Device Wire Adapter (DWA) Test Specification

Designed using the Certified Wireless USB Base Specification, Revision 1.0

Date: September 27, 2006
Revision: 1.0
The information in this document is under review and is subject to change.

### Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Issue Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>July 8, 2005</td>
<td>Initial Version</td>
</tr>
<tr>
<td>0.4</td>
<td>October 17, 2005</td>
<td>Split WA test specification document into DWA and HWA.</td>
</tr>
<tr>
<td>0.6</td>
<td>October 31, 2005</td>
<td>DWA tests complete except ISO &amp; Mixed traffic.</td>
</tr>
<tr>
<td>0.7</td>
<td>November 7, 2005</td>
<td>Added ISO tests and mixed traffic tests.</td>
</tr>
<tr>
<td>0.8</td>
<td>25 November, 2005</td>
<td>Incorporated specification authors review comments into the document.</td>
</tr>
<tr>
<td>0.9</td>
<td>27 December, 2005</td>
<td>Incorporated additional review comments.</td>
</tr>
<tr>
<td>1.0 RC</td>
<td>June 20, 2006</td>
<td>Incorporated .9 promotor group review comments and updates from initial test implementation.</td>
</tr>
<tr>
<td>1.0</td>
<td>September 27, 2006</td>
<td>Incorporated 1.0 RC member review comments.</td>
</tr>
</tbody>
</table>

**THIS TEST SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. The USB-IF disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification. No license, express or implied, to any intellectual property rights is granted herein.**

The USB-IF assumes no responsibility for any errors that may appear in this document. The USB-IF makes no commitment to update the information contained herein, and may make changes at any time without notice.

Copyright © 2005, 2006 USB-IF. All rights reserved.

Third-party brands and names are the property of their respective owners. Other product and corporate names may be trademarks of other companies and are only for explanation and to the owners' benefit, without intent to infringe.

**Significant Contributors:**

- Daniel S Froelich Intel
- Abdul Ismail Intel
- Mark Maszak Microsoft
- Robby Putzeys Professional Multimedia Test Centre
1. Introduction 5
2. Test Assertions .............................................................................................................................. 5
3. Test Descriptions .......................................................................................................................... 31
   DWA States ...................................................................................................................................... 31
   DWA Specific Configuration Routine.............................................................................................. 32
   RPipe Initialization routine............................................................................................................. 32
   Device Enumeration test.................................................................................................................. 35
      TD.1.1 Device Enumeration on DWA .......................................................................................... 35
   Descriptor tests .............................................................................................................................. 36
      TD.1.2 DWA Device descriptor test............................................................................................. 36
      TD.1.3 DWA Configuration descriptor test.................................................................................. 36
      TD.1.4 DWA Interface Association descriptor test ..................................................................... 37
      TD.1.5 DWA Data Transfer Interface descriptor test .................................................................. 38
      TD.1.6 DWA Wire Adapter Class descriptor test ....................................................................... 38
      TD.1.7 DWA Notification Endpoint descriptor test .................................................................... 39
      TD.1.8 DWA Notification Endpoint Companion descriptor test ............................................. 40
      TD.1.9 DWA Isochronous Streaming Interface descriptor test ............................................... 40
      TD.1.10 DWA Isochronous Streaming OUT Endpoint descriptor test ..................................... 41
      TD.1.11 DWA Isochronous Streaming IN Endpoint descriptor test ........................................ 41
      TD.1.12 DWA Wire Adapter RPipe descriptor test .................................................................... 42
   RPipe Status and Features tests ..................................................................................................... 44
      TD.1.13 Abort RPipe Feature test ............................................................................................... 44
      TD.1.14 RPipe Status and RPIPE_PAUSE Feature test .............................................................. 45
      TD.1.15 Set and Get RPipe Descriptor test ................................................................................ 46
      TD.1.16 Reset RPipe test .......................................................................................................... 47
   Wire Adapter Status and Features test .......................................................................................... 48
      TD.1.17 WIRE_ADAPTER_ENABLE Feature test ..................................................................... 48
      TD.1.18 WIRE_ADAPTER_RESET Feature test ........................................................................ 48
      TD.1.19 Get Wire Adapter Status request test ......................................................................... 49
   Standard Loopback tests ............................................................................................................... 51
      TD.1.20 Control Loopback test .................................................................................................. 51
      TD.1.21 Bulk Loopback test ...................................................................................................... 51
      TD.1.22 Interrupt Loopback test ............................................................................................... 52
      TD.1.23 Isochronous Streaming Loopback test ....................................................................... 53
   Functional tests ............................................................................................................................. 55
      TD.1.24 Control Loopback behind High Speed hub test ........................................................... 55
      TD.1.25 Bulk Loopback behind High Speed hub test ............................................................... 55
TD.1.26 Interrupt Loopback behind High Speed hub test ................................................................. 55
TD.1.27 Isochronous Loopback behind High Speed hub test .......................................................... 56
TD.1.28 Target Device Address and Endpoint Address test .......................................................... 56
TD.1.29 DWA Set Isochronous Endpoint Attributes test ............................................................... 56
TD.1.30 Interrupt IN downstream polling interval test ................................................................. 57
TD.1.31 Interrupt OUT downstream polling interval test ............................................................. 58
TD.1.32 Isochronous IN downstream polling interval test ............................................................ 58
TD.1.33 Isochronous OUT downstream polling interval test ....................................................... 59

Transfer Request and Transfer Result tests .................................................................................. 60
TD.1.34 DWA Control Write Transfer Requests and Transfer Result test ........................................ 60
TD.1.35 DWA Control Read Transfer Requests and Transfer Result test ..................................... 61
TD.1.36 DWA Bulk OUT Transfer Requests and Transfer Result test .......................................... 62
TD.1.37 DWA Bulk IN Transfer Requests and Transfer Result test ............................................. 63
TD.1.38 DWA Interrupt OUT Transfer Requests and Transfer Result test ..................................... 64
TD.1.39 DWA Interrupt IN Transfer Requests and Transfer Result test ....................................... 66
TD.1.40 Bulk Maximum Transfer Request Size Loopback test ..................................................... 67
TD.1.41 Short Packet test …………………………………………………………………………………………… 67
TD.1.42 TRANSFER_STATUS_HALTED test ………………………………………………………………………... 68
TD.1.43 TRANSFER_STATUS_BABBLE test ……………………………………………………………………… 68
TD.1.44 TRANSFER_STATUS_TRANSACTION_ERROR with limited bmRetryOptions test …………… 69
TD.1.45 TRANSFER_STATUS_TRANSACTION_ERROR with unlimited bmRetryOptions test ……… 70
TD.1.46 TRANSFER_STATUS_ABORTED test ……………………………………………………………………… 70

Resume and Reset tests …………………………………………………………………………………………… 72
TD.1.47 Reset test ………………………………………………………………………………………………………. 72

Isochronous tests ……………………………………………………………………………………………………… 73
TD.1.48 Isochronous Streaming Downstream IN Token generation test ........................................ 73
TD.1.49 Isochronous IN Data Split test …………………………………………………………………………….. 73
TD.1.50 Isochronous IN buffering test …………………………………………………………………………….. 74
TD.1.51 Downstream SOF Framenumber test ……………………………………………………………………… 75

Mixed traffic tests ……………………………………………………………………………………………………… 76
TD.1.52 Isochronous Simultaneous Streaming test ………………………………………………………………. 76
TD.1.53 Mixed traffic test ……………………………………………………………………………………………. 76

Request errors tests …………………………………………………………………………………………………… 77
TD.1.54 Set Wire Adapter Feature Request error test ……………………………………………………………. 77
TD.1.55 Clear Wire Adapter Feature Request error test ………………………………………………………….. 77
TD.1.56 Clear RPipe Feature Request error test …………………………………………………………………. 77
TD.1.57 Set RPipe Feature Request error test ……………………………………………………………………… 78
1. Introduction

This document is the test specification for USB-IF testing of Device Wire Adapter products against the Wireless USB Wire Adapter Class. The Wire Adapter Class is described in Chapter 8 of the Certified Wireless USB Specification.

Device Wire Adapters (DWA) are specific implementations of Wireless USB devices. And in this respect a DWA must comply to various other test specifications outside the scope of this test specification. A DWA must comply with:

- This test specification
- WUSB Command Verifier (WUSBCV) Test Specification
- Applicable portions of the USBCV Test Specification. (Hub Port and device level tests).

2. Test Assertions

Unless otherwise noted, subsection references are to the Certified Wireless USB Specification, Revision 1.0.

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action: Fail/Warning/NoTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wire Adapter Assertions outside of Chapter 8</td>
<td>Subsection reference: 4.3 Wireless USB Communication Flows</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subsection reference: 4.3.3 USB Time across Device Wire Adapters</td>
<td>4.3.3#2 The FrameNumber value of the SOFs on the downstream wired bus must match bits 13:3 of the 17 bits long 1/8th millisecond value field in the timestamp of any MMCs transmitted during a 1/8th millisecond period.</td>
<td>TD.1.51</td>
<td></td>
</tr>
<tr>
<td>4.3.3#3 The FrameNumber value of the SOFs on the downstream wired bus must match bits 13:3 of the 17 bits long 1/8th millisecond value field in the timestamp of any MMCs transmitted during a 1/8th millisecond period.</td>
<td>TD.1.51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chapter 8 Assertions</td>
<td>Subsection reference: 8.1 Operational Model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subsection reference: 8.1.1 Functional Characteristics</td>
<td>8.1.1#1 The WA Device must support USB 2.0 or Wireless USB standard device requests on the Default Control Pipe. These requests are defined in Chapter 9 of the USB 2.0 specification and Section 7.3 of the Wireless USB Specification.</td>
<td>USBCV &amp; WUSBCV</td>
<td></td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
<td>Action:</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td><strong>Subsection reference: 8.1.2 Data Transfer Interface</strong></td>
<td></td>
<td>Fail/</td>
</tr>
<tr>
<td>8.1.2#1</td>
<td>The Data Transfer Interface of a WA must have a minimum of three function endpoints: a Notification Endpoint (Interrupt IN) and a Data Transfer Endpoint pair (paired Bulk IN/OUT).</td>
<td>TD.1.3</td>
<td>Warning/</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TD.2.4</td>
<td>NoTest</td>
</tr>
<tr>
<td></td>
<td><strong>Subsection reference: 8.1.3 Remote Pipe</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.1.3#1</td>
<td>The minimum number of Remote Pipes is twice the number of devices supported by a HWA.</td>
<td>N/A (HWA)</td>
<td></td>
</tr>
<tr>
<td>8.1.3#2</td>
<td>The minimum number of Remote Pipes is twice the number of ports supported by a DWA.</td>
<td>TD.1.6</td>
<td></td>
</tr>
<tr>
<td>8.1.3#3</td>
<td>The length and transfer request type of the Transfer Request packet received by a wire adapter must match.</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.1.3#4</td>
<td>The target Remote Pipe of a wire adapter must be configured to the same transfer type as the Transfer Request.</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.1.3#5</td>
<td>A WA must not STALL the endpoint when a Transfer Request is incorrect.</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.1.3#6</td>
<td>After receiving an incorrect Transfer Request, a WA must continue to accept an incorrect Transfer Request and any data that may follow the request. The wire adapter must then send a Transfer Completion notification on the notification endpoint”</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.1.3#7</td>
<td>After receiving an incorrect Transfer Request, and when the Data Transfer IN endpoint of a WA is polled by the host to get the Transfer Result, the WA must state that the Wire Adapter detected an error in the Transfer Request.</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.1.3#8</td>
<td>When the transfer completes, a WA must send a Transfer Complete Notification to the host on the notification endpoint.</td>
<td>TD.1.34</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TD.1.35</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TD.1.36</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TD.1.37</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TD.1.38</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TD.1.39</td>
<td></td>
</tr>
<tr>
<td>8.1.3#9</td>
<td>If the host sends more Transfer Requests than a Remote Pipe in a WA can concurrently handle (as reported in its RPipe descriptor) the WA must NAK the transaction until it has completed a pending transfer on that Remote Pipe.</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Subsection reference: 8.1.5 Downstream Port(s)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.1.5#1</td>
<td>A DWA monitors the status of all of the ports and reports them to the host if there is any change.</td>
<td>USBCV</td>
<td></td>
</tr>
<tr>
<td>8.1.5#2</td>
<td>The maximum number of downstream ports that can be implemented on a WA is 127.</td>
<td>TD.1.6</td>
<td></td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
<td>Action:</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>8.1.5#3</td>
<td>The functions and behavior of the downstream port on a DWA are the same as the ones of a USB 2.0 Hub which are described in the Section 11.5 of the USB 2.0 Specification.</td>
<td>USBCV</td>
<td>Fail/Warning/NoTest</td>
</tr>
<tr>
<td>8.1.5#4</td>
<td>An HWA will forward all asynchronous notifications (connect/reconnect/disconnect/sleep etc) received from a downstream device to Host software.</td>
<td>N/A (HWA)</td>
<td></td>
</tr>
<tr>
<td>8.1.5#5</td>
<td>An HWA specifies the total number of devices that can be connected to it in the bNumPorts field in its Wire Adapter descriptor.</td>
<td>N/A (HWA)</td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.1.6 Upstream Port**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1.6#1</td>
<td>The upstream port of an HWA must operate at Full-speed and High-speed.</td>
<td>N/A (HWA)</td>
<td></td>
</tr>
<tr>
<td>8.1.6#2</td>
<td>Isochronous transfers must not be supported when the upstream port of an HWA is operating at full speed.</td>
<td>N/A (HWA)</td>
<td></td>
</tr>
<tr>
<td>8.1.6#3</td>
<td>The upstream port of a DWA must be compliant to the device side interface of a Wireless USB device which is described in Chapter 7 of the Wireless USB specification.</td>
<td>WUSBCV V</td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.1.9 Remote Pipe Controller**

**Subsection reference: 8.1.9.1 RPipe Descriptor**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1.9.1#1</td>
<td>The RPipe Descriptor of a WA must be able to be overwritten to retarget the Remote Pipe at a different endpoint using another SetRPipeDescriptor request.</td>
<td>TD.1.28</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TD.1.53</td>
<td></td>
</tr>
<tr>
<td>8.1.9.1#2</td>
<td>A WA must only accept SetRPipeDescriptor requests when that Rpipe is in the Idle or UnConfigured state.</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.1.9.2 Bulk OUT Overview**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1.9.2#1</td>
<td>A WA is required to move the data portions of a Bulk OUT data stream sent by the host software to the Wired or Wireless USB Endpoint in the same order as the host sent them.</td>
<td>TD.1.20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TD.1.21</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TD.1.22</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TD.1.23</td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.1.9.3 Bulk IN Overview**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1.9.3#1</td>
<td>A WA must be able to handle a Bulk IN transfer request up to size $2^{32} - 1$. (If the client buffer is larger than the WA has buffering for, host software will split the buffer into multiple segments that the WA can accommodate and then manage the appropriate short packet semantics when short packets occur in the data stream.)</td>
<td>TD.1.40</td>
<td></td>
</tr>
<tr>
<td>8.1.9.3#2</td>
<td>Whenever a WA observes that the associated Wired or Wireless USB IN Endpoint provides a short packet, the Wire Adapter must send a transfer complete notification to the host on the Notification endpoint and send the residual queued data with a transfer status indicating the transfer request</td>
<td>TD.1.41</td>
<td></td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
<td>Action:</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------</td>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>is completed to the host on the Data transfer read endpoint.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.1.9.4 Control Transfer Overview**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action:</th>
<th>Fail/ Warning/ NoTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1.9.4#1</td>
<td>A WA must only send the Setup data included in the transfer request segment with segment number zero of a multi-segment control transfer request.</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.1.9.4#2</td>
<td>A WA must ignore the Setup data included in transfer request segments with segment number other than zero.</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.1.9.5 Interrupt Transfer Overview**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action:</th>
<th>Fail/ Warning/ NoTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1.9.5#1</td>
<td>A WA must service an Interrupt endpoint at least as frequent as the period requested by the bInterval field in the Remote Pipe descriptor.</td>
<td>TD.1.30 TD.1.31</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.1.10 Suspend and Resume**

**Subsection reference: 8.1.10.1 DWA Suspend and Resume**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action:</th>
<th>Fail/ Warning/ NoTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1.10.1#1</td>
<td>A DWA must always attempt to propagate resume signaling, regardless of whether it has been enabled for remote wake itself.</td>
<td>USBCV Hubtests</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.1.10.1#2</td>
<td>A DWA must wake from the Sleep state least once every TrustTimeout period.</td>
<td>USBCV Hubtests</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.1.10.1#3</td>
<td>If a DWA is sleeping while its wireless host is awake, and if any event originates from a suspended downstream port of the DWA, the DWA must resume the downstream port and the DWA must send Reconnect to the host or the DWA must send DN_Alive to the host.</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.1.10.1#4</td>
<td>If a DWA is sleeping and its wireless host has stopped the Wireless USB channel, the DWA must check for host awake and the DWA must ignore any event on its downstream ports.</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.1.10.1#5</td>
<td>If a DWA is sleeping with remote wake disabled and the host has stopped the Wireless USB channel, but resumes the Wireless USB Channel on a periodic basis, the DWA must check for host awake and the DWA must ignore any event on its downstream ports except for resume signaling.</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.1.10.1#6</td>
<td>If a DWA is sleeping with remote wake disabled and the host has stopped the Wireless USB channel, but resumes the Wireless USB Channel on a periodic basis and when the DWA detects resume signaling on a downstream port, the DWA must send Remote Wake Notification, send Reconnect request and send wake event notification to the host.</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.1.10.1#7</td>
<td>If a DWA is sleeping with remote wake enabled and the host has stopped the Wireless USB channel, but resumes the Wireless USB Channel on a periodic basis, the DWA must check for host awake and the DWA must react to any wake event on its downstream bus.</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.1.10.1#8</td>
<td>If a DWA is sleeping with remote wake enabled and the host has stopped the Wireless USB channel, but resumes the Wireless USB Channel on a</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Assertion # Assertion Description Test # Action:

periodic basis and when the DWA detects any event on a downstream port, the DWA must send Remote Wake Notification, send Reconnect request and send wake event notification to the host.  

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1.11#1</td>
<td>After bus reset, a WA must clear all the status, state machines and registers and set default values in the descriptors.</td>
<td>TD.1.47 TD.1.67</td>
<td>Fail/Warning/NoTest</td>
</tr>
<tr>
<td>8.1.11#2</td>
<td>A DWA must reset itself after receiving a SetAddress (0) command or a Wireless USB Reset DeviceIE.</td>
<td>TD.1.47 TD.1.67</td>
<td>FAIL/WARNING/NOTEST</td>
</tr>
<tr>
<td>8.1.11#3</td>
<td>A DWA must reset the port status after receiving a SetFeature(WIRE_ADAPTER_RESET) request.</td>
<td>TD.1.18</td>
<td>FAIL/WARNING/NOTEST</td>
</tr>
<tr>
<td>8.1.11#4</td>
<td>An HWA must disable the schedule and stop sending MMC’s after receiving a Set Feature(WIRE_ADAPTER_RESET) request.</td>
<td>N/A (HWA)</td>
<td>FAIL/WARNING/NOTEST</td>
</tr>
</tbody>
</table>

### Subsection reference: 8.1.11 Reset Behavior

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1.13#1</td>
<td>If the number of buffer blocks per RPipe is dynamically manageable by host software then a value of “zero” must be reported in the wBlocks field of an RPipe Descriptor after reset.</td>
<td>TD.1.47</td>
<td>FAIL/WARNING/NOTEST</td>
</tr>
</tbody>
</table>

### Subsection reference: 8.3 Requests

### Subsection reference: 8.3.1 Wire Adapter Class-Specific Requests

### Subsection reference: 8.3.1.1 Abort RPipe

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3.1.1#1</td>
<td>A WA must terminate all pending transfers for the given RPipe and place the RPipe in the Idle state upon receipt of an Abort_RPipe(RPipe Index) request.</td>
<td>TD.1.13</td>
<td>FAIL/WARNING/NOTEST</td>
</tr>
<tr>
<td>8.3.1.1#2</td>
<td>A WA must return a transfer completion notification, transfer result that indicates that the Transfer Request was Aborted and any data that was received and acknowledged from the targeted endpoint for all terminated transfers after receipt of an Abort_RPipe(RPipe Index) request.</td>
<td>TD.1.13</td>
<td>FAIL/WARNING/NOTEST</td>
</tr>
<tr>
<td>8.3.1.1#3</td>
<td>A WA must respond with a Request Error to an Abort_RPipe(wValue, RPipe Index, wLength) request with wValue or wLength other than 0 or with wIndex specifying an RPipe that does not exist.</td>
<td>N/A</td>
<td>FAIL/WARNING/NOTEST</td>
</tr>
</tbody>
</table>

### Subsection reference: 8.3.1.2 Clear RPipe Feature

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3.1.2#1</td>
<td>A WA must disable the RPIPE_PAUSE feature and return to its state before it was paused upon receipt of a Clear_Feature(RPIPE_PAUSE, RPipe Index, 0).</td>
<td>TD.1.14</td>
<td>FAIL/WARNING/NOTEST</td>
</tr>
<tr>
<td>8.3.1.2#2</td>
<td>A WA must respond with a Request Error to a Clear_Feature(Feature selector, RPipe Index, wLength) if wValue is not the feature selector RPIPE_PAUSE, if wIndex specifies an RPipe that does not exist or if</td>
<td>TD.1.56</td>
<td>FAIL/WARNING/NOTEST</td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
<td>Action: Fail/Warning/NoTest</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
</tr>
<tr>
<td></td>
<td>wLength is other than 0.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.3.1.3 Clear Wire Adapter Feature**

| 8.3.1.3#1  | A WA must disable the WIRE_ADAPTER_ENABLE feature upon receipt of a Clear_Feature(WIRE_ADAPTER_ENABLE, Interface Number, 0). | TD.1.17  |                            |
| 8.3.1.3#2  | A WA must respond with a Request Error to a Clear_Feature(Feature Selector, Interface Number, wLength) if wValue is not the feature selector WIRE_ADAPTER_ENABLE, if wIndex specifies an interface that does not exist or if wLength is other than 0. | TD.1.55  |                            |

**Subsection reference: 8.3.1.4 Get RPipe Descriptor**

| 8.3.1.4#1  | A WA must respond to a Get_Descriptor(Descriptor Type, RPipe Index, Descriptor Length) with the actual length of the RPipe Descriptor if the requested wLength was larger than the actual length of the descriptor. | TD.1.15  |                            |
| 8.3.1.4#2  | A WA must respond to a Get_Descriptor(Descriptor Type, RPipe Index, Descriptor Length) with the first wLength bytes of the RPipe Descriptor if the requested wLength was less than the actual length of the descriptor. | TD.1.15  |                            |
| 8.3.1.4#3  | A WA must respond with a Request Error to a Get_Descriptor(Descriptor Type, RPipe Index, Descriptor Length) if wValue is not a valid Descriptor Type or if wIndex is not a valid RPipe Index. | N/A      |                            |

**Subsection reference: 8.3.1.5 Get RPipe Status**

<p>| 8.3.1.5#1  | A WA must respond with the RPipe Status of 1 byte to a valid Get_Status(RPipe Index). | TD.1.14  |                            |
| 8.3.1.5#2  | If a WA RPipe is in Idle state, a WA must respond with the RPipe Status with Bit0 set to 0 to a valid Get_Status(RPipe Index). | TD.1.14  |                            |
| 8.3.1.5#3  | If a WA RPipe is in Active state, a WA must respond with the RPipe Status with Bit0 set to 1 to a valid Get_Status(RPipe Index). | TD.1.14  |                            |
| 8.3.1.5#4  | If a WA RPipe is Not Paused, a WA must respond with the RPipe Status with Bit1 set to 0 to a valid Get_Status(RPipe Index). | TD.1.14  |                            |
| 8.3.1.5#5  | If a WA RPipe is Paused, a WA must respond with the RPipe Status with Bit1 set to 1 to a valid Get_Status(RPipe Index). | TD.1.14  |                            |
| 8.3.1.5#6  | If a WA RPipe is unconfigured, a WA must respond with the RPipe Status with Bit2 set to 0 to a valid Get_Status(RPipe Index). | TD.1.14  |                            |
| 8.3.1.5#7  | If a WA RPipe is configured, a WA must respond with the RPipe Status with Bit2 set to 1 to a valid Get_Status(RPipe Index). | TD.1.14  |                            |
| 8.3.1.5#8  | A WA must respond with a Request Error to a Get_Status(wValue, RPipe Index, wLength) if wValue is not 0, if wIndex specifies an RPipe that does not exist or if wLength is other than 1. | N/A      |                            |
| 8.3.1.5#9  | A WA RPipe Status bit7:3 are reserved and must be set to zero. | TD.1.14  |                            |</p>
<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action: Fail/Warning/NoTest</th>
</tr>
</thead>
</table>

### Subsection reference: 8.3.1.6 Get Wire Adapter Status

<table>
<thead>
<tr>
<th>8.3.1.6#1</th>
<th>A WA must respond to a valid Get_Status(Interface Number) with the Wire Adapter Status with Bit0 set to 0 if the WA Controller is disabled.</th>
<th>TD.1.19</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3.1.6#2</td>
<td>A WA must respond to a valid Get_Status(Interface Number) with the Wire Adapter Status with Bit0 set to 1 if the WA Controller is enabled.</td>
<td>TD.1.19</td>
</tr>
<tr>
<td>8.3.1.6#3</td>
<td>A WA must respond to a valid Get_Status(Interface Number) with the Wire Adapter Status with Bit1 set to 1 if a Reset is in progress.</td>
<td>TD.1.19</td>
</tr>
<tr>
<td>8.3.1.6#4</td>
<td>A WA must respond to 0 to a valid Get_Status(Interface Number) with the Wire Adapter Status with Bit1 set if no Reset is in progress. (e.g. Reset completed)</td>
<td>TD.1.19</td>
</tr>
<tr>
<td>8.3.1.6#5</td>
<td>A WA must respond to 0 to a valid Get_Status(Interface Number) with the Wire Adapter Status with Bits 31:2 set.</td>
<td>TD.1.19</td>
</tr>
<tr>
<td>8.3.1.6#6</td>
<td>A WA must respond with a Request Error to Get_Status(wValue, Interface Number, wLength) if wValue is not 0, if wIndex specifies an Interface that does not exist or if wLength is other than 4.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### Subsection reference: 8.3.1.7 Set RPipe Descriptor

<table>
<thead>
<tr>
<th>8.3.1.7#1</th>
<th>A WA must support a valid Set_Descriptor(Descriptor Type, RPipe Index, Descriptor Length) request.</th>
<th>TD.1.15</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3.1.7#2</td>
<td>A WA must respond with a Request Error to a Set_Descriptor(Descriptor Type, RPipe Index, Descriptor Length) request if wValue is not a valid Descriptor Type, if wIndex specifies an RPipe Index that does not exist or if wLength is not equal to the RPipe Descriptor Length.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### Subsection reference: 8.3.1.8 Set RPipe Feature

<table>
<thead>
<tr>
<th>8.3.1.8#1</th>
<th>A WA pause all its pending transfer requests on the RPipe upon receipt of a Set_Feature(RPIPE_PAUSE, RPipe Index, 0) request.</th>
<th>TD.1.14</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3.1.8#2</td>
<td>A WA must respond with a Request Error to a Set_Feature(Feature Selector, RPipe Index, wLength) request if wValue is not the Feature Selector RPIPE_PAUSE, if wIndex specifies an RPipe Index that does not exist or if wLength is other than 0.</td>
<td>TD.1.57</td>
</tr>
</tbody>
</table>

### Subsection reference: 8.3.1.9 Set Wire Adapter Feature

<table>
<thead>
<tr>
<th>8.3.1.9#1</th>
<th>A WA must support a valid Set_Feature(WIRE_ADAPTER_ENABLE, Interface Number, 0) request.</th>
<th>TD.1.17</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3.1.9#2</td>
<td>A WA must respond with a Request Error to a Set_Feature(Feature Selector, Interface Number, wLength) request if wValue is not the Feature Selector WIRE_ADAPTER_ENABLE or WIRE_ADAPTER_RESET, if wIndex specifies an Interface that does not exist or if wLength is other than 0.</td>
<td>TD.1.54</td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.3.1.10 Reset RPipe Feature**

8.3.1.10#1  A WA must reset its RPipe to the UnConfigured State upon receipt of a valid Reset_RPipe(RPipe Index) request for that RPipe.  TD.1.16

8.3.1.10#2  A WA must respond with a Request Error to a Reset_RPipe(wValue, RPipe Index, wLength) request if wValue is other than 0, if wIndex specifies an RPipe Index that does not exist or if wLength is other than 0.  N/A

**Subsection reference: 8.3.3 Transfer Requests**

8.3.3#1  A WA’s R Pipes must support at least two concurrent requests per Interrupt RPipe in order to support Interrupt transfers.  TD.1.12  TD.1.22

8.3.3#2  An HWA must support at least four concurrent requests per Isochronous RPipe to support Isochronous transfers.  N/A  (HWA)

**Subsection reference: 8.3.3.1 Control Transfers**

8.3.3.1#1  A WA must support Control Transfer Requests.  TD.1.20

8.3.3.1#2  A WA must send back intermediate transfer completion notifications and transfer results whenever a Transfer Request segment completes.  TD.1.34

8.3.3.1#3  A WA must send the contents of baSetupData downstream to the device, when the Segment Number bits (Bit 6:0) of the bTransferSegment field of a Control Transfer Request are 0.  N/A

8.3.3.1#4  A WA must perform a status stage transaction, when the Last Segment bit (Bit 7) of the bTransferSegment field of a Control Transfer Request is 1.  TD.1.20

8.3.3.1#5  A WA must perform a status stage transaction, when it receives a short packet from the device during a Control Transfer Request.  TD.1.20

8.3.3.1#6  An HWA must perform secured control transfers to the device when Control Transfer Request bmAttribute field bit 1 is set to zero.  N/A  (HWA)

8.3.3.1#7  An HWA must perform unsecured control transfers to the device when Control Transfer Request bmAttribute field bit 1 is set to one.  N/A  (HWA)

8.3.3.1#8  A DWA must ignore the Control Transfer Request bmAttribute field bit 1.  N/A

**Subsection reference: 8.3.3.2 Bulk and Interrupt Transfers**

N/A  No assertions  N/A

**Subsection reference: 8.3.3.3 Transfer Completion Notification**

8.3.3.3#1  A WA Transfer Completion Notification bLength field must be set to 4 decimal.  TD.1.34  TD.1.35

8.3.3.3#2  A WA Transfer Completion Notification bReserved field must be set to zero.  TD.1.34  TD.1.35

**Subsection reference: 8.3.3.4 Transfer Result**
<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action: Fail/Warning/NoTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3.3.4#1</td>
<td>A WA must present the Transfer Result at the Data Transfer Read endpoint number indicated in the previous transfer completion notification.</td>
<td>TD.1.34</td>
<td>TD.1.35 TD.1.36 TD.1.37 TD.1.38 TD.1.39</td>
</tr>
<tr>
<td>8.3.3.4#2</td>
<td>If the corresponding transfer was an IN transfer (Bulk/Interrupt IN or Control Transfer Read), a WA must be able to return the transfer result and the IN data as separate and consecutive transfers to the host.</td>
<td>TD.1.35</td>
<td>TD.1.37 TD.1.39</td>
</tr>
<tr>
<td>8.3.3.4#3</td>
<td>A WA Transfer Result bLength field must be set to 10H.</td>
<td>TD.1.34</td>
<td>TD.1.35</td>
</tr>
<tr>
<td>8.3.3.4#4</td>
<td>After receiving a transfer completion notification a WA must present a Transfer Result. (This Transfer Result is identified by bResultType field of 83H.)</td>
<td>TD.1.34</td>
<td>TD.1.35 TD.1.36 TD.1.37 TD.1.38 TD.1.39</td>
</tr>
<tr>
<td>8.3.3.4#5</td>
<td>A WA Transfer Result dwTransferID field must be set to the host assigned ID for the transfer.</td>
<td>TD.1.34</td>
<td>TD.1.35 TD.1.36 TD.1.37 TD.1.38 TD.1.39</td>
</tr>
<tr>
<td>8.3.3.4#6</td>
<td>A WA Transfer Result dwTransferLength field must be set to the amount of data transferred for either OUT or IN.</td>
<td>TD.1.34</td>
<td>TD.1.35 TD.1.36 TD.1.37 TD.1.38 TD.1.39</td>
</tr>
<tr>
<td>8.3.3.4#7</td>
<td>A WA Transfer Result bTransferSegment field Bit6:0 must be set to the Segment Number.</td>
<td>TD.1.34</td>
<td>TD.1.35 TD.1.36 TD.1.37 TD.1.38 TD.1.39</td>
</tr>
<tr>
<td>8.3.3.4#8</td>
<td>A WA Transfer Result bTransferSegment field Bit7 must be set to one if the transfer result describes the last segment of the transfer.</td>
<td>TD.1.34</td>
<td>TD.1.35 TD.1.36 TD.1.37 TD.1.38 TD.1.39</td>
</tr>
<tr>
<td>8.3.3.4#9</td>
<td>A WA must report TRANSFER_STATUS_SUCCESS in the bTransferStatus field of a Transfer Result if the transfer completed successfully.</td>
<td>TD.1.34</td>
<td>TD.1.35 TD.1.36 TD.1.37 TD.1.38 TD.1.39</td>
</tr>
<tr>
<td>8.3.3.4#10</td>
<td>A WA must report TRANSFER_STATUS_HALTED in the bTransferStatus.</td>
<td>TD.1.42</td>
<td>-</td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
<td>Action: Fail/Warning/NoTest</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------</td>
<td>---------------------------</td>
</tr>
<tr>
<td></td>
<td>field of a Transfer Result if the endpoint that the transfer was attempted on is currently halted. A WA must report a TRANSFER STATUS_HALTED by setting Bits 5:0 of bTransferStatus to Status Value 1 decimal and setting Bits 7:6 of bTransferStatus to 10b.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.3.3.4#11</td>
<td>A WA must report TRANSFER_STATUS_DATA_BUFFER_ERROR in the bTransferStatus field of a Transfer Result if a data buffer under/over run occurred. A WA must report a TRANSFER_STATUS_DATA_BUFFER_ERROR by setting Bits 5:0 of bTransferStatus to Status Value 2 decimal and setting Bits 7:6 of bTransferStatus to 10b.</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.3.3.4#12</td>
<td>A WA must report TRANSFER_STATUS_BABBLE in the bTransferStatus field of a Transfer Result if babble was detected on the transfer. This can be frame babble, packet babble or both. A WA must report a TRANSFER_STATUS_BABBLE by setting Bits 5:0 of bTransferStatus to Status Value 3 decimal and setting Bits 7:6 of bTransferStatus to 10b.</td>
<td>TD.1.43</td>
<td></td>
</tr>
<tr>
<td>8.3.3.4#13</td>
<td>A WA must report TRANSFER_STATUS_NOT_FOUND in the bTransferStatus field of a Transfer Result in response to an Abort Transfer request that has an invalid or already completed TransferID. A WA must report a TRANSFER_STATUS_NOT_FOUND by setting Bits 5:0 of bTransferStatus to Status Value 5 decimal and setting Bits 7:6 of bTransferStatus to 10b.</td>
<td>TD.1.63</td>
<td></td>
</tr>
<tr>
<td>8.3.3.4#14</td>
<td>A WA must report TRANSFER_STATUS_INSUFFICIENT_RESOURCE in the bTransferStatus field of a Transfer Result if the Wire Adapter could not get enough resources to complete a previously accepted transfer request. A WA must report a TRANSFER_STATUS_INSUFFICIENT_RESOURCE by setting Bits 5:0 of bTransferStatus to Status Value 6 decimal and setting Bits 7:6 of bTransferStatus to 10b.</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.3.3.4#15</td>
<td>A WA must report TRANSFER_STATUS_TRANSACTION_ERROR in the bTransferStatus field of a Transfer Result if the transfer completed successfully but transaction errors occurred which were successfully retried. A WA must report a TRANSFER_STATUS_TRANSACTION_ERROR by setting Bits 5:0 of bTransferStatus to Status Value 7 decimal and setting Bits 7:6 of bTransferStatus to 01b.</td>
<td>TD.1.45</td>
<td></td>
</tr>
<tr>
<td>8.3.3.4#16</td>
<td>A WA must report TRANSFER_STATUS_TRANSACTION_ERROR in the bTransferStatus field of a Transfer Result if the transaction failed after the number of retry attempts specified in bmRetryOptions field of the RPipe descriptor. A WA must report a TRANSFER_STATUS_TRANSACTION_ERROR by setting Bits 5:0 of bTransferStatus to Status Value 7 decimal and setting Bits 7:6 of bTransferStatus to 10b.</td>
<td>TD.1.44</td>
<td></td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
<td>Action: Fail/Warning/NoTest</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>8.3.3.4#17</td>
<td>A WA must report TRANSFER_STATUS_ABORTED in the bTransferStatus field of a Transfer Result if the transfer was aborted by an Abort Transfer Request or by an AbortRPipe command. A WA must report a TRANSFER_STATUS_ABORTED by setting Bits 5:0 of bTransferStatus to Status Value 8 decimal and setting Bits 7:6 of bTransferStatus to 10b.</td>
<td>TD.1.46</td>
<td></td>
</tr>
<tr>
<td>8.3.3.4#18</td>
<td>A WA must report TRANSFER_STATUS_RPIPE_NOT_READY in the bTransferStatus field of a Transfer Result if the transfer request was sent to an unconfigured RPipe. A WA must report a TRANSFER_STATUS_RPIPE_NOT_READY by setting Bits 5:0 of bTransferStatus to Status Value 9 decimal and setting Bits 7:6 of bTransferStatus to 10b.</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.3.3.4#19</td>
<td>A WA must report INVALID_REQUEST_FORMAT in the bTransferStatus field of a Transfer Result if the transfer request length was not equal to the length field for the specified request type. A WA must report an INVALID_REQUEST_FORMAT by setting Bits 5:0 of bTransferStatus to Status Value 10 decimal and setting Bits 7:6 of bTransferStatus to 10b.</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.3.3.4#20</td>
<td>A WA must report INVALID_REQUEST_FORMAT in the bTransferStatus field of a Transfer Result if the request type was unknown. A WA must report an INVALID_REQUEST_FORMAT by setting Bits 5:0 of bTransferStatus to Status Value 10 decimal and setting Bits 7:6 of bTransferStatus to 10b.</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.3.3.4#21</td>
<td>A WA must report UNEXPECTED_SEGMENT_NUMBER in the bTransferStatus field of a Transfer Result if the transfer request segment numbers were not received in incrementing order starting with zero. A WA must report an UNEXPECTED_SEGMENT_NUMBER by setting Bits 5:0 of bTransferStatus to Status Value 11 decimal and setting Bits 7:6 of bTransferStatus to 10b.</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.3.3.4#22</td>
<td>A WA must report TRANSFER_STATUS_RPIPE_TYPE_MISMATCH in the bTransferStatus field of a Transfer Result if the transfer type in the transfer request did not match the transfer type that the RPipe was previously configured to. A WA must report a TRANSFER_STATUS_RPIPE_TYPE_MISMATCH by setting Bits 5:0 of bTransferStatus to Status Value 12 decimal and setting Bits 7:6 of bTransferStatus to 10b.</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.3.3.4#23</td>
<td>A DWA Transfer Result dwNumOfPackets field must be set to zero.</td>
<td>TD.1.34 TD.1.35 TD.1.36 TD.1.37 TD.1.38T D.1.39</td>
<td></td>
</tr>
<tr>
<td>8.3.3.4#24</td>
<td>An HWA Transfer Result dwNumOfPackets field must be set to zero for</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action: Fail/Warning/NoTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>non Isochronous transfers.</td>
<td>(HWA)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.3.3.4#25</td>
<td>An HWA Transfer Result dwNumOfPackets field for an Isochronous transfer must be set to the Number of Packet lengths and status following.</td>
<td>N/A</td>
<td>(HWA)</td>
</tr>
<tr>
<td>Subsection reference: 8.3.3.5 Abort Transfer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.3.3.5#1</td>
<td>A WA must ACK a valid Abort Transfer Request.</td>
<td>TD.1.46</td>
<td></td>
</tr>
<tr>
<td>Subsection reference: 8.4 DWA Interfaces, Descriptors and Control</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subsection reference: 8.4.1 DWA Isochronous Streaming Interface</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.4.1#1</td>
<td>A DWA must support a number of simultaneous downstream isochronous streams equal to the number of isochronous endpoints in the DWA Isochronous Streaming Interface.</td>
<td>TD.1.52</td>
<td></td>
</tr>
<tr>
<td>Subsection reference: 8.4.2 DWA Isochronous Streaming Overview</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.4.2#1</td>
<td>A DWA must only support isochronous transfers if it has one or more upstream Wireless USB isochronous endpoints.</td>
<td>TD.1.3</td>
<td></td>
</tr>
<tr>
<td>Subsection reference: 8.4.3 DWA Descriptors</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subsection reference: 8.4.3.1 Device Descriptor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.4.3.1#1</td>
<td>A DWA Device Descriptor bLength field must have a value of 12H.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.1#2</td>
<td>A DWA Device Descriptor bDescriptorType field must have a value of 1 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.1#3</td>
<td>A DWA Device Descriptor bcdUSB field must have a value of 250H.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.1#4</td>
<td>A DWA Device Descriptor bMaxPacketSize0 field must have a value of FFH.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.1#5</td>
<td>A DWA Device Descriptor bNumConfigurations field must have a value of 1 decimal.</td>
<td>TD.1.2</td>
<td></td>
</tr>
<tr>
<td>8.4.3.1#6</td>
<td>If a DWA exports an Isochronous interface then it must present an Interface Association Descriptor to group the two interfaces (Data Transfer Interface and Isochronous Streaming Interface) together so that one driver is loaded for both.</td>
<td>TD.1.3</td>
<td>TD.1.4</td>
</tr>
<tr>
<td>8.4.3.1#7</td>
<td>If a DWA exports an Isochronous interface a DWA Device Descriptor bDeviceClass field must be set to EFH.</td>
<td>TD.1.2</td>
<td></td>
</tr>
<tr>
<td>8.4.3.1#8</td>
<td>If a DWA exports an Isochronous interface a DWA Device Descriptor bDeviceSubClass field must be set to 02H.</td>
<td>TD.1.2</td>
<td></td>
</tr>
<tr>
<td>8.4.3.1#9</td>
<td>If a DWA exports an Isochronous interface a DWA Device Descriptor bDeviceProtocol field must be set to 02H.</td>
<td>TD.1.2</td>
<td></td>
</tr>
<tr>
<td>8.4.3.1#10</td>
<td>If a DWA does not export an Isochronous interface, the DWA Device Descriptor bDeviceClass field, bDeviceSubClass field and bDeviceProtocol field must all be set to 0x0 or must be set to E0H, 02H</td>
<td>TD.1.2</td>
<td></td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test</td>
<td>Action: Fail/Warning/NoTest</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td>and 02H respectively.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Subsection reference: 8.4.3.2 Binary Device Object (BOS) Descriptor</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.4.3.2#1</td>
<td>A DWA must define a BOS descriptor.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.2#2</td>
<td>A DWA BOS descriptor must be compliant to section 7.4.1 of the Wireless USB Specification.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.2#3</td>
<td>A DWA must present its BOS descriptor upon a GetDescriptor() request with a descriptor type set to BOS.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.2#4</td>
<td>A DWA must always have a Wireless USB Device Capabilities on UWB descriptor as part of its BOS Descriptor set.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td><strong>Subsection reference: 8.4.3.3 Configuration Descriptor</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.4.3.3#1</td>
<td>A DWA Configuration Descriptor bLength field must be set to 9 decimal.</td>
<td>TD.1.3</td>
<td></td>
</tr>
<tr>
<td>8.4.3.3#2</td>
<td>A DWA Configuration Descriptor bDescriptorType field must be set to 2 decimal.</td>
<td>TD.1.3</td>
<td></td>
</tr>
<tr>
<td>8.4.3.3#3</td>
<td>A DWA Configuration Descriptor wTotalLength field must be set to the total length of all descriptors in the configuration.</td>
<td>TD.1.3</td>
<td></td>
</tr>
<tr>
<td>8.4.3.3#4</td>
<td>A DWA Configuration Descriptor bNumInterfaces field must be set to the number of interfaces included in the configuration.</td>
<td>TD.1.3</td>
<td></td>
</tr>
<tr>
<td>8.4.3.3#5</td>
<td>A DWA Configuration Descriptor bConfigurationValue field must be set to the value to use to reference the configuration.</td>
<td>TD.1.3</td>
<td></td>
</tr>
<tr>
<td>8.4.3.3#6</td>
<td>A DWA Configuration Descriptor iConfiguration field must be set to the index of the String Descriptor describing the configuration.</td>
<td>TD.1.3</td>
<td></td>
</tr>
<tr>
<td>8.4.3.3#7</td>
<td>A DWA Configuration Descriptor bmAttributes bit 3:0 must be set to zero (0000b).</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.3#8</td>
<td>A DWA Configuration Descriptor bmAttributes bit 4 must be set to one (1b) if the DWA is battery powered.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.3#9</td>
<td>A DWA Configuration Descriptor bmAttributes bit 5 must be set to one (1b) if the DWA supports Remote Wakeup.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.3#10</td>
<td>A DWA Configuration Descriptor bmAttributes bit 6 must be set to one (1b).</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.3#11</td>
<td>A DWA Configuration Descriptor bmAttributes bit 7 must be set to one (1b).</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.3#12</td>
<td>A DWA Configuration Descriptor bMaxPower must be set to zero (0b).</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td><strong>Subsection reference: 8.4.3.4 Security Descriptors</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.4.3.4#1</td>
<td>A DWA must respond with its security descriptors upon a Get Descriptor (SECURITY type) request.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.4#2</td>
<td>DWA Security Descriptors and Key descriptors should be compliant to</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
<td>Action: Fail/Warning/NoTest</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------</td>
<td>-------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>7.4.5 of the Wireless USB Specification.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.4.3.5 Interface Association Descriptor**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.3.5#1</td>
<td>A DWA Interface Association Descriptor bLength field must be set to 8 decimal.</td>
<td>TD.1.4</td>
</tr>
<tr>
<td>8.4.3.5#2</td>
<td>A DWA Interface Association Descriptor bDescriptorType field must be set to 0BH.</td>
<td>TD.1.4</td>
</tr>
<tr>
<td>8.4.3.5#3</td>
<td>A DWA Interface Association Descriptor bFirstInterface field must be set to zero.</td>
<td>TD.1.4</td>
</tr>
<tr>
<td>8.4.3.5#4</td>
<td>A DWA Interface Association Descriptor bInterfaceCount field must be set to 2 decimal.</td>
<td>TD.1.4</td>
</tr>
<tr>
<td>8.4.3.5#5</td>
<td>A DWA Interface Association Descriptor bFunctionClass field must be set to E0H.</td>
<td>TD.1.4</td>
</tr>
<tr>
<td>8.4.3.5#6</td>
<td>A DWA Interface Association Descriptor bFunctionSubClass field must be set to 02H.</td>
<td>TD.1.4</td>
</tr>
<tr>
<td>8.4.3.5#7</td>
<td>A DWA Interface Association Descriptor bFunctionProtocol field must be set to 02H.</td>
<td>TD.1.4</td>
</tr>
</tbody>
</table>

**Subsection reference: 8.4.3.6 Data Transfer Interface Descriptor**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.3.6#1</td>
<td>A DWA Data Transfer Interface Descriptor bLength field must be set to 9 decimal.</td>
<td>WUSBCV</td>
</tr>
<tr>
<td>8.4.3.6#2</td>
<td>A DWA Data Transfer Interface Descriptor bDescriptorType field must be set to 4 decimal.</td>
<td>WUSBCV</td>
</tr>
<tr>
<td>8.4.3.6#3</td>
<td>A DWA Data Transfer Interface Descriptor bInterfaceNumber field must be set to zero.</td>
<td>TD.1.5</td>
</tr>
<tr>
<td>8.4.3.6#4</td>
<td>A DWA Data Transfer Interface Descriptor bAlternateSetting field must be set to zero.</td>
<td>TD.1.5</td>
</tr>
<tr>
<td>8.4.3.6#5</td>
<td>A DWA Data Transfer Interface Descriptor bNumEndpoints field must be set to 3 decimal.</td>
<td>TD.1.5</td>
</tr>
<tr>
<td>8.4.3.6#6</td>
<td>A DWA Data Transfer Interface Descriptor bInterfaceClass field must be set to E0H.</td>
<td>TD.1.5</td>
</tr>
<tr>
<td>8.4.3.6#7</td>
<td>A DWA Data Transfer Interface Descriptor bInterfaceSubclass field must be set to 02H.</td>
<td>TD.1.5</td>
</tr>
<tr>
<td>8.4.3.6#8</td>
<td>A DWA Data Transfer Interface Descriptor bInterfaceProtocol field must be set to 02H.</td>
<td>TD.1.5</td>
</tr>
</tbody>
</table>

**Subsection reference: 8.4.3.7 Wire Adapter Class Descriptor**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.3.7#1</td>
<td>A DWA Wire Adapter Class Descriptor bLength field must accurately reflect the length of the descriptor. (13 bytes + DeviceRemovable field size)</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------</td>
<td>--------</td>
</tr>
<tr>
<td>8.4.3.7#2</td>
<td>A DWA Wire Adapter Class Descriptor bDescriptorType field must be set to 21H.</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#3</td>
<td>A DWA Wire Adapter Class Descriptor bcdWAVersion field must be set to 0100H.</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#4</td>
<td>A DWA Wire Adapter Class Descriptor bNumPorts field must be set to the total number of ports supported by the DWA (including non exposed ports).</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#5</td>
<td>A DWA Wire Adapter Class Descriptor bNumPorts field must be set to one or greater.</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#6</td>
<td>A DWA Wire Adapter Class Descriptor bmAttributes bit 0 must be set to zero (0b) if the DWA uses Ganged power switching.</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#7</td>
<td>A DWA Wire Adapter Class Descriptor bmAttributes bit 0 must be set to one (1b) if the DWA uses Individual port power switching.</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#8</td>
<td>A DWA Wire Adapter Class Descriptor bmAttributes bit 1 must be set to zero (0b) if the DWA uses Global Over-current Protection.</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#9</td>
<td>A DWA Wire Adapter Class Descriptor bmAttributes bit 1 must be set to one (1b) if the DWA uses Individual port Over-current Protection.</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#10</td>
<td>A DWA Wire Adapter Class Descriptor bmAttributes bit 2 must be set to zero (0b) if the DWA does not support Port Indicators on its downstream ports.</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#11</td>
<td>A DWA Wire Adapter Class Descriptor bmAttributes bit 2 must be set to one (1b) if the DWA support Port Indicators on its downstream ports.</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#12</td>
<td>A DWA Wire Adapter Class Descriptor bmAttributes bit 6:3 must be set to zero (0000b).</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#13</td>
<td>A DWA Wire Adapter Class Descriptor bmAttributes bit 7 must be set to zero (0b).</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#14</td>
<td>A DWA Wire Adapter Class Descriptor wNumRPipes field must be set to the number of RPipes supported by the DWA.</td>
<td>TD.1.12</td>
</tr>
<tr>
<td>8.4.3.7#15</td>
<td>A DWA Wire Adapter Class Descriptor wNumRPipes field must not be zero.</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#16</td>
<td>A DWA Wire Adapter Class Descriptor wNumRPipes field must contain a value at least 2 x (DWA Wire Adapter Class Descriptor wNumPorts field value).</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#17</td>
<td>A DWA must support a maximum total number of buffer blocks over its RPipes as defined in the wRPipeMaxBlock field of the DWA Wire Adapter Class Descriptor.</td>
<td>TD.1.20 TD.1.21 TD.1.22 TD.1.23</td>
</tr>
<tr>
<td>8.4.3.7#18</td>
<td>A DWA Wire Adapter Class Descriptor bRPipeBlockSize field must be equal to (2^{(bRpipeBlockSize-1)}).</td>
<td>N/A</td>
</tr>
<tr>
<td>8.4.3.7#19</td>
<td>A DWA Wire Adapter Class Descriptor bRPipeBlockSize field must be</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------</td>
<td>--------</td>
</tr>
<tr>
<td>greater than zero.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.4.3.7#20</td>
<td>A DWA Wire Adapter Class Descriptor bPwrOn2PwrGood field must be set to the time (in 2 ms intervals) from the time the power-on sequence begins on a port until the power is good on that port.</td>
<td>N/A</td>
</tr>
<tr>
<td>8.4.3.7#21</td>
<td>A DWA Wire Adapter Class Descriptor bNumMMCIEs field must be set to zero.</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#22</td>
<td>A DWA Wire Adapter Class Descriptor DeviceRemovable field size in bytes must equal the integer value of (bNumPorts / 8) + 1.</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#23</td>
<td>A DWA Wire Adapter Class Descriptor DeviceRemovable bit n must be set to 1b if the device on the port n of the DWA is non-removable.</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#24</td>
<td>A DWA Wire Adapter Class Descriptor DeviceRemovable bit n must be set to 0b if the device on the port n of the DWA is removable or if port n does not exist.</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#25</td>
<td>A DWA Wire Adapter Class Descriptor DeviceRemovable field can not describe more than 127 ports.</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#26</td>
<td>A DWA Wire Adapter Class Descriptor DeviceRemovable bit 0 is reserved and must be set to 0b.</td>
<td>TD.1.6</td>
</tr>
<tr>
<td>8.4.3.7#27</td>
<td>A DWA Wire Adapter Class Descriptor DeviceRemovable field cannot describe more ports than the DWA Wire Adapter Class Descriptor bNumPorts field</td>
<td>TD.1.6</td>
</tr>
</tbody>
</table>

**Subsection reference: 8.4.3.8 Notification Endpoint Descriptor**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action: Fail/Warning/NoTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.3.8#1</td>
<td>A DWA Notification Endpoint Descriptor bLength field must be set to 7 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.8#2</td>
<td>A DWA Notification Endpoint Descriptor bDescriptorType field must be set to 5 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.8#3</td>
<td>A DWA Notification Endpoint Descriptor bEndpointAddress bit 6:4 must be set to zero (000b).</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.8#4</td>
<td>A DWA Notification Endpoint Descriptor bmAttributes field must be set to 03H.</td>
<td>TD.1.7</td>
<td></td>
</tr>
<tr>
<td>8.4.3.8#5</td>
<td>A DWA Notification Endpoint Descriptor wMaxPacketSize field must be set to 200H.</td>
<td>TD.1.7</td>
<td></td>
</tr>
<tr>
<td>8.4.3.8#6</td>
<td>A DWA Notification Endpoint Descriptor bInterval field must be set to 6 decimal.</td>
<td>TD.1.7</td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.4.3.9 Notification Endpoint Companion Descriptor**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action: Fail/Warning/NoTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.3.9#1</td>
<td>A DWA Notification Endpoint Companion Descriptor bLength field must be set to 0AH.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.9#2</td>
<td>A DWA Notification Endpoint Companion Descriptor bDescriptorType field must be set to 17 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
<td>Action: Fail/ Warning/ NoTest</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------</td>
<td>------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>8.4.3.9#3</td>
<td>A DWA Notification Endpoint Companion Descriptor bMaxBurst field must be set to one.</td>
<td>TD.1.8</td>
<td></td>
</tr>
<tr>
<td>8.4.3.9#4</td>
<td>A DWA Notification Endpoint Companion Descriptor bMaxSequence field must contain a value in the range 2 to 32.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.9#5</td>
<td>A DWA Notification Endpoint Companion Descriptor bMaxStreamDelay field must be set to 00H.</td>
<td>TD.1.8</td>
<td></td>
</tr>
<tr>
<td>8.4.3.9#6</td>
<td>A DWA Notification Endpoint Companion Descriptor wOverTheAirPacketSize field must be set to 00H.</td>
<td>TD.1.8</td>
<td></td>
</tr>
<tr>
<td>8.4.3.9#7</td>
<td>A DWA Notification Endpoint Companion Descriptor wOverTheAirInterval field must be set to 00H.</td>
<td>TD.1.8</td>
<td></td>
</tr>
<tr>
<td>8.4.3.9#8</td>
<td>A DWA Notification Endpoint Companion Descriptor bmCompAttributes field must be set to 00H.</td>
<td>TD.1.8</td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.4.3.10 Data Transfer Write Endpoint Descriptor**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action: Fail/ Warning/ NoTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.3.10#1</td>
<td>A DWA Data Transfer Write Endpoint Descriptor bLength field must be set to 7 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.10#2</td>
<td>A DWA Data Transfer Write Endpoint Descriptor bDescriptorType field must be set to 5 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.10#3</td>
<td>A DWA Data Transfer Write Endpoint Descriptor bEndpointAddress bit 6:4 must be set to zero (000b).</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.10#4</td>
<td>A DWA Data Transfer Write Endpoint Descriptor bmAttributes field must have bit 6:0 set to 0000010b.</td>
<td>TD.1.3</td>
<td></td>
</tr>
<tr>
<td>8.4.3.10#5</td>
<td>A DWA Data Transfer Write Endpoint Descriptor bmAttributes bit 7 must be set to one (1b) if Data packet size adjustment is supported by the DWA.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.10#6</td>
<td>A DWA Data Transfer Write Endpoint Descriptor wMaxPacketSize field must be set to a multiple of 512 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.10#7</td>
<td>A DWA Data Transfer Write Endpoint Descriptor wMaxPacketSize field must contain a value between 512 decimal and 3584 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.10#8</td>
<td>A DWA Data Transfer Write Endpoint Descriptor bInterval field must be set to zero.</td>
<td>WUSBCV</td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.4.3.11 Data Transfer Write Endpoint Companion Descriptor**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action: Fail/ Warning/ NoTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.3.11#1</td>
<td>A DWA Data Transfer Write Endpoint Companion Descriptor bLength field must be set to 0AH.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.11#2</td>
<td>A DWA Data Transfer Write Endpoint Companion Descriptor bDescriptorType field must be set to 17 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.11#3</td>
<td>A DWA Data Transfer Write Endpoint Companion Descriptor bMaxBurst field must contain a value in the range 1 to 16.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.11#4</td>
<td>A DWA Data Transfer Write Endpoint must support a maximum burst size of bMaxBurst.</td>
<td>TD.1.20, TD.1.21, TD.1.22</td>
<td></td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
<td>Action:</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------</td>
<td>------------------</td>
</tr>
<tr>
<td>8.4.3.11#5</td>
<td>A DWA Data Transfer Write Endpoint Companion Descriptor bMaxSequence field must contain a value in the range 2 to 32.</td>
<td>TD.1.23</td>
<td>Fail/Warning/NoTest</td>
</tr>
<tr>
<td>8.4.3.11#6</td>
<td>A DWA Data Transfer Write Endpoint must support a maximum sequence of bMaxSequence-1. (With bMaxSequence from the Data Transfer Write Endpoint Companion Descriptor)</td>
<td>TD.1.20</td>
<td>TD.1.21 TD.1.22 TD.1.23</td>
</tr>
<tr>
<td>8.4.3.11#7</td>
<td>A DWA Data Transfer Write Endpoint Companion Descriptor bMaxStreamDelay field must be set to 00H.</td>
<td>TD.1.20</td>
<td>TD.1.21 TD.1.22 TD.1.23</td>
</tr>
<tr>
<td>8.4.3.11#8</td>
<td>A DWA Data Transfer Write Endpoint Companion Descriptor wOverTheAirPacketSize field must be set to 00H.</td>
<td>TD.1.20</td>
<td>TD.1.21 TD.1.22 TD.1.23</td>
</tr>
<tr>
<td>8.4.3.11#9</td>
<td>A DWA Data Transfer Write Endpoint Companion Descriptor wOverTheAirInterval field must be set to 00H.</td>
<td>TD.1.20</td>
<td>TD.1.21 TD.1.22 TD.1.23</td>
</tr>
<tr>
<td>8.4.3.11#10</td>
<td>A DWA Data Transfer Write Endpoint Companion Descriptor bmCompAttributes field must be set to 00H.</td>
<td>TD.1.20</td>
<td>TD.1.21 TD.1.22 TD.1.23</td>
</tr>
</tbody>
</table>

**Subsection reference: 8.4.3.12 Data Transfer Read Endpoint Descriptor**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.3.12#1</td>
<td>A DWA Data Transfer Read Endpoint Descriptor bLength field must be set to 7 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.12#2</td>
<td>A DWA Data Transfer Read Endpoint Descriptor bDescriptorType field must be set to 5 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.12#3</td>
<td>A DWA Data Transfer Read Endpoint Descriptor bEndpointAddress bit 6:4 must be set to zero (000b).</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.12#4</td>
<td>A DWA Data Transfer Read Endpoint Descriptor bmAttributes field must have bit 6:0 set to 0000010b.</td>
<td>TD.1.3</td>
<td></td>
</tr>
<tr>
<td>8.4.3.12#5</td>
<td>A DWA Data Transfer Read Endpoint Descriptor bmAttributes bit 7 must be set to one (1b) if Data packet size adjustment is supported by the DWA.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.12#6</td>
<td>A DWA Data Transfer Read Endpoint Descriptor wMawPacketSize field must be set to a multiple of 512 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.12#7</td>
<td>A DWA Data Transfer Read Endpoint Descriptor wMawPacketSize field must contain a value between 512 decimal and 3584 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.12#8</td>
<td>A DWA Data Transfer Read Endpoint Descriptor bInterval field must be set to zero.</td>
<td>WUSBCV</td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.4.3.13 Data Transfer Read Endpoint Companion Descriptor**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.3.13#1</td>
<td>A DWA Data Transfer Read Endpoint Companion Descriptor bLength field must be set to 0AH.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.13#2</td>
<td>A DWA Data Transfer Read Endpoint Companion Descriptor bDescriptorType field must be set to 17 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.13#3</td>
<td>A DWA Data Transfer Read Endpoint Companion Descriptor bMaxBurst</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
<td>Action: Fail/Warning/NoTest</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------</td>
<td>--------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>8.4.3.13#4</td>
<td>A DWA Data Transfer Read Endpoint must support a maximum burst size of bMaxBurst.</td>
<td>TD.1.20 TD.1.21 TD.1.22 TD.1.23</td>
<td></td>
</tr>
<tr>
<td>8.4.3.13#5</td>
<td>A DWA Data Transfer Read Endpoint Companion Descriptor bMaxSequence field must contain a value in the range 2 to 32.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.13#6</td>
<td>A DWA Data Transfer Read Endpoint must support a maximum sequence of bMaxSequence-1. (With bMaxSequence from the Data Transfer Write Endpoint Companion Descriptor)</td>
<td>TD.1.20 TD.1.21 TD.1.22 TD.1.23</td>
<td></td>
</tr>
<tr>
<td>8.4.3.13#7</td>
<td>A DWA Data Transfer Read Endpoint Companion Descriptor bMaxStreamDelay field must be set to 00H.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.13#8</td>
<td>A DWA Data Transfer Read Endpoint Companion Descriptor wOverTheAirPacketSize field must be set to 00H.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.13#9</td>
<td>A DWA Data Transfer Read Endpoint Companion Descriptor bOverTheAirInterval field must be set to 00H.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.13#10</td>
<td>A DWA Data Transfer Read Endpoint Companion Descriptor bmCompAttributes field must be set to 00H.</td>
<td>WUSBCV</td>
<td></td>
</tr>
</tbody>
</table>

Subsection reference: 8.4.3.14 Isochronous Streaming Interface Descriptor

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action: Fail/Warning/NoTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.3.14#1</td>
<td>A DWA Isochronous Streaming Interface Descriptor bLength field must be set to 9 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.14#2</td>
<td>A DWA Isochronous Streaming Interface Descriptor bDescriptorType field must be set to 4 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.14#3</td>
<td>A DWA Isochronous Streaming Interface Descriptor bInterfaceNumber field must be set to one.</td>
<td>TD.1.9</td>
<td></td>
</tr>
<tr>
<td>8.4.3.14#4</td>
<td>A DWA Isochronous Streaming Interface Descriptor bAlternateSetting field must be set to zero.</td>
<td>TD.1.9</td>
<td></td>
</tr>
<tr>
<td>8.4.3.14#5</td>
<td>A DWA Isochronous Streaming Interface Descriptor bNumEndpoints field must contain a value in the range 1 to 28.</td>
<td>TD.1.9</td>
<td></td>
</tr>
<tr>
<td>8.4.3.14#6</td>
<td>A DWA Isochronous Streaming Interface Descriptor bInterfaceClass field must be set to E0H.</td>
<td>TD.1.9</td>
<td></td>
</tr>
<tr>
<td>8.4.3.14#7</td>
<td>A DWA Isochronous Streaming Interface Descriptor bInterfaceSubClass field must be set to 02H.</td>
<td>TD.1.9</td>
<td></td>
</tr>
<tr>
<td>8.4.3.14#8</td>
<td>A DWA Isochronous Streaming Interface Descriptor bInterfaceProtocol field must be set to 03H.</td>
<td>TD.1.9</td>
<td></td>
</tr>
</tbody>
</table>

Subsection reference: 8.4.3.15 Isochronous Streaming OUT Endpoint Descriptor

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action: Fail/Warning/NoTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.3.15#1</td>
<td>A DWA Isochronous Streaming OUT Endpoint Descriptor bLength field must be set to 7 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
<td>Action: Fail/Warning/NoTest</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>8.4.3.15#2</td>
<td>A DWA Isochronous Streaming OUT Endpoint Descriptor bDescriptorType field must be set to 5 decimal.</td>
<td></td>
<td>WUSBCV</td>
</tr>
<tr>
<td>8.4.3.15#3</td>
<td>A DWA Isochronous Streaming OUT Endpoint Descriptor bEndpointAddress bit 6:4 must be set to zero (000b).</td>
<td></td>
<td>WUSBCV</td>
</tr>
<tr>
<td>8.4.3.15#4</td>
<td>A DWA Isochronous Streaming OUT Endpoint Descriptor bmAttributes field must be set to 01H.</td>
<td></td>
<td>TD.1.10</td>
</tr>
<tr>
<td>8.4.3.15#5</td>
<td>A DWA Isochronous Streaming OUT Endpoint Descriptor wMaxPacketSize field must contain the same value as the DWA Isochronous Streaming OUT Endpoint Companion Descriptor wOverTheAirPacketSize field value.</td>
<td></td>
<td>TD.1.10</td>
</tr>
<tr>
<td>8.4.3.15#6</td>
<td>A DWA Isochronous Streaming OUT Endpoint Descriptor wMaxPacketSize field must be less or equal to 3584.</td>
<td></td>
<td>WUSBCV</td>
</tr>
<tr>
<td>8.4.3.15#7</td>
<td>A DWA Isochronous Streaming OUT Endpoint Descriptor bInterval field must be set to zero.</td>
<td></td>
<td>TD.1.10</td>
</tr>
</tbody>
</table>

**Subsection reference: 8.4.3.16 Isochronous Streaming OUT Endpoint Companion Descriptor**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action: Fail/Warning/NoTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.3.16#1</td>
<td>A DWA Isochronous Streaming OUT Endpoint Companion Descriptor bLength field must be set to 0AH.</td>
<td></td>
<td>WUSBCV</td>
</tr>
<tr>
<td>8.4.3.16#2</td>
<td>A DWA Isochronous Streaming OUT Endpoint Companion Descriptor bDescriptorType field must be set to 17 decimal.</td>
<td></td>
<td>WUSBCV</td>
</tr>
<tr>
<td>8.4.3.16#3</td>
<td>A DWA Isochronous Streaming OUT Endpoint Companion Descriptor bMaxBurst field must contain a value in the range 1 to 16.</td>
<td></td>
<td>WUSBCV</td>
</tr>
<tr>
<td>8.4.3.16#4</td>
<td>A DWA Isochronous Streaming OUT Endpoint Companion Descriptor bMaxSequence field must contain a value in the range 2 to 32.</td>
<td></td>
<td>WUSBCV</td>
</tr>
<tr>
<td>8.4.3.16#5</td>
<td>A DWA Isochronous Streaming OUT Endpoint Companion Descriptor bMaxStreamDelay field must contain a value in the range 1 to 65535.</td>
<td></td>
<td>WUSBCV</td>
</tr>
<tr>
<td>8.4.3.16#6</td>
<td>A DWA Isochronous Streaming OUT Endpoint Companion Descriptor wOverTheAirInterval field must contain a value in the range 4 to 255.</td>
<td></td>
<td>WUSBCV</td>
</tr>
<tr>
<td>8.4.3.16#7</td>
<td>A DWA Isochronous Streaming OUT Endpoint Companion Descriptor bmCompAttributes field must be set to 02H.</td>
<td></td>
<td>WUSBCV</td>
</tr>
</tbody>
</table>

**Subsection reference: 8.4.3.17 Isochronous Streaming IN Endpoint Descriptor**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action: Fail/Warning/NoTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.3.17#1</td>
<td>A DWA Isochronous Streaming IN Endpoint Descriptor bLength field must be set to 7 decimal.</td>
<td></td>
<td>WUSBCV</td>
</tr>
<tr>
<td>8.4.3.17#2</td>
<td>A DWA Isochronous Streaming IN Endpoint Descriptor bDescriptorType field must be set to 5 decimal.</td>
<td></td>
<td>WUSBCV</td>
</tr>
<tr>
<td>8.4.3.17#3</td>
<td>A DWA Isochronous Streaming IN Endpoint Descriptor bEndpointAddress bit 6:4 must be set to zero (000b).</td>
<td></td>
<td>WUSBCV</td>
</tr>
<tr>
<td>8.4.3.17#4</td>
<td>A DWA Isochronous Streaming IN Endpoint Descriptor bmAttributes field must be set to 01H.</td>
<td></td>
<td>TD.1.11</td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
<td>Action:</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------</td>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>8.4.3.17#5</td>
<td>A DWA Isochronous Streaming IN Endpoint Descriptor wMaxPacketSize field must contain the same value as the DWA Isochronous Streaming IN Endpoint Companion Descriptor wOverTheAirPacketSize field value.</td>
<td>TD.1.11</td>
<td></td>
</tr>
<tr>
<td>8.4.3.17#6</td>
<td>A DWA Isochronous Streaming IN Endpoint Descriptor wMaxPacketSize field must be less or equal to 3584.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.17#7</td>
<td>A DWA Isochronous Streaming IN Endpoint Descriptor bInterval field must be set to zero.</td>
<td>TD.1.11</td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.4.3.18 Isochronous Streaming IN Endpoint Companion Descriptor**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.3.18#1</td>
<td>A DWA Isochronous Streaming IN Endpoint Companion Descriptor bLength field must be set to 0AH.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.18#2</td>
<td>A DWA Isochronous Streaming IN Endpoint Companion Descriptor bDescriptorType field must be set to 17 decimal.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.18#3</td>
<td>A DWA Isochronous Streaming IN Endpoint Companion Descriptor bMaxBurst field must contain a value in the range 1 to 16.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.18#4</td>
<td>A DWA Isochronous Streaming IN Endpoint Companion Descriptor bMaxSequence field must contain a value in the range 2 to 32.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.18#5</td>
<td>A DWA Isochronous Streaming IN Endpoint Companion Descriptor bMaxStreamDelay field must contain a value in the range 1 to 65535.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.18#6</td>
<td>A DWA Isochronous Streaming IN Endpoint Companion Descriptor wOverTheAirInterval field must contain a value in the range 4 to 255.</td>
<td>WUSBCV</td>
<td></td>
</tr>
<tr>
<td>8.4.3.18#7</td>
<td>A DWA Isochronous Streaming IN Endpoint Companion Descriptor bmCompAttributes field must be set to 02H.</td>
<td>WUSBCV</td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.4.3.19 Wire Adapter RPipe Descriptor**

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.3.19#1</td>
<td>A DWA must not return the Wire Adapter RPipe descriptors as part of the configuration descriptor for the DWA.</td>
<td>TD.1.3</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#2</td>
<td>A DWA must return each RPipe descriptor upon receipt of a Get RPipe Descriptor for that RPipe from the host.</td>
<td>TD.1.12</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#3</td>
<td>A DWA Wire Adapter RPipe Descriptor bLength field must be set to 1CH.</td>
<td>TD.1.12</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#4</td>
<td>A DWA Wire Adapter RPipe Descriptor bDescriptorType field must be set to 22H.</td>
<td>TD.1.12</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#5</td>
<td>A DWA Wire Adapter RPipe Descriptor wRPipeIndex must never be greater than the Wire Adapter Class Descriptor wNumRPIPipes-1.</td>
<td>TD.1.12</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#6</td>
<td>A DWA must respond with the RPipe Descriptor that belongs to the RPipe with index wRPipeIndex to a GetRPipeDescriptor (RPipe index) request.</td>
<td>TD.1.12</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#7</td>
<td>A DWA RPipe must support the number of concurrent requests on the RPipe as specified in the DWA Wire Adapter RPipe Descriptor wRequests field.</td>
<td>TD.1.21 TD.1.22</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#8</td>
<td>A DWA Wire Adapter RPipe Descriptor wBlocks field that is set to zero</td>
<td>TD.1.20 TD.1.21</td>
<td></td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
<td>Action: Fail/ Warning/ NoTest</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------</td>
<td>--------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td>must be able to be modified by host software by means of a Set RPipe Descriptor request.</td>
<td>TD.1.22</td>
<td>TD.1.23</td>
</tr>
<tr>
<td>8.4.3.19#9</td>
<td>A DWA Wire Adapter RPipe Descriptor wBlocks field that is not set to zero must not be able to be modified by host software by means of a Set RPipe Descriptor request.</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#10</td>
<td>A DWA RPipe must provide buffering for wBlocks * bRPipeBlockSize</td>
<td>TD.1.20</td>
<td>TD.1.21 TD.1.22 TD.1.23</td>
</tr>
<tr>
<td>8.4.3.19#11</td>
<td>A DWA Wire Adapter RPipe Descriptor wMaxPacketSize is the maximum packet size that this RPipe will use to communicate with the attached device.</td>
<td>TD.1.20</td>
<td>TD.1.21 TD.1.22 TD.1.23</td>
</tr>
<tr>
<td>8.4.3.19#12</td>
<td>A DWA RPipe must be able to communicate to a full/low speed device endpoint via the attached device’s parent high speed hub when the host software sets the DWA Wire Adapter RPipe Descriptor bHSHubAddress field to the attached device’s parent high speed hub address.</td>
<td>TD.1.24</td>
<td>TD.1.25 TD.1.26 TD.1.27</td>
</tr>
<tr>
<td>8.4.3.19#13</td>
<td>A DWA RPipe must be able to communicate to a full/low speed device endpoint that is directly connected to the DWA when the host software sets the DWA Wire Adapter RPipe Descriptor bHSHubAddress field to zero.</td>
<td>TD.1.20</td>
<td>TD.1.21 TD.1.22 TD.1.23</td>
</tr>
<tr>
<td>8.4.3.19#14</td>
<td>A DWA RPipe must be able to communicate to a full/low speed device endpoint that is connected to the DWA through a high speed hub when the host software sets the DWA Wire Adapter RPipe Descriptor bHSHubPort field to the port of the parent high speed hub of the attached device.</td>
<td>TD.1.24</td>
<td>TD.1.25 TD.1.26 TD.1.27</td>
</tr>
<tr>
<td>8.4.3.19#15</td>
<td>A DWA RPipe must be able to communicate to a full/low speed device endpoint that is directly connected to the DWA when the host software sets the DWA Wire Adapter RPipe Descriptor bHSHubPort field to zero.</td>
<td>TD.1.20</td>
<td>TD.1.21 TD.1.22 TD.1.23</td>
</tr>
<tr>
<td>8.4.3.19#16</td>
<td>A DWA RPipe must use full speed transactions to a device endpoint directly connected to the DWA when the host software sets the DWA Wire Adapter RPipe Descriptor bSpeed field to 00B.</td>
<td>TD.1.20</td>
<td>TD.1.21 TD.1.22 TD.1.23</td>
</tr>
<tr>
<td>8.4.3.19#17</td>
<td>A DWA RPipe must use split transactions to a device endpoint connected through aHS hub to the DWA when the host software sets the DWA Wire Adapter RPipe Descriptor bSpeed field to 00B.</td>
<td>TD.1.24</td>
<td>TD.1.25 TD.1.26 TD.1.27</td>
</tr>
<tr>
<td>8.4.3.19#18</td>
<td>A DWA RPipe must use low speed transactions to a device endpoint directly connected to the DWA when the host software sets the DWA Wire Adapter RPipe Descriptor bSpeed field to 01B.</td>
<td>TD.1.20</td>
<td>TD.1.22</td>
</tr>
<tr>
<td>8.4.3.19#19</td>
<td>A DWA RPipe must use split transactions to a device endpoint connected through aHS hub to the DWA when the host software sets the DWA WireAdapter RPipe Descriptor bSpeed field to 01B.</td>
<td>TD.1.24</td>
<td>TD.1.26 TD.1.27</td>
</tr>
<tr>
<td>8.4.3.19#20</td>
<td>A DWA RPipe must use high speed transactions when the host software sets the DWA Wire Adapter RPipe Descriptor bSpeed field to 10B.</td>
<td>TD.1.20</td>
<td>TD.1.21 TD.1.22</td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
<td>Action: Fail/Warning/NoTest</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------</td>
<td>--------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>8.4.3.19#21</td>
<td>A DWA RPipe must use the Wire Adapter RPipe Descriptor bDeviceAddress field set by the host software as the device address of the target device endpoint.</td>
<td>TD.1.23, TD.1.24, TD.1.25, TD.1.26, TD.1.27</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#22</td>
<td>A DWA Wire Adapter RPipe Descriptor bDeviceAddress field must be set to an address value of 0000H to 007FH (0 – 127 decimal).</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#23</td>
<td>A DWA RPipe must use the Wire Adapter RPipe Descriptor bEndpointAddress field bit 3:0 set by the host software as the target device endpoint number.</td>
<td>TD.1.20, TD.1.21, TD.1.22, TD.1.23, TD.1.24, TD.1.25, TD.1.26, TD.1.27, TD.1.28</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#24</td>
<td>A DWA Wire Adapter RPipe Descriptor bEndpointAddress field bit 6:4 is reserved and must be set to zero (000b).</td>
<td>TD.1.12</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#25</td>
<td>A DWA RPipe must use OUT transfers to the targeted endpoint if the host software has set the DWA Wire Adapter RPipe Descriptor bEndpointAddress field bit 7 to 0.</td>
<td>TD.1.20, TD.1.21, TD.1.22, TD.1.23</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#26</td>
<td>A DWA RPipe must use IN transfers to the targeted endpoint if the host software has set the DWA Wire Adapter RPipe Descriptor bEndpointAddress field bit 7 to 1.</td>
<td>TD.1.20, TD.1.21, TD.1.22, TD.1.23</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#27</td>
<td>A DWA must ignore the Wire Adapter RPipe Descriptor bEndpointAddress field bit 7 when the target endpoint for the RPipe is a control endpoint.</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#28</td>
<td>A DWA must use the Wire Adapter RPipe Descriptor bDataSequence field set by the host software as the next data sequence value to be used when sending data to the targeted endpoint.</td>
<td>TD.1.20, TD.1.21, TD.1.22, TD.1.23</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#29</td>
<td>A DWA Wire Adapter RPipe Descriptor bwCurrentWindow field is reserved and must be set to zero.</td>
<td>TD.1.12</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#30</td>
<td>A DWA Wire Adapter RPipe Descriptor bMaxDataSequence field is reserved and must be set to zero.</td>
<td>TD.1.12</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#31</td>
<td>A DWA must use DWA Wire Adapter RPipe Descriptor bInterval field value as the polling interval for downstream communications.</td>
<td>TD.1.22, TD.1.23, TD.1.30</td>
<td></td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
<td>Action: Fail/Warning/NoTest</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>8.4.3.19#32</td>
<td><strong>A DWA must send/receive isochronous packets on its upstream isochronous endpoint at the polling interval as set by the host software in the DWA Wire Adapter RPipe Descriptor bOverTheAirInterval field for the RPipe that is associated to the upstream isochronous endpoint.</strong></td>
<td>TD.1.31</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TD.1.32</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TD.1.33</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#33</td>
<td><strong>A DWA RPipe must perform Control transfers to the targeted endpoint if the host software sets the DWA Wire Adapter RPipe Descriptor bmAttribute field bit 1:0 to 00b.</strong></td>
<td>TD.1.20</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#34</td>
<td><strong>A DWA RPipe must perform Isochronous transfers to the targeted endpoint if the host software sets the DWA Wire Adapter RPipe Descriptor bmAttribute field bit 1:0 to 01b.</strong></td>
<td>TD.1.23</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#35</td>
<td><strong>A DWA RPipe must perform Bulk transfers to the targeted endpoint if the host software sets the DWA Wire Adapter RPipe Descriptor bmAttribute field bit 1:0 to 10b.</strong></td>
<td>TD.1.21</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#36</td>
<td><strong>A DWA RPipe must perform Interrupt transfers to the targeted endpoint if the host software sets the DWA Wire Adapter RPipe Descriptor bmAttribute field bit 1:0 to 11b.</strong></td>
<td>TD.1.22</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#37</td>
<td><strong>A DWA must send/receive isochronous data to the RPipe via the upstream Isochronous Endpoint specified in the DWA Wire Adapter RPipe Descriptor bmAttribute field bit 5:2.</strong></td>
<td>TD.1.23</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TD.1.52</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#38</td>
<td><strong>A DWA Wire Adapter RPipe Descriptor bmAttribute field bit 7:6 is reserved and must be set to zero.</strong></td>
<td>TD.1.12</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#39</td>
<td><strong>A DWA Wire Adapter RPipe Descriptor bmCharacteristics field bit 0 must be set to one if Control Transfers are supported on the RPipe.</strong></td>
<td>TD.1.20</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#40</td>
<td><strong>A DWA Wire Adapter RPipe Descriptor bmCharacteristics field bit 0 must be set to zero if Control Transfers are not supported on the RPipe.</strong></td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#41</td>
<td><strong>A DWA Wire Adapter RPipe Descriptor bmCharacteristics field bit 1 must be set to one if Isochronous Transfers are supported on the RPipe.</strong></td>
<td>TD.1.23</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#42</td>
<td><strong>A DWA Wire Adapter RPipe Descriptor bmCharacteristics field bit 1 must be set to zero if Isochronous Transfers are not supported on the RPipe.</strong></td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#43</td>
<td><strong>A DWA Wire Adapter RPipe Descriptor bmCharacteristics field bit 2 must be set to one if Bulk Transfers are supported on the RPipe.</strong></td>
<td>TD.1.21</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#44</td>
<td><strong>A DWA Wire Adapter RPipe Descriptor bmCharacteristics field bit 2 must be set to zero if Bulk Transfers are not supported on the RPipe.</strong></td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#45</td>
<td><strong>A DWA Wire Adapter RPipe Descriptor bmCharacteristics field bit 3 must be set to one if Interrupt Transfers are supported on the RPipe.</strong></td>
<td>TD.1.22</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#46</td>
<td><strong>A DWA Wire Adapter RPipe Descriptor bmCharacteristics field bit 3 must be set to zero if Interrupt Transfers are not supported on the RPipe.</strong></td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#47</td>
<td><strong>A DWA Wire Adapter RPipe Descriptor bmCharacteristics field bit 7:4 is</strong></td>
<td>TD.1.12</td>
<td></td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
<td>Action: Fail/Warning/NoTest</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------</td>
<td>--------</td>
<td>----------------------------</td>
</tr>
<tr>
<td></td>
<td>reserved and must be set to zero.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#48</td>
<td>A DWA must retry a transaction the maximum number of times as set in DWA Wire Adapter RPipe Descriptor bmRetryOptions field bit 2:0, before failing the transfer request.</td>
<td>TD.1.44</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#49</td>
<td>A DWA must not have a maximum number for retrying a transaction if the number reported in the DWA Wire Adapter RPipe Descriptor bmRetryOptions field bit 2:0 is zero.</td>
<td>TD.1.45</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#50</td>
<td>A DWA Wire Adapter RPipe Descriptor bmRetryOptions field bit 7:3 is reserved and must set to zero.</td>
<td>TD.1.12</td>
<td></td>
</tr>
<tr>
<td>8.4.3.19#51</td>
<td>A DWA must increment the DWA Wire Adapter RPipe Descriptor wNumTransactionErrors field when it encounters an error while performing transactions to the downstream endpoint targeted by the RPipe.</td>
<td>TD.1.45</td>
<td></td>
</tr>
</tbody>
</table>

Subsection reference: 8.4.4 DWA Specific Requests

Subsection reference: 8.4.4.1 Clear Port Feature

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action: Fail/Warning/NoTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.4.1#1</td>
<td>Assertions for Clear Port Feature (ref USBCV Test Assertions 2.2 Chapter 11)</td>
<td>USBCV Hub Tests</td>
<td></td>
</tr>
<tr>
<td>8.4.4.1#2</td>
<td>A DWA must respond with a Request Error to a Clear_Port_Feature(wValue, wIndex, 0) if wValue is not a valid feature selector, if wIndex specifies a port that does not exist or if wLength is not zero.</td>
<td>USBCV Hub Tests</td>
<td></td>
</tr>
</tbody>
</table>

Subsection reference: 8.4.4.2 Get Port Status

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action: Fail/Warning/NoTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.4.2#1</td>
<td>Assertions for Get Port Status (format and settings of wPortStatus and wPortChange replies from DWA) (ref USBCV Test Assertions 2.2 Chapter 11)</td>
<td>TD.1.1 TD.1.47 &amp; USBCV Hub Tests</td>
<td></td>
</tr>
<tr>
<td>8.4.4.2#2</td>
<td>A DWA must respond with a Request Error to a Get_Port_Status(wValue, wIndex, wLength) request if wValue is not 0, if wIndex is not a valid Port Index or if wLength is not 4.</td>
<td>USBCV Hub Tests</td>
<td></td>
</tr>
</tbody>
</table>

Subsection reference: 8.4.4.3 Set Isochronous Endpoint Attributes

<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action: Fail/Warning/NoTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.4.3#1</td>
<td>A DWA endpoint must send or receive data as per wOverTheAirPacketSize upon reception of a valid Set_Isochronous_Endpoint_Attribute request.</td>
<td>TD.1.29</td>
<td></td>
</tr>
<tr>
<td>8.4.4.3#2</td>
<td>A DWA endpoint must provide buffering to support wMaxStreamDelay upon reception of a valid Set_Isochronous_Endpoint_Attribute request.</td>
<td>TD.1.29</td>
<td></td>
</tr>
<tr>
<td>8.4.4.3#3</td>
<td>A DWA must respond with a Request Error to a Set_Isochronous_Endpoint_Attribute (wValue, wIndex, wLength) request if wValue is not 0, if wIndex is not an existing Endpoint Address or if N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assertion #</td>
<td>Assertion Description</td>
<td>Test #</td>
<td>Action: Fail/Warning/NoTest</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------</td>
<td>--------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td>wLength is not 6.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.4.4.4 Set Port Feature**

<table>
<thead>
<tr>
<th>8.4.4.4#1</th>
<th>Assertions for Set Port Feature (ref USBCV Test Assertions 2.2 Chapter 11)</th>
<th>USBCV Hub Tests</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.4.4#2</td>
<td>A DWA must respond with a Request Error to a Set_Port_Feature(wValue, wIndex, wLength) request if wValue is not a valid feature selector, if wIndex is not a valid Port Index or if wLength is not 0.</td>
<td>USBCV Hub Tests</td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.4.5 DWA Notification Information**

**Subsection reference: 8.4.5.1 Remote Wake**

<table>
<thead>
<tr>
<th>8.4.5.1#1</th>
<th>A DWA must send a Remote Wake notification to the host when a DWA detects a remote wake from any of its downstream connected devices and it is armed for remote wake.</th>
<th>USBCV Hub Tests</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.5.1#2</td>
<td>A DWA Remote Wake Notification bLength field must be set to 2 decimal.</td>
<td>USBCV Hub Tests</td>
<td></td>
</tr>
<tr>
<td>8.4.5.1#3</td>
<td>A DWA Remote Wake Notification bNotifyType field must be set to 91H.</td>
<td>USBCV Hub Tests</td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.4.5.2 Port Status Change**

<table>
<thead>
<tr>
<th>8.4.5.2#1</th>
<th>A DWA must send a Port Status Change notification to the host when the status on a downstream port of the DWA changes.</th>
<th>TD.1.1 TD.1.47 &amp; USBCV Hub Tests</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.5.2#2</td>
<td>A DWA Port Status Change Notification bLength field must be set to 3 decimal.</td>
<td>TD.1.1 TD.1.47 &amp; USBCV Hub Tests</td>
<td></td>
</tr>
<tr>
<td>8.4.5.2#3</td>
<td>A DWA Port Status Change Notification bNotifyType field must be set to 92H.</td>
<td>TD.1.1 TD.1.47 &amp; USBCV Hub Tests</td>
<td></td>
</tr>
<tr>
<td>8.4.5.2#4</td>
<td>A DWA Port Status Change Notification bPortIndex field must be set the Index of the Port on which a change has occurred.</td>
<td>TD.1.1 TD.1.47 &amp; USBCV Hub Tests</td>
<td></td>
</tr>
</tbody>
</table>

**Subsection reference: 8.4.6 DWA Isochronous Transfers**

**Subsection reference: 8.4.6.1 DWA Isochronous OUT Responsibilities**

<p>| 8.4.6.1#1  | A DWA must transmit the isochronous data OUT segments over the wire when the presentation time equals the downstream frame value. | TD.1.51 |                             |</p>
<table>
<thead>
<tr>
<th>Assertion #</th>
<th>Assertion Description</th>
<th>Test #</th>
<th>Action:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.6.2#1</td>
<td>A DWA must perform isochronous IN requests to the downstream wired endpoint every bInterval (micro)frames. (with bInterval as specified in the RPipe descriptor)</td>
<td>TD.1.32</td>
<td>Fail/ Warning/ NoTest</td>
</tr>
<tr>
<td>8.4.6.2#2</td>
<td>A DWA must not begin generating IN Tokens to the downstream device until the host begins polling for data on the associated DWA isochronous data stream function endpoint.</td>
<td>TD.1.48</td>
<td></td>
</tr>
<tr>
<td>8.4.6.2#3</td>
<td>A DWA must aggregate wired isochronous IN packet data into the largest packets that can be sent over the air by the associated Wireless USB Isochronous endpoint.</td>
<td>TD.1.23</td>
<td></td>
</tr>
<tr>
<td>8.4.6.2#4</td>
<td>A DWA must not split wired isochronous IN packet data from a single (micro)frame across multiple over-the-air packets.</td>
<td>TD.1.48</td>
<td></td>
</tr>
<tr>
<td>8.4.6.2#5</td>
<td>A DWA must respond with the oldest data in its Rpipe buffer for each Wireless USB Isochronous IN request.</td>
<td>TD.1.50</td>
<td></td>
</tr>
<tr>
<td>8.4.6.2#6</td>
<td>A DWA must only discard data if the buffer associated with the Rpipe overflows.</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.4.6.2#7</td>
<td>The DWA must discard the oldest data first.</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>8.4.6.2#8</td>
<td>A DWA’s SOF values transmitted downstream must match the Wireless USB channel times on the bus between the DWA and its host.</td>
<td>TD.1.51</td>
<td></td>
</tr>
<tr>
<td>8.4.6.2#9</td>
<td>A DWA must respond with a NAK handshake to a W_DVTCTA transaction token from the host asking for Isochronous IN data on the DWA upstream Isochronous Endpoint, when the DWA does not have any data to respond to this transaction token.</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

3. Test Descriptions

**Test Description Fail Conditions**

The test descriptions provide a summary of fail conditions for the test. These lists are not always complete.

**DWA States**

For all tests described the DWA under test has been successfully enumerated and configured by the test software stack with support for Wireless USB. The procedures used to enumerate the DWA under test and place it in a desired starting state are documented below for reference.

Un-Authenticated
1. Put the DWA in the un-authenticated state following the procedure below.
2. Test software configures the host to accept only connections (not new or reconnects).
3. Send a ResetDevice_IE with matching CDID element value in 6 MMC’s.
4. Test software waits for device under test to connect.

**Default State from Unauthenticated:**
1. Put the DWA in the un-authenticated state following the Un-authenticated procedure.
2. Complete 4-way handshake
3. Send a SetKey(GTK) request to the DWA
4. Issue a valid Set Address command to the DWA with address zero.

**Default State from Authenticated:**
Issue a valid Set Address command to the DWA with address zero.

**Address State from Unauthenticated:**
1. Put the DWA in the default state following the default from Un-authenticated procedure.
2. Issue a valid Set Address command to the DWA with a valid non-zero address.

**Address State from Authenticated:**
1. Issue a valid Set Address command to the DWA with address zero.
2. Issue a valid Set Address command to the DWA with a valid non-zero address.

**Configured State from Un-Authenticated:**
The DWA is enumerated using the following procedure.
1. Put the DWA in the address state following the Address state from Un-authenticated procedure.
2. Issue a valid set configuration command for the configuration to be tested.
3. Issue a valid get configuration command and verify that the correct configuration is returned.

**Configured State from Authenticated:**
The DWA is enumerated using the following procedure.
1. Issue a valid Set Address command to the DWA with address zero.
2. Issue a valid Set Address command to the DWA with a non-zero address.
3. Issue a valid set configuration command for the configuration to be tested.
   Issue a valid get configuration command and verify that the correct configuration is returned.

---

**DWA Specific Configuration Routine**
1. Use standard WUSBCV device enumeration routine.
2. Get the DWA Wire Adapter class descriptor.
3. Get the RPipe Descriptors for all RPIPES
4. Initialize the DWA host controller:
   - Send a Set Wire Adapter Feature Request (WIRE_ADAPTER_RESET) to the DWA
   - Send a Set Wire Adapter Feature Request (WIRE_ADAPTER_ENABLE) to the DWA
   - Send a Set Port Feature Request with PORT_POWER feature selector for each port of the DWA to the DWA.
   - Sleep for 1 second
   - Send a Get Port Status Request for each port of the DWA and check if the connect bit is set.
   - If the connect bit is not set then take no further actions for that port
   - If the connect bit is set, perform DWA downstream device enumeration routinesend a Set Port Feature Request with PORT_RESET to the port
   - Sleep for 50 ms
   - Send a Get Port Status Request for each port of the DWA and check if the enable bit is set.

---

**RPipe Initialization routine**
This routine explains the procedure used to set up an RPipe to an endpoint of the attached test device. Among other things, this section contains default values used for the set RPipe descriptor data fields. It will be indicated in the test case if specific values are used, other than the default.
The host chooses an RPipe which `bmCharacteristics` field indicates that it supports the intended transfer type. Repeat steps multiple times for mapping multiple test device endpoints.

1. Choose an RPipe in Idle state.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bLength</code></td>
<td>1CH</td>
<td>Read only field, value determined by get RPipe descriptor and static for all DWA’s.</td>
</tr>
<tr>
<td><code>bDescriptorType</code></td>
<td>22H</td>
<td>Read only field, value determined by get RPipe descriptor and static for all DWA’s.</td>
</tr>
<tr>
<td><code>wRPipeIndex</code></td>
<td></td>
<td>Index of the RPipe</td>
</tr>
<tr>
<td><code>wRequests</code></td>
<td></td>
<td>The number of concurrent requests that can be assigned to the RPipe.</td>
</tr>
</tbody>
</table>
| `wBlocks`      |       | 1. Value set by host software (refer to test case for value) 
2. Or value repeated by host software if the wBlocks value is statically defined for the DWA. |
| `wMaxPacketSize` |       | Maxpacketsize of the attached test device by default. Unless otherwise specified in the test case:
1. For an RPipe to endpoint zero of the test device: Set to 64 decimal by default and for standard control requests. Resized to a smaller value for downstream devices with a smaller maximum packet size.
2. For an RPipe to another endpoint of the test device: Set to the maxpacketsize of the Endpoint Descriptor of the attached test device by default. |
| `bHSHubAddress` |       | Zero or High Speed Hub address 
1. Set to zero if: - Full/low speed device is directly connected to the DWA. 
- `bSpeed` field is set to High Speed. 
2. Set to the address of the attached device’s parent high speed hub if: A full/low device is connected to the DWA through a high speed hub. |
| `bHSHubPort`   |       | Zero or Port number 
1. Set to zero if: - Full/low speed device is directly connected to the DWA. 
- `bSpeed` field is set to High Speed. 
2. Set to the Port number on which the attached device is connected to its parent high speed hub if: A full/low device is connected to the DWA through a high speed hub. |
| `bSpeed`       |       | Speed-value of the attached test device 
The speed of the test device to be targeted by RPipe. | | Value | Description |
|                | 00B   | Full-Speed (12Mbs) |
|                | 01B   | Low-Speed (1.5Mbs) |
|                | 10B   | High-Speed (480 Mbs) |
| `bDeviceAddress` |       | Address of the test device 
Address to be used with attached test device |
| `bEndpointAddress` |       | Endpoint Address of the attached test device 
Endpoint Address to be used with this RPipe. | | Bit | Description |
|                 | 3:0   | The endpoint number |
|                 | 6:4   | Reserved; set to zero |
|                 | 7     | Direction, ignored for control endpoints 
0 = OUT endpoint 
1 = IN endpoint |
| `bDataSequence` |       | Number 
Current data sequence. This is the next data sequence value to be used when sending data to the endpoint that this RPipe is targeted at. 
Set to zero for standard control requests. |
| `dwCurrentWindow` |       | Zero 
Reserved and must be set to zero. |
| `bMaxDataSequence` |       | Zero 
Reserved and must be set to zero. |
| `bInterval`    |       | binterval of the attached test device 
Unless otherwise noted in the test case, binterval (polling |
The interval to be used by this RPipe in downstream communications is set to the bInterval value of the Endpoint Descriptor of the attached test device.

**bOverTheAirInterval**
- Zero or polling interval for upstream wireless isochronous endpoint.
  - 1. Set to polling interval for upstream wireless isochronous endpoint if: The transfer type is Isochronous.
  - 2. Set to zero if: For all other transfer types.

**bmAttribute**
- Bitmap as described

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:0</td>
<td>Value</td>
</tr>
<tr>
<td>00B</td>
<td>Control</td>
</tr>
<tr>
<td>01B</td>
<td>Isochronous</td>
</tr>
<tr>
<td>10B</td>
<td>Bulk</td>
</tr>
<tr>
<td>11B</td>
<td>Interrupt</td>
</tr>
</tbody>
</table>

The host sets one of the above values, depending on the targeted endpoint transfer type.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5:2</td>
<td>The host sets this field to the associated upstream Isochronous endpoint on the DWA, if the transfer type is Isochronous. The host sets this field to zero for all other transfer types.</td>
</tr>
<tr>
<td>7:6</td>
<td>Set to zero</td>
</tr>
</tbody>
</table>

**bmCharacteristics**
- Bitmap as described (as returned by get RPipe descriptor)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1: Control Transfer supported 0: Control Transfer NOT supported</td>
</tr>
<tr>
<td>1</td>
<td>1: Isochronous Transfer supported 0: Isochronous Transfer NOT supported</td>
</tr>
<tr>
<td>2</td>
<td>1: Bulk Transfer supported 0: Bulk Transfer NOT supported</td>
</tr>
<tr>
<td>3</td>
<td>1: Interrupt Transfer supported 0: Interrupt Transfer NOT supported</td>
</tr>
<tr>
<td>7:4</td>
<td>Zero</td>
</tr>
</tbody>
</table>

Transfer types supported on this RPipe

Read only field, value determined by get RPipe descriptor.

**bmRetryOptions**
- Set to 1 by default.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2:0</td>
<td>The maximum number of times a transaction must be retried before the transfer request is failed. The valid values are 0 through 3. A value of zero in this field indicates that the DWA must not count errors and there is no limit on the retries. For Isochronous transfers this field is set to zero. Set to 2 for standard control requests.</td>
</tr>
<tr>
<td>7:3</td>
<td>Zero.</td>
</tr>
</tbody>
</table>

**wNumTransactionErrors**
- Zero.

The host is responsible for resetting this field. (The DWA increments this field when it encounters an error while performing transactions to the downstream endpoint targeted by this RPipe.)
Device Enumeration test

TD.1.1 Device Enumeration on DWA

This test verifies that low, full and high speed USB devices enumerate correctly under each user accessible downstream port of the Device Wire Adapter.

Device States for Test

This test is run with the DWA in Configured state.

Channels for Test

Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps

The test software performs the following steps.

- Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
- Place the DWA in the desired starting state.
- Test software prompts the user to connect a low speed test device to port 1 of the DWA.
- Verify if the DWA sends back a Port Status Change Notification when its notification endpoint is polled.
  - bLength field must be set to 3 decimal.
  - bNotifyType field must be set to 92H.
  - bPortIndex field must be set the Index of the Port on which the test device is attached.
- The host issues a Get Port Status command to the DWA and verifies that an attached device is reported.
- Verify if Bit 9 and 10 accurately indicate the speed of the attached device.
- The host enumerates the test device connected behind the DWA. (Refer to DWA downstream device enumeration routine)

Repetitions

Repeat this test for all RPipes that support Control Transfers,
And repeat for all test device speeds
And repeat for all downstream ports.

Test fails if

- DWA does not send back a Port Status Change Notification.
- Port Status Change Notification indicates the wrong port index.
- Standard enumeration procedure encounters any errors.
- Bit 9 and 10 of the Get Port Status command do not reflect the correct speed of the test device.
**Descriptor tests**

**TD.1.2 DWA Device descriptor test**
This test verifies the fields that come from the device are formatted in compliance with the specification and have appropriate values. Some fields of this descriptor are already verified in WUSBCV and not repeated in this test. This descriptor test is complementary to the descriptor test in WUSBCV.

**Device States for Test**
This test is run with the DWA in UnAuthenticated, Default, Address, and Configured state.

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.

- Place the DWA in the desired starting state.
- Send `Get_Descriptor(wValue, wIndex, wLength, Data)` request with the following values:
  - `wValue` – set to One (Device Descriptor).
  - `wIndex` – set to Zero.
  - `wLength` – 18 decimal
- Check the returned descriptor for the following values:
  - All:
    - `bNumConfigurations` must be 1.
    - If the DWA exports an Isochronous Interface:
      - `bDeviceClass` must be EFH
      - `bDeviceSubClass` must be 02H.
      - `bDeviceProtocol` must be 02H.
    - If the DWA does not export an Isochronous Interface:
      - `bDeviceClass` must be 0x0 and `bDeviceSubClass` must be 0x0 and `bDeviceProtocol` must be 0x0
      - or:
      - `bDeviceClass` must be E0H and `bDeviceSubClass` must be 02H and `bDeviceProtocol` must be 02H
  - All:
- Repeat test with the device starting in the following states
  - UnAuthenticated
  - Default
  - Address
  - Configured

**Test fails if**
The descriptor cannot be read from the DWA.
Any of the fields differ from the description in Chapter 8 of the Wireless USB Specification.

**TD.1.3 DWA Configuration descriptor test**
This test verifies the fields that come from the device are formatted in compliance with the specification and have appropriate values. Some fields of this descriptor are already verified in WUSBCV and not repeated in this test. This descriptor test is complementary to the descriptor test in WUSBCV.

**Device States for Test**
This test is run with the DWA in UnAuthenticated, Default, Address, and Configured state.

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.

- Place the DWA in the desired starting state.
- Send `Get_Descriptor(wValue, wIndex, wLength, Data)` request with the following values:
  - `wValue` – set to 2 (Configuration descriptor).
  - `wIndex` – set to Zero.
  - `wLength` – 9
Test fails if the device returns bLength set to anything but 9.

Send Get Descriptors(wValue, wIndex, wLength, Data) request with the following values:
- wValue – set to 2 (Configuration).
- wIndex – set to Zero.
- wLength – wTotalLength (All of the Descriptors).

Verify that there is one and only one data transfer interface.
Verify a Notification Endpoint (Interrupt IN) and a Data Transfer Endpoint pair (paired Bulk IN/OUT) are provided among the Endpoint Descriptors.
Verify that there is zero or one isochronous streaming interface. If an Isochronous Streaming Interface Descriptor is provided, verify if one or more Isochronous Streaming Endpoint Descriptors are provided.
Verify that no Wire Adapter RPipe descriptors are returned as part of the configuration descriptor for the DWA.
Parse the data returned; only keeping the Configuration Descriptor.
Check the returned descriptor for the following values:
- bLength must be 9 decimal.
- bDescriptorType must be 2 decimal.
- wTotalLength field must be set to the total length of all descriptors in the configuration.
- bNumInterfaces field must be set to the number of interfaces included in the configuration.
- bConfigurationValue field must be set to the value to use to reference the configuration.
- iConfiguration field must be set to the index of the String Descriptor describing the configuration. (report the value)

Repetitions
Repeat test with the device starting in the following states
- UnAuthenticated
- Default
- Address
- Configured

Test fails if
The descriptor cannot be read from the DWA.
Any of the fields differ from the description in Chapter 8 of the Wireless USB Specification.

TD.1.4 DWA Interface Association descriptor test

This test verifies the fields that come from the device are formatted in compliance with the specification and have appropriate values.

Device States for Test
This test is run with the DWA in UnAuthenticated, Default, Address, and Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
Place the DWA in the desired starting state.
Get the full configuration descriptor.
Parse the data returned; only keeping the Interface Association descriptor.
Check that one and only one Interface Association descriptor with the following values is present when the DWA has an Isochronous Streaming Interface descriptor.
Check the returned descriptor for the following values:
- bLength must be 8 decimal
- bDescriptorType must be 0BH
- bFirstInterface must be set to zero
- bInterfaceCount must be set to 2 decimal
- bFunctionClass must be set to E0H
- bFunctionSubClass must be set to 02H
- bFunctionProtocol must be set to 02H

Check that the interface association descriptor with the above values is for the data transfer and isochronous streaming interfaces.

Repetitions
Repeat test with the device starting in the following states
- UnAuthenticated
TD.1.5 DWA Data Transfer Interface descriptor test

This test verifies the fields that come from the device are formatted in compliance with the specification and have appropriate values. Some fields of this descriptor are already verified in WUSBCV and not repeated in this test. This descriptor test is complementary to the descriptor test in WUSBCV.

Device States for Test
This test is run with the DWA in UnAuthenticated, Default, Address, and Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
Place the DWA in the desired starting state.
Get the full configuration descriptor.
Parse the data returned; only keeping the Data Transfer Interface and associated endpoint descriptors.
Check the returned descriptor for the following values:
- bLength must be 9 decimal
- bDescriptorType must be 4 decimal
- bInterfaceNumber must be zero
- bAlternateSetting must be zero
- bNumEndpoints must be 3 decimal
- bInterfaceClass must be E0H
- bInterfaceSubclass must be 02H
- bInterfaceProtocol must be 02H
Check that the interface includes two bulk and one interrupt endpoint.

Repetitions
Repeat test with the device starting in the following states
- UnAuthenticated
- Default
- Address
- Configured

Test fails if
The descriptor cannot be read from the DWA.
Any of the fields differ from the description in Chapter 8 of the Wireless USB Specification.

TD.1.6 DWA Wire Adapter Class descriptor test

This test verifies the fields that come from the device are formatted in compliance with the specification and have appropriate values.

Device States for Test
This test is run with the DWA in UnAuthenticated, Default, Address, and Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
Place the DWA in the desired starting state.
Parse the data returned; only keeping the Wire Adapter Class descriptor.
Get the complete configuration descriptor.

Test fails if
The descriptor cannot be read from the DWA.
Any of the fields differ from the description in Chapter 8 of the Wireless USB Specification.
Check the returned descriptor for the following values:
- \( bLength \) must accurately reflect the length of the descriptor.
- \( bDescriptorType \) must be 21H.
- \( bcdWAVersion \) must be 0100H.
- \( bNumPorts \) must be greater than zero.
- \( bNumPorts \) must not be a number greater than 127
- \( bmAttributes \) bit 0 must be zero (0b) if the DWA uses Ganged power switching. (question posed to operator \( \rightarrow \) verified against operator input)
- \( bmAttributes \) bit 0 must be one (1b) if the DWA uses Individual port power switching. (question posed to operator \( \rightarrow \) verified against operator input)
- \( bmAttributes \) bit 1 must be zero (0b) if the DWA uses Global Over-current Protection. (question posed to operator \( \rightarrow \) verified against operator input)
- \( bmAttributes \) bit 1 must be one (1b) if the DWA uses Individual port Over-current Protection. (question posed to operator \( \rightarrow \) verified against operator input)
- \( bmAttributes \) bit 2 must be zero (0b) if the DWA does not support Port Indicators on its downstream ports. (question posed to operator \( \rightarrow \) verified against operator input)
- \( bmAttributes \) bit 2 must be one (1b) if the DWA supports Port Indicators on its downstream ports. (question posed to operator \( \rightarrow \) verified against operator input)
- \( bmAttributes \) bit 6:3 must be zero (0000b).
- \( bmAttributes \) bit 7 must be zero (0b).
- \( wNumRPipes \) is verified in separate test and reported to user in this test
- \( wNumRPipes \) must be at least 2 x (DWA Wire Adapter Class Descriptor \( bNumPorts \) field value).
- \( wRPipeMaxBlock \) is verified in separate test and reported to user in this test
- \( bRPipeBlockSize \) is verified in separate test and reported to user in this test
- \( bRPipeBlockSize \) field must be greater than zero
- \( bPwrOn2PwrGood \) is not functionally tested, but is reported to user in this test
- \( bNumMMCIEs \) must be set to zero.
- \( DeviceRemovable \) field size in bytes must equal the integer value of \( \text{floor}(bNumPorts / 8) + 1 \)
- The test software prompts a port removable-question for the test operator to verify every bit of the DeviceRemovable field.
- \( bit \ n \) must be set to 1b if the device on the port \( n \) of the DWA is non-removable.
- \( bit \ n \) must be set to 0b if the device on the port \( n \) of the DWA is removable or if port \( n \) does not exist.
- \( DeviceRemovable \) field can not describe more than 127 ports.
- \( DeviceRemovable \) bit 0 is reserved and must be set to 0b.
- \( DeviceRemovable \) field cannot describe more ports than the \( bNumPorts \) field.

**Repetitions**
Repeat test with the device starting in the following states
- UnAuthenticated
- Default
- Address
- Configured

**Test fails if**
The descriptor cannot be read from the DWA.
Any of the fields differ from the description in Chapter 8 of the Wireless USB Specification.

**TD.1.7 DWA Notification Endpoint descriptor test**
This test verifies the fields that come from the device are formatted in compliance with the specification and have appropriate values. Some fields of this descriptor are already verified in WUSBCV and not repeated in this test. This descriptor test is complementary to the descriptor test in WUSBCV.

**Device States for Test**
This test is run with the DWA in UnAuthenticated, Default, Address, and Configured state

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.
Place the DWA in the desired starting state.
Get full configuration descriptor.
TD.1.8 DWA Notification Endpoint Companion descriptor test
This test verifies the fields that come from the device are formatted in compliance with the specification and have appropriate values. Some fields of this descriptor are already verified in WUSBCV and not repeated in this test. This descriptor test is complementary to the descriptor test in WUSBCV.

Device States for Test
This test is run with the DWA in UnAuthenticated, Default, Address, and Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
Place the DWA in the desired starting state.
Get full configuration descriptor test.
Parse the data returned; only keeping the Notification Endpoint Companion descriptor.
Check the returned descriptor for the following values:
- bMaxBurst must be one.
- bMaxStreamDelay must be 00H.
- wOverTheAirPacketSize must be 00H.
- wOverTheAirInterval must be 00H.
- bmCompAttributes must be 00H.

Repetitions
Repeat test with the device starting in the following states
- UnAuthenticated
- Default
- Address
- Configured

Test fails if
The descriptor cannot be read from the DWA.
Any of the fields differ from the description in Chapter 8 of the Wireless USB Specification.

TD.1.9 DWA Isochronous Streaming Interface descriptor test
This test verifies the fields that come from the device are formatted in compliance with the specification and have appropriate values. Some fields of this descriptor are already verified in WUSBCV and not repeated in this test. This descriptor test is complementary to the descriptor test in WUSBCV.

Device States for Test
This test is run with the DWA in UnAuthenticated, Default, Address, and Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
Place the DWA in the desired starting state.
Get the full configuration descriptor.
Parse the data returned; only keeping the Isochronous Streaming Interface descriptor.
Check the returned descriptor for the following values:
- bInterfaceNumber must be one.
- bAlternateSetting must be zero.
- bNumEndpoints must be in the range 1 to 27.
- bNumEndpoints endpoint descriptors (isochronous IN or Isochronous OUT must exist).
- bInterfaceClass must be E0H.
- bInterfaceSubClass must be 02H.
- bInterfaceProtocol must be 03H.
Check that all endpoints in the isochronous streaming interface are either isochronous IN or isochronous OUT endpoints.

Repetitions
Repeat test with the device starting in the following states
- UnAuthenticated
- Default
- Address
- Configured

Test fails if
The descriptor cannot be read from the DWA.
Any of the fields differ from the description in Chapter 8 of the Wireless USB Specification.

TD.1.10 DWA Isochronous Streaming OUT Endpoint descriptor test
This test verifies the fields that come from the device are formatted in compliance with the specification and have appropriate values. Some fields of this descriptor are already verified in WUSBCV and not repeated in this test. This descriptor test is complementary to the descriptor test in WUSBCV.

Device States for Test
This test is run with the DWA in UnAuthenticated, Default, Address, and Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
Place the DWA in the desired starting state.
Get the complete configuration descriptor.
Parse the data returned: only keeping the Isochronous Streaming OUT Endpoint descriptor.
Check the returned descriptor for the following values:
- bmAttributes field must be 01H.
- wMaxPacketSize field must contain the same value as the DWA Isochronous Streaming OUT Endpoint Companion Descriptor wOverTheAirPacketSize field value.
- bInterval field must be zero.

Repetitions
Repeat test with the device starting in the following states
- UnAuthenticated
- Default
- Address
- Configured

Test fails if
The descriptor cannot be read from the DWA.
Any of the fields differ from the description in Chapter 8 of the Wireless USB Specification.

TD.1.11 DWA Isochronous Streaming IN Endpoint descriptor test
This test verifies the fields that come from the device are formatted in compliance with the specification and have appropriate values. Some fields of this descriptor are already verified in WUSBCV and not repeated in this test. This descriptor test is complementary to the descriptor test in WUSBCV.
Device States for Test
This test is run with the DWA in UnAuthenticated, Default, Address, and Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
   - Place the DWA in the desired starting state.
   - Get full configuration descriptor.
   - Parse the data returned; only keeping the Isochronous Streaming IN Endpoint descriptor.
   - Check the returned descriptor for the following values:
     - bmAttributes field must be 01H.
     - wMaxPacketSize field must contain the same value as the DWA Isochronous Streaming OUT Endpoint Companion Descriptor wOverTheAirPacketSize field value.
     - bInterval field must be zero.

Repetitions
Repeat test with the device starting in the following states
   - UnAuthenticated
   - Default
   - Address
   - Configured

Test fails if
The descriptor cannot be read from the DWA.
Any of the fields differ from the description in Chapter 8 of the Wireless USB Specification.

TD.1.12 DWA Wire Adapter RPipe descriptor test
This test verifies the fields that come from the device are formatted in compliance with the specification and have appropriate values.

Device States for Test
This test is run with the DWA in Configured state without performing any enumeration for downstream devices.( with the RPipe in the unconfigured state)

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
   - Place the DWA in the desired starting state.
   - Send Get_Descriptor(wValue, wIndex, wLength, Data) request with the following values:
     - wValue – set to 22H (RPipe descriptor)
     - wIndex – set to RPipe Index.
     - wLength – set to 1CH.
   - Verify if the DWA returns an RPipe descriptor
   - Check the returned descriptor for the following values:
     - For all R Pipes:
       - bLength field must be 1CH
       - bDescriptorType field must be 22H
       - wRPipeIndex must match the RPipe Index that was requested (wIndex field of the GetRPipeDescriptor)
       - (wRequests field value is reported here; functionally tested elsewhere)
       - (wBlocks is reported here; functionally tested elsewhere)
       - bEndpointAddress field bit 6:4 must be zero
       - bwCurrentWindow field must be zero
       - bMaxDataSequence field must be zero
       - bmAttribute field bit 7:6 must be zero
       - bmCharacteristics field is reported here in the form of “Transfer Types supported”; functionally tested elsewhere)
       - bmCharacteristics field bit 7:4 must be zero.
       - bmRetryOptions field bit 7:3 must be zero
     - For R PIPes that support Interrupt or isochronous transfers:
       - wRequests must be at least 2.
Verify that the DWA supports the exact number of R Pipes with wRPipeIndex ranging from zero to wNumRPipes - 1.

**Repetitions**
Repeat for all R PIPes of the DWA.

**Test fails if**
The descriptor cannot be read from the DWA. (all R PIPes are checked)
Any of the descriptor fields differ from the description in Chapter 8 of the Wireless USB Specification.
**RPipe Status and Features tests**

**TD.1.13 Abort RPipe Feature test**

This test verifies that a DWA correctly handles the Abort RPipe feature.

**Device States for Test**

This test is run with the DWA in Configured state on all exposed ports.

**Channels for Test**

Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**

The test software performs the following steps.

- Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
- Place the DWA in Configured state.
- Prompt the user to connect a low speed test device to port 1 of the DWA.
- Configure the test device to always NAK on endpoint to use for test. (This guarantees that the transfer will still be in progress)
- Place the RPipe in Active state for communication to the test device endpoint:
  - Send a total of 'wRequest' of concurrent transfer requests to the RPipe via the Data Transfer Write Endpoint. All transfer requests must be correctly formatted.
- Send an Abort RPipe (RPipe Index) request to the Rpipe
- Verify if the DWA ACK's the Abort RPipe request
- Verify if the DWA correctly aborts all concurrent transfers:
  - Verify if the DWA sends a Transfer Completion Notification for every transfer request.
  - Verify if the DWA sends a Transfer Result to the host with bit 5:0 set to 8 Decimal (TRANSFER_STATUS_ABORTED) and with bit 7 set to 1 (ERROR) for all the aborted transfer requests.
- Send a Get RPipe Status (RPipe Index) request to the DWA.
- Verify if the RPipe has returned to Idle State after aborting the transfer requests
  - Verify if the RPipeState field bit 2 is set to 1 (Configured).
  - Verify if the RPipeState field bit 1 is set to 0 (Not Paused).
  - Verify if the RPipeState field bit 0 is set to 1 (Idle).
- Read NAK count two times from test device (default control endpoint). Verify if the NAK count is not increasing.
- Place the RPipe in Active state for communication to the test device endpoint:
  - Send a total of 'wRequest' of concurrent transfer requests to the RPipe via the Data Transfer Write Endpoint. All transfer requests must be correctly formatted.
- Send a Get RPipe Status (RPipe Index) request to the DWA.
- Verify if the RPipe is in Active State
  - Verify if the RPipeState field bit 2 is set to 1 (Configured).
  - Verify if the RPipeState field bit 1 is set to 0 (Not Paused).
  - Verify if the RPipeState field bit 0 is set to 0 (Active).
- Read NAK count two times from test device. Verify NAK count is increasing.

**Repetitions**

Repeat with the RPipe set to every supported transfer type (Control, Bulk, Interrupt and Isochronous Transfer type).

Repeat with the RPipe set to every supported direction (IN and OUT direction).

And repeat with the RPipe set up to communicate to the test device behind all DWA downstream ports.

And repeat test for all RPIPES of the DWA.

Repeat with test device set up to NAK on all exposed ports and abort all transfers at the same time.

Repeat with test devices running loopback on all exposed ports except the port under test and make sure the abort does not affect the other ports.

**Test fails if**

Any of the described verifications fail.
TD.1.14 RPipe Status and RPIPE_PAUSE Feature test

This test verifies that a DWA correctly handles reports the RPipe status and correctly sets and clears the RPipe RPIPE_PAUSE feature.

**Device States for Test**

This test is run with the DWA in Configured state on all exposed ports.

**Channels for Test**

Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**

The test software performs the following steps.

Initially no downstream devices are attached to the DWA, unless they are non-removable devices.

Place the DWA in Configured state and the RPipe in Un-Configured State.

Send a Get RPipe Status (RPipe Index) request to the DWA.

Verify if the DWA responds with the RPipe status

Verify if the RPipe is in Un-Configured State:
- Verify if the RPipeState field bit 2 is set to 0 (Un-Configured).
- Verify if the RPipeState field bit 7:3 is set to 0. (Reserved)

Prompt the user to connect a low speed test device to port 1 of the DWA.

Configure the test device to always NAK on endpoint to use for test. (This guarantees that the transfer will still be in progress)

Place the RPipe in Idle state for communication to the test device endpoint.

Verify if the DWA responds with the RPipe status

Verify if the RPipe is in Idle State and Configured State:
- Verify if the RPipeState field Bit 0 is set to 1. (Idle)
- Verify if the RPipeState field bit 1 is set to 0. (Not Paused)
- Verify if the RPipeState field bit 2 is set to 1 (Configured).
- Verify if the RPipeState field bit 7:3 is set to 0. (Reserved)

Send a Set RPipe Feature (RPIPE_PAUSE, RPipe Index) request to the RPipe.

Verify if the DWA ACK’s the Set RPipe Feature Pause request

Send a Get RPipe Status (RPipe Index) request to the DWA.

Verify if the RPipe is in Paused State:
- Verify if the RPipeState field Bit 1 is set to 1. (Paused)
- Verify if the RPipeState field bit 2 is set to 1 (Configured).
- Verify if the RPipeState field bit 7:3 is set to 0. (Reserved)

Read NAK count two times from test device (default control endpoint). Verify if the NAK count is not increasing.

Send a Clear RPipe Feature (RPIPE_PAUSE, RPipe Index) request to the RPipe.

Verify if the DWA ACK’s the Clear RPipe Feature Pause request

Send a Get RPipe Status (RPipe Index) request to the DWA.

Verify if the RPipe is in Idle State and Configured State:
- Send a total of ‘wRequest’ of concurrent transfer requests to the RPipe via the Data Transfer Write Endpoint. All transfer requests must be correctly formatted.
- Verify if the RPipe is in Active State:
  - Verify if the RPipe Status request data RPipeState field bit 0 is set to 0. (Active)
  - Verify if the RPipe Status request data RPipeState field bit 1 is set to 0. (Not Paused)
  - Verify if the RPipeState field bit 2 is set to 1 (Configured)

Send a Set RPipe Feature (RPIPE_PAUSE, RPipe Index) request to the RPipe.

Verify if the DWA ACK’s the Set RPipe Feature Pause request

Send a Get RPipe Status (RPipe Index) request to the DWA.

Verify if the RPipe is in Paused State before completing the pending transfers:
- Verify if the RPipeState field Bit 1 is set to 1. (Paused)
- Verify if the RPipeState field bit 2 is set to 1 (Configured).
- Verify if the RPipeState field bit 7:3 is set to 0. (Reserved)

Read NAK count two times from test device (default control endpoint). Verify if the NAK count is not increasing

Send a Clear RPipe Feature (RPIPE_PAUSE, RPipe Index) request to the RPipe.

Send a Get RPipe Status (RPipe Index) request to the DWA.

Verify if the RPipe has returned to the Not Paused State and Active State:
- Verify if the RPipe Status request data RPipeState field bit 0 is set to 0. (Active)
- Verify if the RPipe Status request data RPipeState field bit 1 is set to 0. (Not Paused)
- Verify if the RPipeState field bit 2 is set to 1. (Configured)

Read NAK count two times from test device (default control endpoint). Verify if the NAK count is increasing.

**Repetitions**
Repeat with the RPipe set to every supported transfer type (Control, Bulk, Interrupt and Isochronous Transfer type).
And repeat with the RPipe set to every supported direction (IN and OUT direction).
And repeat with the RPipe set up for LS, FS and HS communication to the test device.
And repeat with the RPipe set up to communicate to the test device behind all DWA downstream ports.
And repeat with test for all R Pipes of the DWA
Repeat with active devices on one or more other ports running loopback and make sure they are not affected by pause on port under test.

**Test fails if**
Any of the described verifications fail.

---

**TD.1.15 Set and Get RPipe Descriptor test**
This test verifies that a DWA correctly handles Set and Get RPipe Descriptor.

**Device States for Test**
This test is run with the DWA in Configured state.

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.

Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
Place the DWA in Configured state and the RPipe in Unconfigured state.
Send a Get RPipe Descriptor (RPipe Index, Descriptor length of 28 decimal)
Verify if the DWA ACK’s the request
Verify if the DWA returns an RPipe descriptor of 28 bytes
Verify if all RPipe Descriptor values are set to default values.
Place the RPipe in Idle state:
- Send a Set RPipe Descriptor (RPipe Index)
Send a Get RPipe Descriptor (RPipe Index, Descriptor length of 28 decimal)
Verify if the DWA ACK’s the request
Verify if the DWA returns an RPipe descriptor of 28 bytes
Verify if the field values of the RPipe Descriptor retrieved by the Get RPipe descriptor match the field values set by the Set RPipe Descriptor request.
Verify if all RPipe Descriptor values of static fields are set to the right values.
Place the RPipe in Active state:
- Send a Transfer Request
Send a Get RPipe Descriptor (RPipe Index, Descriptor length of 28 decimal)
Verify if the DWA ACK’s the request
Verify if the DWA returns an RPipe descriptor of 28 bytes
Verify if the field values of the RPipe Descriptor retrieved by the Get RPipe descriptor match the field values set by the Set RPipe Descriptor request.
Verify if all RPipe Descriptor values of static fields are set to the right values.
Place the RPipe in Paused state:
Send a Get RPipe Descriptor (RPipe Index, Descriptor length of 28 decimal)
Verify if the DWA ACK’s the request
Verify if the DWA returns an RPipe descriptor of 28 bytes
Verify if the field values of the RPipe Descriptor retrieved by the Get RPipe descriptor match the field values set by the Set RPipe Descriptor request.
Verify if all RPipe Descriptor values of static fields are set to the right values.

**Repetitions**
Repeat with the RPipe set to every supported transfer type (Control, Bulk, Interrupt and Isochronous Transfer type).
And repeat with the RPipe set to every supported direction (IN and OUT direction).
And repeat with the RPipe set up for LS, FS and HS communication to the test device.
And repeat with the RPipe set up to communicate to the test device behind all DWA downstream ports.
And repeat test for all R Pipes of the DWA

Test fails if
Any of the described verifications fail.

TD.1.16 Reset RPipe test
This test verifies that a DWA correctly handles the Reset RPipe request.

Device States for Test
This test is run with the DWA in Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
Place the DWA in Configured state.
Prompt the user to connect a low speed test device to port 1 of the DWA.
Place the RPipe in Active state first:
- Do a standard transfer request.
Wait for communication to finish and RPipe to return to Idle state.
Send a Reset RPipe (RPipe Index) request to the DWA.
Send a Get RPipe Status (RPipe Index) request to the DWA.
Verify if the RPipe is in Un-Configured State:
- Verify if the RPipeState field bit 2 is set to 0 (Un-Configured).
Verify if RPipe Descriptor has default values

Repetitions
Repeat with the RPipe set to every supported transfer type (Control, Bulk, Interrupt and Isochronous Transfer type).
And repeat with the RPipe set to every supported direction (IN and OUT direction).
And repeat with the RPipe set up for LS, FS and HS communication to the test device.
And repeat with the RPipe set up to communicate to the test device behind all DWA downstream ports.
And repeat test for all R Pipes of the DWA

Test fails if
Any of the described verifications fail.
Wire Adapter Status and Features test

TD.1.17 WIRE_ADAPTER_ENABLE Feature test
This test verifies if the host controller in the DWA can correctly be enabled and disabled using the WIRE_ADAPTER_ENABLE feature.

Device States for Test
This test is run with the DWA in Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
Place the DWA in Configured state.
Prompt the user to connect a low speed test device to port 1 of the DWA.
Verify if the DWA sends SOF’s on its downstream ports.
- Get SOF count from the test device.
Send a Get Wire Adapter Status request to the DWA.
Verify if the Wire Adapter Host Controller is in Enabled State:
- Verify if the Wire Adapter Status Bit 0 is set to 1 (Enabled)
Send a Clear Wire Adapter feature (WIRE_ADAPTER_ENABLE) request to the DWA.
- wValue – set to One.
- wIndex – lower byte set to target interface number.
- wLength –set to Zero.
Wait for a fixed period of 1 second.
Send a Get Wire Adapter Status request to the DWA.
Verify if the Wire Adapter Host Controller is in Disabled State:
- Verify if the Wire Adapter Status Bit 0 is set to 0 (Disabled)
Send a Set Wire Adapter feature (WIRE_ADAPTER_ENABLE) request to the DWA.
Verify if the DWA did not send SOF’s on its downstream ports during the period the DWA host controller was disabled.
- Immediately get the SOF count – make sure that it hasn’t increased by more than 50. (should be nowhere near 1000 which represents a second worth of increase).
Verify if the DWA sends SOF’s on its downstream ports.

Repetitions
And repeat with the RPipe set up for LS, FS and HS communication to the test device

Test fails if
Any of the described verifications fail.

TD.1.18 WIRE_ADAPTER_RESET Feature test
This test verifies if the host controller in the DWA can correctly be enabled and disabled using the WIRE_ADAPTER_RESET feature.

Device States for Test
This test is run with the DWA in Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
Place the DWA in Configured state.
Prompt the user to connect a low speed test device to port 1 of the DWA
Enumerate the test device
Place the RPipe in Idle state:
- Send a Set RPipe Descriptor (RPipe Index)
Send a Set Wire Adapter feature (WIRE_ADAPTER_RESET) request to the DWA.
Send a Get RPipe Status (RPipe Index) request to the DWA.
Verify if the RPipe is in Un-Configured State.
- Verify if the RPipeState field bit 2 is set to 0 (Un-Configured).
- Verify if the RPipeState field bit 7:3 is set to 0. (Reserved)
Send a Get Port Status command to the DWA.
Verify if the port status is in default state.
- Verify if bit 1 is set to 0. (Port disabled)
Verify if RPipe Descriptor has default values

**Repetitions**
Repeat with the RPipe set to every supported transfer type (Control, Bulk, Interrupt and Isochronous Transfer type).
And repeat with the RPipe set to every supported direction (IN and OUT direction).
And repeat with the RPipe set up for LS, FS and HS communication to the test device.
And repeat with the RPipe set up to communicate to the test device behind all DWA downstream ports.
And repeat test for all RPipes of the DWA.
And repeat for all downstream ports of the DWA tested individually
And Repeat for all downstream ports of the DWA tested simultaneously

**Test fails if**
Any of the described verifications fail.

---

**TD.1.19 Get Wire Adapter Status request test**
This test verifies if the DWA correctly reports the DWA host controller status upon a Get Wire Adapter Status request.

**Device States for Test**
This test is run with the DWA in Configured state.

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.
- Place the DWA in Configured state but do not send the set feature reset and enable to the DWA host controller yet.
- Send a Get Wire Adapter Status request to the DWA.
- Verify if the Wire Adapter Host Controller is in Disabled State:
  - Verify if the Wire Adapter Status Bit 0 is set to 0 (Disabled)
  - Verify if the Wire Adapter Status Bit 1 is set to 0 (No Reset in progress)
  - Verify if the Wire Adapter Status bits 31:2 are set to zero.
- Send a Set Wire Adapter feature (WIRE_ADAPTER_RESET) request to the DWA.
- Send a Set Wire Adapter feature (WIRE_ADAPTER_ENABLE) request to the DWA.
- Send a Get Wire Adapter Status request to the DWA.
- Verify if the Wire Adapter Host Controller is in Enabled State:
  - Verify if the Wire Adapter Status Bit 0 is set to 1 (Enabled)
  - Verify if the Wire Adapter Status Bit 1 is set to 0 (No Reset in progress)
  - Verify if the Wire Adapter Status bits 31:2 are set to zero.
- Send a Clear Wire Adapter feature (WIRE_ADAPTER_ENABLE) request to the DWA.
- Send a Get Wire Adapter Status request to the DWA.
- Verify if the Wire Adapter Host Controller is in Disabled State:
  - Verify if the Wire Adapter Status Bit 0 is set to 0 (Disabled)
  - Verify if the Wire Adapter Status Bit 1 is set to 0 (No Reset in progress)
  - Verify if the Wire Adapter Status bits 31:2 are set to zero.
- Send a Set Wire Adapter feature (WIRE_ADAPTER_ENABLE) request to the DWA.
- Send a Set Wire Adapter feature (WIRE_ADAPTER_RESET) request to the DWA.
- Send a Get Wire Adapter Status request to the DWA.
- Verify if the Wire Adapter Host Controller is in the Reset State if the reset is still in progress or in the Disabled State (default) if the reset is already completed:
  - Verify if the Wire Adapter Status Bit 0 is set to 0 (Disabled) or verify if the Wire Adapter Status Bit 1 is set to 1 (Reset in progress)
- Wait for 10 seconds.
- Send a Get Wire Adapter Status request to the DWA.
- Verify that the Wire Adapter Host Controller is in the Disabled State.
  - Verify if the Wire Adapter Status Bit 0 is set to 0 (Disabled)
  - Verify if the Wire Adapter Status Bit 1 is set to 0 (No Reset in progress)
- Verify if the Wire Adapter Status bits 31:2 are set to zero.

**Test fails if**

Any of the described verifications fail.
**Standard Loopback tests**

**TD.1.20 Control Loopback test**
This test performs functional testing of control transfers by exercising streams of OUT and IN transfer pairs to the test device control endpoint. All DWA RPipes that support control transfers in their RPpipe descriptor are tested. Control Write and Read Transfer Requests are scheduled one after another to the DWA RPipe under test targeting the test device control loopback endpoint.

**Device States for Test**
This test is run with the DWA in Configured state.

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.

- Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
- Place the DWA in Configured state.
- Prompt the user to connect a full speed test device to port 1 of the DWA.
- Configure the test device for control endpoint loopback.
  - MaxPacketSize of the control endpoint is initially set to 64 bytes.
- Place the RPipe in Idle state and set up for communication to a test device control endpoint.
- Schedule a Control Write Transfer Requests to the DWA RPipe to transfer control data to the test device control endpoint.
- Check the Transfer Result fields for every segment:
  - bTransferStatus bit 5:0 must be set to 0 Decimal (TRANSFER_STATUS_SUCCESS)
- Schedule a Control Read Transfer Requests to the DWA RPipe to transfer control data from the test device control endpoint.
- Check the Transfer Result fields for every segment:
  - bTransferStatus bit 5:0 must be set to 0 Decimal (TRANSFER_STATUS_SUCCESS)
- Verify if the OUT data matches the IN data. (the same data and in the same order)

**Repetitions**
Repeat for all control transfer sizes (from single packet to 64K) (Note: OUT and IN transfer request segments should be sent one after another to the DWA Data Transfer Write Endpoint because of limited buffering on the test device endpoints)
And repeat with the DWA RPipe and the default control endpoint of the test device set to all allowed Max Packetsizes for Control Transfers:
  - for LS: 8 bytes
  - for FS: 8, 16, 32 and 64 bytes
  - for HS: 64 bytes
And repeat for LS, FS & HS test device
And repeat for all byte repeating data patterns
And repeat for all RPipe number of blocks (if configurable)
And repeat for all allowed burst sizes up to bMaxBust of the DWA Data Transfer Endpoints
And repeat for all supported PHY Data Rates
And repeat for test device on every downstream port of the DWA
And repeat for all RPipes that support Control traffic.

**Test fails if**
Any of the described verifications fail.

**TD.1.21 Bulk Loopback test**
This test performs functional testing of bulk transfers by exercising concurrent streams of OUT and IN transfer pairs to the test device bulk loopback endpoint pair. All DWA RPipes that support bulk transfers in their RPpipe descriptor are tested. Bulk Write and Read Transfer Requests are scheduled to the 2 DWA RPipes under test targeting the test device bulk loopback endpoint pair.

**Device States for Test**
This test is run with the DWA in Configured state.

**Channels for Test**
Overview of Test Steps
The test software performs the following steps.

Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
Place the DWA in Configured state.
Prompt the user to connect a full speed test device to port 1 of the DWA.
Configure the test device for a bulk loopback endpoint.
- MaxPacketSize of the bulk endpoint is initially set to 8 bytes.
Place one RPipe in Idle state and set up for communication to a test device bulk OUT endpoint.
Place a second RPipe in Idle state and set up for communication to a test device bulk IN endpoint.
Schedule a Bulk OUT Transfer Request to the DWA RPipe that targets the test device Bulk OUT loopback endpoint.
Check the Transfer Result fields for every segment:
  - bTransferStatus bit 5:0 must be set to 0 Decimal (TRANSFER_STATUS_SUCCESS)
Schedule a Bulk IN Transfer Request to the DWA RPipe that targets the test device Bulk IN loopback endpoint.
Check the Transfer Result fields for every segment:
  - bTransferStatus bit 5:0 must be set to 0 Decimal (TRANSFER_STATUS_SUCCESS)
Verify if the OUT data matches the IN data. (the same data and in the same order)

Repetitions
Repeat for all bulk transfer sizes (2 X number of blocks X block size) (Note: OUT and IN transfer request segments should be sent concurrently to the DWA Data Transfer Write Endpoint because of limited buffering on the test device endpoints)
And repeat for the DWA RPipe and the test device endpoint configured to all allowed Bulk max packet sizes:
- FS: 8, 16, 32 and 64 bytes
- HS: 512 bytes
And repeat for FS and HS test device
And repeat for an incrementing 4 byte patterns (00 00 00 1 to FF FF FF FF)
And repeat for all RPipe number of blocks (if configurable)
And repeat for all allowed burst sizes up to bMaxBust of the DWA Data Transfer Endpoints
And repeat for all possible number of concurrent requests
And repeat for all supported PHY Data Rates
And repeat for test device on every downstream port of the DWA
And repeat for all RPIpes that support Bulk traffic

Test fails if
Any of the described verifications fail.

TD.1.22 Interrupt Loopback test
This test performs functional testing of interrupt transfers by exercising concurrent streams of OUT and IN transfer pairs to the test device interrupt loopback endpoint pair. All DWA RPIpes that support interrupt transfers in their RPipe descriptor are tested. Interrupt Write and Read Transfer Requests are scheduled to the 2 DWA RPIpes under test targeting the test device interrupt loopback endpoint pair.

Device States for Test
This test is run with the DWA in Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.

Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
Place the DWA in Configured state.
Prompt the user to connect a full speed test device to port 1 of the DWA.
Configure the test device for a interrupt loopback endpoint.
- MaxPacketSize of the interrupt endpoint is initially set to 8 bytes.
- Interval of the interrupt endpoint is initially set to 1.
Place one RPipe in Idle state and set up for communication to a test device interrupt OUT endpoint.
Place a second RPipe in Idle state and set up for communication to a test device interrupt IN endpoint.
Schedule an Interrupt OUT Transfer Requests to the DWA RPipe that targets the test device Interrupt OUT loopback endpoint.
Check the Transfer Result fields for every segment:
- bTransferStatus bit 5:0 must be set to 0 Decimal (TRANSFER_STATUS_SUCCESS)
Schedule an Interrupt IN Transfer Requests to the DWA RPipe that targets the test device Interrupt IN loopback endpoint.
Check the Transfer Result fields for every segment:
- bTransferStatus bit 5:0 must be set to 0 Decimal (TRANSFER_STATUS_SUCCESS)
Verify that the OUT data matches the IN data. (the same data and in the same order)
Verify that the DWA used the correct rates to poll the downstream interrupt OUT and IN endpoint pair of the test device
- Measure the completion time of the transfer request loopback at the host side
- This time should be less than 2 X (bOverTheAirInterval + bInterval) (for OUT + IN) and more than 2 X (bOverTheAirInterval + next and faster polling rate bInterval)
- Measure with test device if possible

Repetitions
Repeat for all interrupt transfer sizes (2 X number of blocks X block size) (Note: OUT and IN transfer request segments should be sent concurrently to the DWA Data Transfer Write Endpoint because of limited buffering on the test device endpoints)
And repeat for all MaxPacketSizes
And repeat for the DWA RPipe and the test test device endpoint configured to all allowed Interrupt packet sizes.
And repeat with the DWA RPipe and the test device endpoint set to all binterval settings.
And repeat for LS, FS and HS test device
And repeat for an incrementing 4 byte patterns (00 00 00 1 to FF FF FF FF)
And repeat for all RPipe number of blocks (if configurable)
And repeat for all allowed burst sizes up to bMaxBust of the DWA Data Transfer Endpoints
And repeat for all possible number of concurrent requests
And repeat for all supported PHY Data Rates
And repeat for test device on every downstream port of the DWA
And repeat for all RPIPES that support Interrupt with the RPipe configured in OUT direction
And repeat for all RPIPES that support Interrupt with the RPipe configured in IN direction
Test fails if
Any of the described verifications fail.

TD.1.23 Isochronous Streaming Loopback test
This test performs functional testing of isochronous streams by exercising streams of isochronous OUT and IN pairs to the test device loopback endpoint pair. All DWA RPIPES that support isochronous streaming in their RPipe descriptor are tested. The isochronous OUT stream and IN stream are scheduled to the 2 DWA RPIPES under test targeting the test device isochronous loopback endpoint pair.
Device States for Test
This test is run with the DWA in Configured state.
Channels for Test
Test will be run on one FFI channel and one TFI channel.
Overview of Test Steps
The test software performs the following steps.
Place the host and the DWA in optimal conditions for wireless communication. (in eachothers line of sight and very close to eachother)
Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
Place the DWA in Configured state.
Prompt the user to connect a full speed test device to port 1 of the DWA.
Place one RPipe in Idle state and set up for communication to a test device isochronous OUT endpoint.
Place a second RPipe in Idle state and set up for communication to a test device isochronous IN endpoint.
Configure the test device for an isochronous loopback loopback endpoint pair.
Isochronous OUT endpoint of the test device will be varied:
- MaxPacketSize of the isochronous test device endpoint is initially set to 1 byte
- bInterval of 1.
Isochronous IN endpoint of the test device will be varied:
   - MaxPacketSize of the isochronous test device endpoint is initially set to 1 byte
   - bInterval of 1.
Start Isochronous OUT Stream to the DWA
Start Isochronous IN Stream to the DWA
Verify that the DWAIsochronous Streaming OUT endpoint does not NAK any Isochronous data
   packets
Verify if the OUT data matches the IN data. (the same data and in the same order)
Verify if the DWA aggregates wired isochronous IN packet data into the largest packets that can be
   sent over the air by the associated Wireless USB Isochronous endpoint

Repetitions
Repeat for all MaxPacketSizes
And repeat for the test device endpoint configured to all Isochronous packet sizes and number of
transactions per microframe.
And repeat with the test device endpoint set to all bInterval settings
And repeat for all bOverTheAirInterval Settings
And repeat for FS and HS test device
And repeat for all RPipe number of blocks (if configurable)
And repeat for all allowed burst sizes up to bMaxBust of the DWA Data Transfer Endpoints
And repeat for all supported PHY Data Rates for isochronous streaming
And repeat for all Upstream Isochronous Streaming OUT and IN Endpoints of the DWA
And repeat for test device on every downstream port of the DWA
And repeat for all RPipes that support isochronous

Test fails if
Any of the described verifications fail.
**Functional tests**

**TD.1.24 Control Loopback behind High Speed hub test**

This test functionally validates if control transfers are correctly sent and received by the DWA to and from a test device when the test device is connected to the DWA behind a High Speed Hub.

**Device States for Test**
This test is run with the DWA in Configured state.

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.
- Perform the Control loopback test (TD.1.20) with the following changes:
  - Connect the test device behind a High Speed Hub to the DWA

**Repetitions**
Repeat test for LS, FS and HS test device
And repeat with chain of hubs from 1 till 5
Any of the verifications in the parent test fails. (TD.1.20)

**TD.1.25 Bulk Loopback behind High Speed hub test**

This test functionally validates if bulk transfers are correctly sent and received by the DWA to and from a test device when the test device is connected to the DWA behind a High Speed Hub.

**Device States for Test**
This test is run with the DWA in Configured state.

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.
- Perform the Bulk loopback test (TD.1.21) with the following changes:
  - Connect the test device behind a High Speed Hub to the DWA

**Repetitions**
Repeat test for FS and HS test device
And repeat with chain of hubs from 1 till 5
Any of the verifications in the parent test fails. (TD.1.21)

**TD.1.26 Interrupt Loopback behind High Speed hub test**

This test functionally validates if interrupt transfers are correctly sent and received by the DWA to and from a test device when the test device is connected to the DWA behind a High Speed Hub.

**Device States for Test**
This test is run with the DWA in Configured state.

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.
- Perform the Interrupt loopback test (TD.1.22) with the following changes:
  - Connect the test device behind a High Speed Hub to the DWA

**Repetitions**
Repeat test for LS, FS and HS test device
And repeat with chain of hubs from 1 till 5
Any of the verifications in the parent test fails. (TD.1.22)
TD.1.27 Isochronous Loopback behind High Speed hub test
This test functionally validates if isochronous transfers are correctly sent and received by the DWA to and from a test device when the test device is connected to the DWA behind a High Speed Hub.

Device States for Test
This test is run with the DWA in Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
- Perform the Isochronous loopback test (TD.1.23) with the following changes:
  - Connect the test device behind a High Speed Hub to the DWA

Repetitions
Repeat test for FS and HS test device
And repeat with chain of hubs from 1 till 5

Test fails if
Any of the verifications in the parent test fails. (TD.1.22)

TD.1.28 Target Device Address and Endpoint Address test
This test functionally validates the DWA’s use of the Wire Adapter RPipe Descriptor bDeviceAddress, bEndpointAddress and bDeviceInfoIndex fields.

Device States for Test
This test is run with the DWA in Configured state.

Overview of Test Steps
The test software performs the following steps.
- Place the DWA in Configured state.
- Connect the test device to the DWA.
- Configure the test device for loopback endpoint.
- Place two RPipes in Idle state and set up for communication to the test device loopback endpoint pair.
- Perform loopback testing
- Send a Set Address command to the test device to assign a new and different device address to the test device
- Place the two RPipes in Idle state and set up for communication to the test device loopback endpoint pair using the new device address in the RPipe Descriptor.
- Perform loopback testing

Repetitions
Repeat for all traffic types.
And repeat for switching between various target device addresses
And repeat for switching between various target endpoint addresses
And repeat for all RPipes.
And repeat for the RPipe configured for OUT traffic and configured for IN traffic.
And repeat test for LS, FS and HS test device
And repeat for test device connected behind 5 hubs

Test fails if
Any of the verifications in the parent test fails. (TD.1.20 TD.1.21 TD.1.22 TD.1.23)
Any of the described verifications fail.

TD.1.29 DWA Set Isochronous Endpoint Attributes test
This test verifies if a DWA correctly sets and applies the attributes that are provided by a Set Isochronous Endpoint Attributes request.

Device States for Test
This test is run with the DWA in Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.

Perform the Isochronous loopback test (TD.1.23) with the following changes:
- Re-configure the DWA’s upstream Isochronous endpoint for the OUT stream:
  - wValue – set to Zero.
  - wIndex – lower byte set to target upstream wireless Isochronous endpoint of the DWA.
  - wLength – set to 6 decimal.
- Endpoint Attributes:
  - wMaxStreamDelay of at least big enough to hold 8,192 ms of data
  - wOverTheAirPacketSize set to various values all big enough to provide bandwidth for
downstream communication (bInterval x maxpacketsize downstream) during a
  wMaxStreamDelay interval.
- Re-configure the DWA’s upstream Isochronous endpoint for the IN stream in the same
  manner.

Repetitions
Repeat test for FS and HS test device
Repeat for all possible wMaxStreamDelay and wOverTheAirPacketSize combinations per loopback
instance.
Test fails if
Any of the verifications in the parent test fails. (TD.1.23).

TD.1.30 Interrupt IN downstream polling interval test
This test performs functional testing of the downstream polling interval set in the DWA RPipe descriptor
and used for downstream interrupt IN transfers.

Device States for Test
This test is run with the DWA in Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.

- Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
- Place the DWA in Configured state.
- Prompt the user to connect a full speed test device to port 1 of the DWA.
- Configure the test device for an interrupt IN endpoint in a special mode.
  - MaxPacketSize of the interrupt endpoint is initially set to 8 bytes.
  - Interval of the interrupt endpoint is initially set to 1.
  - the test device endpoint is placed in a special interrupt IN source mode where the test device
    places the current SOF values in the interrupt data packets
- Place one RPipe in Idle state and set up for communication to a test device interrupt IN endpoint.
- Schedule an Interrupt IN Transfer Requests to the DWA RPipe that targets the test device Interrupt
  IN loopback endpoint.
- Check the Transfer Result fields for every segment:
  - bTransferStatus bit 5:0 must be set to 0 Decimal (TRANSFER_STATUS_SUCCESS)
- Verify that the DWA used the correct rates to poll the downstream interrupt IN endpoint pair of the
  test device
  - Analyze the interrupt data packets containing the SOF values
  - The downstream polling interval time for the IN transactions should be less or equal than
    bInterval of the DWA RPipe descriptor

Repetitions
Repeat for all interrupt transfer sizes (2 X number of blocks X block size) (Note: OUT and IN transfer
request segments should be sent concurrently to the DWA Data Transfer Write Endpoint because of
limited buffering on the test device endpoints)
And repeat for all MaxPacketSizes
And repeat for the DWA RPipe and the test test device endpoint configured to all allowed Interrupt packet sizes.
And repeat with the DWA RPipe and the test device endpoint set to all binterval settings.
And repeat for LS, FS and HS test device
And repeat for all RPipe number of blocks (if configurable)
And repeat for all allowed burst sizes up to bMaxBust of the DWA Data Transfer Endpoints
And repeat for all supported PHY Data Rates
And repeat for test device on every downstream port of the DWA
And repeat for adding heavy traffic (loopbacks)
And repeat for all RPipes that support Interrupt

Test fails if
Any of the described verifications fail.

**TD.1.31 Interrupt OUT downstream polling interval test**

This test performs functional testing of the downstream polling interval set in the DWA RPipe descriptor and used for downstream interrupt OUT transfers.

**Device States for Test**

This test is run with the DWA in Configured state.

**Channels for Test**

Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**

The test software performs the following steps.

Perform the Interrupt loopback test (TD.1.22) with the following changes:

- Configure the test device for interrupt IN endpoint in a special mode:
  - the test device endpoint is placed in a special interrupt IN mode as part of an interrupt loopback pair where the IN endpoint appends the current SOF values to the interrupt data packets received via the OUT endpoint
  - Verify that the DWA used the correct rates to poll the downstream interrupt IN endpoint pair of the test device
  - Analyze the interrupt data packets containing the SOF values
  - The downstream polling interval time for the OUT transactions should be less or equal than binterval of the DWA RPipe descriptor

**Repetitions**

And repeat for adding heavy traffic (loopbacks)

Test fails if
Any of the described verifications fail.
Any of the verifications in the parent test fails. (TD.1.22)

**TD.1.32 Isochronous IN downstream polling interval test**

This test performs functional testing of the downstream polling interval set in the DWA RPipe descriptor and used for downstream isochronous IN transfers.

**Device States for Test**

This test is run with the DWA in Configured state.

**Channels for Test**

Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**

The test software performs the following steps.

Place the host and the DWA in optimal conditions for wireless communication. (in eachothers line of sight and very close to eachother)
Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
Place the DWA in Configured state.
Prompt the user to connect a full speed test device to port 1 of the DWA.
Configure the test device for an isochronous IN endpoint in a special mode.
- MaxPacketSize of the isochronous test device endpoint is initially set to 1 byte
- binterval of 1.
*the test device endpoint is placed in a special isochronous IN source mode where the test device places the current SOF values in the isochronous data packet*

Place an RPipe in Idle state and set up for communication to a test device isochronous IN endpoint.

Start Isochronous IN Stream to the DWA

Verify that the DWA used the correct rates to poll the downstream isochronous IN endpoint pair of the test device

- Analyze the isochronous data packets containing the SOF values
- The downstream polling interval time for the IN transactions should be less or equal than bInterval of the DWA RPipe descriptor

**Repetitions**

Repeat for all MaxPacketSizes
And repeat for the test device endpoint configured to all Isochronous packet sizes and number of transactions per microframe.
And repeat with the test device endpoint set to all bInterval settings
And repeat for all bOverTheAirInterval Settings
And repeat for FS and HS test device
And repeat for all RPipe number of blocks (if configurable)
And repeat for all allowed burst sizes up to bMaxBust of the DWA Data Transfer Endpoints
And repeat for all supported PHY Data Rates for isochronous streaming
And repeat for all Upstream Isochronous Streaming OUT and IN Endpoints of the DWA
And repeat for test device on every downstream port of the DWA
And repeat for adding heavy traffic (loopbacks)
And repeat for all R Pipes that support isochronous

*Test fails if*  
Any of the described verifications fail.

---

**TD.1.33 Isochronous OUT downstream polling interval test**

This test performs functional testing of the downstream polling interval set in the DWA RPipe descriptor and used for downstream isochronous OUT transfers.

**Device States for Test**

This test is run with the DWA in Configured state.

**Channels for Test**

Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**

The test software performs the following steps.

Perform the Isochronous loopback test (TD.1.23) with the following changes:

- Configure the test device for isochronous IN endpoint in a special mode:
  - the test device endpoint is placed in a special isochronous IN mode as part of an isochronous loopback pair where the IN endpoint appends the current SOF values to the interrupt data packets received via the OUT endpoint
  - Verify that the DWA used the correct rates to poll the downstream isochronous IN endpoint pair of the test device
  - Analyze the isochronous data packets containing the SOF values
  - The downstream polling interval time for the OUT transactions should be less or equal than bInterval of the DWA RPipe descriptor

**Repetitions**

And repeat for adding heavy traffic (loopbacks)

*Test fails if*  
Any of the described verifications fail.  
Any of the verifications in the parent test fails. (TD.1.23)
Transfer Request and Transfer Result tests

TD.1.34 DWA Control Write Transfer Requests and Transfer Result test

This test verifies if the DWA correctly handles various Control Transfer Write Requests in terms of producing transfer completion notifications, reading the transfer requests and producing the transfer results.

Device States for Test
This test is run with the DWA in Configured state, with the RPipe in Idle state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.

Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
Place the DWA in Configured state.
Prompt the user to connect a full speed test device to port 1 of the DWA.
Place the RPipe in Idle state and set up for communication to a test device control endpoint.
Configure the test device for control endpoint with a perfect sink for data.
- MaxPacketSize of the control endpoint is initially set to 64 bytes.

Schedule a Control Write Transfer Requests to the DWA RPipe to transfer control data to the test device control endpoint with the following characteristics all Transfer Request segments:
- bLength set to 18H
- bRequestType set to 80H
- wRPipe set to the index of the RPipe under test
- dwTransferLength set to a unique ID
- dwTransferID set to the amount of data following: (N - #segments transferred) x RPipe wMaxPacketSize
- bTransferSegment bit 6:0 set to incrementing segment numbers and bit 7 set to 0 for all segments excepts for the last segment
- bmAttribute set bit 0 set to 0 (set to control transfer write)
- wReserved set to zero
- baSetupData set to the setup packet data.

Verify if the DWA sends back a Transfer Complete Notification on the Notification Endpoint after every Control Transfer Request Segment.

Check the Transfer Complete Notification fields:
- bLength set to 4 decimal.
- bNotifyType set to 93H.
- bEndpoint set to the Data Transfer Read Endpoint.
- bReserved set to zero.

Verify if the DWA sends back a Transfer Result on the Data Transfer Read Endpoint after every Transfer Complete Notification.

Check the Transfer Result fields for every segment:
- bLength set to 10H.
- bResultType set to 83H.
- dwTransferID must be set to dwTransferID of the Transfer Request
- dwTransferLength must be set to to the amount of data transferred: #segments transferred x wMaxPacketSize.
- bTransferSegment bit 6:0 must be set to incrementing segment numbers and bit 7 set to 0 for all segments excepts for the last segment
- bTransferStatus bit 5:0 must be set to 0 Decimal (TRANSFER_STATUS_SUCCESS) and bit 6:7 must be set to 0 as well.
- dwNumOfPackets must be set to zero.

Verify if the Transfer Results are recieved in incrementing order of TransferSegment number (WA Transfer Result bTransferSegment field Bit6:0 must be incrementing over the received WA Transfer Result Packets)
Verify if no more Transfer Results (and data segments for IN) are received with the dwTransferID after a WA Transfer Result with bTransferSegment field Bit7 set to one is received. (wait for 5 sec to determine)

**Repetitions**
Repeat for multiple Transfer Request sizes (Transfer Request with incrementing number of segments from 1 till 128)
And repeat for with the default control endpoint of the test device set to all allowed Max Packetsizes for Control Transfers:
- for LS: 8 bytes
- for FS: 8, 16, 32 and 64 bytes
- for HS: 64 bytes
And repeat for LS, FS & HS test device.
And repeat for all supported PHY Data Rates
And repeat for test device on every downstream port of the DWA
And repeat for all R Pipes that support Control traffic

**Test fails if**
Any of the described verifications fail.

---

**TD.1.35 DWA Control Read Transfer Requests and Transfer Result test**

This test verifies if the DWA correctly handles various Control Transfer Read Requests in terms of producing transfer completion notifications, reading the transfer requests and producing the transfer results.

**Device States for Test**
This test is run with the DWA in Configured state, with the RPipe in Idle state.

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.

- Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
- Place the DWA in Configured state.
- Prompt the user to connect a full speed test device to port 1 of the DWA.
- Place the RPipe in Idle state and set up for communication to a test device control endpoint.
- Configure the test device for control endpoint with a perfect source for data.
  - MaxPacketSize of the control endpoint is initially set to 64 bytes.
- Schedule a Control Read Transfer Requests to the DWA RPipe to transfer control data to the test device control endpoint with the following characteristics for all Transfer Request segments:
  - bLength set to 18H
  - bRequestType set to 80H
  - wRPipe set to the index of the RPipe under test
  - dwTransferID set to a unique ID
  - dwTransferLength set to amount of data following: (N - #segments transferred) x RPipe wMaxPacketSize
  - bTransferSegment bit 6:0 set to incrementing segment numbers and bit 7 set to 0 for all segments excepts for the last segment
  - bmAttribute set bit 0 set to 0 (set to control transfer read)
  - wReserved set to zero
  - baSetupData set to the setup packet data.
- Verify if the DWA sends back a Transfer Complete Notification on the Notification Endpoint after every Control Transfer Request Segment.
- Check the Transfer Complete Notification fields:
  - bLength set to 4 decimal.
  - bNotifyType set to 93H.
  - bEndpoint set to the Data Transfer Read Endpoint.
  - bReserved set to zero.
- Verify if the DWA sends back a Transfer Result on the Data Transfer Read Endpoint after every Transfer Complete Notification.
- Check the Transfer Result fields for every segment:
- bLength set to 10H.
- bResultType set to 83H.
- dwTransferID must be set to dwTransferID of the Transfer Request
- dwTransferLength must be set to to the amount of data transferred (#segments transferred x wMaxPacketSize).
- bTransferSegment bit 6:0 must be set to incrementing segment numbers and bit 7 set to 0 for all segments excepts for the last segment
- bTransferStatus bit 5:0 must be set to 0 Decimal (TRANSFER_STATUS_SUCCESS) and bit 6:7 must be set to set to 0 as well.
- dwNumOfPackets must be set to zero.

Verify if the DWA provides data on the Data Transfer Read Endpoint as the consecutive transfer after a Transfer Result.
Verify if the data of all the segments have the size of wMaxPacketSize of the RPipe Descriptor.
Verify if the Transfer Results are received in incrementing order of TransferSegment number (WA Transfer Result bTransferSegment field Bit6:0 must be incrementing over the received WA Transfer Result Packets)
Verify if no more Transfer Results and data segments for IN are received with the dwTransferID after a WA Transfer Result with bTransferSegment field Bit7 set to one is received. (wait for 5 sec to determine)

**Repetitions**
Repeat for multiple Transfer Request sizes (Transfer Request with incrementing number of segments from 1 till 128)
And repeat for with the default control endpoint of the test device set to all allowed Max Packetsizes for Control Transfers:
- for LS: 8 bytes
- for FS: 8, 16, 32 and 64 bytes
- for HS: 64 bytes
And repeat for LS, FS & HS test device.
And repeat for all supported PHY Data Rates
And repeat for test device on every downstream port of the DWA
And repeat for all RPipes that support Control traffic

**Test fails if**
Any of the described verifications fail.

**TD.1.36 DWA Bulk OUT Transfer Requests and Transfer Result test**
This test verifies if the DWA correctly handles various Bulk OUT Transfer Requests in terms of producing transfer completion notifications, reading the transfer requests and producing the transfer results

**Device States for Test**
This test is run with the DWA in Configured state, with the RPipe in Idle state

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.
Initially no downstream devices are attached to the DWA, unless they are non-removable devices. Place the DWA in Configured state.
Prompt the user to connect a Full Speed test device to port 1 of the DWA for the initial test cycle. Configure the test device for bulk endpoint with a perfect sink for data.
- MaxPacketSize of the bulk endpoint is initially set to 8 bytes.
Schedule a Bulk OUT Transfer Requests to the DWA RPipe to transfer bulk data to the test device control endpoint with the following characteristics all Transfer Request segments:
- bLength set to 10H
- bRequestType set to 81H
- wRPipe set to the index of the RPipe under test
- dwTransferID set to a unique ID
- dwTransferLength set to amount of data following: (N - #segments transferred) x RPipe wMaxPacketSize.
- bTransferSegment bit 6:0 must be set to the Transfer Segment and bit 7 must be set to 1 if this is the last Transfer Request segment.
- bReserved set to zero
- wReserved set to zero
Verify if the DWA sends back a Transfer Complete Notification on the Notification Endpoint after every Transfer Request Segment.
Check the Transfer Complete Notification fields:
- bLength set to 4 decimal.
- bNotifyType set to 93H.
- bEndpoint set to the Data Transfer Read Endpoint.
- bReserved set to zero.
Verify if the DWA sends back a Transfer Result after every Transfer Segment on the Data Transfer Read Endpoint.
Check the Transfer Result fields for every segment:
- bLength set to 10H.
- bResultType set to 83H.
- dwTransferID must be set to dwTransferID of the Transfer Request
- dwTransferLength must be set to to the amount of data transferred: #segments transferred x wMaxPacketSize.
- bTransferSegment bit 6:0 must be set to the Transfer Segment and bit 7 must be set to 1 if this is the last Transfer Request segment.
- bTransferStatus bit 5:0 must be set to 0 Decimal (TRANSFER_STATUS_SUCCESS) and bit 6:7 must be set to set to 0 as well.
- dwNumOfPackets must be set to zero.
Verify if the Transfer Results are recieved in incrementing order of TransferSegment number (WA Transfer Result bTransferSegment field Bit6:0 must be incrementing over the received WA Transfer Result Packets)
Verify if no more Transfer Results are received with the dwTransferID after a WA Transfer Result with bTransferSegment field Bit7 set to one is received. (wait for 5 sec to determine)

Repetitions
Repeat for multiple Transfer Request sizes (Transfer Request with incrementing number of segments from 1 till 128)
And repeat for the test device endpoint configured to all allowed Bulk max packet sizes:
- FS: 8, 16, 32 and 64 bytes
- HS: 512 bytes
And repeat for FS and HS test device
And repeat for all supported PHY Data Rates
And repeat for test device on every downstream port of the DWA
And repeat for all RPIpes that support Bulk

Test fails if
Any of the described verifications fail.

TD.1.37 DWA Bulk IN Transfer Requests and Transfer Result test
This test verifies if the DWA correctly handles various Bulk IN Transfer Requests in terms of producing transfer completion notifications, reading the transfer requests and producing the transfer results

Device States for Test
This test is run with the DWA in Configured state, with the RPipe in Idle state

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
Place the DWA in Configured state.
Prompt the user to connect a Full Speed test device to port 1 of the DWA for the initial test cycle.
Configure the test device for bulk endpoint with a perfect source for data.
- MaxPacketSize of the bulk endpoint is initially set to 8 bytes.
Schedule a Bulk IN Transfer Requests to the DWA RPipe to transfer bulk data to the test device endpoint with the following characteristics all Transfer Request segments:
- bLength set to 10H
- bRequestType set to 81H
- wRPipe set to the index of the RPipe under test
- dwTransferID set to a unique ID
- dwTransferLength set to amount of data following: (N - #segments transferred) x RPipe wMaxPacketSize.
- bTransferSegment bit 6:0 must be set to the Transfer Segment and bit 7 must be set to 1 if this is the last Transfer Request segment.
- bReserved set to zero
- wReserved set to zero

Verify if the DWA sends back a Transfer Complete Notification on the Notification Endpoint after every Transfer Request Segment.

Check the Transfer Complete Notification fields:
- bLength set to 4 decimal.
- bNotifyType set to 93H.
- bEndpoint set to the Data Transfer Read Endpoint.
- bReserved set to zero.

Verify if the DWA sends back a Transfer Result after every Transfer Segment on the Data Transfer Read Endpoint.

Check the Transfer Result fields for every segment:
- bLength set to 10H.
- bResultType set to 83H.
- dwTransferID must be set to dwTransferID of the Transfer Request
- dwTransferLength must be set to to the amount of data transferred (#segments transferred x wMaxPacketSize).
- bTransferSegment bit 6:0 must be set to the Transfer Segment and bit 7 must be set to 1 if this is the last Transfer Request segment.
- bTransferStatus bit 5:0 must be set to 0 Decimal (TRANSFER_STATUS_SUCCESS) and bit 6:7 must be set to set to 0 as well.
- dwNumOfPackets must be set to zero.

Verify if the DWA provides data on the Data Transfer Read Endpoint as the consecutive transfer after a Transfer Result.

Verify if the data of all the segments have the size of wMaxpacketSize of the RPipe Descriptor.
Verify if the Transfer Results are recieved in incrementing order of TransferSegment number (WA Transfer Result bTransferSegment field Bit6:0 must be incrementing over the received WA Transfer Result Packets)
Verify if no more Transfer Results and data segments are received with the dwTransferID after a WA Transfer Result with bTransferSegment field Bit7 set to one is received. (wait for 5 sec to determine)

Repetitions
Repeat for multiple Transfer Request sizes (Transfer Request with incrementing number of segments from 1 till 128)
And repeat for the test device endpoint configured to all allowed Bulk max packet sizes:
- FS: 8, 16, 32 and 64 bytes
- HS: 512 bytes

And repeat for FS and HS test device
And repeat for all supported PHY Data Rates
And repeat for test device on every downstream port of the DWA
And repeat for all RPipes that support Bulk

Test fails if
Any of the described verifications fail.

TD.1.38 DWA Interrupt OUT Transfer Requests and Transfer Result test
This test verifies if the DWA correctly handles various Interrupt OUT Transfer Requests in terms of producing transfer completion notifications, reading the transfer requests and producing the transfer results.

Device States for Test
This test is run with the DWA in Configured state, with the RPipe in Idle state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.
Overview of Test Steps
The test software performs the following steps.
Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
Place the DWA in Configured state.
Prompt the user to connect a full speed test device to port 1 of the DWA.
Place an RPipe in Idle state and set up for communication to a test device interrupt OUT endpoint.
Configure the test device for an interrupt endpoint with a perfect sink for data.
- MaxPacketSize of the interrupt endpoint is initially set to 8 bytes.
- Interval of the interrupt endpoint is initially set to 1.
Schedule an Interrupt OUT Transfer Requests to the DWA RPipe to transfer interrupt data to the
test device endpoint with the following characteristics all Transfer Request segments:
- bLength set to 10H
- bRequestType set to 81H
- wRPipe set to the index of the RPipe under test
- dwTransferID set to a unique ID
- dwTransferLength set to amount of data following: (N - #segments transferred) x RPipe
  wMaxPacketSize
- bTransferSegment bit 6:0 must be set to the Transfer Segment and bit 7 must be set to 1 if
  this is the last Transfer Request segment.
- bReserved set to zero
- wReserved set to zero
Verify if the DWA sends back a Transfer Complete Notification on the Notification Endpoint after
every Transfer Request Segement.
Check the Transfer Complete Notification fields:
- bLength set to 4 decimal.
- bNotifyType set to 93H.
- bEndpoint set to the Data Transfer Read Endpoint.
- bReserved set to zero.
Verify if the DWA sends back a Transfer Result after every Transfer Segment on the Data Transfer
Read Endpoint.
Check the Transfer Result fields for every segment:
- bLength set to 10H.
- bResultType set to 83H.
- dwTransferID must be set to dwTransferID of the Transfer Request
- dwTransferLength must be set to the amount of data transferred: #segments transferred x
  wMaxPacketSize.
- bTransferSegment bit 6:0 must be set to the Transfer Segment and bit 7 must be set to 1 if
  this is the last Transfer Request segment.
- bTransferStatus bit 5:0 must be set to 0 Decimal (TRANSFER_STATUS_SUCCESS) and bit
  6:7 must be set to set to 0 as well.
- dwNumOfPackets must be set to zero.
Verify if the Transfer Results are recieved in incrementing order of TransferSegment number (WA
Transfer Result bTransferSegment field Bit8:0 must be incrementing over the received WA
Transfer Result Packets)
Verify if no more Transfer Results are received with the dwTransferID after a WA Transfer Result
with bTransferSegment field Bit7 set to one is received. (wait for 5 sec to determine)
Repetitions
Repeat for multiple Transfer Request sizes
And repeat for the test test device endpoint configured to a sample of allowed Interrupt max packet
sizes:
- LS: a number of data payloads from 0 to 8 bytes
- FS: a number of data payloads from 0 to 64 bytes
- HS: a number of data payloads from 0 to 1024 bytes with the number of transactions per
  microframe set to 1, 2 and 3
And repeat with the test device endpoint set to a number of bInterval settings:
- for LS and FS: 1, 10, 50, 100, 200, 255
- for HS: 1, 2, 3, 7, 8, 9, 14, 15, 16
And repeat for LS, FS and HS test device
And repeat for all supported PHY Data Rates
And repeat for every upstream Isochronous Streaming OUT Endpoint
And repeat for test device on every downstream port of the DWA
And repeat for all RPipes that support Interrupt
Test fails if
Any of the described verifications fail.

TD.1.39 DWA Interrupt IN Transfer Requests and Transfer Result test

This test verifies if the DWA correctly handles various Interrupt IN Transfer Requests in terms of producing transfer completion notifications, reading the transfer requests and producing the transfer results.

Device States for Test
This test is run with the DWA in Configured state, with the RPipe in Idle state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
Place the DWA in Configured state.
Prompt the user to connect a full speed test device to port 1 of the DWA.
Place an RPipe in Idle state and set up for communication to a test device interrupt IN endpoint.
Configure the test device for an interrupt endpoint with a perfect source for data.
- MaxPacketSize of the interrupt endpoint is initially set to 8 bytes.
- Interval of the interrupt endpoint is initially set to 1.
Schedule an Interrupt IN Transfer Requests to the DWA RPipe to transfer interrupt data to the test device endpoint with the following characteristics all Transfer Request segments:
  - bLength set to 10H
  - bRequestType set to 81H
  - wRPipe set to the index of the RPipe under test
  - dwTransferID set to a unique ID
  - dwTransferLength set to amount of data following: (N - #segments transferred) x RPipe wMaxPacketSize
  - bTransferSegment bit 6:0 must be set to the Transfer Segment and bit 7 must be set to 1 if this is the last Transfer Request segment.
  - bReserved set to zero
  - wReserved set to zero
Verify if the DWA sends back a Transfer Complete Notification on the Notification Endpoint after every Transfer Request Segment.
Check the Transfer Complete Notification fields:
  - bLength set to 4 decimal.
  - bNotifyType set to 93H.
  - bEndpoint set to the Data Transfer Read Endpoint.
  - bReserved set to zero.
Verify if the DWA sends back a Transfer Result after every Transfer Segment on the Data Transfer Read Endpoint.
Check the Transfer Result fields for every segment:
  - bLength set to 10H.
  - bResultType set to 83H.
  - dwTransferID must be set to dwTransferID of the Transfer Request
  - dwTransferLength must be set to to the amount of data transferred: #segments transferred x wMaxPacketSize
  - bTransferSegment bit 6:0 must be set to the Transfer Segment and bit 7 must be set to 1 if this is the last Transfer Request segment.
  - bTransferStatus bit 5:0 must be set to 0 Decimal (TRANSFER_STATUS_SUCCESS) and bit 6:7 must be set to set to 0 as well.
  - dwNumOfPackets must be set to zero.
Verify if the DWA provides data on the Data Transfer Read Endpoint as the consecutive transfer after a Transfer Result.
Verify if the data of all the segments have the size of wMaxpacketSize of the RPipe Descriptor.
Verify if the Transfer Results are received in incrementing order of TransferSegment number (WA Transfer Result bTransferSegment field Bit6:0 must be incrementing over the received WA Transfer Result Packets)
Verify if no more Transfer Results and data segments are received with the dwTransferID after a WA Transfer Result with bTransferSegment field Bit7 set to one is received. (wait for 5 sec to determine)

Repetitions
Repeat for multiple Transfer Request sizes
And repeat for the test test device endpoint configured to a sample of allowed Interrupt max packet sizes:
- LS: a number of data payloads from 0 to 8 bytes
- FS: a number of data payloads from 0 to 64 bytes
- HS: a number of data payloads from 0 to 1024 bytes with the number of transactions per microframe set to 1, 2 and 3
And repeat with the test device endpoint set to a number of bInterval settings:
- for LS and FS: 1, 10, 50, 100, 200, 255
- for HS: 1, 2, 3, 7, 8, 9, 14, 15, 16
And repeat for LS, FS and HS test device
And repeat for all supported PHY Data Rates
And repeat for every upstream Isochronous Streaming OUT Endpoint
And repeat for test device on every downstream port of the DWA
And repeat for all RPipes that support Interrupt
Test fails if
Any of the described verifications fail.

TD.1.40 Bulk Maximum Transfer Request Size Loopback test
This test functionally validates if the DWA can correctly handle the maximum defined size of $2^{32} - 1$.

Device States for Test
This test is run with the DWA in Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
- Perform the Bulk loopback test (TD.1.21) with the following changes:
  - Send a multi-segment Transfer Request to the Bulk OUT RPipe of total size $2^{32} - 1$.

Repetitions
Repeat test for FS and HS test device
Test fails if
Any of the verifications in the parent test fails. (TD.1.21)

TD.1.41 Short Packet test
This test functionally validates if the DWA correctly recognizes a short packet on the downstream side in IN direction as the end of the IN Transfer Request.

Device States for Test
This test is run with the DWA in Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
- Perform the Bulk loopback test (TD.1.21) with the following changes:
  - It is important that the bulk IN and OUT loopback endpoints are set to the same MaxPacketSize.
  - Send a one segment long Bulk OUT Transfer Request to the Bulk OUT RPipe with a short packet.
  - Send a one segment long Bulk IN Transfer Request to the Bulk IN RPipe with the following characteristics:
    - Do NOT set the last segment flag in the bTransferSegment field.
- Verify if the DWA provides a Transfer Complete Notification to the host with 5 seconds after the start of the test
- Verify if the DWA provides Transfer Result with the following settings:
  - bTransferSegment bit 6:0 must be set to the Transfer Segment and bit 7 must be set to 1.
  - bTransferStatus bit 5:0 must be set to 0 Decimal (TRANSFER_STATUS_SUCCESS) and bit 6:7 must be set to set to 0 as well.
- Verify if the DWA provides the short packet data to the host in the current transaction.

**Repetitions**
Repeat test for FS and HS test device
Repeat for various locations of short packet across all possible packets in segment

**Test fails if**
Any of the verifications in the parent test fails. (TD.1.21)
Any of the described verifications fail.

**TD.1.42 TRANSFER_STATUS_HALTED test**
This test verifies if the DWA correctly detects the TRANSFER_STATUS_HALTED error case and correctly reports it back to the host.

**Device States for Test**
This test is run with the DWA in Configured state, with the RPipe in Idle state.

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.
- Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
- Place the DWA in Configured state.
- Prompt the user to connect a FS test device to port 1 of the DWA.
- Configure an endpoint of the test device for STALL Handshaking.
- Initialize an RPipe for the appropriate communication to the endpoint of the test device. (refer to RPipe Initialization routine)
- Send a maximum segment Transfer Request to the RPipe.
- Verify that the DWA ACK's the Transfer Request.
- Verify if the DWA sends back a Transfer Complete Notification on the Notification Endpoint for every Transfer Request Segment.
- Check the Transfer Complete Notification fields:
  - bEndpoint set to the Data Transfer Read Endpoint.
- Verify if the DWA sends back a Transfer Result after every Transfer Segment on the Data Transfer Read Endpoint.
- Check the Transfer Result fields for every Transfer Result:
  - dwTransferID must be set to dwTransferID of the Transfer Request
  - dwTransferLength must be set to the amount of data transferred.
  - bTransferSegment bit 6:0 must be set to the Transfer Segment and bit 7 must be set to 1 if this is the last Transfer Request segment.
- bTransferStatus bit 5:0 must be set to 1 Decimal (TRANSFER_STATUS_HALTED), bit 6 must be set to 0 and bit 7 must be set to 1.

**Repetitions**
Repeat the test for IN and OUT directions (This means a re-configuration of the test device endpoint and the RPipe under test)
And repeat the test for Bulk, Control and Interrupt traffic type (This means a re-configuration of the test device endpoint and the RPipe under test)
And repeat for all RPipes

**Test fails if**
Any of the described verifications fail.

**TD.1.43 TRANSFER_STATUS_BABBLE test**
This test verifies if the DWA correctly detects the TRANSFER_STATUS_BABBLE case and correctly reports it back to the host.
**Device States for Test**
This test is run with the DWA in Configured state, with the RPipe in Idle state.

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.
- Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
- Place the DWA in Configured state.
- Prompt the user to connect a test device to port 1 of the DWA.
- Configure the test device for an endpoint.
- Initialize an RPipe to the endpoint of the test device. (refer to RPipe Initialization routine)
  - Set the RPipe Max packet to a smaller value than real value (this will make the test device appear to be babbling)
- Send a Transfer Request to the RPipe.
- Verify if the DWA sends back a Transfer Complete Notification on the Notification Endpoint for every Transfer Request Segment.
- Verify if the DWA sends back a Transfer Result after every Transfer Segment on the Data Transfer Read Endpoint.
- Check the Transfer Result fields:
  - bTransferStatus bit 5:0 must be set to 3 Decimal (TRANSFER_STATUS_BABBLE), bit 6 must be set to 0 and bit 7 must be set to 1.

**Repetitions**
Repeat for all RPipes
Repeat for testing for frame babble

**Test fails if**
Any of the described verifications fail.

**TD.1.44 TRANSFER_STATUS_TRANSACTION_ERROR with limited bmRetryOptions test**
This test verifies if the DWA correctly detects and reports the TRANSFER_STATUS_TRANSACTION_ERROR case for RPipes with limited downstream transaction retries.

**Device States for Test**
This test is run with the DWA in Configured state, with the RPipe in Idle state.

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.
- Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
- Place the DWA in Configured state.
- Prompt the user to connect a test device to port 1 of the DWA.
- Configure the test device for a Bulk OUT Endpoint with "Always CRCERROR".
- Set up an RPipe in Idle state for communication to the endpoint of the test device. (refer to RPipe Initialization routine)
  - Set bmRetryOptions to 3. (Downstream transaction will be retried 3 times before being failed.)
- Send a Transfer Request to the RPipe.
- Verify if the DWA sends back a Transfer Complete Notification on the Notification Endpoint after every Transfer Request Segement.
- Verify if the DWA sends back a Transfer Result after every Transfer Segment on the Data Transfer Read Endpoint.
- Verify if the DWA correctly completes the Transfer Request by checking the TransferStatus fields for every Transfer Result:
  - bTransferStatus bit 5:0 must be set to 7 Decimal (TRANSFER_STATUS_TRANSACTION_ERROR) and bit 7:6 must be set to set to 10B.
- Send a Get RPipe Descriptor to the RPipe
- Verify if wNumTransactionErrors is set to 3.

**Repetitions**
Repeat for various bmRetryOptions settings (various values except 0)
And repeat for Control and Interrupt
And repeat for all RPipes
And repeat with the test device endpoint configured to timeout all transactions.
And repeat with the test device endpoint configured to produce bad PID’s for all transactions
Test fails if
Any of the described verifications fail.

**TD.1.45 TRANSFER_STATUS_TRANSACTION_ERROR with unlimited bmRetryOptions test**

This test verifies if the DWA correctly detects and reports the TRANSFER_STATUS_TRANSACTION_ERROR case for RPipes with unlimited downstream transaction retries.

**Device States for Test**
This test is run with the DWA in Configured state, with the RPipe in Idle state.

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.

- Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
- Place the DWA in Configured state.
- Prompt the user to connect a test device to port 1 of the DWA.
- Configure the test device for a Bulk OUT Endpoint with “Always CRCERROR.”
- Set up an RPipe in Idle state for communication to the endpoint of the test device. (refer to RPipe Initialization routine)
  - Set bmRetryOptions to 0. (Downstream transaction will be retried an unlimited time.)
- Send a Transfer Request to the RPipe.
- Verify if the DWA does not send a TransferComplete Notification.
- Wait for 2 seconds and then reconfigure the test device endpoint to disable the “Always CRCERROR.”
- Verify if the DWA sends back a Transfer Complete Notification on the Notification Endpoint.
- Verify if the DWA sends back a Transfer Result on the Data Transfer Read Endpoint.
- Verify the TransferStatus fields for every Transfer Result:
  - bTransferStatus bit 5:0 must be set to 7 Decimal
  - (TRANSFER_STATUS_TRANSACTION_ERROR) and bit 7:6 must be set to set to 01B.
- Send a Get RPipe Descriptor to the RPipe
- Verify if wNumTransactionErrors is set to 0.

**Repetitions**
Repeat the test for IN and OUT directions (This means a re-configuration of the testdevice endpoint and the RPipe under test)
And repeat for all RPipes
Repeat for Control and Interrupt
And repeat with the test device endpoint configured to timeout all transactions.
And repeat with the test device endpoint configured to produce bad PID’s for all transactions
Test fails if
Any of the described verifications fail.

**TD.1.46 TRANSFER_STATUS_ABORTED test**

This test verifies if the DWA correctly detects the TRANSFER_STATUS_ABORTED case and correctly reports it back to the host.

**Device States for Test**
This test is run with the DWA in Configured state, with the RPipe in Idle state.

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.

- Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
- Place the DWA in Configured state.
Prompt the user to connect a test device to port 1 of the DWA.
Configure the test device for a Bulk OUT Endpoint.
Initialize an RPipe to the endpoint of the test device. (refer to RPipe Initialization routine)
  - Set bmRetryOptions to 3. (Downstream transaction will be retried 3 times before being failed.)
Send a wRequest of concurrent Transfer Requests to the RPipe.
Send an Abort Transfer Request to the RPipe to abort one of the queued transfer requests.
  (immediately after the Transfer Requests)
    - bLength – set to 08H.
    - bRequestType – set to 84H
    - wRPipe – set to RPipe Index.
    - dwTransferID – set to the transfer ID of the Transfer Request that was issued
Verify if the DWA ACK’s the Abort Transfer Request.
Verify if the DWA sends back a Transfer Complete Notification on the Notification Endpoint after every Transfer Request Segment.
Verify if the DWA sends back a Transfer Result after every Transfer Segment on the Data Transfer Read Endpoint.
Verify the TransferStatus fields for every Transfer Result:
  - bTransferStatus bit 5:0 must be set to 8 Decimal
    (TRANSFER_STATUS_STATS_ABORTED), bit 6 must be set to 0 and bit 7 must be set to 1.
  Verify if the DWA successfully completes the other transfer requests that have not been aborted.

**Repetitions**
Repeat the test with aborting all of the wRequest of concurrent transfer requests instead of aborting only one.
And repeat the test for IN and OUT directions (This means a re-configuration of the testdevice endpoint and the RPipe under test)
And repeat the test for Control and Interrupt traffic type (This means a re-configuration of the testdevice endpoint and the RPipe under test)
And repeat for all RPipes

**Test fails if**
Any of the described verifications fail.
Resume and Reset tests

TD.1.47 Reset test
This test verifies if a DWA correctly handles a SetAddress (zero) – Reset and a ResetDevice_IE – Reset by transferring into the Default State and resetting RPipes and Port Status.

Device States for Test
This test is run with the DWA in Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps:
- Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
- Place the DWA in Configured state.
- Prompt the user to connect a test device to port 1 of the DWA.
- Configure the test device for a Bulk loopback.
- Set up an RPipe pair in Idle state for communication to the endpoint of the test device. (refer to RPipe Initialization routine)
- Send a sufficiently large Transfer Request (eg. size 128k) to the Bulk OUT RPipe to be certain that the Transfer Request will still be active when the reset will happen.
- Reset the DWA
- Send a GetDeviceDescriptor request to the device with address zero.
- Verify if the DWA is in the Default State by verifying if the DWA responds with the DWA Device Descriptor.
- Send a GetRPipeDescriptor to all RPipes.
- Verify if all values have been set to default.
- Verify if RPipe Descriptor value for wBlocks is set to zero if the RPipe support dynamically changeable buffer
- Send a GetPortStatus request to all the DWA downstream ports.
- Verify if the DWA downstream port are in disabled state.
- Place the DWA in Configured state.
- Verify if the DWA sends back a Port Status Change Notification when its notification endpoint is polled.
  - bLength field must be set to 3 decimal.
  - bNotifyType field must be set to 92H.
  - bPortIndex field must be set the Index of the Port on which the test device is attached.
- The host issues a Get Port Status command to the DWA and verifies that an attached device is reported.
- Verify if Bit 9 and 10 accurately indicate the speed of the attached device.
- Perform loopback on the DWA Rpipes under test and verify if loopback works.

Repetitions
Repeat with the RPipe to all possible number of blocks if the number of buffer blocks per RPipe is dynamically manageable by host software
Repeat the test for resetting the DWA by means of sending a SetAddress (zero) and a ResetDevice_IE.
Repeat for all traffic types.
Repeat for all DWA Rpipes.

Test fails if
Any of the described verifications fail.
**Isochronous tests**

**TD.1.48 Isochronous Streaming Downstream IN Token generation test**

This test verifies if a DWA does not begin generating IN Tokens to the downstream device until the host begins polling for data on the associated DWA isochronous data stream function endpoint.

**Device States for Test**

This test is run with the DWA in Configured state.

**Channels for Test**

Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**

The test software performs the following steps:

1. Place the host and the DWA in optimal conditions for wireless communication. (in each other’s line of sight and very close to each other)
2. Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
3. Place the DWA in Configured state.
4. Prompt the user to connect a full speed test device to port 1 of the DWA.
5. Place one RPipe in Idle state and set up for communication to a test device isochronous OUT endpoint.
6. Place a second RPipe in Idle state and set up for communication to a test device isochronous IN endpoint.
7. Configure the test device for an isochronous loopback loopback endpoint pair.
8. Isochronous OUT endpoint of the test device will be varied:
   - MaxPacketSize of the isochronous test device endpoint is initially set to 1 byte
   - bInterval of 1.
9. Isochronous IN endpoint of the test device will be varied:
   - MaxPacketSize of the isochronous test device endpoint is initially set to 1 byte
   - bInterval of 1.
   - the test device endpoint is placed in a special isochronous IN mode as part of an isochronous loopback pair where the IN endpoint appends the current SOF values to the interrupt data packets received via the OUT endpoint
10. Start Isochronous IN Stream to the DWA
11. Verify if the DWA Iso IN endpoint NAKs the WDTCTA transaction token from the host asking for Isochronous IN data
12. Verify if no data is provided to the DWA
13. Start Isochronous OUT Stream to the DWA concurrently with the Isochronous IN Stream
14. Verify if the first isochronous data packet that was provided via the Isochronous OUT Stream is provided back to the host.
15. Verify if the OUT data matches the IN data. (the same data and in the same order)
16. Verify if the DWA did not start polling the test device endpoint before the first upstream IN transaction
   - Verify by means of the appended SOF information in the isochronous data
   - Or get NAK count from test device

**Repetitions**

And repeat for all Upstream Isochronous Streaming OUT and IN Endpoints of the DWA
And repeat for test device on every downstream port of the DWA
And repeat for all R Pipes that support isochronous

**Test fails if**

Any of the described verifications fail.

---

**TD.1.49 Isochronous IN Data Split test**

This test verifies if if the DWA does not split wired isochronous IN packet data from a single (micro)frame across multiple over-the-air packets.

**Device States for Test**

This test is run with the DWA in Configured state.
Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
- Place the host and the DWA in optimal conditions for wireless communication. (in eachother's line of sight and very close to eachother)
- Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
- Place the DWA in Configured state.
- Prompt the user to connect a full speed test device to port 1 of the DWA.
- Configure the test device for an interrupt IN endpoint in a special mode.
  - MaxPacketSize of the isochronous test device endpoint is initially set to 1 byte
  - bInterval of 1.
  - the test device endpoint is placed in a special isochronous IN source mode where the test device places the current SOF values in the interrupt data packet
- Place an RPipe in Idle state and set up for communication to a test device isochronous IN endpoint.
- Start Isochronous IN Stream to the DWA
- Verify if the DWA does not split wired isochronous IN packet data from a single (micro)frame across multiple over-the-air packets.
  - Analyze the isochronous data packets containing the SOF values

Test fails if
Any of the described verifications fail.

TD.1.50 Isochronous IN buffering test
This test verifies if the DWA responds with the oldest data in its Rpipe buffer for each Wireless USB Isochronous IN request.

Device States for Test
This test is run with the DWA in Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
- Place the host and the DWA in optimal conditions for wireless communication. (in eachother's line of sight and very close to eachother)
- Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
- Place the DWA in Configured state.
- Prompt the user to connect a full speed test device to port 1 of the DWA.
- Configure the test device for an interrupt IN endpoint in a special mode.
  - MaxPacketSize of the isochronous test device endpoint is initially set to 1 byte
  - bInterval of 1.
  - the test device endpoint is placed in a special isochronous IN source mode where the test device places the current SOF values in the interrupt data packet
- Place an RPipe in Idle state and set up for communication to a test device isochronous IN endpoint.
- Send a first wireless IN transaction to the DWA
- Wait
- Continue sending wireless IN transactions to the DWA
- Verify if the DWA responds with the oldest data in its Rpipe buffer for each Wireless USB Isochronous IN request.
  - Analyze the isochronous data packets containing the SOF values

Repetitions
Repeat for various delays between the first wireless IN transaction and the second wireless IN transaction to the DWA representing half full, nearly full and overflowed buffers on the DWA
Repeat for all MaxPacketSizes
And repeat for the test device endpoint configured to all Isochronous packet sizes and number of transactions per microframe.
And repeat with the test device endpoint set to all bInterval settings
And repeat for all bOverTheAirInterval Settings
And repeat for FS and HS test device
And repeat for all RPipe number of blocks (if configurable)
And repeat for all allowed burst sizes up to bMaxBust of the DWA Data Transfer Endpoints
And repeat for all supported PHY Data Rates for isochronous streaming
And repeat for all Upstream Isochronous Streaming OUT and IN Endpoints of the DWA
And repeat for test device on every downstream port of the DWA
   And repeat for all R Pipes that support isochronous

**Test fails if**
Any of the described verifications fail.

---

**TD.1.51 Downstream SOF Framenumber test**

This test functionally validates if the FrameNumber value of the SOFs on the downstream wired bus match bits 13:3 of the 17 bits long 1/8th millisecond value field in the timestamp of any MMCs transmitted during a 1/8th millisecond period and if Bits 2:0 of the 17 bits long 1/8th millisecond value field in the MMC timestamp reflect the SOF instance (microframe) on the downstream bus

**Device States for Test**
This test is run with the DWA in Configured state.

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.
   - Place the host and the DWA in optimal conditions for wireless communication. (in each others line of sight and very close to each other)
   - Initially no downstream devices are attached to the DWA, unless they are non-removable devices.
   - Place the DWA in Configured state.
   - Prompt the user to connect a full speed test device to port 1 of the DWA.
   - Configure the test device for an interrupt IN endpoint in a special mode.
      - MaxPacketSize of the isochronous test device endpoint is initially set to 1 byte
      - bInterval of 1.
      - the test device endpoint is placed in a special isochronous IN source mode where the test device places the current SOF values in the interrupt data packet
   - Place an RPipe in Idle state and set up for communication to a test device isochronous IN endpoint.
   - Start Isochronous IN Stream to the DWA
   - Verify if the DWA correctly synchronizes its downstream bus with the upstream wireless USB channel.
      - Analyze the isochronous data packets containing the SOF values
      - Compare the SOF values to the MMC time for the first IN
      - verify if the FrameNumber value of the SOFs on the downstream wired bus matches bits 13:3 of the 17 bits long 1/8th millisecond value field in the timestamp of any MMCs transmitted during a 1/8th millisecond period.
      - Verify if and if Bits 2:0 of the 17 bits long 1/8th millisecond value field in the MMC timestamp reflect the SOF instance (microframe) on the downstream bus

**Test fails if**
Any of the verifications fail.
Mixed traffic tests

TD.1.52 Isochronous Simultaneous Streaming test
This test verifies if the DWA supports the number of simultaneous isochronous streams equal to the number of upstream isochronous streaming endpoints on the DWA.

Device States for Test
This test is run with the DWA in Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
- Perform the Isochronous streaming loopback test (TD.1.23) for every pair of upstream isochronous IN and OUT streaming endpoints

Repetitions
In case of uneven IN/OUT endpoints on DWA. Repeat the test using different endpoints outside of the paired endpoint so that each endpoint has been tested in an iso IN/OUT pair.

Test fails if
Any of the verifications in the parent test fails. (TD.1.23)

TD.1.53 Mixed traffic test
This test verifies if the DWA RPipes can be overwritten to target endpoints of various specifications (type, size, interval, ...).

Device States for Test
This test is run with the DWA in Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.
- Perform a mix of instances (one instance is a certain maxPacketSize, bInterval, wBlocks value, downstream port, ..) of the following loopback tests so that all DWA RPipes are in use:
  - TD.1.20 Control Loopback test
  - TD.1.21 Bulk Loopback test
  - TD.1.22 Interrupt Loopback test
  - TD.1.23 Isochronous Streaming Loopback test
- Reconfigure the RPipes to accommodate a different mix of instances of the above loopback test. (one Rpipe will run another transfer type, with other maxPacketsizes, possibly different wBlocks allocation and different downstream port)

Test fails if
Any of the verifications in the parent test fail. (TD.1.20, TD.1.21, TD.1.22, TD.1.23)
**Request errors tests**

**TD.1.54 Set Wire Adapter Feature Request error test**
This test verifies if the DWA properly handles a Set Wire Adapter Feature Request that is incorrectly specified.

**Device States for Test**
This test is run with the DWA in Configured state, with the RPipe in Idle state.

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.
- Place the DWA in Configured state.
- Send a Set Wire Adapter Feature request to the DWA.
  - wValue – set to an illegal value decimal.
  - wIndex – lower byte set to the target interface number.
  - wLength – set to Zero.
- Verify that the DWA responds with a STALL.
- Verify that the DWA responds with a STALL.
- Re-enumerate the DWA. If this fails then the test aborts.
- Repeat for any illegal wValue value.

**Test fails if**
Any of the described verifications fail.

**TD.1.55 Clear Wire Adapter Feature Request error test**
This test verifies if the DWA properly handles a Clear Wire Adapter Feature Request that is incorrectly specified.

**Assertions Used in Test**
8.3.1.3#2

**Device States for Test**
This test is run with the DWA in Configured state, with the RPipe in Idle state.

**Channels for Test**
Test will be run on one FFI channel and one TFI channel.

**Overview of Test Steps**
The test software performs the following steps.
- Place the DWA in Configured state.
- Send a Clear Wire Adapter Feature request to the DWA.
  - wValue – set to an illegal value decimal.
  - wIndex – lower byte set to the target interface number.
  - wLength – set to Zero.
- Verify that the DWA responds with a STALL.
- Repeat for various illegal values.
- Re-enumerate the DWA. If this fails then the test aborts.

**Test fails if**
Any of the described verifications fail.

**TD.1.56 Clear RPipe Feature Request error test**
This test verifies if the DWA properly handles a Clear RPipe Feature request that is incorrectly specified.

**Device States for Test**
This test is run with the DWA in Configured state, with the RPipe in Idle state.
Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.

Initially no downstream devices are attached to the DWA, unless they are non-removable devices (that are part of the DWA).
Place the DWA in Configured state.
Initialize an RPipe (refer to RPipe Initialization routine)
Send a Clear RPipe Feature (RPipe Index) request to the RPipe.
  - wValue – set to an illegal value.
  - wIndex – set to RPipe Index.
  - wLength – set to Zero.
Verify that the DWA responds with a STALL.
Repeat for all illegal values.
Re-enumerate the DWA. If this fails then the test aborts.

Test fails if
Any of the described verifications fail.

TD.1.57 Set RPipe Feature Request error test
This test verifies if the DWA properly handles a Set RPipe Feature request that is incorrectly specified.

Assertions Used in Test
8.3.1.8#2

Device States for Test
This test is run with the DWA in Configured state, with the RPipe in Idle state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the following steps.

Initially no downstream devices are attached to the DWA, unless they are non-removable devices (that are part of the DWA).
Place the DWA in Configured state.
Initialize an RPipe (refer to RPipe Initialization routine)
Send Set RPipe Feature (wValue, wIndex, wLength, Data) request with the following values:
  - wValue – set to an illegal value.
  - wIndex – set to RPipe Index.
  - wLength – set to Zero.
Verify that the DWA responds with a STALL.
Repeat for any illegal value.
Re-enumerate the DWA. If this fails then the test aborts.

Test fails if
Any of the described verifications fail.

TD.1.63 Standard Loopback Tests With Abort
This test repeats the standard loopback tests for each transfer type.

Device States for Test
This test is run with the DWA in Configured state.

Channels for Test
Test will be run on one FFI channel and one TFI channel.

Overview of Test Steps
The test software performs the standard loopback tests with the following change: Intermittently, when the test software receives a successful transfer completion notification, the test software sends a command to abort the transfer. The DWA must respond with a transfer not found error code. Subsequent loopback transfers must still function as expected.

Test fails if
Any of the described verifications fail.