2.2.7#5, 6.5#1, 6.5.1#1, 6.5.1.13#2, 6.5.1.13#3

Description:

As Consumer (UFP), the Tester sends a Get_Source_Cap_Extended message to the UUT, waits for a Source_Capabilities_Extended message from the UUT and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported to 1 in Request message during this process.
b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a Get_Source_Cap_Extended message to the UUT.
c) If a Not Supported message is received, the test passes and stops here.

d) Upon receipt of the Source Capabilities Extended message, the Tester verifies:
   1. **Port Power Role** field = 1b (Source)
   2. **Specification Revision** field = 10b (Rev 3.0)
   3. **Port Data Role** field = 1b (DFP)
   4. **Extended** = 1b
   5. **Chunk Number** in Extended Message Header = 0
   6. **Request Chunk** in Extended Message Header = 0
   7. Bit 9 (Reserved) in Extended Message Header = 0
   8. **Data Size** in Extended Message Header = 24
   9. If both UUT and the Tester support unchunked extended messages
      - **Chunked** bit in Extended Message Header = 0
      - **Number of Data Objects** field = 000b
   10. If either the UUT or the Tester doesn’t support unchunked extended messages
       - **Chunked** bit in Extended Message Header = 1
       - **Number of Data Objects** field = 7
       - The total number of data bytes is consistent with the **Number of Data Objects** field
       - The last 2 bytes of the 7th Data Object are 0
   11. For the SCEDB
       - Bits 0..1 of Voltage Regulation field = 00b or 01b
       - Bits 3..7 (Reserved) of Voltage Regulation filed are 0
       - Bits 3..7 (Reserved) of Compliance field are 0
       - Bits 3..7 (Reserved) of Touch Current field are 0
       - Touch Temp field is 0, 1 or 2
       - If bit 0 of Source Inputs field is 0, bit 1 is 0
       - Bits 3..7 (Reserved) of Source Inputs field are 0
       - Upper Nibble of Number of Batteries/Battery Slots field (Number of Hot Swappable Battery Slots) <= 4
       - Lower Nibble of Number of Batteries/Battery Slots field (Number of Fixed Batteries) <= 4
       - Bit 7 (Reserved) of Source PDP field is 0
   12. If any field of Peak Current1, Peak Current2, or Peak Current3 in the SCEDB is not 0, the Tester verifies Peak Current is set to 0 in the Source Capabilities message from the UUT.

e) Rerun the test, with Unchunked Extended Messages Supported set to 0 in step a)
TD. PD. SRC3. E7. Battery Status sent timely

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:
6.4.5#1, 6.12.2#4, 8.3.3.11.2.1#1, 8.3.3.11.2.1#2

Description:
As Consumer (UFP), the Tester verifies that the UUT replies Get_Battery_Status message with a Battery_Status message timely.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a Get_Source_Cap_Extended message to the UUT.

c) If a Not_Supported message is received, and Num_Fixed_Batteries and Num_Swappable_Battery_Slots in the VIF are 0, the test passes and stops here.

d) If the Number of Batteries/Battery Slots field in the returned SourceCapabilities_Extended message is 0, the test passes and stops here.

e) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a Get_Battery_Status message to the UUT.

f) If a Battery_Status message is not received within tReceiverResponse max, the test fails. This delay is measured from the time the last bit of Get_Battery_Status message EOP has been transmitted to the time the first bit of the Battery_Status message preamble has been received.

TD. PD. SRC3. E8. Battery Status Fields Checks

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:
6.2.1.1.1#1, 6.2.1.1.8#1, 6.4.5#3, 6.4.5#4, 6.4.5#5, 6.4.5#6, 6.4.5.2#1, 6.4.5.2#2, 6.4.5.2.2#1, 6.4.5.2.2#2, 6.4.5.2.3#1, 6.4.5.2.3#2

Description:
As Consumer (UFP), the Tester sends a Get_Battery_Status message to the UUT, verifies the UUT respond with a Battery_Status or Not_Supported message. If a Battery_Status message is received, the Tester verifies correct field values.
Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a `Get_Source_Cap_Extended` message to the UUT.

c) If a `Source_Capabilities_Extended` message is received, the Tester record the `Number of Batteries/Battery Slots` field. If a `Not_Supported` message is received, the Tester reads the `Number of Batteries/Battery Slots field` (combine `Num_Fixed_Batteries` and `Num_Swappable_Battery_Slots`) from the VIF.

d) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a `Get_Battery_Status` message to the UUT.

e) The Tester verifies a `Not_Supported` or `Battery_Status` message is received within `tReceiverResponse` max, and the UUT always replies with the same message type. i.e. The UUT cannot replies with a `Not_Supported` message sometimes and `Battery_Status` message other times.

f) If a `Battery_Status` message is received, the Tester verifies:
   1. `Number of Data Objects` field = 001b
   2. `Port Power Role` field = 1b (Source)
   3. `Specification Revision` field = 10b (Rev 3.0)
   4. `Port Data Role` field = 1b (DFP)
   5. `Extended` = 0b
   6. `Invalid Battery Reference` field (Bit 0) of the Battery Info field in the BSDO matches with the recorded `Number of Batteries/Battery Slots` field
   7. If `Battery Status Ref` referred to a fixed battery and `Invalid Battery Reference` field is 0, the `Battery is present` field (Bit 1) shall be 1
   8. If `Invalid Battery Reference` field is 1, `Battery is present` field shall be 0
   9. If Battery is present, `Battery charging status` (Bits 3..2) of Battery Info field is not 11b
   10. If Battery is not present, Bits 3..2 of Battery Info field is 00b
   11. Bits 7..4 of Battery Info field are 0
   12. Bits 7..0 of the BSDO are 0

  g) Rerun steps d) to f) with `Battery Status Ref` set to 1 - 7


Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.2.1.1.1#1, 6.2.1.1.8#1, 6.4.5.2#1, 6.4.5.2#2, 6.4.5.2.1#1
Power Delivery 3.0 Tests

Description:

As Consumer (UFP), the Tester sends a *Get_Battery_Status* message with an invalid battery reference to the UUT, waits for a *Battery_Status* message from the Provider (DFP, UUT) and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a *Get_Source_Cap_Extended* message to the UUT.

c) If a *Source_Capabilities_Extended* message is returned, the Tester records the *Number of Batteries/Battery Slots* field. If a *Not_Supported* message is received, the Tester reads the *Number of Batteries/Battery Slots* field (combine Num_Fixed_Batteries and Num_Swappablable_Battery_Slots) from the VIF.

d) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a *Get_Battery_Status* message, with *Battery Status Ref* set to 8, to the UUT.

e) If a *Not_Supported* message is received, and
   1. If the recorded Number of Batteries/Battery Slots field is 0, the test passes and stops here.
   2. If the recorded Number of Batteries/Battery Slots field is not 0, the test fails.

f) Upon receipt of the *Battery_Status* message, the Tester verifies:
   1. *Number of Data Objects* field = 001b
   2. *Port Power Role* field = 1b (Source)
   3. *Specification Revision* field = 10b (Rev 3.0)
   4. *Port Data Role* field = 1b (DFP)
   5. *Extended* = 0b
   6. Invalid Battery Reference field (Bit 0) of the Battery Info field in the BSDO is 1
   7. Bits 7..1 of Battery Info field in the BSDO are 0
   8. Bits 7..0 of the BSDO are 0

**TD. PD. SRC3. E10. Unrecognized Message Received in Ready State**

Applicable to:

- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.3.14#1, 6.8.1#6, 8.3.3.5.1.1#1, 8.3.3.5.1.1#2
Description:

As Consumer (UFP), after setting up a contract with the UUT, the Tester sends an unrecognized message, with a reserved message type, to the UUT, and verifies the UUT replies with a Not_Supported message.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a control message, with Message Type set to 01110 (Reserved), to the UUT.

c) If a Not_Supported message is not received within tReceiverResponse max, the test fails. This delay is measured from the time the last bit of unrecognized message EOP has been transmitted to the time the first bit of the Not_Supported message preamble has been received.

TD. PD. SRC3. E11. Get_Status Fields Checks

Applicable to:

- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.2.1.1.1#1, 6.2.1.1.5#7, 6.2.1.1.8#1, 8.3.3.7.4.1#1, 8.3.3.7.4.1#2, 8.3.3.7.4.1#3, 8.3.3.9.3.1#1, 8.3.3.9.3.1#2

Description:

As Consumer (UFP), after setting up a contract with the UUT, the Tester sends an Alert message to the UUT, waits for a Get_Status message from the UUT and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends an Alert message to the UUT, with bit 5 of Type of Alert set to 1 (Source Input Change Event).

c) If a Not_Supported message is received, the test passes and stops here.

d) If a Get_Status message is not received within 500 ms after the Alert message was sent, the test passes and stops here.

e) Upon receipt of the Get_Status message, the Tester verifies:

   1. Number of Data Objects field = 000b
   2. Port Power Role field = 1b (Source)
   3. Specification Revision field = 10b (Rev 3.0)
   4. Port Data Role field = 1b (DFP)
   5. Extended = 0b
TD. PD. SRC3. E12. Get_Battery_Status Fields Checks

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:
6.2.1.1.1#1, 6.2.1.1.1#2, 6.2.1.1.1#4, 6.2.1.1.2#3, 6.2.1.1.2#4, 6.2.1.1.5#7, 6.2.1.1.8#1, 6.2.1.2#1, 6.2.1.2#3, 6.2.1.2.1#1, 6.2.1.2.1#2, 6.2.1.2.1#4, 6.2.1.2.1#6, 6.2.1.2.1#7, 6.2.1.2.2#1, 6.2.1.2.2#2, 6.2.1.2.3#1, 6.2.1.2.3#3, 6.2.1.2.4#1, 6.2.1.2.4#3, 6.5#1, 6.5.4#2, 6.5.4#3, 8.3.3.7.4.1#1, 8.3.3.7.4.1#2, 8.3.3.7.4.1#3, 8.3.3.11.1.1#1, 8.3.3.11.1.1#2

Description:
As Consumer (UFP), after setting up a contract with the UUT, the Tester sends an Alert message to the UUT, waits for a Get_Battery_Status message from the UUT and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported to 1 in Request message during this process.
   b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends an Alert message to the UUT, with bit 1 (Battery Status Change Event) of Type of Alert set to 1, and B20 set to 1 (Battery 0).
   c) If a Not_Supported message is received, the test passes and stops here.
   d) If a Get_Battery_Status message is not received within 500 ms after the Alert message was sent, the test passes and stops here.
   e) Upon receipt of the Get_Battery_Status message, the Tester verifies:
      1. Port Power Role field = 1b (Source)
      2. Specification Revision field = 10b (Rev 3.0)
      3. Port Data Role field = 1b (DFP)
      4. Extended = 1b
      5. Chunk Number in Extended Message Header = 0
      6. Request Chunk in Extended Message Header = 0
      7. Bit 9 (Reserved) in Extended Message Header = 0
      8. Data Size in Extended Message Header = 1
      9. If both the UUT and the Tester supports unchunked extended messages
         - Chunked bit in Extended Message Header = 0
         - Number of Data Objects field = 000b
      10. If either the UUT or the Tester doesn’t support unchunked extended messages
         - Chunked bit in Extended Message Header = 1
         - Number of Data Objects field = 001b
The total number of data bytes is consistent with the *Number of Data Objects* field.

- The last byte of the Data Object is 0

11. **Battery Status Ref** = 0

   f) Rerun the test, with Unchunked Extended Messages Supported set to 0 in step a)

### TD. PD. SRC3. E13. Status sent timely

**Applicable to:**
- UUT: Provider Only, Provider / Consumer, Dual-role Port

**Assertions Tested:**

6.3.16#1, 6.5.2#1, 8.3.3.9.2.1#1, 8.3.3.9.2.1#2

**Description:**

As Consumer (UFP), the Tester verifies that the UUT replies *Get_Status* message with a *Status* message timely.

**Steps:**

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a *Get_Status* message to the UUT.

c) If a *Status* or *Not_Supported* message is not received within tReceiverResponse max, the test fails. This delay is measured from the time the last bit of *Get_Status* message EOP has been transmitted to the time the first bit of the *Status* message preamble has been received.

### TD. PD. SRC3. E14. Status Fields Checks

**Applicable to:**
- UUT: Provider Only, Provider / Consumer, Dual-role Port

**Assertions Tested:**

6.2.1.1.1#1, 6.2.1.1.1#2, 6.2.1.1.1#4, 6.2.1.1.2#3, 6.2.1.1.2#4, 6.2.1.1.5#7, 6.2.1.1.8#1, 6.2.1.2#1, 6.2.1.2#3, 6.2.1.2.1#1, 6.2.1.2.1#2, 6.2.1.2.1#4, 6.2.1.2.1#6, 6.2.1.2.1#7, 6.2.1.2.2#1, 6.2.1.2.2#2, 6.2.1.2.3#1, 6.2.1.2.3#3, 6.2.1.2.4#1, 6.2.1.2.4#3, 6.5#1, 6.5.2#2, 6.5.2#4, 6.5.2.4#2, 6.5.2.5#1

**Description:**

As Consumer (UFP), the Tester sends a *Get_Status* message to the UUT, waits for a *Status* message from the UUT and verifies correct field values.
Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported to 1 in Request message during this process.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a Get_Status message to the UUT.

c) If a Not_SUPPORTED message is received, the test passes and stops here.

d) Upon receipt of the Status message, the Tester verifies:
   1. Port Power Role field = 1b (Source)
   2. Specification Revision field = 10b (Rev 3.0)
   3. Port Data Role field = 1b (DFP)
   4. Extended = 1b
   5. Chunk Number in Extended Message Header = 0
   6. Request Chunk in Extended Message Header = 0
   7. Bit 9 (Reserved) in Extended Message Header = 0
   8. Data Size in Extended Message Header = 5
   9. If both the UUT and the Tester support unchunked extended messages
      - Chunked bit in Extended Message Header = 0
      - Number of Data Objects field = 000b
   10. If either the UUT or the Tester doesn’t support unchunked extended messages
      - Chunked bit in Extended Message Header = 1
      - Number of Data Objects field = 2
      - The total number of data bytes is consistent with the Number of Data Objects field
      - The last 1 byte of the 2nd Data Object are 0

11. For the SDB
      - Bit 0 of Present Input field = 0
      - Bits 5..7 of Present Input field are 0
      - If bit 3 of Present Input field is 0, Present Battery Input field shall be 0
      - Bit 0 and Bits 5..7 of Event Flags field are 0
      - OVT Event is not set in Event Flags field
      - Bit 0, Bits 3..7 of Temperature Status field are 0
      - If OTP event is set in Event Flags, Temperature Status shall be Over temperature. And vice versa.

e) Rerun the test, with Unchunked Extended Messages Supported set to 0 in step a)
**TD. PD. SRC3. E15. Battery_Capabilities sent timely**

Applicable to:

- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.5.3#1, 6.12.3#3, 8.3.3.10.2.1#1, 8.3.3.10.2.1#2

Description:

As Consumer (UFP), the Tester verifies that the UUT replies `Get_Battery_Cap` message with a `Battery_Capabilities` message timely.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a `Get_Source_Cap_Extended` message to the UUT.

c) If a *Not_Supported* message is received, and the Num_Fixed_Batteries and Num_Swappable_Battery_Slots in the VIF are 0, the test passes and stops here.

d) If the Number of Batteries/Battery Slots field in the returned `Source_Capabilities_Extended` message is 0, the test passes and stops here.

e) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a `Get_Battery_Cap` message to the UUT.

f) If a `Battery_Capabilities` message is not received within tReceiverResponse max, the test fails. This delay is measured from the time the last bit of `Get_Battery_Cap` message EOP has been transmitted to the time the first bit of the `Battery_Capabilities` message preamble has been received.

**TD. PD. SRC3. E16. Battery_Capabilities Fields Checks**

Applicable to:

- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.2.1.1.1#1, 6.2.1.1.1#2, 6.2.1.1.1#4, 6.2.1.1.2#4, 6.2.1.1.5#7, 6.2.1.1.8#1, 6.2.1.2#1, 6.2.1.2#3, 6.2.1.2.1#1, 6.2.1.2.1#2, 6.2.1.2.1#4, 6.2.1.2.1#6, 6.2.1.2.1#7, 6.2.1.2.2#1, 6.2.1.2.2#2, 6.2.1.2.3#1, 6.2.1.2.3#3, 6.2.1.2.4#1, 6.2.1.2.4#3, 6.5#1, 6.5.5#2, 6.5.5.3.2#1

Description:

As Consumer (UFP), the Tester sends a `Get_Battery_Cap` message, verifies the UUT respond with a `Battery_Capabilities` or *Not_Supported* message. If a `Battery_Capabilities` is received, the Tester verifies correct field values.
Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported to 1 in Request message during this process.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a Get_Source_Cap_Extended message to the UUT.

c) If a Source_Capabilities_Extended message is received, the Tester record the Number of Batteries/Battery Slots field. If a Not_Supported message is received, the Tester reads the Number of Batteries/Battery Slots field (combine Num_Fixed_Batteries and Num_Swappable_Battery_Slots) from the VIF.

d) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a Get_Battery_Cap message to the UUT, with Battery Cap Ref set to 0.

e) The Tester verifies a Not_Supported or Battery_Capabilities message is received within tReceiverResponse max, and the UUT always replies with the same message type. i.e. The UUT cannot replies with a Not_Supported message sometimes and Battery_Capabilities message other times.

f) If a Battery_Capabilities message is received, the Tester verifies:

1. Port Power Role field = 1b (Source)
2. Specification Revision field = 10b (Rev 3.0)
3. Port Data Role field = 1b (DFP)
4. Extended = 1b
5. Chunk Number in Extended Message Header = 0
6. Request Chunk in Extended Message Header = 0
7. Bit 9 (Reserved) in Extended Message Header = 0
8. Data Size in Extended Message Header = 9
9. If both the UUT and the Tester supports unchunked extended messages
   – Chunked bit in Extended Message Header = 0
   – Number of Data Objects field = 000b
10. If either the UUT or the Tester doesn’t support unchunked extended messages
   – Chunked bit in Extended Message Header = 1
   – Number of Data Objects field = 3
   – The total number of data bytes is consistent with the Number of Data Objects field
   – The last byte of the 3rd Data Object is 0
11. For the BCDB
   – Invalid Battery reference bit of Battery Type field matches with the recorded Number of Batteries/Battery Slots field
   – Bits 1..7 of Battery Type field are 0

g) Rerun steps d) to f) with Battery Cap Ref set to 1 - 7
h) Rerun the test, not including step g), with Unchunked Extended Messages Supported set to 0 in step a)

**TD. PD. SRC3. E17. Battery_Capabilities Fields Checks - Invalid Battery Reference**

Applicable to:

- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.2.1.1.1#1, 6.2.1.1.1#2, 6.2.1.1.1#4, 6.2.1.1.2#3, 6.2.1.1.2#4, 6.2.1.1.5#7, 6.2.1.1.8#1, 6.2.1.2#1, 6.2.1.2#3, 6.2.1.2.1#1, 6.2.1.2.1#2, 6.2.1.2.1#4, 6.2.1.2.1#6, 6.2.1.2.1#7, 6.2.1.2.2#1, 6.2.1.2.2#2, 6.2.1.2.3#1, 6.2.1.2.3#3, 6.2.1.2.4#1, 6.2.1.2.4#3, 6.5#1, 6.5.5#2, 6.5.5.3.1#1

Description:

As Consumer (UFP), the Tester sends a **Get_Battery_Cap** message with an invalid Battery Reference to the UUT, waits for a **Battery_Capabilities** message from the UUT, and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported to 1 in Request message during this process.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a **Get_Source_Cap_Extended** message to the UUT.

c) If a **Source_Capabilities_Extended** message is returned, the Tester records the Number of Batteries/Battery Slots field. If a **Not_Supported** message is received, the Tester reads the Number of Batteries/Battery Slots field (combine Num_Fixed_Batteries and Num_Swappable_Battery_Slots) from the VIF.

d) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a **Get_Battery_Cap** message, in which Battery Cap Ref is set to 8, to the UUT.

e) If a **Not_Supported** message is received, and

   1. If the recorded Number of Batteries/Battery Slots field is 0, the test passes and stops here.
   2. If the recorded Number of Batteries/Battery Slots field is not 0, the test fails.

f) Upon receipt of the **Battery_Capabilities** message, the Tester verifies:

   1. **Port Power Role** field = 1b (Source)
   2. **Specification Revision** field = 10b (Rev 3.0)
   3. **Port Data Role** field = 1b (DFP)
   4. **Extended** = 1b
   5. **Chunk Number** in Extended Message Header = 0
   6. **Request Chunk** in Extended Message Header = 0
   7. Bit 9 (**Reserved**) in Extended Message Header = 0
8. **Data Size** in Extended Message Header = 9
9. If both the UUT and the Tester support unchunked extended messages
   - **Chunked** bit in Extended Message Header = 0
   - **Number of Data Objects** field = 000b
10. If either the UUT or the Tester doesn’t support unchunked extended messages
    - **Chunked** bit in Extended Message Header = 1
    - **Number of Data Objects** field = 3
    - The total number of data bytes is consistent with the **Number of Data Objects** field
    - The last byte of the 3rd Data Object is 0
11. For the BCDB
    - Invalid Battery reference bit of Battery Type field = 1
    - Bits 1..7 of Battery Type field are 0
    g) Rerun the test, with Unchunked Extended Messages Supported set to 0 in step a)

**TD. PD. SRC3. E18. Manufacturer_Info Sent Timely**

Applicable to:
- **UUT**: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.5.6#1, 6.5.7#1, 8.3.3.12.2.1#1, 8.3.3.12.2.1#2

Description:
As Consumer (UFP), the Tester verifies that the UUT replies **Get_Manufacturer_Info** message with a **Manufacturer_Info** message timely.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a **Get_Manufacturer_Info** message to the UUT.

c) If a **Manufacturer_Info** or **Not_Supported** message is not received within **tReceiverResponse** max, the test fails. This delay is measured from the time the last bit of **Get_Manufacturer_Info** message EOP has been transmitted to the time the first bit of the **Manufacturer_Info** message preamble has been received.
TD. PD. SRC3. E19. Manufacturer_Info Fields Checks

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

 Assertions Tested:

 6.2.1.1.1#1, 6.2.1.1.1#2, 6.2.1.1.1#4, 6.2.1.1.2#3, 6.2.1.1.2#4, 6.2.1.1.5#7, 6.2.1.1.8#1, 6.2.1.2#1,
 6.2.1.2#3, 6.2.1.2.1#1, 6.2.1.2.1#2, 6.2.1.2.1#4, 6.2.1.2.1#6, 6.2.1.2.1#7, 6.2.1.2.2#1, 6.2.1.2.2#2,
 6.2.1.2.3#1, 6.2.1.2.3#3, 6.2.1.2.4#1, 6.2.1.2.4#3, 6.5#1

Description:

As Consumer (UFP), the Tester sends a Get_Manufacturer_Info message to the UUT, wait for a
Manufacturer_Info message from the UUT and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended
   Messages Supported to 1 in Request message during this process.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a
   Get_Manufacturer_Info message, in which Manufacturer_Info Target is set to 0, to the UUT.

c) If a Not_Supported message is received, the test passes and stops here.

d) Upon receipt of the Manufacturer_Info message, the Tester verifies:
   1. Port Power Role field = 1b (Source)
   2. Specification Revision field = 10b (Rev 3.0)
   3. Port Data Role field = 1b (DFP)
   4. Extended = 1b
   5. Chunk Number in Extended Message Header = 0
   6. Request Chunk in Extended Message Header = 0
   7. Bit 9 (Reserved) in Extended Message Header = 0
   8. Data Size in Extended Message Header is 4..26
   9. If both the UUT and the Tester support unchunked extended messages
      - Chunked bit in Extended Message Header = 0
      - Number of Data Objects field = 000b
   10. If either the UUT or the Tester doesn’t support unchunked extended messages
      - Chunked bit in Extended Message Header = 1
      - Number of Data Objects field = (Data Size + 5) / 4
      - The Packet payload is padded to the next 4-byte Data Object boundary with zeros (0x00).

e) Rerun the test, with Unchunked Extended Messages Supported set to 0 in step a)
TD. PD. SRC3. E20. Manufacturer_Info Fields Checks - Invalid Manufacturer Info Target

Applicable to:

- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.2.1.1.1#1, 6.2.1.1.1#2, 6.2.1.1.2#3, 6.2.1.1.2#4, 6.2.1.1.5#7, 6.2.1.1.8#1, 6.2.1.2#1, 6.2.1.2#3, 6.2.1.2.1#1, 6.2.1.2.1#2, 6.2.1.2.1#4, 6.2.1.2.1#6, 6.2.1.2.1#7, 6.2.1.2.2#1, 6.2.1.2.2#2, 6.2.1.2.3#1, 6.2.1.2.3#3, 6.2.1.2.4#1, 6.2.1.2.4#3, 6.5#1, 6.5.7#2, 6.5.7.3#2

Description:

As Consumer (UFP), the Tester sends a Get_Manufacturer_Info message, in which Manufacturer Info Target is set to 2, to the UUT, wait for a Manufacturer_Info message from the UUT and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported to 1 in Request message during this process.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a Get_Manufacturer_Info message, in which Manufacturer Info Target is set to 2, to the UUT.

c) If a Not_Supported message is received, the test passes and stops here.

d) Upon receipt of the Manufacturer_Info message, the Tester verifies:

1. **Port Power Role** field = 1b (Source)
2. **Specification Revision** field = 10b (Rev 3.0)
3. **Port Data Role** field = 1b (DFP)
4. **Extended** = 1b
5. **Chunk Number** in Extended Message Header = 0
6. **Request Chunk** in Extended Message Header = 0
7. Bit 9 (**Reserved**) in Extended Message Header = 0
8. **Data Size** in Extended Message Header is 4
9. If both the UUT and the Tester support unchunked extended messages
   - **Chunked** bit in Extended Message Header = 0
   - **Number of Data Objects** field = 000b
10. If either the UUT or the Tester doesn’t support unchunked extended messages
    - **Chunked** bit in Extended Message Header = 1
    - **Number of Data Objects** field = 2
    - The total number of data bytes is consistent with the **Number of Data Objects** field
    - The last 2 bytes of the 2\textsuperscript{nd} Data object are 0
e) Rerun the test, with Unchunked Extended Messages Supported set to 0 in step a)

**TD. PD. SRC3. E21. Manufacturer_Info Fields Checks - Invalid Manufacturer Info Ref**

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.2.1.1.1#1, 6.2.1.1.1#2, 6.2.1.1.1#4, 6.2.1.1.2#3, 6.2.1.1.2#4, 6.2.1.1.5#7, 6.2.1.1.8#1, 6.2.1.2#1, 6.2.1.2#3, 6.2.1.2.1#1, 6.2.1.2.1#2, 6.2.1.2.1#4, 6.2.1.2.1#6, 6.2.1.2.1#7, 6.2.1.2.2#1, 6.2.1.2.2#2, 6.2.1.2.3#1, 6.2.1.2.3#3, 6.2.1.2.4#1, 6.2.1.2.4#3, 6.5#1, 6.5.7#2, 6.5.7.3#2

Description:

As Consumer (UFP), the Tester sends a Get_Manufacturer_Info message with an invalid Manufacturer Info Ref to the UUT, waits for a Manufacturer_Info message from the UUT, and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported to 1 in Request message during this process.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a Get_Manufacturer_Info message, in which Manufacturer Info Target is set to 1 and Manufacturer Info Ref is set to 8, to the UUT.

c) If a Not_Supported message is received, the test passes and stops here.

d) Upon receipt of the Manufacturer_Info message, the Tester verifies:
   1. **Port Power Role** field = 1b (Source)
   2. **Specification Revision** field = 10b (Rev 3.0)
   3. **Port Data Role** field = 1b (DFP)
   4. **Extended** = 1b
   5. **Chunk Number** in Extended Message Header = 0
   6. **Request Chunk** in Extended Message Header = 0
   7. **Bit 9 (Reserved)** in Extended Message Header = 0
   8. **Data Size** in Extended Message Header is 4
   9. If both the UUT and the Tester support unchunked extended messages
      - **Chunked** bit in Extended Message Header = 0
      - **Number of Data Objects** field = 000b
   10. If either the UUT or the Tester doesn’t support unchunked extended messages
       - **Chunked** bit in Extended Message Header = 1
       - **Number of Data Objects** field = 2
Power Delivery 3.0 Tests

- The total number of data bytes is consistent with the Number of Data Objects field
- The last 2 bytes of the 2nd Data object are 0
e) Rerun the test, with Unchunked Extended Messages Supported set to 0 in step a)

TD. PD. SRC3. E22. Cable Type Detection

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:
4.4#1, 4.4#2, 4.4#5, 10.2.3.2#6

Description:
As Consumer (UFP), the Tester verifies that the UUT does cable detection if it offers currents in excess of 3A and/or voltages in excess of 20V.

Steps:
a) If Captive_Cable is set to YES in the VIF, the test passes and stops here.
b) The test starts in a disconnected state.
c) The Tester applies Rd and Ra and waits for a connection with the UUT.
d) If a SOP' Discover Identity message is received, the Tester replies as Passive Cable with Vbus Current Handling Capability set to 5A and Maximum VBus Voltage set to 50V.
e) Upon receipt of the Source_Capabilities message from the UUT, the Tester checks whether the UUT offers currents in excess of 3A and/or voltages in excess of 20V. If so and no SOP' Discovery Identity message was received, the test fails; if not, the test passes and stops here.
f) The Tester removes all terminations and simulates a disconnection
g) The Tester applies Rd and Ra and waits for a connection with the UUT.
h) If a SOP' Discover Identity message is received, the Tester replies as Passive Cable with Vbus Current Handling Capability set to 3A and Maximum VBus Voltage set to 20V.
i) Upon receipt of the Source_Capabilities message from the UUT, the Tester checks whether the UUT offers currents in excess of 3A and/or voltages in excess of 20V. If so, the test fails.

TD. PD. SRC3. E23. Vconn Swap

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:
6.3.3#4, 6.3.11#7, 6.3.11#8, 6.3.11#10, 6.3.11#11, 6.3.12.4#1, 6.3.12.4#2
Description:

As Consumer (UFP), the Tester verifies that the UUT accepts \textit{VCONN\_Swap} when it’s presently the Vconn source.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) In addition to emulating a UFP, the Tester will also present Ra and will reply to \textit{SOP’ Discover Identity} as a Passive Cable with Vbus Current Handling Capability set to 5A and Maximum VBus Voltage set to 20V.

c) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a \textit{VCONN\_Swap} message to the UUT.

d) If an \textit{Accept} or \textit{Not\_Supported} message is not received within \textit{tReceiverResponse} max, the test fails. This delay is measured from the time the last bit of \textit{VCONN\_Swap} message \textit{EOP} has been transmitted to the time the first bit of the \textit{Accept} or \textit{Not\_Supported} message preamble has been received.

e) The Tester verifies the UUT never sources VConn if the message received in d) is \textit{Not\_Supported}.

TD. PD. SRC3. E24. Unexpected Message Received in Ready State

Applicable to:

- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.8.1#4

Description:

As Consumer (UFP), after setting up a contract with the UUT, the Tester sends a unexpected message, to the UUT, and verifies the UUT does Soft Reset.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends an \textit{Accept Message} to the UUT.

c) If a \textit{Soft Reset} message is not received within \textit{tProtErrSoftReset} max, the test fails. This delay is measured from the \textit{EOP} of the \textit{GoodCRC} corresponding to the \textit{Accept message} has been received to the time the \textit{EOP} the \textit{Soft Reset} message has been received.
TD. PD. SRC3. E25. Receiving chunked extended message

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.5.7#1, 6.6.17.1#1, 6.6.17.1#2, 6.11.2.1.1.1#1, 6.11.2.1.2.1#3, 6.11.2.1.2.2#1, 6.11.2.1.2.3#2, 6.11.2.1.2.3#3, 6.11.2.1.2.4#1, 6.11.2.1.2.4#2, 6.11.2.1.2.5#2, 8.3.3.5.1.3#1, 8.3.3.5.1.3#2, 8.3.3.5.1.3#3, 8.3.3.12.2.1#1, 8.3.3.12.2.1#2

Description:

As Consumer (UFP), the Tester verifies that the UUT receives a chunked extended message correctly by sending messages to request chunks.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported set to 0 in Request message during this process.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends the first chunk of a chunked extended message to the UUT, with Data Size set to 260 and Message Type set to 1111b. Bytes 0 to 259 of data block contain incrementing values (mod 256) starting at 0x00.

c) If a message is not received within tChunkingNotSupported max, this test fails. The delay is measured from the time the last bit of the EOP of the chunk has been transmitted until the first bit of the response Message Preamble has been received.

d) If the received message is Not_Supported, the Tester verifies the message is received after tChunkingNotSupported min and stops here.

e) If the message is not received within tChunkReceiverRequest max, the test fails.

f) Upon receipt of the response from the UUT, the Tester verifies the following:

1. For Message Header
   - Extended = 1
   - Number of Data Objects = 1
   - Port Power Role field = 1b (Source)
   - Port Data Role field = 1b (DFP)
   - Specification Revision = 10b (Rev 3.0)
   - Message Type = 11111b

2. For Extended Message Header
   - Chunked = 1
   - Chunk Number = next chunk in the series
   - Request Chunk = 1
   - Bit 9 = 0 (Reserved)
Power Delivery 3.0 Tests

- **Data Size** = 0
  
g) The Tester sends the requested chunk to the UUT.
  
h) Repeat e) and g) until the Tester has finished sending all 10 chunks.

**TD. PD. SRC3. E26. Soft_Reset sent regardless of Rp value**

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:
6.8.1#7

Description:
As Consumer (UFP), the Tester forces the UUT to send Soft_Reset and verifies Soft_Reset is sent regardless of the Rp value is SinkTxOK or SinkTxNG.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a `Get_Source_Cap` message to the UUT.

c) Upon receipt of the `Source_Capabilities` Message, the Tester doesn’t reply with GoodCRC.

d) The Tester verifies that a `Soft_Reset` message is sent by the UUT within \( t_{Receive} \) max + \( t_{SoftReset} \) max, the delay is measured from the time the last bit of the last retransmitted `Source_Capabilities` Message EOP has been received to the time the last bit of the `Soft_Reset` message EOP has been received.

**TD. PD. SRC3. E27. PPS_Status Sent Timely**

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:
6.3.18#1, 6.5.10#1, 6.12.1#10, 6.12.3#10, 8.3.3.9.6.1#1, 8.3.3.9.6.1#2

Description:
As Consumer (UFP), the Tester verifies that the UUT replies `Get_PPS_Status` message with a `PPS_Status` message timely.

Steps:

a) Run PROC.PD.E2 Bring-up procedure for PPS Tests according to the UUT role.
b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a `Get_PPS_Status` message to the UUT.

c) If a `PPS_Status` message is not received within `tReceiverResponse` max, the test fails. This delay is measured from the time the last bit of `Get_PPS_Status` message EOP has been transmitted to the time the first bit of the `PPS_Status` message preamble has been received.

**TD. PD. SRC3. E28. PPS_Status Fields Check**

Applicable to:

- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.5.10.3, 6.5.10.1#1, 6.5.10.1#2, 6.5.10.2#1, 6.5.10.2#2, 6.5.10.3#1, 6.5.10.3#2

Description:

As Consumer (UFP), the Tester sends a `Get_PPS_Status` message to the UUT, wait for a `PPS_Status` message from the UUT and verifies correct field values.

Steps:

a) Run PROC.PD.E2 Bring-up procedure for PPS Tests according to the UUT role. The Tester sets Unchunked Extended Messages Supported to 1 in `Request` message during this process.

b) The Tester doesn’t draw any current, waits until it can start an AMS (Run PROC.PD.E3) and sends a `Get_PPS_Status` message to the UUT.

c) Upon receipt of the `PPS_Status` message, the Tester verifies:

1. `Port Power Role` field = 1b (Source)
2. `Specification Revision` field = 10b (Rev 3.0)
3. `Port Data Role` field = 1b (DFP)
4. `Extended` = 1b
5. `Chunk Number` in Extended Message Header = 0
6. `Request Chunk` in Extended Message Header = 0
7. Bit 9 (Reserved) in Extended Message Header = 0
8. `Data Size` in Extended Message Header is 4

9. If both the UUT and the Tester support unchunked extended messages
   - `Chunked` bit in Extended Message Header = 0
   - `Number of Data Objects` field = 000b

10. If either the UUT or the Tester doesn’t support unchunked extended messages
    - `Chunked` bit in Extended Message Header = 1
    - `Number of Data Objects` field = 2
The total number of data bytes is consistent with the **Number of Data Objects** field.

The last 2 bytes of the 2\(^{nd}\) Data object are 0.

11. For the PPSSDB

- Output Voltage field shall be within ±3% of the actual output voltage rounded to the nearest 20mV or 0xFFFF
- Output Current field shall be 0x00 to 0x03(inclusive) or 0xFF
- For Real Time Flags field
  - Bit 0 shall be 0
  - PTF shall be 0x00 (Not Supported) or 0x01 (Normal)
  - OMF shall be 0 (Constant Voltage mode)
  - Bits 4..7 shall be 0

  d) Rerun the test, with Unchunked Extended Messages Supported set to 0 in step a)

**TD. PD. SRC3. E29. SourcePPSCCommTimer Deadline**

Applicable to:

- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.6.18.2#1, 6.6.18.2#2, 6.6.18.2#3, 6.6.18.2#4

Description:

As Consumer (UFP), the Tester sends the second Request message after \(t_{PPSRequest}\) max, verifies SourcePPSCCommTimer is not timeout at the UUT and the timer is correctly reinitialized.

Steps:

a) Run PROC.PD.E2 Bring-up procedure for PPS Tests according to the UUT role.

b) If a Hard Reset is received within \(t_{PPSRequest}\) max, the test fails. The delay is measured from the time the last bit of the EOP of the GoodCRC Message sent by the UUT in response to the previous Request Message to the time the first bit of the Hard Reset is received.

c) The Tester sends the second Request Message to request the PPS APDO at 4V 1A, \(t_{PPSRequest}\) max after the previous Request Message, and verifies the UUT responds with an Accept and then PS_RDY message. If the Tester cannot send the second Request Message because it cannot start an AMS, the test fails.

  d) If a Hard Reset is received within \(t_{PPSRequest}\) max, the test fails. The delay is measured from the time the last bit of the EOP of the GoodCRC Message sent by the UUT in response to the second Request Message to the time the first bit of the Hard Reset is received.
TD. PD. SRC3. E30. SourcePPSCommTimer Timeout

Applicable to:
  - UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:
  6.4.1.2#10, 6.6.18.2#1, 6.6.18.2#2, 6.6.18.2#5, 8.3.3.2.6#6

Description:
  As Consumer (UFP), the Tester intentionally does not send the second Request message, in order to force a SourcePPSCommTimer timeout on the UUT and verifies it is correctly implemented.

Steps:
   a) Run PROC.PD.E2 Bring-up procedure for PPS Tests according to the UUT role.
   b) The Tester intentionally does not send the second Request Message, if a Hard Reset is received before tPPSTimeout min, or not received within tPPSTimeout max, the test fails. The delay is measured from the time the last bit of the EOP of the GoodCRC Message sent by the UUT in response to the previous Request Message to the time the first bit of the Hard Reset is received.

TD. PD. SRC3. E31. SourcePPSCommTimer Stopped

Applicable to:
  - UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:
  6.4.1.2#11, 6.6.18.2#1

Description:
  As Consumer (UFP), the Tester request a PPS APDO first and then request a fixed PDO, verifies SourcePPSCommTimer is stopped at the UUT.

Steps:
   a) Run PROC.PD.E2 Bring-up procedure for PPS Tests according to the UUT role.
   b) The Tester wait till it can start an AMS and sends the second Request Message to request the fixed PDO at 5V 0.5A, and verifies the UUT responds with an Accept and then PS_RDY message.
   c) If a Hard Reset is received within tPPSTimeout max, the test fails. The delay is measured from the time the last bit of the EOP of the GoodCRC Message sent by the UUT in response to the second Request Message to the time the first bit of the Hard Reset is received.
TD. PD. SRC3. E32. ChunkSenderResponseTimer Timeout

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.6.17.3#1, 6.6.17.3#2, 6.6.17.3#5, 6.11.2.1.2.5#1, 6.11.2.1.2.5#3, 6.11.2.1.2.6#4

Description:

As Consumer (UFP), the Tester verifies that the UUT recovers correctly after the Tester stops sending chunked messages in the middle.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported set to 0 in Request message during this process.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends the first chunk of a chunked extended message to the UUT, with Data Size set to 260 and Message Type set to 11111b. Bytes 0 to 259 of data block contain incrementing values (mod 256) starting at 0x00.

c) If a message is not received within tChunkingNotSupported max, this test fails. The delay is measured from the time the last bit of the EOP of the chunk has been transmitted until the first bit of the response Message Preamble has been received.

d) If the received message is Not_Supported, the Tester verifies the message is received after tChunkingNotSupported min and stops here.

e) If the message is not received within tChunkReceiverRequest max, the test fails.

f) Upon receipt of the message from the UUT to request for the next chunk, the Tester sends the requested chunk to the UUT.

g) Repeat f) until the Tester has finished sending 4 chunks and intentionally does not send the 5th chunk to the UUT.

h) The Tester waits for tChunkSenderResponse max + 5 ms, waits until it can start an AMS (Run PROC.PD.E3) and sends the first chunk to the UUT.

i) If a message is not received within tChunkReceiverRequest max, the test fails.

j) Upon receipt of the message, the Tester verifies the following:

1. For Message Header
   - Extended = 1
   - Number of Data Objects = 1
   - Port Power Role field = 1b (Source)
   - Port Data Role field = 1b (DFP)
   - Specification Revision = 10b (Rev 3.0)
   - Message Type = 11111b

2. For Extended Message Header
Power Delivery 3.0 Tests

- **Chunked** = 1
- **Chunk Number** = 1
- **Request Chunk** = 1
- **Bit 9** = 0 (Reserved)
- **Data Size** = 0

3. The total number of data bytes is consistent with the *Number of Data Objects* field

4. The last 2 bytes of the Data Object are 0

**TD. PD. SRC3. E33. Country_Codes Sent Timely**

Applicable to:

- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.3.19#1, 6.5.11#1, 8.3.3.13.2.1#1, 8.3.3.13.2.1#2

Description:

As Consumer (UFP), the Tester verifies that the UUT replies *Get_Country_Codes* message with a *Country_Codes* or *Not_Supported* message timely.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a *Get_Country_Codes* message to the UUT.

c) If a *Country_Codes* or *Not_Supported* message is not received within \(t\text{ReceiverResponse}\) max, the test fails. This delay is measured from the time the last bit of *Get_Country_Codes* message EOP has been transmitted to the time the first bit of the *Country_Codes* message preamble has been received.

**TD. PD. SRC3. E34. Country_Codes Fields Checks**

Applicable to:

- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:

6.5.11#2, 6.5.11.1#1

Description:

As Consumer (UFP), the Tester sends a *Get_Country_Codes* message to the UUT, wait for a *Country_Codes* message from the UUT and verifies correct field values.
Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported to 1 in Request message during this process.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a Get_Country_Codes message to the UUT.

c) If a Not_Supported message is received, the test passes and stops here.

d) Upon receipt of the Country_Codes message, the Tester verifies:
   1. Port Power Role field = 1b (Source)
   2. Specification Revision field = 10b (Rev 3.0)
   3. Port Data Role field = 1b (DFP)
   4. Extended = 1b
   5. Request Chunk in Extended Message Header = 0
   6. Bit 9 (Reserved) in Extended Message Header = 0
   7. Data Size in Extended Message Header is 4..260, and is the same as that in the first Chunk
   8. If both the UUT and the Tester support unchunked extended messages
      a) Chunked bit in Extended Message Header = 0
      b) Number of Data Objects field = 000b
      c) Chunk Number in Extended Message Header = 0
      d) The actual bytes in CCDB matches with Data Size
   9. If either the UUT or the Tester doesn’t support unchunked extended messages
      a) Chunked bit in Extended Message Header = 1
      b) Chunk Number in Extended Message Header is correctly set
      c) Number of Data Objects field = 7 or (Remaining Data Size + 5) / 4 (whichever is smaller)
      d) The Packet payload is padded to the next 4-byte Data Object boundary with zeros (0x00) if this is the last Chunk.
   10. For the CCDB
      a) Length = (Data Size) / 2 – 1
      b) All other bytes are ascii codes of capital letters (65 – 90 inclusive)
   11. The Tester Requests for the next Chunk if the message is chunked and this is not the last Chunk, and go to step d)

e) Rerun the test, with Unchunked Extended Messages Supported set to 0 in step a)
TD. PD. SRC3. E35. Country_Info Sent Timely

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:
6.5.12#1, 8.3.3.13.4.1#1, 8.3.3.13.4.1#2

Description:
As Consumer (UFP), the Tester verifies that the UUT replies Get_Country_Info message with a Country_Info or Not_Supported message timely.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.
b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a Get_Country_Codes message to the UUT.
c) If a Not_Supported message is received, the test passes and stops here.
d) For each Country Code, the Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a Get_Country_Info message to the UUT.
e) If a Country_Info message is not received within tReceiverResponse max, the test fails. This delay is measured from the time the last bit of Get_Country_Info message EOP has been transmitted to the time the first bit of the Country_Info message preamble has been received.
f) If the Country_Info is chunked, the Tester finishes receiving the message.

TD. PD. SRC3. E36. Country_Info Fields Checks

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:
6.5.12#2, 6.5.12.1#1

Description:
As Consumer (UFP), the Tester sends a Get_Country_Info message to the UUT, wait for a Country_Info message from the UUT and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported to 1 in Request message during this process.
b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a `Get_Country_Codes` message to the UUT.

c) If a `Not_Supported` message is received, the test passes and stops here.

d) For the first Country Code, the Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a `Get_Country_Info` message to the UUT.

e) Upon receipt of the `Country_Info` message, the Tester verifies:
   1. `Port Power Role` field = 1b (Source)
   2. `Specification Revision` field = 10b (Rev 3.0)
   3. `Port Data Role` field = 1b (DFP)
   4. `Extended` = 1b
   5. `Request Chunk` in Extended Message Header = 0
   6. Bit 9 (Reserved) in Extended Message Header = 0
   7. `Data Size` in Extended Message Header is 4..260, and is the same as that in the first Chunk
   8. If both the UUT and the Tester support unchunked extended messages
      – `Chunked` bit in Extended Message Header = 0
      – `Number of Data Objects` field = 000b
      – `Chunk Number` in Extended Message Header = 0
      – The actual bytes in CIDB matches with `Data Size`
   9. If either the UUT or the Tester doesn’t support unchunked extended messages
      – `Chunked` bit in Extended Message Header = 1
      – `Chunk Number` in Extended Message Header is correctly set
      – `Number of Data Objects` field = 7 or (Remaining Data Size + 5) / 4 (whichever is smaller)
      – The Packet payload is padded to the next 4-byte Data Object boundary with zeros (0x00) if this is the last Chunk.
   10. For the CIDB
       – `Country Code` field is the same as that in the Get_Country_Info message
       – Byte 2 and Byte 3 are 0
   11. The Tester Requests for the next Chunk if the message is chunked and this is not the last Chunk, and go to step d)

f) Rerun the test, with Unchunked Extended Messages Supported set to 0 in step a)
Sink Tests

TD. PD. SNK3. E1. Request Fields Checks

Applicable to:
- UUT: Consumer Only, Consumer / Provider, Dual-role Port

Assertions Tested:
6.2.1.1.1#1, 6.2.1.1.5#1, 6.2.1.1.5#6, 6.2.1.1.8#1

Description:
As Provider, the Tester waits for a Request message from the UUT and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role until step e).
b) Upon receipt of the Request message, the Tester verifies:
   1. Number of Data Objects field = 001b
   2. MessageID field = 000b
   3. Port Power Role field = 0b (Sink)
   4. Port Data Role field = 0b (UFP)
   5. Specification Revision field = 10b (Rev 3.0)
   6. Extended Field = 0b
c) For the Sink Request Data Object, the Tester verifies:
   1. Object position field is valid.
   2. The Operating Current/Power is less than or equal to the maximum current/power offered in the Source Capabilities Message.
   3. If GiveBack flag is 0
      - The Maximum Operating Current/Power is larger than or equal to the Operating Current/Power.
      - If Capability Mismatch bit is 0, the Maximum Operating Current/Power is less than or equal to the maximum current/power offered in the Source Capabilities Message.
   4. If GiveBack flag is 1, the Minimum Operating Current/Power is less than the Operating Current/Power.
   5. Bit 31 = 0b (Reserved)
   6. Bits 22..20 = 000b (Reserved)
d) The test fails if any verification is not satisfied.
TD. PD. SNK3. E2. Unrecognized Message Received in Ready State

Applicable to:

- UUT: Consumer Only, Consumer / Provider, Dual-role Port

Assertions Tested:

6.3.14#1, 6.8.1#6, 8.3.3.5.2.1#1, 8.3.3.5.2.1#2

Description:

As Provider (DFP), after setting up a contract with the UUT, the Tester sends an unrecognized message, with a reserved Message Type, to the UUT, and verifies the UUT reply with a Not_Supported message.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester sends a control message, with Message Type set to 01110, to the UUT.

c) If a Not_Supported message is not received within tReceiverResponse max, the test fails. This delay is measured from the time the last bit of unrecognized message EOP has been transmitted to the time the first bit of the Not_Supported message preamble has been received.

TD. PD. SNK3. E3. Get_Source_Cap_Extended Fields Checks

Applicable to:

- UUT: Consumer Only, Consumer / Provider, Dual-role Port

Assertions Tested:

6.2.1.1.1#1, 6.2.1.1.5#7, 6.4.1.2.2.7#6, 6.11.2.2.3.1#1, 6.11.2.2.3.1#2, 6.11.2.2.3.2#1, 8.3.3.8.1.1#1, 8.3.3.8.1.1#2

Description:

As Provider (DFP), after setting up a contract with the UUT, the Tester waits for a Get_Source_Cap_Extended message from the Consumer (UUT) and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. In Source_Capabilities message, the Tester sets Peak Current of fixed supply PDOs to 0.

b) The Tester sets termination to SinkTxOk.

c) If a Get_Source_Cap_Extended message is not received within 500 ms after termination set to SinkTxOk, the test passes and stops here.
d) Upon receipt of the `Get_Source_Cap_Extended` message, the Tester verifies:

1. **Number of Data Objects** field = 000b
2. **Port Power Role** field = 0b (Sink)
3. **Specification Revision** field = 10b (Rev 3.0)
4. **Port Data Role** field = 0b (UFP)
5. **Extended** = 0b

**TD. PD. SNK3. E4. SenderResponseTimer Deadline - Source_Capabilities_Extended**

Applicable to:
- UUT: Consumer Only, Consumer / Provider, Dual-role Port

Assertions Tested:

8.3.3.8.1.1#1, 8.3.3.8.1.1#2, 8.3.3.8.1.1#3, 8.3.3.8.1.1#4

Description:

As Provider (DFP), the Tester verifies that the UUT accepts a `Source_Capabilities_Extended` message sent at the deadline limit of `tSenderResponse` min in reply to `Get_Source_Cap_Extended` message.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. In `Source_Capabilities` message, the Tester sets Peak Current of fixed supply PDOs to 0.

b) The Tester sets termination to *SinkTxOk*.

c) If a `Get_Source_Cap_Extended` message is not received within 500 ms after termination set to *SinkTxOk*, the test passes and stops here.

d) Upon receipt of the `Get_Source_Cap_Extended` message from the UUT, the Tester replies with a *GoodCRC* message.

e) The Tester sends a `Source_Capabilities_Extended` message at the deadline limit of `tSenderResponse` min after the *GoodCRC* message. This delay is measured from the time the last bit of the *GoodCRC* message has been transmitted to the time the last bit of the `Source_Capabilities_Extended` message has been transmitted.

f) The Tester verifies a *GoodCRC* message corresponding to the `Source_Capabilities_Extended` message is received.

g) The Tester verifies a *Soft Reset* is not detected after `tProtErrSoftReset` max from the time the EOP of the *GoodCRC* sent in response to the `Source_Capabilities_Extended` message has been received.
TD. PD. SNK3. E5. SenderResponseTimer Timeout - Source_Capabilities_Extended

Applicable to:
- UUT: Consumer Only, Consumer / Provider, Dual-role Port

Assertions Tested:
8.3.3.8.1.1#1, 8.3.3.8.1.1#2, 8.3.3.8.1.1#3, 8.3.3.8.1.1#4

Description:
As Provider (DFP), the Tester intentionally does not send the Source_Capabilities_Extended message in reply to a Get_Source_Cap_EXTENDED message from the UUT, in order to force a SenderResponseTimer timeout on the UUT and verifies it is correctly implemented.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. In Source_Capabilities message, the Tester sets Peak Current of fixed supply PDOs to 0.

b) The Tester sets termination to SinkTxOk.

c) If a Get_Source_Cap_EXTENDED message is not received within 500 ms after termination set to SinkTxOk, the test passes and stops here.

d) Upon receipt of the Get_Source_Cap_EXTENDED message from the UUT, the Tester replies with a GoodCRC message.

e) The Tester intentionally does not send the Source_Capabilities_Extended message in reply to the Get_Source_Cap_EXTENDED message from the UUT, in order to force a SenderResponseTimer timeout on the UUT.

f) If a Hard Reset or Soft Reset is detected within tSenderResponse max from the time the last bit of the GoodCRC message EOP has been sent, the test fails.

TD. PD. SNK3. E6. Get_Status Fields Checks

Applicable to:
- UUT: Consumer Only, Consumer / Provider, Dual-role Port

Assertions Tested:
6.2.1.1.1#1, 6.2.1.1.5#7, 6.2.1.1.8#1, 6.11.2.2.3.1#1-2, 6.11.2.2.3.2#1, 8.3.3.7.2.1#1, 8.3.3.7.2.1#2, 8.3.3.7.2.1#3, 8.3.3.9.1.1#1, 8.3.3.9.1.1#2

Description:
As Provider (DFP), after setting up a contract with the UUT, the Tester sends an Alert message to the UUT, waits for a Get_Status message from the Consumer (UUT) and verifies correct field values.
Power Delivery 3.0 Tests

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.
b) The Tester sends an Alert message to the UUT, with bit 3 of Type of Alert set (OTP event).
c) The Tester sets termination to SinkTxOk.
d) If a Get_Status message is not received within 500 ms after termination set to SinkTxOk, the test passes and stops here.
e) Upon receipt of the Get_Status message, the Tester verifies:
   1. Number of Data Objects field = 000b
   2. Port Power Role field = 0b (Sink)
   3. Specification Revision field = 10b (Rev 3.0)
   4. Port Data Role field = 0b (UFP)
   5. Extended = 0b

TD. PD. SNK3. E7. Get_Battery_Status Fields Checks

Applicable to:

- UUT: Consumer Only, Consumer / Provider, Dual-role Port

Assertions Tested:

6.2.1.1.1#1, 6.2.1.1.1#2, 6.2.1.1.2#3, 6.2.1.1.2#4, 6.2.1.1.5#7, 6.2.1.1.8#1, 6.2.1.2#1, 6.2.1.2#3, 6.2.1.2.1#1, 6.2.1.2.1#2, 6.2.1.2.1#4, 6.2.1.2.1#6, 6.2.1.2.1#7, 6.2.1.2.2#1, 6.2.1.2.2#2, 6.2.1.2.3#3, 6.2.1.2.4#1, 6.2.1.2.4#3, 6.5#1, 6.5.4#2, 6.5.4#3, 6.11.2.2.3.1#1, 6.11.2.2.3.1#2, 6.11.2.2.3.2#1, 8.3.3.7.2.1#1, 8.3.3.7.2.1#2, 8.3.3.7.2.1#3, 8.3.3.11.1.1#1, 8.3.3.11.1.1#2

Description:

As Provider (DFP), after setting up a contract with the UUT, the Tester sends an Alert message to the UUT, waits for a Get_Battery_Status message from the UUT and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported to 1 in Source_Capabilities message during this process.
b) The Tester sends an Alert message to the UUT, with bit 1 (Battery Status Change Event) of Type of Alert set to 1, and B20 set to 1 (Battery 0).
c) The Tester sets termination to SinkTxOk.
d) If a Get_Battery_Status message is not received within 500 ms after the Alert message was sent, the test passes and stops here.
e) Upon receipt of the Get_Battery_Status message, the Tester verifies:
   1. Port Power Role field = 0b (Sink)
   2. Specification Revision field = 10b (Rev 3.0)
3. **Port Data Role field = 0b (UFP)**

4. **Extended** = 1b

5. **Chunk Number** in Extended Message Header = 0

6. **Request Chunk** in Extended Message Header = 0

7. Bit 9 (**Reserved**) in Extended Message Header = 0

8. **Data Size** in Extended Message Header = 1

9. If both the UUT and the Tester support unchunked extended messages
   - **Chunked** bit in Extended Message Header = 0
   - **Number of Data Objects** field = 000b

10. If either the UUT or the Tester doesn’t support unchunked extended messages
    - **Chunked** bit in Extended Message Header = 1
    - **Number of Data Objects** field = 001b
    - The total number of data bytes is consistent with the **Number of Data Objects** field
    - The last byte of the Data Object is 0

11. **Battery Status Ref** = 0
    f) Rerun the test, with Unchunked Extended Messages Supported set to 0 in step a)

**TD. PD. SNK3. E8. Status sent timely**

Applicable to:
- UUT: Consumer Only, Consumer / Provider, Dual-role Port

Assertions Tested:

6.3.16#1, 6.5.2#1, 8.3.3.9.4.1#1, 8.3.3.9.4.1#2

Description:

As Provider (DFP), the Tester verifies that the UUT replies **Get_Status** message with a **Status** message timely.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester sends a **Get_Status** message to the UUT.

c) If a **Status** or **Not_Supported** message is not received within **tReceiverResponse** max, the test fails. This delay is measured from the time the last bit of **Get_Status** message **EOP** has been transmitted to the time the first bit of the **Status** message preamble has been received.
**TD. PD. SNK3. E9. Manufacturer_Info Sent Timely**

**Applicable to:**
- UUT: Consumer Only, Consumer / Provider, Dual-role Port

**Assertions Tested:**
- 6.5.6#1, 6.5.7#1, 8.3.3.12.2.1#1, 8.3.3.12.2.1#2

**Description:**
As Provider (DFP), the Tester verifies that the UUT replies `Get_Manufacturer_Info` message with a `Manufacturer_Info` message timely.

**Steps:**

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester sends a `Get_Manufacturer_Info` message to the UUT.

c) If a `Manufacturer_Info` or Not_Supported message is not received within `tReceiverResponse` max, the test fails. This delay is measured from the time the last bit of `Get_Manufacturer_Info` message EOP has been transmitted to the time the first bit of the `Manufacturer_Info` message preamble has been received.

---

**TD. PD. SNK3. E10. Source_Capabilities_Extended sent timely**

**Applicable to:**
- UUT: Consumer / Provider, Dual-role Port

**Assertions Tested:**
- 6.12.1#6, 8.3.3.16.12.1#1, 8.3.3.15.12.1#2

**Description:**
As Provider (DFP), the Tester verifies that the UUT replies `Get_Source_Cap_Extended` message with a `Source_Capabilities_Extended` message timely.

**Steps:**

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester sends a `Get_Source_Cap_Extended` message to the UUT.

c) If a `Source_Capabilities_Extended` or Not_Supported message is not received within `tReceiverResponse` max, the test fails. This delay is measured from the time the last bit of `Get_Source_Cap_Extended` message EOP has been transmitted to the time the first bit of the `Source_Capabilities_Extended` message preamble has been received.
**TD. PD. SNK3. E11. Receiving chunked extended message**

**Applicable to:**

**UUT:** Consumer Only, Consumer / Provider, Dual-role Port

**Assertions Tested:**

6.5.7#1, 6.6.17.1#1, 6.6.17.1#2, 6.11.2.1.1#1, 6.11.2.1.2.1#3, 6.11.2.1.2.2#1, 6.11.2.1.2.3#2, 6.11.2.1.2.3#3, 6.11.2.1.2.4#1, 6.11.2.1.2.4#2, 6.11.2.1.2.5#2, 8.3.3.5.2.3#1, 8.3.3.5.2.3#2, 8.3.3.5.2.3#3, 8.3.3.12.2.1#1, 8.3.3.12.2.1#2

**Description:**

As Provider (DFP), the Tester verifies that the UUT receives a chunked extended message correctly by sending messages to request chunks.

**Steps:**

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported set to 0 in `Source_Capabilities` message during this process.

b) The Tester sends the first chunk of a chunked extended message to the UUT, with `Data Size` set to 260 and `Message Type` set to 11111b. Bytes 0 to 259 of data block contain incrementing values (mod 256) starting at 0x00.

c) If a message is not received within `tChunkingNotSupported` max, this test fails. The delay is measured from the time the last bit of the EOP of the chunk has been transmitted until the first bit of the response Message Preamble has been received.

d) If the received message is `Not_Supported`, the Tester verifies the message is received after `tChunkingNotSupported` min and stops here.

e) If the message is not received within `tChunkReceiverRequest` max, the test fails.

f) Upon receipt of the response from the UUT, the Tester verifies the following:

1. For Message Header
   - `Extended` = 1
   - `Number of Data Objects` = 1
   - `Port Power Role field = 0b (Sink)`
   - `Port Data Role field = 0b (UFP)`
   - `Specification Revision` = 10b (Rev 3.0)
   - `Message Type` = 11111b

2. For Extended Message Header
   - `Chunked` = 1
   - `Chunk Number` = next chunk in the series
   - `Request Chunk` = 1
   - `Bit 9` = 0 (Reserved)
Power Delivery 3.0 Tests

- **Data Size** = 0
  
g) The Tester sends the requested chunk to the UUT.

h) Repeat e) and g) until the Tester has finished sending all 10 chunks.

**TD. PD. SNK3. E12. Soft_Reset sent regardless of Rp value**

Applicable to:

- UUT: Consumer Only, Consumer / Provider, Dual-role Port

 Assertions Tested:

6.8.1#7

Description:

As Provider (DFP), the Tester forces the UUT to send Soft_Reset and verifies Soft_Reset is sent regardless even though the Rp value is SinkTxNG.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester keeps the Rp value as SinkTXNG and sends a Get_Sink_Cap message to the UUT.

c) Upon receipt of the Sink_Capabilities Message, the Tester doesn’t reply with GoodCRC.

d) The Tester verifies that a Soft_Reset message is sent by the UUT within tReceive max + tSoftReset max, the delay is measured from the time the last bit of the last retransmitted Sink_Capabilities Message EOP has been received to the time the last bit of the Soft_Reset message EOP has been received.

**TD. PD. SNK3. E13. SinkPPSPeriodicTimer Timeout**

Applicable to:

- UUT: Consumer Only, Consumer / Provider, Dual-role Port

 Assertions Tested:

6.4.1.2#9, 6.6.18.1#1, 6.6.18.1#2, 6.6.18.1#3, 8.3.3.3.7#4

Description:

As Provider (DFP), the Tester verifies the UUT periodically re-request the PPS APDO at least every tPPSRequest.

Steps:

a) Run PROC.PD.E2 Bring-up procedure for PPS Tests according to the UUT role.
b) The Tester sets termination to \textit{SinkTxOk}.

c) If a \textit{Request} message is not received within \textit{tPPSRequest max}, the test fails. The delay is measured from the time the last bit of the \textit{EOP} of the \textit{GoodCRC} Message sent by the Tester in response to the previous \textit{Request} Message to the time the first bit of the \textit{Preamble} of the new \textit{Request} Message is received.

**TD. PD. SNK3. E14. Request Fields Checks - PPS**

Applicable to:
- UUT: Consumer Only, Consumer / Provider, Dual-role Port
- UUTs which requests a PPS APDO

Assertions Tested:

\[ \text{6.4.2.7\#6, 6.4.2.13\#1, 6.4.2.13\#2, 6.4.2.13\#3} \]

Description:

As Provider, the Tester waits for a \textit{Request} message from the UUT, which requests a PPS APDO, and verifies correct field values.

Steps:

a) Run \texttt{PROC.PD.E2 Bring-up procedure for PPS Tests} according to the UUT role, but stops after a Request for APDO is received or times out.

b) Upon receipt of the \textit{Request} message, the Tester verifies:

1. \textit{Number of Data Objects} field = 001b
2. \textit{MessageID} field = 000b
3. \textit{Port Power Role} field = 0b (Sink)
4. \textit{Port Data Role} field = 0b (UFP)
5. \textit{Specification Revision} field = 10b (Rev 3.0)
6. \textit{Extended} Field = 0b

c) For the Sink Request Data Object, the Tester verifies:

1. \textit{Object Position} field is 2.
2. The \textit{Operating Current} is less than or equal to the maximum current offered in the \textit{Source Capabilities} Message.
3. The \textit{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.
4. Bit 31 = 0b (\textit{Reserved})
5. Bit 27 = 0b (\textit{Reserved})
6. Bits 22..20 = 000b (\textit{Reserved})
7. Bits 8..7 = 00b (\textit{Reserved})
TD. PD. SNK3. E15. Status Fields Checks

Applicable to:
- UUT: Consumer Only, Consumer / Provider, Dual-role Port

Assertions Tested:
6.2.1.1.1#1, 6.2.1.1.1#2, 6.2.1.1.2#1, 6.2.1.1.2#2, 6.2.1.1.2#3, 6.2.1.1.2#4, 6.2.1.1.5#7, 6.2.1.1.8#1, 6.2.1.2#1, 6.2.1.2#2, 6.2.1.2.1#1, 6.2.1.2.1#2, 6.2.1.2.1#4, 6.2.1.2.2#1, 6.2.1.2.2#1, 6.2.1.2.2#2, 6.2.1.2.2.1#3, 6.2.1.2.3#1, 6.2.1.2.3#2, 6.2.1.2.3#3, 6.2.1.2.4#1, 6.2.1.2.4#3, 6.5#1, 6.5.2#1, 6.5.2#2, 6.5.2.4#2, 6.5.2.5#1

Description:
As Provider (DFP), the Tester sends a *Get_Status* message to the UUT, waits for a *Status* message from the UUT and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported to 1 in *Source_Capabilities* message during this process.

b) The Tester sends a *Get_Status* message to the UUT.

c) If a *Not_Supported* message is received, the test passes and stops here.

d) Upon receipt of the *Status* message, the Tester verifies:

1. **Port Power Role field = 0b (Sink)**
2. **Specification Revision** field = 10b (Rev 3.0)
3. **Port Data Role field = 0b (UFP)**
4. **Extended** = 1b
5. **Chunk Number** in Extended Message Header = 0
6. **Request Chunk** in Extended Message Header = 0
7. Bit 9 (**Reserved**) in Extended Message Header = 0
8. **Data Size** in Extended Message Header = 5
9. If both the UUT and the Tester support unchunked extended messages
   - **Chunked** bit in Extended Message Header = 0
   - **Number of Data Objects** field = 000b
10. If either the UUT or the Tester doesn’t support unchunked extended messages
    - **Chunked** bit in Extended Message Header = 1
    - **Number of Data Objects** field = 2
    - The total number of data bytes is consistent with the **Number of Data Objects** field
    - The last 1 byte of the 2nd Data Object are 0
11. For the SDB
Power Delivery 3.0 Tests

- Bit 0 of Present Input field = 0
- Bits 5..7 of Present Input field are 0
- If bit 3 of Present Input field is 0, Present Battery Input field shall be 0
- Bit 0 and Bits 5..7 of Event Flags field are 0
- OVT Event is not set in Event Flags field
- Bit 0, Bits 3..7 of Temperature Status field are 0
- If OTP event is set in Event Flags, Temperature Status shall be Over temperature. And vice versa.

e) Rerun the test, with Unchunked Extended Messages Supported set to 0 in step a)

TD. PD. SNK3. E16. Manufacturer_Info Fields Checks

Applicable to:
- UUT: Consumer Only, Consumer / Provider, Dual-role Port

Assertions Tested:

6.2.1.1.1#1, 6.2.1.1.1#2, 6.2.1.1.1#4, 6.2.1.1.2#3, 6.2.1.1.2#4, 6.2.1.1.5#7, 6.2.1.1.8#1, 6.2.1.2#1, 6.2.1.2#3, 6.2.1.2.1#1, 6.2.1.2.1#2, 6.2.1.2.1#4, 6.2.1.2.1#6, 6.2.1.2.1#7, 6.2.1.2.2#1, 6.2.1.2.2#2, 6.2.1.2.3#1, 6.2.1.2.3#3, 6.2.1.2.4#1, 6.2.1.2.4#3, 6.5#1

Description:

As Provider (DFP), the Tester sends a Get_Manufacturer_Info message to the UUT, wait for a Manufacturer_Info message from the UUT and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported to 1 in in Source_Capabilities message during this process.
b) The Tester sends a Get_Manufacturer_Info message, in which Manufacturer_Info Target is set to 0, to the UUT.
c) If a Not_Supported message is received, the test passes and stops here.
d) Upon receipt of the Manufacturer_Info message, the Tester verifies:
   1. Port Power Role field = 0b (Sink)
   2. Specification Revision field = 10b (Rev 3.0)
   3. Port Data Role field = 0b (UFP)
   4. Extended = 1b
   5. Chunk Number in Extended Message Header = 0
   6. Request Chunk in Extended Message Header = 0
   7. Bit 9 (Reserved) in Extended Message Header = 0
   8. Data Size in Extended Message Header is 4..26
9. If both the UUT and the Tester support unchunked extended messages
   - **Chunked** bit in Extended Message Header = 0
   - **Number of Data Objects** field = 000b

10. If either the UUT or the Tester doesn’t support unchunked extended messages
    - **Chunked** bit in Extended Message Header = 1
    - **Number of Data Objects** field = (Data Size + 5) / 4
    - The Packet payload is padded to the next 4-byte Data Object boundary with zeros (0x00).

    e) Rerun the test, with Unchunked Extended Messages Supported set to 0 in step a)

**TD. PD. SNK3. E17. Manufacturer_Info Fields Checks - Invalid Manufacturer Info Target**

**Applicable to:**
- **UUT:** Consumer Only, Consumer / Provider, Dual-role Port

**Assertions Tested:**

6.2.1.1.1#1, 6.2.1.1.1#2, 6.2.1.1.1#4, 6.2.1.1.2#3, 6.2.1.1.2#4, 6.2.1.1.5#7, 6.2.1.1.8#1, 6.2.1.2#1, 6.2.1.2#3, 6.2.1.2.1#1, 6.2.1.2.1#2, 6.2.1.2.1#4, 6.2.1.2.1#6, 6.2.1.2.1#7, 6.2.1.2.2#1, 6.2.1.2.2#2, 6.2.1.2.3#1, 6.2.1.2.3#3, 6.2.1.2.4#1, 6.2.1.2.4#3, 6.5#1, 6.5.7#2, 6.5.7.3#2

**Description:**

As Provider (DFP), the Tester sends a **Get_Manufacturer_Info** message, in which Manufacturer Info Target is set to 2, to the UUT, wait for a **Manufacturer_Info** message from the UUT and verifies correct field values.

**Steps:**

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported to 1 in in **Source_Capabilities** message during this process.

b) The Tester sends a **Get_Manufacturer_Info** message, in which Manufacturer Info Target is set to 2, to the UUT.

c) If a **Not_Supported** message is received, the test passes and stops here.

d) Upon receipt of the **Manufacturer_Info** message, the Tester verifies:

   1. **Port Power Role field** = 0b (Sink)
   2. **Specification Revision** field = 10b (Rev 3.0)
   3. **Port Data Role field** = 0b (UFP)
   4. **Extended** = 1b
   5. **Chunk Number** in Extended Message Header = 0
   6. **Request Chunk** in Extended Message Header = 0
   7. Bit 9 (**Reserved**) in Extended Message Header = 0
8. **Data Size** in Extended Message Header is 4
9. If both the UUT and the Tester support unchunked extended messages
   - **Chunked** bit in Extended Message Header = 0
   - **Number of Data Objects** field = 000b
10. If either the UUT or the Tester doesn’t support unchunked extended messages
    - **Chunked** bit in Extended Message Header = 1
    - **Number of Data Objects** field = 2
    - The total number of data bytes is consistent with the **Number of Data Objects** field
    - The last 2 bytes of the 2nd Data object are 0
e) Rerun the test, with Unchunked Extended Messages Supported set to 0 in step a)

**TD. PD. SNK3. E18. Manufacturer_Info Fields Checks - Invalid Manufacturer Info Ref**

Applicable to:
- UUT: Consumer Only, Consumer / Provider, Dual-role Port

Assertions Tested:

6.2.1.1.1#1, 6.2.1.1.1#2, 6.2.1.1.1#4, 6.2.1.1.2#3, 6.2.1.1.2#4, 6.2.1.1.5#7, 6.2.1.1.8#1, 6.2.1.2#1,
6.2.1.2#3, 6.2.1.2.1#1, 6.2.1.2.1#2, 6.2.1.2.1#4, 6.2.1.2.1#6, 6.2.1.2.1#7, 6.2.1.2.2#1, 6.2.1.2.2#2,
6.2.1.2.3#1, 6.2.1.2.3#3, 6.2.1.2.4#1, 6.2.1.2.4#3, 6.5#1, 6.5.7#2, 6.5.7.3#2

Description:

As Provider (DFP), the Tester sends a **Get_Manufacturer_Info** message with an invalid **Manufacturer Info Ref** to the UUT, waits for a **Manufacturer_Info** message from the UUT, and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported to 1 in **Source_Capabilities** message during this process.
b) The Tester sends a **Get_Manufacturer_Info** message, in which **Manufacturer Info Target** is set to 1 and **Manufacturer Info Ref** is set to 8, to the UUT.
c) If a **Not_Supported** message is received, the test passes and stops here.
d) Upon receipt of the **Manufacturer_Info** message, the Tester verifies:
   1. **Port Power Role field = 0b (Sink)**
   2. **Specification Revision** field = 10b (Rev 3.0)
   3. **Port Data Role field = 0b (UFP)**
   4. **Extended** = 1b
   5. **Chunk Number** in Extended Message Header = 0
6. **Request Chunk** in Extended Message Header = 0
7. Bit 9 (**Reserved**) in Extended Message Header = 0
8. **Data Size** in Extended Message Header is 4
9. If both the UUT and the Tester support unchunked extended messages
   - **Chunked** bit in Extended Message Header = 0
   - **Number of Data Objects** field = 000b
10. If either the UUT or the Tester doesn’t support unchunked extended messages
    - **Chunked** bit in Extended Message Header = 1
    - **Number of Data Objects** field = 2
    - The total number of data bytes is consistent with the **Number of Data Objects** field
    - The last 2 bytes of the 2\(^{nd}\) Data object are 0
e) Rerun the test, with Unchunked Extended Messages Supported set to 0 in step a)

**TD. PD. SNK3. E19. ChunkSenderResponseTimer Timeout**

Applicable to:
- **UUT**: Consumer Only, Consumer / Provider, Dual-role Port

Assertions Tested:

6.6.17.3#1, 6.6.17.3#2, 6.6.17.3#5, 6.11.2.1.2.5#1, 6.11.2.1.2.5#3, 6.11.2.1.2.6#4

Description:

As Provider (DFP), the Tester verifies that the UUT recovers correctly after the Tester stops sending chunked messages in the middle.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role. The Tester sets Unchunked Extended Messages Supported set to 0 in **Source_Capabilities** message during this process.
b) The Tester sends the first chunk of a chunked extended message to the UUT, with **Data Size** set to 260 and **Message Type** set to 11111b. Bytes 0 to 259 of data block contain incrementing values (mod 256) starting at 0x00.
c) If a message is not received within **tChunkingNotSupported** max, this test fails. The delay is messaged from the time the last bit of the EOP of the chunk has been transmitted until the first bit of the response Message Preamble has been received.
d) If the received message is **Not_Supported**, the Tester verifies the message is received after **tChunkingNotSupported** min and stops here.
e) If the message is not received within **tChunkReceiverRequest** max, the test fails.
f) Upon receipt of the message from the UUT to request for the next chunk, the Tester sends the requested chunk to the UUT.
g) Repeat f) until the Tester has finished sending 4 chunks and intentionally does not send the 5th chunk to the UUT.
h) The Tester waits for tChunkSenderResponse max + 5 ms, and sends the first chunk to the UUT.
i) If a message is not received within tChunkReceiverRequest max, the test fails.
j) Upon receipt of the message, the Tester verifies the following:

1. For Message Header
   - Extended = 1
   - Number of Data Objects = 1
   - Port Power Role field = 0b (Sink)
   - Port Data Role field = 0b (UFP)
   - Specification Revision = 10b (Rev 3.0)
   - Message Type = 11111b

2. For Extended Message Header
   - Chunked = 1
   - Chunk Number = 1
   - Request Chunk = 1
   - Bit 9 = 0 (Reserved)
   - Data Size = 0

3. The total number of data bytes is consistent with the Number of Data Objects field

4. The last 2 bytes of the Data Object are 0

**TD. PD. SNK3. E20. Collision Avoidance after PR_Swap**

Applicable to:

- UUT: Consumer Only, Consumer / Provider, Dual-role Port

Assertions Tested:

6.11.2.2.2.1#1, 6.11.2.2.2.1#2

Description:

As Provider (DFP), the Tester initiates a **PR_Swap** and verifies the UUT set the Rp value to **SinkTXOK** if it doesn’t have anything to send in 1s after **PR_Swap**.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.
b) The Tester sends a **PR_Swap** to the UUT, if the UUT rejects the **PR_Swap**, the test passes and stops here.
c) The Tester keep monitoring the Rp value for 10s after `PR_Swap` finished and verifies the UUT set the value to `SinkTXOK` if it doesn’t have anything to send in 1s. During this time period, the Tester replies any message sent from the UUT with a proper response.
Cable Tests

TD.PD.CBL3.E1. Receiving Chunked Extended Message

Applicable to:
- UUT: Cable

Assertions Tested:

6.5.7#1, 8.3.3.12.2.1#1, 8.3.3.12.2.1#2

Description:

As Provider (DFP), the Tester verifies that the UUT receives a chunked extended message correctly by sending messages to request chunks.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester sends the first chunk of a chunked extended message to the UUT, with Data Size set to 260 and Message Type set to 111111b. Bytes 0 to 259 of data block contain incrementing values (mod 256) starting at 0x00.

c) If a message is not received within \textit{t\textsubscript{ChunkReceiverRequest}} \textup{max}, this test passes and stops here (in case of the first chunk) or failed (for other chunks). The delay is measured from the time the last bit of the EOP of the chunk has been transmitted until the first bit of the response Message Preamble has been received.

d) Upon receipt of the message from the UUT, the Tester verifies the following:

   1. For Message Header
      - \textit{Extended} = 1
      - \textit{Number of Data Objects} = 1
      - \textit{Cable Plug} = 1b (From Cable Plug)
      - \textit{Bit 5} = 0 (Reserved)
      - \textit{Specification Revision} = 10b (Rev 3.0)
      - \textit{Message Type} = 11111b

   2. For Extended Message Header
      - \textit{Chunked} = 1
      - \textit{Chunk Number} = the next chunk in the series
      - \textit{Request Chunk} = 1
      - \textit{Bit 9} = 0 (Reserved)
      - \textit{Data Size} = 0

e) The Tester sends the requested chunk to the UUT.

f) Repeat c) to e) until the Tester has finished sending all 10 chunks.
TD.PD.CBL3.E2. ChunkSenderResponseTimer Timeout

Applicable to:
- UUT: Cable

Assertions Tested:
6.6.17.3#1, 6.6.17.3#2, 6.6.17.3#5, 6.11.2.1.2.5#1, 6.11.2.1.2.5#3, 6.11.2.1.2.6#4

Description:
As Provider (DFP), the Tester verifies that the UUT recovers correctly after the Tester stops sending chunked messages in the middle.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.
b) The Tester sends the first chunk of a chunked extended message to the UUT, with Data Size set to 260 and Message Type set to 111111b. Bytes 0 to 259 of data block contain incrementing values (mod 256) starting at 0x00.
c) If a message is not received within $t_{ChunkReceiverRequest}$ max, this test passes and stops here. The delay is messaged from the time the last bit of the EOP of the chunk has been transmitted until the first bit of the response Message Preamble has been received.
d) Upon receipt of the message from the UUT to request for the next chunk, the Tester sends the requested chunk to the UUT.
e) Repeat d) until the Tester has finished sending 4 chunks and intentionally does not send the 5th chunk to the UUT.
f) The Tester waits for $t_{ChunkSenderResponse}$ max + 5 ms, and sends the first chunk to the UUT.
g) If a message is not received within $t_{ChunkReceiverRequest}$ max, the test fails.
h) Upon receipt of the message, the Tester verifies the following:
   1. For Message Header
      - $Extended = 1$
      - $Number of Data Objects = 1$
      - $Cable Plug = 1b$ (From Cable Plug)
      - $Bit 5 = 0$ (Reserved)
      - $Specification Revision = 10b$ (Rev 3.0)
      - $Message Type = 111111b$
   2. For Extended Message Header
      - $Chunked = 1$
      - $Chunk Number = 1$
      - $Request Chunk = 1$
      - $Bit 9 = 0$ (Reserved)
Power Delivery 3.0 Tests

- **Data Size** = 0
  3. The total number of data bytes is consistent with the **Number of Data Objects** field
  4. The last 2 bytes of the Data Object are 0

**TD.PD.CBL3.E3. Manufacturer_Info Fields Checks**

Applicable to:

- UUT: Cable

Assertions Tested:

6.2.1.1.1#1, 6.2.1.1.2#3, 6.2.1.1.2#4, 6.2.1.1.5#8, 6.2.1.2#1, 6.2.1.2#3, 6.2.1.2.3#1, 6.2.1.2.3#3, 6.2.1.2.4#1, 6.2.1.2.4#3, 6.5#1

Description:

As Provider (DFP), the Tester sends a **Get_Manufacturer_Info** message to the UUT, wait for a **Manufacturer_Info** message from the UUT and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester sends a **Get_Manufacturer_Info** message, in which **Manufacturer Info Target** is set to 0, to the UUT.

c) If a message is not received within \( t_{ReceiverResponse} \) max, the test passes and stops here. The delay is measured from the time the last bit of the EOP of the **Get_Manufacturer_Info** message has been transmitted until the first bit of the response Message Preamble has been received.

d) If the received message is not **Manufacturer_Info** message, the Tester reports an error.

e) Upon receipt of the **Manufacturer_Info** message, the Tester verifies:
   1. **Cable Plug** field = 1b (From Cable Plug)
   2. **Specification Revision** field = 10b (Rev 3.0)
   3. **Bit 5** = 0 (Reserved)
   4. **Extended** = 1b
   5. **Chunk Number** in Extended Message Header = 0
   6. **Request Chunk** in Extended Message Header = 0
   7. Bit 9 (Reserved) in Extended Message Header = 0
   8. **Data Size** in Extended Message Header is 4..26
   9. **Chunked** bit in Extended Message Header = 1
   10. **Number of Data Objects** field = \( \frac{\text{Data Size} + 5}{4} \)
   11. The Packet payload is padded to the next 4-byte Data Object boundary with zeros (0x00).
TD.PD.CBL3.E4. Manufacturer_Info Fields Checks - Invalid Manufacturer Info Target

Applicable to:
- UUT: Cable

Assertions Tested:

6.2.1.1.1#1, 6.2.1.1.2#3, 6.2.1.1.5#8, 6.2.1.1.8#1, 6.2.1.2#1, 6.2.1.2#3, 6.2.1.2.1#1,
6.2.1.2.1#2, 6.2.1.2.1#6, 6.2.1.2.1#7, 6.2.1.2.2#1, 6.2.1.2.2#2, 6.2.1.2.3#1, 6.2.1.2.3#3, 6.2.1.2.4#1,
6.2.1.2.4#3, 6.5#1, 6.5.7#2, 6.5.7.3#2

Description:

As Provider (DFP), the Tester sends a \textit{Get\_Manufacturer\_Info} message, in which Manufacturer Info Target is set to 2, to the UUT, wait for a \textit{Manufacturer\_Info} message from the UUT and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester sends a \textit{Get\_Manufacturer\_Info} message, in which Manufacturer Info Target is set to 2, to the UUT.

c) If a message is not received within \textit{tReceiverResponse} max, the test passes and stops here. The delay is measured from the time the last bit of the EOP of the \textit{Get\_Manufacturer\_Info} message has been transmitted until the first bit of the response Message Preamble has been received.

d) If the received message is not \textit{Manufacturer\_Info} message, the Tester reports an error.

e) Upon receipt of the \textit{Manufacturer\_Info} message, the Tester verifies:

1. \textit{Cable Plug} field = 1b (From Cable Plug)
2. \textit{Specification Revision} field = 10b (Rev 3.0)
3. \textit{Bit 5} = 0 (Reserved)
4. \textit{Extended} = 1b
5. \textit{Chunk Number} in Extended Message Header = 0
6. \textit{Request Chunk} in Extended Message Header = 0
7. Bit 9 (Reserved) in Extended Message Header = 0
8. \textit{Data Size} in Extended Message Header is 4
9. \textit{Chunked} bit in Extended Message Header = 1
10. \textit{Number of Data Objects} field = 2
11. The total number of data bytes is consistent with the \textit{Number of Data Objects} field
12. The last 2 bytes of the 2\textsuperscript{nd} Data object are 0
TD.PD.CBL3.E5. Unrecognized Message Received

Applicable to:
- UUT: Cable

Assertions Tested:

6.5.7#1, 8.3.3.12.2.1#1, 8.3.3.12.2.1#2

Description:

As Provider (DFP), the Tester verifies that the UUT ignores an unrecognized message.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.
b) The Tester sends the first chunk of a chunked extended message to the UUT, with Data Size set to 26 and Message Type set to 11111b. Bytes 0 to 25 of data block contain incrementing values starting at 0x00.
c) If a message is received within tChunkReceiverRequest max, the test fails. The delay is measured from the time the last bit of the EOP of the chunk message has been transmitted until the first bit of the response Message Preamble has been received.


Applicable to:
- UUT: Cable

Assertions Tested:

6.2.1.1.5#13, 6.2.1.1.5#14

Description:

As Provider (DFP), the Tester verifies that the UUT doesn’t save the agreed Specification Revision and always respond with the highest Specification Revision it supports that is equal to or lower than the Specification Revision contained in the Message received.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.
b) The Tester sends a Discover Identity message, with specification revision set to 10b (Rev 3), to the UUT.
c) If the specification revision is 01b (Rev 2) in the Discover Identity ACK message, this test passes and stops here.
d) The Tester sends a GoodCRC message to the UUT.
e) The Tester sends a Discover SVIDs message, with specification revision set to 01b (Rev 2), to the UUT.

f) The Tester verifies the specification revision in the responded message is also set to 01b.

TD.PD.CBL3.E7. Status sent timely

Applicable to:
- UUT: Cable

Assertions Tested:

Description:
As Provider (DFP), the Tester verifies that the UUT replies Get_Status message with a Status message timely.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) If the Product Type (Cable Plug) in the returned Discovery Identity ACK message is not Active Cable, the test passes and stops here.

c) The Tester sends a Get_Status message to the UUT.

d) If a Status message is not received within tReceiverResponse max, the test fails. This delay is measured from the time the last bit of Get_Status message EOP has been transmitted to the time the first bit of the Status message preamble has been received.

TD.PD.CBL3.E8. Status Fields Checks

Applicable to:
- UUT: Consumer Only, Consumer / Provider, Dual-role Port

Assertions Tested:

Description:
As Provider (DFP), the Tester sends a Get_Status message to the UUT, waits for a Status message from the UUT and verifies correct field values.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) If the Product Type (Cable Plug) in the returned Discovery Identity ACK message is not Active Cable, the test passes and stops here.

c) The Tester sends a Get_Status message to the UUT.

d) Upon receipt of the Status message, the Tester verifies:
1. **Cable Plug** field = 1b (From Cable Plug)
2. **Specification Revision** field = 10b (Rev 3.0)
3. **Bit 5** = 0 (Reserved)
4. **Extended** = 1b
5. **Chunk Number** in Extended Message Header = 0
6. **Request Chunk** in Extended Message Header = 0
7. Bit 9 (Reserved) in Extended Message Header = 0
8. **Data Size** in Extended Message Header is 2
9. **Chunked** bit in Extended Message Header = 1
10. **Number of Data Objects** field = 1
11. For the SDB
    - Internal Temp shall be less than the Shutdown Temperature reported in Active Cable VDO2
    - Bit 0 of the Flags field is 0 (Thermal Shutdown cleared)
    - Bits 1..7 of the Flags field are 0 (Reserved)
VDM Tests

TD. PD. VDM3. E1. Fields Checks - Discover Identity

Applicable to:
- UUT: Consumer Only, Consumer / Provider, Dual-role Port, Provider / Consumer, Provider Only, Cable

Assertions Tested:
6.2.1.1.5#8, 6.4.4.2#10, 6.4.4.2.3#4, 6.4.4.3.1.1.6#2, 6.4.4.3.1.4.1#3, 6.4.4.3.1.4.1#4, 6.4.4.3.1.4.1#5, 6.4.4.3.1.4.2#3, 6.4.4.3.1.4.2#4, 6.4.4.3.1.4.2#5, 6.12.4#2

Description:
This test verifies that the UUT responds a well-formed message to a Discover Identity.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester sends Discover Identity.

c) The test passes and stops here if the UUT is not a cable and a Not_Supported message is received.

d) Wait for the Discover Identity Response and verify the following:

1. For VDM Header (VDO #1)
   - SVID = 0xFF00 (PD SID)
   - VDM Type = 1b (Structured VDM)
   - Structured VDM Version = 01b (Version 2.0)
   - Bits 12-11 are 0 (reserved).
   - Object Position = 000b
   - Command Type = 01b (ACK) if the UUT is a Cable and Command Type != 00b (REQ) otherwise
   - Bit 5 is 0 (reserved).
   - Command = 0001b (Discover Identity).

2. If the Command Type in VDM Header is not 01b (ACK), the test verifies Number of Data Objects is 1 and stops here.

3. For ID Header (VDO #2)
   - Product Type(UFP) or Product Type(Cable Plug) and Product Type (DFP) are not Reserved
   - Bits 22..16 are 0 (reserved)

4. If Product Type is Passive Cable, for Passive Cable VDO (VDO #5)
Power Delivery 3.0 Tests

- VDO Version field is 000b (Version 1.0)
- USB Type-C plug to USB TypeC/Captive field is 10b (USB Type-C) or 11b (Captive)
- Bit 20 is 0 (Reserved)
- Bit 17 is 0 (Reserved)
- Bits 8..7 are 0 (Reserved)
- Bits 4..3 are 0 (Reserved)

5. If Product Type is Active Cable
   - For Active Cable VDO 1 (VDO #5)
     - VDO Version field is 001b (Version 1.1)
     - USB Type-C plug to USB TypeC/Captive field is 10b (USB Type-C) or 11b (Captive)
     - Bit 20 is 0 (Reserved)
     - Bit 17 is 0 (Reserved)
     - Bits 2..0 are 0 (Reserved)
   - For Active Cable VDO 2 (VDO #6)
     - Bits 15..6 are 0 (Reserved)
     - Bit 2 is 0 (Reserved)
     - SuperSpeed Signaling field shall be 00b (Gen1) or 01b (Gen2)

6. If Product Type is Alternate Mode Adapter, for AMA VDO (VDO #5)
   - VDO Version field is 000b (Version 1.0)
   - Bits 20..8 are 0 (Reserved)

   e) The test fails if any of these checks are invalid.
   f) The test fails if the Product Type declared in the ID Header (VDO #2) is Hub (1), Peripheral (2) or Power Brick(3) and the Number of Data Objects is not exactly 4.
   g) The test fails if the Product Type declared in the ID Header (VDO #2) is Passive Cable (3), or AMA (5) and the Number of Data Objects is not exactly 5.
   h) The test fails if the Product Type declared in the ID Header (VDO #2) is Active Cable (4) and the Number of Data Objects is not exactly 6.

TD. PD. VDM3. E2. Unrecognized VID in Unstructured VDM

Applicable to:

- UUT: Consumer Only, Consumer / Provider, Dual-role Port

Assertions Tested:

6.4.4.1#9
Power Delivery 3.0 Tests

Description:

As a DFP, the Tester verifies that the UUT responds a Not_Supported message to an unstructured VDM message with an unrecognized VID.

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester sends an unstructured VMD message to the UUT, in which VID in the Unstructured VDM Header is set to the Tester manufacture’s VID.

c) If a Not_Supported message is not received within tReceiverResponse max, the test fails. This delay is measured from the time the last bit of unstructured VMD message EOP has been transmitted to the time the first bit of the Not_Supported message preamble has been received.
Consistency Tests

TD. PD. VNDI3. E1. Source Capabilities

Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:
10.2.3.2#1, 10.2.3.2#3, 10.2.3.2#4, 10.2.3.2#5, 10.2.3.2#7

Description:
This test verifies the returned Source Capabilities match vendor-supplied information.

Steps:

a) Run PROC.PD.E1 Bring-up For DFP UUT steps a and b.

b) In addition to emulating an UFP, the Tester will also present Ra and will reply to SOP’ Discover Identity as Passive Cable with Maximum VBUS Voltage set to 50V and Vbus Current Handling Capability set to 5A.

c) Upon receipt of the Source Capabilities, the Tester checks that:
   1. The Number of Data Objects equals Num_Src_PDOs
   2. The Specification Revision matches with Spec_Revision

d) If Specification Revision is Rev 2.0, the test stops here.

e) For the first PDO, the Tester checks consistency of:
   1. Unchunked_Extended_Messages_Supported if Specification Revision is 10b (Rev 3.0)

f) For each PDO, the Tester checks consistency of:
   1. Src_PDO_Supply_TypeN
   2. If Fixed:
      - Src_PDO_VoltageN
      - Src_PDO_Peak_CurrentN
      - Src_PDO_Max_CurrentN
   3. If Battery:
      - Src_PDO_MinVoltageN
      - Src_PDO_MaxVoltageN
      - Src_PDO_Max_PowerN
   4. If Variable:
      - Src_PDO_Min_VoltageN
      - Src_PDO_Max_VoltageN
      - Src_PDO_Max_CurrentN
5. If Augmented:
   – \texttt{Src\_PDO\_Min\_VoltageN}
   – \texttt{Src\_PDO\_Max\_VoltageN}
   – \texttt{Src\_PDO\_Max\_CurrentN}
   
g) The Tester checks all PDOs against the PD\_Power\_as\_Source field in the VIF, and it applies the rules and considerations as outlined in the latest version of USB PD Revision 3.0 specification (including but not limited to the changes as outlined in “USB PD R3.0 V1.1 ECN APDO Min Voltage” and Rounding Rules ECN).

\textbf{TD. PD. VNDI3. E2. Request}

Applicable to:
- \texttt{UUT: Consumer Only, Consumer / Provider, Dual-role Port}

Assertions Tested:
6.4.2.6#1

Description:
The Tester verifies that the Specification Revision and Unchunked Extended Messages Supported in Request message match vendor specific information.

Steps:
\begin{itemize}
  \item[a)] Run PROC.PD.E1 Bring-up for UFP UUT steps a) to d).
  \item[b)] The Tester sends a Source\_Capabilities message to the UUT.
  \item[c)] Upon receipt of the Request message from the UUT, the Tester checks consistency of:
          \begin{itemize}
            \item Specification\_Revision
            \item Unchunked\_Extended\_Messages\_Supported if Specification Revision is 10b (Rev 3.0)
          \end{itemize}
\end{itemize}

\textbf{TD. PD. VNDI3. E3. VDM Identity}

Applicable to:
- \texttt{UUT: Consumer Only, Consumer / Provider, Dual-role Port, Provider / Consumer, Provider Only, Cable}

Assertions Tested:
6.2.1.1.5#6, 6.4.4.2#6, 6.4.4.3.1.1.3#1, 6.4.4.3.1.1.6#1, 6.4.4.3.1.1.6#3, 6.4.4.3.1.1.6#4, 6.4.4.3.1.1.6#5, 6.4.4.3.1.1.6#6, 6.4.4.3.1.4.1#14, 6.4.4.3.1.4.2#14

Description:
This test verifies that the VDM Information is as specified in the vendor-supplied information.
Power Delivery 3.0 Tests

Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester executes a Discover Identity exchange.

c) If the UUT is not a cable and if Responds_To_Discover_SOP is set to No, the Tester checks that the UUT replies Not_Supported. The test stops here in this case.

d) For Cables, the Tester checks consistency of Specification_Revision.

e) For all devices, the Tester checks in the ID Header consistency of:
   1. Product_Type(UFP)
   2. Product_Type(Cable Plug)
   3. Product_Type(DFP)
   4. USB_VID(_SOP)
   5. Modal_Operation_Supported(_SOP)
   6. Data_Capable_as_USB_Host(_SOP)
   7. Data_Capable_as_USB_Device(_SOP)

f) For all devices, the Tester checks in the Cert Stat VDO consistency of:
   1. XID(_SOP)

g) For all devices, the Tester checks in the Product VDO consistency of:
   1. PID(_SOP)
   2. bcdDevice(_SOP)

h) For Cables, the Tester checks in the Cable VDO consistency of:
   1. Cable_HW_Vers
   2. Cable_FW_Vers
   3. Type_C_to_Type_C_Capt_Vdm_V2
   4. Cable_Latency
   5. Cable_Termination_Type
   6. Max_VBUS_Voltage_Vdm_V2
   7. Cable_VBUS_Current
   8. VBUS_through_cable
   9. Cable_SOP"_controller
   10. Cable_Superspeed_Support

i) For Alt Mode Adapters, the Tester checks in the AMA VDO consistency of:
   1. AMA_HW_Vers
   2. AMA_FW_Vers
   3. AMA_VCONN_power
   4. AMA_VCONN_reqd
   5. AMA_VBUS_reqd
   6. AMA_Superspeed_Support
**TD. PD. VNDI3. E4. Manufacturer Info**

**Applicable to:**
- UUT: Consumer Only, Consumer / Provider, Dual-role Port, Provider / Consumer, Provider Only, Cable

**Assertions Tested:**
6.5.7.1#1, 6.5.7.2#1

**Description:**
This test verifies that the returned manufacture info is as specified in the vendor-supplied information.

**Steps:**

a) Run PROC.PD.E1 Bring-up according to the UUT role.
b) The Tester waits till it can start an AMS if it is in Sink Role.
c) The Tester sends a `Get_Manufacturer_Info` message, in which `Manufacturer Info Target` is set to 0, to the UUT.
d) If `Manufacturer_Info_Supported` is set to no, the Tester checks that the UUT replies `Not_Supported` (if the UUT is not a cable) or ignores the message (if the UUT is a cable). The test stops here in this case.
e) Upon receipt of the `Manufacturer_Info` message, the Tester checks consistency of:
   1. `Manufacturer_Info_VID(_SOP)`
   2. `Manufacturer_Info_PID(_SOP)`

**TD. PD. VNDI3. E5. Chunking Implemented**

**Applicable to:**
- UUT: Consumer Only, Consumer / Provider, Dual-role Port, Provider / Consumer, Provider Only, Cable

**Assertions Tested:**

**Description:**
This test verifies that the chunking behavior is as specified in the vendor-supplied information.

**Steps:**

a) Run PROC.PD.E1 Bring-up according to the UUT role.
b) The Tester waits till it can start an AMS if it is in Sink Role.
c) The Tester sends the first chunk of a chunked extended message to the UUT, with Data Size set to 260 and Message Type set to 11111b. Bytes 0 to 259 of data block contain incrementing values (mod 256) starting at 0x00.

d) If a message is not received within tChokingNotSupported max, the Tester verifies Chunking_Implemented(SOP) is set to no, and the test stops here. The delay is measured from the time the last bit of the EOP of the chunk has been transmitted until the first bit of the response Message Preamble has been received.

e) If the received message is Not_Supported and the UUT is not Cable, the Tester verifies Chunking_Implemented(SOP) is set to no, and the test stops here.

f) The Tester verifies the message is received within tChunkReceiverRequest max and Chunking_Implemented(SOP) is set to yes. The delay is measured from the time the last bit of the EOP of the chunk has been transmitted until the first bit of the response Message Preamble has been received.

**TD. PD. VNDI3. E6. Unchunked_Extended_Messages_Supported**

Applicable to:
- UUT: Consumer Only, Consumer / Provider, Dual-role Port, Provider / Consumer, Provider Only

Assertions Tested:
6.4.1.2.2.6#1

Description:
This test verifies that if Unchunked_Extended_Messages_Supported is set to yes, the UUT can receive Extended Messages with Data Size > MaxExtendedMsgLegacyLen bytes in a single, Unchunked Message.

Steps:

a) If Unchunked_Extended_Messages_Supported is set to no, this test passes and stops here.

b) Run PROC.PD.E1 Bring-up according to the UUT role.

c) The Tester waits till it can start an AMS if it is in Sink Role.

d) The Tester sends an extended message to the UUT, with Data Size set to 260 and Message Type set to 11111b. Bytes 0 to 259 of data block contain incrementing values (mod 256) starting at 0x00.

e) If a GoodCRC is not received in time, the test fails.

f) If a Not_Supported message is not received within tReceiverResponse max, the test fails. This delay is measured from the time the last bit of the extended message EOP has been transmitted to the time the first bit of the Not_Supported message preamble has been received.
TD. PD. VNDI3. E7. Security_Msgs_Supported

Applicable to:
- UUT: Consumer Only, Consumer / Provider, Dual-role Port, Provider / Consumer, Provider Only, Cable

Assertions Tested:
8.3.3.14.2.1#1, 8.3.3.14.2.1#2

Description:
This test verifies that the support of security messages is as specified in the vendor-supplied information.

Steps:
1. Run PROC.PD.E1 Bring-up according to the UUT role.
2. The Tester waits till it can start an AMS if it is in Sink Role.
3. The Tester sends a Security_Request message, in which SRQDB is set to {0x01, 0x81, 0x00, 0x00}, to the UUT.
4. If Security_Msgs_Supported(_SOP) is set to no, the Tester checks that the UUT replies Not_Supported (if the UUT is not a cable) or ignores the message (if the UUT is a cable). The test stops here in this case.
5. The Tester verifies a Security_Response message is received within tDigestSent (pay attention to the different values for Unchunked and Chunked), and notifies the user that the UUT must be run through Auth compliance tests in CV.

TD. PD. VNDI3. E8. Sink Capabilities

Applicable to:
- UUT: Consumer Only, Provider / Consumer, Consumer / Provider, Dual-role Port

Assertions Tested:
6.4.1.3.1.6#1, 6.4.1.3.4#1, 6.4.1.3.4#2

Description:
This test verifies the returned Sink Capabilities match vendor-supplied information.

Steps:
1. Run PROC.PD.E1 Bring-up according to the UUT role.
2. The Tester waits till it can start an AMS if it is in Sink Role.
3. The Tester sends a Get_Sink_Cap message to the UUT.
d) The Tester waits for the Sink Capabilities and then checks that:
   1. The Number of Data Objects equals Num_Snk_PDOs
   2. The Specification Revision matches with Spec_Revision

e) If Specification Revision is Rev 2.0, the test stops here.

f) For the first PDO, the Tester checks consistency of:
   1. FR_Swap_Reqd_Type_C_Current

g) For each PDO, the Tester checks consistency of:
   1. Snk_PDO_Supply_TypeN
   2. If Fixed:
      - Snk_PDO_VoltageN
      - Snk_PDO_Peak_CurrentN
      - Snk_PDO_Max_CurrentN
   3. If Battery:
      - Snk_PDO_MinVoltageN
      - Snk_PDO_MaxVoltageN
      - Snk_PDO_Max_CurrentN
   4. If Variable:
      - Snk_PDO_Min_VoltageN
      - Snk_PDO_Max_VoltageN
      - Snk_PDO_Max_CurrentN
   5. If Augmented:
      - Snk_PDO_Min_VoltageN
      - Snk_PDO_Max_VoltageN
      - Snk_PDO_Max_CurrentN

h) The Tester checks that no PDO is above PD_Power_as_Sink.


Applicable to:
- UUT: Provider Only, Provider / Consumer, Dual-role Port

Assertions Tested:
6.5.1.14#1

Description:
As a Sink (UFP), the Tester verifies the returned Source_Capabilities_Extended match vendor-supplied information.
Steps:

a) Run PROC.PD.E1 Bring-up according to the UUT role.

b) The Tester waits until it can start an AMS (Run PROC.PD.E3) and sends a `Get_Source_Cap_Extended` message to the UUT.

c) If a `Not_Supported` message is received, the test passes and stops here.

d) Upon receipt of the `Source_Capabilities_Extended` message, the Tester verifies:

   1. `Source PDP` field is the floor of the PD_Power_as_Source
   2. `Number of Batteries/Battery Slots` field matches with Num_Fixed_Batteries and Num_Swappable_Battery_Slots fields in the VIF