# Media Agnostic Universal Serial Bus

# Test Specification

# Revision 1.1

Date: April 14, 2016

Revision: 1.1

Copyright © 2015, USB Implementers Forum, Inc.

All rights reserved.

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

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

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

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

**Revision History**

| Revision | Issue Date | Comments |
|---|---|---|
| 1.0 | | Initial Version |
| 1.1 | | Updated for MA USB 1.0b |
| | | |

**Significant Contributors:**

| | |
|---|---|
| Chris Kelly | Atmel Corporation |
| Ashwini Ananthateerta | Broadcom Corp. |
| Hui (Winnie) Xu | Broadcom Corp. |
| Dan Ellis | DisplayLink (UK) Ltd. |
| Andy Saul | DisplayLink (UK) Ltd. |
| Martin Turnock | DisplayLink (UK) Ltd. |
| Mario Pasquali | Ellisys |
| Chuck Trefts | Ellisys |
| Abdul Rahman Ismail | Intel Corporation |
| Wojciech Omilian | Intel Corporation |
| Bahareh Sadeghi | Intel Corporation |
| Sean Stalley | Intel Corporation |
| Stephanie Wallick (Editor) | Intel Corporation |
| Mu-Huan Chiang | Qualcomm, Inc. |
| Kitty Ho | Qualcomm, Inc. |
| Hemant Kumar | Qualcomm, Inc. |
| Jacob Anderson | Specwerkz, LLC |
| Diane Lenox | Specwerkz, LLC |
| Soren Petersen | Specwerkz, LLC |
| Yuichi Mizoguchi | Toshiba Corporation |

***Please send comments via electronic mail to: techadmin@usb.org***

# Contents

# 1 Introduction

This Document provides the compliance criteria and test descriptions for USB Hosts and Devices that conform to the Media Agnostic Universal Serial Bus (MA USB) Specification, Release 1.0. The document is divided into three major sections. The first section lists the compliance criteria. The second and third sections list the test descriptions used to verify an MA USB Host's and an MA USB Device's respective conformance to these compliance criteria.

Compliance criteria are provided as a list of assertions that describe specific characteristics or behaviors that must be met. Each assertion provides a reference to the MA USB Specification or other documents from which the assertion was derived. In addition, each assertion provides a reference to the specific test description(s) where the assertion is tested.

Test descriptions provide a high-level overview of the tests that are performed to check the compliance criteria. The test descriptions are provided with enough detail so that a reader can understand what the test does. The descriptions do not describe the actual step-by-step procedure to perform the test.

Each test assertion is formatted as follows:

| Assertion # | Assertion Description | Test # | Comments |
|---|---|---|---|

**Assertion #:** Unique identifier for each requirement. The identifier is in the form X.y where X is the section corresponding to the assertion and y is the subsection corresponding the assertion.

**Assertion Description:** Specific requirement from the MA USB specification.

**Test #:** The label for a specific test description in this specification that tests this requirement. Test # can have one of the following values:

- **GTP:** This item is covered by the General Test Procedure.
- **Interop:** This item is tested as part of interoperability testing.
- **N/A:** This item is not explicitly tested in an MA USB test description. Items labeled as N/A include items that are not testable, not important to test for interoperability, or are indirectly tested by other compliance testing operations.
- **TBD:** A test description is not yet written for this assertion. Assertion will be addressed in future revisions of this test specification.

**Comments:** Provides additional information on requirement

# 2 Assertions

Unless otherwise noted, subsection references are to the MA USB Specification.

## *Chapter 5 Test Assertions*

| Assertion # | Assertion Description | Test # | Comments |
|---|---|---|---|
| **Subsection reference: 5 Data flow model** | | | |
| **Subsection reference: 5.1 Communication flow** | | | |
| **Subsection reference: 5.2 Protocol overview** | | | |
| **Subsection reference: 5.2.1 Packet exchange** | | | |
| **Subsection reference: 5.2.1.1 Management packet exchange** | | | |
| 5.2.1.1#1 | MA USB devices are managed through MA USB packets of type 0 (management packet). | Interop | |
| 5.2.1.1#2 | All management packets shall be transmitted over the management channel. | Interop | |
| 5.2.1.1#3 | Unless otherwise specified, a responding PAL shall release the response packet to the management channel within aManagementResponseTime from the moment it receives the corresponding request packet. | TD 5.1<br>TD 6.1<br>TD 6.24<br>TD 6.25<br>TD 6.27 | |
| 5.2.1.1#4 | Unless otherwise specified, if a requesting PAL does not receive a response packet by aManagementRequestTimeout after it has successfully submitted the management request packet, it shall retransmit the request packet. | TD 5.2 | |
| 5.2.1.1#5 | A retransmitted management request packet shall keep all packet fields the same as those in the original request packet, except the Retry field shall be set to 1. | TD 5.2 | |
| 5.2.1.1#6 | The requesting PAL shall repeat transmitting a management packet for which it did not receive a timely response aManagementRetries times. | TD 5.2 | |
| 5.2.1.1#7 | If the Ping protocol fails after a retried management packets, or if the requesting PAL decides not to exercise the Ping protocol, the requesting PAL shall transition to the Session down state and inform the lower layers of the session state transition. | TD 5.2 | |
| 5.2.1.1#8 | Management packet retry count does not include possible local retries before the request packet is successfully release to the management channel. | N/A | |
| 5.2.1.1#9 | A responding PAL shall respond to a request packet that has the Retry field set to 1, even if it has responded to an earlier instance of the request packet. | TD 5.1<br>TD 6.2 | |

| Subsection reference: 5.2.1.2 Data packet exchange | | | |
|---|---|---|---|
| 5.2.1.2#1 | MA USB transfers are executed through MA USB packets of type 2 (Data type). | Interop | |
| 5.2.1.2#2 | All data packets are transmitted over one or more data channels. | Interop | |
| 5.2.1.2#3 | Unless otherwise specified, a responding PAL in an immediate response exchange shall release the response packet to the assigned data channel within aTransferResponseTime from the moment it receives the corresponding request packet over the assigned data channel. | TD 6.3<br>TD 6.9<br>TD 6.10 | |
| 5.2.1.2#4 | Unless otherwise specified, if a requesting PAL does not receive a response packet by aTransferRequestTimeout after it has successfully submitted the data packet, it shall retransmit the data packet. | TD 5.3<br>TD 6.13 | |
| 5.2.1.2#5 | A retransmitted data packet shall keep all packet fields the same as those in the original data packet, except the Retry field shall be set to 1. | TD 5.3 | |
| 5.2.1.2#6 | The requesting PAL shall repeat transmitting a data packet for which it did not receive a timely response aControlTransferRetries, aBulkTransferRetries or aInterruptTransferRetries times for control, bulk and interrupt transfers, respectively. | TD 5.3 | |
| 5.2.1.2#7 | If the corresponding response packet to a data packet is not received after the maximum number of retries, the requesting PAL shall start the Ping protocol unless the requesting PAL is an MA USB device PAL, in which case starting the Ping protocol is optional. | TD 5.3 | |
| 5.2.1.2#8 | If the Ping protocol fails after retry of a data packet, the requesting PAL shall report the appropriate USBDI error indicating the transfer failure to the application, move to the Session Down state, and inform the lower layers of the transition of the session to the Session Down state. | TD 5.3 | |
| 5.2.1.2#9 | A MA USB host and device PAL implementation shall report transfer failures to the application in a timely manner. | N/A | Not tested |
| 5.2.1.2#10 | A responding PAL shall respond to a data request packet that has the Retry field set to 1, even if it has responded to an earlier instance of the request packet. | TD 6.4 | |
| 5.2.1.2#11 | Unless otherwise specified, a transmitting PAL that is expected to keep a transfer alive shall release successive data packets to the assigned data channel no more than aTransferRepeatTime apart. | TD 5.6 | |
| 5.2.1.2#12 | Unless otherwise specified, a receiving PAL that expects a transfer to be kept alive shall start an inquiring action | TD 5.3 | |

| | | | |
|---|---|---|---|
| | such as releasing another transfer request packet to the assigned data channel if it does not receive a new data packet by aTransferKeepAlive after it receives the last data packet over the assigned data channel. | | |
| **Subsection reference: 5.2.2 Ping protocol** | | | |
| 5.2.2#1 | Any MA USB device that receives a PingReq packet with the Device Address field set to its device address or the broadcast address and the SSID field matching the MA USB device Service Set shall respond with a PingResp packet with the Device Address field set to the MA USB address of that device. | TD 6.6 | |
| 5.2.2#2 | A PingReq packet transmitted by the MA USB device shall carry the device MA USB address in the Device Address field | TD 6.6 | |
| 5.2.2#3 | An MA USB host receiving a PingReq packet with the SSID field matching its Service Set shall respond with a PingResp packet with the Device Address field set to the same value as the Device Address field in the PingReq packet. | TD 5.4 | |
| 5.2.2#4 | An MA USB device shall not transmit a PingReq packet with the Device Address field set to broadcast address. | TD 6.6 | |
| 5.2.2#5 | If a PingResp packet is not received within aManagementRequestTimeout after releasing the last PingReq packet retry to the management channel, the MA USB PAL transmitting the PingReq packet shall transition to the Session Down state and inform the lower layers of the session state transition. | TD 5.5 | |
| 5.2.2#6 | A trigger for start of the Ping protocol shall result in a PingReq packet only if there is not another PingReq packet pending a response. | TD 5.5 | |
| **Subsection reference: 5.2.3 Data transfer** | | | |
| 5.2.3#1 | The byte written to or read from a USB pipe are made available through one or more MA USB transfers, executed sequentially to preserve the byte stream order and integrity. | Interop | |
| 5.2.3#2 | All MA USB transfers are initiated by the MA USB host. | Interop | |
| 5.2.3#3 | An MA USB IN transfer delivers USB payload from a target USB endpoint to the pipe interface associated with that endpoint. | Interop | |
| 5.2.3#4 | An MA USB OUT transfer delivers USB payload from a pipe interface to the target USB endpoint that the pipe is associated with. | Interop | |
| 5.2.3#5 | An MA USB IN transfer is not complete until all requested USB payload has been placed in the receive buffer designated by the MA USB host PAL. | Interop | |

| | | | |
|---|---|---|---|
| 5.2.3#6 | An MA USB OUT transfer is not complete until all transmitted USB payload is successfully delivered to the target USB endpoint. | Interop | |
| 5.2.3#7 | Each MA USB transfer is identified by a Transfer Request ID or Request ID. | Interop | |
| 5.2.3#8 | All MA USB packets belonging to the same MA USB transfer carry the same Request ID. | TD 5.8 | |
| **Subsection reference: 5.3 Transfer models** | | | |
| 5.3#1 | The local buffer residing in the MA USB host is fully allocated to the transfer before the transfer is initiated. | Interop | |
| 5.3#2 | The remote buffer residing in a target MA USB device is managed by the MA USB device. | Interop | |
| 5.3#3 | Support of the p-managed transfer model is mandatory for all MA USB hosts and devices. | Interop | |
| **Subsection reference: 5.4 IN transfers** | | | |
| 5.4#1 | The MA USB host PAL initiates an MA USB IN transfer by transmitting a TransferReq packet to a target MA USB device. | Interop | |
| 5.4#2 | The MA USB host PAL assigns a Request ID to each transfer request and associated TransferReq packet. | Interop | |
| 5.4#3 | The Request ID for a TransferReq packet is reset to 0 at session initialization or upon any configuration event that returns the state of the endpoint or stream flow to the initial state. | TD 5.7 | |
| 5.4#4 | The Request ID for a TransferReq is incremented by 1 for each new transfer request. | TD 5.6 | |
| 5.4#5 | The Request ID for a TransferReq wraps around to 0 after reaching the maximum value of aMaxRequestID. | TD 5.6 | |
| 5.4#6 | Each TransferReq packet includes a Remaining Size field that carries the number of remaining bytes the MA USB host PAL expects to receive to complete the transfer. | Interop | |
| 5.4#7 | Each TransferReq packet includes a Sequence Number field that carries the sequence number value the MA USB host is expecting to receive next. | Interop | |
| 5.4#8 | The TransferReq Sequence Number is set to 0 upon any configuration event that returns the state of the target endpoint or stream flow to the initial state. | TD 5.7 | |
| 5.4#9 | In response to a TransferReq packet, the target MA USB device shall transmit one or more TransferReq packets to the MA USB host to transfer the USB payload from the target USB endpoint or stream in the strict order it was received from the target endpoint or stream. | Interop | |

| | | | |
|---|---|---|---|
| 5.4#10 | Each TransferResp packet carries the same EP Handle, Stream ID and Request ID as in the TransferReq packet that initiated the transfer. | GTP | |
| 5.4#11 | Each TransferResp packet carries a Sequence Number field which is set to 0 at session initialization or upon any configuration event that returns the state of the endpoint or stream flow to the initial state. | TD 6.7 | |
| 5.4#12 | The Sequence Number value in a TransferResp packet is incremented by 1 after each new TransferResp packet. | GTP | |
| 5.4#13 | The Sequence Number value in a TransferResp packet wraps around to 0 after reaching the maximum value of aMaxSequenceNumber. | N/A | Not tested |
| 5.4#14 | The MA USB device shall have no more than (aMaxSequenceNumber+1)/2 outstanding TransferResp packets. | N/A | Not tested |
| 5.4#15 | In the absence of an active IN transfer, and if the target endpoint or stream has USB data to transmit, a TransferReq packet and the first corresponding TransferResp packet form an immediate exchange with timings and behavior, except that a retried TransferReq packet may carry an updated Sequence Number value, in which case it is transmitted with the Retry field set to 0. | Interop | |
| 5.4#16 | The target MA USB device queues the received TransferReq packets in a request buffer in the increasing order of Request ID values for subsequent processing. | Interop | |
| 5.4#17 | Upon completion of an IN transfer, the target MA USB device shall immediately process the next TransferReq packet in its request buffer if it is not empty, except that a retried TransferReq packet may carry an updated Sequence Number value and is transmitted with the Retry bit set to 0. | N/A | Not tested |
| 5.4#18 | A retried TransferReq packet for an active transfer may not change the size of the transfer. | TD 5.3 | |
| 5.4#19 | The MA USB host shall not retry a TransferReq packet unless all previous transfer requests have been completed, and the MA USB host has not received a corresponding TransferResp packet for more than aTransferTimeout since the last time the MA USB host acknowledged the last TransferResp packet belonging to the immediately preceding MA USB transfer. | TD 5.3 | |
| 5.4#20 | The MA USB host shall have no more than (aMaxRequestID + 1)/2 outstanding TransferReq packets for each target IN endpoint or stream. | N/A | Not tested |
| 5.4#21 | The total number of outstanding transfers across all IN and OUT endpoints and streams shall not exceed the | N/A | Not tested |

| | | | |
|---|---|---|---|
| | number returned by the target MA USB driver in the Number of Outstanding Requests field in the CapResp packet. | | |
| 5.4#22 | A TransferReq packet queued by the target MA USB device is not acknowledged unless it is invalid (i.e. gap in the packet Request ID field), in which case the target MA USB device PAL shall discard the invalid TransferReq packet, and shall return a null TransferResp packet with no payload. | TD 6.8 | |
| 5.4#23 | A TransferResp packet in response to an invalid TransferReq packet shall be sent within aTransferResponseTime from the moment it received the invalid TransferReq packet. | TD 6.8 | |
| 5.4#24 | A TransferResp packet in response to an invalid TransferReq packet shall have the Request ID set to the next Request ID the MA USB device is expecting. | TD 6.8 | |
| 5.4#25 | A TransferResp packet in response to an invalid TransferReq packet shall have the Status Code field set to MISSING_REQUEST_ID if the received TransferReq packet is valid but shows a gap in its Request ID field. | TD 6.8 | |
| 5.4#26 | A TransferResp packet in response to an invalid TransferReq packet shall have the Status Code set to INVALID_REQUEST if the received TransferReq packet is invalid independent of the value of its Request ID field. | TD 6.8 | |
| 5.4#27 | In response to a TransferResp packet with the Status Code field set to MISSING_REQUEST_ID or INVALID_REQUEST, the MA USB host shall invalidate all outstanding TransferReq packets with Request ID fields set to values larger than indicated in the TransferResp packet and start transmitting new TransferReq packets starting with the Request ID value in the TransferReq packet. | TD 5.9 | |
| 5.4#28 | The target MA USB device fulfills an IN transfer request by transmitting one or more TransferResp packets. | Interop | |
| 5.4#29 | The interval between the release times of two successive TransferResp packets belonging to the same IN transfer assigned data channel shall not exceed aTransferRepeatTime, unless no data is available from the target USB endpoint or stream, in which case the target MA USB device may indicate a pending status and enter longer periods of inactivity. | Interop | |
| 5.4#30 | For an active IN transfer involving multiple TransferResp packets, if the MA USB host PAL expects a TransferResp packet and does not receive the packet within aTransferKeepAlive from the moment it received the last TransferResp packet through the assigned data channel, it shall transmit a TransferReq packet to the target MA USB device with the Request | TD 5.10 | |

| | | | |
|---|---|---|---|
| | ID field set to the value identifying the IN transfer, the Sequence Number field set to the value the MA USB host is expecting next, and the Remaining Size field set to the remaining number of bytes expected to complete the transfer. | | |
| 5.4#31 | An MA USB device that is experiencing delay in receiving data from a target USB endpoint or stream shall indicate a pending status and transition to a longer timeout by transmitting a null TransferResp packet with no payload. | TD 6.9 | |
| 5.4#32 | A TransferResp packet to indicate a pending status shall set the Status code to TRANSFER_PENDING and optionally the ARQ field to 1. | TD 6.9 | |
| 5.4#33 | A TransferResp packet to indicate a pending status shall have the Sequence Number set to 2^24 - 1 (aInvalidSequenceNumber) and Remaining Size field set to 0. | GTP<br>TD 6.9 | |
| 5.4#34 | If the ARQ field in a null TransferResp packet is set to 1, the packet starts an immediate exchange with the MA USB host that requires the host to acknowledge the null TransferResp packet through a TransferAck packet with the same Request ID and Status Code field values as those in the null TransferResp packet. | TD 5.12 | |
| 5.4#35 | The first TransferResp packet with the payload that follows a null TransferResp packet with Status code field set to TRANSFER_PENDING shall have the ARQ field set to 1, resulting in an immediate exchange with the MA USB host. | TD 6.9 | |
| 5.4#36 | Once the MA USB host PAL receives a TransferResp packet with the Status Code field set to TRANSFER_PENDING, it shall reset the TransferReq retry logic, and set the transfer timeout period to K * aTransferKeepAlive. | TD 5.12 | |
| 5.4#37 | The MA USB host PAL shall maintain the extended timeout period as long as the target endpoint or stream is in pending state, and shall change it to aTransferKeepAlive after receiving the first TransferResp packet with the Status code field set to a value other than TRANSFER_PENDING. | TD 5.12 | |
| 5.4#38 | When an MA USB device has one or more endpoints with pending status, it shall be able to respond to requests from the MA USB host targeted to its other endpoints. | TD 6.26 | |
| 5.4#39 | The target MA USB device shall mark the last TransferResp packet belonging to an IN transfer by setting the EoT field to 1, which results in an immediate exchange that requires the MA USB host to acknowledge the packet through a TransferAck packet, or optionally, through an outstanding | Interop<br>TD 5.11 | |

| | | | |
|---|---|---|---|
| | TransferReq packet if the TransferResp packet has the Status Code field set to NO_ERROR. | | |
| 5.4#40 | If acknowledging a TransferResp with EoT set to 1, a TransferAck packet shall have the same values for the Request ID, Sequence Number and Status Code fields as the corresponding values in the TransferResp packet. | TD 5.11 | |
| 5.4#41 | If the total data delivered to the MA USB host is less than the amount indicated in the Remaining Size field of the TransferReq packet, then the TransferResp packet carrying EoT field set to 1 shall have the Status Code field set to TRANSFER_SHORT_TRANSFER. | TD 6.34 | |
| 5.4#42 | A null TransferResp packet with the Status Code field set to TRANSFER_PENDING shall not have the EoT field set to 1. | TD 6.9 | |
| 5.4#43 | If the target endpoint returns a STALL handshake during a transfer, the target MA USB device shall proceed to deliver the entire payload it has received from the target endpoint or stream to the MA USB host before concluding the transfer in error. | N/A | Not testable |
| 5.4#44 | In the case of a STALL handshake, the MA USB device shall set the Status Code field to NO_ERROR in all TransferResp packets it transmits as part of the transfer, except the last TransferResp packet, which shall have the Status Code field set to TRANSFER_EP_STALL, the EoT field set to 1, and optionally the ARQ field set to 1. | TD 6.33 | |
| 5.4#45 | In the case of a STALL handshake, the last TransferResp packet initiates an immediate exchange with the MA USB host, requiring the host to acknowledge the TransferResp packet. | TD 5.13 | |
| 5.4#46 | In the case of a STALL handshake, the TransferAck packet shall have the same values for the Request ID, Sequence Number and Status Code fields as the corresponding values in the TransferResp packet. | TD 5.13 | |
| 5.4#47 | The target MA USB device shall queue any new transfer request that targets a stalled endpoint or stream for possible later processing. | N/A | Not testable |
| 5.4#48 | The last TransferResp packet belonging to an IN transfer that experiences a STALL condition shall have a nonzero Remaining Size field, although the EoT field in the packet is set to 1. | TD 6.33 | |
| 5.4#49 | In response to a TransferResp packet with gap in the Sequence Number field, the MA USB host PAL shall release to the assigned data channel a TransferReq packet with the Request ID field identifying the transfer request the earliest missing Sequence Number value belongs to, the Sequence Number field | TD 5.14 | |

| | | | |
|---|---|---|---|
| | set to the earliest missing Sequence Number value, the Remaining Size field set to the remaining size of the transfer identified by the Request ID field value, and the Status Code field set to NO_ERROR. | | |
| 5.4#50 | In response to a TransferResp packet with gap in the Sequence Number field, the MA USB host PAL shall release the required TransferReq packet within aTransferResponseTime from the moment the MA USB host PAL receives the TransferResp packet. | TD 5.14 | |
| 5.4#51 | A target MA USB device shall not remove any transfer data associated with an MA USB IN transfer from its buffer unless it receives one of the following packets: (a) a TransferAck packet acknowledging all TransferResp packets with Sequence Number values less than or equal to the value of the Sequence Number field in the packet in which case all acknowledged data is removed from the MA USB device buffer; (b) a TransferReq packet (including new outstanding transfer requests) acknowledge all TransferResp packets with Sequence Number values less than the value of the Sequence Number field in the packet, in which case all acknowledged data is removed from the MA USB device buffer; (c) a DevResetReq packet or DevDisconnectReq packet in which case all data for all endpoints and streams is removed from the MA USB device buffer; (d) a ClearTransfersReq packet, in which case all data related to corresponding endpoint(s) is removed from the MA USB device buffer; (e) a USBDevDisconnectReq packet for the target USB device, in which case all data related to the target USB device is removed from the MA USB device buffer; or (f) a CancelTransferReq packet, in which case all data related to the target request is removed from the MA USB device buffer. | N/A | Not testable |
| 5.4#52 | If a transfer is cancelled before any data was moved from the USB device (i.e., the Cancellation Status field in CancelTransferResp packet set to 1), the MA USB devices shall respond to a TransferReq packet for the cancelled transfer with a TransferResp packet with EOT set to 1, Status Code field set to TRANSFER_CANCELLED, the Remaining Size field set to 0, and ARQ field set to 1. | Interop | |
| 5.4#53 | If a transfer is cancelled before any data was moved from the USB device (i.e., the Cancellation Status field in CancelTransferResp packet set to 1), the MA USB device shall keep the state of the transfer until it receives the TransferAck packet from the MA USB host. | N/A | Not testable |
| 5.4#54 | If a transfer is cancelled before any data was moved from the USB device (i.e., the Cancellation Status field in CancelTransferResp packet set to 1), the MA USB host shall transmit a TransferAck packet only after it | Interop | |

| | | | |
|---|---|---|---|
| | has received all the TransferResp packets as well as the CancelTransferResp packet related to the cancelled transfer. | | |
| 5.4#55 | If a transfer is cancelled after some data was moved from the USB device (i.e., the Cancellation Status field in CancelTransferResp packet set to 2), the MA USB device shall transmit all the data received from the USB device to the MA USB host in TransferResp packets. | N/A | Not testable |
| 5.4#56 | If a transfer is cancelled after some data was moved from the USB device (i.e., the Cancellation Status field in CancelTransferResp packet set to 2), the last TransferResp packet of the transfer shall carry EOT field set to 1 with the Status Code field set to TRANSFER_CANCELLED. | Interop | |
| 5.4#57 | If a transfer is cancelled after some data was moved from the USB device (i.e., the Cancellation Status field in CancelTransferResp packet set to 2), the MA USB device shall keep the state of the transfer until it receives the TransferAck packet from the MA USB host. | N/A | Not testable |
| 5.4#58 | If a transfer is cancelled after some data was moved from the USB device (i.e., the Cancellation Status field in CancelTransferResp packet set to 2), the MA USB host shall transmit a TransferAck packet only after it has received all the TransferResp packets as well as the CancelTransferResp packet related to the cancelled transfer. | Interop | |
| 5.4#59 | If a transfer is cancelled after completion (i.e., the Cancellation Status field in CancelTransferResp packet set to 3), the MA USB device shall transmit all the data received from the USB device to the MA USB host in TransferResp packets, with the Status Code field set to the appropriate values (the same values carried if the transfer was not cancelled). | N/A | Not testable |
| 5.4#60 | If a transfer is cancelled after completion (i.e., the Cancellation Status field in CancelTransferResp packet set to 3), the last TransferResp packet of the transfer shall carry EOT field set to 1. | Interop | |
| 5.4#61 | If a transfer is cancelled after completion (i.e., the Cancellation Status field in CancelTransferResp packet set to 3), the MA USB device shall keep the state of the transfer until it receives the TransferAck packet from the MA USB host. | N/A | Not testable |
| 5.4#62 | If a transfer is cancelled after completion (i.e., the Cancellation Status field in CancelTransferResp packet set to 3), the MA USB host shall transmit a TransferAck packet only after it has received all the TransferResp packets as well as the | Interop | |

| | | | |
|---|---|---|---|
| | CancelTransferResp packet related to the cancelled transfer | | |
| 5.4#63 | If a cancelled transfer is not yet received (i.e., the Cancellation Status field in CancelTransferResp packet set to 4): If a TransferReq packet is received at the MA USB device, the MA USB device shall respond to it with no required knowledge of whether the transfer was previously cancelled. | N/A | Not testable |
| 5.4#64 | If a transfer with RequestID was serviced as part of ClearTransfersReq processing without any data being moved from the USB Device (i.e., the Cancellation Status field in CancelTransferResp packet set to 5), the MA USB device shall generate CancelTransferResp without generating TransferResp. | Interop | |
| 5.4#65 | If a transfer with RequestID was serviced as part of ClearTransfersReq processing without any data being moved from the USB Device (i.e., the Cancellation Status field in CancelTransferResp packet set to 5), the MA USB host, following receipt of the CancelTransferResp packet, shall not wait for TransferResp packets for this transfer. | N/A | Not testable |
| 5.4#66 | A target MA USB device that receives a ClearTransfersReq packet shall deliver all the data received from the target endpoint to the MA USB host through TransferResp packets. | N/A | Not testable |
| 5.4#67 | For a target MA USB device that receives a ClearTransfersReq packet where the transfer is cancelled before its completion (i.e., not all data related to the transfer is received from the target endpoint), the last TransferResp packet carrying data related to the transfer shall carry EOT field set to 1 with the Status Code field set to TRANSFER_CANCELLED. | N/A | Not testable |
| 5.4#68 | For a target MA USB device that receives a ClearTransfersReq packet, the TransferResp packets related to transfers that are completed without cancellation shall not set the Status Code field to TRANSFER_CANCELLED. | Interop | |
| 5.4#69 | An MA USB device shall discard any TransferReq packet received for a cancelled transfer (TransferReq packets carrying a Request ID value less than the value indicated in the Start Request ID field in the ClearTransfersReq packet). | Interop | |
| 5.4#70 | An MA USB device shall respond to a ClearTransfersReq packet with a ClearTransfersResp packet, only after all the TransferResp packets for the data already received from cancelled transfers are generated. | Interop | |
| 5.4#71 | An MA USB host shall reset the Sequence Number value to 0 before transmitting a TransferReq packet with | Interop | |

| | | | |
|---|---|---|---|
| | Request ID value indicated in the Start Request ID field in the ClearTransfersReq packet. | | |
| **Subsection reference: 5.4.1 Transfer description** | | | |
| **Subsection reference: 5.4.1.1 MA USB host PAL operation** | | | |
| **Subsection reference: 5.4.1.2 MA USB device PAL operation** | | | |
| **Subsection reference: 5.5 Protocol-managed OUT description** | | | |
| 5.5#1 | The MA USB host PAL initiates an MA USB OUT transfer by transmitting a TransferReq packet to a target MA USB device, which also carries some payload belonging to the transfer. | Interop | |
| 5.5#2 | Any remaining payload for an MA USB OUT transfer that is not included in the initial TransferReq packet shall be transferred through additional TransferReq packets. | Interop | |
| 5.5#3 | All p-managed OUT TransferReq packets belonging to a transfer, except possibly the last packet, shall include a multiple of maximum packet size of data supported by the target endpoint. | GTP | |
| 5.5#4 | The MA USB host assigns a request ID to each p-managed OUT transfer request and all associated TransferReq packets. | Interop | |
| 5.5#5 | The Request ID for a p-managed OUT TransferReq packet is reset to 0 at session initialization or upon any configuration event that returns the state of the endpoint or stream flow to the initial state. | TD 5.7 | |
| 5.5#6 | The Request ID for a p-managed OUT TransferReq packet is incremented by 1 for each new transfer request. | TD 5.6 | |
| 5.5#7 | The Request ID for a p-managed OUT TransferReq packet shall wraparound to 0 after reaching the maximum value of aMaxRequestID. | TD 5.6 | |
| 5.5#8 | The MA USB host shall have no more than (aMaxRequestID+1)/2 outstanding TransferReq packets for each target OUT endpoint or stream. | N/A | Not tested |
| 5.5#9 | The total number of outstanding transfers across all IN and OUT endpoints and streams shall not exceed the number returned by the target MA USB device in the Number of Outstanding Requests field in the CapResp packet. | N/A | Not tested |
| 5.5#10 | To complete an MA USB OUT transfer, the MA USB host PAL transmits one or more TransferReq packets to the target MA USB device PAL to transfer the USB payload to the target USB endpoint or stream in the strict order it is available in the host system. | Interop | |
| 5.5#11 | Each OUT TransferReq carries the same EP Handle, Stream ID, and Request ID as in the TransferReq packet that initiated the transfer. | Interop | |

| | | | |
|---|---|---|---|
| 5.5#12 | The OUT TransferReq Sequence Number field is set to 0 at session initialization or upon any configuration event that returns the state of the endpoint or stream flow to the initial state. | TD 5.7 | |
| 5.5#13 | The OUT TransferReq Sequence Number value is incremented by 1 after each new (not retried) TransferReq packet. | TD 5.6 | |
| 5.5#14 | The OUT TransferReq Sequence Number value wraps around to 0 after reaching the maximum value of aMaxSequenceNumber. | N/A | Not tested |
| 5.5#15 | The OUT TransferReq Sequence Number value keeps increasing across successive transfers. | TD 5.6 | |
| 5.5#16 | The MA USB host PAL shall have no more than (aMaxSequenceNumber+1)/2 outstanding TransferReq packets. | N/A | Not tested |
| 5.5#17 | The Remaining Size field of an OUT TransferReq packet carries the number of remaining bytes the host expects to transmit to complete the transfer. | Interop | |
| 5.5#18 | If the MA USB host PAL transmits additional OUT TransferReq packets to the same endpoint or stream, the MA USB device shall queue the received TransferReq packets in the increasing order of Request ID and Sequence Number for subsequent processing. | Interop | |
| 5.5#19 | In response to an invalid OUT TransferReq packet, the target MA USB device PAL shall discard the TransferReq packet and release to the data channel a TransferResp packet. | TD 6.8 | |
| 5.5#20 | In response to an invalid OUT TransferReq packet, the target MA USB device PAL shall send the TransferResp packet within aTransferResponseTime from the moment it receives the invalid TransferReq packet. | TD 6.8 | |
| 5.5#21 | A TransferResp packet sent in response to an invalid OUT TransferReq packet shall have the Request ID and Sequence Number set to the Request ID and Sequence Number of the last TransferReq packet received. | TD 6.8 | |
| 5.5#22 | The TransferResp packet sent in response to an invalid OUT TransferReq packet shall have the Status Code set to MISSING_REQUEST_ID if the TransferReq shows a gap in its Request ID field | TD 6.8 | |
| 5.5#23 | The TransferResp packet sent in response to an invalid OUT TransferReq packet shall have its Status Code set to INVALID_REQUEST if the received TransferReq packet is invalid independent of its Request ID field. | TD 6.8 | |

| | | | |
|---|---|---|---|
| 5.5#24 | In response to a TransferResp packet with the Status Code set to MISSING_REQUEST_ID or INVALID_REQUEST, the MA USB host PAL shall invalidate all outstanding TransferReq packets and start transmitting the new TransferReq packets starting with the Request ID and Sequence Number values in the TransferResp packet. | TD 5.9 | |
| 5.5#25 | A retried OUT TransferReq packet for an active transfer shall not carry a different payload size and may not change the size of the transfer. | TD 5.3 | |
| 5.5#26 | In response to a valid TransferReq packet with the ARQ field set to 1, the target MA USB device PAL shall release a TransferResp packet to the assigned data channel with the Response ID and Sequence Number fields set to the Request ID and Sequence number values of the last TransferReq packet the MA USB device PA has received in order. | TD 6.10 | |
| 5.5#27 | At least one TransferResp packet is transmitted for each OUT transfer request to indicate the transfer conclusion. | Interop | |
| 5.5#28 | The target MA USB device PAL shall deliver all received USB payload to the target endpoint or stream in the strict order of Sequence Number value, and transfer requests shall be completed in strict order of Request ID value. | Interop | |
| 5.5#29 | The target MA USB device PAL shall deliver all received USB payload to the target endpoint or stream in the strict order of Sequence Number value. | Interop | |
| 5.5#30 | The target MA USB device PAL shall complete all transfer requests in strict order of Request ID value. | Interop | |
| 5.5#31 | The MA USB device PAL shall notify the MA USB host PAL of transfer completion by transmitting a TransferResp packet with Request ID identifying the completed transfer, the Sequence Number field set to the latest Sequence Number value received by the device PAL, updated Credit field, the EoT field set to 1, and the Status Code field set accordingly. | Interop | |
| 5.5#32 | The MA USB host PAL shall acknowledge the receipt of a valid TransferResp packet with the EoT field set to 1 by transmitting a TransferAck packet to the MA USB device PAL with Request ID, Sequence Number and Status code fields set to the same values as those in the TransferResp packet. | TD 5.6 | |
| 5.5#33 | In response to a valid TransferReq packet with a gap in the Sequence Number value, the target MA USB device PAL shall release a TransferResp packet to the assigned data channel within aTransferRespTime from the moment the MA USB device PAL receives the TransferReq packet. | TD 6.11 | |

| | | | |
|---|---|---|---|
| 5.5#34 | In response to a valid TransferReq packet with a gap in the Sequence Number value, the target MA USB device PAL shall send a TransferResp with the Request ID and Sequence Number fields set to the Request ID and Sequence Number values the MA USB device PAL is expecting next, and the Status Code field set to MISSING SEQUENCE NUMBER. | TD 6.11 | |
| 5.5#35 | If the target endpoint returns a STALL handshake during a transfer, the target MA USB device shall transmit a TransferResp packet to the MA USB host, with the Request ID field set to the Request ID value of the STALLed TransferReq, the EoT field set to 1, and the Status Code field set to TRANSFER_EP_STALL. | TD 6.33 | |
| 5.5#36 | Receiving a TransferResp packet with the Status Code set to TRANSFER_EP_STALL results in completion of the transfer on the MA USB host. | TD 5.13 | |
| 5.5#37 | The MA USB device shall discard but acknowledge any subsequent data received for the transfer that experienced the STALL condition. | N/A | Not tested |
| 5.5#38 | Until the MA USB host clears the STALL condition and the target EP handle returns to the Active state, the target MA USB device shall buffer and acknowledge each TransferReq packet received that does not belong to the transfer request that experienced the STALL condition by transmitting a TransferResp packet with the Status Code set to INVALID_EP_HANDLE_STATE. | TD 6.33 | |
| 5.5#39 | The MA USB host shall not remove any unacknowledged data associated with an endpoint in STALL condition. | N/A | Not tested |
| 5.5#40 | If a MA USB device PAL receives a CancelTransferReq packet corresponding to an active transfer, it should discard all the data corresponding to the transfer identified in the CancelTransferReq packet. | N/A | Not tested |
| 5.5#41 | If the MA USB device PAL receives a CancelTransferReq packet corresponding to an active transfer, it shall keep account of the sequence numbers of the TransferReq packets received. | N/A | Not tested |
| 5.5#42 | If a transfer is cancelled before any data was moved to the USB device (i.e., the Cancellation Status field in CancelTransferResp packet set to 1), the MA USB devices shall respond to a TransferReq packet for the cancelled transfer with a TransferResp packet with EOT set to 1, Status Code field set to TRANSFER_CANCELLED, and Sequence Number field set to aInvalidSequenceNumber. | Interop | |
| 5.5#43 | If a transfer is cancelled before any data was moved to the USB device (i.e., the Cancellation Status field in CancelTransferResp packet set to 1, the MA USB device shall keep the state of the transfer until it | N/A | Not tested |

| | | | |
|---|---|---|---|
| | receives the TransferAck packet from the MA USB host. | | |
| 5.5#44 | If a transfer is cancelled before any data was moved to the USB device (i.e., the Cancellation Status field in CancelTransferResp packet set to 1), the MA USB host shall transmit the TransferAck packet only after it has received all the TransferResp packets as well as the CancelTransferResp packet related to the cancelled transfer. | Interop | |
| 5.5#45 | If a transfer is cancelled after some data was moved to the USB device (i.e., the Cancellation Status field in CancelTransferResp packet set to 2), the MA USB devices shall transmit a TransferResp packet with EOT set to 1, Status Code field set to TRANSFER_CANCELLED. | Interop | |
| 5.5#46 | If a transfer is cancelled after some data was moved to the USB device (i.e., the Cancellation Status field in CancelTransferResp packet set to 2), the MA USB device shall keep the state of the transfer until it receives the TransferAck packet from the MA USB host. | N/A | Not tested |
| 5.5#47 | If a transfer is cancelled after some data was moved to the USB device (i.e., the Cancellation Status field in CancelTransferResp packet set to 2), the MA USB host shall transmit the TransferAck packet only after it has received all the TransferResp packets as well as the CancelTransferResp packet related to the cancelled transfer. | Interop | |
| 5.5#48 | If a transfer is completed (i.e., the Cancellation Status field in CancelTransferResp packet set to 3), the MA USB devices shall transmit a TransferResp packet with EOT set to 1, Status Code field set to the appropriate value (the same value carried if the transfer was not cancelled). | Interop | |
| 5.5#49 | If a transfer is completed (i.e., the Cancellation Status field in CancelTransferResp packet set to 3), the MA USB device shall keep the state of the transfer until it receives the TransferAck packet from the MA USB host. | N/A | Not tested |
| 5.5#50 | If a transfer is completed (i.e., the Cancellation Status field in CancelTransferResp packet set to 3), the MA USB host shall transmit the TransferAck packet only after it has received all the TransferResp packets as well as the CancelTransferResp packet related to the cancelled transfer. | Interop | |
| 5.5#51 | If a transfer is not yet received (i.e., the Cancellation Status field in CancelTransferResp packet set to 4), if the MA USB device receives a TransferReq packet, the MA USB device shall respond to it with no required | N/A | Not tested |

| | | | |
|---|---|---|---|
| | knowledge of whether the transfer was previously cancelled. | | |
| 5.5#52 | If a transfer with RequestID was serviced as part of ClearTransfersReq processing without any data being moved to the USB Device (i.e., the Cancellation Status field in CancelTransferResp packet set to 5): The MA USB device shall generate CancelTransferResp without generating TransferResp. | N/A | Not tested |
| 5.5#53 | If a transfer with RequestID was serviced as part of ClearTransfersReq processing without any data being moved to the USB Device (i.e., the Cancellation Status field in CancelTransferResp packet set to 5), the MA USB host, following receipt of the CancelTransferResp packet, shall not wait for TransferResp packets for this transfer. | N/A | Not tested |
| 5.5#54 | If a MA USB device PAL receives a ClearTransfersReq packet corresponding to an endpoint with active transfers, it should discard all the data corresponding to the endpoint identified in the ClearTransfersReq packet. | N/A | Not tested |
| 5.5#55 | If a MA USB device PAL receives a ClearTransfersReq packet corresponding to an endpoint with active transfers, it shall keep account of the sequence numbers of the TransferReq packets received. | N/A | Not tested |
| 5.5#56 | If a MA USB device receives a TransferReq packet for a cancelled transfer (TransferReq packets carrying a Request ID value less than the value indicated in the Start Request ID field in the ClearTransfersReq packet) and it has not yet delivered any data related to the transfer to the device, it shall discard the TransferReq packet, otherwise it shall respond with a TransferResp packet. | N/A | Not tested |
| 5.5#57 | If a transfer is cancelled before its completion (i.e., not all data related to the transfer is delivered to the target endpoint), the last TransferResp packet related to the transfer shall carry EOT field set to 1 with the Status Code field set to TRANSFER_CANCELLED. | N/A | Not tested |
| 5.5#58 | The Status Code field shall not be set to TRANSFER_CANCELLED if the transfer is completed. | N/A | Not tested |
| 5.5#59 | The MA USB host shall not remove the data for any unacknowledged TransferReq packets of a cancelled transfer before receiving the ClearTransfersResp packet. | N/A | Not tested |
| 5.5#60 | The MA USB host shall reset the Sequence Number value to 0 before transmitting a TransferReq packet with Request ID value indicated in the Start Request ID field in the ClearTransfersReq packet. | Interop | |

| Subsection reference: 5.5.1 MA USB device buffer management for OUT transfers | | | |
|---|---|---|---|
| 5.5.1#1 | The MA USB device PAL is required to inform the MA USB host PAL of its available buffer space for each USB endpoint behind the MA USB device PAL. | Interop | |
| 5.5.1#2 | The MA USB device delivers the initial credit for an endpoint to the MA USB host using the Buffer Size field in the EPHandleResp packet. | Interop | |
| 5.5.1#3 | Credits are allocated at the MA USB device PAL for each endpoint. | Interop | |
| 5.5.1#4 | In the case of an Enhanced SuperSpeed bulk OUT endpoint that supports Enhanced SuperSpeed Stream Protocol, the allocated credits are the total available for all the streams. | Interop | |
| 5.5.1#5 | The counter used to track credits for each endpoint is initialized to the value of the Buffer Size field in the EPHandleResp packet at any configuration event intended to return the state of an endpoint to the initial state. | Interop | |
| 5.5.1#6 | Credits are de-allocated when the deliverable TransferReq packet arrives and its data payload is accepted into the buffer. | Interop | |
| 5.5.1#7 | The MA USB device PAL uses the Sequence Number value in the TransferReq packet to determine whether the payload in the TransferReq packet is deliverable to the target endpoint. | Interop | |
| 5.5.1#8 | The data associated with a TransferReq packet targeted to an endpoint is deliverable if the data for all preceding TransferReq packets target to the endpoint has successfully been accepted. | N/A | Not testable |
| 5.5.1#9 | If the TransferReq packet has the Retry field set to 1 and the associated data has already Been accepted by the MA USB device PAL, the data is dropped. | N/A | Not testable |
| 5.5.1#10 | Credits are allocated when data is removed from the buffer. | N/A | Not testable |
| 5.5.1#11 | If a MA USB device PAL receives a DevResetReq packet or DevDisconnectReq packet, it shall remove all data for all endpoints and streams from the MA USB device buffer. | N/A | Not testable |
| 5.5.1#12 | If a MA USB device PAL receives a ClearTransfersReq packet, it shall remove data from for all corresponding endpoints from the MA USB device buffer. | N/A | Not testable |
| 5.5.1#13 | If a MA USB device PAL receives a USBDevDisconnectReq packet, it shall remove all data related to the target USB device from the MA USB device buffer. | N/A | Not testable |

| | | | |
|---|---|---|---|
| 5.5.1#14 | If a MA USB device PAL receives a CancelTransferReq packet, it shall remove all data related to the target request from the MA USB device buffer. | N/A | Not testable |
| 5.5.1#15 | A MA USB device shall provide credits for each of the endpoints in multiples of Credit Consumption Unit of that endpoint. | TBD | |
| 5.5.1#16 | Retransmitted data does not count toward credit available for unacknowledged data. | TBD | |
| 5.5.1#17 | The MA USB host shall set its local counter for tracking credits on an endpoint to the value of the Credit field in a received TransferResp or EPHandleResp packet. | N/A | Not testable |
| 5.5.1#18 | The MA USB host shall accurately track credits available and used during data streaming. | N/A | Not tested |
| 5.5.1#19 | Credits are consumed (released) when the TransferResp packet associated with those credits is received. | N/A | Not tested |
| 5.5.1#20 | The MA USB host shall account for consumed credits in multiples of Credit Consumption Unit of that endpoint. | N/A | Not tested |
| 5.5.1#21 | If the Elastic Buffer Capability field in the CapResp packet is set to 0, the MA USB host PAL shall not transmit TransferReq packets targeted for an endpoint unless it has credits for transmitting data to that endpoint. | TBD | |
| 5.5.1#22 | The MA USB host PAL shall limit the amount of data transmission to the target endpoint to the credit available for that endpoint. | TBD | |
| **Subsection reference: 5.5.2 Transfer description** | | | |
| **Subsection reference: 5.5.2.1 MA USB host PAL operation** | | | |
| **Subsection reference: 5.5.2.2 MA USB device PAL operation** | | | |
| **Subsection reference: 5.6 Link-managed OUT transfers** | | | |
| **Subsection reference: 5.6.1 Transfer description** | | | |
| 5.6.1#1 | The MA USB host shall start the set up phase of an l-managed OUT transfer by sending a TransferSetupReq packet to the target MA USB device over control or any available data channel. | TBD | |
| 5.6.1#2 | After sending the TransferSetupReq packet for an l-managed OUT transfer the MA USB host sets up the flow-controlled connection required for the transfer. | TBD | |
| 5.6.1#3 | MA USB host shall share the connection context with the MA USB device through a TransferReq packet before starting any new connection set up. | TBD | |
| 5.6.1#4 | If no connection can be secured to support the transfer, MA USB host shall release any local or link-layer resources allocated to the transfer and return the appropriate USBDI error to the application to indicate the transfer failure. | TBD | |

| | | | |
|---|---|---|---|
| 5.6.1#5 | If the connection setup is successful, the target MA USB device programs local resources to direct the OUT transfer payload arriving over the established connection to the target endpoint. | TBD | |
| 5.6.1#6 | If the connection is successful, the target MA USB device sends a TransferSetupResp packet to the MA USB host. | TBD | |
| 5.6.1#7 | If the MA USB host does not receive a TransferSetupResp packet by aTransferTimeout after the connection has been established from the MA USB host perspective, the MA USB host shall send another TransferSetupReq packet with the packet Retry bit set to 1 for a maximum of aTransferSetupRetries times. | TBD | |
| 5.6.1#8 | The MA USB starts the data phase of a transfer by sending one or more TransferReq packets to the MA USB device over the established connection. | TBD | |
| 5.6.1#9 | With the exception of the last TransferReq data packet of a transfer, the payload size for each TransferReq packet belonging to a transfer is a multiple of the maximum packet size the target endpoint supports. | TBD | |
| 5.6.1#10 | All TransferReq packets belonging to the same transfer carry the same number for the Request ID field. | TBD | |
| 5.6.1#11 | Request ID is selected by the MA USB host and shall be unique within the scope of a target endpoint. | TBD | |
| 5.6.1#12 | The Request ID used for each new transfer shall be strictly increasing, with no gaps. | TBD | |
| 5.6.1#13 | The MA USB host shall not have more than one pending transfer using same Request ID and targeting the same endpoint. | TBD | |
| 5.6.1#14 | Each TransferReq packet carries a sequence number, which is set to zero in the first TransferReq packet for a given transfer. | TBD | |
| 5.6.1#15 | The sequence number in a TransferReq packet is incremented by one in every subsequent TransferReq packet. | TBD | |
| 5.6.1#16 | Each TransferReq packet carries in its Transfer Size field the number of remaining bytes for the transfer. | TBD | |
| 5.6.1#17 | The target MA USB device shall send a TransferResp packet to the MA USB host when the payload in the final TransferReq packet belonging to a transfer is successfully delivered to the target endpoint. | TBD | |
| 5.6.1#18 | When the MA USB device sends a TransferResp packet in response to a successfully delivered transfer, the Request ID and Sequence Number fields shall be set to the same values as the final TransferReq packet, the remaining Size shall be set to 0, and the Status field set to SUCCESS. | TBD | |

| | | | |
|---|---|---|---|
| 5.6.1#19 | The target MA USB device shall send a TransferResp packet to the MA USB host when the delivering payload faces a locally non-recoverable error. | TBD | |
| 5.6.1#20 | When the MA USB device sends a TransferResp packet in response to a non-recoverable error of a transfer, the Request ID and Sequence Number fields shall be set to the same values as those in the TransferReq packet whose payload experienced the error, the Remaining Size field set to the transfer size minus the number of bytes successfully delivered to the endpoint, and the Status field set to an appropriate error code from the list in Table 6 of the MA USB spec. | TBD | |
| 5.6.1#21 | The interval between when the final TransferReq packet is received by the target MA USB device and when the corresponding TransferResp packet appears over the link on the MA USB device side shall not exceed aTransferKeepAlive. | TBD | |
| 5.6.1#22 | If the MA USB host does not receive a TransferReq packet by aTransferKeepAlive after it has successfully submitted the final TransferReq packet for a transfer for transmission, it shall attempt to solicit a TransferResp packet by sending another TransferReq packet with all packet fails set to the same value as those in the final TransferReq packet for the transfer, except for the Retry bit set to 1. | TBD | |
| 5.6.1#23 | If necessary, the MA USB host shall repeat sending the TransferReq packet for a maximum number of times equal to aControlTransferRetries for control transfers, aBulkTransferRetries for bulk transfers, and aInterruptTransferRetries for interrupt transfers. | TBD | |
| 5.6.1#24 | If no TransferResp packet is received after a maximum number of retries allowable for the transfer type, the MA USB host shall return the appropriate USBDI error to the application indicating the transfer failure, and shall start the tear down phase of the transfer. | TBD | |
| 5.6.1#25 | From the MA USB host perspective, an OUT transfer is complete when the MA USB host receives a TransferReq packet with the Status field set to a value other than SUCCESS. | TBD | |
| 5.6.1#26 | When the MA USB host receives a TransferResp packet with the status field set to a value other than SUCCESS, the MA USB host shall return to the application a status code that indicates failure of the write request that prompted the MA USB transfer, as well as the reason for the failure. | TBD | |
| 5.6.1#27 | From the MA USB host perspective, an OUT transfer is complete when the MA USB host receives a TransferResp packet with the Status field set to SUCCESS and the Remaining Size field set to 0. | TBD | |

| | | | |
|---|---|---|---|
| 5.6.1#28 | When the MA USB host receives a TransferResp packet with the Status field set to SUCCESS and the Remaining Size field set to 0, if the final TransferReq packet associated with the transfer has not been sent, or the Sequence Number does not equal the Sequence Number in the final TransferReq packet, the MA USB host shall return to the application a status code that indicates failure of the write request that prompted the MA USB transfer, as well as the reason for the failure. | TBD | |
| 5.6.1#29 | When the MA USB host receives a TransferResp packet with the Status field set to SUCCESS and the Remaining Size field set to 0, and if the final TransferReq packet associated with the transfer has been sent, the Sequence Number in the TransferResp packet equals the Sequence Number in the final TransferReq packet, the MA USB host shall assume the OUT transfer is complete with no error. | TBD | |
| 5.6.1#30 | Once all OUT transfers serving a common application-level write request are complete with no error, the MA USB host shall return a status code to the application that indicates the success of the write request. | TBD | |
| 5.6.1#31 | At any point during the set up or data phase of an l-managed transfer, the MA USB host may start the tear down phase by tearing down the lower-layer link resources set up in the support of the transfer, and sending a TransferTearDownConf packet to the target MA USB device on the control channel. | TBD | |
| 5.6.1#32 | A target MA USB device that detects releasing of lower-layer link resources associated with an l-managed transfer on a target endpoint, or receives a TransferTearDownConf packet for an l-managed transfer on a target endpoint, shall stop sending TransferResp packets for any pending transfer requests, and shall silently discard all transfer requests associated with the target endpoint. | TBD | |
| **Subsection reference: 5.6.2 Transfer mode selection** | | | |
| 5.6.2#1 | When targeting a non-isochronous OUT endpoint, the exercised transfer mode is decided by the MA USB host. | | |
| 5.6.2#2 | If the target endpoint indicates support for l-managed transfer mode in its MA USB EP descriptor, the MA USB host may choose to exercise either the p-managed or l-managed transfer mode when targeting the endpoint. | TBD | |
| 5.6.2#3 | Successful set up of an l-managed transfer indicates switch to l-managed mode. | TBD | |
| 5.6.2#4 | Successful tear-down of an l-managed transfer indicates switch to p-managed mode. | TBD | |

| | | | |
|---|---|---|---|
| **Subsection reference: 5.7 Control transfers** | | | |
| **Subsection reference: 5.7.1 Setup stage** | | | |
| 5.7.1#1 | An MA USB device shall have buffer space of at least aMinControlTransferBufferSize bytes available for the first control transfer targeting the default control endpoint of each USB device behind the MA USB device. | N/A | Not testable |
| 5.7.1#2 | Each MA USB control transfer is initiated through a control TransferReq packet which carries a Request ID field with a new value. | Interop | |
| 5.7.1#3 | Each MA USB control transfer is initiated through a control transfer which carries a Sequence Number field set to zero. | GTP | |
| 5.7.1#4 | The sequence number of both the MA USB host and MA USB device re-initialize at the beginning of each control transfer request. | Interop | |
| 5.7.1#5 | Each MA USB control transfer setup stage includes 8 bytes of setup data as payload. | Interop | |
| 5.7.1#6 | The setup stage of a control OUT transfer includes as many bytes of the control write data that the target MA USB device can accept. | Interop | |
| 5.7.1#7 | A device shall transmit a TransferResp packet to the start of a new control transfer. | Interop | |
| 5.7.1#8 | For control OUT transfers, the TransferResp packet shall include the Credit field for subsequent data TransferReq packets and initialize its expected sequence number to 1. | TBD | |
| **Subsection reference: 5.7.2 Data stage for control OUT transfers** | | | |
| 5.7.2#1 | The data stage of control OUT transfers follows the USB p-managed OUT or l-managed OUT data transfer model. | TBD | |
| 5.7.2#2 | If the size of the Data stage OUT data is larger than what the MA USB device can accept, then the host increments the sequence number counter and continues with TransferReq for OUT transfers following the transmission of the control TransferReq packet. | TBD | |
| 5.7.2#3 | On receipt of the Setup TransferReq for control OUT requests, the target MA USB device initializes its expected sequence number to 1. | Interop | |
| **Subsection reference: 5.7.3 Data stage for control IN transfers** | | | |
| 5.7.3#1 | The MA USB host sets its expected sequence number to 0 after transmission of the control TransferReq packet for an IN transfer. | Interop | |
| 5.7.3#2 | On receipt of the setup TransferReq for control IN requests, the target MA USB device initializes it | Interop | |

| | | | |
|---|---|---|---|
| | sequence number to 0 and continues with the IN transfer with transmitting TransferResp packets with the IN data. | | |
| **Subsection reference: 5.7.4 Status stage** | | | |
| 5.7.4#1 | For a control IN transfer, the status stage result is included in the last TransferResp packet carrying payload. | Interop | |
| 5.7.4#2 | For a control OUT transfer, the status stage result is included in the TransferResp packet with the EoT field set to 1. | Interop | |
| 5.7.4#3 | Control endpoints shall not support functional STALL as defined in [USB2.0] and [USB 3.1]. | N/A | Not tested |
| **Subsection reference: 5.8 Bulk transfers** | | | |
| **Subsection reference: 5.9 Interrupt transfers** | | | |
| **Subsection reference: 5.10 Isochronous transfers** | | | |
| **Subsection reference: 5.10.1.1 Isochronous data blocks** | | | |
| 5.10.1.1#1 | At the beginning of the Isochronous packet are isochronous headers, placed one after another with the Segment Number field strictly increasing. | GTP | |
| 5.10.1.1#2 | Isochronous data blocks follow the isochronous headers, and in the exact same order of the headers. | Interop | |
| 5.10.1.1#3 | If the MA USB device that sends or receives the isochronous data packet has required DWORD alignment for the isochronous payload, each isochronous data block is padded with 0 to 3 bytes to make the block length a multiple of 4 bytes. | TD 5.15<br>TD 6.5 | |
| 5.10.1.1#4 | If the MA USB device that sends or receives the isochronous data packet does not require DWORD alignment for the isochronous payload, isochronous data blocks are packed after each other with no padding. | TD 5.15<br>TD 6.5 | |
| 5.10.1.1#5 | All isochronous headers in a packet have the same length, which can be 4, 8, or 12 bytes. | TD 6.5 | |
| 5.10.1.1#6 | The format (and length) of the isochronous headers is defined by the Isochronous Header Type field in the isochronous data packet header. | Interop | |
| 5.10.1.1#7 | The short format (Isochronous Header Format = 0) is 4 bytes in length, and does not support fragmentation. | TBD | |
| 5.10.1.1#8 | The standard format (Isochronous Header Format = 1) is 8 bytes in length, and supports fragmentation of segments as large as 64KB. | TBD | |
| 5.10.1.1#9 | The long format (Isochronous Header Format = 2) and is 12 bytes in length, and supports fragmentation of segments as large as 4GB. | TBD | |

| | | | |
|---|---|---|---|
| 5.10.1.1#10 | The long format of isochronous headers shall not be used unless the packet is carrying a fragment of an isochronous segment that is larger than 64KB. | TBD | |
| 5.10.1.1#11 | The 16-bit Block Length of an Isochronous header carries the length of the associated data block in bytes. | Interop | |
| 5.10.1.1#12 | The 12-bit Segment Number of an Isochronous header identifies the segment the data block belongs to. | Interop | |
| 5.10.1.1#13 | Isochronous Segment Numbers start at 0. | Interop | |
| 5.10.1.1#14 | Isochronous segments are sent in order, and segments with no data may be skipped. | Interop | |
| 5.10.1.1#15 | No null data block shall be sent with the Fragment bit set to 1. | TBD | |
| 5.10.1.1#16 | The Fragment bit of the S-Flags field of an Isochronous header is set to 1 if the data block associated with the isochronous header is carrying an incomplete isochronous segment, otherwise 0. | TBD | |
| 5.10.1.1#17 | The Fragment bit of the S-Flags field of an Isochronous header is set to 0 in short isochronous headers with no fragmentation support. | TBD | |
| 5.10.1.1#18 | The Last Fragment bit of the S-Flags field of an Isochronous header is set to 1 if the carried fragment is the last fragment of an isochronous segment, and 0 otherwise. | TBD | |
| 5.10.1.1#19 | The Last Fragment bit of the S-Flags field of an Isochronous header is reserved and set to 0 if Fragment subfield is 0. | TBD | |
| 5.10.1.1#20 | The Reserved bits of the S-Flags field of an Isochronous header shall be set to 0. | TBD | |
| 5.10.1.1#21 | The long and extended isochronous headers include the Segment Length field, which indicates the length of the isochronous segment in bytes. | TBD | |
| 5.10.1.1#22 | The Segment Length field is 2 bytes long in the long format. | TBD | |
| 5.10.1.1#23 | The Segment length field is 4 bytes long in the extended format. | TBD | |
| 5.10.1.1#24 | The long and extended isochronous headers include the Fragment Offset field, which indicates the byte offset of the carried fragment relative to the segment beginning. | TBD | |
| 5.10.1.1#25 | The Fragment Offset field is 2 bytes long in the long format. | TBD | |
| 5.10.1.1#26 | The Fragment Offset field is 4 bytes long in the extended format. | TBD | |
| 5.10.1.1#27 | The Last Fragment bit of the S-Flags field is set to 1 to indicate that the fragment carried by the isochronous | TBD | |

| | | | |
|---|---|---|---|
| | data block is the last fragment of an isochronous segment. | | |
| 5.10.1.1#28 | No isochronous data packet shall include more than two fragments. | TBD | |
| 5.10.1.1#29 | The first isochronous fragment, if present, shall satisfy exactly one of the following conditions | TBD | |
| 5.10.1.1#30 | The first fragment of an Isochronous data packet shall either have (Last Fragment = 1) or (Last Fragment = 0 and Fragment Offset != 0) or (Fragment Offset field = 0). | TBD | |
| 5.10.1.1#31 | The second fragment of an Isochronous data packet shall have (Fragment Offset field = 0). | TBD | |
| 5.10.1.1#32 | No incomplete Isochronous segments shall be delivered with lost bytes (gap) in the middle. | TBD | |
| 5.10.1.1#33 | All isochronous data packets that begin with the fragments of the same isochronous segment carry the same value for the Presentation Time field, except for require packets corresponding to an isochronous OUT transfer with ASAP delivery, where the Presentation Time field in each of the request packets is reserved. | TBD | |
| 5.10.1.1#34 | Regardless of the delivery mode of an isochronous transfer, the target MA USB device returns the presentation time of the first segment of the transfer in the Presentation Time field of each of the response packets corresponding to the transfer. | TBD | |
| **Subsection reference: 5.10.1.2 Isochronous read size blocks** | | | |
| 5.10.1.2#1 | Isochronous read size blocks are carried immediately after the IsochTransferReq packet header. | TBD | |
| 5.10.1.2#2 | All IRS blocks in a packet have the same length, which can be 4 or 8 bytes. | TBD | |
| 5.10.1.2#3 | The long format for IRS blocks shall not be used unless the IsochTransferReq packet needs to specify at least one Maximum Segment Length value that exceeds 1MB. | TBD | |
| 5.10.1.2#4 | The 12-bit Service Intervals field defines the number of consecutive target Service Intervals to which the IRS block applies | TBD | |
| 5.10.1.2#5 | The Maximum Segment Length field indicates the size of the largest isochronous segment that the MA USB host can receive during any of the target Service Intervals the IRS block applies to. | TBD | |
| 5.10.1.2#6 | The IRS blocks in an IsochTransferReq packet apply to the target Service Intervals in strict order. | TBD | |
| 5.10.1.2#7 | IsochTransferReq packets shall not include an IRS block with the Service Intervals field set to 0. | TBD | |

| | | | |
|---|---|---|---|
| 5.10.1.2#8 | The sum of the values in the Service Intervals fields of all IRS blocks in an IsochTransferReq packet shall be equal to the value of the Number of Segments field in the IsochTransferReq packet header. | TBD | |
| **Subsection reference: 5.10.2 Isochronous IN transfers** | | | |
| 5.10.2#1 | The MA USB host initiates an isochronous IN transfer by sending an IsochTransferReq packet to the target MA USB device, indicating the target isochronous endpoint on the MA USB device, the Request ID assigned to the transfer request by the MA USB host, the number of isochronous segments that are being requested, the beginning time of the first target Service Interval, and a set of IRS blocks. | TBD | |
| 5.10.2#2 | The Request ID assigned to each Isochronous transfer shall be unique within the scope of the target endpoint. | TBD | |
| 5.10.2#3 | The Request ID starts at zero for the first isochronous IN transfer after any configuration event intended to return the state of an endpoint flow to the initial state. | TD 5.7 | |
| 5.10.2#4 | The Request ID is incremented by 1 for each successive isochronous IN request. | TD 5.6 | |
| 5.10.2#5 | The MA USB host shall not have more than one pending isochronous transfer using the same Request ID and targeting the same endpoint. | TBD | |
| 5.10.2#6 | The rules for assigning Sequence Numbers to successive IsochTransferResp packets belonging to the same transfer are the same as other MA USB IN transfers except the Sequence Number field is reset to 0 for every new MA USB isochronous IN transfer. | TBD | |
| 5.10.2#7 | The Presentation Time field in successive IsochTransferReq packets that target the same endpoint and do not indicate ASAP delivery shall be strictly increasing. | TBD | |
| 5.10.2#8 | In response to an IsochTransferReq packet, the target MA USB device programs its local resources to read from the target endpoint during the target Service intervals. | TBD | |
| 5.10.2#9 | The target MA USB device shall packetize the isochronous payload as defined in Section 5.10.1 and shall transmit the payload back to the MA USB host in the strict order it was received from the target endpoint, using one or more IsochTransferResp. | TBD | |
| 5.10.2#10 | Each IsochTransferResp packet belonging to an isochronous IN transfer carries the Request ID that was present in the IsochTransferReq packet initiating the transfer. | TBD | |
| 5.10.2#11 | Each IsochTransferResp packet carries a sequence number which is set to zero in the first IsochTransferResp packet belonging to that transfer. | TBD | |

| | | | |
|---|---|---|---|
| 5.10.2#12 | Each IsochTransferResp packet carries in its Presentation Time field the Global Time pointing to the beginning of the Service Interval corresponding to the first isochronous segment or fragment carried in the packet. | TBD | |
| 5.10.2#13 | The EoT field shall be set to 0 in all IsochTransferResp packets belonging to an isochronous IN transfer, except the last packet, where it shall be set to 1. | TBD | |
| 5.10.2#14 | The EPS and Status code fields in each IsochTransferResp packet carry the status of the endpoint and the status code related to the transfer. | TBD | |
| 5.10.2#15 | The target MA USB device shall transmit an IsochTransferResp packet when the MA USB isochronous IN transfer has not generated any data. | TBD | |
| 5.10.2#16 | An IsochTransferReq packet initiates the transfer, targeting a remote isochronous IN endpoint over S successive Service Intervals, with the first Service Interval starting at the MA USB Global Time of F:M. | TBD | |
| 5.10.2#17 | The target MA USB device, synchronized with the MA USB host, schedules one or more isochronous IN transfers on the local bus, and packetizes and transmits isochronous segments through one or more IsochTransferResp packets as segments become available. | TBD | |
| 5.10.2#18 | The Presentation Time field in all IsochTransferResp packets belonging to an MA USB transfer carries the actual delivery time of the first isochronous segment, except when the target MA USB device experiences an error during the transfer, in which case the Presentation Time field is reserved and set to 0. | TBD | |
| **Subsection reference: 5.10.2.1 MA USB host requirements** | | | |
| 5.10.2.1#1 | The MA USB host PAL shall not release an IsochTransferReq to the network that indicates a start time (Presentation Time) more than aMaxFrameDistance USB frames before or after the MA USB Global Time at the moment the first bit of the packet is released to the network. | TBD | |
| 5.10.2.1#2 | The MA USB host PAL shall conclude a pending MA USB isochronous IN transfer when it receives an IsochTransferResp packet that belongs to the transfer and has the EoT subfield set to 1 or when it receives an IsochTransferResp packet that belongs to the transfer and indicates a non-recoverable error, or at pMaxDeviceIsochINRespDelay + aMaxIsochLinkDelay time units after the end of the last Service Interval targeted by the transfer. | TBD | |
| 5.10.2.1#3 | Once all pending MA USB isochronous IN transfer corresponding to an application-level isochronous read request are concluded, the MA USB host PAL | TBD | |

| | | | |
|---|---|---|---|
| | shall return to the application with all isochronous segments received in entirety during the target Service Intervals. | | |
| **Subsection reference: 5.10.2.2 MA USB device requirements** | | | |
| 5.10.2.2#1 | In response to an IsochTransferReq packet that indicates any Presentation Time more than aMaxFrameDistance USB frames before or after the current MA USB Global Time, the target MA USB device shall send an IsochTransferResp packet with the same Request ID as the IsochTransferReq packet, and with the Status Code field set to ISOCH_TIME_INVALID. | TBD | |
| 5.10.2.2#2 | In response to an IsochTransferReq packet that indicates a valid Presentation Time, but the presentation time points to a time before the current MGT, or the presentation time falls within pMaxDeviceIsochINProgDelay of the current MGT, the target MA USB device shall send an IsochTransferResp packet with the same Request ID as the IsochTransferReq packet, and with the Status Code field set to ISOCH_TIME_EXPIRED. | TBD | |
| **Subsection reference: 5.10.2.3 Application design guidelines** | | | |
| **Subsection reference: 5.10.3 Isochronous OUT transfers** | | | |
| 5.10.3#1 | The MA USB host executes an isochronous OUT transfer by sending one or more IsochTransferReq packets to the target MA USB device. | TBD | |
| 5.10.3#2 | Each IsochTransferReq packet indicates the target isochronous endpoint on the MA USB device, the Request ID assigned to the transfer by the MA USB host, the number of isochronous segments that are being transmitted, and the beginning time of the first Service Interval or ASAP delivery option. | TBD | |
| 5.10.3#3 | The request ID assigned to the Isochronous OUT transfer shall be unique within the scope of the target endpoint. | TBD | |
| 5.10.3#4 | The Request ID starts at zero for the first isochronous OUT transfer after any configuration event intended to return the state of an endpoint flow to the initial state. | TD 5.7 | |
| 5.10.3#5 | The Request ID for an Isochronous OUT request is incremented by 1 for each successive request. | TD 5.6 | |
| 5.10.3#6 | The MA USB host shall not have more than one pending isochronous OUT transfer using the same Request ID and targeting the same endpoint. | TBD | |
| 5.10.3#7 | The rules for assigning Sequence Numbers to successive IsochTransferReq packets belonging to the same transfer are also the same as other MA USB OUT transfers, except the Sequence Number field is reset | TBD | |

| | | | |
|---|---|---|---|
| | to zero for every new MA USB isochronous OUT transfer. | | |
| 5.10.3#8 | The Presentation Time field in successive IsochTransferReq packets that target the same endpoint and do not indicate ASAP delivery shall be strictly increasing. | TBD | |
| 5.10.3#9 | Each IsochTransferReq packet carries one or more partial or complete isochronous segments subject to the packetization rules defined in section 5.10.1 | TBD | |
| 5.10.3#10 | In response to each IsochTransferReq packet, the target MA USB device programs its local resources to write the target endpoint during the target Service Intervals. | TBD | |
| 5.10.3#11 | The Presentation Time field in the IsochTransferResp packet carries the actual delivery time of the first isochronous segment, except when the target MA USB device experiences an error during the transfer, in which case, the Presentation Time field is reserved and set to 0. | TBD | |
| **Subsection reference: 5.10.3.1 MA USB host requirements** | | | |
| 5.10.3.1#1 | The MA USB host PAL shall not release an IsochTransferReq to the network that indicates a start time (Presentation Time) more than aMaxFrameDistance USB frames before or after the Global Time at the moment of the first bit of the packet is released to the network. | TBD | |
| 5.10.3.1#2 | The MA USB host PAL shall conclude a pending MA USB isochronous OUT transfer when it receives an IsochTransferResp packet that belongs to the transfer and has the EoT subfield set to1, or when it receives an IsochTransferResp packet that belongs to the transfer and indicates a non-recoverable error, or at pMaxDeviceIsochOUTRespDelay+aMaxIsochLinkDelay time units after the end of the last Service Interval targeted by the transfer. | TBD | |
| 5.10.3.1#3 | Once all pending MA USB isochronous OUT transfers corresponding to an application-level isochronous write request are concluded, the MA USB host PAL shall return the application the status of the isochronous write operation. | TBD | |
| **Subsection reference: 5.10.3.2 MA USB device requirements** | | | |
| 5.10.3.2#1 | In response to an IsochTransferReq packet that indicates any Presentation Time more than aMaxFrameDistance USB frames before or after the current Global Time, the target MA USB device shall send an IsochTransferResp packet with the same Request ID as the IsochTransferReq packet, and with the Status Code field set to ISOCH_TIME_INVALID. | TBD | |
| 5.10.3.2#2 | In response to an IsochTransferReq packet that indicates a valid Presentation Time, but the presentation time | TBD | |

| | | | |
|---|---|---|---|
| | points to a time before the MGT, or the presentation time falls within pMaxDeviceIsochOUTProgDelay of the current MGT, the target MA USB device shall send an IsochTransferResp packet with the same Request ID as the IsochTransferReq packet, and with the Status Code field set to ISOCH_TIME_EXPIRED. | | |
| **Subsection reference: 5.10.3.3 Application design guidelines** | | | |
| **Subsection reference: 5.11 Device notifications** | | | |
| 5.11#1 | The MA USB devices use the DevNotificationReq packet to carry the device notifications to the MA USB host. | N/A | |
| 5.11#2 | The MA USB host shall respond to a DevNotificationReq packet with a DevNotificationResp packet to inform the device whether the Device Notification Request was successfully received. | TD 5.1 | |
| **Subsection reference: 5.12 Reliability** | | | |
| **Subsection reference: 5.13 Efficiency** | | | |

# *Chapter 6 Test Assertions*

| Assertion # | Assertion Description | Test # | Comments |
|---|---|---|---|
| **Subsection reference: 6.1 Packet Types** | | | |
| 6.1#1 | An MA USB packet shall be no larger than 64KB. | GTP | *Original Text: MA USB packets can be as large as 64 KB.* |
| **Subsection reference: 6.2 Packet Formats** | | | |
| 6.2#1 | All packet headers have a length (in bytes) that is a multiple of 4, and are defined as a series of 4-byte data units known as double words (DWORDs). | Interop | |
| 6.2#2 | No header field is larger than a single DWORD in size, and no header field crosses DWORD boundaries. | N/A | No test required |
| 6.2#3 | The payload of the packet, if present, starts at bit number 0 of the first DWORD following the header fields. | N/A | No test required |
| 6.2#4 | The receiver shall ignore the value of a reserved field in the received packets. | TD 6.30 | |
| **Subsection reference 6.2.1 Common Header Fields** | | | |
| **Subsection reference 6.2.1.1 Version** | | | |
| 6.2.1.1#1 | MA USB 1.0 devices and hosts shall set the Version field to 0000b at transmit. | GTP | |
| **Subsection reference 6.2.1.2 Flags** | | | |
| 6.2.1.2#1 | The host field is set to 1 if the packet is sent by the host, and set to 0 otherwise. | GTP<br>TD 5.17 | |
| 6.2.1.2#2 | The Retry field is set to 1 if the packet is a retry, and set to 0 otherwise. | TD 5.3<br>TD 6.13 | |
| 6.2.1.2#3 | The Timestamp field is set to 1 if the packet header includes MA USB Timestamp and Media Time/Transmission Delay fields, and set to 0 otherwise. | TBD | |
| 6.2.1.2#4 | The fourth bit in the Flags field is reserved and set to 0. | GTP | |
| **Subsection reference 6.2.1.3 Type and Subtype** | | | |
| 6.2.1.3#1 | The Type field does not contain a value of 11b. | GTP | |
| 6.2.1.3#2 | Valid Type and Subtype combinations are listed in Table 5. | GTP | |
| **Subsection reference 6.2.1.4 Length** | | | |
| 6.2.1.4#1 | The 16-bit Length field carries the length of the MA USB packet, including the packet header, in bytes. | GTP | |

| | | | |
|---|---|---|---|
| **Subsection reference 6.2.1.5 EP Handle/Device Handle** | | | |
| 6.2.1.5#1 | Management packets have a Device Handle field that carries a handle for a USB device. | Interop | |
| 6.2.1.5#2 | Data and Control packets have an EP Handle field that carries the handle for a USB endpoint. | Interop | |
| 6.2.1.5#3 | The Bus Number subfield in the EP Handle field carries the USB bus number for the EP assigned by the MA USB device PAL. | Interop | |
| 6.2.1.5#4 | For virtual USB devices, the Bus Number subfield in the EP Handle field is set to the reserved value of 15. | Interop | |
| 6.2.1.5#5 | The USB Device Address subfield in the EP Handle field carries the USB address of the USB device to which the EP belongs. | Interop | |
| 6.2.1.5#6 | The USB address is allocated by the MA USB device. | Interop | |
| 6.2.1.5#7 | Prior to allocation of the USB address to the device, the USB Device Address field is set to the default value of 0. | Interop | |
| 6.2.1.5#8 | The EP Number subfield in the EP Handle field carries the EP number as defined in [USB 2.0] | Interop | |
| 6.2.1.5#9 | Prior to allocating an address, the only valid EP Number is zero. | Interop | |
| 6.2.1.5#10 | The Direction (D) subfield carries the direction of the EP as defined in [USB 2.0]. | Interop | |
| **Subsection reference 6.2.1.6 MA USB Device Address** | | | |
| 6.2.1.6#1 | The 8-bit MA USB Device Address field carries the address assigned by an MA USB host to an MA USB device, or a value of 0xFF to indicate any MA USB device within the MSS. | Interop<br>TD 6.14 | |
| **Subsection reference 6.2.1.7 SSID** | | | |
| 6.2.1.7#1 | The 8-bit SSID field identifies the MA USB Service Set (MSS) to which the target MA USB device belongs. | Interop | |
| **Subsection reference 6.2.1.8 Status Code** | | | |
| 6.2.1.8#1 | If an operation is successful, or involves no error, the Status Code is set to 0. Otherwise, depending on the operation, one of the values in Table 6 is returned in a response packet. | GTP | |
| 6.2.1.8#2 | Management packets with Status Code field set to values other than 0, shall still carry all the fields defined for the packet subtype. | GTP | |
| **Subsection reference 6.3 Management Packets** | | | |

| | | | |
|---|---|---|---|
| 6.3#1 | Bits 18 through 31 of DWORD 2 in an MA USB Management packet header are reserved and set to 0. | GTP | |
| **Subsection reference 6.3.1 Common Header Fields** | | | |
| **Subsection reference 6.3.1.1 Dialog Token** | | | |
| 6.3.1.1#1 | The Dialog Token value is incremented by 1 after transmitting each new management packet carrying a request, with wraparound to 1 after reaching the maximum value of aMaxDialogToken. | GTP<br>TD 6.15 | |
| 6.3.1.1#2 | To avoid ambiguity in tracking the request management packets waiting for response, the MA USB PAL shall have no more than aMaxDialogToken/2 (half the size of the Dialog Token space) outstanding management requests. | N/A | Not tested |
| 6.3.1.1#3 | A management packet carrying an unexpected Dialog Token value shall be ignored by the recipient. | TD 6.15 | |
| 6.3.1.1#4 | The Dialog Token field is set to 0 for management packets with broadcast MA USB Device Address, and other management packets that do not require a response correlated with the request. | GTP | |
| **Subsection 6.3.2 MA USB Capability Request (CapReq)** | | | |
| 6.3.2#1 | The Type and Subtype fields in an MA USB Capability Request packet are set to 0 (Management) and 0 (CapReq), respectively. | TD 5.18 | |
| 6.3.2#2 | The Status Code field in an MA USB Capability Request packet is set to 0 (NO_ERROR). | TD 5.18 | |
| 6.3.2#3 | The Device Handle field in an MA USB Capability Request packet is reserved and set to 0. | TD 5.18 | |
| 6.3.2#4 | The Number of Outstanding Management Requests field indicates the maximum number of device initiated outstanding management requests that the MA USB host can track. | Not tested | |
| 6.3.2#5 | An MA USB host should be able to track at least 127 outstanding requests per device it supports. | TD 5.18 | Non-compliance generates warning only<br>*Original Text: The MA USB host is recommended to be able to track at least 127 outstanding requests per each MA USB device it supports.* |

| | | | |
|---|---|---|---|
| 6.3.2#6 | Bits 12 through 31 of DWORD 3 in an MA USB Capability Request are be reserved and set to 0. | TD 5.18 | |
| 6.3.2#7 | The Length field in an MA Host Capability descriptor indicates the length of the descriptor in bytes. | TD 5.18 | |
| 6.3.2#8 | The MA Host Capability Type field in an MA Host Capability descriptor indicates the type of the descriptor. Descriptor types are listed in Table 9. | TD 5.18 | |
| **Subsection reference 6.3.2.1 Synchronization Capabilities Descriptor** | | | |
| 6.3.2.1#1 | The Synchronization Capabilities descriptor reports the synchronization related capabilities of the MA USB host and shall be present if the MA USB host has access to the Media Time. | TD 5.18.1 | |
| 6.3.2.1#2 | The Length field indicates the length of the descriptor in bytes and is set to 3. | TD 5.18.1 | |
| 6.3.2.1#3 | The MA Host Capability Type field indicates the type of the descriptor and is set to 3. | TD 5.18.1 | |
| 6.3.2.1#4 | The Media Time Available field in a Synchronization Capabilities descriptor is set to 1 if the MA USB host has access to Media Time. Otherwise, the Media Time Available field is set to 0. | TD 5.18.1 | |
| 6.3.2.1#5 | The last 7 bits in a Synchronization Capabilities descriptor are reserved and set to 0. | TD 5.18.1 | |
| **Subsection reference 6.3.2.2 Link Sleep Capability Descriptor** | | | |
| 6.3.2.2#1 | The absence of a Link Sleep Capability Descriptor is equivalent to the Link Sleep Capable field set to 0. | TD 5.18.2 | |
| 6.3.2.2#2 | The Length field indicates the length of the descriptor in bytes and is set to 3. | TD 5.18.2 | |
| 6.3.2.2#3 | The MA Host Capability Type field indicates the type of the descriptor and is set to 5. | TD 5.18.2 | |
| 6.3.2.2#4 | The Link Sleep Capable field in a Link Sleep Capability descriptor indicates whether the MA USB host can receive a Sleep Request packet from a target MA USB device to take the session state to Session Inactive without suspending the integrated USB device and is set to 1 if the MA USB host can receive a Sleep Request packet without USB suspend. Otherwise, the Link Sleep Capable field is set to 0. | TD 5.18.2<br>TD 5.19 | |
| 6.3.2.2#5 | The last 7 bits of a Link Sleep Capability descriptor are reserved and set to 0. | TD 5.18.2 | |
| **Subsection Reference 6.3.3 MA USB Capability Response (CapResp)** | | | |

| | | | |
|---|---|---|---|
| 6.3.3#1 | An MA USB Capability Response (CapResp) packet is transmitted by the target MA USB device in response to an MA USB Capability Request (CapReq) packet. | TD 6.16 | |
| 6.3.3#2 | The Device Handle field in an MA USB Capability Response packet is reserved and set to 0. | TD 6.16 | |
| 6.3.3#3 | The Type and Subtype fields in an MA USB Capability Response Packet are set to 0 (Management) and 1 (CapResp), respectively. | TD 6.16 | |
| 6.3.3#4 | The Status Code field of an MA USB Capability Response packet indicates whether the request was successfully completed. | TD 6.16 | |
| 6.3.3#5 | The Number of Endpoints field indicates the maximum number of endpoints for which the MA USB device can track state. | TD 6.16 | |
| 6.3.3#6 | For MA USB hubs, the minimum value of the Number of Endpoints field is 16. | TD 6.16.1 | |
| 6.3.3#7 | The Number of Devices field indicates the maximum number of USB devices the MA USB device can manage. | TD 6.16 | |
| 6.3.3#8 | Only MA USB hubs can manage multiple USB devices. | TD 6.16.2 | |
| 6.3.3#9 | The Number of Streams field in an MA USB Capability Response packet carries a value where 2 to the power of the value of this field is the maximum number of streams supported by the MA USB device for any of its Enhanced SuperSpeed bulk endpoints. | N/A | Not tested |
| 6.3.3#10 | The Device Type field in an MA USB Capability Response packet is 0 if the MA USB device is not an MA USB hub. | TD 6.16.2 | |
| 6.3.3#11 | The Device Type field in an MA USB Capability Response packet is 1 if the MA USB device is an MA USB hub with an integrated USB 2.0 hub or 2 if the MA USB device is an MA USB hub with an integrated USB 3.1 hub. | TD 6.16.1 | |
| 6.3.3#12 | Values 3 - 8 for the Device Type field in the Capability Response packet are reserved. | TD 6.16 | |
| 6.3.3#13 | The Descriptors Count field indicates the total number of MA USB Device Capabilities descriptors present. | TD 6.16 | |
| 6.3.3#14 | The Descriptors Length field indicates the total size of MA Device Capabilities descriptors in bytes. | TD 6.16 | |
| 6.3.3#15 | The Number of Outstanding Transfer Requests field indicates the maximum number of total outstanding transfer requests that the MA USB device can track. | N/A | Not tested |

| | | | |
|---|---|---|---|
| 6.3.3#16 | The Number of Outstanding Management Requests indicates the maximum number of host initiated outstanding management requests that MA USB device can track. | N/A | Not tested |
| 6.3.3#17 | The Number of Outstanding Management requests an MA USB device supports should be at least 2 times the maximum number of endpoints plus 2 times the total number of devices it supports. | TD 6.16 | |
| 6.3.3#18 | Bits 28 through 31 of DWORD 5 in an MA USB Capability Response packet are reserved and set to 0. | TD 6.16 | |
| 6.3.3#19 | The Length field in an MA Device Capability descriptor indicates the length of the descriptor in bytes. | TD 6.16 | |
| 6.3.3#20 | The MA Device Capability Type field in an MA Device Capability descriptor indicates the type of the descriptor. Descriptor types are listed in Table 14. | TD 6.16 | |
| 6.3.3#21 | If the Device Type field is set to 0 or 2, the Speed Capability descriptor shall be present. | TD 6.16 | |
| 6.3.3#22 | A P-managed OUT Capabilities type descriptor shall be present if the MA USB device supports any of the P-managed OUT optional capabilities. | TD 6.16 | |
| 6.3.3#23 | An Isochronous Capabilities type descriptor shall be present if the Device Type field is not set to 0, or if the MA USB device supports isochronous endpoints. | TD 6.16 | |
| 6.3.3#24 | A Synchronization Capabilities type descriptor shall be present if the Device Type field is not set to 0, or if the MA USB device supports isochronous endpoints. | TD 6.16 | |
| 6.3.3#25 | A Container ID Capability type descriptor shall be present if the integrated USB device supports Container ID descriptor. | TD 6.16 | |
| 6.3.3#26 | A Link Sleep Capability type descriptor shall be present if the MA USB device supports transition of the session state to Session Inactive without the integrated USB device being suspended. | TD 6.16 | |
| **Subsection reference 6.3.3.1 Speed Capability Descriptor** | | | |
| 6.3.3.1#1 | If the Device Type field is set to 2, the CapResp packet carries the Speed Capability descriptor for the integrated Enhanced SuperSpeed hub in the MA USB hub. | TD 6.16.1 | |
| 6.3.3.1#2 | The Length field indicates the length of the descriptor and is set to 4. | TD 6.16.3 | |
| 6.3.3.1#3 | The MA Capability Type field indicates the type of descriptor and is set to 0. | TD 6.16.3 | |
| 6.3.3.1#4 | Bits 16 through 19 of a Speed Capability descriptor are reserved and set to 0. | TD 6.16.3 | |

| | | | |
|---|---|---|---|
| 6.3.3.1#5 | The Speed field indicates the speed of the USB device behind the MA USB device. | TD 6.16.3 | |
| 6.3.3.1#6 | Values 5 - 15 for the Speed field are reserved. | TD 6.16.3 | |
| 6.3.3.1#7 | Bits 24 through 27 of a Speed Capability descriptor are reserved and set to 0. | TD 6.16.3 | |
| 6.3.3.1#8 | The Lane Speed Exponent (LSE) field indicates the LSE of the USB device as defined in [USB 3.1] if the Speed field is set to SuperSpeed or SuperSpeedPlus. Reserved otherwise. | TD 6.16.3 | |
| 6.3.3.1#9 | Bits 31 and 32 of a Speed Capability descriptor are reserved and set to 0. | TD 6.16.3 | |
| **Subsection reference 6.3.3.2 P-managed OUT Capabilities Descriptor** | | | |
| 6.3.3.2#1 | The Length field indicates the length of the descriptor in bytes and is set to 3. | TD 6.16.4 | |
| 6.3.3.2#2 | The MA Capability field indicates the type of descriptor and is set to 1. | TD 6.16.4 | |
| 6.3.3.2#3 | The Elastic Buffer Capability field is set to 1 if the MA USB host may transmit data if no credit is available or 0 otherwise. | TD 6.16.4 | |
| 6.3.3.2#4 | The Drop Notification field is set to 1 if an MA USB device may return a DROPPED_PACKET status or 0 otherwise. | TD 6.16.4 | |
| 6.3.3.2#5 | If the Drop Notification field is set to 0, an MA USB device shall not return a DROPPED_PACKET status. | GTP | |
| 6.3.3.2#6 | Bits 18 through 23 of a P-managed OUT Capabilities descriptor are reserved and set to 0. | TD 6.16.4 | |
| **Subsection reference 6.3.3.3 Isochronous Capabilities Descriptor** | | | |
| 6.3.3.3#1 | The Length field indicates the length of the descriptor in bytes and is set to 3. | TD 6.16.5 | |
| 6.3.3.3#2 | The MA Capability Type field indicates the type of the descriptor and is set to 2. | TD 6.16.5 | |
| 6.3.3.3#3 | The Isochronous Payload Alignment field is set to 1 if the MA USB device transmits and expects to receive DWORD-aligned isochronous payload, or is otherwise set to 0. | TD 6.16.5 | |
| 6.3.3.3#4 | Bits 17 through 23 of an Isochronous Capabilities descriptor are reserved and set to 0. | TD 6.16.5 | |
| **Subsection reference 6.3.3.4 Synchronization Capabilities Descriptor** | | | |
| 6.3.3.4#1 | The Length field indicates the length of the descriptor in bytes and is set to 3. | TD 6.16.6 | |

| | | | |
|---|---|---|---|
| 6.3.3.4#2 | The MA Capability Type field indicates the type of descriptor and is set to 3. | TD 6.16.6 | |
| 6.3.3.4#3 | The Media Time field is set to 1 if the MA USB device has access to Media Time or is 0 otherwise. | TD 6.16.6 | |
| 6.3.3.4#4 | The Timestamp Request field is set to 1 to indicate the need to receive MA USB timestamps regardless of endpoint configuration or 0 to indicate the need to receive MA USB timestamps only when the MA USB device has isochronous endpoints selected. | TD 6.16.6 | |
| 6.3.3.4#5 | MA USB hubs shall set the Timestamp Request field to 1. | TD 6.16.6 | |
| 6.3.3.4#6 | Bits 18 through 23 of a Synchronization Capabilities descriptor are reserved and set to 0. | TD 6.16.6 | |
| **Subsection reference 6.3.3.5 Container ID Capability Descriptor** | | | |
| 6.3.3.5#1 | The Length field indicates the length of the descriptor in bytes and is set to 18. | TD 6.16.7 | |
| 6.3.3.5#2 | The MA Capability Type field indicates the type of the descriptor and is set to 4. | TD 6.16.7 | |
| 6.3.3.5#3 | The Container ID field indicates the value of the Container ID. | TD 6.16.7 | |
| **Subsection reference 6.3.3.6 Link Sleep Capability Descriptor** | | | |
| 6.3.3.6#1 | The Length field indicates the length of the descriptor in bytes and is set to 3. | TD 6.16.8 | |
| 6.3.3.6#2 | The MA Device Capability Type field indicates the type of the descriptor and is set to 5. | TD 6.16.8 | |
| 6.3.3.6#3 | The Link Sleep Capable field is set to 1 if an MA USB device can receive a Sleep Request packet without USB suspend or 0 otherwise. | TD 5.19<br>TD 6.16.8 | |
| 6.3.3.6#4 | Bits 17 through 23 of a Link Sleep Capability descriptor are reserved and set to 0. | TD 6.16.8 | |
| **Subsection reference 6.3.4 USB Device Handle Request (USBDevHandleReq)** | | | |
| 6.3.4#1 | The Type and Subtype fields in a USB Device Handle Request packet are set to 0 (Management) and 2 (USBDevHandleReq), respectively. | TD 5.18 | |
| 6.3.4#2 | The Status Code field in a USB Device Handle Request packet is set to 0 (NO_ERROR). | TD 5.18 | |
| 6.3.4#3 | The Device Handle field in a USB Device Handle Request packet is reserved and set to 0. | TD 5.18 | |
| 6.3.4#4 | The MA USB Route String field Identifies the USB topology as defined in Section 4.2 and the corresponding route string, as defined in [USB 3.1]. | Interop | |

| | | | |
|---|---|---|---|
| 6.3.4#5 | The Speed field identifies the speed of the USB device. | TD 5.18 | |
| 6.3.4#6 | The Speed field does not contain a value greater than 4. | TD 5.18 | |
| 6.3.4#7 | Bits 24 through 31 of DWORD 3 are reserved and set to 0. | TD 5.18 | |
| 6.3.4#8 | The Hub field identifies the device handle of the hub inside the MA USB hub through which the USB device is accessed. Reserved and set to 0 for integrated USB devices. | TD 5.18 | |
| 6.3.4#9 | Bits 16 through 31 of DWORD 4 are reserved and set to 0. | TD 5.18 | |
| 6.3.4#10 | If the device is LS or FS and is accessed through an HS hub, the Parent HS Hub field identifies the device handle of the “parent” HS hub that isolates LS or FS signaling on its downstream facing port from HS signaling on its upstream facing port. Reserved and set to 0 for other device speeds. | TD 5.18 | |
| 6.3.4#11 | If the device is LS or FS accessed through an HS hub, the Parent HS Hub Port field identifies the port number on the parent HS hub to which the LS or FS device is directly attached. Reserved and set to 0 for other device speeds. | TD 5.18 | |
| 6.3.4#12 | The Multiple Transaction Translators field is set to 1 for an LS or FS USB device, if it is connected through an HS hub that has Multiple TTs support enabled by Software or set to 0 otherwise. | Interop | |
| 6.3.4#13 | The Lane Speed Exponent (LSE) field indicates the LSE of the USB device as defined in [USB 3.1] if the Speed field is set to SuperSpeed or SuperSpeedPlus. Reserved and set to 0 otherwise. | TD 5.18 | |
| 6.3.4#14 | Bits 23 through 31 of DWORD 4 are reserved and set to 0. | TD 5.18 | |
| **Subsection reference 6.3.5 USB Device Handle Response (USBDevHandleResp)** | | | |
| 6.3.5#1 | The USB Device Handle Request (USBDevHandleResp) packet is transmitted by the target MA USB device to the MA USB host in response to a USBDevHandleReq packet. | TD 6.16 | |
| 6.3.5#2 | The Type and Subtype fields are set to 0 (Management) and 3 (USBDevHandleResp), respectively. | TD 6.16 | |
| 6.3.5#3 | The Device Handle field is reserved and set to 0. | TD 6.16 | |
| 6.3.5#4 | The Status Code field indicates whether the request was successfully completed. | TD 6.16 | |
| 6.3.5#5 | The USB Device Handle field carries the handle of the USB device. | TD 6.16 | |
| 6.3.5#6 | Bits 16 through 31 of DWORD 3 in a USB Device Handle response packet are reserved and set to 0. | TD 6.16 | |

| | | | |
|---|---|---|---|
| **Subsection reference 6.3.6 Endpoint Handle Request (EPHandleReq)** | | | |
| 6.3.6#1 | The Type and Subtype fields are set to 0 (Management) and 4 (EPHandleReq), respectively. | TD 5.18 | |
| 6.3.6#2 | The Status Code field is set to 0 (NO_ERROR). | TD 5.18 | |
| 6.3.6#3 | The Device Handle field carries the handle of the USB device behind the MA USB device for which the endpoint handle list is requested. | N/A | Not testable |
| 6.3.6#4 | The Number of EP Descriptors field indicates the number of EP descriptors in the Endpoint Handle Request packet. | TD 5.18 | |
| 6.3.6#5 | The Size of EP Descriptor field indicates the size of each EP descriptor included in the Endpoint Handle Request packet, in bytes. | TD 5.18 | |
| 6.3.6#6 | Bits 11 through 31 of an Endpoint Handle Request packet are reserved and set to 0. | TD 5.18 | |
| 6.3.6#7 | For an LS, FS, or HS device, the EP descriptor takes 8 bytes (including 1 byte of zero padding). | TD 5.18 | |
| 6.3.6#8 | For an Enhanced SuperSpeed device the EP descriptor takes 16 bytes (including 3 bytes of zero padding). | TD 5.18 | |
| 6.3.6#9 | For an Enhanced SuperSpeed device operating at above Gen 1 speed with isochronous endpoint(s), the EP descriptor takes 24 bytes (including 3 bytes of zero padding). | TD 5.18 | |
| **Subsection reference 6.3.7 Endpoint Handle Response (EPHandleResp)** | | | |
| 6.3.7#1 | The Endpoint Handle Response (EPHandleResp) packet is transmitted by the target MA USB device to the MA USB host, and includes the list of endpoint handles requested by the MA USB host. | TD 6.16 | |
| 6.3.7#2 | The Type and Subtype fields are set to 0 (Management) and 5 (EPHandleResp), respectively. | TD 6.16 | |
| 6.3.7#3 | The Device Handle field carries the handle of the USB device behind the MA USB device for which the endpoint handle list is returned. | N/A | Not testable |
| 6.3.7#4 | The Status Code field indicates whether the request was successfully completed. | TD 6.16 | |
| 6.3.7#5 | The request is considered successful if there is at least one endpoint for which EP handle is successfully allocated. | TD 6.16 | |
| 6.3.7#6 | The Number of MA USB EP Descriptors field indicates the number of MA USB EP descriptors carried in the packet and is set to the same value as the Number of | TD 6.16 | |

| | | | |
|---|---|---|---|
| | MA USB EP Descriptors field in the corresponding EPHandleReq packet. | | |
| 6.3.7#7 | Bits 5 through 31 in an Endpoint Handle Response packet are reserved and set to 0. | TD 6.16 | |
| 6.3.7#8 | The EPHandleResp packet carries one or more MA USB EP descriptors inserted in the same order as the corresponding EP descriptors in the EPHandleReq packet. | TD 6.16 | |
| 6.3.7#9 | The EP Handle field contains the handle of the endpoint requested in the EP Handle Request packet. | N/A | Not tested |
| 6.3.7#10 | The Direction field is set to 1 for an IN endpoint or 0 for a Control or OUT endpoint. | N/A | Not tested |
| 6.3.7#11 | The Isochronous field is set to 1 for an isochronous endpoint or 0 for a non-isochronous endpoint. | N/A | Not tested |
| 6.3.7#12 | The L-Managed field is set to 1 if L-managed transfers are supported, 0 if L-managed transfers are not supported, or reserved for endpoint types other than control and non-isochronous OUT. | N/A | Not tested |
| 6.3.7#13 | The Valid field indicates whether the handle for the endpoint was successfully allocated, and is set to 1 if the handle in the EP Handle field is not valid, or 0 if the handle is valid. | N/A | Not tested |
| 6.3.7#14 | Bits 20 through 31 of DWORD 0 in an MA USB EP descriptor are reserved and set to 0. | TD 6.16 | |
| 6.3.7#15 | For a control or non-isochronous OUT endpoint, the Credit Consumption Unit field indicates the buffer size in bytes that the MA USB host must assume as the unit of credit consumption for all p-managed transfers targeting the endpoint. The field is reserved for all other endpoints. | N/A | Not tested |
| 6.3.7#16 | Bits 16 through 31 of DWORD 1 in an MA USB EP descriptor are reserved and set to 0. | TD 6.16 | |
| 6.3.7#17 | For control and OUT endpoints, the Buffer Size field indicates the buffer size in bytes available to transfers targeting the endpoint. The field is reserved for IN endpoints. | TD 6.16 | |
| 6.3.7#18 | The Isochronous Programming Delay field is reserved for non-isochronous endpoints and set to 0. | TD 6.16 | |
| 6.3.7#19 | The Isochronous Response Delay field is reserved for non-isochronous endpoints and set to 0. | TD 6.16 | |
| **Subsection reference 6.3.8 Endpoint Activate Request (EPActivateReq)** | | | |
| 6.3.8#1 | The Type and Subtype fields are set to 0 (Management) and 6 (EPActivateReq), respectively. | TD 5.19 | |

| | | | |
|---|---|---|---|
| 6.3.8#2 | The Status Code field is set to 0 (NO_ERROR). | TD 5.19 | |
| 6.3.8#3 | The Device Handle field carries the handle of the USB device behind the MA USB device the request is targeting. | N/A | Not testable |
| 6.3.8#4 | The Number of EP Handles field carries the number of EP handles included in the packet. | TD 5.19 | |
| 6.3.8#5 | Bits 5 through 31 of DWORD 3 in an Endpoint Activate Request packet are reserved and set to 0. | TD 5.19 | |
| 6.3.8#6 | The EP Handle List field carries a list of EP handles that the MA USB host is requesting to activate, concatenated in 16-bit increments. | TD 5.19 | |
| **Subsection reference 6.3.9 Endpoint Activate Response (EPActivateResp)** | | | |
| 6.3.9#1 | The Endpoint Activate Response (EPActivateResp) packet is transmitted by the target MA USB device to the MA USB host in response to an EPActivateReq packet. | TD 6.17 | |
| 6.3.9#2 | The Type and Subtype fields are set to 0 (Management) and 7 (EPActivateResp), respectively. | TD 6.17 | |
| 6.3.9#3 | The Status Code field indicates whether the request was successfully completed. | TD 6.17 | |
| 6.3.9#4 | The Device Handle field carries the handle of the USB device behind the MA USB device for which the response is being returned. | N/A | Not testable |
| 6.3.9#5 | The Number of EP Handles with Error field carries the number of EP handles whose activation did not complete. | TD 6.17 | |
| 6.3.9#6 | The Number of EP Handles with Error field is nonzero only if the Status Code field in the packet carries a value other than SUCCESS. | TD 6.17 | |
| 6.3.9#7 | Bits 5 through 31 of DWORD 3 in an Endpoint Activate Response packet are reserved and set to zero. | TD 6.17 | |
| 6.3.9#8 | The EP Handle List field carries a list of EP handles whose activation failed, concatenated in 16-bit increments. | TD 6.17 | |
| **Subsection reference 6.3.10 Endpoint Inactivate Request (EPInactivateReq)** | | | |
| 6.3.10#1 | The Type and Subtype fields are set to 0 (Management) and 8 (EPInactivateReq), respectively. | TD 5.19<br>TD 5.22 | |
| 6.3.10#2 | The Status Code field is set to 0 (NO_ERROR). | TD 5.19<br>TD 5.22 | |
| 6.3.10#3 | The Device Handle field carries the handle of the USB device the request is targeting. | N/A | Not testable |

| | | | |
|---|---|---|---|
| 6.3.10#4 | The Number of EP Handles field carries the number of EP handles included in the packet. | TD 5.19<br>TD 5.22 | |
| 6.3.10#5 | The Suspend Flag field indicates whether the EPInactivateReq is issued due to suspension of the endpoints for which EP Handles are included and is set to 1 for endpoints suspended or 0 otherwise. | N/A | |
| 6.3.10#6 | Bits 6 through 31 of DWORD 3 in an Endpoint Inactivate request packet are reserved and set to 0. | TD 5.19<br>TD 5.22 | |
| 6.3.10#7 | The EP Handle List field carries a list of EP handles the MA USB host is requesting to inactivate, concatenated in 16-bit increments. | TD 5.19<br>TD 5.22 | |
| **Subsection reference 6.3.11 Endpoint Inactive Response (EPInactivateResp)** | | | |
| 6.3.11#1 | The Endpoint Inactivate Response (EPInactivateResp) packet is transmitted by the target MA USB device to the MA USB host in response to an EPInactivateReq packet. | TD 6.17 | |
| 6.3.11#2 | The Type and Subtype fields are set to 0 (Management) and 9 (EPInactivateResp), respectively. | TD 6.17 | |
| 6.3.11#3 | The Device Handle field carries the handle of the USB device for which the response is being returned. | N/A | Not testable |
| 6.3.11#4 | The Status Code field indicates whether the request was successfully completed. | TD 6.17 | |
| 6.3.11#5 | The Number of EP Handles with Error field carries the number of EP handles whose inactivation did not successfully complete. | TD 6.17 | |
| 6.3.11#6 | The Number of EP Handles with Error field is nonzero only if the Status Code field in the packet carries a value other than SUCCESS (NO_ERROR). | TD 6.17 | |
| 6.3.11#7 | Bits 5 through 31 of DWORD 3 in an Endpoint Inactivate Response packet are reserved and set to zero. | TD 6.17 | |
| 6.3.11#8 | The EP Handle List field carries a list of EP handles whose inactivation failed, concatenated in 16-bit increments. | TD 6.17 | |
| **Subsection reference 6.3.12 Endpoint Reset Request (EPResetReq)** | | | |
| 6.3.12#1 | The Type and Subtype fields are set to 0 (Management) and 10 (EPResetReq), respectively. | TD 5.21 | |
| 6.3.12#2 | Status Code field is set to 0 (NO_ERROR). | TD 5.21 | |
| 6.3.12#3 | The Device Handle field carries the handle of the USB device the request is targeting. | N/A | Not testable |

| | | | |
|---|---|---|---|
| 6.3.12#4 | The Number of EP Reset Information Blocks field carries the number of the EP reset information blocks included in the packet. | TD 5.21 | |
| 6.3.12#5 | Bits 5 through 31 of DWORD 3 in an Endpoint Reset Request packet are reserved and set to 0. | TD 5.21 | |
| 6.3.12#6 | The EP Handle field of an EP Reset Information Block contains the EP handle the MA USB host is requesting to reset. | TD 5.21 | |
| 6.3.12#7 | The Transfer State Preserve Flag field in an EP Reset Information Block is set to 1 if the value of the data toggle bit [USB 2.0] or sequence number [USB 3.1] is preserved, otherwise 0. | N/A | Not tested |
| 6.3.12#8 | If the value of the Transfer State Preserve Flag field is 0, the USB controller hardware should invalidate any cached Transfer Descriptors and fetch the next Transfer Request Block when endpoint transits from stopped to running state. | N/A | |
| 6.3.12#9 | If the value of the Transfer State Preserve Flag field is 1, the USB controller hardware should retry last transaction once the endpoint is transitioned from stopped to running state, providing that no other commands were performed on the endpoint. | N/A | |
| 6.3.12#10 | Bits 17 through 31 in an EP Reset Information Block are reserved and set to 0. | TD 5.21 | |
| **Subsection reference 6.3.13 Endpoint Reset Response (EPResetResp)** | | | |
| 6.3.13#1 | The Endpoint Reset Response (EPResetResp) packet is transmitted by the target MA USB device to the MA USB host in response to an EPResetResp packet. | TD 6.18 | |
| 6.3.13#2 | The Type and Subtype fields are set to 0 (Management) and 11 (EPResetResp), respectively. | TD 6.18 | |
| 6.3.13#3 | The Device Handle field carries the handle of the USB device for which the response is being returned. | N/A | Not testable |
| 6.3.13#4 | The Status Code field indicates whether the request was successfully completed. | TD 6.18 | |
| 6.3.13#5 | The Number of EP Handles with Error field carries the number of EP handles whose reset did not successfully complete. | TD 6.18 | |
| 6.3.13#6 | The Number of EP Handles with Error field is nonzero only if the Status Code field in the packet carries a value other than 0. | TD 6.18 | |
| 6.3.13#7 | Bits 5 through 31 of DWORD 3 in an Endpoint Reset Response packet are reserved and set to 0. | TD 6.18 | |

| | | | |
|---|---|---|---|
| 6.3.13#8 | The EP Handle List field carries a list of EP handles whose reset failed, concatenated in 16-bit increments. | TD 6.18 | |
| **Subsection reference 6.3.14 Clear Transfer Request (ClearTransfersReq)** | | | |
| 6.3.14#1 | The Type and Subtype fields are set to 0 (Management) and 12 (ClearTransfersReq), respectively. | TD 5.22 | |
| 6.3.14#2 | The Status Code field is set to 0 (NO_ERROR). | TD 5.22 | |
| 6.3.14#3 | The Device Handle field carries the handle of the USB device the request is targeting. | N/A | Not testable |
| 6.3.14#4 | The Number of clear transfers information blocks field carries the number of clear transfers information blocks included in the packet. | TD 5.22 | |
| 6.3.14#5 | Bits 8 through 31 of DWORD 3 in an Endpoint Clear Transfer Request packet are reserved and set to 0. | TD 5.22 | |
| 6.3.14#6 | Each clear transfer information block has the EP Handle field set to the endpoint handle the request is targeting | TD 5.22 | |
| 6.3.14#7 | Each clear transfer information block has the Stream ID field set to indicate the target stream when the target endpoint is an Enhanced SuperSpeed bulk endpoint that supports the Enhanced SuperSpeed Stream Protocol; reserved otherwise. | TD 5.22 | |
| 6.3.14#8 | Each clear transfer information block has the Start Request ID field se to indicate the value of the Request ID that MA USB host uses following the ClearTransfersReq packet, which is not the target of this request | N/A | Not tested |
| 6.3.14#9 | Each clear transfer information block has bits 8 through 31 of DWORD 1 reserved and set to 0. | TD 5.22 | |
| **Subsection reference 6.3.15 Clear Transfer Response (ClearTransfersResp)** | | | |
| 6.3.15#1 | The Endpoint Clear Transfers Response (ClearTransfersResp) packet is transmitted by the target MA USB device to the MA USB host in response to a ClearTransfersReq packet. | TD 6.20 | |
| 6.3.15#2 | The Type and Subtype fields are set to 0 (Management) and 13 (ClearTransfersResp), respectively. | TD 6.20 | |
| 6.3.15#3 | The Device Handle field carries the handle of the USB device for which the response is being returned. | N/A | Not testable |
| 6.3.15#4 | The Status Code field indicates whether the request was successfully completed. | TD 6.20 | |
| 6.3.15#5 | The Number of cancel transfers status blocks field carries the number of cancel transfers status blocks shall be equal to the value of Number of cancel transfers | TD 6.20 | |

| | | | |
|---|---|---|---|
| | information blocks field in the corresponding ClearTransfersReq packet. | | |
| 6.3.15#6 | Bits 5 through 31 of DWORD 3 in an Endpoint Clear Transfer Response packet are reserved and set to 0. | TD 6.20 | |
| 6.3.15#7 | Each cancel transfer status block has the EP Handle field set to indicate the endpoint for which the status is being returned. | TD 6.20 | |
| 6.3.15#8 | Each cancel transfer status block has the Stream ID field set to indicate the stream to which the response is related when the endpoint identified in the EP Handle field is an Enhanced SuperSpeed bulk endpoint that supports the Enhanced SuperSpeed Stream Protocol; reserved otherwise. | TD 6.20 | |
| 6.3.15#9 | Each cancel transfer status block has the Cancellation Status field set to 1 if the request was successfully completed for the transfers identified by the EP Handle, Stream ID, and Start Request ID of the corresponding request or the Status Code field is set to SUCCESS. Set to 0 otherwise. | N/A | Not tested |
| 6.3.15#10 | Each cancel transfer status block has the Partial Delivery field set to 1 if the last cancelled transfer was not completed, i.e., some (but not all) data related to the transfer was moved to/from the device before its cancellation. | N/A | Not testable |
| 6.3.15#11 | Each cancel transfer status block has bits 2 through 31 of DWORD 1 reserved and set to 0. | TD 6.20 | |
| 6.3.15#12 | Each cancel transfer status block has the Last Request ID field set to indicate the value of the Request ID field in the last TransferResp packet created by the MA USB Device. | TD 6.20 | |
| 6.3.15#13 | For an OUT transfer, if the Partial Delivery field is set to 1, a cancel transfer status block has the Delivered Sequence Number field set to indicate the last sequence number delivered to the device. Otherwise the Delivered Sequence Number field is reserved and set to zero. | TD 6.20 | |
| 6.3.15#14 | For an OUT transfer, if the Partial Delivery field is set to 1, a cancel transfer status block has the Delivered Byte Offset field set to indicate the total amount of data related to the transfer delivered to the device identified by the sequence number value in the Delivered Sequence Number field. Otherwise the Delivered Byte Offset field is reserved and set to zero. | N/A | Not tested |
| **Subsection reference 6.3.16 Endpoint Handle Delete Request (EPHandleDeleteReq)** | | | |

| | | | |
|---|---|---|---|
| 6.3.16#1 | The Type and Subtype fields are set to 0 (Management) and 14 (EPHandleDeleteReq), respectively. | TD 5.22 | |
| 6.3.16#2 | The Status Code field is set to 0 (NO_ERROR). | TD 5.22 | |
| 6.3.16#3 | The Device Handle field carries the handle of the USB device the request is targeting. | N/A | Not testable |
| 6.3.16#4 | The Number of EP Handles field carries the number of the EP handles included in the packet. | TD 5.22 | |
| 6.3.16#5 | Bits 5 through 31 of DWORD 3 in an Endpoint Handle Delete Request packet are reserved and set to 0. | TD 5.22 | |
| 6.3.16#6 | The EP Handle List field carries a list of EP handles the MA USB host is requesting to delete, concatenated in 16-bit increments. | TD 5.22 | |
| **Subsection reference 6.3.17 Endpoint Handle Delete Response (EPHandleDeleteResp)** | | | |
| 6.3.17#1 | The Endpoint Handle Delete Response (EPHandleDeleteResp) packet is transmitted by the target MA USB device to the MA USB host in response to an EPHandleDeleteReq packet. | TD 6.20 | |
| 6.3.17#2 | The Type and Subtype fields are set to 0 (Management) and 15 (EPHandleDeleteResp), respectively. | TD 6.20 | |
| 6.3.17#3 | The Device Handle field carries the handle of the USB device for which the response is being returned. | N/A | Not testable |
| 6.3.17#4 | The Status Code field indicates whether the request was successfully completed. | TD 6.20 | |
| 6.3.17#5 | The Number of EP Handles with Error field carries the number of EP handles that could not be deleted. | TD 6.20 | |
| 6.3.17#6 | Bits 5 through 31 of DWORD 3 in an Endpoint Handle Delete Response packet are reserved and set to 0. | TD 6.20 | |
| 6.3.17#7 | The EP Handle List field carries a list of EP handles that could not be cleared, concatenated in 16-bit increments. | TD 6.20 | |
| **Subsection reference 6.3.18 MA USB Device Reset Request (DevResetReq)** | | | |
| 6.3.18#1 | The Type and Subtype fields are set to 0 (Management) and 16 (DevResetReq), respectively | TD 5.18 | |
| 6.3.18#2 | The Status Code field is set to 0 (NO_ERROR). | TD 5.18 | |
| 6.3.18#3 | The Device Handle and Dialog Token fields are reserved and set to 0. | TD 5.18<br>TD 6.15 | |
| **Subsection reference 6.3.19 MA USB Device Reset Response (DevResetResp)** | | | |
| 6.3.19#1 | The MA USB Device Reset Response (DevResetResp) packet is transmitted by a target MA USB device to | TD 6.16 | |

| | | | |
|---|---|---|---|
| | the MA USB host in response to a DevResetReq packet. | | |
| 6.3.19#2 | The Type and Subtype fields are set to 0 (Management) and 17 (DevResetResp), respectively. | TD 6.16 | |
| 6.3.19#3 | The Device Handle and Dialog Token fields are reserved and set to 0. | TD 6.16 | |
| 6.3.19#4 | The Status Code field indicates whether the request was successfully completed. | TD 6.16 | |
| **Subsection reference 6.3.20 Modify EP0 Request (ModifyEP0Req)** | | | |
| 6.3.20#1 | The Type and Subtype fields are set to 0 (Management) and 18 (ModifyEP0Req), respectively. | TD 5.18 | |
| 6.3.20#2 | The Status Code field is set to 0 (NO_ERROR). | TD 5.18 | |
| 6.3.20#3 | The Device Handle field carries the handle of the USB device the request is targeting. | N/A | Not testable |
| 6.3.20#4 | The EP0 Handle field carries the EP0 handle of the USB device identified by the Device Handle field. | N/A | Not testable |
| 6.3.20#5 | The Max Packet Size field carries the updated value of the maximum packet size for the EP0 handle in bytes. | N/A | Not tested |
| **Subsection Reference 6.3.21 Modify EP0 Response (ModifyEP0Resp)** | | | |
| 6.3.21#1 | The Modify EP0 Response (ModifyEP0Resp) packet is transmitted by the target MA USB device to the MA USB host in response to a ModifyEP0Req packet. | TD 6.16 | |
| 6.3.21#2 | The Type and Subtype fields are set to 0 (Management) and 19 (ModifyEP0Resp), respectively. | TD 6.16 | |
| 6.3.21#3 | The Device Handle field carries the handle of the USB device for which the response is being returned. | N/A | Not testable |
| 6.3.21#4 | The Status Code field indicates whether the request was successfully completed. | TD 6.16 | |
| 6.3.21#5 | An EP0 Handle is present only if: (a) the USB Address subfield of the EP0 Handle field in the corresponding Modify EP0 Request packet is set to 0 and the USB device identified by the Device Handle field is in the addressed state; or (b) the USB Address subfield of the EP0 Handle field in the corresponding Modify EP0 Request packet is not set to 0 and the USB device identified by the Device Handle field is in the default state. | TBD | |
| 6.3.21#6 | If the USB Address subfield of the EP0 Handle field in the corresponding Modify EP0 Request packet is set to 0 and the USB device identified by the Device Handle field is in the addressed state, a Modify EP0 Response packet carries in the EP0 Handle field, the | TBD | |

| | | | |
|---|---|---|---|
| | EP0 handle of the USB device with a nonzero USB Address subfield. | | |
| 6.3.21#7 | If the USB Address subfield of the EP0 Handle field in the corresponding Modify EP0 Request packet is not set to 0 and the USB device identified by the Device Handle field is in the default state, a Modify EP0 Response packet carries in the EP0 Handle field, the EP0 handle of the USB device with a USB Address subfield set to 0. | TBD | |
| 6.3.21#8 | Bits 16 through 31 of a Modify EP0 Response packet are reserved and set to 0. | TD 6.16 | |
| **Subsection reference 6.3.22 Set USB Device Address Request (SetUSBDevAddrReq)** | | | |
| 6.3.22#1 | The Type and Subtype fields are set to 0 (Management) and 20 (SetUSBDevAddrReq), respectively. | TD 5.18 | |
| 6.3.22#2 | The Status Code field is set to 0 (NO_ERROR). | TD 5.18 | |
| 6.3.22#3 | The Device Handle field carries the handle of the USB device the request is targeting. | N/A | Not testable |
| 6.3.22#4 | The Response Timeout field carries the duration in milliseconds by which the USB device behind the MA USB device is expected to respond to the USB Address Device command. | TD 5.18 | |
| 6.3.22#5 | If no response is received from the USB device in the expected response time the MA USB device shall respond to the SetUSBDevAddrReq with status code set to UNSUCCESSFUL. | N/A | Not testable |
| 6.3.22#6 | Bits 16 through 31 of DWORD 3 in a Set USB Device Address Request packet are reserved and set to 0. | TD 5.18 | |
| **Subsection reference 6.3.23 Set USB Device Address Response (SetUSBDevAddrResp)** | | | |
| 6.3.23#1 | The Set Device USB Address Response (SetUSBDevAddrResp) packet is transmitted by the target MA USB device to the MA USB host in response to a SetUSBDevAddrReq packet. | TD 6.16 | |
| 6.3.23#2 | The Type and Subtype fields are set to 0 (Management) and 21 (SetUSBDevAddrResp), respectively. | TD 6.16 | |
| 6.3.23#3 | The Device Handle field carries the handle of the USB device for which the response is being returned. | N/A | Not testable |
| 6.3.23#4 | The Status Code field indicates whether the request was successfully completed. | TD 6.16 | |
| 6.3.23#5 | The USB Device Address field carries the USB address of the USB device requested in the corresponding Set USB Device Address Request packet. | Interop | |

| | | | |
|---|---|---|---|
| 6.3.23#6 | Bits 7 through 31 of a Set USB Device Address Response packet are reserved and set to 0. | TD 6.16 | |
| **Subsection reference 6.3.24 Update Device Request (UpdateDevReq)** | | | |
| 6.3.24#1 | The Type and Subtype fields are set to 0 (Management) and 22 (UpdateDevReq), respectively. | TD 5.18 | |
| 6.3.24#2 | The Status Code field is set to 0 (NO_ERROR). | TD 5.18 | |
| 6.3.24#3 | The Device Handle field carries the handle of the USB device the request is targeting. | N/A | Not testable |
| 6.3.24#4 | The Max Exit Latency field carries the worst case exit latency for the links and hubs between the target USB device and the integrated USB hub on the MA USB hub. | TBD | |
| 6.3.24#5 | The Max Exit Latency is set to 0 if the integrated USB device is not a hub. | TD 5.18 | |
| 6.3.24#6 | The Hub field is set to 1 if the USB device is a hub, or set to 0 if the USB device in not a hub. | TD 5.18 | |
| 6.3.24#7 | The Number of Ports field carries the number of downstream facing ports of the USB device if the USB device is a hub. | TD 5.18 | |
| 6.3.24#8 | The Number of Ports field is reserved and set to 0 if the USB device is not a hub. | TD 5.18 | |
| 6.3.24#9 | The Multiple Transaction Translators field is set to 1 for an HS hub that has Multiple Transaction Translators (TTs) enabled by Software and set to 0 otherwise. | TBD | |
| 6.3.24#10 | If the USB device is a hub, the Transaction Translator Think Time field carries the time required by an HS hub to proceed to the next FS/LS transaction. | TBD | |
| 6.3.24#11 | The Transaction Translator Think Time field is reserved and set to 0 if the target USB device is not an HS hub. | TD 5.18 | |
| 6.3.24#12 | The Integrated Hub Latency field is set to 1 if the target USB device is a hub and the integrated hub latency is included in the value of the Max Exit Latency field. | N/A | Not testable |
| 6.3.24#13 | If the USB device is a hub, the Integrated hub Latency field indicates whether the latency associated with the integrated hub is included in the value of Max Exit Latency field. | N/A | Not testable |
| 6.3.24#14 | The Integrated Hub Latency field is reserved and set to 0 if the USB device is not a hub. | TD 5.18 | |
| 6.3.24#15 | Bits 25 through 31 of DWORD 3 of an Update Device Request packet are reserved and set to 0. | TD 5.18 | |
| 6.3.24#16 | The UpdateDevReq packet carries the device descriptor starting at bit 0 of DWORD 4. | TD 5.18 | |

| | | | |
|---|---|---|---|
| 6.3.24#17 | The device descriptor contains an 18-byte standard device descriptor as defined in USB 2.0 or USB 3.1. | TD 5.18 | |
| 6.3.24#18 | Bits 16 through 31 of DWORD 8 are reserved and set to 0. | TD 5.18 | |
| **Subsection reference 6.3.25 Update Device Response (UpdateDevResp)** | | | |
| 6.3.25#1 | The Update Device Response (UpdateDevResp) packet is transmitted by the target MA USB device to the MA USB host in response to an UpdateDevReq packet. | TD 6.16 | |
| 6.3.25#2 | The Type and Subtype fields are set to 0 (Management) and 23 (UpdateDevResp), respectively. | TD 6.16 | |
| 6.3.25#3 | The Device Handle field carries the handle of the USB device for which the response is being returned. | N/A | Not testable |
| 6.3.25#4 | The Status Code field indicates whether the request was successfully completed. | TD 6.16 | |
| **Subsection reference 6.3.26 USB Device Disconnect Request (USBDevDisconnectReq)** | | | |
| 6.3.26#1 | The Type and Subtype fields are set to 0 (Management) and 24 (USBDevDisconnectReq), respectively. | TD 5.22 | |
| 6.3.26#2 | Status Code field is set to 0 (NO_ERROR). | TD 5.22 | |
| 6.3.26#3 | The Device Handle field carries the handle of the USB device the request is targeting. | N/A | Not testable |
| **Subsection reference 6.3.27 USB Device Disconnect Response (USBDevDisconnectResp)** | | | |
| 6.3.27#1 | The USB Device Disconnect Response (USBDevDisconnectResp) packet is transmitted by the target MA USB device to the MA USB host in response to a DisconnectDevReq packet. | TD 6.20 | |
| 6.3.27#2 | The Type and Subtype fields are set to 0 (Management) and 25 (USBDevDisconnectResp), respectively. | TD 6.20 | |
| 6.3.27#3 | The Device Handle field carries the handle of the USB device for which the response is being returned. | N/A | Not testable |
| 6.3.27#4 | The Status Code field indicates whether the request was successfully completed. | TD 6.20 | |
| **Subsection reference 6.3.28 USB Suspend Request (USBSuspendReq)** | | | |
| 6.3.28#1 | Type and Subtype fields are set to 0 (Management) and 26 (USBSuspendReq), respectively. | TD 5.19 | |
| 6.3.28#2 | The Status Code value is set to 0 (NO_ERROR). | TD 5.19 | |
| 6.3.28#3 | The Device Handle field identifies the integrated USB device the request is targeting | N/A | Not testable |
| **Subsection reference 6.3.29 USB Suspend Response (USBSuspendResp)** | | | |
| 6.3.29#1 | The USB Suspend Response (USBSuspendResp) packet is transmitted by the target MA USB device to the MA USB host in response to a USBSuspendReq packet. | TD 6.17 | |

| | | | |
|---|---|---|---|
| 6.3.29#2 | The Type and Subtype fields are set to 0 (Management) and 27 (USBSuspendResp), respectively. | TD 6.17 | |
| 6.3.29#3 | The Device Handle field carries the same value as the Device Handle field in the corresponding USBSuspendReq packet. | TD 6.17 | |
| 6.3.29#4 | The Status Code value indicates whether the request was successfully completed. | TD 6.17 | |
| **Subsection reference 6.3.30 USB Resume Request (USBResumeReq)** | | | |
| 6.3.30#1 | The Type and Subtype fields are set to 0 (Management) and 28 (USBResumeReq), respectively. | TD 5.19 | |
| 6.3.30#2 | The Status Code field is set to 0 (NO_ERROR). | TD 5.19 | |
| 6.3.30#3 | The Device Handle field identifies the integrated USB device behind the MA USB device PAL the request is targeting. | N/A | Not testable |
| **Subsection reference 6.3.31 USB Resume Response (USBResumeResp)** | | | |
| 6.3.31#1 | The USB Resume Response (USBResumeResp) packet is transmitted by the target MA USB device to the MA USB host in response to a USBResumeReq packet. | TD 6.17 | |
| 6.3.31#2 | The Type and Subtype fields are set to 0 (Management) and 29 (USBResumeResp), respectively. | TD 6.17 | |
| 6.3.31#3 | The Device Handle field carries the same value as the Device Handle field in the corresponding USBResumeReq packet. | TD 6.17 | |
| 6.3.31#4 | The Status Code field indicates whether the request was successfully completed. | TD 6.17 | |
| **Subsection reference 6.3.32 Remote Wake Request (RemoteWakeReq)** | | | |
| 6.3.32#1 | The Type and Subtype fields are set to 0 (Management) and 30 (RemoteWakeReq), respectively. | TD 6.21 | |
| 6.3.32#2 | The Status Code field is set to 0 (NO_ERROR). | TD 6.21 | |
| 6.3.32#3 | The Device Handle field carries the handle of the integrated USB device the request is targeting. | N/A | Not testable |
| 6.3.32#4 | The USB Device Resumed field indicates whether the USB device initiating the remote wake request is already resumed and ready to receive USB control packets or the MA USB device PAL expects a USB Device Resume Request packet to resume the USB device. The field is set to 1 if the USB device is not in the suspend state, or 0 otherwise. | TD 6.21 | |
| 6.3.32#5 | Bits 1 through 31 of DWORD 3 in a Remote Wake Request packet are reserved and set to 0. | TD 6.21 | |
| **Subsection reference 6.3.33 Remote Wake Response (RemoteWakeResp)** | | | |

| | | | |
|---|---|---|---|
| 6.3.33#1 | The Remote Wake Response (RemoteWakeResp) packet is transmitted by the MA USB host to the target MA USB device in response to a RemoteWakeReq packet. | TD 5.19 | |
| 6.3.33#2 | The Type and Subtype fields are set to 0 (Management) and 31 (RemoteWakeResp), respectively. | TD 5.19 | |
| 6.3.33#3 | The Device Handle field carries the handle of the USB device for which the response is being returned. | N/A | Not testable |
| 6.3.33#4 | The Status Code field indicates whether the request was successfully completed. | TD 5.19 | |
| **Subsection reference 6.3.34 Ping Request (PingReq)** | | | |
| 6.3.34#1 | The Type and Subtype fields are set to 0 (Management) and 32 (PingReq), respectively. | TD 5.3<br>TD 6.6 | |
| 6.3.34#2 | The Status Code field is set to 0 (NO_ERROR). | TD 5.3<br>TD 6.6 | |
| 6.3.34#3 | The MA USB Device Address field is set to the address of the target MA USB device, or to 0xFF (any device). | N/A | |
| 6.3.34#4 | The Device Handle field is reserved and set to 0. | TD 5.3<br>TD 6.6 | |
| **Subsection reference 6.3.35 Ping Response (PingResp)** | | | |
| 6.3.35#1 | The Ping Response (PingResp) packet is transmitted by the MA USB device or the MA USB host in response to a PingReq packet. | TD 5.4<br>TD 6.6 | |
| 6.3.35#2 | The Type and Subtype fields are set to 0 (Management) and 33 (PingResp), respectively. | TD 5.4<br>TD 6.6 | |
| 6.3.35#3 | The Device Handle field is reserved and set to 0. | TD 5.4<br>TD 6.6 | |
| 6.3.35#4 | The Status Code field is set to 0 (NO_ERROR). | TD 5.4<br>TD 6.6 | |
| **Subsection reference 6.3.36 MA USB Device Disconnect Request (DevDisconnectReq)** | | | |
| 6.3.36#1 | The Type and Subtype fields are set to 0 (Management) and 34 (DevDisconnectReq), respectively. | TD 5.22 | |
| 6.3.36#2 | The Status Code field is set to 0 (NO_ERROR). | TD 5.22 | |
| 6.3.36#3 | The Device Handle field is reserved and set to 0. | TD 5.22 | |
| **Subsection reference 6.3.37 MA USB Device Disconnect Response (DevDisconnectResp)** | | | |
| 6.3.37#1 | The MA USB device Disconnect Response (DevDisconnectResp) packet is transmitted by the | TD 6.21 | |

| | | | |
|---|---|---|---|
| | target MA USB device to the MA USB host in response to a DevDisconnectReq packet. | | |
| 6.3.37#2 | The Type and Subtype fields are set to 0 (Management) and 35 (DevDisconnectResp), respectively | TD 6.21 | |
| 6.3.37#3 | The Device Handle field is reserved and set to 0. | TD 6.21 | |
| 6.3.37#4 | The Status Code field indicates whether the request was successfully completed. | TD 6.21 | |
| **Subsection reference 6.3.38 MA USB Device Initiated Disconnect Request (DevInitDisconnectReq)** | | | |
| 6.3.38#1 | The Type and Subtype fields are set to 0 (Management) and 36 (DevInitDisconnectReq), respectively. | TD 6.18 | |
| 6.3.38#2 | The Status Code field is set to 0 (NO_ERROR). | TD 6.18 | |
| 6.3.38#3 | The Device Handle field is reserved and set to 0. | TD 6.18 | |
| **Subsection reference 6.3.39 MA USB Device Initiated Disconnect Response (DevInitDisconnectResp)** | | | |
| 6.3.39#1 | The MA USB device Initiated Disconnect Response (DevInitDisconnectResp) packet is transmitted by the MA USB host to the target MA USB device in response to an DevInitDisconnectReq packet | TD 5.22 | |
| 6.3.39#2 | The Type and Subtype fields are set to 0 (Management) and 37 (DevInitDisconnectResp), respectively. | TD 5.22 | |
| 6.3.39#3 | The Device Handle field is reserved and set to 0. | TD 5.22 | |
| 6.3.39#4 | The Status Code field indicates whether the request was successfully completed. | TD 5.22 | |
| **Subsection reference 6.3.40 Synchronization Request (SynchReq)** | | | |
| 6.3.40#1 | The Type and Subtype fields are set to 0 (Management) and 38 (SynchReq), respectively. | TBD | |
| 6.3.40#2 | The Status Code field is set to 0 (NO_ERROR). | TBD | |
| 6.3.40#3 | The Device Handle and Dialog Token fields are reserved and set to 0. | TBD | |
| 6.3.40#4 | The MTD Valid field is set to 1 if the Media Time/Transmission Delay field in the packet carries a valid value and set to 0 otherwise | N/A | Not testable |
| 6.3.40#5 | The Response Required field is set to 1 if a Synchronization Response packet is required. | TBD | |
| 6.3.40#6 | Bits 2 through 31 of DWORD 3 in a Synchronization Request packet are reserved and set to 0. | TBD | |
| 6.3.40#7 | The MA USB timestamp field carries a sample of the MA USB Global Time. | Interop | |
| 6.3.40#8 | A Synchronization Request packet carries in the Media Time/Transmission Delay field either the synchronized Media Time of the MA link at the time Media Global | N/A | Not testable |

| | | | |
|---|---|---|---|
| | Time is captured into the MA USB Timestamp field, or the time elapsed in nanoseconds from the moment Media Global Time is captured into the MA USB Timestamp field to the moment the first bit of the Media Time/Transmission Delay field is released to the physical medium. | | *Original Text:* *Depending on the latency compensation method applied to an MA link, the 32-bit Media Time/Transmission Delay field carries the synchronized Media Time of the MA link at the time MGT is captured into the MA USB Timestamp field, or the time elapsed (in nanoseconds) from the moment MGT is captured into the MA USB Timestamp field to the moment the first bit of the Media Time/Transmission Delay field is released to the physical medium.* |
| **Subsection reference 6.3.41 Synchronization Response (SynchResp)** | | | |
| 6.3.41#1 | The Synchronization Response (SynchResp) packet is transmitted by the target MA USB device to the MA USB host in response to a SynchReq packet with unicast MA USB Device Address. An MA USB device may also transmit a SynchResp packet to the MA USB host in response to a SynchReq packet with broadcast MA USB Device Address. | TBD | |
| 6.3.41#2 | The Type and Subtype fields are set to 0 (Management) and 39 (SynchResp), respectively | TBD | |
| 6.3.41#3 | The Device Handle and Dialog Token fields are reserved and set to 0. | TBD | |
| 6.3.41#4 | The Status Code field is set to SUCCESS. | TBD | |
| 6.3.41#5 | The MTD Valid field is set to 1 if the Media Time/Transmission Delay field in the packet carries a valid value and set to 0 otherwise. | N/A | Not testable |
| 6.3.41#7 | Bits 2 through 31 of DWORD 3 in a Synchronization Response packet are reserved and set to 0. | TBD | |
| 6.3.41#8 | The Response Required field is reserved and set to 0. | TBD | |

| | | | |
|---|---|---|---|
| 6.3.41#9 | The MA USB Timestamp field carries the MA USB Global Time at the target MA USB device at the moment the field is initialized. | TBD | |
| 6.3.41#10 | A Synchronization Response packet carries in the Media Time/Transmission Delay field either the synchronized Media Time of the MA link at the time Media Global Time is captured into the MA USB Timestamp field, or the time elapsed in nanoseconds from the moment Media Global Time is captured into the MA USB Timestamp field to the moment the first bit of the Media Time/Transmission Delay field is released to the physical medium. | N/A | Not testable<br>*Original Text:*<br>*Depending on the latency compensation method applied to an MA link, the 32-bit Media Time/Transmission Delay field carries the synchronized Media Time of the MA link at the time MGT is captured into the MA USB Timestamp field, or the time elapsed (in nanoseconds) from the moment MGT is captured into the MA USB Timestamp field to the moment the first bit of the Media Time/Transmission Delay field is released to the physical medium.* |
| **Subsection reference 6.3.42 Cancel Transfer Request (CancelTransferReq)** | | | |
| 6.3.42#1 | The Type and Subtype fields are set to 0 (Management) and 40 (CancelTransferReq), respectively. | TD 5.22 | |
| 6.3.42#2 | The Status Code field is set to 0 (NO_ERROR). | TD 5.22 | |
| 6.3.42#3 | The Device Handle field carries the handle of the USB device the request is targeting. | N/A | Not testable |
| 6.3.42#4 | The Dialog Token field is reserved and set to 0. | TD 5.22 | |
| 6.3.42#5 | The EP Handle field indicates the endpoint the request is targeting. | N/A | Not testable |
| 6.3.42#6 | The Stream ID field indicates the target stream when the target endpoint is an Enhanced SuperSpeed bulk endpoint that supports the Enhanced SuperSpeed Stream Protocol; reserved otherwise. | TD 5.22 | |
| 6.3.42#7 | The Request ID field indicates the target request. | TD 5.22 | |
| 6.3.42#8 | Bits 8 through 31 of DWORD 4 are reserved and set to 0. | TD 5.22 | |
| **Subsection reference 6.3.43 Cancel Transfer Response (CancelTransferResp)** | | | |

| | | | |
|---|---|---|---|
| 6.3.43#1 | The Cancel Transfer Response (CancelTransferResp) packet is transmitted by the MA USB device in response to a CancelTransferReq packet | TD 6.20 | |
| 6.3.43#2 | The Type and Subtype fields are set to 0 (Management) and 41 (CancelTransferResp), respectively. | TD 6.20 | |
| 6.3.43#3 | The Device Handle field carries the handle of the USB device for which the response is being returned. | N/A | Not testable |
| 6.3.43#4 | The Status Code field indicates whether the request was successfully completed | TD 6.20 | |
| 6.3.43#5 | The Dialog Token field is reserved and set to 0. | TD 6.20 | |
| 6.3.43#6 | The EP Handle field indicates the endpoint for which the response is being returned. | N/A | Not testable |
| 6.3.43#7 | The Stream ID field indicates the stream to which the response is related when the endpoint identified in the EP Handle field is an Enhanced SuperSpeed bulk endpoint that supports the Enhanced SuperSpeed Stream Protocol; reserved and set to 0 otherwise. | TD 6.20 | |
| 6.3.43#8 | The Request ID field indicates the request for which the response is being returned. | TD 6.20 | |
| 6.3.43#9 | The Cancellation Status is set to 0 if the Status Code field is not set to SUCCESS. Set to 1 if the transfer was cancelled before any data was moved to/from the USB device. Set to 2 if the transfer was cancelled after some data was moved to/from the USB device. Set to 3 if the transfer was completed. Set to 4 if the transfer was not yet received. Set to 5 if the transfer was cleared without any data moved during ClearTransfersReq processing. | N/A | Not testable |
| 6.3.43#10 | Bits 10 through 31 of DWORD 4 are reserved and set to 0. | TD 6.20 | |
| 6.3.43#11 | For an OUT transfer, the Delivered Sequence Number field indicates the last sequence number delivered to the device. It is valid if Cancellation Status field is set to 2 and reserved otherwise. | N/A | |
| 6.3.43#12 | The Delivered Sequence Number field is reserved and set to 0 for IN transfers. | TD 6.20 | |
| 6.3.43#13 | Bits 24 through 31 of DWORD 5 of a Cancel Transfer Response packet are reserved and set to 0. | TD 6.20 | |
| 6.3.43#14 | The Delivered Byte Offset field is reserved and set to 0 if the Cancellation Status field is not set to 2. | TD 6.20 | |
| 6.3.43#15 | The Delivered Byte Offset field is reserved and set to 0 for IN transfers. | TD 6.20 | |
| **Subsection reference 6.3.44 Endpoint Open Stream Request (EPOpenStreamReq)** | | | |

| | | | |
|---|---|---|---|
| 6.3.44#1 | The Type and Subtype fields are set to 0 (Management) and 42 (EPOpenStreamReq), respectively. | TD 5.23 | |
| 6.3.44#2 | The Status Code field is set to 0 (NO_ERROR). | TD 5.23 | |
| 6.3.44#3 | The Device Handle field carries the handle of the USB device the request is targeting | N/A | Not testable |
| 6.3.44#4 | The EP Handle field indicates the Enhanced SuperSpeed bulk endpoint the request is targeting | N/A | Not testable |
| 6.3.44#5 | The Number of Streams field indicates the number of streams to be opened for the target endpoint. | N/A | Not tested |
| 6.3.44#6 | The Stream ID Index field indicates the lower bound for the stream IDs the MA USB host is retrieving, if the value of Open Stream field is set to 0. Reserved and set to 0 otherwise. | TD 5.23 | |
| 6.3.44#7 | The Allocation Mode field is set to 1 if the allocation of streams shall be sequential and start at 1. Otherwise the allocation of stream IDs is free of rules and the Allocation Mode field is set to 0. | N/A | Not testable |
| 6.3.44#8 | The Allocation Mode field may be set to 1 only if the number of streams is less than 256. | N/A | |
| 6.3.44#9 | The Open Stream field is set to 1 if the request is to open new streams or set to 0 if the request is to retrieve stream IDs of previously opened streams. | TBD | |
| 6.3.44#10 | Bits 18 through 31 of DWORD 4 in an Endpoint Open Stream Request field are reserved and set to 0. | TD 5.23 | |
| **Subsection reference 6.3.45 Endpoint Open Stream Response (EPOpenStreamResp)** | | | |
| 6.3.45#1 | The Endpoint Open Stream Response (EPOpenStreamResp) packet is transmitted by the target MA USB device to the MA USB host in response to an EPOpenStreamReq packet. | TD 6.24 | |
| 6.3.45#2 | The Type and Subtype fields are set to 0 (Management) 43 (EPOpenStreamResp), respectively. | TD 6.24 | |
| 6.3.45#3 | The Device Handle field carries the handle of the USB device for which the response is being returned. | N/A | Not testable |
| 6.3.45#4 | The Status Code field indicates whether the request was successfully completed. | TD 6.24 | |
| 6.3.45#5 | The Number of Streams field indicates the number of streams that are included in the packet. | TD 6.24 | |
| 6.3.45#6 | If the Status Code field in the packet carries a value indicating SUCCESS (NO_ERROR) the Number of Streams field shall carry a value less than or equal to the value of the Number of Streams field in the corresponding EPOpenStreamReq packet. | TD 6.24 | |

| | | | |
|---|---|---|---|
| 6.3.45#7 | The Number of Stream ID Blocks field indicates the number of stream ID blocks included in this packet. | TD 6.24 | |
| 6.3.45#8 | The stream ID interval blocks in an Endpoint Open Stream Response packet shall be included in the increasing order of the stream ID values. | TD 6.24 | |
| 6.3.45#9 | The First Stream ID field in a stream ID interval block indicates the value of the first stream ID in the interval block. | TD 6.24 | |
| 6.3.45#10 | The Last Stream ID field in a stream ID interval block indicates the value of the last stream ID in the interval block. | TD 6.24 | |
| **Subsection reference 6.3.46 Endpoint Close Stream Request (EPCloseStreamReq)** | | | |
| 6.3.46#1 | The Type and Subtype fields are set to 0 (Management) and 44 (EPCloseStreamReq), respectively. | TD 5.23 | |
| 6.3.46#2 | The Status Code field is set to 0 (NO_ERROR). | TD 5.23 | |
| 6.3.46#3 | The Device Handle field carries the handle of the USB device the request is targeting | N/A | Not testable |
| 6.3.46#4 | The EP Handle field indicates the EP handle of the endpoint the request is targeting. | N/A | Not testable |
| 6.3.46#5 | The Close All field indicates whether the request targets all the open streams and is set to 1 if the request is for all of the open streams, or 0 otherwise. | TBD | |
| 6.3.46#6 | Bits 17 through 31 of DWORD 3 are reserved and set to 0. | TD 5.23 | |
| 6.3.46#7 | The Number of Stream ID Blocks field indicates the number of stream ID blocks included in the packet. | TD 5.23 | |
| 6.3.46#8 | The Number of Stream ID Blocks field is set to 0 if the Close All field is set to 1. | TD 5.23 | |
| 6.3.46#9 | Bits 16 through 31 of DWORD 4 are reserved and set to 0. | TD 5.23 | |
| **Subsection reference 6.3.47 Endpoint Close Stream Response (EPCloseStreamResp)** | | | |
| 6.3.47#1 | The Endpoint Close Stream Response (EPCloseStreamResp) packet is transmitted by the target MA USB device to the MA USB host in response to an EPCloseStreamReq packet | TD 6.24 | |
| 6.3.47#2 | The Type and Subtype fields are set to 0 (Management) and 45 (EPCloseStreamResp), respectively. | TD 6.24 | |
| 6.3.47#3 | The Device Handle field carries the handle of the USB device for which the response is being returned. | N/A | Not testable |
| 6.3.47#4 | The Status Code field indicates whether the request was successfully completed. | TD 6.24 | |
| **Subsection reference 6.3.48 USB Device Reset Request (USBDevResetReq)** | | | |

| | | | |
|---|---|---|---|
| 6.3.48#1 | The Type and Subtype fields are set to 0 (Management) and 46 (USBDevResetReq), respectively. | Interop | |
| 6.3.48#2 | The Status Code field is set to 0 (NO_ERROR). | Interop | |
| 6.3.48#3 | The Device Handle field in a USB Device Reset Request packet shall carry the handle of the USB device that the request is targeting. | N/A | Not testable |
| **Subsection reference 6.3.49 USB Device Reset Response (USBDevResetResp)** | | | |
| 6.3.49#1 | The USB Device Reset Response (USBDevResetResp) packet is transmitted by the target MA USB device to the MA USB host in response to a USBDevResetReq packet. | TD 6.25 | |
| 6.3.49#2 | The Type and Subtype fields are set to 0 (Management) 47 (USBDevResetResp), respectively | TD 6.25 | |
| 6.3.49#3 | The Device Handle field carries the handle of the USB device for which the response is being returned. | N/A | Not testable |
| 6.3.49#4 | The Status Code field indicates whether the request was successfully completed. | TD 6.25 | |
| **Subsection reference 6.3.50 Device Notification Request (DevNotificationReq)** | | | |
| 6.3.50#1 | The Type and Subtype fields are set to 0 (Management) and 48 (DevNotificationReq), respectively. | TD 6.21 | |
| 6.3.50#2 | The Status Code field is set to 0 (NO_ERROR). | TD 6.21 | |
| 6.3.50#3 | The Device Handle field carries the handle of the USB device that is issuing the notification. | N/A | Not testable |
| 6.3.50#4 | Bits 0 through 3 of DWORD3 are reserved and set to 0. | TD 6.21 | |
| 6.3.50#5 | The Notification Type field identifies the type of the device notification as defined in Section 8.5.6 of USB 3.1. | TD 6.21 | |
| 6.3.50#6 | The Notification Type Specific Data field carries payload dependent on Notification Type. | N/A | Not testable |
| **Subsection reference 6.3.51 Device Notification Response (DevNotificationResp)** | | | |
| 6.3.51#1 | The Device Notification Response (DevNotificationResp) packet is transmitted by the MA USB host to the target MA USB device in response to a DevNotificationReq packet. | N/A | Not tested |
| 6.3.51#2 | The Type and Subtype fields are set to 0 (Management) and 49 (DevNotificationResp), respectively. | N/A | Not tested |
| 6.3.51#3 | The Device Handle field carries the handle of the USB device for which the response is being returned. | N/A | Not testable |

| | | | |
|---|---|---|---|
| 6.3.51#4 | The Status Code field indicates whether the request was successfully completed. | N/A | Not tested |
| **Subsection reference 6.3.52 Endpoint Set Keep-Alive Request (EPSetKeepAliveReq)** | | | |
| 6.3.52#1 | The Type and Subtype fields are set to 0 (Management) and 50 (EPSetKeepAliveReq), respectively. | TD 6.9 | |
| 6.3.52#2 | The Status Code field is set to 0 (NO_ERROR). | TD 6.9 | |
| 6.3.52#3 | The Device Handle field carries the handle of the USB device behind the MA USB device the request is targeting. | N/A | Not testable |
| 6.3.52#4 | The Keep Alive Duration field carries the duration of the keep-alive measured in units of aTransferKeepAlive. | N/A | Not testable |
| 6.3.52#5 | If the Keep-Alive Duration field is 0, the keep-alive is reset to the default value aDefaultKeepAliveDuration. | TD 5.12 | |
| 6.3.52#6 | The EP Handle indicates the endpoint the request is targeting. | N/A | Not Testable |
| **Subsection reference 6.3.53 Endpoint Set Keep-Alive Response (EPSetKeepAliveResp)** | | | |
| 6.3.53#1 | The Endpoint Set Keep-Alive Response (EPSetKeepAliveResp) packet is transmitted by the MA USB host to an MA USB device in response to an EPSetKeepAliveReq packet. | TD 5.12 | |
| 6.3.53#2 | The Type and Subtype fields are set to 0 (Management) and 51 (EPSetKeepAliveResp), respectively. | TD 5.12 | |
| 6.3.53#3 | The Device Handle field carries the handle of the USB device behind the MA USB device for which the response is being returned. | N/A | Not testable |
| 6.3.53#4 | The Status Code field indicates whether the request was successfully completed. | TD 5.12 | |
| 6.3.53#5 | The Old Keep-Alive Duration field indicates the previous value of the keep-alive measured in units of aTransferKeepAlive. | TD 5.12 | |
| 6.3.53#6 | The Start Request ID field indicates the request ID at which the new keep-alive duration will take effect. | N/A | |
| 6.3.53#7 | Bits 24 through 31 of an Endpoint Set Keep-Alive Response packet are reserved and set to 0. | TD 5.12 | |
| 6.3.53#8 | The Stream ID field indicates the stream to which the response is related when the endpoint identified in the corresponding EPSetKeepAliveReq packet is an Enhanced SuperSpeed bulk endpoint the supports SuperSpeed Stream Protocol. Otherwise, the Stream ID field is reserved and set to 0. | TD 5.12 | |
| 6.3.53#9 | The EPHandle field indicates the endpoint to which the response is related. | TD 5.12 | |

| Subsection reference 6.3.54 Get Port Bandwidth Request (GetPortBWReq) | | | |
|---|---|---|---|
| 6.3.54#1 | The Type and Subtype fields are set to 0 (Management) and 52 (GetPortBWReq), respectively. | TD 5.24 | |
| 6.3.54#2 | The Status Code field is set to 0 (NO_ERROR). | TD 5.24 | |
| 6.3.54#3 | The Device Handle field carries the handle of the USB hub the request is targeting. | N/A | Not testable |
| 6.3.54#4 | An MA USB host shall not transmit a GetPortBWReq packet to MA USB devices. | GTP | |
| 6.3.54#5 | An MA USB host shall not transmit another GetPortBWReq packet to an MA USB hub after it has received a GetPortBWResp packet with Status Code field set to NOT_SUPPORTED. | TD 5.24 | |
| 6.3.54#6 | The Speed field indicates the speed of the port(s) for which the available bandwidth shall be returned. | Interop | |
| 6.3.54#7 | Values 5 - 15 in the Speed field are reserved. | TD 5.24 | |
| 6.3.54#8 | Bits 4 through 31 of DWORD 3 and bits 0 through 3 of DWORD 4 in a Get Port Bandwidth Request packet are reserved and set to 0. | TD 5.24 | |
| 6.3.54#9 | If the Speed field of a Get Port Bandwidth Request packet is set to 3 or 4, the Lane Speed Exponent field carries the lane speed exponent of the USB device as defined in USB 3.1. Otherwise, the Lane Speed Exponent field is reserved and set to 0. | TD 5.24 | *Original Text: The Lane Speed Exponent field Indicates the LSE of the USB device as defined in [USB 3.1] if the Speed field is set to SuperSpeed or SuperSpeedPlus. Reserved otherwise.* |
| 6.3.54#10 | If the Speed field of a Get Port Bandwidth Request packet is set to 3 or 4, the Lane Count field carries the lane count of the USB device as defined in USB 3.1. Otherwise, the Lane Count field is reserved and set to 0. | TD 5.24 | *Original Text: The Lane Count field indicates the lane count of the USB device as defined in [USB 3.1] if the Speed field is set to SuperSpeed or SuperSpeedPlus. Reserved otherwise.* |
| 6.3.54#11 | If the Speed field of a Get Port Bandwidth Request packet is set to 3 or 4, the Link Protocol field carries the link protocol of the USB device as defined in USB 3.1. Otherwise, the Link Protocol field is reserved and set to 0. | TD 5.24 | *Original Text: the Link Protocol field indicates the link protocol of the USB device as defined* |

| | | | |
|---|---|---|---|
| | | | *in [USB 3.1] if the Speed field is set to SuperSpeed or SuperSpeedPlus. Reserved otherwise.* |
| 6.3.54#12 | Bits 12 through 16 of DWORD 4 are reserved and set to 0. | TD 5.24 | |
| 6.3.54#13 | If the Speed field of a Get Port Bandwidth Request packet is set to 3 or 4, the Lane Speed Mantissa field carries the lane speed mantissa of the USB device as defined in USB 3.1. Otherwise, the Lane Speed Mantissa field is reserved and set to 0. | TD 5.24 | *Original Text: the Lane Speed Mantissa field indicates the LSM of the USB device as defined in [USB 3.1] if the Speed field is set to SuperSpeed or SuperSpeedPlus. Reserved otherwise.* |
| **Subsection reference 6.3.55 Get Port Bandwidth Response (GetPortBWResp)** | | | |
| 6.3.55#1 | The Get Port Bandwidth Response (GetPortBWResp) packet is transmitted by the target MA USB hub to the MA USB host in response to a GetPortBWReq packet to report the percentage of the periodic bandwidth available on each port of the USB hub the request is targeting. | TD 6.27 | |
| 6.3.55#2 | The Type and Subtype fields are set to 0 (Management) and 53 (GetPortBWResp), respectively. | TD 6.27 | |
| 6.3.55#3 | The Status Code field indicates whether the MA USB hub supports the feature, and if it does whether the request was successfully completed. | TD 6.27 | |
| 6.3.55#4 | If the MA USB hub does not support the GetPortBWReq packet it shall respond with a GetPortBWResp packet with Status Code field set to NOT_SUPPORTED. | TD 6.27 | |
| 6.3.55#5 | If the MA USB hub does not support the speed indicated by the value of the Sublink Speed Attribute ID field it shall respond with a GetPortBWResp packet with Status Code field set to UNSUCCESSFUL. | TD 6.27 | |
| 6.3.55#6 | If the status code field is set to SUCCESS, the Number of Ports field indicates the number of ports on the USB hub for which bandwidth information is reported. | TD 6.27 | |
| 6.3.55#7 | Bits 8 through 31 of DWORD 3 in a Get Port Bandwidth Response packet are reserved and set to 0. | TD 6.27 | |
| 6.3.55#8 | If the status code field is set to SUCCESS, the Port Available Bandwidth List field carries a list of port available bandwidth reports, concatenated in 8-bit | TD 6.27 | |

| | | | |
|---|---|---|---|
| | increments, starting with the report of port number 1 in increasing order. | | |
| 6.3.55#9 | Each 8-bit entry in the Port Available Bandwidth List field has a range of 0 to 100 and indicates the percentage of the periodic bandwidth available to the corresponding port. | TD 6.27 | |
| **Subsection reference 6.3.56 Sleep Request (SleepReq)** | | | |
| 6.3.56#1 | The Type and Subtype fields are set to 0 (Management) and 54 (SleepReq), respectively. | TD 5.19<br>TD 6.28 | |
| 6.3.56#2 | The Status Code field is set to 0 (NO_ERROR). | TD 5.19<br>TD 6.28 | |
| 6.3.56#3 | The Device Handle field is reserved and set to 0. | TD 5.19<br>TD 6.28 | |
| 6.3.56#4 | The Management Request Timeout field indicates the maximum time (in milliseconds) between the moment the remote MA USB PAL releases a management packet requiring a response to the management channel, and the moment it can expect the corresponding response packet through the management channel, while its session is in Session Inactive state. | TD 5.1<br>TD 5.19<br>TD 6.28<br>TD 6.29 | |
| 6.3.56#5 | The Data Request Timeout field indicates the maximum time (in milliseconds) between the moment the remote MA USB PAL releases a control or data packet requiring a response to the assigned data channel, and the moment it can expect the corresponding response packet through the assigned data channel, while its session is in Session Inactive state. | TD 5.19<br>TD 5.25<br>TD 6.28<br>TD 6.29 | |
| **Subsection reference 6.3.57 Sleep Response (SleepResp)** | | | |
| 6.3.57#1 | The Sleep Response (SleepResp) packet is transmitted in response to a SleepReq packet to acknowledge the desire to move the session state to Session Inactive. | TD 5.19<br>TD 5.20<br>TD 6.17 | |
| 6.3.57#2 | The Type and Subtype fields are set to 0 (Management) and 55 (SleepResp), respectively. | TD 5.19<br>TD 5.20<br>TD 6.17 | |
| 6.3.57#3 | The Device Handle field is reserved and set to 0. | TD 5.19<br>TD 5.20<br>TD 6.17 | |
| 6.3.57#4 | The Status Code value indicates whether the request to move the session to Session Inactive is granted. | TD 5.19<br>TD 5.20 | |

| | | | |
|---|---|---|---|
| | | TD 6.17 | |
| 6.3.57#5 | When the Status Code field is set to 0 (NO_ERROR), the Management Request Timeout value is less than or equal to the corresponding value in the SleepReq packet and indicates the management exchange timeout value when the exchange is initiated by the recipient of the SleepResp packet. | TD 5.19<br>TD 5.20<br>TD 5.25<br>TD 6.17<br>TD 6.29 | |
| 6.3.57#6 | When the Status Code field is set to 0 (NO_ERROR) Data Request Timeout value is less than or equal to the corresponding value in the SleepReq packet and indicates the control or data packet exchange timeout value when the exchange is initiated by the recipient of the SleepResp packet. | TD 5.19<br>TD 5.20<br>TD 5.25<br>TD 6.17<br>TD 6.29 | |
| 6.3.57#7 | When the Status Code field is set to REQUEST_DENIED, the Management Request Timeout value is less than or equal to the corresponding value in the SleepReq packet and indicates the maximum timeout value acceptable to the recipient of the SleepReq packet. | N/A | Not testable |
| 6.3.57#8 | When the Status Code field is set to REQUEST_DENIED, the Data Request Timeout value is less than or equal to the corresponding value in the SleepReq packet and indicates the maximum timeout value acceptable to the recipient of the SleepReq packet. | N/A | Not testable |
| **Subsection reference 6.3.58 Wake Request (WakeReq)** | | | |
| 6.3.58#1 | The Type and Subtype fields are set to 0 (Management) and 56 (WakeReq), respectively. | TD 6.28 | |
| 6.3.58#2 | The Status Code field is set to 0 (NO_ERROR). | TD 6.28 | |
| 6.3.58#3 | The Device Handle field is reserved and set to 0. | TD 6.28 | |
| **Subsection 6.3.59 Wake Response (WakeResp)** | | | |
| 6.3.59#1 | The Wake Response (WakeResp) packet is transmitted in response to a WakeReq packet to acknowledge the desire to move the session state to Session Active. | TD 5.20<br>TD 6.17 | |
| 6.3.59#2 | The Type and Subtype fields are set to 0 (Management) and 57 (WakeResp), respectively. | TD 5.20<br>TD 6.17 | |
| 6.3.59#3 | The Device Handle field is reserved and set to 0. | TD 5.20<br>TD 6.17 | |
| 6.3.59#4 | The Status Code field is set to 0 (NO_ERROR). | TD 5.20<br>TD 6.17 | |
| **Subsection reference 6.3.60 Vendor Specific Request (VendorSpecificReq)** | | | |

| | | | |
|---|---|---|---|
| 6.3.60#1 | The Type and Subtype fields are set to 0 (Management) and 62 (VendorSpecificReq), respectively. | TBD | |
| 6.3.60#2 | The Status Code field is set to 0 (NO_ERROR). | TBD | |
| 6.3.60#3 | The Device Handle field is reserved and set to 0. | TBD | |
| 6.3.60#4 | The Vendor Identifier field carries the value assigned by the USB-IF to the organization that has defined the content of the particular vendor-specific request. | TBD | |
| 6.3.60#5 | Bits 16 through 31 of DWORD 3 in a Vendor Specific Request are reserved and set to 0. | TBD | |
| **Subsection reference 6.3.61 Vendor Specific Response (VendorSpecificResp)** | | | |
| 6.3.61#1 | The Vendor Specific Response (VendorSpecificResp) packet is transmitted by the MA USB host or the MA USB device in response to a VendorSpecificReq packet. | TBD | |
| 6.3.61#2 | The Type and Subtype fields are set to 0 (Management) and 63 (VendorSpecificResp), respectively. | TBD | |
| 6.3.61#3 | The Device Handle field is reserved and set to 0. | TBD | |
| 6.3.61#4 | The Status Code field is set to SUCCESS. | TBD | |
| 6.3.61#5 | The Vendor Identifier field carries the value assigned by the USB-IF to the organization that has defined the content of the particular vendor-specific request that triggered the response. | TBD | |
| 6.3.61#6 | Bits 16 through 31 of DWORD 3 in a Vendor Specific Request are reserved and set to 0. | TBD | |
| **Subsection reference 6.4 Control Packets** | | | |
| **Subsection reference 6.4.1 Transfer Setup Request (TransferSetupReq)** | | | |
| 6.4.1#1 | The Type and Subtype fields are set to 1 (Control) and 0 (TransferSetupReq), respectively. | TBD | |
| 6.4.1#2 | The Status Code field is set to 0 (NO_ERROR). | TBD | |
| 6.4.1#3 | The Link Type field in a Transfer Setup Request packet is set to 1 for an IEEE 802.11 link type. All other values for the Link Type field are reserved. | TBD | |
| 6.4.1#4 | For an IEEE 802.11 link, the Connection ID field is 8 bits wide, with the lower-order 4 bits set to the Traffic Identifier (TID) on which a dedicated traffic flow is established or will be established to serve the l-managed OUT transfer. | TBD | |
| 6.4.1#5 | For an IEEE 802.11 link, the Connection ID field is 8 bits wide, with the higher-order 4 bits of the field reserved. | TBD | |
| **Subsection reference 6.4.2 Transfer Setup Response (TransferSetupResp)** | | | |

| | | | |
|---|---|---|---|
| 6.4.2#1 | The Transfer Setup Response (TransferSetupResponse) packet indicates the success or failure of the set up phase for an I-managed OUT transfer. | TBD | |
| 6.4.2#2 | The Type and Subtype fields are set to 1 (Control) and 1 (TransferSetupResp), respectively. | TBD | |
| 6.4.2#3 | The Status Code field indicates the success or failure of the I-managed OUT transfer set up phase. | TBD | |
| 6.4.2#4 | The Link Type field in a Transfer Setup Response packet is set to 1 for an IEEE 802.11 link type. All other values for the Link Type field are reserved. | TBD | |
| 6.4.2#5 | For an IEEE 802.11 link, the Connection ID field is 8 bits wide, with the lower-order 4 bits set to the Traffic Identifier (TID) on which a dedicated traffic flow is established or will be established to serve the I-managed OUT transfer. | TBD | |
| 6.4.2#6 | For an IEEE 802.11 link, the Connection ID field is 8 bits wide, with the higher-order 4 bits of the field reserved. | TBD | |
| **Subsection reference 6.4.3 Transfer Tear Down Confirmation (TransferTearDownConf)** | | | |
| 6.4.3#1 | The Type and Subtype fields are set to 1 (Control) and 2 (TransferTearDownConf), respectively. | TBD | |
| 6.4.3#2 | The Status Code field is set to 0 (NO_ERROR). | TBD | |
| **Subsection reference 6.5 Data Packets** | | | |
| **Subsection reference 6.5.1 Common Data Header Fields** | | | |
| **Subsection 6.5.1.1 EPS** | | | |
| 6.5.1.1#1 | The EPS field is reserved in data packets transmitted by the MA USB host. | GTP | |
| 6.5.1.1#2 | If a data packet is transmitted by an MA USB device, the EPS field carries a value of 0 if the EP handle contained in the packet is not assigned to any endpoint. | GTP | |
| 6.5.1.1#3 | If a data packet is transmitted by an MA USB device, the EPS field carries a value of 1 if the EP handle contained in the packet is assigned and is in an active state. | N/A | Not tested |
| 6.5.1.1#4 | If a data packet is transmitted by an MA USB device, the EPS field carries a value of 2 if the EP handle contained in the packet is assigned but is in an inactive state. | GTP | |
| 6.5.1.1#5 | If a data packet is transmitted by an MA USB device, the EPS field carries a value of 3 if the EP handle contained in the packet is in the Halted state. | GTP | |
| **Subsection reference 6.5.1.2 T-Flags** | | | |

| | | | |
|---|---|---|---|
| 6.5.1.2#1 | The ARQ subfield is set to 1 to indicate an acknowledgment request and 0 otherwise. | N/A | |
| 6.5.1.2#2 | The ARQ subfield is reserved and set to 0 for all data packet subtypes other than Transfer Request and Transfer Response. | GTP | |
| 6.5.1.2#3 | The NEG subfield in the T-Flags field may be set to 1 only if the device indicates support for elastic buffer capability in the CapResp packet it sends to the MA USB host. | GTP | |
| 6.5.1.2#4 | For an OUT transfer, the EoT subfield in the T-Flags field is set to 1 to indicate the completion of the transfer. | N/A | Not tested |
| 6.5.1.2#5 | For an IN transfer, the EoT subfield in the T-Flags field is set to 1 to indicate the last TransferResp packet related to the transfer. | N/A | Not tested |
| 6.5.1.2#6 | The Transfer Type subfield in the T-Flags field is set to 0 for a Control transfer, 1 for an Isochronous transfer, 2 for a Bulk transfer, or 3 for an Interrupt transfer. | Interop | |
| 6.5.1.2#7 | In a Control Transfer Request packet with the Sequence Number field set to 0, the first 2 DWORDs of the payload in the packet are control transfer setup data. | GTP | |
| 6.5.1.2#8 | Bit 15 in the T-Flags field in a common data header are reserved and set to 0. | GTP | |
| **Subsection reference 6.5.1.3 Stream ID** | | | |
| 6.5.1.3#1 | The Stream ID field identifies the target stream when the target endpoint is an Enhanced SuperSpeed bulk endpoint that supports the Enhanced SuperSpeed Stream Protocol. It is reserved and set to 0 in all other cases. | GTP | |
| **Subsection reference 6.5.1.4 Sequence Number** | | | |
| 6.5.1.4#1 | The Sequence Number field is set to 0 for the first data bearing packet, and is incremented by one for each subsequent data bearing packet, with rollover to 0 after aMaxSequenceNumber | Interop | |
| **Subsection reference 6.5.1.5 Request ID** | | | |
| 6.5.1.5#1 | All data packets (request or response) belonging to the same MA USB transfer carry the same Request ID. | Interop | |
| **Subsection reference 6.5.1.6 Remaining Size/Credit** | | | |
| 6.5.1.6#1 | The 32-bit Remaining Size/Credit field the number of bytes remaining to complete a transfer (assuming the current packet is successfully received), or the number of bytes that the receiver can accept. | Interop | |
| **Subsection reference 6.5.1.7 Number of Headers** | | | |
| 6.5.1.7#1 | When sent in an MA USB isochronous transfer request packet for an OUT transfer or an MA USB | GTP | |

| | | | |
|---|---|---|---|
| | isochronous transfer response for an IN transfer, the 12-bit Number of Headers field carries the number of isochronous headers that are present in the packet. | | |
| 6.5.1.7#2 | When sent in an MA USB isochronous transfer request packet for an IN transfer, the Number of Headers field carries the number of IRS headers that are present in the packet. | GTP | |
| 6.5.1.7#3 | For a data packet that is neither a Transfer Request associated with an isochronous IN transfer, a Transfer Request packet associated with an isochronous OUT transfer, nor a Transfer Response associated with an isochronous IN transfer, the Number of Headers field in the common data header is reserved and set to 0. | GTP | |
| **Subsection reference 6.5.1.8 I-Flags** | | | |
| 6.5.1.8#1 | The MTD Valid subfield in the I-Flags field is set to 1 if the Media Time/Transmission Delay field in the packet carries a valid value and set to 0 otherwise. | TBD | |
| 6.5.1.8#2 | When applied to isochronous headers, valid values for the Isochronous Header Format subfield are 0 (short format), 1 (standard format) and 2 (long format). | TBD | |
| 6.5.1.8#3 | When applied to isochronous read size, (IRS) blocks valid values for the Isochronous Header Format field are 0 (standard format) and 1 (long format). | TBD | |
| 6.5.1.8#4 | The ASAP subfield is set to 1 to request immediate (as soon as possible) delivery to or from the target isochronous endpoint, or to 0 to indicate delivery at the time specified by the Presentation Time field in an isochronous transfer request packet. | N/A | |
| 6.5.1.8#5 | The ASAP field is reserved and set to 0 for isochronous transfer response packets. | TBD | |
| **Subsection reference 6.5.1.9 Presentation Time** | | | |
| 6.5.1.9#1 | The Presentation Time field is reserved and set to 0 in isochronous transfer request packets that indicate as soon as possible (ASAP) delivery. | TBD | |
| 6.5.1.9#2 | The Microframe Number subfield indicates the modulo-8 number of the USB microframe (the 125 µs time base) at the time of interest and is set to 0 when not supported. | TBD | |
| 6.5.1.9#3 | The Frame Number subfield indicates the USB frame number, maintained modulo 2^17 or a smaller number, but not smaller than 2^11. | TBD | |
| **Subsection reference 6.5.1.10 Number of Segments** | | | |
| 6.5.1.10#1 | When sent in an MA USB isochronous transfer request packet, the 12-bit Number of Segments field carries the total number of isochronous segments that are | TBD | |

| | | | |
|---|---|---|---|
| | being requested or transmitted for the MA USB transfer. | | |
| 6.5.1.10#2 | When sent in an MA USB isochronous transfer response packet, the Number of Segments field carries the number of complete segments that have been read from or delivered to the target isochronous endpoint in the MA USB device at the time the response packet was generated. | TBD | |
| **Subsection reference 6.5.1.11 MA USB Timestamp** | | | |
| **Subsection reference 6.5.1.12 Media Time/Transmission Delay** | | | |
| 6.5.1.12#1 | If a data packet is an isochronous Transfer Request, the Media Time/Transmission Delay field in the common data header carries either the synchronized Media Time of the MA link at the time MGT is captured into the MA USB Timestamp field, or the time elapsed (in nanoseconds) from the moment MGT is captured into the MA USB Timestamp field to the moment the first bit of the Media Time/Transmission Delay field is released to the physical medium. | TBD | *Original Text: Depending on the latency compensation method applied to an MA link, the 32-bit Media Time/Transmission Delay field carries the synchronized Media Time of the MA link at the time MGT is captured into the MA USB Timestamp field, or the time elapsed (in nanoseconds) from the moment MGT is captured into the MA USB Timestamp field to the moment the first bit of the Media Time/Transmission Delay field is released to the physical medium.* |
| **Subsection reference 6.5.2 Transfer Request** | | | |
| 6.5.2#1 | A Transfer Request packet carries the transfer payload for OUT transfers. | Interop | |
| 6.5.2#2 | When initiating a control transfer a Transfer Request packet carries the control set up data. | Interop | |
| 6.5.2#3 | When initiating a bulk or interrupt IN transfer a Transfer Request packet carries no payload. | GTP | |
| 6.5.2#4 | The Type and Subtype fields are set to 2 (Data) and 0 (TransferReq), respectively. | TD 5.6 | |

| | | | |
|---|---|---|---|
| 6.5.2#5 | The Status Code field is set to 0 (NO_ERROR). | TD 5.6 | |
| 6.5.2#6 | The EPS field is reserved and set to 0. | TD 5.6 | |
| 6.5.2#7 | The Remaining Size/Credit field carries the remaining number of bytes to complete an OUT transfer. | TD 5.6 | |
| 6.5.2#8 | The Remaining Size/Credit field carries the remaining number of bytes that the device can transmit in the case of an IN transfer. | TD 5.6 | |
| **Subsection reference 6.5.3 Transfer Response** | | | |
| 6.5.3#1 | The Type and Subtype fields are set to 2 (Data) and 1 (TransferResp), respectively | Interop | |
| 6.5.3#2 | The Status Code field is set to one of the values listed in Table 6 to indicate successful or failed reception of the payload belonging to one or more Transfer Request packets. | Interop | |
| 6.5.3#3 | The EPS field is set to one of the values listed in Table 62 to indicate the status of the EP handle contained in the packet. | Interop | |
| 6.5.3#4 | For IN transfers, the Remaining Size/Credit field carries the remaining number of bytes to complete the transfer. | Interop | |
| 6.5.3#5 | For OUT transfers, the Remaining Size/Credit field carries the number of bytes the MA USB device can accept throughout the OUT transfer. | Interop | |
| **Subsection reference 6.5.4 Transfer Acknowledgement** | | | |
| 6.5.4#1 | The Type and Subtype fields are set to 2 (Data) and 2 (TransferAck), respectively. | TD 5.6 | |
| 6.5.4#2 | The Status Code field is set to one of the values listed in Table 6. | TD 5.6 | |
| 6.5.4#3 | The Status Code field shall be set to TRANSFER_PENDING if the TransferAck is sent in response to a TransferResp packet that had the ARQ bit set to 1 and a Status Code value of TRANSFER_PENDING. | TD 5.26 | |
| 6.5.4#4 | The EPS field is reserved and set to 0. | TD 5.6 | |
| 6.5.4#5 | The Remaining Size/Credit field is reserved and set to 0. | TD 5.6 | |
| **Subsection reference 6.5.5 Isochronous Transfer Request** | | | |
| 6.5.5#1 | When initiating an isochronous IN transfer, an Isochronous Transfer Request packet carries no payload. | TBD | |
| 6.5.5#2 | When initiating an isochronous OUT transfer, an Isochronous Transfer Request packet carries the first segment of the OUT transfer payload. | TBD | |

| | | | |
|---|---|---|---|
| 6.5.5#3 | The Type and Subtype fields are set to 2 (Data) and 3 (IsochTransferReq), respectively. | TBD | |
| 6.5.5#4 | The Status Code field is set to 0 (NO_ERROR). | TBD | |
| 6.5.5#5 | The EPS field is reserved and set to 0. | TBD | |
| 6.5.5#6 | The Sequence Number field is set to aInvalidSequenceNumber for IN transfers. | TBD | |
| **Subsection reference 6.5.6 Isochronous Transfer Response** | | | |
| 6.5.6#1 | An Isochronous Transfer Response carries isochronous payload for IN transfers. | TBD | |
| 6.5.6#2 | An Isochronous Transfer Response provides feedback for isochronous OUT transfers. | TBD | |
| 6.5.6#3 | The Type and Subtype fields are set to 2 (Data) and 4 (IsochTransferReq), respectively. | TBD | |
| **Subsection reference 6.6 Clock Synchronization** | | | |
| **Subsection reference 6.6.1 Clock Model** | | | |
| 6.6.1#1 | The MA USB Global Clock has the same frequency as the Enhanced SuperSpeed USB Bus Clock (8KHz). | TBD | |
| 6.6.1#2 | Each MA USB hub is expected to synchronize the Bus Clock for each of its wired USB segments downstream to the MA USB Global Clock. | TBD | |
| 6.6.1#3 | The frame counter in the PID must match bits 26:16 of the MGT. | TBD | |
| 6.6.1#4 | For SuperSpeed and SuperSpeed Plus segments, bits 13:0 of the ITS in the ITP, when issued from the root port of the segment, must match bits 26:13 of the MGT. | TBD | |
| 6.6.1#5 | Bits 26:14 in the ITS of the ITP must be set appropriately relative to the bus boundary established by the MGT when the MGT Delta field is zero. | TBD | |
| 6.6.1#6 | The Delta subfield in the MA USB Global time field indicates the value of Delta field in Isochronous Timestamp Packet as defined in [USB 3.1]. | TBD | |
| 6.6.1#7 | The Delta subfield in the MA USB Global time field is maintained in units of 8 nominal HS bit times (8/480 µs). | TBD | |
| 6.6.1#8 | The Nominal Bus Interval subfield in the MA USB Global time field indicates the nominal USB microframe number, i.e. the 125µs time base. | TBD | |
| 6.6.1#9 | The Nominal Bus Interval is maintained in modulo 2^19. | TBD | |
| **Subsection reference 6.6.2 Synchronization** | | | |

| | | | |
|---|---|---|---|
| 6.6.2#1 | The Media Clock of an MA link shall have an accuracy of +/-aMaxMediaTimeError, i.e., the Media Time at the two ends of an MA link shall not be different by more than 2 * aMaxMediaTimeError. | TBD | |
| 6.6.2#2 | Sampling of the Media Clock shall have an accuracy of +/-aMaxMediaTimeSamplingError, i.e., the value inserted into the Media Time field of an MA USB packet shall not be more than +/-aMaxMediaTimeSamplingError different from the value of the Media Time at the moment of insertion. | TBD | |
| 6.6.2#3 | A received MA USB timestamp carries an error no smaller than the difference between the local Media Time at the moment the MA USB timestamp is captured and the Media Time transmitted in the MA USB packet. | TBD | |
| 6.6.2#4 | The delay estimate, expressed in nanoseconds, is carried in the Transmission Delay field of the MA USB packet containing the timestamp, and shall have an accuracy of +/-aMaxTransmissionDelayError | TBD | |
| 6.6.2#5 | The compensation method for each MA link is decided by the MA USB host and does not change through the lifetime of an MA link. | TBD | |
| 6.6.2#6 | The latency compensation method applied to the MA link in both directions shall be based on Media Time if both MA USB host and device have indicated Media Time availability, and shall be based on Transmission Delay otherwise. | TBD | |
| 6.6.2#7 | The MA USB host shall not transmit unicast SynchReq packets to an MA USB device that has not requested to receive MA USB timestamps. | TBD | |
| 6.6.2#8 | An MA USB device that has requested to receive MA USB timestamps shall respond to a unicast or broadcast SynchReq packet that has a Response Required bit set to 1 with a SynchResp packet that includes a local reading of the MGT. | TBD | |
| 6.6.2#9 | The MA USB host shall distribute the MGT to each MA USB device in Session Active state that has at least one configured isochronous endpoint. | TBD | |
| 6.6.2#10 | MGT shall be delivered at least once every aMinSynchFrequencyActive if the target MA USB device has at least one active isochronous endpoint (i.e., a configured isochronous endpoint that has not been idle for more than aMinSynchFrequencyIdle), and at least every aMinSynchFrequencyIdle otherwise. | TBD | |

# *Chapter 7 Test Assertions*

| Assertion # | Assertion Description | Test # | Comments |
|---|---|---|---|
| **Subsection reference: 7.1 Device States** | | | |
| **Subsection reference: 7.2 EP handle states** | | | |
| 7.2#1 | An endpoint handle is either not assigned or assigned. | N/A | |
| 7.2#2 | An assigned endpoint handle can be in Active, Inactive or Halt State. | N/A | |
| **Subsection reference: 7.2.1 Active state** | | | |
| 7.2.1#1 | When an endpoint handle is first assigned by the MA USB device in response to an EPHandleReq packet it shall enter the Active state. | N/A | |
| 7.2.1#2 | When an MA USB device receives an EPActivateReq packet for an endpoint handle in Inactive state it shall enter the Active State. | N/A | |
| 7.2.1#3 | While an EP handle is in the Active state it shall be capable of USB transactions. | N/A | |
| 7.2.1#4 | While an EP handle is in the Active state the MA USB host may transmit TransferReq packets targeting the endpoint. | N/A | |
| **Subsection reference: 7.2.2 Halted state** | | | |
| 7.2.2#1 | The Halted state is entered from the Active State when an endpoint encounters a USB halt condition. | N/A | |
| 7.2.2#2 | Any TransferReq packet received targeting an EP in Halted State shall be buffered and acknowledged with a TransferResp packet with status code set to INVALID_EP_HANDLE_STATE. | N/A | |
| **Subsection reference: 7.2.3 Inactive state** | | | |
| 7.2.3#1 | The Inactive state is entered if the MA USB device receives an EPInactivateReq packet from the MA USB host. | N/A | |
| 7.2.3#2 | The Inactive state is entered from the Halted State if the MA USB device receives an EPResetReq packet from the MA USB host. | N/A | |
| 7.2.3#3 | Any TransferReq packet received targeting the EP in the Inactive State shall be buffered and acknowledged with a TransferResp packet with status code set to INVALID_EP_HANDLE_STATE. | TD 6.18 | |
| 7.2.3#4 | The transition to Inactive state shall not clear outstanding transfers. | N/A | |
| 7.2.3#5 | The MA USB host may transmit EP ClearTransfersReq or CancelTransferReq packets to request clearing | N/A | |

| | | | |
|---|---|---|---|
| | outstanding transfers on an endpoint with an endpoint handle in Inactive State. | | |
| 7.2.3#6 | If the MA USB host transmits an EPHandleDeleteReq packet to transition the EP Handle from Inactive to Not Assigned state, the MA USB device shall cancel any pending requests for the EP Handle. | N/A | |
| **Subsection reference: 7.2.4 Unassigned state** | | | |
| 7.2.4#1 | The Unassigned state is entered from power-on-reset. | N/A | |
| 7.2.4#2 | The Unassigned state is entered from any state if the MA USB device receives DevResetReq packet from the MA USB host. | N/A | |
| 7.2.4#3 | The Unassigned state is entered from the Inactive State if the MA USB device receives an EPHandleDeleteReq packet from the MA USB host. | N/A | |
| 7.2.4#4 | The Unassigned state is entered if the MA USB device receives a USBDevDisconnectReq packet from the MA USB host targeting a device to which the EP handle belongs. | N/A | |
| 7.2.4#5 | The Unassigned state is entered if the MA USB device receives a DevDisconnectReq packet from the MA USB host. | N/A | |
| 7.2.4#6 | If the MA USB PAL receives a TransferReq packet targeting an EP handle in the Unassigned state, then the MA USB device shall respond with a TransferResp packet with status code set to INVALID_EP_HANDLE. | N/A | |
| **Subsection reference: 7.3 Device set up** | | | |
| **Subsection reference: 7.3.1 Discovery mechanism** | | | |
| **Subsection reference: 7.3.2 USB device enumeration** | | | |
| **Subsection reference: 7.3.2.1 USB device handle allocation** | | | |
| 7.3.2.1#1 | The Open Device action triggers the MA USB host to transmit a USBDevHandleReq packet to the MA USB device managing the target USB device. | Interop | |
| 7.3.2.1#2 | The USBDevHandleReq packet shall carry the USB route string for the USB device, the port number on which the USB device is connected to its parent port, and the speed of the USB device. | Interop | |
| 7.3.2.1#3 | Upon receipt of the USBDevHandleReq packet, the target MA USB device shall respond with a USBDevHandleResp packet. | Interop | |
| 7.3.2.1#4 | If the target USB device is already allocated a device handle, then the MA USB device shall respond with a USBDevHandleResp packet carrying the device handle of the USB device and with the value of the Status Code field set to INVALID_REQUEST. | TD 6.22 | |

| | | | |
|---|---|---|---|
| 7.3.2.1#5 | The MA USB host shall ensure that at any given time there is only one USBDevHandleReq packet pending a response. | Interop | |
| **Subsection reference: 7.3.2.2 Endpoint handle allocation** | | | |
| 7.3.2.2#1 | Both Open EP and Configure device actions shall trigger the MA USB host to transmit an EPHandleReq packet to the MA USB device managing the target USB device identified by the Device Handle field in the EPHandleReq packet. | Interop | |
| 7.3.2.2#2 | The EPHandleReq packet is only valid after the USB device handle allocation has been successfully completed. | Interop | |
| 7.3.2.2#3 | When triggered by an Open EP action, the EPHandleReq packet is transmitted to request an EP handle for the default control endpoint on the target USB device. | Interop | |
| 7.3.2.2#4 | An EPHandleReq packet triggered by an Open EP action shall carry a single EP descriptor corresponding to the default control endpoint on the device. | TD 5.18 | |
| 7.3.2.2#5 | The EP0 descriptor included in an EPHandleReq packet for EP0 shall have fields set as described in section 7.3.2.2 of MA USB spec. | TD 5.18 | |
| 7.3.2.2#6 | When triggered by a Configure Device action, the EPHandleReq packet is transmitted to request EP handles for each of the endpoints activated under the selected device configuration. | Interop | |
| 7.3.2.2#7 | Upon receipt of the EPHandleReq packet, the target MA USB device shall respond with an EPHandleResp packet to inform the MA USB host whether the request was successfully completed and to return the attributes of the EP handle. | Interop | |
| **Subsection reference: 7.3.2.3 Modification of EP0 parameters** | | | |
| 7.3.2.3#1 | The Modify EP0 action shall trigger the MA USB host to transmit a ModifyEP0Req packet to the MA USB device in control of the target USB device, which includes the device handle, EP0 handle and maximum packet size for EP0. | TD 5.18 | |
| 7.3.2.3#2 | The MA USB host shall transmit a ModifyEP0Req packet after receiving the response to a Set USB Device Address Request packet sent to a USB device in Default or Address states. | TD 5.18 | |
| 7.3.2.3#3 | Upon receipt of the ModifyEP0Req packet, the target MA USB device shall respond with a ModifyEP0Resp packet. | Interop | |
| 7.3.2.3#4 | The MA USB host PAL shall return the result of ModifyEP0Req to the MA USB host USB core system. | Interop | |
| **Subsection reference: 7.3.2.4 USB device address allocation** | | | |

| | | | |
|---|---|---|---|
| 7.3.2.4#1 | The address device action shall trigger the MA USB host to transmit a SetUSBDevAddrReq packet to the MA USB device managing the target USB device. | TD 5.18 | |
| 7.3.2.4#2 | Upon receipt of a SetUSBDevAddrReq packet the target MA USB device shall respond with a SetUSBDevAddrResp packet to inform the MA USB host whether the request was successful and return the USB device address to the MA USB host. | Interop | |
| **Subsection reference: 7.3.2.5 Update of USB device parameters** | | | |
| 7.3.2.5#1 | The evaluate device action shall trigger the MA USB host to transmit an UpdateDevReq packet to the target MA USB device. | TBD | |
| 7.3.2.5#2 | Upon receipt of the UpdateDevReq packet, the target MA USB device shall respond with an UpdateDevResp packet to inform the MA USB host whether the request was successfully completed. | TD 6.16 | |
| 7.3.2.5#3 | The MA USB host shall return the result of UpdateDevReq to the MA USB host USB core system. | Interop | |
| **Subsection reference: 7.3.3 Enumeration of a USB device downstream of an MA USB hub** | | | |
| 7.3.3#1 | When enumerating a non-integrated USB device behind an MA USB hub the port status change event is not emulated by MA USB host and is a notification event delivered to MA USB host over the air. | Interop | |
| 7.3.3#2 | When enumerating a non-integrated USB device behind an MA USB hub the port manipulation actions are not handled locally by MA USB host and are transferred over the medium to the MA USB hub as control transfer requests. | Interop | |
| **Subsection reference: 7.3.4 Support of Stream Protocol** | | | |
| 7.3.4#1 | In order to use the Enhanced SuperSpeed Stream Protocol on a bulk endpoint, the MA USB host first opens the streams on the endpoint by transmitting an EPOpenStreamReq packet indicating the number of streams to be opened. | TD 5.29 | |
| 7.3.4#2 | The MA USB host shall ensure that the number of streams specified in EPOpenStreamReq is supported by the MA USB device as indicated in CapResp packet. | TD 5.29 | |
| 7.3.4#3 | The MA USB device shall respond to an EPOpenStreamReq packet with an EPOpenStreamResp packet and inform the MA USB host whether the EPOpenStreamReq was successfully completed and return the Stream IDs for the opened streams. | Interop | |
| 7.3.4#4 | If the number of streams requested by the host in an EPOpenStreamReq packet is larger than the value supported by the MA USB device, the MA USB device shall set the value of the Status Code field in | Interop | |

| | | | |
|---|---|---|---|
| | EPOpenStreamResp packet to INSUFFICIENT_RESOURCES. | | |
| 7.3.4#5 | If the number of Stream IDs that are included in the EPOpenStreamResp is less than the number of streams requested by the MA USB host, the host may transmit additional EPOpenStreamReq packets with Open Stream field set to 0 to retrieve the remaining Stream IDs. | TBD | |
| 7.3.4#6 | To change the number of open streams on an endpoint, the host shall first close the open streams on the endpoint. | TBD | |
| 7.3.4#7 | To change the USB device configuration or any other action that requires the endpoint to cease operation, the MA USB host shall first close the open streams on the endpoint. | TBD | |
| 7.3.4#8 | A MA USB host shall transmit an EPCloseStreamReq packet to the MA USB device to close open streams on an endpoint. | TBD | |
| 7.3.4#9 | The endpoint handle of a target endpoint shall be in Inactive state prior to receiving the EPCloseStreamReq packet and all pending transfers are considered cancelled following closing of the streams. | TBD | |
| 7.3.4#10 | The MA USB device shall respond to an EPCloseStreamReq packet with an EPCloseStreamResp packet and inform the MA USB host whether the EPCloseStreamReq was successfully completed. | TBD | |
| **Subsection reference: 7.3.5 USB device reset** | | | |
| 7.3.5#1 | The MA USB host transmits a USBDevResetReq packet to the target MA USB device to request reset of the integrated USB device. | TBD | |
| 7.3.5#2 | In the case of an MA USB hub, the MA USB host informs the MA USB hub of the reset of a downstream USB device. | TBD | |
| 7.3.5#3 | If the USB device is connected to a physical USB controller implemented in the MA USB device, the USBDevResetReq packet informs the controller of transition of a USB device under its control to Default state and triggers relevant actions if applicable. | TBD | |
| 7.3.5#4 | If the MA USB device does not implement a physical USB controller then the USBDevResetReq packet may result in no operation by the MA USB device. | N/A | |
| 7.3.5#5 | The MA USB device shall respond to a USBDevResetReq packet with a USBDevResetResp packet. | TD 6.25 | |

| | | | |
|---|---|---|---|
| 7.3.5#6 | The MA USB host shall follow a USBDevResetResp packet with ModifyEP0Req packet to receive the updated EP0 Handle from the MA USB device. | TBD | |
| 7.3.5#7 | The receipt of a USBDevResetReq packet results in transition of the assigned EP0 Handle to Active state and returns the state of the endpoint to the initial state. | N/A | |

# *Chapter 8 Test Assertions*

| Assertion # | Assertion Description | Test # | Comments |
|---|---|---|---|
| **Subsection reference 8.1 Session management** | | | |
| **Subsection reference 8.1.1 Session states** | | | |
| **Subsection reference 8.1.1.1 Session Down state** | | | |
| 8.1.1.1#1 | The Session Down state is entered upon Power-on-reset | N/A | |
| 8.1.1.1#2 | The Session Down state is entered upon management packet transmission failure. | TD 5.22 | |
| 8.1.1.1#3 | The Session Down state is entered upon DevDisconnectReq/Resp packet exchange. | TD 5.22 | |
| 8.1.1.1#4 | While in Session Down state, there is no communication between the MA USB host and the MA USB device and neither MA USB host nor MA USB device is active. | TD 5.22 | |
| **Subsection reference 8.1.1.2 Session Connecting state** | | | |
| 8.1.1.2#1 | The Session Connecting state is entered when an MA USB host or MA USB device receives an indication from the lower layers that discovery of an MA USB Device or MA USB host has completed. | Interop | |
| 8.1.1.2#2 | The Session Connecting state is entered from the Session Active state following DevResetReq/Resp packet exchange. | Interop | |
| 8.1.1.2#3 | Entering the Session Connecting state triggers the session setup procedure by the MA USB host, consisting of MA USB reset and capability exchange mechanisms. | Interop | |
| 8.1.1.2#4 | While in the Session Connecting state the MA USB device waits for either session setup related packets, or MA USB session teardown related packets from the MA USB host. | Interop | |
| **Subsection reference 8.1.1.3 Session Active State** | | | |
| 8.1.1.3#1 | The Session Active state is entered after USBDevHandleReq/Resp packet exchange following completion of session setup procedure and the decision to continue the session with the MA USB device. | Interop | |
| 8.1.1.3#2 | The Session Active state is entered upon WakeReq/Resp packet exchange. | TD 5.20<br>TD 6.17 | |
| 8.1.1.3#3 | The Session Active state is entered upon implicit WakeReq/Resp packet exchange | | |
| **Subsection reference 8.1.1.4 Session Inactive state** | | | |

| | | | |
|---|---|---|---|
| 8.1.1.4#1 | The Session Inactive state is entered upon SleepReq/Resp packet exchange. | N/A | Not testable |
| 8.1.1.4#2 | While in the Session Inactive state, there is no communication between the MA USB host and the MA USB device except for power management related packets including implicit WakeReq/Resp packets. | TD 5.19 | |
| **Subsection reference 8.1.2 Session setup** | | | |
| 8.1.2#1 | After device discovery is completed, the establishment of a secure communication link between an MA USB host and MA USB device triggers the initiation of the MA USB host PAL on the MA USB host. | Interop | |
| 8.1.2#2 | After device discovery is completed, the establishment of a secure communication link between an MA USB host and MA USB device triggers the initiation of the MA USB device PAL on the device. | Interop | |
| 8.1.2#3 | After being initiated, an MA USB device PAL takes no action and waits for packets from the host. | Interop | |
| **Subsection reference 8.1.2.1 MA USB device reset** | | | |
| 8.1.2.1#1 | An MA USB host transmits an MA USB device Reset Request packet to the target MA USB device, to request the MA USB device to clear all its internal states and join the MSS operated by the MA USB host. | TD 5.18 | |
| 8.1.2.1#2 | The DevResetReq packet shall include the SSID selected by the MA USB host in the SSID field, as well as the nonzero address assigned to the MA USB device by the MA USB host in the MA USB Device Address field. | GTP<br>TD 5.18 | |
| 8.1.2.1#3 | The SSID value selected by the MA USB host shall be a random integer between 1 and 254, and shall be different from any SSID values that the MA USB host has possibly observed during media-agnostic discovery as well as during normal operation after discovery. | TD 5.30 | |
| 8.1.2.1#4 | SSID values 0 and 255 are reserved and shall not be used. | GTP<br>TD 5.30 | |
| 8.1.2.1#5 | An MA USB device address shall be unique within the MSS in which the MA USB device is operating. | Interop | |
| 8.1.2.1#6 | A target MA USB device that responds to a DevResetReq packet and indicates successful reset, shall store the SSID and the device address it received from the DevResetReq packet. | N/A | |
| 8.1.2.1#7 | A target MA USB device that responds to a DevResetReq packet and indicates successful reset, shall ignore any MA USB packets that do not carry the same SSID and device address received | TD 6.31 | |

| | | | |
|---|---|---|---|
| | in the DevResetReq packet with the exception of PingReq packets with MA USB Device Address field set to 0xFF. | | |
| 8.1.2.1#8 | An MA USB host shall use the same SSID and device address values as sent in a DevResetReq packet in all following packet exchanges with the target MA USB device. | GTP<br>TD 5.30 | |
| 8.1.2.1#9 | An MA USB host shall not transmit any packet other than a DevResetReq packet to an MA USB device unless the host has previously received an DevResetResp packet from the device with the Status Code field set to 0 (SUCCESS). | TD 5.28 | |
| **Subsection reference 8.1.2.2 Capability exchange** | | | |
| 8.1.2.2#1 | For an MA USB capability exchange, the MA USB host transmits an MA USB Capability Request packet to the target MA USB device. | Interop | |
| 8.1.2.2#2 | After a successful MA USB device capability exchange, the session between an MA USB host and an MA USB device is considered established. | N/A | |
| 8.1.2.2#3 | Establishment of an MA USB session shall trigger emulation of one Port Status Change event in the MA USB host if the session is established with an MA USB device or an MA USB hub with an integrated USB 2.0 hub. | Interop | |
| 8.1.2.2#4 | Establishment of an MA USB session shall trigger emulation of two Port Status Change events in the MA USB host if the session is established with an MA USB hub with an integrated USB 3.1 hub. | Interop | |
| **Subsection reference 8.1.3 Session tear down** | | | |
| 8.1.3#1 | In case of an explicit tear down, the initiator shall notify the peer MA USB entity that the MA USB session is being torn down. | TD 5.22 | |
| 8.1.3#2 | When a loss of connectivity is detected, the MA USB host and MA USB device shall each locally initiate session tear down procedures. | Interop | |
| **Subsection reference 8.1.3.1 Implicit session tear down** | | | |
| 8.1.3.1#1 | Implicit tear down shall be initiated when either an MA USB host or MA USB device is notified by the lower layers of the loss of connectivity. | TD 5.22 | |
| 8.1.3.1#2 | Implicit tear down shall be initiated when the transmission failure of a management packet indicates a connection loss. | TD 5.22 | |
| 8.1.3.1#3 | In an implicit session tear down, an MA USB host shall emulate a port status change equivalent to the unplugging of a wired USB device from one of the | Interop | |

| | | | |
|---|---|---|---|
| | root ports, and shall handle all actions resulting from the port status change locally. | | |
| 8.1.3.1#4 | In an implicit session tear down, an MA USB device shall emulate USB device removal and clear all allocated resources for the session. | Interop | |
| **Subsection reference 8.1.3.2 Host initiated session tear down** | | | |
| 8.1.3.2#1 | Following successful completion of the USB device removal procedure, an MA USB host shall transmit an DevDisconnectReq packet to the MA USB device to explicitly terminate the session between the MA USB host and MA USB device. | TD 5.22 | |
| 8.1.3.2#2 | Following receipt of a DevDisconnectResp packet, an MA USB host shall clear all the resources allocated to the USB device as well as all the resources allocated to the MA USB device. | TD 5.22 | |
| **Subsection reference 8.1.3.3 Device initiated session tear down** | | | |
| 8.1.3.3#1 | The MA USB host shall respond to a DevInitDisconnectReq packet with an MA USB device Initiated Disconnect Response packet. | TD 5.22 | |
| 8.1.3.3#2 | Upon receipt of a DevInitDisconnectReq packet, an MA USB host shall initiate session tear down. | N/A | |
| **Subsection reference 8.1.3.4 USB device removal procedure** | | | |
| 8.1.3.4#1 | The first step in the USB device removal procedure consists of the MA USB host inactivating and clearing all outstanding transfers on all active endpoints. | TD 5.22 | |
| 8.1.3.4#2 | The second step in the USB device removal process consists of the MA USB host deleting all EP handles except for the EP handle of EP0. | TD 5.22 | |
| 8.1.3.4#3 | The third step in the USB device removal process consists of the MA USB host deleting the EP handle of EP0. | TD 5.22 | |
| 8.1.3.4#4 | The fourth and final step in the USB device removal process consists of the MA USB host sending a DevDisconnectReq packet to the MA USB device. | TD 5.22 | |
| **Subsection reference 8.2 Power management** | | | |
| **Subsection reference 8.2.1 Transition to Session Inactive state** | | | |
| **Subsection reference 8.2.1.1 Initiation by the MA USB host** | | | |
| 8.2.1.1#1 | An MA USB host initiating the process to move its session state to Session Inactive must first inactivate all EP handles on the target MA USB device. | TD 5.19 | |
| 8.2.1.1#2 | An MA USB host initiating the process to move its session state to Session Inactive must second | TD 5.19 | |

| | | | |
|---|---|---|---|
| | suspend the integrated USB device behind the MA USB device. | | |
| 8.2.1.1#3 | An MA USB host initiating the process to move its session state to Session Inactive must third and lastly transmit a SleepReq packet to the target MA USB device. | TD 5.19 | |
| 8.2.1.1#4 | To inactivate EP handles on a target MA USB device, an MA USB host transmits one or more EPInactivateReq packets to the MA USB device. | TD 5.19 | |
| 8.2.1.1#5 | To suspend the integrated USB device behind a target MA USB device an MA USB host transmits a USBSuspendReq packet to the MA USB device. | TD 5.19 | |
| 8.2.1.1#6 | After a target MA USB device transmits a SleepResp packet with the Status Code field set to 0 (SUCCESS), it shall move its session state to Session Inactive and instruct its local management entity for the MA link connecting the MA USB device and the MA USB host to perform the required actions to put the MA link in a suitable low-power mode that meets the timeout values specified in the SleepResp packet. | TD 6.29 | |
| 8.2.1.1#7 | An MA USB host shall move its session state to Session Inactive after receiving a SleepResp packet with the Status Code field set to 0 (SUCCESS) and instruct its local management entity for the MA link connecting the MA USB host and the target MA USB device to perform required actions to put the MA link in a suitable low-power mode that meets the timeout values specified in the SleepReq packet. | TD 5.25 | |
| 8.2.1.1#8 | When transmitting an EPInactivateReq packet, an MA USB host shall set the Suspend Flag field to 1 to indicate to the MA USB device that an EP handle is being inactivated in preparation for the transition to the Suspend state. | TD 5.19 | |
| 8.2.1.1#9 | An MA USB host shall not transmit an EPInactivateReq packet with the Suspend Flag field set to 1 to target EP handles that are already in the Inactive state. | TD 5.19 | |
| **Subsection reference 8.2.1.2 Initiation by the MA USB device** | | | |
| 8.2.1.2#1 | When both the MA USB host and the target MA USB device indicate support for Link Sleep capability, an MA USB device may initiate the process to move its session state to Session Inactive if it has only control or non-isochronous IN endpoints behind it, all IN endpoints are in the pending state, and it has had the pending state for each IN endpoint acknowledged by the MA USB host PAL. | TD 6.28 | |
| 8.2.1.2#2 | Once an MA USB device has received the MA USB host acknowledgement to the pending state for all its IN endpoints, it may initiate the process to move its | TD 6.28 | |

| | | | |
|---|---|---|---|
| | session state to Session Inactive by transmitting a SleepReq packet to the MA USB host. | | |
| 8.2.1.2#3 | After an MA USB host transmits a SleepResp packet with the Status Code field set to 0 (NO_ERROR), it shall move its session state to Session Inactive, and instruct its local management entity for the MA link connecting the MA USB host and the MA USB device to perform required actions to put the MA link in a suitable low-power mode that meets the timeout values specified in the SleepResp packet. | TD 5.25 | |
| 8.2.1.2#4 | Upon receiving a SleepResp packet with the Status Code field set to 0 (SUCCESS), an MA USB device shall move its session state to Session Inactive and instruct its local management entity for the MA link connecting the MA USB device and the MA USB host to put the MA link in a suitable low-power mode that meets the timeout values specified in the SleepReq packet. | TD 6.28<br>TD 6.29 | |
| **Subsection reference 8.2.2 Transition to Session Active state** | | | |
| **Subsection reference 8.2.2.1 Initiation by the MA USB host** | | | |
| 8.2.2.1#1 | An MA USB host may move its session state from Session Inactive to Session Active by instructing its local management entity for the MA link connecting the MA USB host and the target MA USB device to perform required actions to exit the low-power mode, then transmitting a WakeReq packet to the target MA USB device. | N/A | |
| 8.2.2.1#2 | In response to a WakeReq packet, a target MA USB device shall release a WakeResp packet with the Status Code field set to 0 (NO_ERROR) to the management channel, move its session state to Session Active, and instruct its local management entity for the link connecting the MA USB device and the MA USB host to perform the required actions to exit the low-power mode. | N/A | Not tested |
| 8.2.2.1#3 | An MA USB host shall move its session state to Session Active after receiving a WakeResp packet. | N/A | Not tested |
| **Subsection reference 8.2.2.2 Initiation by an MA USB device** | | | |
| **Move Session to Session Active state through a WakeReq packet (explicit request)** | | | |
| 8.2.2.2#1 | If both an MA USB host and MA USB device indicate support for the Link Sleep capability, and the MA USB device has moved its session state to Session Inactive, the MA USB device may transmit a WakeReq packet to initiate the process to move its session state to Session Active. | Interop | |
| 8.2.2.2#2 | An MA USB device shall instruct its local management entity for the MA link connecting the MA USB device and the MA USB host to perform required actions to | TD 6.28 | |

| | | | |
|---|---|---|---|
| | exit the low-power mode before transmitting a WakeReq packet to the MA USB host. | | |
| 8.2.2.2#3 | In response to a WakeReq packet, an MA USB host shall release a WakeResp packet with the Status Code field set to 0 (NO_ERROR) to the management channel, move its session state to Session Active, and instruct its local MA link management entity to perform required actions to exit the low-power mode. | TD 5.19<br>TD 5.20 | |
| **USB Remote wake (implicit request)** | | | |
| 8.2.2.2#4 | An MA USB device with integrated USB device in Suspend state may transmit a RemoteWakeReq packet. | TD 6.21 | |
| 8.2.2.2#5 | In response to a remote wake indication by the integrated USB device, an MA USB device shall instruct its local management entity for the MA link connecting the MA USB device and the MA USB host to perform the required actions to exit the low-power mode and transmit a RemoteWakeReq packet to the MA USB host, with the Device Handle field in the packet identifying the USB device that initiated the remote wake function. | TD 6.21 | |
| 8.2.2.2#6 | Upon receiving a RemoteWakeReq packet while in Session Inactive session state, the MA USB host shall move its session state to Session Active. | TD 5.19 | |
| 8.2.2.2#7 | In response to a RemoteWakeReq packet with the USB Device Resumed field set to 0, an MA USB host shall first move the integrated USB device behind the MA USB device PAL out of the Suspend state by transmitting a USBResumeReq packet, and then activate the relevant endpoints on the MA USB device by transmitting one or more EPActivateReq packets. | TD 5.19 | |
| 8.2.2.2#8 | In response to a RemoteWakeReq packet with the USB Device Resumed field set to 1, an MA USB host shall directly activate the relevant endpoints on the MA USB device without transmitting a USBResumeReq packet to the MA USB device. | TD 5.19 | |
| **Data or management packet exchange (implicit request)** | | | |
| 8.2.2.2#9 | When both an MA USB host and target MA USB device indicate support for Link Sleep capability and the MA USB device has control or non-isochronous IN endpoints only with all IN endpoints in the pending state, and the pending state for each IN endpoint acknowledged by the MA USB host, the MA USB device may move its session state from Session Inactive to Session Active, and trigger a similar transition in the MA USB host, by transmitting any packet that requires a response. | TD 5.20<br>TD 6.28 | |

| | | | |
|---|---|---|---|
| 8.2.2.2#10 | In response to detecting traffic from a previously pending IN endpoint, an MA USB device shall instruct its local management entity for the MA link connecting the MA USB device and the MA USB host to perform the required actions to exit the low-power mode, and resume a pending IN transfer by transmitting one or more TransferResp packets belonging to the transfer, with the EoT field set to 1 in at least the first TransferResp packet that the MA USB device PAL transmits. | N/A | |
| 8.2.2.2#11 | In response to the first TransferResp packet received while its session is in Session Inactive state, an MA USB host shall move its session state to Session Active, instruct its local management entity for the MA link connecting the MA USB host and the MA USB device to perform required actions to exit the low-power mode, and reset the transfer timers for the pending transfer request the received TransferResp packet identifies through its Request ID field. | N/A | Not tested |

## *Chapter 9 Test Assertions*

| Assertion # | Assertion Description | Test # | Comments |
|---|---|---|---|
| **Subsection reference 9.1 MA USB hub enumeration** | | | |
| 9.1#1 | The enumeration of an integrated USB hub on an MA USB device is the same as enumeration of a non-hub integrated USB device, unless the integrated hub is a USB 3.1 hub. | Interop | |
| 9.1#2 | If an integrated MA USB hub is a USB 3.1 hub, the integrated USB device enumeration procedure shall occur twice. Enumeration of an MA USB hub with an integrated USB 3.1 hub consists of the enumeration of the Enhanced SuperSpeed hub as a standalone integrated USB device followed by the enumeration of the USB 2.0 hub as a standalone integrated USB device. | Interop | Note - for a fully compliant hub it does not matter which hub is enumerated first, but it is recommended that the Enhanced SuperSpeed hub be enumerated first. |
| **Subsection reference 9.2 MA USB hub session tear down** | | | |
| 9.2#1 | The session teardown for an MA USB hub is similar to an MA USB device except that if the MA USB hub integrates a USB 3.1 hub, there are two port status change events and USB device removal events and procedures instead of one. | Interop | |
| 9.2#2 | The session teardown for an MA USB hub is similar to an MA USB device except that the USB device removal procedure is first carried out for all the USB devices downstream of the integrated hub(s), and then for the integrated hub(s) on the MA USB hub. | Interop | |
| **Subsection reference 9.2.1** | | | |
| 9.2.1#1 | Removal of a USB device connected downstream of an MA USB hub follows the USB device removal procedure outlined in section 8.1.3.4 except that the host is notified of the port status change event on the MA USB hub via USB control transfers. | Interop | |
| **Subsection reference 9.3 MA USB hub power management** | | | |
| 9.3#1 | For transition of the session state of an MA USB hub to Inactive State, all the endpoints on and behind the integrated USB 2.0 and the Enhanced SuperSpeed hubs shall be first in the Suspend state and the corresponding endpoint handles deactivated. | TBD | |
| 9.3#2 | For transition of the session state of an MA USB hub to Inactive State, after all the endpoints on and behind the integrated USB 2.0 and enhanced SuperSpeed hubs are suspended and their corresponding endpoint handles deactivated, one USBSuspendReq packet shall be transmitted for each integrated hub to transition the hub to the Suspend state. | TBD | |

| | | | |
|---|---|---|---|
| 9.3#3 | For transition of the session state of an MA USB hub to Inactive State, after one USBSuspendReq packet has been transmitted for each integrated hub, a single SleepReq packet shall then move the session state to the Inactive State. | TBD | |
| 9.3#4 | For the session state transition of an MA USB hub out of the inactive State, one WakeReq packet is transmitted by the MA USB host to the MA USB hub. | TBD | |
| 9.3#5 | For the session state transition of an MA USB hub out of the Inactive State, a WakeReq packet shall be followed by USBResumeReq packets for each integrated hub and the transition of all the endpoints and their corresponding endpoint handles on and behind the integrated USB 2.0 and/or the Enhanced SuperSpeed hubs to Active state. | TBD | |

# 3 Definitions

| Term | Definition |
|---|---|
| MA USB Sniffer | A media-dependent analyzer capable of viewing and decoding MA USB traffic. |
| MA USB Packet Analysis Software | Software that parses an MA USB Sniffer trace. |
| HUT | The Host Under Test. |
| DUT | The Device Under Test. |
| MA USB Compliance Host | An MA USB host or exerciser that can be directed to behave in a specific way depending on the test. Used for device-based testing. |
| MA USB Compliance Device | An MA USB device or exerciser that can be directed to behave in a specific way depending on the test. Used for host-based testing. Includes an MA USB device or exerciser with hub functionality. |
| USB Compliance Device | LS, FS, HS, or SS wired compliance devices running loopbacks or other USB test cases. Note that USB compliance devices will require an OS-specific device driver for anything beyond basic enumeration. |
| KG USB | Known Good USB. Refers to a hub, host or device that has been certified USB compliant. |
| KG MA USB | Known Good MA USB. Refers to a hub, host or device that has been certified MA USB compliant. |

# 4 MA USB Compliance Checklist

| Ref. | Field (all fields must be filled in) | |
|---|---|---|
| *Vendor Information* | | |
| | Vendor Name: | |
| | Vendor Street Address: | |
| | Vendor City, State, Postal Code: | |
| | Vendor Country: | |
| | Vendor Contact: | |
| | Vendor Phone Number: | |
| | Vendor Email: | |
| *Product Information* | | |
| | Product Name: | |
| | Product Model Number: | |
| | Product Revision Level: | |
| | Product Test ID Number (issued by USB-IF): | |
| | Product Silicon Name: | |
| | Product Silicon Model Number: | |
| | Product Silicon Revision Level: | |
| | Product Silicon Test ID Number (issued by USB-IF): | |
| | Product or Building Block (select one): | ☐ Consumer Product ☐ Building Block<br>☐ Other (please specify) _____ |
| | Type of Product (select one) | ☐ Host ☐ Integrated Device<br>☐ 2.0 Hub ☐ 3.0 Hub |
| | Message Interface: | ☐ Graphics Screen ☐ Text Screen<br>☐ Other (please specify) _____ |
| | Operating System and Version: | |
| *Supported Features and Capabilities* | | |
| F1 | Signaling speeds: | ☐ SuperSpeed Plus ☐ SuperSpeed ☐ High ☐ Full ☐ Low |
| F2 | Media: | ☐ Wi-Fi ☐ Wi-Gig ☐ TCP/IP<br>☐ Other (please specify) _____ |
| F3 | Endpoint types: | ☐ Bulk IN ☐ Bulk OUT<br>☐ Interrupt IN ☐ Interrupt OUT<br>☐ Isochronous IN ☐ Isochronous OUT |
| F4 | Power management: | ☐ Link Sleep ☐Remote Wake |
| F5 | Elastic Buffer Capability:<br>*Note: Host must support* | ☐ Yes ☐ No |
| F6 | Enhanced SuperSpeed Stream Protocol: | ☐ Yes ☐ No |
| F7 | Link-Managed Out Transfers: | ☐ Yes ☐ No |
| F8 | GetPortBWReq Packets: | ☐ Yes ☐ No |
| F9 | DWORD-aligned isochronous payload required : | ☐ Yes ☐ No ☐ Isoch not Supported |
| F10 | Access to synchronized Media Time :<br>*Note: not applicable to Hosts* | ☐ Yes ☐ No ☐ Isoch not Supported<br>☐ N/A |
| F11 | Need to receive MA USB timestamps regardless of endpoint configuration:<br>*Note: not applicable to Hosts* | ☐ Yes ☐ No ☐ Isoch not Supported<br>☐ N/A |

| *Supported Features and Capabilities Cont.* | | | | |
|---|---|---|---|---|
| F12 | P-Managed OUT Optional Capabilities<br>*Note: not applicable to Hosts* | ☐ Yes | ☐ No | ☐ N/A |
| F13 | Container IDs<br>*Note: not applicable to Hosts* | ☐ Yes | ☐ No | ☐ N/A |
| F14 | DROPPED_PACKET status return<br>*Note: not applicable to Hosts* | ☐ Yes | ☐ No | ☐ N/A |
| *Parameters* | | | | |
| F15 | MAC address: | | | |
| F16 | IP address: | | | |
| F17 | aBulkTransferRetries:<br>*Note: minimum value is 5* | | | |
| F18 | aControlTransferRetries:<br>*Note: minimum value is 5* | | | |
| F19 | aInterruptTransferRetries:<br>*Note: minimum value is 3* | | | |
| F20 | aManagementRetries:<br>*Note: minimum value is 4* | | | |
| F21 | aTransferSetupRetries:<br>*Note: minimum value is 4* | | | |
| F22 | Max number of endpoints DUT can support<br>*Note: not applicable to Hosts* | *Note: minimum value for MA USB hubs is 16* | | |
| F23 | Max number of USB devices DUT can manage<br>*Note: not applicable to Host* | *Note: only MA USB hubs can manage more than 1 device* | | |
| Signature of Preparer: | | Date: | | |

# 5 Test Descriptions for MA USB Host

*Note: The MA USB Compliance Device will ignore power management packets from the HUT unless such packets are explicitly mentioned in the test being performed.*

*Note: Unless otherwise mentioned, the Compliance Device will include a Speed Capability Descriptor, and a Link Sleep Capability Descriptor with the Link Sleep Capable field set to 1, in a CapResp packet.*

## General Test Procedure

**Required Equipment:**

1. MA USB Sniffer
2. MA USB Packet Analysis Software

The following test steps will be run in conjunction with the tests in this chapter on all MA USB packets generated by the HUT during other test procedures. Unless otherwise specified, the entire test will be recorded by the MA USB Sniffer and the trace saved in a form readable by the MA USB Analysis Software. Packets from the HUT will be identified by MAC address.

1. **For all packets (Common Header Fields):**
   a. Verify that packet is not greater than 64K. (6.1#1)
   b. Verify that Version field is 0. (6.2.1.1#1)
   c. Verify that Host field is 1. (6.2.1.2#1)
   d. Verify that the fourth bit in the Flags field is 0 (reserved). (6.2.1.2#4)
   e. Verify that the Type field is not 11b. (6.2.1.3#1)
   f. Verify that the Type and Subtype fields contain a value listed in Table 5 (6.2.1.3#2)
   g. Verify that the Length field is equal to the length of the packet in Bytes. (6.2.1.4#1)
   h. Verify that the Status Code field is one of the values in Table 6. (6.2.1.8#1)
   i. If the Status Code field is set to a value other than 0 (SUCCESS), verify that the packet contains all fields defined for its subtype. (6.2.1.8#2)
   j. Verify that the SSID field is the same as the SSID field in all previous packets. (8.1.2.1#2)
   k. Verify that the SSID field is not 0 or 255. (8.1.2.1#4)
   l. Verify that the same SSID and device address values as sent in a DevResetReq packet are used in all following packet exchanges with the target MA USB device. (8.1.2.1#8)
2. **For Management packets only:**
   a. Verify that bits 18 through 31 of DWORD 2 are set to 0 (reserved). (6.3#1)
   b. If the packet is not a CancelTransferReq packet or a DevResetReq packet, and the Device Address is not set to 0xFF (broadcast), verify that the Dialog Token field increments by 1 for each new management packet carrying a request. (6.3.1.1#1)
   c. Verify that the Dialog Token field is set to 1 after reaching aMaxDialogToken. (6.3.1.1#1)
   d. Verify that the Dialog Token field is set to 0 when the MA USB Device Address is 0xFF (broadcast). (6.3.1.1#4)
   e. Verify that when connected to an MA USB device that is not a hub, no GetPortBWReq packets are sent. (6.3.54#4)
3. **For Data packets only:**
   a. Verify that the EPS field is 0 (reserved). (6.5.1.1#1)
   b. If packet type is not TransferReq, verify that the ARQ field is 0 (reserved). (6.5.1.2#2)
   c. If the NEG subfield in the T-Flags field is 1, verify that the device supports elastic buffer capability. (6.5.1.2#3)
   d. Verify that a control TransferReq packet with sequence number set to 0 contains control transfer setup data as the first two DWORDs of its payload. (6.5.1.2#7)
   e. Verify that bit 15 in the T-Flags field is 0 (reserved). (6.5.1.2#8)
   f. Verify that the Stream ID field is 0 (reserved) unless the target endpoint is an Enhanced SuperSpeed bulk endpoint that supports the Enhanced SuperSpeed Stream Protocol. (6.5.1.3#1)

g. If a packet is a TransferResp for an OUT transfer, verify that it does not have the status code field set to 135 (TRANSFER_PENDING).

4. **For TransferReq packets only:**
   a. Verify that the TransferReq packets of a bulk IN or interrupt IN transfer carry no payload. (6.5.2#3)
   b. Verify that all packets are a multiple of wMaxPacketSize, except the last packet in a transfer, which can be any size that is less than or equal to wMaxPacketSize. (5.5#3)
   c. Verify that the Sequence Number is 0 for the first TransferReq packet in a control transfer. (5.7.1#3)
5. **For Isochronous Data packets only:**
   a. For an isochronous OUT TransferReq, the Number of Headers field is set to the number of isochronous headers in the packet. (6.5.1.7#1)
   b. For an isochronous IN TransferReq, the Number of Headers field is set to the number of IRS headers in the packet. (6.5.1.7#2)
   c. Unless the packet is an isochronous IN TransferReq, verify that the Number of Headers field is 0 (reserved). (6.5.1.7#3)
   d. Verify that the beginning of the packet contains isochronous headers placed one after the other in increasing order of Segment Number. (5.10.1.1#1)
6. **While the HUT is connected to an MA USB Device:**
   a. Verify that if the MA USB link is in Session Inactive, the HUT does not send any packets without first sending a WakeReq packet (8.1.1.4#2).

# TD 5.1 Management Packet Response Test

**Assertions Covered**

1. 5.2.1.1#3, 5.2.1.1#9, 5.11#2
2. 6.3.56#4, 6.3.57#5

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. MA USB Compliance Device with hub functionality

**Test Steps**

1. If the Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
   a. Connect/reconnect the Compliance Device to the HUT.
   b. Enumerate and configure the Compliance Device to support Link Sleep.
2. Ensure that the following steps are captured by the MA USB Sniffer.
3. Instruct the Compliance Device to send a DevNotificationReq packet to the HUT.
4. Verify that the HUT transmits a DevNotificationResp packet in response to the DevNotificationReq packet. (5.11#2)
5. Verify that the DevNotificationResp packet is received within aManagementResponseTime + 2 * aManagementChannelDelay from when the DevNotificationReq packet was released to the management channel. (5.2.1.1#3)
6. Instruct the Compliance Device to send an EPSetKeepAliveReq packet to the HUT.
7. Verify that an EPSetKeepAliveResp packet is received from the HUT within aManagementResponseTime + 2 * aManagementChannelDelay from when the EPSetKeepAliveReq packet was released to the management channel. (5.2.1.1#3)
8. Instruct the Compliance Device to resend the EPSetKeepAliveReq packet with the Retry flag set to 1, after receiving the EPSetKeepAliveResp from the HUT.
9. Verify that the HUT responds with another EPSetKeepAliveResp packet. (5.2.1.1#9)
10. Instruct the Compliance Device to send a PingReq packet to the HUT.
11. Verify that a PingResp packet is received from the HUT within aManagementResponseTime + 2 * aManagementChannelDelay from when the PingReq packet was released to the management channel. (5.2.1.1#3)
12. If the HUT supports Link Sleep:
    a. Instruct the Compliance Device to send a TransferResp packet with the Status Code field set to 135 (TRANSFER_PENDING) and the ARQ bit set to 1.

   b. Wait for the HUT to send a TransferAck packet acknowledging the TransferResp packet from the Compliance Device.
   c. Instruct the Compliance Device to send a SleepReq packet to the HUT.
   d. Verify that a SleepResp packet is received from the HUT within aManagementResponseTime + 2 * aManagementChannelDelay from when the SleepReq packet was released to the management channel. (5.2.1.1#3)
   e. Instruct the Compliance Device to resend the SleepReq packet with the Retry flag set to 1, after receiving the SleepResp from the HUT.
   f. Verify that the HUT responds with another SleepResp packet. (5.2.1.1#9)
   g. Instruct the Compliance Device to send the HUT a WakeReq packet.
   h. Verify that a WakeResp packet is received from the HUT within the number of milliseconds indicated in the Management Request Timeout field of the SleepResp packet. (5.2.1.1#3, 6.3.57#5)
   i. Instruct the Compliance Device to resend the WakeReq packet with the Retry flag set to 1, after receiving the WakeResp from the HUT.
   j. Verify that the HUT responds with another WakeResp packet. (5.2.1.1#9)
13. Let the Compliance Device sit idle for a period of time sufficient for HUT to suspend it.
14. Wait for the HUT to initiate transition of the MA USB session with the Compliance Device to the Inactive State.
15. Instruct the Compliance Device to respond normally to each management packet from the HUT.
16. After sending a SleepResp packet to the HUT, instruct the Compliance Device to send a RemoteWakeReq packet to the HUT.
17. Verify that a RemoteWakeResp packet is received from the HUT within the number of milliseconds indicated in the Management Request Timeout field of the SleepReq packet. (5.2.1.1#3, 6.3.56#4)
18. Instruct the Compliance Device to resend the RemoteWakeReq packet with the Retry flag set to 1, after receiving the RemoteWakeResp from the HUT.
19. Verify that the HUT responds with another RemoteWakeResp packet. (5.2.1.1#9)
20. Instruct the Compliance Device to send a DevInitDisconnectReq packet to the HUT.
21. Verify that a DevInitDisconnectResp packet is received from the HUT within aManagementResponseTime + 2 * aManagementChannelDelay from when the DevInitDisconnectReq packet was released to the management channel. (5.2.1.1#3)
22. Instruct the Compliance Device to resend the DevInitDisconnectReq packet with the Retry flag set to 1, after receiving a DevInitDisconnectResp from the HUT.
23. Verify that the HUT responds with another DevInitDisconnectResp packet. (5.2.1.1#9)

# TD 5.2 Management Packet Exchange Timeout Test

**Assertions Covered**

1. 5.2.1.1#4, 5.2.1.1#5, 5.2.1.1#6, 5.2.1.1#7

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. MA USB Compliance Device

**Test Steps**

1. Disconnect the Compliance Device if previously connected.
2. Ensure that the following steps are captured by the MA USB Sniffer.
3. Connect the Compliance Device to the HUT.
4. Start enumeration of the Compliance Device.
5. Instruct the Compliance Device to stop responding to the HUT after it receives a USBDevHandleReq packet.
6. Verify that the HUT waits at least aManagementRequestTimeout before retrying any USBDevHandleReq packets. (5.2.1.1#4)
7. Verify that each retransmitted test packet has the same fields as the original request, except Retry field is 1. (5.2.1.1#5)
8. Verify that the HUT retries the USBDevHandleReq aManagementRetries times. (5.2.1.1#6)
9. If the HUT initiates the ping protocol, wait for the HUT to stop sending PingReq packets. If the HUT does not initiate the ping protocol, continue to the next step.

10. Wait for 1 minute.
11. Verify that the HUT did not send any more MA USB packets to the Compliance Device. (5.2.1.1#7)
12. Disconnect the Compliance Device.

**Repetitions**

- Repeat test for each of the following packet types with that packet type replacing the USBDevHandleReq packet.
    - EPHandleReq
    - ModifyEP0Req
    - SetUSBDevAddrReq
    - UpdateDevReq

# TD 5.3 Data Packet Retry Test

**Assertions Covered**

1. 5.2.1.2#4, 5.2.1.2#5, 5.2.1.2#6, 5.2.1.2#7, 5.2.1.2#8, 5.2.1.2#12, 5.4#18, 5.4#19, 5.5#25
1. 6.2.1.2#2, 6.3.34#1, 6.3.34#2, 6.3.34#4

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. MA USB Compliance Device with hub functionality
4. KG USB Mass Storage Device
5. KG USB Mouse

**Test Steps**

**Case 1 – Failed Control IN/OUT Transfers**

1. Disconnect the Compliance Device if previously connected.
2. Connect the Compliance Device to the HUT.
3. Start enumeration of the Compliance Device.
4. Instruct the Compliance Device to stop responding to any MA USB packets after receiving a TransferReq packet carrying a USB GetDescriptor request.
5. Verify that the HUT retries the TransferReq packet with the USB GetDescriptor request. (5.2.1.2#4)
6. Verify that the TransferReq packet with the USB GetDescriptor request is retried aControlTransferRetries number of times. (5.2.1.2#6).
7. Verify that each retried TransferReq packet:
    a. Does not change the original transfer size. (5.4#18)
    b. Has the same fields as the original TransferReq packet, except that the Retry field is 1. (5.2.1.2#5, 6.2.1.2#2)
8. Verify that the HUT sends a PingReq packet. (5.2.1.2#7)
9. Verify that the PingReq packet has:
    a. Type field set to 0. (6.3.34#1)
    b. Subtype field set to 32. (6.3.34#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.34#2)
    d. Device Handle field set to 0 (reserved). (6.3.34#4)
10. Verify that the HUT reports failure of the data transfer to the user. (5.2.1.2#8)
11. Disconnect then reconnect the Compliance Device to the HUT.
12. Start enumeration of the Compliance Device.
13. Instruct the Compliance Device to stop responding to any MA USB packets after receiving a TransferReq packet carrying a USB SetConfig request.
14. Verify that the HUT retries the TransferReq packet with the USB SetConfig request. (5.2.1.2#4)
15. Verify that the TransferReq packet with the USB SetConfig request is retried aControlTransferRetries number of times. (5.2.1.2#6).
16. Verify that each retried TransferReq packet:
    a. Does not change the original transfer size. (5.4#18)
    b. Has the same fields as the original TransferReq packet, except that the Retry field is 1. (5.2.1.2#5, 6.2.1.2#2)

    c. Has the same payload and Remaining Size field as the original TransferReq packet. (5.5#25)
17. Verify that the HUT sends a PingReq packet. (5.2.1.2#7)
18. Verify that the PingReq packet has:
    a. Type field set to 0. (6.3.34#1)
    b. Subtype field set to 32. (6.3.34#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.34#2)
    d. Device Handle field set to 0 (reserved). (6.3.34#4)
19. Verify that the HUT reports failure of the data transfer to the user. (5.2.1.2#8)

**Case 2 – Failed Bulk IN/OUT Transfers**

1. Disconnect then reconnect the Compliance Device to the HUT.
2. Enumerate and configure the Compliance Device.
3. Connect the USB Device to the Compliance Device.
4. Enumerate and configure the USB Mass Storage Device.
5. Ensure that the following steps are captured by the MA USB Sniffer.
6. Initiate a bulk IN transfer.
7. Instruct the Compliance Device to send a TransferResp packet to the HUT after receiving the first TransferReq packet for the bulk IN transfer, then stop responding to any MA USB packets.
8. Verify that the HUT retries the TransferReq packet. (5.2.1.2#4)
9. Verify that the HUT waits at least aTransferKeepAlive after receiving the TransferResp packet before retrying the TransferReq packet. (5.2.1.2#12)
10. Verify that the TransferReq packet is retried aBulkTransferRetries number of times. (5.2.1.2#6).
11. Verify that each retried TransferReq packet:
    a. Does not change the original transfer size. (5.4#18)
    b. Has the same fields as the original TransferReq packet, except that the Retry field is 1. (5.2.1.2#5, 6.2.1.2#2)
12. Verify that the HUT sends a PingReq packet. (5.2.1.2#7)
13. Verify that the PingReq packet has:
    a. Type field set to 0. (6.3.34#1)
    b. Subtype field set to 32. (6.3.34#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.34#2)
    d. Device Handle field set to 0 (reserved). (6.3.34#4)
14. Verify that the HUT reports failure of the data transfer to the user. (5.2.1.2#8)
15. Disconnect the Compliance Device from the HUT.
16. Reconnect the Compliance Device to the HUT.
17. Initiate a bulk OUT transfer.
18. Instruct the Compliance Device to stop responding to any MA USB packets after receiving the first TransferReq packet.
19. Verify that the HUT retries the TransferReq packet. (5.2.1.2#4)
20. Verify that the TransferReq packet is retried aBulkTransferRetries number of times. (5.2.1.2#6).
21. Verify that each retried TransferReq packet:
    a. Does not change the original transfer size. (5.4#18)
    b. Has the same fields as the original TransferReq packet, except that the Retry field is 1. (5.2.1.2#5, 6.2.1.2#2)
    c. Has the same payload and Remaining Size field as the original TransferReq packet. (5.5#25)
22. Verify that the HUT sends a PingReq packet. (5.2.1.2#7)
23. Verify that the PingReq packet has:
    a. Type field set to 0. (6.3.34#1)
    b. Subtype field set to 32. (6.3.34#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.34#2)
    d. Device Handle field set to 0 (reserved). (6.3.34#4)
24. Verify that the HUT reports failure of the data transfer to the user. (5.2.1.2#8)
25. Disconnect the USB Mass Storage device and Compliance Device.
26. Reconnect the Compliance Device to the HUT.
27. Enumerate and Configure the Compliance Device.
28. Connect the USB Mouse to the Compliance Device.

29. Enumerate and configure the USB Mouse.
30. Instruct the MA USB Compliance Device to stop responding to any MA USB packets after receiving the first TransferReq packet.
31. Verify that the HUT retries the TransferReq packet. (5.2.1.2#4)
32. Verify that the TransferReq packet is retried aInterruptTransferRetries number of times. (5.2.1.2#6).
33. Verify that each retried TransferReq packet:
    a. Does not change the original transfer size. (5.4#18)
    b. Has the same fields as the original TransferReq packet, except that the Retry field is 1. (5.2.1.2#5, 6.2.1.2#2)
34. Verify that the HUT sends a PingReq packet. (5.2.1.2#7)
35. Verify that the PingReq packet has:
    a. Type field set to 0. (6.3.34#1)
    b. Subtype field set to 32. (6.3.34#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.34#2)
36. Device Handle field set to 0 (reserved). (6.3.34#4)
37. Verify that the HUT reports failure of the data transfer to the user. (5.2.1.2#8)

**Case 3 – Successful Bulk IN/OUT Transfer with Randomly Dropped Packets**

1. Disconnect then reconnect the Compliance Device to the HUT.
2. Enumerate and configure the Compliance Device.
3. Connect the USB Mass Storage Device to the Compliance Device.
4. Enumerate and Configure the USB Mass Storage Device.
5. Initiate a bulk IN transfer.
6. Instruct the Compliance Device to randomly drop TransferResp packets at a rate of approximately 1 packet per 100 sent out.
7. Verify that the transfer completes, the HUT does not generate any error messages, and the full amount of data expected is received.
8. Initiate a second bulk IN transfer.
9. Instruct the Compliance Device to randomly drop TransferResp packets at a rate of approximately 1 packet per 50 sent out.
10. Verify that the transfer completes, the HUT does not generate any error messages, and the full amount of data expected is received.
11. Initiate a third bulk IN transfer.
12. Instruct the Compliance Device to randomly drop TransferResp packets at a rate of approximately 1 packet per 10 sent out.
13. Verify that the transfer completes, the HUT does not generate any error messages, and the full amount of data expected is received.

**Case 4 – Multiple Pending Transfers**

1. If MA USB session is not Active, reconnect the Compliance Device to the HUT, then enumerate and configure.
2. Ensure that the following steps are captured by the MA USB Sniffer.
3. Initiate two interrupt IN transfers on a single endpoint.
4. Instruct the Compliance Device to send a TransferResp packet with the status set to 135 (TRANSFER_PENDING) for the first transfer, then wait at least aTransferTimeout before completing the first transfer.
5. Instruct the Compliance Device to not send any MA USB packets for the second transfer.
6. Verify that the HUT does not retry the second transfer until the first transfer completes. (5.4#19)
7. Verify that the HUT does not retry the second transfer until at least aTransferTimeout after it acknowledges the last TransferResp packet (with EoT bit set) of the first transfer. (5.4#19)

# TD 5.4 Ping Response Test

**Assertions Covered**

1. 5.2.2#3
2. 6.3.35#1, 6.3.35#2, 6.3.35#3, 6.3.35#4

**Required Equipment**

1. MA USB Compliance Device

**Test Steps**

1. If the Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the Compliance Device to the HUT.
    b. Enumerate and configure the Compliance Device.
2. Instruct the Compliance Device to issue a PingReq with the Device Address field set to the address of the Compliance Device and the SSID field set to the SSID value observed during enumeration.
3. Verify that the HUT responds with a PingResp packet. (6.3.35#1)
4. Verify that the PingResp packet has:
    a. Device Address field is set to the device address in the PingReq. (5.2.2#3)
    b. Type field set to 0. (6.3.35#2)
    c. Subtype field set to 33. (6.3.35#2)
    d. Device Handle field set to 0 (reserved). (6.3.35#3)
    e. Status Code field set to 0 (SUCCESS). (6.3.35#4)
5. Instruct the Compliance Device to issue a PingReq with the Device Address field set to the address of the Compliance Device and the SSID field set to a different value than the SSID value observed during enumeration.
6. Verify that the HUT does not respond.
7. Instruct the Compliance Device to issue a PingReq with the Device Address field set to 0xFF (broadcast) and the SSID field set to the SSID value observed during enumeration.
8. Verify that the HUT does not respond.
9. Instruct the Compliance Device to issue a PingReq with the device address field set to the address of the Compliance Device and the SSID field set to the SSID value observed during enumeration.
10. Verify that the HUT responds with a PingResp packet. (6.3.35#1)
11. Verify that the PingResp packet has:
    a. Device Address field is set to the device address in the PingReq. (5.2.2#3)
    b. Type field set to 0. (6.3.35#2)
    c. Subtype field set to 33. (6.3.35#2)
    d. Device Handle field set to 0 (reserved). (6.3.35#3)
    e. Status Code field set to 0 (SUCCESS). (6.3.35#4)
12. Repeat for 10 iterations.

# TD 5.5 Multiple Device Ping Test

**Assertions Covered**

1. 5.2.2#5, 5.2.2#6

**Required Equipment**

1. MA USB Compliance Device with hub functionality
2. KG USB Mass Storage Class (MSC) Devices (2)

**Test Steps**

1. If the Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the Compliance Device to the HUT.
    b. Enumerate and configure the Compliance Device.
2. Connect both KG USB MSC Devices to the MA USB Compliance Device.
3. Enumerate and configure the USB MSC Devices.
4. Simultaneously initiate a number of large bulk IN transfer for each of the USB Devices so that there are transfers pending on two bulk IN endpoints.
5. Before either transfer completes, instruct the Compliance Device to stop sending MA USB packets.
6. Wait for the HUT to finish retrying TransferReq packets and initiate ping protocol.
7. Verify that the HUT only sends one PingReq packet. (5.2.2#6)
    *Note: retries of the PingReq are permitted*
8. Wait for the HUT to stop retrying the PingReq packet.

9. Instruct the Compliance Device to issue a PingReq with the Device Address field set to the address of the Compliance Device and the SSID field set to the SSID value observed during enumeration.
10. Verify that the HUT does not send a PingResp packet to the Compliance Device. (5.2.2#5)

# TD 5.6 Data Packet Fields Test

**Assertions Covered**

1. 5.2.1.2#11, 5.4#4, 5.4#5, 5.5#6, 5.5#7, 5.5#13, 5.5#15, 5.5#32, 5.10.2#4, 5.10.3#5
2. 6.5.2#4, 6.5.2#5, 6.5.2#6, 6.5.2#7, 6.5.2#8, 6.5.4#1, 6.5.4#2, 6.5.4#4, 6.5.4#5

**Required Equipment**

1. MA USB Compliance Device
2. LS, FS, HS and SS Loopback-Capable USB Compliance Devices[1]

**Test Steps**

For each of the test cases outlined below, verify the following for all non-isochronous OUT transfers.

1. The HUT releases each successive data packet within aTransferRepeatTime of the previous data packet belonging to the same transfer. (5.2.1.2#11)
2. Request ID is incremented by 1 for each new transfer with no gaps in between, or is reset to 0 after aMaxRequestID is reached. (5.5#6, 5.5#7)
3. Sequence Number is incremented by 1 for each new[2] TransferReq packet with no gaps in between, or is reset to 0 after aMaxSequenceNumber is reached. (5.5#13, 5.5#15)
4. After receiving a TransferResp packet with the EoT field set to 1, the HUT responds with a TransferAck packet that has the same Request ID, Sequence Number and Status Code as the TransferResp packet. (5.5#32)
5. Verify that each OUT TransferReq packet has:
    a. Type field set to 2. (6.5.2#4)
    b. Subtype field set to 0 (6.5.2#4)
    c. Status code field set to 0 (SUCCESS). (6.5.2#5)
    d. EPS field set to 0 (reserved). (6.5.2#6)
    e. Remaining Size field with remaining number of bytes left to complete transfer. (6.5.2#7)
6. Verify that each OUT TransferAck packet has:
    a. Type field set to 2. (6.5.4#1)
    b. Subtype field set to 2 (6.5.4#1)
    c. Status code field set to one of the values in Table 6. (6.5.4#2)
    d. EPS field set to 0 (reserved). (6.5.4#4)
    e. Remaining Size field set to 0 (reserved). (6.5.4#5)

For each of the test cases outlined below, verify the following for all non-isochronous IN transfers.

1. Request ID is incremented by 1 for each new transfer with no gaps in between, or is reset to 0 after aMaxRequestID is reached. (5.4#4, 5.4#5)
2. Verify that each IN TransferReq packet has:
    a. No payload. (6.5.2#3)
    b. Type field set to 2. (6.5.2#4)
    c. Subtype field set to 0 (6.5.2#4)
    d. Status code field set to 0 (SUCCESS). (6.5.2#5)
    e. EPS field set to 0 (reserved). (6.5.2#6)
    f. Remaining Size field with remaining number of bytes the device can transfer. (6.5.2#8)
3. Verify that each IN TransferAck packet has:
    a. Type field set to 2. (6.5.4#1)
    b. Subtype field set to 2 (6.5.4#1)
    c. Status code field set to one of the values in Table 6. (6.5.4#2)

[1] Note: If USB Compliance Devices are not available, KG USB Devices with multiple endpoint types can also be used for testing.
[2] Note: retried packets are not considered “new” and can have repeat sequence numbers.

d. EPS field set to 0 (reserved). (6.5.4#4)
e. Remaining Size field set to 0 (reserved). (6.5.4#5)

For each of the test cases outlined below, verify the following for all isochronous IN transfers.

1. Request ID is incremented by 1 for each successive isochronous IN request. (5.10.2#4)

For each of the test cases outlined below, verify the following for all isochronous OUT transfers.

1. Request ID is incremented by 1 for each successive isochronous OUT request. (5.10.3#5)

**Case 1 – Control Loopback**

1. If the MA USB Hub is not connected to the HUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the MA USB Hub to the HUT.
    b. Enumerate and configure the MA USB Hub.
2. If the USB Compliance Device is not connected to the MA USB Hub or is connected but not operational:
    a. Connect/reconnect the USB Compliance Device downstream of the MA USB Hub.
    b. Enumerate and configure the USB Compliance Device for loopback transfer.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Initiate an OUT transfer to the control endpoint.
5. Initiate an IN transfer to the control endpoint.
6. Verify that the OUT data matches the IN data (data is the same and is in the same order).
7. Repeat for LS, FS, HS, and SS USB devices.

**Case 2 – Bulk Loopback**

1. If the USB Compliance Device is not operational, reconnect the USB Compliance Device behind the MA USB Hub, then enumerate.
2. Configure the Compliance Device for bulk endpoint loopback.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Initiate an OUT transfer to the bulk OUT endpoint.
5. Verify that the OUT transfer completes successfully.
6. Initiate an IN transfer to the bulk IN endpoint.
7. Verify that the IN transfer completes successfully.
8. Verify that the OUT data matches the IN data (data is the same and is in the same order).
9. Repeat for FS, HS, and SS USB devices.

**Case 3 – Interrupt Loopback**

1. If the USB Compliance Device is not operational, reconnect the USB Compliance Device behind the MA USB Hub, then enumerate.
2. Configure the Compliance Device for interrupt endpoint loopback.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Initiate an OUT transfer to the interrupt OUT endpoint.
5. Verify that the OUT transfer completes successfully.
6. Initiate an IN transfer to the interrupt IN endpoint.
7. Verify that the IN transfer completes successfully.
8. Verify that the OUT data matches the IN data (data is the same and is in the same order).
9. Repeat for LS, FS, HS, and SS USB devices.

**Case 4 – Isochronous OUT Loopback**

1. If the USB Compliance Device is not operational, reconnect the USB Compliance Device behind the MA USB Hub, then enumerate.
2. Configure the Compliance Device for isochronous endpoint OUT looped back to bulk IN.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Initiate an OUT transfer.
5. Verify that the OUT transfer completes successfully.

6. Initiate an IN transfer to the bulk IN endpoint.
7. Verify that the OUT transfer completes successfully.
8. Verify that the OUT data matches the IN data (data is the same and is in the same order).
9. Repeat for FS, HS, and SS USB devices.

**Case 5 – Isochronous IN Loopback**

1. If the USB Compliance Device is not operational, reconnect the USB Compliance Device behind the MA USB Hub, then enumerate.
2. Configure the Compliance Device for isochronous endpoint IN looped back to bulk IN.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Initiate an OUT transfer.
5. Verify that the OUT transfer completes successfully.
6. Initiate an IN transfer to the bulk IN endpoint.
7. Verify that the IN transfer completes successfully.
8. Verify that the OUT data matches the IN data (data is the same and is in the same order).
9. Repeat for FS, HS, and SS USB devices.

# TD 5.7 Endpoint Configuration Event Test

**Assertions Covered**

1. 5.4#3, 5.4#8, 5.5#5, 5.5#12, 5.10.2#3, 5.10.3#4

**Required Equipment**

1. MA USB Compliance Device with hub functionality
2. HS Loopback-Capable USB Compliance Device

**Test Steps**

**Case 1 – Bulk/Interrupt Endpoints**

1. If the MA USB Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the MA USB Compliance Device to the HUT.
    b. Enumerate and configure the MA USB Compliance Device.
2. If the USB Compliance Device is not connected to the MA USB Compliance Device or is connected but not operational:
    a. Connect/reconnect the USB Compliance Device downstream of the MA USB Compliance Device.
    b. Enumerate and configure the USB Compliance Device for loopback transfer.
3. From the HUT system, initiate five separate loopback transfers on the Compliance Device.
4. For each loopback, verify that both the IN and OUT transfers complete successfully.
5. Initiate another loopback transfer.
6. Instruct the USB Compliance Device to return STALL status for the OUT endpoint. The MA USB Compliance Device should subsequently send the HUT a TransferResp with the Status Code set to 136 (TRANSFER_EP_STALL), the Request ID field set to the Request ID of the last received TransferReq packet, the Sequence Number field set to the Sequence Number of the last received TransferReq packet, and the EoT bit set to 1.
7. Ensure that the driver for the USB Compliance Device clears the halted endpoint.
8. Initiate another loopback transfer.
9. For the OUT transfer:
    a. Verify that the first TransferReq packet sent by the HUT has the Request ID field set to 0. (5.5#5)
    b. Verify that the first TransferReq packet sent by the HUT has the Sequence Number field set to 0. (5.5#12)
10. Instruct the USB Compliance Device to return STALL status for the IN endpoint. The MA USB Compliance Device should subsequently send the HUT a TransferResp with the Status Code set to 136 (TRANSFER_EP_STALL) and the EoT bit set to 1.
11. Ensure that the driver for the USB Compliance Device clears the halted endpoint.
12. Initiate another loopback transfer.
13. For the IN transfer:
    a. Verify that the first TransferReq packet sent by the HUT has the Request ID field set to 0. (5.4#3)
    b. Verify that the first TransferReq packet sent by the HUT has the Sequence Number field set to 0. (5.4#8)

**Repetitions**

- Repeat for bulk and interrupt endpoints

**Case 2 – Isochronous Endpoints**

1. Configure the USB Compliance Device for isochronous loopback.
2. Initiate an isochronous loopback transfer.
3. Verify that the first OUT TransferReq packet sent by the HUT has the Request ID field set to 0. (5.10.3#4)
4. Verify that the first IN TransferReq packet sent by the HUT has the Request ID field set to 0. (5.10.2#3)

# TD 5.8 Bulk IN Request ID Test

**Assertions Covered**

1. 5.2.3#8

**Required Equipment**

1. MA USB Compliance Device with hub functionality
2. FS, HS and SS USB Loopback-Capable Compliance Devices

**Test Steps**

1. If the MA USB Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the MA USB Compliance Device to the HUT.
    b. Enumerate and configure the MA USB Compliance Device.
2. If the USB Compliance Device is not connected to the MA USB Compliance Device or is connected but not operational:
    c. Connect/reconnect the USB Compliance Device downstream of the MA USB Compliance Device.
    d. Enumerate and configure the USB Compliance Device for loopback transfer.
3. From the HUT system, initiate a loopback transfer.
4. When handling the IN transfer for loopback, instruct the Compliance Device to break the transfer into multiple IN TransferResp packets. Instruct the Compliance Device to inject into the middle of the transfer a TransferResp packet with a Request ID that is equal to the Request ID of the transfer plus 10 and carries data that is not part of the loopback transfer.
5. Verify that both IN and OUT transfers complete successfully.
6. Verify that the OUT data matches the IN data (data is the same and is in the same order). (5.2.3#8)

**Repetitions**

- Repeat with FS, HS, and SS USB Compliance Devices

# TD 5.9 Bulk IN/OUT Request ID Status Handling Test

**Assertions Covered**

1. 5.4#27, 5.5#24

**Required Equipment**

1. MA USB Compliance Device
2. Loopback-Capable USB Compliance Device

**Test Steps**

**Case 1 – Bulk IN transfers**

1. If the MA USB Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the MA USB Compliance Device to the HUT.
    b. Enumerate and configure the MA USB Compliance Device.
2. If the loopback-capable USB Compliance Device is not connected to the HUT or is not operational:
    a. Connect/reconnect the USB Compliance Device downstream of the MA USB Compliance Device.
    b. Enumerate and configure the USB Compliance Device for loopback.
3. Initiate three large bulk IN data transfers in close succession (so that the second and third transfers are initiated before first has completed).

4. Instruct the Compliance Device to respond to the first TransferReq with a TransferResp packet that has the Status Code set to 156 (MISSING_REQUEST_ID) and the Request ID field set to a value that is 2 larger than the HUT expects.
5. Verify that the HUT does not retry the second or third transfers.
6. Verify that the HUT sends a new TransferReq packet with the same Request ID as the TransferResp packet from the MA USB Compliance Device that had the 156 (MISSING_REQUEST_ID) status code. (5.4#27)
7. Verify that all three transfers complete successfully.
8. Initiate three large bulk IN data transfers in close succession (so that the second and third transfers are initiated before first has completed).
9. Instruct the Compliance Device to respond to the first TransferReq with a TransferResp packet that has the Status Code set to 133 (INVALID_REQUEST) and the Request ID field set to a value that is 2 smaller than the HUT expects.
10. Verify that the HUT does not retry the second or third transfers.
11. Verify that the HUT sends a new TransferReq packet with the same Request ID as the TransferResp packet from the MA USB Compliance Device that had the 133 (INVALID_REQUEST) status code. (5.4#27)
12. Verify that all three transfers complete successfully.
13. Initiate three large bulk IN data transfers in close succession (so that the second and third transfers are initiated before first has completed).
14. Instruct the Compliance Device to respond to the second TransferReq with a TransferResp packet that has the Status Code set to 156 (MISSING_REQUEST_ID) and the Request ID field set to a value that is 2 larger than the HUT expects.
15. Verify that the HUT does not retry the third transfer.
16. Verify that the HUT sends a new TransferReq packet with the same Request ID as the TransferResp packet from the MA USB Compliance Device that had the 156 (MISSING_REQUEST_ID) status code. (5.4#27)
17. Verify that all three transfers complete successfully.
18. Initiate three large bulk IN data transfers in close succession (so that the second and third transfers are initiated before first has completed).
19. Instruct the Compliance Device to respond to the second TransferReq with a TransferResp packet that has the Status Code set to 133 (INVALID_REQUEST) and the Request ID field set to a value that is 2 smaller than the HUT expects.
20. Verify that the HUT does not retry the third transfer.
21. Verify that the HUT sends a new TransferReq packet with the same Request ID as the TransferResp packet from the MA USB Compliance Device that had the 133 (INVALID_REQUEST) status code. (5.4#27)
22. Verify that all three transfers complete successfully.

**Case 2 – Bulk OUT transfers**

1. Initiate three large bulk OUT data transfers in close succession (so that the second and third transfers are initiated before first has completed).
2. Instruct the Compliance Device to respond to the first TransferReq with a TransferResp packet that has the Status Code set to 156 (MISSING_REQUEST_ID) and the Request ID field set to a value that is 2 larger than the HUT expects.
3. Verify that the HUT does not retry the second or third transfers.
4. Verify that the HUT sends a new TransferReq packet with the same Request ID and Sequence Number as the TransferResp packet from the MA USB Compliance Device that had the 156 (MISSING_REQUEST_ID) status code. (5.5#24)
5. Verify that all three transfers complete successfully.
6. Initiate three large bulk OUT data transfers in close succession (so that the second and third transfers are initiated before first has completed).
7. Instruct the Compliance Device to respond to the first TransferReq with a TransferResp packet that has the Status Code set to 133 (INVALID_REQUEST) and the Request ID field set to a value that is 2 smaller than the HUT expects.
8. Verify that the HUT does not retry the second or third transfers.
9. Verify that the HUT sends a new TransferReq packet with the same Request ID and Sequence Number as the TransferResp packet from the MA USB Compliance Device that had the 133 (INVALID_REQUEST) status code. (5.5#24)
10. Verify that all three transfers complete successfully.
11. Initiate three large bulk OUT data transfers in close succession (so that the second and third transfers are initiated before first has completed).
12. Instruct the Compliance Device to respond to the second TransferReq with a TransferResp packet that has the Status Code set to 156 (MISSING_REQUEST_ID) and the Request ID field set to a value that is 2 larger than the HUT expects.

13. Verify that the HUT does not retry the third transfer.
14. Verify that the HUT sends a new TransferReq packet with the same Request ID and Sequence Number as the TransferResp packet from the MA USB Compliance Device that had the 156 (MISSING_REQUEST_ID) status code. (5.5#24)
15. Verify that all three transfers complete successfully.
16. Initiate three large bulk OUT data transfers in close succession (so that the second and third transfers are initiated before first has completed).
17. Instruct the Compliance Device to respond to the second TransferReq with a TransferResp packet that has the Status Code set to 133 (INVALID_REQUEST) and the Request ID field set to a value that is 2 smaller than the HUT expects.
18. Verify that the HUT does not retry the third transfer.
19. Verify that the HUT sends a new TransferReq packet with the same Request ID and Sequence Number as the TransferResp packet from the MA USB Compliance Device that had the 133 (INVALID_REQUEST) status code. (5.5#24)
20. Verify that all three transfers complete successfully.

# TD 5.10 Bulk IN Transfer Keep Alive Test

**Assertions Covered**

1. 5.4#30

**Required Equipment**

1. MA USB Compliance Device
2. Loopback-Capable USB Compliance Device

**Test Steps**

1. If the MA USB Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the MA USB Compliance Device to the HUT.
    b. Enumerate and configure the MA USB Compliance Device.
2. If the loopback-capable USB Compliance Device is not connected to the HUT or is not operational:
    a. Connect/reconnect the USB Compliance Device downstream of the MA USB Compliance Device.
    b. Enumerate and configure the USB Compliance Device for loopback.
3. Initiate a large bulk IN transfer such that multiple TransferResp packets are required to complete the transfer.
4. Instruct the Compliance Device to send a first TransferResp packet, but not any subsequent TransferResp packets.
5. Verify that the HUT sends a TransferReq packet to the Compliance Device after aTransferKeepAlive time has passed since receiving the first TransferResp packet. (5.4#30)
6. Verify that the TransferReq packet has:
    a. Request ID set to the same value as the first TransferReq packet in the transfer. (5.4#30)
    b. Sequence Number set to the value to host is expecting next. (5.4#30)
    c. Remaining Size field set a value equal to the Remaining Size field in the first TransferReq packet minus the payload size of the first TransferResp packet (i.e. the number of bytes expected to complete the transfer). (5.4#30)
7. Verify that the transfer completes successfully.
8. Repeat for 10 iterations.

# TD 5.11 Bulk IN Transfer ARQ Test

**Assertions Covered**

1. 5.4#39, 5.4#40

**Required Equipment**

1. MA USB Compliance Device
2. Loopback-Capable USB Compliance Device

**Test Steps**

1. If the MA USB Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the MA USB Compliance Device to the HUT.
    b. Enumerate and configure the MA USB Compliance Device.

2. If the loopback-capable USB Compliance Device is not connected to the HUT or is not operational:
    a. Connect/reconnect the USB Compliance Device downstream of the MA USB Compliance Device.
    b. Enumerate and configure the USB Compliance Device for loopback.

**Case 1 – Successful Transfer**

1. Initiate a bulk IN transfer.
2. For odd numbered transfers, instruct the Compliance Device to set the ARQ flag to 1 and the Status Code to 0 (SUCCESS) in the last TransferResp packet belonging to the Transfer (i.e. TransferResp packet with EoT = 1).
3. For even numbered transfers, instruct the Compliance Device to set the ARQ flag to 0 and the Status Code to 0 (SUCCESS) in the last TransferResp packet belonging to the Transfer (i.e. TransferResp packet with EoT = 1).
4. Verify that the HUT responds with either a TransferAck packet or a TransferReq packet for a new transfer. (5.4#39)
5. If the HUT sent a TransferAck packet, verify that the TransferAck has the same Request ID, Sequence Number and Status Code as the TransferResp packet being acknowledged. (5.4#40)
6. Repeat the above steps for 4 iterations.

*NOTE: a Status Code of 141 (TRANSFER_SHORT_TRANSFER) can be used instead of 0 (SUCCESS) for a transfer that ends with a short packet.*

**Case 2 – Unsuccessful Transfer**

1. Initiate a bulk IN transfer.
2. Instruct the Compliance Device to set the ARQ flag to 1 and the Status Code to 128 (UNSUCCESSFUL) in the last TransferResp packet belonging to the Transfer (i.e. TransferResp packet with EoT = 1).
3. Verify that the HUT responds with a TransferAck packet. (5.4#39)
4. Verify that the TransferAck has the same Request ID, Sequence Number and Status Code as the TransferResp packet being acknowledged. (5.4#40)
5. Initiate a bulk IN transfer.
6. Instruct the Compliance Device to set the ARQ flag to 0 and the Status Code to 128 (UNSUCCESSFUL) in the last TransferResp packet belonging to the Transfer (i.e. TransferResp packet with EoT = 1).
7. Verify that the HUT responds with a TransferAck packet. (5.4#39)
8. Verify that the TransferAck has the same Request ID, Sequence Number and Status Code as the TransferResp packet being acknowledged. (5.4#40)

# TD 5.12 Bulk IN Pending Transfer Test

**Assertions Covered**

1. 5.4#34, 5.4#36, 5.4#37
2. 6.3.52#5, 6.3.53#1, 6.3.53#2, 6.3.53#4, 6.3.53#5, 6.3.53#7, 6.3.53#8, 6.3.53#9

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. MA USB Compliance Device
4. Loopback-Capable USB Compliance Device

**Test Steps**

**Case 1 – successful transfer completion**

1. If the MA USB Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the MA USB Compliance Device to the HUT.
    b. Enumerate and configure the MA USB Compliance Device.
2. If the loopback-capable USB Compliance Device is not connected to the HUT or is not operational:
    a. Connect/reconnect the USB Compliance Device downstream of the MA USB Compliance Device.
    b. Enumerate and configure the USB Compliance Device for loopback.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Instruct the MA USB Compliance Device to send the HUT an EPSetKeepAliveReq packet with the Keep-Alive Duration (K) field set to 2.

5. Verify that the HUT Responds with an EPSetKeepAliveResp packet. (6.3.53#1)
6. Verify that the EPSetKeepAliveResp has:
    a. Type field set to 0. (6.3.53#2)
    b. Subtype field set to 51. (6.3.53#2)
    c. Status Code Field that indicates whether or not request was successfully completed. (6.3.53#4)
    d. Old Keep-Alive Duration field equal to the Keep-Alive Duration field in the EPSetKeepAliveReq packet received prior to the corresponding one. (6.3.53#5)
    e. Bits 24 through 31 set to 0 (reserved). (6.3.53#7)
    f. Stream ID field set to 0 (reserved) if endpoint in corresponding EPSetKeepAliveReq packet is not an Enhanced SuperSpeed bulk endpoint that supports Enhanced SuperSpeed Stream Protocol. (6.3.53#8)
    g. EPHandle set to the endpoint handle of the endpoint to which the response is related (6.3.53#9)
7. Initiate a large Bulk IN transfer such that multiple TransferResp packets are required to complete the transfer.
8. Instruct the MA USB Compliance Device to send a TransferResp with the Status Code set to 135 (TRANSFER_PENDING) and the ARQ flag set to 1 in response to the TransferReq initiating the transfer.
9. Verify that the HUT sends a TransferAck packet in response. (5.4#34)
10. Verify that the TransferAck packet has the same Request ID and Status Code field values as the TransferResp packet with the TRANSFER_PENDING status. (5.4#34)
11. Instruct the Compliance Device to wait until K * aTransferKeepAlive has passed, where K equals the Keep-alive Duration from the previously sent EPSetKeepAliveReq packet.
12. Verify that the HUT retries the TransferReq packet. (5.4#36)
13. Instruct the MA USB Compliance Device to respond to the retried TransferReq with a TransferResp that has the Status Code set to 0 (SUCCESS).
14. Instruct the MA USB Compliance Device to wait until after aTransferKeepAlive has passed.
15. Verify that the HUT sends a TransferReq packet. (5.4#37)
16. Instruct the Compliance Device to send the remaining TransferResp packets.
17. Verify that transfer completes successfully.

**Repetitions**

- Repeat for Keep-Alive Duration values of 3, 4, 5, and 10.

**Case 2 – transfer failure while pending**

1. Initiate a large bulk IN transfer such that multiple TransferResp packets are required to complete the transfer.
2. Instruct the MA USB Compliance Device not to respond to the first TransferReq packet and first TransferReq retry.
3. Wait for the HUT to retry the TransferReq packet a second time.
4. Instruct the MA USB Compliance Device to send a TransferResp with the Status Code set to 135 (TRANSFER_PENDING) in response to the second TransferReq retry.
5. Instruct the MA USB Compliance Device to stop responding.
6. Verify that the HUT does not retry the TransferReq packet before K * aTransferKeepAlive has passed. (5.4#36)
7. Verify that the HUT retries the TransferReq aBulkTransferRetries times. (5.4#36)
8. Disconnect the MA USB Compliance Device.

**Case 3 – Infinite Keep-Alive**

1. Reconnect the MA USB Compliance Device to the HUT, then enumerate and configure.
2. Enumerate and configure the USB Compliance Device behind the MA USB Compliance Device.
3. Instruct the MA USB Compliance device to send the HUT an EPSetKeepAliveReq packet with the Keep-Alive Duration (K) field set to 0.
4. Verify that the HUT responds with an EPSetKeepAliveResp packet. (6.3.53#1)
5. Verify that the EPSetKeepAliveResp has:
    a. Type field set to 0. (6.3.53#2)
    b. Subtype field set to 51. (6.3.53#2)
    c. Status Code Field that indicates whether or not request was successfully completed. (6.3.53#4)
    d. Old Keep-Alive Duration field equal to the Keep-Alive Duration field in the EPSetKeepAliveReq packet received prior to the corresponding one. (6.3.53#5)

   e. Bits 24 through 31 set to 0 (reserved). (6.3.53#7)
   f. Stream ID field set to 0 (reserved) if endpoint in corresponding EPSetKeepAliveReq packet is not an Enhanced SuperSpeed bulk endpoint that supports Enhanced SuperSpeed Stream Protocol. (6.3.53#8)
   g. EPHandle set to the endpoint handle of the endpoint to which the response is related (6.3.53#9)
6. Initiate a large bulk IN transfer such that multiple TransferResp packets are required to complete the transfer.
7. Instruct the MA USB Compliance Device to send a TransferResp with the Status Code set to 135 (TRANSFER_PENDING) in response to the TransferReq packet initiating the transfer.
8. Instruct the MA USB Compliance Device to stop sending or responding to MA USB packets.
9. For the next minute, verify that the HUT does not retry the TransferReq packet. (6.3.52#5)

# TD 5.13 Bulk IN/OUT Stall Test

**Assertions Covered**

1. 5.4#45, 5.4#46, 5.5#36

**Required Equipment**

1. MA USB Compliance Device
2. Loopback-Capable USB Compliance Device

**Test Steps**

**Case 1 – Bulk IN Transfer**

1. If the MA USB Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
   a. Connect/reconnect the MA USB Compliance Device to the HUT.
   b. Enumerate and configure the MA USB Compliance Device.
2. If the loopback-capable USB Compliance Device is not connected to the HUT or is not operational:
   a. Connect/reconnect the USB Compliance Device downstream of the MA USB Compliance Device.
   b. Enumerate and configure the USB Compliance Device for loopback.
3. Initiate a bulk IN transfer.
4. Instruct the Compliance Device to respond to the first TransferReq packet with a TransferResp packet with the Status Code field set to 136 (TRANSFER_EP_STALL) and EoT field set to 1.
5. Verify that the HUT responds with a TransferAck packet. (5.4#45)
6. Verify that the TransferAck packet has the same Request ID, Sequence Number, and Status Code fields as the TransferResp packet from the Compliance Device. (5.4#46)

**Case 2 – Bulk OUT Transfer**

1. Initiate a bulk OUT transfer.
2. Instruct the Compliance Device to respond to the first TransferReq packet with a TransferResp packet with Status Code field set to 136 (TRANSFER_EP_STALL) and the EoT bit set to 1.
3. Verify that the HUT sends a TransferAck with the same Request ID and Sequence Number as the TransferResp from the Compliance Device. (5.5#36)
4. Verify that the HUT does not send any more TransferReq packets. (5.5#36)

# TD 5.14 Bulk IN Missing Sequence Number Test

**Assertions Covered**

1. 5.4#49, 5.4#50

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. MA USB Compliance Device with hub functionality
4. Loopback-Capable USB Compliance Device

**Test Steps**

1. If the MA USB Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:

   a. Connect/reconnect the MA USB Compliance Device to the HUT.
   b. Enumerate and configure the MA USB Compliance Device.
2. If the USB Compliance Device is not connected to the MA USB Compliance Device or is connected but not operational:
   a. Connect/reconnect the USB Compliance Device downstream of the MA USB Compliance Device.
   b. Enumerate and configure the USB Compliance Device for loopback transfer.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. From the HUT system, initiate loopback on the Compliance Device.
5. Instruct the Compliance Device to send a TransferResp packet for the IN transfer with a Sequence Number that is 1 larger than expected (i.e. skip a sequence number).
6. Verify that the HUT sends a new TransferReq packet with:
   a. The Request ID set to the same Request ID as the transfer with the skipped Sequence Number. (5.4#49)
   b. The Sequence Number field set to the value of the skipped Sequence Number. (5.4#49)
   c. The Remaining Size field set to the number of bytes remaining in the transfer. (5.4#49)
   d. The Status Code field set to 0 (NO_ERROR). (5.4#49)
7. Verify that the Compliance Device receives the new TransferReq packet within aTransferResponseTime + 2 * aDataChannelDelay of sending the TransferResp packet with the out-of-order sequence number. (5.4#50)
8. Verify that loopback completes successfully.
9. Repeat for 10 iterations.

# TD 5.15 Isochronous OUT DWORD Alignment Test

**Assertions Covered**

1. 5.10.1.1#3
2. 5.10.1.1#4

**Required Equipment**

1. MA USB Compliance Device with hub functionality
2. Loopback-Capable USB Compliance Device

**Test Steps**

**Case 1 – DWORD Alignment**

1. Instruct the MA USB Compliance Device to use an Isochronous Payload Alignment value of 1 in the CapResp packet.
2. Connect the MA USB Compliance Device to the HUT.
3. Enumerate and configure the MA USB Compliance Device.
4. Connect the USB Compliance Device downstream of the MA USB Compliance Device.
5. Enumerate and configure the USB Compliance Device for isochronous loopback.
6. For 10 iterations, repeat the following:
   a. From the HUT system, initiate loopback on the Compliance Device.
   b. Verify that all MA USB packets from the HUT have DWORD padding (Block Length is a multiple of 4). (5.10.1.1#3)
   c. Verify that loopback completes successfully.
7. Disconnect the USB and MA USB Compliance Devices.
8. Repeat with 1, 2, 3, and 4 byte packets

**Case 2 – No DWORD Alignment**

1. Instruct the MA USB Compliance Device to use an Isochronous Payload Alignment value of 0 in the CapResp packet.
2. Connect the MA USB Compliance Device to the HUT.
3. Enumerate and configure the MA USB Compliance Device.
4. Connect the USB Compliance Device downstream of the MA USB Compliance Device.
5. Enumerate and configure the USB Compliance Device for isochronous loopback.
6. For 10 iterations, repeat the following:
   a. From the HUT system, initiate loopback on the Compliance Device.
   b. Verify that no MA USB packets from the HUT have DWORD padding. (5.10.1.1#4)

    c. Verify that loopback completes successfully.
7. Disconnect the USB and MA USB Compliance Devices.
8. Repeat with 1, 2, 3, and 4 byte packets.

# TD 5.16 Protocol Version Compatibility Test

**Assertions Covered**

1. None

**Required Equipment**

1. MA USB Compliance Device

**Test Steps**

1. Instruct the Compliance Device to set the Version field to 0001b in all MA USB packets sent.
2. Connect the MA USB Compliance Device to the HUT.
3. Enumerate and configure the Compliance Device.
4. Verify that the Compliance Device shows up in the USB tree of the HUT system without any warnings or error flags.
5. Disconnect the Compliance Device.

**Repetitions**

- Repeat with Version field set to 1000b, 0110b, 0100b, and 0010b.

# TD 5.17 Invalid Host Bit Test

**Assertions Covered**

1. 6.2.1.2#1

**Required Equipment**

1. MA USB Compliance Device
2. Loopback-Capable USB Compliance Device

**Test Steps**

**Case 1 – Enumeration**

1. Connect the Compliance Device to the HUT.
2. Start enumeration.
3. Instruct the Compliance Device to respond to the CapReq packet from the HUT with a CapResp packet that has the Host field set to 1.
4. Verify that the HUT ignores the CapResp packet and retries the CapReq packet. (6.2.1.2#1)
5. Instruct the Compliance Device to finish enumeration and configuration with the Host field set to 0.
6. Verify that the Compliance Device shows up in the USB tree of the HUT.
7. Disconnect then reconnect the Compliance Device to the HUT.
8. Start enumeration.
9. Instruct the Compliance Device to respond to the SetUSBDevAddrReq packet with a SetUSBDevAddrResp packet that has the Host field set to 1.
10. Verify that the HUT ignores the SetUSBDevAddrResp packet and retries the SetUSBDevAddrReq packet. (6.2.1.2#1)
11. Instruct the Compliance Device to finish enumeration and configuration with the Host field set to 0.
12. Verify that the Compliance Device shows up in the USB tree of the HUT.

**Case 2 – Loopback Transfer**

1. Connect the USB Compliance Device downstream of the MA USB Compliance Device.
2. Enumerate and configure the USB Compliance Device for loopback transfer.
3. Initiate loopback.
4. Instruct the Compliance Device to send a TransferResp packet with Host field set to 1.
5. Verify that the HUT ignores the TransferResp packet and resends the TransferReq packet. (6.2.1.2#1)
6. Instruct Compliance Device to finish the transfer with the Host field set to 0.

7. Verify that the transfer completes successfully.
8. Repeat for 10 iterations where in 5 iterations the TransferResp packet with the Host field set to 1 is for an OUT transfer and in 5 iterations the TransferResp is for an IN transfer.

# TD 5.18 Enumeration and Device Initialization Test

**Assertions Covered**

1. 6.3.2#1, 6.3.2#2, 6.3.2#3, 6.3.2#5, 6.3.2#6, 6.3.2#7, 6.3.2#8, 6.3.4#1, 6.3.4#2, 6.3.4#3, 6.3.4#5, 6.3.4#6, 6.3.4#7, 6.3.4#8, 6.3.4#9, 6.3.4#10, 6.3.4#11, 6.3.4#15, 6.3.6#1, 6.3.6#2, 6.3.6#4, 6.3.6#5, 6.3.6#6, 6.3.6#7, 6.3.6#8, 6.3.6#9, 6.3.18#1, 6.3.18#2, 6.3.18#3, 6.3.20#1, 6.3.20#2, 6.3.22#1, 6.3.22#2, 6.3.22#4, 6.3.22#6, 6.3.24#1, 6.3.24#2, 6.3.24#5, 6.3.24#6, 6.3.24#7, 6.3.24#8, 6.3.24#11, 6.3.24#14, 6.3.24#15, 6.3.24#16
2. 7.3.2.2#4, 7.3.2.2#5, 7.3.2.3#1, 7.3.2.4#1
3. 8.1.2.1#1, 8.1.2.1#2, 8.1.2.2#1

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. KG LS, FS, HS and SS MA USB Devices
4. KG HS and SS MA USB Hubs

*Note: if KG MA USB devices are not available, Compliance Device with a KG USB device downstream can be substituted. When USB Device is being enumerated instead of MA USB device, steps 4 through 9 are omitted.*

**Test Steps**

1. Ensure that the following steps are captured by the MA USB Sniffer.
2. Connect an MA USB Device to the HUT.
3. Start enumeration.
4. Verify that the first management packet sent is a DevResetReq packet. (8.1.2.1#1)
5. Verify that the DevResetReq packet has:
    a. Type field set to 0. (6.3.18#1)
    b. Subtype field set to 16. (6.3.18#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.18#2)
    d. Device Handle field set to 0 (reserved). (6.3.18#3)
    e. Dialog Token field set to 0 (reserved). (6.3.18#3)
    f. MA USB Device Address field set to a non-zero value. (8.1.2.1#2)
6. Verify that second management request packet is a CapReq packet. (8.1.2.2#1)
7. Verify that CapReq packet has:
    a. Type field set to 0. (6.3.2#1)
    b. Subtype field set to 0. (6.3.2#2)
    c. Device Handle field set to 0 (reserved). (6.3.2#3)
    d. Bits 12 through 31 of DWORD 3 are set to 0 (reserved). (6.3.2#6)
8. Verify that any MA Host Capability Descriptors have:
    a. Length field is equal to the length of the descriptor in bytes. (6.3.2#7)
    b. MA Host Capability Type field is set to a descriptor type listed in Table 9. (6.3.2#8)
9. If the Number of Outstanding Management Requests fields in a CapReq is less than 127, generate a warning. (6.3.2#5)
10. Verify that a USBDevHandleReq packet is sent.
11. Verify that the USBDevHandleReq packet has:
    a. Type field set to 0. (6.3.4#1)
    b. Subtype field set to 2. (6.3.4#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.4#2)
    d. Device Handle field set to 0 (reserved). (6.3.4#3)
    e. Speed field set to a value that identified the speed of the USB device and is not greater than 4. (6.3.4#5, 6.3.4#6)
    f. Bits 24 through 31 of DWORD 3 set to 0 (reserved). (6.3.4#7)
    g. For an integrated USB device, the Hub field is reserved. (6.3.4#8)
    h. Bits 16 through 31 of DWORD 4 are set to 0 (reserved). (6.3.4#9)

i. For HS and SS devices, the Parent HS Hub field is set to 0 (reserved). (6.3.4#10)
j. For HS and SS device, the Parent HS Hub Port field is set to 0 (reserved). (6.3.4#11)
k. For LS, FS, and HS devices, the Lane Speed Exponent field is set to 0 (reserved). (6.3.4#13)
l. Bits 23 through 31 of DWORD 5 are set to 0 (reserved). (6.3.4#14)

12. Verify that the first EPHandleReq packet contains a single EP descriptor. (7.3.2.2#4)
13. Verify that the EP descriptor has the following fields (7.3.2.2#5)
    a. bLength = 7
    b. bDescriptorType = 5
    c. bEndpointAddress = 0
    d. bmAttributes = 0
    e. wMaxPacketSize = 8 (for LS Devices) 8, 16, 32, or 64 (for FS Devices), 64 (for HS Devices) or 512 (for SS Devices)
14. Verify that the EPHandleReq packet has:
    a. Type field set to 0. (6.3.6#1)
    b. Subtype field set to 40. (6.3.6#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.6#2)
    d. Number of EP Descriptors field set to number of EP descriptors in the packet. (6.3.6#4)
    e. Size of EP Descriptor field set to the size of each EP descriptor in the corresponding EPHandleReq packet. (6.3.6#5)
    f. Bits 11 through 31 set to 0 (reserved). (6.3.6#6)
    g. For an LS, FS, or HS device, the EP descriptor is 8 bytes long including 1 byte of zero padding. (6.3.6#7)
    h. For an enhanced SS device, the EP descriptor is 16 bytes long including 3 bytes of zero padding. (6.3.6#8)
    i. For an enhanced SS device operating at above Gen1 speed, the EP Descriptor is 24 bytes long including 3 bytes of zero padding. (6.3.6#9)
15. Verify that the HUT sends a SetUSBDevAddrReq packet. (7.3.2.4#1)
16. Verify that the SetUSBDevAddrReq packet has:
    a. Type field set to 0. (6.3.22#1)
    b. Subtype field set to 20. (6.3.22#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.22#2)
    d. Non-zero value in the Response Timeout field. (6.3.22#4)
    e. Bits 16 through 31 of DWORD 3 are set to 0 (reserved). (6.3.22#6)
17. Verify that the HUT sends a ModifyEP0Req packet after receiving a SetUSBDevAddrResp packet. (7.3.2.3#2)
18. Verify that ModifyEP0Req packet has the Device Handle field set to the handle of the target USB device, the EP0 Handle field set to the EP0 handle of the target USB device, and the Max Packet field set to a value greater than zero. (7.3.2.3#1)
19. Verify that the ModifyEP0Req packet has:
    a. Type field set to 0. (6.3.20#1)
    b. Subtype field set to 18. (6.3.20#1)
    c. Status Code field set to 0. (SUCCESS). (6.3.20#2)
20. Verify that, if enumerating a hub, an UpdateDevReq packet is sent. *Note: a HUT may send an UpdateDevReq packet when enumerating a device other than a hub, but it is not required.*
21. If the HUT sent an UpdateDevReq packet, verify that the UpdateDevReq packet has:
    a. Type field set to 0. (6.3.24#1)
    b. Subtype field set to 22. (6.3.24#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.24#2)
    d. Max Exit Latency set to 0 if USB device is not a hub. (6.3.24#5)
    e. Hub field set to 1 if the USB device is a hub or set to 0 if the USB device is not a hub. (6.3.24#6)
    f. Number of Ports field set to the number of downstream-facing ports if the device is a hub. (6.3.24#7)
    g. Number of Ports field is 0 if the USB device is not a hub. (6.3.24#8)
    h. Translator Think Time field is set to 0 (reserved) if the USB device is not a hub. (6.3.24#11)
    i. Integrated Hub Latency field is set to 0 (reserved) if the USB device is not a hub. (6.3.24#14)
    j. Bits 25 through 31 of DWORD 3 are set to 0 (reserved). (6.3.24#15)
    k. A device descriptor starting at bit 0 of DWORD 4. (6.3.24#16)

l. The device descriptor contains a valid standard device descriptor as defined in USB 2.0 or USB 3.1. (6.3.24#17)
m. Bits 16 through 31 of DWORD 8 are set to 0 (reserved). (6.3.24#18)

22. Disconnect the MA USB Device.

Repetitions:

- Repeat with LS, FS, HS, SS MA USB Devices.
- Repeat with HS and SS MA USB Hubs.

## TD 5.18.1 Synchronization Capabilities Descriptor Test

*Run this test in conjunction with TD 5.18 if:*

- *The HUT is a Wi-Fi host that supports 2.4/5 GHz; or*
- *The HUT is not a Wi-Fi host that supports 2.4/5 GHz but a Synchronization Capabilities descriptor is present in the CapReq packet from the HUT*

**Assertions Covered**

1. 6.3.2.1#1, 6.3.2.1#2, 6.3.2.1#3, 6.3.2.1#4, 6.3.2.1#5

**Test Steps**

1. Verify that a Synchronization Capabilities Descriptor is present. (6.3.2.1#1)
2. Verify that the Synchronization Capabilities Descriptor has:
    a. A length of 3 bytes. (6.3.2.1#2)
    b. Length field set to 3. (6.3.2.1#2)
    c. MA Host Capability Type field set to 3. (6.3.2.1#3)
    d. Media Time field set to 1 if the HUT has access to media time and/or Wi-Fi host supports 2.4/5 GHz. (6.3.2.1#4)
    e. Bits n+17 through n+23 of DWORD N are set to 0 (reserved). (6.3.2.1#5)

## TD 5.18.2 Link Sleep Capabilities Descriptor Test

*Run this test in conjunction with TD 5.18 if:*

- *The HUT supports Link Sleep; or*
- *The HUT does not support Link Sleep but a Link Sleep Capabilities descriptor is present*

**Assertions Covered**

1. 6.3.2.2#1, 6.3.2.2#2, 6.3.2.2#3, 6.3.2.2#4, 6.3.2.2#5

**Test Steps**

1. Verify that a Link Sleep Capability Descriptor is present if the HUT supports Link Sleep. (6.3.2.2#1)
2. Verify that the Link Sleep Capability Descriptor has:
    a. A length of 3 bytes. (6.3.2.2#2)
    b. Length field set to 3. (6.3.2.2#2)
    c. MA Host Capability Type field set to 5. (6.3.2.2#3)
    d. Link Sleep Capable field set to 1 if host can receive a SleepReq packet from a device without USB suspend. (6.3.2.2#4)
    e. Link Sleep Capable field set to 0 if the host cannot receive a SleepReq packet from a device without USB suspend. (6.3.2.2#4)
    f. Bits n+17 through n+23 of DWORD N are set to 0 (reserved). (6.3.2.2#5)

## TD 5.19 Remote Wake Test

**Assertions Covered**

1. 6.3.2.2#4, 6.3.3.6#3, 6.3.10#1, 6.3.10#2, 6.3.10#4, 6.3.10#6, 6.3.10#7, 6.3.28#1, 6.3.28#2, 6.3.56#1, 6.3.56#2, 6.3.56#3, 6.3.56#4, 6.3.56#5, 6.3.57#1, 6.3.57#2, 6.3.57#3, 6.3.57#4, 6.3.57#5, 6.3.57#6

2. 8.2.1.1#1, 8.2.1.1#2, 8.2.1.1#3, 8.2.1.1#4, 8.2.1.1#5, 8.2.1.1#8, 8.2.1.1#9, 8.2.2.2#3, 8.2.2.2#6, 8.2.2.2#7, 8.2.2.2#8

**Required Equipment**

1. MA USB Compliance Device with hub functionality
2. KG USB Mouse

**Iterations**

Repeat each of the following test cases where the Compliance Device:

- Does not carry a Link Sleep Capability Descriptor in the CapResp packet (Link Sleep not supported)
- Carries a Link Sleep Descriptor in the CapResp packet with the Link Sleep Capable field set to 0 (Link Sleep not supported)
- Carries a Link Sleep Descriptor in the CapResp packet with the Link Sleep Capable field set to 1 (Link Sleep is Supported)

**Test Steps**

**Case 1 – Host-initiated transition**

1. If the Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the Compliance Device to the HUT.
    b. Enumerate and configure the Compliance Device.
2. If a KG USB mouse is not connected to the Compliance Device or is connected but not operational:
    a. Connect/reconnect the KG USB mouse behind the Compliance Device and enumerate. *Note: this test assumes that the HUT will enable the mouse for remote wake during normal operation*.
3. Let the mouse sit idle for a period of time sufficient for HUT to suspend it.
4. Wait for the HUT to initiate transition of the MA USB session with the Compliance Device to the Inactive State in one of the following ways:

    *Link Sleep*

5. If the HUT does not support Link Sleep in its capability descriptor, the test fails if the HUT sends a SleepReq packet to the Compliance Device without USB suspend. (6.3.2.2#4)
6. If the Compliance Device does not support Link Sleep in this iteration of the test, the test fails if the HUT sends a SleepReq packet to the Compliance Device without USB suspend. (6.3.3.6#3)
7. Verify that a SleepReq packet from the HUT has:
    a. Type field set to 0. (6.3.56#1)
    b. Subtype field set to 54. (6.3.56#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.56#2)
    d. Device Handle field is set to 0 (reserved). (6.3.56#3)
    e. Management Request Timeout field set to non-zero value. (6.3.56#4)
    f. Data Request Timeout field set to non-zero value. (6.3.56#5)
8. Instruct the Compliance Device to send a SleepResp packet.
9. For the next minute, verify that the HUT either 1) stays in Session Inactive and does not send any MA USB packets to the Compliance Device; or 2) initiates transition to Session Active by sending a WakeReq packet to the Compliance Device. (8.1.1.4#2)
10. If the HUT sends a WakeReq during step 11, instruct the Compliance Device to respond with a WakeResp packet and restart the test from step 5.
11. Instruct the Compliance Device to send a WakeReq packet to the HUT.
12. Verify that HUT responds with a WakeResp packet. (6.3.33#1)
13. Verify that the WakeResp packet has:
    a. Type field set to 0. (6.3.59#2)
    b. Subtype field set to 57. (6.3.59#2)
    c. Device Handle field set to 0 (reserved). (6.3.59#3)
    d. Status Code field set to 0 (SUCCESS). (6.3.59#4, 8.2.2.2#3)
14. Verify that the HUT does not send a USBResumeReq packet. (6.3.3.6#3)
15. Operate the mouse to verify that MA USB session is active and the USB Device is functional. (8.2.2.2#6)

*Sleep with USB Suspend (Note: HUT must support regardless of Link Sleep Settings)*

5. Verify that the HUT first sends one or more EPInactivateReq packets to inactivate all EP Handles on the Compliance Device. (8.2.1.1#4)
6. Verify that the endpoint handles for each Compliance Device endpoint are contained in at least one EPInactivateReq packet from the HUT. (8.2.1.1#1)
7. Verify that each EPInactivateReq packet has:
   a. Type field set to 0. (6.3.10#1)
   b. Subtype field set to 8. (6.3.10#1)
   c. Status Code field set to 0 (SUCCESS). (6.3.10#2)
   d. Number of EP Handles field that is equal to the number of EP handles entries in the EP Handle List field. (6.3.10#4)
   e. Bits 6 through 31 of DWORD 3 are set to 0 (reserved). (6.3.10#6)
   f. EP Handle List field that carries a list of the EP handles concatenated in 16-bit increments. (6.3.10#7)
   g. Suspend Flag field set to 1 if all target endpoints are active. (8.2.1.1#8)
   h. Suspend Flag field set to 0 if any target endpoints are inactive. (8.2.1.1#9)
8. Verify that the HUT next sends a USBSuspendReq packet. (8.2.1.1#2, 8.2.1.1#5).
9. Verify that the USBSuspendReq packet has:
   a. Type field set to 0. (6.3.28#1)
   b. Subtype field set to 26. (6.3.28#1)
   c. Status Code field set to 0 (SUCCESS). (6.3.28#2)
10. Verify that the HUT next sends a SleepReq packet. (8.2.1.1#3)
11. Verify that the SleepReq packet has:
   a. Type field set to 0. (6.3.56#1)
   b. Subtype field set to 54. (6.3.56#1)
   c. Status Code field set to 0 (SUCCESS). (6.3.56#2)
   d. Device Handle field is set to 0 (reserved). (6.3.56#3)
   e. Management Request Timeout field set to non-zero value. (6.3.56#4)
   f. Data Request Timeout field set to non-zero value. (6.3.56#5)
12. Instruct the Compliance Device to send a SleepResp packet.
13. For the next minute, verify that the HUT either 1) stays in Session Inactive and does not send any MA USB packets to the Compliance Device; or 2) initiates transition to Session Active by sending a WakeReq packet to the Compliance Device. (8.1.1.4#2)
14. If the HUT sends a WakeReq during step 16, instruct the Compliance Device to respond with a WakeReq packet and restart the test from step 5.
15. Move the mouse to initiate Remote Wake.
16. Instruct the Compliance Device to send a RemoteWakeReq packet with the USB Device Resumed field set to 0.
17. Verify that HUT responds with a RemoteWakeResp packet. (6.3.33#1)
18. Verify that the RemoteWakeResp packet has:
   a. Type field set to 0. (6.3.33#2)
   b. Subtype field set to 31. (6.3.33#2)
   c. Status Code field set to value in Table 6. (6.3.33#4)
19. Verify that the HUT sends a USBResumeReq packet to the Compliance Device. (8.2.2.2#7)
20. Verify that the USBResumeReq packet has:
   a. Type field set to 0. (6.3.30#1)
   b. Subtype field set to 28. (6.3.30#1)
   c. Status Code Field set to 0 (SUCCESS). (6.3.30#2)
21. Verify that the HUT sends at least one EPActivateReq packet to the device. (8.2.2.2#7)
22. Verify that the EPActivateReq packet has:
   a. Type field set to 0. (6.3.8#1)
   b. Subtype field set to 6. (6.3.8#1)
   c. Status Code field set to 0 (SUCCESS). (6.3.8#2)
   d. Number of EP Handles field set to the number of EP Handles included in the packet. (6.3.8#4)
   e. Bits 5 through 31 set to 0 (reserved). (6.3.8#5)

   f. EP Handles List field with a list of EP handles concatenated in 16-bit increments. (6.3.8#6)
23. Operate the mouse to verify that MA USB session is active and the USB Device is functional. (8.2.2.2#6)
24. Repeat test with these modifications:
   a. In step 19, instruct the Compliance Device to send a RemoteWakeReq packet with the USB Device Resumed field set to 1.
   b. If HUT sends a USBResumeReq to the Compliance Device, test fails (8.2.2.2#8)

**Case 2 – Device-initiated transition**

1. Unless previously transitioned to Session Active, instruct the Compliance Device to send a RemoteWakeReq packet to the HUT. If remote wake is unsuccessful, disconnect then reconnect the Compliance Device, enumerate and configure as in Case 1 above.
2. From the HUT, initiate an IN transfer.
3. After receiving the first TransferReq packet for the IN transfer, instruct the Compliance Device to send a null TransferResp with the Status Code field set to 135 (TRANSFER_PENDING) and the ARQ bit set to 1.
4. Wait for the HUT to send a TransferAck packet acknowledging the TransferResp.
5. Instruct the Compliance Device to send a SleepReq packet to the HUT.
6. If either the HUT or the Compliance Device do not support Link Sleep, verify that the HUT either responds with an error or does not respond to the SleepReq packet. End Case 2 here.
7. If the HUT supports Link Sleep and the Compliance Device has indicated that it supports Link Sleep, verify that the HUT sends a SleepResp packet. (6.3.57#1)
8. Verify that the SleepResp packet has:
   a. Type field set to 0. (6.3.57#2)
   b. Subtype field set to 55. (6.3.57#2)
   c. Device Handle field set to 0 (reserved). (6.3.57#3)
   d. Status Code field set to either 0 (SUCCESS) or 155 (REQUEST_DENIED). (6.3.57#4)
   e. If Status Code is 0 (SUCCESS), Management Request Timeout field is less than or equal to the corresponding SleepReq packet. (6.3.57#5)
   f. If Status Code is 0 (SUCCESS), Data Request Timeout field is less than or equal to the corresponding SleepReq packet. (6.3.57#6)
9. For the next minute, verify that the HUT either 1) stays in Session Inactive and does not send any MA USB packets to the Compliance Device; or 2) initiates transition to Session Active by sending a WakeReq packet to the Compliance Device. (8.1.1.4#2)
10. If the HUT sends a WakeReq during step 16, instruct the Compliance Device to respond with a WakeReq packet and restart the test from step 6.

# TD 5.20 Link Sleep Test

*Note: Only perform this test if the HUT supports Link Sleep*

**Assertions Covered**

1. 6.3.57#1, 6.3.57#2, 6.3.57#3, 6.3.57#4, 6.3.57#5, 6.3.57#6, 6.3.59#1, 6.3.59#2, 6.3.59#3, 6.3.59#4
2. 8.1.1.3#2, 8.2.2.2#3, 8.2.2.2#9

**Required Equipment**

1. MA USB Compliance Device with hub functionality
2. KG USB Mouse

**Test Steps**

**Case 1 – Host-initiated transition**

1. If the Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
   a. Connect/reconnect the Compliance Device to the HUT.
   b. Enumerate and configure the Compliance Device with Link Sleep enabled (Link Sleep Capable field set to 1 in Link Sleep Descriptor).
2. If a KG USB mouse is not connected to the Compliance Device or is connected but not operational:
   a. Connect/reconnect the KG USB mouse behind the Compliance Device and enumerate.

3. Instruct the Compliance Device to send a SleepReq packet to the HUT.
4. Verify that the HUT sends SleepResp packet. (6.3.57#1)
5. Verify that the SleepResp packet has:
    a. Type field set to 0. (6.3.57#2)
    b. Subtype field set to 55. (6.3.57#2)
    c. Device Handle field set to 0 (reserved). (6.3.57#3)
    d. Status Code field set to either 0 (SUCCESS) or 155 (REQUEST_DENIED). (6.3.57#4)
    e. If Status Code is 0 (SUCCESS), Management Request Timeout field is less than or equal to the corresponding SleepReq packet. (6.3.57#5)
    f. If Status Code is 0 (SUCCESS), Data Request Timeout field is less than or equal to the corresponding SleepReq packet. (6.3.57#6)
6. Instruct Compliance Device to send a WakeReq packet to the HUT.
7. Verify that the WakeResp packet has:
    a. Type field set to 0. (6.3.59#2)
    b. Subtype field set to 57. (6.3.59#2)
    c. Device Handle field set to 0 (reserved). (6.3.59#3)
    d. Status Code field set to 0 (SUCCESS). (6.3.59#4, 8.2.2.2#3)
8. Verify that the mouse operates correctly. (8.1.1.3#2)

**Repetitions:**

- Repeat with PingReq from Compliance Device instead of WakeReq

**Case 2 – Device-initiated transition with data packet exchange (implicit request)**

1. Instruct the Compliance Device to send a TransferResp packet with the Status Code field set to 135 (TRANSFER_PENDING) and the ARQ bit set to 1. *Note: with the mouse attached and device inactive, there should be an interrupt IN TransferReq packet pending*.
2. Wait for the HUT to send a TransferAck packet acknowledging the TransferResp packet from the Compliance Device.
3. Instruct the Compliance Device to send a SleepReq packet to the HUT.
4. Verify that the HUT sends SleepResp packet. (6.3.57#1)
5. Verify that the SleepResp packet has:
    a. Type field set to 0. (6.3.57#2)
    b. Subtype field set to 55. (6.3.57#2)
    c. Device Handle field set to 0 (reserved). (6.3.57#3)
    d. Status Code field set to either 0 (SUCCESS) or 155 (REQUEST_DENIED). (6.3.57#4)
    e. If Status Code is 0 (SUCCESS), Management Request Timeout field is less than or equal to the corresponding SleepReq packet. (6.3.57#5)
    f. If Status Code is 0 (SUCCESS), Data Request Timeout field is less than or equal to the corresponding SleepReq packet. (6.3.57#6)
6. Move or click the mouse.
7. Instruct the Compliance Device to send a TransferResp packet to the HUT for the pending IN transfer with the EoT field set to 1.
8. Verify that the HUT sends a TransferAck or TransferReq packet acknowledging the TransferResp packet from the Compliance Device. (8.2.2.2#9)
9. Verify that the mouse functions correctly.

# TD 5.21 Endpoint Halt Test

**Assertions Covered**

1. 6.3.12#1, 6.3.12#2, 6.3.12#4, 6.3.12#5, 6.3.12#6, 6.3.12#10

**Required Equipment**

1. MA USB Compliance Device

**Test Steps**

1. If the MA USB Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the MA USB Compliance Device to the HUT.
    b. Enumerate and configure the MA USB Compliance Device.
2. Instruct the Compliance Device to respond to the first interrupt IN TransferReq packet from the HUT with a TransferResp packet with the Status Code set to 136 (TRANSFER_EP_STALL) and the EoT bit set to 1.
3. Wait for the HUT to clear the halted endpoint.
4. Verify that the HUT sends an EPResetReq packet.
5. Verify that the EPResetReq packet has:
    i. Type field set to 0. (6.3.12#1)
    ii. Subtype field set to 10. (6.3.12#1)
    iii. Status Code field is set to 0 (NO_ERROR). (6.3.12#2)
    iv. Number of EP Reset Information Blocks field set to the number of EP reset information blocks included in the packet. (6.3.12#4)
    v. Bits 5 through 31 set to 0 (reserved). (6.3.12#5)
6. Verify that any EP reset information blocks in the EPResetReq packet have:
    a. EP Handle field set to target endpoint handle. (6.3.12#6)
    b. Bits 17 through 31 set to 0 (reserved). (6.3.12#10)

# TD 5.22 Session Tear Down Test

**Assertions Covered**

1. 6.3.10#1, 6.3.10#2, 6.3.10#4, 6.3.10#6, 6.3.10#7, 6.3.14#1, 6.3.14#2, 6.3.14#4, 6.3.14#5, 6.3.14#6, 6.3.14#7, 6.3.14#9, 6.3.16#1, 6.3.16#2, 6.3.16#4, 6.3.16#5, 6.3.16#6, 6.3.26#1, 6.3.26#2, 6.3.36#1, 6.3.36#2, 6.3.36#3, 6.3.39#1, 6.3.39#2, 6.3.39#3, 6.3.39#4, 6.3.42#1, 6.3.42#2, 6.3.42#4, 6.3.42#6, 6.3.42#7, 6.3.42#8
2. 8.1.1.1#2, 8.1.1.1#3, 8.1.1.1#4, 8.1.3#1, 8.1.3.1#1, 8.1.3.1#2, 8.1.3.2#1, 8.1.3.2#2, 8.1.3.3#1, 8.1.3.4#1, 8.1.3.4#2, 8.1.3.4#3, 8.1.3.4#4

**Required Equipment**

1. MA USB Compliance Device

**Test Steps**

**Case 1 – Implicit Session Tear Down – Loss of Connectivity**

1. Connect the Compliance Device to the HUT.
2. Enumerate and configure the Compliance Device.
3. Disconnect or otherwise remove medium connection between the HUT and Compliance Device.
4. Verify that the USB device does not show up in the USB tree. (8.1.3.1#1)

**Case 2 – Implicit Session Tear Down – Management Packet Transmission Failure**

1. Connect the Compliance Device to the HUT.
2. Begin enumeration.
3. Instruct the Compliance Device to stop responding to MA USB packets after receiving a ModifyEP0Req packet from the HUT.
4. If the HUT does not initiate the Ping Protocol:
    a. Instruct the Compliance Device to wait for 1 minute after receiving the last ModifyEP0Req packet from the HUT.
    b. Verify that the HUT does not send any more MA USB packets to the Compliance Device. (8.1.3.1#2)
5. If the HUT initiates the Ping Protocol:
    a. Instruct the Compliance Device to wait for 1 minute after receiving the last PingReq packet from the HUT.
    b. Verify that the HUT does not send any MA USB packets to the Compliance Device. (8.1.3.1#2)
6. Verify that the Compliance Device either does not show up in the USB Device Tree of the HUT or shows up but is flagged with an error message. (8.1.1.1#2)

**Case 3 – Host-Initiated Session Tear Down**

1. Connect the Compliance Device to the HUT.
2. Enumerate and configure the Compliance Device.

3. Eject or otherwise remove the Compliance Device from the HUT system.
4. For each integrated USB device, perform the USB Device Removal Procedure Verification Steps outlined below.
5. Verify that the HUT sends a DevDisconnectReq packet to the Compliance Device. (8.1.3#1, 8.1.3.2#1)
6. Verify that HUT sends DevDisconnectReq with:
    a. Type field set to 0. (6.3.36#1)
    b. Subtype field set to 34. (6.3.36#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.36#2)
    d. Device handle field set to 0 (reserved). (6.3.36#3)
7. Verify that Compliance Device does not show up in the USB tree on HUT system. (8.1.1.1#3, 8.1.3.2#2)
8. Instruct Compliance Device to send a PingReq packet to the HUT.
9. Verify that the HUT does not respond. (8.1.1.1#4)

**Case 4 – Device-Initiated Session Tear Down**

1. Connect the Compliance Device to the HUT.
2. Enumerate and configure the Compliance Device.
3. Instruct the Compliance Device to send a DevInitDisconnectReq packet.
4. Verify that the HUT sends a DevInitDisconnectResp packet. (6.3.39#1, 8.1.3.3#1)
5. Verify that the DevInitDisconnectResp has:
    a. Type field set to 0. (6.3.39#2)
    b. Subtype field set to 37. (6.3.39#2)
    c. Device Handle field set to 0 (reserved). (6.3.39#3)
    d. Status Code field set to 0 (NO_ERROR). (6.3.39#4)
6. For each integrated USB device, perform the USB Device Removal Procedure Verification Steps outlined below.
7. Verify that the HUT sends a DevDisconnectReq packet to the Compliance Device. (8.1.3#1)
8. Verify that HUT sends DevDisconnectReq with:
    a. Type field set to 0. (6.3.36#1)
    b. Subtype field set to 34. (6.3.36#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.36#2)
    d. Device handle field set to 0 (reserved). (6.3.36#3)
9. Verify that Compliance Device does not show up in the USB tree on HUT system. (8.1.1.1#3)
10. Instruct Compliance Device to send a PingReq packet to the HUT.
11. Verify that the HUT does not respond. (8.1.1.1#4)

**USB Device Removal Procedure Verification Steps**

1. Verify that the HUT first inactivates and clears all outstanding transfers by sending either 1) CancelTransferReq packets followed by an EPInactivateReq packet; or 2) an EPInactivateReq packet followed by a ClearTransfersReq packet. (8.1.3.4#1)
2. If the HUT sends any CancelTransferReq packets, verify that each CancelTransferReq packet has:
    a. Type field set to 0. (6.3.42#1)
    b. Subtype field set to 40. (6.3.42#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.42#2)
    d. Dialog Token field set to 0 (reserved). (6.3.42#4)
    e. For endpoints that are not Enhanced SS bulk endpoints that support the Enhanced SS Stream Protocol, Stream ID field set to 0 (reserved). (6.3.42#6)
    f. Request ID field set to a value between 0 and aMaxRequestID. (6.3.42#7)
    g. Bits 8 through 31 of DWORD 4 set to 0 (reserved). (6.3.42#8)
3. Verify that the EPInactivateReq packet has:
    a. Type field set to 0. (6.3.10#1)
    b. Subtype field set to 8. (6.3.10#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.10#2)
    d. Number of EP Handles field that is equal to the number of EP handles entries in the EP Handle List field. (6.3.10#4)
    e. Bits 6 through 31 of DWORD 3 are set to 0 (reserved). (6.3.10#6)

   f. EP Handle List field that carries a list of the EP handles concatenated in 16-bit increments. (6.3.10#7)
4. If the HUT sends a ClearTransfersReq packet, verify that the ClearTransfersReq packet has:
   a. Type field set to 0. (6.3.14#1)
   b. Subtype field set to 12. (6.3.14#1)
   c. Status Code field set to 0 (SUCCESS). (6.3.14#2)
   d. Number of clear transfers information blocks set to the number of clear transfers information blocks included in the packet. (6.3.14#4)
   e. Bits 8 through 31 of DWORD 3 set to 0 (reserved). (6.3.14#5)
   f. For each clear transfer information block, verify that:
      i. The EP Handle field carries the endpoint handle the request is targeting (6.3.14#6)
      ii. If the target endpoint is an Enhanced SuperSpeed bulk endpoint that supports the Stream Protocol, the Stream ID field set to indicate the target stream. (6.3.14#7)
      iii. If the target endpoint is not an Enhanced SuperSpeed bulk endpoint that supports the Stream Protocol, the Stream ID field is reserved and set to 0. (6.3.14#7)
      iv. Bits 8 through 31 of DWORD 1 reserved and set to 0. (6.3.14#9)
5. Verify that the HUT next deletes all EP handles except for EP0. (8.1.3.4#2)
6. Verify that any EPHandleDeleteReq packets have:
   a. Type field set to 0. (6.3.16#1)
   b. Subtype field set to 14. (6.3.16#1)
   c. Status Code field set to 0 (SUCCESS). (6.3.16#2)
   d. Number of EP Handles field that carries the number of EP handles included in the packet. (6.3.16#4)
   e. Bits 5 through 31 of DWORD 3 set to 0 (reserved). (6.3.16#5)
   f. EP Handle List field that carries the number of EP handles indicated in the Number of EP Handles field. (6.3.16#6)
7. Verify that after deleting all other EP handles, the HUT inactivates then deletes EP0. (8.1.3.4#3)
8. Verify that the EPInactivateReq packet has:
   a. Type field set to 0. (6.3.10#1)
   b. Subtype field set to 8. (6.3.10#1)
   c. Status Code field set to 0 (SUCCESS). (6.3.10#2)
   d. Number of EP Handles field set to 1. (6.3.10#4)
   e. Bits 6 through 31 of DWORD 3 are set to 0 (reserved). (6.3.10#6)
   f. EP Handle List field that carries the EP Handle for EP0. (6.3.10#7)
9. Verify that the EPHandleDeleteReq packet has:
   a. Type field set to 0. (6.3.16#1)
   b. Subtype field set to 14. (6.3.16#1)
   c. Status Code field set to 0 (SUCCESS). (6.3.16#2)
   d. Number of EP Handles field set to 1. (6.3.16#4)
   e. Bits 5 through 31 of DWORD 3 set to 0 (reserved). (6.3.16#5)
   f. EP Handle List field that carries the EP Handle for EP0. (6.3.16#6)
10. Verify that the HUT sends a USBDevDisconnectReq packet. (8.1.3.4#4)
11. Verify that the USBDevDisconnectReq packet has:
   a. Type field set to 0. (6.3.26#1)
   b. Subtype field set to 24. (6.3.26#1)
   c. Status Code field set to 0 (SUCCESS). (6.3.26#2)

# TD 5.23 Enhanced SuperSpeed Bulk Stream Test

*Note: Only run this test if the HUT supports Enhanced SuperSpeed Bulk Endpoints.*

**Assertions Covered**

1. 6.3.44#1, 6.3.44#2, 6.3.44#6, 6.3.44#10, 6.3.46#1, 6.3.46#2, 6.3.46#6, 6.3.46#7, 6.3.46#8, 6.3.46#9

**Required Equipment**

1. MA USB Compliance Device with hub functionality

2. UASP-Capable USB Device

**Test Steps**

1. If the MA USB Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the MA USB Compliance Device to the HUT.
    b. Enumerate and configure the MA USB Compliance Device.
2. If the UASP-Capable Device is not connected to the MA USB Compliance Device or is connected but not operational:
    a. Connect/reconnect the UASP-Capable Device to the MA USB Compliance Device.
    b. Enumerate and configure the UASP-Capable Device.
3. Operate the UASP-Capable Device.
4. Verify that any EPOpenStreamReq packets have:
    a. Type field set to 1. (6.3.44#1)
    b. Subtype field set to 42. (6.3.44#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.44#2)
    d. If the Open Stream field is not 0, the Stream ID Index field is set to 0 (reserved). (6.3.44#6)
    e. Bits 18 through 31 of DWORD 4 are set to 0 (reserved). (6.3.44#10)
5. Eject or otherwise remove the UASP-Capable Device.
6. Verify that any EPCloseStreamReq packets have:
    a. Type field set to 1. (6.3.46#1)
    b. Subtype field set to 44. (6.3.46#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.46#2)
    d. Bits 17 through 31 of DWORD 3 are set to 0 (reserved). (6.3.46#6)
    e. Number of Stream ID Block field set to number of stream ID blocks in packet. (6.3.46#7)
    f. Number of Stream ID Block field is set to 0 if Close All field is 1. (6.3.46#8)
    g. Bits 16 through 31 of DWORD 4 set to 0 (reserved). (6.3.46#9)

# TD 5.24 Get Port Bandwidth Test

*Note: Only run this test if the HUT supports GetPortBWReq packets*

**Assertions Covered**

1. 6.3.54#1, 6.3.54#2, 6.3.54#5, 6.3.54#7, 6.3.54#8, 6.3.54#9, 6.3.54#10, 6.3.54#11, 6.3.54#12, 6.3.54#13, 6.3.54#14

**Required Equipment**

1. MA USB Compliance Device with integrated SuperSpeed hub
2. KG USB 3.1 Gen 2 hub (if HUT supports USB 3.1 Gen 2)
3. KG USB 3.1 Gen 1 hub (if HUT does not support USB 3.1 Gen 2)

**Test Steps**

**Case 1 – Device Support for GetPortBWReq**

1. If the MA USB Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the MA USB Compliance Device to the HUT.
    b. Enumerate and configure the MA USB Compliance Device.
2. After enumeration and configuration of the MA USB Compliance Device is complete, connect the KG USB Hub to the MA USB Compliance Device.
3. Enumerate and configure the USB KG Hub.
4. If the HUT sends any GetPortBWReq packets during the test:
    a. Verify that each GetPortBWReq has:
        i. Type field set to 0. (6.3.54#1)
        ii. Subtype field set to 52. (6.3.54#1)
        iii. Status Code field set to 0 (SUCCESS). (6.3.54#2)
        iv. Speed field value that is not greater than 4. (6.3.54#7)
        v. Bits 4 through 31 of DWORD 3 set to 0 (reserved). (6.3.54#8)
        vi. Bits 0 through 3 of DWORD 4 set to 0 (reserved). (6.3.54#8)

vii. If Speed field is 3 or 4, Lane Speed Exponent field set to lane speed exponent of the integrated hub. (6.3.54#9)
viii. If Speed field is less than 3, Lane Speed Exponent field set to 0 (reserved). (6.3.54#9)
ix. If Speed field is 3 or 4, Lane Count field set to lane count of the integrated hub. (6.3.54#10)
x. If Speed field is less than 3, Lane Count field set to 0 (reserved). (6.3.54#10)
xi. If Speed field is 3 or 4, Link Protocol field set to link protocol of the USB integrated hub. (6.3.54#11)
xii. If Speed field is less than 3, Link Protocol field set to 0 (reserved). (6.3.54#11)
xiii. Bits 12 through 16 of DWORD 3 are set to 0 (reserved). (6.3.54#12)
xiv. If Speed field is 3 or 4, Lane Speed Mantissa field set to lane speed mantissa of integrated hub. (6.3.54#13)
xv. If Speed field is less than 3, Lane Speed Mantissa field set to 0 (reserved). (6.3.54#13)

b. Instruct the Compliance Device to respond to each GetPortBWReq packet with a GetPortBWResp packet that has the Status Code field set to 0 (SUCCESS).

**Case 2 – Device does not support GetPortBWReq**

1. Disconnect the KG USB Hub and MA USB Compliance Device
2. Reconnect the MA USB Compliance Device to the HUT.
3. Enumerate and configure the MA USB Compliance Device.
4. After enumeration and configuration of the MA USB Compliance Device is complete, connect the KG USB Hub to the MA USB Compliance Device.
5. Enumerate and configure the USB KG Hub.
6. If the HUT sends a GetPortBWReq packet during the test:
    a. verify that each GetPortBWReq has:
        i. Type field set to 0. (6.3.54#1)
        ii. Subtype field set to 52. (6.3.54#1)
        iii. Status Code field set to 0 (SUCCESS). (6.3.54#2)
        iv. Speed field value that is not greater than 4. (6.3.54#7)
        v. Bits 4 through 31 of DWORD 3 set to 0 (reserved). (6.3.54#8)
        vi. Bits 0 through 3 of DWORD 4 set to 0 (reserved). (6.3.54#8)
        vii. If Speed field is 3 or 4, Lane Speed Exponent field set to lane speed exponent of the integrated hub. (6.3.54#9)
        viii. If Speed field is less than 3, Lane Speed Exponent field set to 0 (reserved). (6.3.54#9)
        ix. If Speed field is 3 or 4, Lane Count field set to lane count of the integrated hub. (6.3.54#10)
        x. If Speed field is less than 3, Lane Count field set to 0 (reserved). (6.3.54#10)
        xi. If Speed field is 3 or 4, Link Protocol field set to link protocol of the USB integrated hub. (6.3.54#11)
        xii. If Speed field is less than 3, Link Protocol field set to 0 (reserved). (6.3.54#11)
        xiii. Bits 12 through 16 of DWORD 3 are set to 0 (reserved). (6.3.54#12)
        xiv. If Speed field is 3 or 4, Lane Speed Mantissa field set to lane speed mantissa of integrated hub. (6.3.54#13)
        xv. If Speed field is less than 3, Lane Speed Mantissa field set to 0 (reserved). (6.3.54#13)
    b. Instruct the Compliance Device to respond with a GetPortBWResp packet with the Status Code field set to 154 (NOT_SUPPORTED).
    c. Verify that the HUT does not send any more GetPortBWReq packets after receiving the GetPortBWResp from the Compliance Device with the Status Code field set to 154 (NOT_SUPPORTED). (6.3.54#5)

# TD 5.25 Inactive Session Timing Test

**Assertions Covered**

1. 6.3.56#5, 6.3.57#5, 6.3.57#6
2. 8.2.1.1#7, 8.2.1.2#3

**Required Equipment**

1. MA USB Sniffer

2. MA USB Packet Analysis Software
3. MA USB Compliance Device
4. Loopback-Capable USB Compliance Device

**Test Steps**

**Case 1 – Device-initiated Sleep**

*Note: only run this test case if HUT supports Link Sleep*

1. Connect the MA USB Compliance Device to the HUT.
2. Enumerate and configure the MA USB Compliance Device to support Link Sleep and have one non-isochronous IN endpoint.
3. Connect the USB Compliance Device downstream of the MA USB Compliance Device.
4. Enumerate and configure the USB Compliance Device for loopback.
5. From the HUT, initiate a bulk IN transfer.
6. After receiving the first TransferReq packet for the bulk IN transfer, instruct the MA USB Compliance Device to send a TransferResp with the Status Code field set to 135 (TRANSFER_PENDING) and the ARQ bit set to 1.
7. Wait for the HUT to send a TransferAck packet acknowledging the TransferResp.
8. Ensure that the following steps are captured by the MA USB Sniffer.
9. Instruct the MA USB Compliance Device to send a SleepReq packet to the HUT with the Management Request Timeout field set to 5000 and the Data Request Timeout field set to 4000.
10. If the HUT responds with a SleepResp packet that has the Status Code field set to 155 (REQUEST_DENIED), instruct the MA USB Compliance Device to send a SleepReq packet to the HUT with the same Management Request Timeout and Data Request Timeout fields as the received SleepResp packet.
11. If the HUT responds with a SleepResp with Status Code Field set to 0 (SUCCESS), continue with the test. Otherwise end test here or retry from beginning.
12. For the next minute:
    a. If the HUT sends any data or control packets, instruct the MA USB Compliance Device to not respond until after the number of milliseconds in the Data Request Timeout field has expired.
        i. Verify that the HUT retries the packet. (8.2.1.1#7, 6.3.57#6)
        ii. Instruct the MA USB Compliance Device to respond to the retried packet within Data Request Timeout value - 2 * aDataChannelDelay.
        iii. Verify that the HUT does not retry the packet. (8.2.1.1#7, 6.3.57#6)
    b. If the HUT sends any management packets, instruct the MA USB Compliance Device to not respond until the number of milliseconds in the Management Request Timeout field has passed.
        i. Verify that the HUT retries the packet. (8.2.1.1#7)
        ii. Instruct the MA USB Compliance Device to respond to the retried packet within Management Request Timeout value - 2 * aManagementChannelDelay.
        iii. Verify that the HUT does not retry the packet. (8.2.1.1#7)
13. After 1 minute has passed, instruct the MA USB Compliance Device to send a WakeReq packet to the HUT.
14. Verify that the Compliance Device receives a WakeResp packet from the HUT within the time indicated by the Management Request Timeout field of the SleepResp packet previously sent by the HUT. (8.2.1.2#3, 6.3.57#5)
15. Disconnect the USB and MA USB Compliance Devices from the HUT.

**Case 2 – Host-Initiated Sleep**

1. Connect the MA USB Compliance Device to the HUT.
2. Enumerate and configure the MA USB Compliance Device.
3. Let the MA USB Compliance Device sit idle for a period of time sufficient for HUT to suspend it.
4. Wait for the HUT to send a SleepReq packet, then record the values in the Management Request Timeout and Data Request Timeout fields.
5. Instruct the MA USB Compliance Device to respond with a SleepResp packet that has the same Management Request Timeout and Data Request Timeout fields as the SleepReq packet.
6. For the next minute:
    a. If the HUT sends any management packets, instruct the MA USB Compliance Device to not respond until the number of milliseconds in the Management Request Timeout field has passed.

i. Verify that the HUT retries the packet. (8.2.1.1#7)
ii. Instruct the MA USB Compliance Device to respond to the retried packet within Management Request Timeout value - 2 * aManagementChannelDelay.
iii. Verify that the HUT does not retry the packet. (8.2.1.1#7)

b. If the HUT sends any data or control packets, instruct the MA USB Compliance Device to not respond until the number of milliseconds in the Data Request Timeout field has passed.
i. Verify that the HUT retries the packet. (8.2.1.1#7, 6.3.56#5)
ii. Instruct the MA USB Compliance Device to respond to the retried packet within Data Request Timeout value - 2 * aDataChannelDelay.
iii. Verify that the HUT does not retry the packet. (8.2.1.1#7, 6.3.56#5)

7. Wait for 1 Minute.
8. If the HUT supports Link Sleep, instruct the MA USB Compliance Device to send a WakeReq packet to the HUT.
9. If the HUT does not support Link Sleep, instruct the MA USB Compliance Device to send a RemoteWakeReq packet to the HUT.
10. Verify that the HUT sends a response packet within the time indicated by the Management Request Timeout field of the SleepReq packet previously sent by the HUT. (8.2.1.1#7, 6.3.57#5)

# TD 5.26 Pending Transfer Acknowledgement Test

**Assertions Covered**

1. 6.5.4#3

**Required Equipment**

1. MA USB Compliance Device
2. Loopback-Capable USB Compliance Device

**Test Steps**

1. If the MA USB Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
   a. Connect/reconnect the MA USB Compliance Device to the HUT.
   b. Enumerate and configure the MA USB Compliance Device.
2. If the loopback-capable USB Compliance Device is not connected to the HUT or is not operational:
   a. Connect/reconnect the USB Compliance Device downstream of the MA USB Compliance Device.
   b. Enumerate and configure the USB Compliance Device for loopback.
3. Initiate loopback on the USB Compliance Device.
4. In response to the first bulk IN TransferReq from the HUT, instruct the Compliance Device to send a TransferResp packet that has the ARQ bit set to 1 and the Status Code set to 135 (TRANSFER_PENDING).
5. Verify that the HUT responds with a TransferAck packet with the Status Code set to 135 (TRANSFER_PENDING). (6.5.4#3)
6. Instruct the Compliance Device to send the remaining TransferResp packets.
7. Verify that transfer completes successfully.

# TD 5.27 Unsuccessful Data Transfer Test

**Assertions Covered**

1. None

**Required Equipment**

1. MA USB Compliance Device
2. Loopback-Capable USB Compliance Device

**Test Steps**

1. If the MA USB Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
   a. Connect/reconnect the MA USB Compliance Device to the HUT.
   b. Enumerate and configure the MA USB Compliance Device.
2. If the loopback-capable USB Compliance Device is not connected to the HUT or is not operational:
   a. Connect/reconnect the USB Compliance Device downstream of the MA USB Compliance Device.

   b. Enumerate and configure the USB Compliance Device for loopback.
3. Initiate loopback on the USB Compliance Device.
4. Instruct the Compliance Device to send a TransferResp packet with the Status Code set to 128 (UNSUCCESSFUL) and the ARQ field set to 1.
5. Verify that the HUT sends a TransferAck packet with the Status Code field set to 128 (UNSUCCESSFUL).
6. Verify that the HUT does not send any more packets with the same Request ID as the transfer that was unsuccessful.
7. Initiate a second loopback in the USB Compliance Device.
8. Verify that subsequent transfers complete successfully.

**Repetitions**

- Repeat for bulk IN and bulk OUT transfers.

# TD 5.28 Unsuccessful Management Packet Test

**Assertions Covered**

1. 8.1.2.1#9

**Required Equipment**

1. MA USB Compliance Device with hub functionality
2. KG USB Device
3. Loopback-Capable USB Compliance Device

**Test Steps**

**Case 1 – MA USB Device Reset**

1. Connect the MA USB Compliance Device to the HUT.
2. Start enumeration.
3. Instruct the Compliance Device to respond to the DevResetReq packet with a DevResetResp packet that has the Status Code field set to 128 (UNSUCCESSFUL).
4. Verify that HUT does not send any packets other than a DevResetReq. (8.1.2.1#9)
5. Verify that the HUT either resends a DevResetReq packet or stops enumeration and reports the error.
6. Disconnect the Compliance Device.

**Case 2 – MA USB Device Enumeration and Configuration**

1. Connect the Compliance Device to the HUT.
2. Start enumeration.
3. Instruct the Compliance Device to respond to the CapReq packet with a CapResp packet that has the Status Code field set to 128 (UNSUCCESSFUL).
4. Verify that the HUT either resends the CapReq packet or stops enumeration and reports the error.
5. Disconnect the Compliance Device.

**Repetitions**

- Repeat Case 2 for each of the following packet types with that packet type replacing the CapReq/Resp packets.
    - USBDevHandleReq/Resp
    - EPHandleReq/Resp
    - ModifyEP0Req/Resp
    - SetUSBDevAddrReq/Resp
    - UpdateDevReq/Resp
- Repeat packet iterations listed above for each integrated USB device.

**Case 3 – USB Device Enumeration and Configuration**

1. Connect the Compliance Device to the HUT.
2. Enumerate and configure the Compliance Device.
3. Connect the KG USB Device to the Compliance Device.

4. Start enumeration.
5. Instruct the Compliance Device to respond to the USBDevHandleReq packet with a USBDevHandleResp packet that has the Status Code field set to 128 (UNSUCCESSFUL).
6. Verify that the HUT either resends the USBDevHandleReq packet or stops enumeration and reports the error.
7. Disconnect the KG USB Device from the Compliance Device.
8. Disconnect the Compliance Device from the HUT.

**Repetitions**

- Repeat Case 3 for each of the following packet types with that packet type replacing the USBDevHandleReq/Resp packets.
    - EPHandleReq/Resp
    - ModifyEP0Req/Resp
    - SetUSBDevAddrReq/Resp
    - UpdateDevReq/Resp

**Case 4 – Device-Initiated Disconnect**

1. Connect the Compliance Device to the HUT.
2. Enumerate and configure the Compliance Device.
3. Instruct the Compliance Device to send a DevInitDisconnectReq packet to the HUT.
4. Verify that the HUT sends a DevInitDisconnectResp packet in response.
5. Instruct the Compliance Device to send an EPInactivateResp packet that has the Status Code field set to 128 (UNSUCCESSFUL) in response to the first EPInactivateReq from the HUT.
6. Verify that the HUT resends the EPInactivateReq packet and/or reports the error.

**Repetitions**

- Repeat Case 4 for each of the following packet types with that packet type replacing the EPInactivateReq /Resp packets.
    - ClearTransfersReq/Resp or CancelTransferReq/Resp*
    - EPHandleDeleteReq/Resp
    - USBDevDisconnectReq/Resp
    - DevDisconnectReq/Resp
- Repeat packet iterations listed above for each integrated USB device.

** Note: the HUT may send ClearTransfersReq and/or CancelTransferReq packets, but is not required to send both.*

**Case 5 – Transition to Session Active**

1. If the MA USB Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the MA USB Compliance Device to the HUT.
    b. Enumerate and configure the MA USB Compliance Device to not support Link Sleep.
2. If the loopback-capable USB Compliance Device is not connected to the HUT or is not operational:
    a. Connect/reconnect the USB Compliance Device downstream of the MA USB Compliance Device.
    b. Enumerate and configure the USB Compliance Device for loopback.
3. Let the USB Compliance Device sit idle for a period of time sufficient for the HUT to suspend it and transition the MA USB session to Inactive.
4. Initiate loopback on the USB Compliance Device.
5. Instruct the Compliance Device to respond to the WakeReq packet with a WakeResp packet that has the Status Code field set to 128 (UNSUCCESSFUL).
6. If the HUT retries the WakeReq packet, instruct the Compliance Device to respond with a WakeResp packet that has the Status Code set to 0 (SUCCESS).
7. If the HUT does not retry the WakeReq packet but takes other recovery measures such as resetting the USB or MA USB Compliance Devices, instruct the Compliance Device to respond to any MA USB packets send by the HUT.
8. Initiate loopback transfer on the USB Compliance Device.
9. Verify that loopback completes successfully.

**Repetitions**

- Repeat test for each of the following packet types with that packet type replacing the WakeReq/Resp packets.
    - USBResumeReq/Resp
    - EPActivateReq/Resp
- Repeat packet iterations listed above for each integrated USB device.

# TD 5.29 Streams Test

**Assertions Covered**

1. 7.3.4#1, 7.3.4#2

**Required Equipment**

1. MA USB Compliance Device with hub capabilities
2. UASP Capable USB Device

**Test Steps**

1. If the MA USB Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the MA USB Compliance Device to the HUT.
    b. Enumerate and configure the MA USB Compliance Device.
2. If the UASP Capable USB Device is not connected to the MA USB Compliance Device or is connected but not operational:
    a. Connect/reconnect the UASP Capable USB Device to the MA USB Compliance Device.
    b. Enumerate and configure the UASP Capable USB Device.
3. Verify that the HUT sends at least one EPOpenStreamReq packet. (7.3.4#1)
4. For each EPOpenStreamReq packet the HUT sends, verify that the number of streams specified is less than or equal to the Number of Streams specified in the CapResp packet from the Compliance Device. (7.3.4#2)

**Repetitions**

- Repeat test with the Number of Streams field in the CapResp set to 1, 2, 3, 4, and 8.

# TD 5.30 SSID Test

**Assertions Covered**

1. 8.1.2.1#3, 8.1.2.1#4, 8.1.2.1#8

**Required Equipment**

1. MA USB Compliance Host
2. KG MA USB Devices (2)

**Test Steps**

1. Connect a KG MA USB Device to the HUT.
2. Enumerate the KG MA USB Device and record the SSID that the HUT uses.
3. Disconnect the KG MA USB Device and shutdown the HUT.
4. Connect a KG MA USB Device to the Compliance Host and enumerate using the same SSID as previously used by the HUT.
5. Restart the HUT.
6. Connect a KG MA USB Device to the HUT.
7. Enumerate the KG MA USB Device.
8. For all MA USB packets from the HUT:
    a. Verify that the SSID field is set to an integer value between 1 and 254. (8.1.2.1#3)
    b. Verify that the SSID value from the HUT is different than the SSID value used by the Compliance Host. (8.1.2.1#3)Verify that the SSID field is not 0 or 255. (8.1.2.1#4)
    c. Verify that the SSID field is the same as the SSID field in all previous packets. (8.1.2.1#8)

# TD 5.31 Short Packet Test

**Assertions Covered**

1. none

**Required Equipment**

1. MA USB Compliance Device with hub functionality
2. Loopback-Capable USB Compliance Device

**Test Steps**

1. If the MA USB Compliance Device is not connected to the HUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the MA USB Compliance Device to the HUT.
    b. Enumerate and configure the MA USB Compliance Device.
2. If the USB Compliance Device is not connected to the MA USB Compliance Device or is connected but not operational:
    a. Connect/reconnect the USB Compliance Device downstream of the MA USB Compliance Device.
    b. Enumerate and configure the USB Compliance Device for loopback transfer.
3. From the HUT system, initiate multi-packet bulk loopback on the Compliance Device.
4. Prior to IN transfer completion, instruct the Compliance Device to send a TransferResp packet with the EoT flag set to 1 and the Status Code set to 141 (TRANSFER_SHORT_TRANSFER).
5. Verify that the HUT reports receipt of a short packet to the application that initiated loopback.
6. Repeat 10 times.

# TD 5.32 Loopback Test

**Assertions Covered**

1. none

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. KG MA USB Hub
4. LS, FS, HS and SS Loopback-Capable USB Compliance Devices[3]

**Test Steps**

**Case 1 – Control Loopback**

1. If the MA USB Hub is not connected to the HUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the MA USB Hub to the HUT.
    b. Enumerate and configure the MA USB Hub.
2. If the USB Compliance Device is not connected to the MA USB Hub or is connected but not operational:
    a. Connect/reconnect the USB Compliance Device downstream of the MA USB Hub.
    b. Enumerate and configure the USB Compliance Device for loopback transfer.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Initiate an OUT transfer to the control endpoint.
5. Initiate an IN transfer to the control endpoint.
6. Verify that the OUT data matches the IN data (data is the same and is in the same order).

**Repetitions**

- Repeat with LS USB Compliance Device with a max packet size of 8 bytes
- Repeat with FS USB Compliance Device with max packet sizes of 8, 16, 32, and 64 bytes
- Repeat with HS USB Compliance Device with a max packet size of 64 bytes
- Repeat with SS USB Compliance Device with a max packet size of 512 bytes

**Case 2 – Bulk Loopback**

[3] Note: If USB Compliance Devices are not available, KG USB Devices with multiple endpoint types can also be used for testing.

1. If the USB Compliance Device is not operational, reconnect the USB Compliance Device behind the MA USB Hub, then enumerate.
2. Configure the Compliance Device for bulk endpoint loopback.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Initiate an OUT transfer to the bulk OUT endpoint.
5. Verify that the OUT transfer completes successfully.
6. Initiate an IN transfer to the bulk IN endpoint.
7. Verify that the IN transfer completes successfully.
8. Verify that the OUT data matches the IN data (data is the same and is in the same order).
9. Repeat for 10 iterations.

**Repetitions**

- Repeat with FS USB Compliance Device with max packet sizes of 8, 16, 32, and 64 bytes
- Repeat with HS USB Compliance Device with a max packet size of 512 bytes
- Repeat with SS USB Compliance Device with a max packet size of 1024 bytes
- Repeat with SS USB Compliance Device with max burst size from 1 to 16

**Case 3 – Interrupt Loopback**

1. If the USB Compliance Device is not operational, reconnect the USB Compliance Device behind the MA USB Hub, then enumerate.
2. Configure the Compliance Device for interrupt endpoint loopback.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Initiate an OUT transfer to the interrupt OUT endpoint.
5. Verify that the OUT transfer completes successfully.
6. Initiate an IN transfer to the interrupt IN endpoint.
7. Verify that the IN transfer completes successfully.
8. Verify that the OUT data matches the IN data (data is the same and is in the same order).
9. Repeat for 10 iterations.

**Repetitions**

- Repeat with LS, FS, HS, and SS USB Compliance Devices
- Repeat for all allowed max packet sizes from 1 to 1040
- Repeat for all allowed max burst size for SS or HS interrupt transfers (1 to 3)

**Case 4 – Isochronous OUT Loopback**

1. If the USB Compliance Device is not operational, reconnect the USB Compliance Device behind the MA USB Hub, then enumerate.
2. Configure the Compliance Device for isochronous endpoint OUT looped back to bulk IN.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Initiate an OUT transfer with wMaxPacketSize Burst.
5. Verify that the OUT transfer completes successfully.
6. Initiate an IN transfer to the bulk IN endpoint.
7. Verify that the OUT transfer completes successfully.
8. Verify that the OUT data matches the IN data (data is the same and is in the same order).
9. Repeat for 10 iterations.

**Repetitions**

- Repeat with FS, HS, and SS USB Compliance Devices
- Repeat for transfer size of (wMaxPacketSize * Burst / 2) +1
- Repeat for all allowed max packets sizes for isochronous transfers
- Repeat for all allowed max burst size for HS isochronous transfers (1 to 3) and all allowed max burst size for SS isochronous transfers (1 to 16)
- Repeat for all allowed number of bursts (multi) for SS

**Case 5 – Isochronous IN Loopback**

1. If the USB Compliance Device is not operational, reconnect the USB Compliance Device behind the MA USB Hub, then enumerate.
2. Configure the Compliance Device for isochronous endpoint IN looped back to bulk IN.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Initiate an OUT transfer with wMaxPacketSize Burst.
5. Verify that the OUT transfer completes successfully.
6. Initiate an IN transfer to the bulk IN endpoint.
7. Verify that the IN transfer completes successfully.
8. Verify that the OUT data matches the IN data (data is the same and is in the same order).
9. Repeat for 10 iterations.

**Repetitions**

- Repeat with FS, HS, and SS USB Compliance Devices
- Repeat for transfer size of (wMaxPacketSize * Burst / 2) +1
- Repeat for all allowed max packets sizes for isochronous transfers
- Repeat for all allowed max burst size for HS isochronous transfers (1 to 3) and all allowed max burst size for SS isochronous transfers (1 to 16)
- Repeat for all allowed number of bursts (multi) for SS

**Case 6 – Isochronous streaming OUT**

1. If the USB Compliance Device is not operational, reconnect the USB Compliance Device behind the MA USB Hub, then enumerate.
2. Configure the Compliance Device for isochronous endpoint OUT looped back to bulk IN.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Stream continuous isochronous traffic of wMaxPacketSize * Burst for 30 minutes.
5. Verify that the Isochronous OUT transfer completed successfully.
6. Verify that the OUT data matches the bulk IN data (data is the same and is in the same order).
7. Repeat for 10 iterations.

**Repetitions**

- Repeat with FS, HS, and SS USB Compliance Devices
- Repeat for transfer size of (wMaxPacketSize * Burst / 2) +1
- Repeat for all allowed max packets sizes for isochronous transfers
- Repeat for all allowed max burst size for HS isochronous transfers (1 to 3) and all allowed max burst size for SS isochronous transfers (1 to 16)
- Repeat for all allowed number of bursts (multi) for SS

**Case 7 – Isochronous streaming IN**

1. If USB Compliance Device is not operational, reconnect the USB Compliance Device behind the MA USB Hub, then enumerate.
2. Configure the Compliance Device for isochronous endpoint IN looped back to bulk OUT.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Stream continuous isochronous traffic of wMaxPacketSize * Burst for 30 minutes.
5. Verify that the IN data matches the bulk OUT data (data is the same and is in the same order).
6. Repeat for 10 iterations.

**Repetitions**

- Repeat with FS, HS, and SS USB Compliance Devices
- Repeat for transfer size of (wMaxPacketSize * Burst / 2) +1
- Repeat for all allowed max packets sizes for isochronous transfers

- Repeat for all allowed max burst size for HS isochronous transfers (1 to 3) and all allowed max burst size for SS isochronous transfers (1 to 16)
- Repeat for all allowed number of bursts (multi) for SS

# 6 Test Descriptions for MA USB Device

*Note: Unless otherwise mentioned, the Compliance Host will indicate support for Link Sleep by including a Link Sleep Capability Descriptor with the Link Sleep Capable field set to 1, in a CapReq packet.*

## General Test Procedure

**Required Equipment:**

1. MA USB MA USB Sniffer
2. Packet parsing/analysis Software

The following test steps will be run in conjunction with the tests in this chapter on all MA USB packets generated by the DUT during other test procedures. Unless otherwise specified, the entire test will be recorded by the MA USB Sniffer and the trace saved in a form readable by the MA USB Analysis Software. Packets from the DUT will be identified by MAC address.

1. **For all packets (Common Header Fields):**
   a. Verify that packet is not greater than 64K. (6.1#1)
   b. Verify that Version field is 0. (6.2.1.1#1)
   c. Verify that Host field is 0 (reserved). (6.2.1.2#1)
   d. Verify that the fourth bit in the Flags field is 0. (6.2.1.2#4)
   e. Verify that the Type field is not 11b. (6.2.1.3#1)
   f. Verify that the Type and Subtype fields contain a value listed in Table 5 (6.2.1.3#2)
   g. Verify that the Length field is the length of the packet in Bytes. (6.2.1.4#1)
   h. Verify that the Status Code field is one of the values in Table 6. (6.2.1.8#1)
   i. If the Status Code field is set to a value other than 0 (SUCCESS), verify that the packet contains all fields defined for its subtype. (6.2.1.8#2)
   j. If the Status Code field is set to DROPPED_PACKET, verify that the Drop Notification field in the P-managed OUT Capabilities Descriptor is set to 1. (6.3.3.2#5)
2. **For Management packets only:**
   a. Verify that bits 18 through 31 of DWORD 2 are 0 (reserved). (6.3#1)
   b. Unless a packet is a CancelTransferReq/Resp, DevResetReq/Resp, SynchReq/Resp, or has the Device Address field set to 0xFF (broadcast), verify that the Dialog Token field increments by 1 for each new management packet carrying a request. (6.3.1.1#1)
   c. Verify that the Dialog Token field is set to 1 after reaching aMaxDialogToken. (6.3.1.1#1)
3. **For Data packets only:**
   a. Verify that Sequence Numbers increment by 1 for each new TransferResp packet in an IN transfer with no gaps in between. (5.4#12)
   b. Verify that if the EPS field is 0, the EP handle in the packet is not yet assigned to any endpoint. (6.5.1.1#2)
   c. Verify that if the EPS field is 2, the EP handle in the packet is assigned but inactive. (6.5.1.1#4)
   d. Verify that if the EPS value is 3, the EP handle in the packet is in the halted state. (6.5.1.1#5)
   e. If a packet type is not TransferResp, verify that the ARQ field is set to 0 (reserved). (6.5.1.2#2)
   f. If the NEG subfield in the T-Flags field is 1, verify that the device supports elastic buffer capability. (6.5.1.2#3)
   g. Bit 15 in the T-Flags field is 0. (6.5.1.2#8)
   h. If Enhanced SuperSpeed Stream Protocol is not supported, verify that the Stream ID field is 0 (reserved). (6.5.1.3#1)
4. **For TransferResp packets only:**
   a. Verify that the EP handle, Stream ID and Request ID fields are the same as the corresponding TransferReq packet that initiated the transfer. (5.4#10)
   b. If the Status Code field is set to 135 (TRANSFER_PENDING), verify that the Sequence Number is set to aInvalidSequenceNumber and the Remaining Size field is set to 0. (5.4#33)

c. If the Status Code field is set to 135 (TRANSFER_PENDING) and the packet carries no data, verify that the EoT field is 0. (5.4#40)

5. **For Isochronous Data packets only:**
   a. For an isochronous IN TransferResp, the Number of Headers field is set to the number of isochronous headers in the packet. (6.5.1.7#1)
   b. Unless the packet is an isochronous IN TransferResp or isochronous OUT TransferResp, verify that the Number of Headers field is 0. (6.5.1.7#3)
   c. Verify that the beginning of the packet consists of isochronous headers placed one after the other in increasing order of Segment Number. (5.10.1.1#1)

# TD 6.1 Management Packet Exchange Timing Test

**Assertions Covered**

1. 5.2.1.1#3

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. KG MA USB Host

**Test Steps**

For the following steps, verify that all management packet responses are received within aManagementResponseTime + 2 * aManagementChannelDelay from when the Compliance Host releases the corresponding request packet to the management channel. (5.2.1.1#3)

**Case 1 – Enumeration and Configuration**

1. Disconnect the DUT from the Host if previously connected.
2. Ensure that the following steps are captured by the MA USB Sniffer.
3. Connect the DUT to the Host.
4. Enumerate and configure the DUT.

**Case 2 – Device Operation**

1. Operate the DUT such that all functions are tested.

**Case 3 – Device Removal**

1. Instruct the Host to eject or otherwise remove the DUT.

# TD 6.2 Management retry Test

**Assertions Covered**

1. 5.2.1.1#9

**Required Equipment**

1. MA USB Compliance Host

**Test Steps**

**Case 1 – Enumeration and Configuration**

1. Disconnect the DUT from the Host if previously connected.
2. Connect the DUT to the Compliance Host.
3. Begin enumeration of the DUT.
4. Instruct the Compliance Host to send a second CapReq packet with the Retry field set to 1 after receiving a CapResp packet from the DUT.
5. Verify that the DUT responds to the second CapReq packet with a CapResp packet. (5.2.1.1#9)
6. Instruct the Compliance Host to finish enumeration and configuration. Verify that the DUT responds to each management request packet from the Compliance Host.

**Repetitions**

- Repeat with the following packet types replacing the CapReq/Resp:
  - USBDevHandleReq/Resp targeting integrated USB 2.0 device (if hub or non-SS device)
  - USBDevHandleReq/Resp targeting integrated USB 3.0 device (if SS hub or SS device)
  - EPHandleReq/Resp
  - ModifyEP0Req/Resp
  - UpdateDevReq/Resp
  - EPHandleReq/Resp

**Case 2 – Sleep/Wake Requests**

1. If the DUT does not support Link Sleep (Link Sleep Capability Descriptor either has Link Sleep Capable field set to 0 or is not present), instruct the Compliance Host to initiate EPInactivateReq/Resp packet exchange with the DUT followed by USBSuspendReq/Resp packet exchange for each integrated USB device.
2. Instruct the Compliance Host to send a SleepReq packet to the DUT.
3. Instruct the Compliance Host to send a second SleepReq packet with the Retry field set to 1 after receiving a SleepResp packet from the DUT.
4. Verify that the DUT responds to the second SleepReq packet with a SleepResp packet. (5.2.1.1#9)
5. Instruct the Compliance Host to send a WakeReq packet to the DUT.
6. Instruct the Compliance Host to send a second WakeReq packet with the Retry field set to 1 after receiving a WakeResp packet from the DUT.
7. Verify that the DUT responds to the second WakeReq packet with a WakeResp packet. (5.2.1.1#9)
8. If the DUT does not support Link Sleep (Link Sleep Capability Descriptor either has Link Sleep Capable field set to 0 or is not present), instruct the Compliance Host to initiate USBResumeReq/Resp packet exchange with the DUT followed by EPActivateReq/Resp packet exchange.

**Case 3 – Repeated Retries**

1. Instruct the Compliance Host to send a CapReq packet to the DUT.
2. Instruct the Compliance Host to send a second identical CapReq packet with the Retry field set to 1 after receiving a CapResp packet from the DUT.
3. Verify that the DUT responds to the second CapReq packet with a CapResp packet. (5.2.1.1#9)
4. Instruct the Compliance Host to send a third identical CapReq packet with the Retry field set to 1 after receiving the second CapResp packet from the DUT.
5. Verify that the DUT responds to the third CapReq packet with a CapResp packet. (5.2.1.1#9)

**Repetitions**

- Repeat with the following packet types replacing the CapReq/Resp:
  - USBDevResetReq/Resp targeting integrated USB 2.0 device (if hub or non-SS device)
  - USBDevResetReq/Resp targeting integrated USB 3.0 device (if SS hub or SS device)
  - PingReq/Resp

**Case 4 – Device Disconnection**

6. Reconnect, enumerate, and configure the DUT if needed.
7. Initiate ejection or removal of DUT from the Compliance Host.
8. Instruct the Compliance Host to send a CancelTransferReq packet to the DUT.
9. Instruct the Compliance Host to send a second CancelTransferReq packet with the Retry field set to 1 after receiving a CancelTransferResp packet from the DUT.
10. Verify that the DUT responds to the second CancelTransferReq packet with a CancelTransferResp packet. (5.2.1.1#9)
11. Complete removal of the DUT. Verify that the DUT responds to all MA USB packets.

**Repetitions**

- Repeat with the following packet types replacing the CancelTransferReq/Resp:
  - EPInactivateReq/Resp

- ClearTransfersReq/Resp
- EPHandleDeleteReq/Resp
- USBDevDisconnectReq/Resp targeting integrated USB 2.0 device (if hub or non-SS device)
- USBDevDisconnectReq/Resp targeting integrated USB 2.0 (if SS hub or SS device)

# TD 6.3 Data Packet Exchange Timing Test

**Assertions Covered**

1. 5.2.1.2#3

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. KG MA USB Host

**Test Steps**

1. Disconnect the DUT from the Host if previously connected.
2. Ensure that the following steps are captured by the MA USB Sniffer.
3. Connect the DUT to the Host.
4. Enumerate and configure the DUT.
5. Operate the DUT such that all functions are tested.
6. Verify that all TransferResp packets are received within aTransferResponseTime + 2 * aDataChannelDelay from when the corresponding request packet is released to the assigned data channel. (5.2.1.2#3)

# TD 6.4 Data Retry Test

**Assertions Covered**

1. 5.2.1.2#10

**Required Equipment**

1. MA USB Compliance Host

**Test Steps**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the Compliance Host.
    b. Enumerate and configure the DUT.
2. For IN endpoints:
    a. Instruct the Compliance Host to send an IN TransferReq packet with the Retry field set to 0.
    b. After receiving a TransferResp packet from the DUT, instruct the Compliance Host to resend the previous TransferReq with the Retry field set to 1.
    c. Verify that the DUT responds to the TransferReq packet. (5.2.1.2#10)
3. For OUT endpoints:
    a. Instruct the Compliance Host to send an OUT TransferReq packet with the Retry field set to 0 and the ARQ field set to 1.
    b. After receiving a TransferResp packet from the DUT, instruct the Compliance Host to resend the previous TransferReq with the Retry field set to 1.
    c. Verify that the device responds to the TransferReq packet. (5.2.1.2#10)

**Repetitions**

- Repeat for all bulk and interrupt endpoints on the DUT.

# TD 6.5 Isochronous Transfer Test

*Note: Only run this test if the DUT supports isochronous transfers.*

**Assertions Covered**

1. 5.10.1.1#3, 5.10.1.1#4, 5.10.1.1#5

**Required Equipment**

1. Either MA USB Compliance Host or KG MA USB Host and MA USB Sniffer

**Test Steps**

1. Disconnect the DUT from the Host/Compliance Host if previously connected.
2. Connect the DUT to the Host.
3. Enumerate and configure the DUT.
4. Record the value in the Isochronous Payload Alignment field of the CapResp from the DUT.
5. If using a KG MA USB Host, ensure that the following steps are captured by an MA USB Sniffer.
6. Operate the isochronous functions on the DUT.
7. If the Isochronous Payload Alignment field in the CapResp from the DUT is 1, verify that all packets have DWORD padding (Block Length is a multiple of 4). (5.10.1.1#3)
8. If the Isochronous Payload Alignment field in the CapResp from the DUT is 0, verify that all packets do not have DWORD padding (If Block Length is a multiple of 4, the first Byte of the corresponding isochronous data block is not 0). (5.10.1.1#4)
9. Verify that all isochronous headers are 4, 8, or 12 bytes. (5.10.1.1#5)
10. Verify that all isochronous headers are the same length. (5.10.1.1#5)
11. Verify that the isochronous functions complete successfully.

# TD 6.6 Ping Protocol Test

**Assertions Covered**

1. 5.2.2#1, 5.2.2#2, 5.2.2#4
2. 6.3.34#1, 6.3.34#2, 6.3.34#4, 6.3.35#1, 6.3.35#2, 6.3.35#3, 6.3.35#4

**Required Equipment**

1. MA USB Compliance Host

**Test Steps**

**Case 1 – Ping Response**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the Compliance Host.
    b. Enumerate and configure the DUT.
2. Instruct the Compliance Host to send a PingReq packet to the DUT with the Device Address field set to the address of the DUT and the SSID set to the Service Set.
3. Verify that the DUT responds with a PingResp packet. (6.3.35#1)
4. Verify that the PingResp packet has:
    a. The Device Address field set to the device address of the DUT. (5.2.2#1)
    b. Type field set to 0. (6.3.35#2)
    c. Subtype field set to 33. (6.3.35#2)
    d. Device Handle field is set to 0 (reserved). (6.3.35#3)
    e. Status Code field set to a value listed in Table 6. (6.3.35#4)
5. Instruct the Compliance Host to send a PingReq packet with the Device Address field set to 0xFF (broadcast) and the SSID set to the Service Set.
6. Verify that the DUT responds with a PingResp packet. (6.3.35#1)
7. Verify that the PingResp packet has:
    a. The Device Address field set to the device address of the DUT. (5.2.2#1)
    b. Type field set to 0. (6.3.35#2)
    c. Subtype field set to 33. (6.3.35#2)
    d. Device Handle field is set to 0 (reserved). (6.3.35#3)
    e. Status Code field set to a value listed in Table 6. (6.3.35#4)
8. Instruct the Compliance Host to send a PingReq packet with the Device Address field to the device address of the DUT and the SSID set to a value different than the Service Set.
9. Verify that the DUT does not respond. (5.2.2#1)

10. Instruct the Compliance Host to send a PingReq packet to the DUT with the Device Address field set to an address that is different than the address of the DUT and the SSID set to the Service Set.
11. Verify that the DUT does not respond. (5.2.2#1)
12. Repeat for 10 iterations.

**Case 2 – Ping Request**

1. Operate the DUT.
2. Instruct the Compliance Host to stop responding to MA USB packets from the DUT while a transfer is in progress.
3. If the DUT sends a PingReq packet, verify the following:
    a. The Device Address field is the same as the device address of the MA USB device. (5.2.2#2)
    b. The Device Address field is not set to 0xFF. (5.2.2#4)
    c. The Type field is set to 0. (6.3.34#1)
    d. The Subtype field is set to 32. (6.3.34#1)
    e. The Status Code field is set to 0 (SUCCESS). (6.3.34#2)
    f. The Device Handle field is set to 0 (reserved). (6.3.34#4)
4. Disconnect the DUT from the Compliance Host.

# TD 6.7 Endpoint Configuration Event Test

*Note: Only run this test if the DUT has one or more bulk and/or interrupt IN endpoints.*

**Assertions Covered**

1. 5.4#11

**Required Equipment**

1. MA USB Compliance Host

**Test Steps**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the Compliance Host.
    b. Enumerate and configure the DUT.
2. Operate the DUT.
3. Instruct the Compliance Host to send a ClearTransfersReq packet to an IN endpoint on the DUT.
4. Operate the DUT.
5. Verify that the first TransferResp packet sent by the DUT from the IN endpoint that received the ClearTransfersReq packet has the Sequence Number field set to 0. (5.4#11)

**Repetitions**

- Repeat for all IN endpoints on the DUT.

# TD 6.8 Invalid Request ID Test

**Assertions Covered**

1. 5.4#22, 5.4#23, 5.4#24, 5.4#25, 5.4#26, 5.5#19, 5.5#20, 5.5#21, 5.5#22, 5.5#23

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. MA USB Compliance Host

**Test Steps**

1. Disconnect the DUT from the Compliance Host if previously connected.
2. Connect the DUT to the Compliance Host.
3. Enumerate and configure the DUT.

**Case 1 – Bulk/Interrupt IN transfers**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:

   a. Connect/reconnect the DUT to the Compliance Host.
   b. Enumerate and configure the DUT.
2. Ensure that the following steps are captured by the MA USB Sniffer.
3. Instruct the Compliance Host to initiate a first IN transfer.
4. Verify that the transfer completes successfully.
5. Instruct the Compliance Host to initiate a second IN transfer where the first TransferReq packet has the Request ID field set to a value that is equal to the Request ID of the last TransferReq packet sent plus 2.
6. Verify that the DUT sends a null TransferResp packet in response to the TransferReq packet with the incorrect Request ID. (5.4#22)
7. Verify that the null TransferResp packet:
   a. Is received within aTransferResponseTime + 2 * aDataChannelDelay from when the corresponding request packet is released to the assigned data channel. (5.4#23)
   b. Has a Request ID field equal to the Request ID in the TransferReq packet minus 1 (expected Request ID). (5.4#24)
   c. Has the Status Code field set to 156 (MISSING_REQUEST_ID). (5.4#25)
8. Instruct the Compliance Host to send the remainder of the packets for the transfer using the Request ID from the DUT.
9. Verify that the transfer completes successfully.
10. Instruct the Compliance Host to initiate a third IN transfer where the first TransferReq packet has the Request ID field set to a value that is equal to the Request ID in the last TransferReq packet sent minus 1.
11. Verify that the DUT sends a null TransferResp packet in response to the TransferReq packet with the incorrect Request ID. (5.4#22)
12. Verify that the null TransferResp packet:
   a. Is received within aTransferResponseTime + 2 * aDataChannelDelay from when the corresponding request packet is released to the assigned data channel. (5.4#23)
   b. Has a Request ID field equal to the Request in the TransferReq packet plus 2 (expected Request ID). (5.4#24)
   c. Has the Status Code field set to 133 (INVALID_REQUEST). (5.4#26)
13. Instruct the Compliance Host to send the remainder of the packets for the transfer using the Request ID from the DUT.
14. Verify that the transfer completes successfully.

**Repetitions**

- Repeat for all bulk and interrupt IN endpoints on the DUT.

**Case 2 – OUT transfers**

1. Instruct the Compliance Host to initiate a first OUT transfer.
2. Verify that the transfer completes successfully.
3. Instruct the Compliance Host to initiate a second OUT transfer where the first TransferReq packet has the Request ID field set to a value that is equal to the Request ID of the last TransferReq packet sent plus 2
4. Verify that the DUT sends a null TransferResp packet in response to the TransferReq packet with the incorrect Request ID. (5.5#19)
5. Verify that the null TransferResp packet:
   a. Is received within aTransferResponseTime + 2 * aDataChannelDelay from when the corresponding request packet is released to the assigned data channel. (5.5#20)
   b. Has the Request ID and Sequence Number fields set to the Request ID and Sequence number in the last TransferReq packet received. (5.5#21)
   c. Has the Status Code field set to 156 (MISSING_REQUEST_ID). (5.5#22)
6. Instruct the Compliance Host to send the remainder of the packets for the transfer using the Request ID from the DUT.
7. Verify that the transfer completes successfully.
8. Instruct the Compliance Host to initiate a third IN transfer where the first TransferReq packet has the Request ID field set to a value that is equal to the Request ID in the last TransferReq packet sent minus 1.
9. Verify that the DUT sends a null TransferResp packet in response to the TransferReq packet with the incorrect Request ID. (5.5#19)
10. Verify that the null TransferResp packet:

   a. Is received within aTransferResponseTime + 2 * aDataChannelDelay from when the corresponding request packet is released to the assigned data channel. (5.5#20)
   b. Has the Request ID and Sequence Number fields set to the Request ID and Sequence number in the last TransferReq packet received. (5.5#21)
   c. Has the Status Code field set to 133 (INVALID_REQUEST). (5.5#23)
11. Instruct the Compliance Host to send the remainder of the packets for the transfer using the Request ID from the DUT.
12. Verify that the transfer completes successfully.

**Repetitions**

- Repeat for all bulk and interrupt OUT endpoints on the DUT.

## TD 6.9 Pending Transfer Test (Hubs Only)

**Assertions Covered**

1. 5.2.1.2#3, 5.4#31, 5.4#32, 5.4#33, 5.4#35, 5.4#42
2. 6.3.52#1, 6.3.52#2

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. KG MA USB Host
4. KG USB Device

**Test Steps**

1. Disconnect the DUT from the Host if previously connected.
2. Ensure that the following steps are captured by the MA USB Sniffer.
3. Connect the DUT to the Compliance Host without any devices connected downstream.
4. Start enumeration.
5. If the DUT sends an EPSetKeepAliveReq packet at any time during the test, verify that the EPSetKeepAliveReq packet that has:
   a. Type field set to 0. (6.3.52#1)
   b. Subtype field set to 51. (6.3.52#1)
   c. Status Code Field set to 0 (SUCCESS). (6.3.52#2)
6. Verify the following for the interrupt IN endpoint:
   a. Verify that the DUT sends a null TransferResp packet with no payload. (5.4#31)
   b. Verify that the TransferResp packet is received within aTransferResponseTime + 2 * aDataChannelDelay from when the corresponding request packet is released to the assigned data channel. (5.2.1.2#3)
   c. Verify that the null TransferResp packet has the Status Code Field set to 135 (TRANSFER_PENDING). (5.4#32)
   d. Verify that the null TransferResp packet has the Sequence Number set to 2^24 - 1 (aInvalidSequenceNumber). (5.4#33)
   e. Verify that the null TransferResp packet has the Remaining Size field set to 0 (reserved). (5.4#33)
   f. Verify that any TransferResp packets with the Status Code set to 135 (TRANSFER_PENDING) do not have the EoT field set to 1. (5.4#42)
7. Plug the USB Device in downstream of the DUT to trigger completion of status change transfer request.
8. Verify that the first TransferResp packet with payload sent by the DUT after the 135 (TRANSFER_PENDING) packet has the ARQ flag set to 1. (5.4#35)

## TD 6.10 Data OUT Transfer ARQ Test

*Note: Only run this test if the DUT has bulk and/or interrupt OUT endpoints*.

**Assertions Covered**

1. 5.2.1.2#3, 5.5#26

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. MA USB Compliance Host

**Test Steps**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the Compliance Host.
    b. Enumerate and configure the DUT.
2. Ensure that the following steps are captured by the MA USB Sniffer.
3. Operate the DUT such that an OUT transfer is initiated.
4. Instruct the Compliance Host to set the ARQ flag to 1 in each TransferReq packet it sends.
5. Verify that the DUT responds to each TransferReq packet with a TransferResp packet that has the same Request ID and Sequence Number as the TransferReq packet. (5.5#26)
6. Verify that each TransferResp packet is received within aTransferResponseTime + 2 * aDataChannelDelay from when the corresponding request packet is released to the assigned data channel. (5.2.1.2#3)

**Repetitions**

- Repeat for all bulk and interrupt OUT endpoints on the DUT.

# TD 6.11 Missing Sequence Number Test

*Note: Only run this test if the DUT has bulk and/or interrupt OUT endpoints.*

**Assertions Covered**

1. 5.5#33, 5.5#34

**Required Equipment**

1. MA USB Compliance Host

**Test Steps**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the Compliance Host.
    b. Enumerate and configure the DUT
2. Operate the DUT such that an OUT transfer is initiated.
3. Instruct the Compliance Host to skip a Sequence Number in the OUT data transfer.
4. Verify that the DUT sends a TransferResp packet in response to the TransferReq packet with the out-of-order sequence number within aTransferResponseTime of receiving the TransferReq packet. (5.5#33)
5. Verify that the TransferResp packet has the Request ID and Sequence Number fields set to the values the DUT expects next. (5.5#34)
6. Verify that the TransferResp packet has the Status Code field set to MISSING_SEQUENCE_NUMBER. (5.5#34)
7. Instruct the Compliance Host to send the remaining TransferReq packets starting with the Sequence Number in the TransferResp from the DUT.
8. Verify that the transfer completes successfully.

**Repetitions**

- Repeat for all bulk and interrupt OUT endpoints on the DUT.

# TD 6.12 Version Compatibility Test

**Assertions Covered**

1. None

**Required Equipment**

1. MA USB Compliance Host

**Test Steps**

1. Disconnect the DUT from the Host if previously connected.

2. Connect the DUT to the Compliance Host.
3. Start enumeration.
4. Instruct the Compliance Host to send a CapReq packet with the Version field set to 0001b.
5. Verify that the DUT responds with a CapResp packet that has the Version field set to 0000b.
6. Complete enumeration and configuration.
7. Verify MA USB operability by instructing the Compliance Host to send a PingReq packet with the Version field set to 000b to the DUT and verifying that the DUT responds with a PingResp packet.
8. Verify USB operability by instructing the Compliance Host to send a GetDescriptor request to the DUT for each integrated USB device and verifying that the DUT responds.

**Repetitions**

- Repeat with the Version field set to 1000b, 0110b, and 0010b.

# TD 6.13 Retry Bit Test

**Assertions Covered**

1. 5.2.1.2#4
2. 6.2.1.2#2

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. MA USB Compliance Host

**Test Steps**

**Case 1 – IN Transfers**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the Compliance Host.
    b. Enumerate and configure the DUT.
2. Ensure that the following steps are captured by the MA USB Sniffer.
3. Instruct the Compliance Host to issue a GetDeviceDescriptor request to the DUT.
4. If the DUT responds with a TransferResp packet that has the EoT field set to 1, the Retry bit set to 0, and Status Code set to 0 (SUCCESS). (6.2.1.2#2)
    a. Instruct the Compliance Host to not acknowledge the TransferResp packet.
    b. If the DUT retries the TransferResp, verify that the Retry bit is set to 1. (6.2.1.2#2)
        i. Verify that the DUT retried the TransferResp packet after aTransferRequestTimeout. (5.2.1.2#4)
        ii. Instruct the Compliance Host to acknowledge the retried TransferResp packet.
5. If the DUT sends a null TransferResp packet with the status code set to 135 (TRANSFER_PENDING) and the ARQ bit set to 1:
    a. Instruct the Compliance Host to not acknowledge the null TransferResp packet.
    b. Verify that the DUT resends the null TransferResp packet after aTransferRequestTimeout. (5.2.1.2#4)
    c. Instruct the Compliance Host to acknowledge the retried null TransferResp packet.
    d. Wait for the DUT to send a TransferResp packet with a Status Code that is set to a value other than 135 (TRANSFER_PENDING).
    e. Instruct the Compliance Host to NOT acknowledge the TransferResp packet.
    f. Verify that the DUT resends the TransferResp packet with the Retry bit set to 1. (6.2.1.2#2)
    g. Verify that the DUT reset the TransferResp packet after aTransferRequestTimeout. (5.2.1.2#4)
    h. Instruct the Compliance Host to acknowledge the retried TransferResp packet.
6. If the DUT sends a null TransferResp packet with the status code set to 135 (TRANSFER_PENDING) and the ARQ bit set to 0:
    a. Wait for the DUT to send a TransferResp packet with a Status Code that is set to a value other than 135 (TRANSFER_PENDING).
    b. Instruct the Compliance Host to NOT acknowledge the TransferResp packet.
    c. Verify that the DUT resends the TransferResp packet with the Retry bit set to 1. (6.2.1.2#2)

d. Verify that the DUT resent the TransferResp packet after aTransferRequestTimeout. (5.2.1.2#4)
e. Instruct the Compliance Host to acknowledge the retried TransferResp packet.

**Repetitions**

- Repeat for all bulk and interrupt IN endpoints.

**Case 2 – OUT Transfers**

1. Instruct the Compliance Host to initiate an OUT transfer.
2. Verify that the DUT responds to the end of the transfer with a TransferResp packet that has the EoT field set to 1 and the Retry bit set to 0. (6.2.1.2#2)
3. Instruct the Compliance Host to not acknowledge the TransferResp packet.
4. Verify that the DUT resends the TransferResp packet with the Retry bit set to 1. (6.2.1.2#2)
5. Verify that the DUT resent the TransferResp packet after aTransferRequestTimeout. (5.2.1.2#4)
6. Instruct the Compliance Host to acknowledge the TransferResp packet.

**Repetitions**

- Repeat for all bulk and interrupt OUT endpoints.

# TD 6.14 Incorrect Address Test

**Assertions Covered**

1. 6.2.1.6#1

**Required Equipment**

1. MA USB Compliance Host

**Test Steps**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the Compliance Host.
    b. Enumerate and configure the DUT.
2. Instruct the Compliance Host to send a CapReq packet to the DUT with the MA USB Device Address field set to the assigned address.
3. Verify that the DUT responds with a CapResp packet.
4. Instruct the Compliance Host to send a CapReq packet to the DUT with the MA USB Device Address field set to a value other than the assigned address.
5. Verify that the DUT does not respond to the CapReq packet. (6.2.1.6#1)
6. Instruct the Compliance Host to send a TransferReq packet to the DUT that has the MA USB Device Address field set to a value other than the assigned address.
7. Verify that the DUT does not respond to the TransferReq packet.
8. Instruct the Compliance Host to send a USB GetDescriptor request to the DUT with the MA USB Device Address field set to a value other than the assigned address.
9. Verify that the DUT does not respond. (6.2.1.6#1)
10. Verify MA USB operability by instructing the Compliance Host to send a PingReq packet to the DUT and verifying that the DUT responds with a PingResp packet.
11. Verify USB operability by instructing the Compliance Host to send a USB GetDescriptor request for each integrated USB device to the DUT and verifying that the DUT responds.

# TD 6.15 Dialog Token Test

**Assertions Covered**

1. 6.3.1.1#3, 6.3.18#3

**Required Equipment**

1. MA USB Compliance Host

**Test Steps**

1. Disconnect the DUT from the Host if previously connected.
2. Connect the DUT to the Compliance Host.
3. Start enumeration.
4. Instruct the Compliance Host to send a SetUSBDevAddressReq packet with a dialog token that is not in sequence with the dialog tokens in previously sent management packets.
5. Verify that the DUT does not respond to the SetUSBDevAddressReq packet. (6.3.1.1#3)
6. Instruct the Compliance Host to send a SetUSBDevAddressReq packet with an in-sequence dialog token.
7. Verify that the DUT sends a SetUSBDevAddrResp packet in response.
8. Complete enumeration and configuration.
9. Instruct the Compliance Host to send a DevResetReq packet with an out-of-order dialog token and the SSID set to the same value as previous packets.
10. Verify that the DUT sends a DevResetResp packet in response. (6.3.18#3)
11. Complete enumeration and configuration.
12. Verify MA USB operability by instructing the Compliance Host to send a PingReq packet to the DUT and verifying that the DUT responds with a PingResp packet.
13. Verify USB operability by instructing the Compliance Host to send a GetDescriptor request to the DUT and verifying that the DUT responds.

# TD 6.16 Enumeration and Device Initialization Test

**Assertions Covered**

1. 6.3.3#1, 6.3.3#2, 6.3.3#3, 6.3.3#4, 6.3.3#5, 6.3.3#7, 6.3.3#12, 6.3.3#13, 6.3.3#14, 6.3.3#17, 6.3.3#18, 6.3.3#19, 6.3.3#20, 6.3.3#21, 6.3.3#22, 6.3.3#23, 6.3.3#24, 6.3.3#25, 6.3.3#26, 6.3.5#1, 6.3.5#2, 6.3.5#3, 6.3.5#4, 6.3.5#5, 6.3.5#6, 6.3.7#1, 6.3.7#2, 6.3.7#4, 6.3.7#5, 6.3.7#6, 6.3.7#7, 6.3.7#8, 6.3.7#10, 6.3.7#11, 6.3.7#14, 6.3.7#16, 6.3.7#17, 6.3.7#18, 6.3.7#19, 6.3.19#1, 6.3.19#2, 6.3.19#3, 6.3.19#4, 6.3.21#1, 6.3.21#2, 6.3.21#4, 6.3.21#8, 6.3.23#1, 6.3.23#2, 6.3.23#4, 6.3.23#6, 6.3.25#1, 6.3.25#2, 6.3.25#4
2. 7.3.2.5#2

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. KG MA USB Host

**Test Steps**

1. Disconnect the DUT from the Host if previously connected.
2. Ensure that the following steps are captured by the MA USB Sniffer.
3. Connect the DUT to the MA USB Host.
4. Start enumeration.
5. Verify that the DUT sends a DevResetResp packet in response to a DevResetReq packet. (6.3.19#1)
6. Verify that the DevResetResp packet has:
    a. Type field set to 0. (6.3.19#2)
    b. Subtype field set to 17. (6.3.19#2)
    c. Device Handle field set to 0 (reserved). (6.3.19#3)
    d. Dialog Token field set to 0 (reserved). (6.3.19#3)
    e. Status Code field set to indicate if request was successfully completed. (6.3.19#4)
7. Verify that the DUT sends a CapResp packet in response to a CapReq packet. (6.3.3#1)
8. Verify that the CapResp has:
    a. Device Handle field set to 0 (reserved). (6.3.3#2)
    b. Type field set to 0. (6.3.3#3)
    c. Subtype field set to 1. (6.3.3#3)
    d. Status Code field set to a value listed in Table 6. (6.3.3#4)
    e. Number of Endpoint field set to the maximum number of endpoints for which the DUT can track state as indicated by the manufacturer (see MA USB Compliance Checklist, row F22). (6.3.3#5)
    f. Number of Devices field set to the number of USB devices the DUT can manage as indicated by the DUT manufacturer (see MA USB Compliance Checklist, row F23). (6.3.3#7)

   g. Device Type field that is less than or equal to 2. (6.3.3#12)
   h. Descriptors count field set to the total number of MA USB Device Capabilities Descriptors present. (6.3.3#13)
   i. Descriptors Length field set to the total size of MA Device Capabilities descriptors in bytes. (6.3.3#14)
   j. Number of Outstanding Management Requests field is set to a value that is at least double the Number of Endpoints field plus double the Number of Devices field. (6.3.3#17)
   k. Bits 28 through 31 of DWORD 5 set to 0 (reserved). (6.3.3#18)
9. Verify that any MA Device Capability Descriptors have:
   a. Length field set to the length of the descriptor in bytes. (6.3.3#19)
   b. MA Device Capability Type field set to a value listed in Table 14. (6.3.3#20)
10. If the Device Type field in the CapResp packet is set to 0 (not an MA USB hub) or 2 (MA USB hub with an integrated USB 3.1 hub), verify that a Speed Capability descriptor is present. (6.3.3#21)
11. If the DUT supports any of the optional capabilities related to P-Managed OUT transfers (see MA USB Compliance Checklist, row F12), verify that a P-Managed OUT Capabilities descriptor is present. (6.3.3#22)
12. If the Device Type field in the CapResp packet is greater than 0 or the DUT supports isochronous endpoints, verify that an isochronous Capabilities descriptor is present. (6.3.3#23)
13. If the Device Type field in the CapResp packet is greater than 0 or the DUT supports isochronous endpoints, verify that a Synchronization Capabilities descriptor is present. (6.3.3#24)
14. If the DUT supports Container ID descriptors (see MA USB Compliance Checklist, row F13), a Container ID Capability descriptor is present. (6.3.3#25)
15. If Link Sleep is supported, verify that a Link Sleep Capability descriptor is present. (6.3.3#26)
16. Verify that the DUT sends a USBDevHandleResp packet in response to a USBDevHandleReq packet. (6.3.5#1)
17. Verify that USBDevHandleResp packet has:
   a. Type field set to 0. (6.3.5#2)
   b. Subtype field set to 3. (6.3.5#2)
   c. Device Handle field set to 0 (reserved). (6.3.5#3)
   d. Status Code field set to indicate if request was successful or not. (6.3.5#4)
   e. USB Device Handle field set to handle of the USB device. (6.3.5#5)
   f. Bits 16 through 31 of DWORD 3 set to 0 (reserved). (6.3.5#6)
18. Verify that the DUT sends an EPHandleResp packet in response to an EPHandleReq packet. (6.3.7#1)
19. Verify that the EPHandleResp packet has:
   a. Type field set to 0. (6.3.7#2)
   b. Subtype field set to 0. (6.3.7#2)
   c. Status Code field set to indicate if request was successful or not. (6.3.7#4)
   d. Status Code field set to 0 (SUCCESS) if at least one EP descriptor has the Valid bit set to 0. (6.3.7#5)
   e. Number of MA USB EP Descriptors field that is the same as the Number of MA USB EP Descriptors in the corresponding EPHandleReq packet. (6.3.7#6)
   f. Bits 5 through 31 set to 0 (reserved). (6.3.7#7)
   g. Any EP descriptors present are inserted in the same order as the corresponding EP descriptors in the EPHandleReq packet. (6.3.7#8)
   h. Any EP descriptor in the packet has:
      i. Bits 20 through 31 of DWORD 0 set to 0 (reserved). (6.3.7#14)
      ii. Bits 16 through 31 of DWORD 1 set to 0 (reserved). (6.3.7#16)
      iii. If the EP descriptor indicates that the endpoint is an IN endpoint, the Buffer Size Field is set to 0 (reserved). (6.3.7#17)
      iv. If the EP descriptor indicates that the endpoint is not an isochronous, the Isochronous Programming Delay field is set to 0 (reserved). (6.3.7#18)
      v. If the EP descriptor indicates that the endpoint is not isochronous, the Isochronous Response Delay field is set to 0 (reserved). (6.3.7#19)
20. Verify that the DUT sends a SetUSBDevAddrResp in response to a SetUSBDevAddrReq packet. (6.3.23#1)
21. Verify that the SetUSBDevAddrResp packet has:
   a. Type field set to 0. (6.3.23#2)
   b. Subtype field set to 21. (6.3.23#2)
   c. Status Code field set to indicate if request was successful or not. (6.3.23#4)

    d. Bits 7 through 31 of DWORD 3 set to 0 (reserved). (6.3.23#6)
22. Verify that a ModifyEP0Req management request follows a SetUSBDevAddrReq packet.
23. Verify that the DUT sends a ModifyEP0Resp packet. (6.3.21#1)
24. Verify that the ModifyEP0Resp packet has:
    a. Type field set to 0. (6.3.21#2)
    b. Subtype field set to 19. (6.3.21#2)
    c. Status Code field set to indicate if request was successful or not. (6.3.21#4)
    d. Bits 16 through 31 are set to 0 (reserved). (6.3.21#8)
25. Verify that the DUT sends an UpdateDevResp packet in response to an UpdateDevReq packet. (6.3.25#1, 7.3.2.5#2)
26. Verify that the UpdateDevResp packet has:
    a. Type field set to 0. (6.3.25#2)
    b. Subtype field set to 23. (6.3.25#2)
    c. Status Code field set to indicate if request was successful or not. (6.3.25#4)

## TD 6.16.1 Hub Enumeration Test

*Run this test in conjunction with TD 6.16 if the DUT is a hub.*

**Test Steps**

1. Verify that the Number of Endpoints field in a CapResp is at least 16. (6.3.3#6)
2. Verify that the Device Type field is set to 1 for USB 2.0 hubs. (6.3.3#11)
3. Verify that the Device Type field is set to 2 for USB 3.1 hubs. (6.3.3#11)
4. Verify that if the Device Type field is set to 2, the CapResp packet carries a Speed Capability Descriptor. (6.3.3.1#1)

## TD 6.16.2 Non-Hub Enumeration Test

*Run this test in conjunction with TD 6.16 if the DUT is not a hub.*

**Test Steps**

1. Verify that the Number of Devices field in a CapResp packet is not greater than 1. (6.3.3#8)
2. Verify that the Device Type field in a CapResp packet is 0. (6.3.3#10)

## TD 6.16.3 Speed Capability Descriptor Test

*Run this test in conjunction with TD 6.16 if a Speed Capability Descriptor is present.*

**Test Steps**

1. If a Speed Capability Descriptor is present:
    a. Verify that the Length field is 4. (6.3.3.1#2)
    b. Verify that the MA Capability Type field is 0. (6.3.3.1#3)
    c. Verify that bits 16 through 19 are set to 0 (reserved). (6.3.3.1#4)
    d. Verify that the Speed field indicates the speed of the USB device behind the DUT. (6.3.3.1#5)
    e. Verify that the Speed field does not contain a value greater than 5. (6.3.3.1#6)
    f. Verify that bits 24 through 27 are set to 0 (reserved). (6.3.3.1#7)
    g. If the Speed field is set to SuperSpeed or SuperSpeedPlus, verify that the Lane Speed Exponent field indicates the lane speed exponent of the USB device. (6.3.3.1#8)
    h. If the Speed field is not set to SuperSpeed or SuperSpeedPlus, verify that the Lane Speed Exponent field is 0 (reserved). (6.3.3.1#8)
    i. Verify that bits 31 and 32 are 0 reserved). (6.3.3.1#9)

## TD 6.16.4 P-Managed OUT Capability Descriptor Test

*Run this test in conjunction with TD 6.16 if a P-Managed OUT Capability Descriptor is present.*

**Test Steps**

1. If a P-Managed OUT Capability descriptor is present:
    a. Verify that the length field is set to 3. (6.3.3.2#1)
    b. Verify that the MA Capability field is set to 1. (6.3.3.2#2)
    c. If the DUT vendor has indicated support for Elastic Buffer Capability (see MA USB Compliance Checklist, row F5), verify that the Elastic Buffer Capability field is set to 1. (6.3.3.2#3)
    d. If the DUT vendor has indicated no support for Elastic Buffer Capability (see MA USB Compliance Checklist, row F5), verify that the Elastic Buffer Capability field is set to 0. (6.3.3.2#3)
    e. If a DUT can return a DROPPED_PACKET status (see MA USB Compliance Checklist, row F14), verify that the Drop Notification Capability field is set to 1. (6.3.3.2#4)
    f. If a DUT cannot return a DROPPED_PACKET status (see MA USB Compliance Checklist, row F14), verify that the Drop Notification Capability field is set to 0. (6.3.3.2#4)
    g. Verify that bits 18 through 23 are set to 0 (reserved). (6.3.3.2#6)

## TD 6.16.5 Isochronous Capabilities Descriptor Test

*Run this test in conjunction with TD 6.16 if an Isochronous Capabilities Descriptor is present.*

**Test Steps**

1. If an Isochronous Capabilities descriptor is present:
    a. Verify that the length field is set to 3. (6.3.3.3#1)
    b. Verify that the MA Capability field is set to 2. (6.3.3.3#2)
    c. If the DUT vendor has indicated that DWORD-aligned isochronous payload is required (see MA USB Compliance Checklist, row F9), verify that the Isochronous Payload Alignment field is 1. (6.3.3.3#3)
    d. If the DUT vendor has indicated that DWORD-aligned isochronous payload is not required (see MA USB Compliance Checklist, row F9), verify that the Isochronous Payload Alignment field is 0. (6.3.3.3#3)
    e. Verify that bits 17 through 23 are set to 0 (reserved). (6.3.3.3#4)

## TD 6.16.6 Synchronization Capabilities Descriptor Test

*Run this test in conjunction with TD 6.16 if a Synchronization Capabilities Descriptor is present.*

**Test Steps**

1. If a Synchronization Capabilities descriptor is present:
    a. Verify that the Length field is set to 3. (6.3.3.4#1)
    b. Verify that the MA Capability Type field is set to 3. (6.3.3.4#2)
    c. If the DUT vendor has indicated that the DUT has access to synchronized Media Time (see MA USB Compliance Checklist, row F10), verify that the Media Time field is 1. (6.3.3.4#3)
    d. If the DUT vendor has indicated that the DUT does not have access to synchronized Media Time (see MA USB Compliance Checklist, row F10), verify that the Media Time field is 0. (6.3.3.4#3)
    e. If the DUT vendor has indicated that the DUT needs to receive MA USB timestamps regardless of endpoint configuration (see MA USB Compliance Checklist, row F11), verify that the Timestamp Request field is set to 1. (6.3.3.4#4)
    f. If the DUT vendor has indicated that the DUT needs to receive MA USB timestamps only if isochronous endpoints are selected (see MA USB Compliance Checklist, row F11), verify that the Timestamp Request field is set to 0. (6.3.3.4#4)
    g. If the DUT is a hub, verify that the Timestamp Request field is set to 1. (6.3.3.4#5)
    h. Verify that bits 18 through 23 are set to 0 (reserved). (6.3.3.4#6)

# TD 6.16.7 Container ID Capability Descriptor Test

*Run this test in conjunction with TD 6.16 if a Container ID Capability Descriptor is present.*

**Test Steps**

1. If a Container ID capability descriptor is present:
    a. Verify that the Length field is set to 18. (6.3.3.5#1)
    b. Verify that the MA Capability Field is set to 4. (6.3.3.5#2)
    c. Verify that the Container ID field is set to the Container ID value. (6.3.3.5#3)

# TD 6.16.8 Link Sleep Capability Descriptor Test

*Run this test in conjunction with TD 6.16 if a Link Sleep Capability Descriptor is present.*

**Test Steps**

1. If a Link Sleep Capability descriptor is present:
    a. Verify that the Length field indicates the length of the descriptor in Bytes and is set to 3. (6.3.3.6#1)
    b. Verify that the MA Device Capability Type field is set to 5. (6.3.3.6#2)
    c. If the DUT vendor has indicated support for Link Sleep (see MA USB Compliance Checklist, row F4), verify that the Link Sleep Capable field is 1. (6.3.3.6#3)
    d. If the DUT vendor has not indicated support for Link Sleep (see MA USB Compliance Checklist, row F4), verify that the Link Sleep Capable field is 0. (6.3.3.6#3)
    e. Verify that bits 17 through 23 are set to 0 (reserved). (6.3.3.6#4)

# TD 6.17 Host-Initiated Session Transition Test

**Assertions Covered**

1. 6.3.9#1, 6.3.9#2, 6.3.9#3, 6.3.9#5, 6.3.9#6, 6.3.9#7, 6.3.9#8, 6.3.11#1, 6.3.11#2, 6.3.11#4, 6.3.11#5, 6.3.11#6, 6.3.11#7, 6.3.11#8, 6.3.29#1, 6.3.29#2, 6.3.29#3, 6.3.29#4, 6.3.31#1, 6.3.31#2, 6.3.31#3, 6.3.31#4, 6.3.57#1, 6.3.57#2, 6.3.57#3, 6.3.57#4, 6.3.57#5, 6.3.57#6, 6.3.59#1, 6.3.59#2, 6.3.59#3, 6.3.59#4
2. 8.1.1.3#2

**Required Equipment**

1. MA USB Compliance Host

**Test Steps**

**Case 1 – Link Sleep Supported**

*Note: Only run Case 1 if DUT supports Link Sleep, otherwise go to Case 2.*

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the Compliance Host.
    b. Enumerate and configure the DUT.
2. Instruct the Compliance Host to send a SleepReq packet to the DUT.
3. Verify that the DUT sends a SleepResp packet in response. (6.3.57#1)
4. Verify that the SleepResp packet has:
    a. Type field set to 0. (6.3.57#2)
    b. Subtype field set to 55. (6.3.57#2)
    c. Device Handle field set to 0 (reserved). (6.3.57#3)
    d. Status Code field set to a value listed in Table 6. (6.3.57#4)
    e. If Status Code is 0 (SUCCESS), Management Request Timeout field is less than or equal to the corresponding SleepReq packet. (6.3.57#5)
    f. If Status Code is 0 (SUCCESS), Data Request Timeout field is less than or equal to the corresponding SleepReq packet. (6.3.57#6)
5. If the DUT sends a SleepResp with the Status Code field set to 155 (REQUEST_DENIED):

   a. Instruct the Compliance Host to send a SleepReq packet with the Management Request Timeout and Data Request Timeout fields set to the same values as the SleepResp from the DUT.
   b. Verify that the DUT sends another SleepResp packet with the Status Code field set to 0 (SUCCESS).
6. Instruct the Compliance Host to send a WakeReq packet to the DUT.
7. Verify that the DUT responds with a WakeResp packet. (6.3.59#1)
8. Verify that the WakeResp packet has:
   a. Type field set to 0. (6.3.59#2)
   b. Subtype field set to 57. (6.3.59#2)
   c. Device Handle field set to 0 (reserved). (6.3.59#3)
   d. Status Code field set to 0 (SUCCESS). (6.3.59#4)
9. Verify that the DUT is in Session Active by instructing the Compliance Host to send a PingReq packet to the DUT and verifying that the DUT responds with a PingResp packet. (8.1.1.3#2)

**Case 2 – Link Sleep Not Supported**

1. Disconnect the DUT from the Compliance Host.
2. Instruct the Compliance Host to not support Link Sleep in the CapReq packet sent to the DUT during enumeration.
3. Connect the DUT to the Compliance Host, then enumerate and configure.
4. Instruct the Compliance Host to send an EPInactivateReq packet to inactivate all EP Handles on the DUT except EP0.
5. Verify that the DUT sends an EPInactivateResp packet. (6.3.11#1)
6. Verify that the EPInactivateResp has:
   a. Type field set to 0. (6.3.11#2)
   b. Subtype field set to 9. (6.3.11#2)
   c. Status Code field set to a value listed in Table 6. (6.3.11#4)
   d. Number of EP Handles with Error field that is equal to the number of EP handles entries in the EP Handle List field. (6.3.11#5)
   e. Number of EP Handles with Error field set to 0 if the Status Code field is 0 (SUCCESS). (6.3.11#6)
   f. Bits 5 through 31 of DWORD 3 set to 0 (reserved). (6.3.11#7)
   g. EP Handle List field that carries a list of the EP handles that failed and is concatenated in 16-bit increments. (6.3.11#8)
7. Instruct the Compliance Host to send a USBSuspendReq packet for each integrated USB device.
8. Verify that the DUT sends a USBSuspendResp packet in response to each USBSuspendReq packet. (6.3.29#1)
9. Verify that each USBSuspendResp packet has:
   a. Type field set to 0. (6.3.29#2)
   b. Subtype field set to 27. (6.3.29#2)
   c. Device Handle field set to the same value as the Device handle field in the corresponding USBSuspendReq packet. (6.3.29#3)
   d. Status Code field set to a value listed in Table 6. (6.3.29#4)
10. Instruct the Compliance Host to send a SleepReq packet.
11. Verify that the DUT sends a SleepResp packet in response. (6.3.57#1)
12. Verify that the SleepResp packet has:
   a. Type field set to 0. (6.3.57#2)
   b. Subtype field set to 55. (6.3.57#2)
   c. Device Handle field set to 0 (reserved). (6.3.57#3)
   d. Status Code field set to a value listed in Table 6. (6.3.57#4)
   e. If Status Code is 0 (SUCCESS), Management Request Timeout field is less than or equal to the corresponding SleepReq packet. (6.3.57#5)
   f. If Status Code is 0 (SUCCESS), Data Request Timeout field is less than or equal to the corresponding SleepReq packet. (6.3.57#6)
13. If the DUT sends a SleepResp with the Status Code field set to 155 (REQUEST_DENIED):
   a. Instruct the Compliance Host to send a SleepReq packet with the Management Request Timeout and Data Request Timeout fields set to the same values as the SleepResp from the DUT.
   b. Verify that the DUT sends another SleepResp packet with the Status Code field set to 0 (SUCCESS).
14. Instruct the Compliance Host to send a WakeReq packet to the DUT.

15. Verify that the DUT responds with a WakeResp packet. (6.3.59#1)
16. Verify that the WakeResp packet has:
    a. Type field set to 0. (6.3.59#2)
    b. Subtype field set to 57. (6.3.59#2)
    c. Device Handle field set to 0 (reserved). (6.3.59#3)
    d. Status Code field set to 0 (SUCCESS). (6.3.59#4)
17. Instruct the Compliance Host to send a USBResumeReq packet to the DUT for each integrated USB device.
18. Verify that the DUT responds to each USBResumeReq packet with a USBResumeResp packet. (6.3.31#1)
19. Verify that each USBResumeResp packet has:
    a. Type field set to 0. (6.3.31#2)
    b. Subtype field set to 29. (6.3.31#2)
    c. Device Handle field set to the same value as the Device handle field in the corresponding USBResumeReq packet. (6.3.31#3)
    d. Status Code field set to a value listed in Table 6. (6.3.31#4)
20. Instruct the Compliance Host to send an EPActivateReq packet to the DUT.
21. Verify that the DUT responds with an EPActivateResp packet. (6.3.9#1)
22. Verify that the EPActivateResp packet has:
    a. Type field set to 0. (6.3.9#2)
    b. Subtype field set to 7. (6.3.9#2)
    c. Status Code field indicates whether or not the request was successful. (6.3.9#3)
    d. Number of EP Handles with Error field that is equal to the number of EP handles entries in the EP Handle List field. (6.3.9#5)
    e. Number of EP Handles with Error field set to 0 if the Status Code field is 0 (SUCCESS). (6.3.9#6)
    f. Bits 5 through 31 of DWORD 3 set to 0 (reserved). (6.3.9#7)
    g. EP Handle List field that carries a list of the EP handles that failed and is concatenated in 16-bit increments. (6.3.9#8)
    h. Verify that the DUT is in Session Active by instructing the Compliance Host to send a PingReq packet to the DUT and verifying that the DUT responds with a PingResp packet. (8.1.1.3#2)

# TD 6.18 Endpoint Reset Test

**Assertions Covered**

1. 6.3.13#1, 6.3.13#2, 6.3.13#4, 6.3.13#5, 6.3.13#6, 6.3.13#7, 6.3.13#8
2. 7.2.3#3

**Required Equipment**

1. MA USB Compliance Host

**Test Steps**

**Case 1 – Single Endpoint Target**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the Compliance Host.
    b. Enumerate and configure the DUT.
2. Instruct the Compliance Host to send an EPResetReq packet to the DUT that targets a single endpoint on the DUT.
3. Verify that the DUT sends an EPResetResp packet in response to the EPResetReq packet. (6.3.13#1)
4. Verify that the EPResetResp packet has:
    a. Type field set to 0. (6.3.13#2)
    b. Subtype field set to 11. (6.3.13#2)
    c. Status Code field set to a value listed in Table 6. (6.3.13#4)
    d. Number of EP Handles with Error field set to the number of EP handles included in the EP Handle List field. (6.3.13#5)
    e. Number of EP Handles with Error field set to 0 if the Status Code field is 0 (SUCCESS). (6.3.13#6)
    f. Bits 5 through 31 of DWORD 3 set to 0 (reserved). (6.3.13#7)

g. EP Handle List field that carries the number of EP handles indicated in the Number of EP Handles with Error field concatenated in 16-bit increments. (6.3.13#8)

5. Instruct the Compliance Host to send a TransferReq packet to the endpoint that was targeted in the EPResetReq packet.
6. Verify that the DUT sends a TransferResp packet with the Status Code field set to INVALID_EP_HANDLE_STATE. (7.2.3#3)

**Repetitions**

- Repeat for all bulk and interrupt endpoints on the DUT.

**Case 2 – Multiple Endpoint Target**

1. Disconnect, then reconnect the DUT to the Compliance Host.
2. Enumerate and configure the DUT.
3. Instruct the Compliance Host to send an EPResetReq packet to the DUT that targets all bulk and interrupt endpoints on the DUT.
4. Verify that the DUT sends an EPResetResp packet in response to the EPResetReq packet. (6.3.13#1)
5. Verify that the EPResetResp packet has:
    a. Type field set to 0. (6.3.13#2)
    b. Subtype field set to 11. (6.3.13#2)
    c. Status Code field set to a value listed in Table 6. (6.3.13#4)
    d. Number of EP Handles with Error field set to the number of EP handles included in the EP Handle List field. (6.3.13#5)
    e. Number of EP Handles with Error field set to 0 if the Status Code field is 0 (SUCCESS). (6.3.13#6)
    f. Bits 5 through 31 of DWORD 3 set to 0 (reserved). (6.3.13#7)
    g. EP Handle List field that carries the number of EP handles indicated in the Number of EP Handles with Error field concatenated in 16-bit increments. (6.3.13#8)
6. For each bulk and interrupt endpoint targeted in the EPResetReq packet:
    a. Instruct the Compliance Host to send a TransferReq packet to the endpoint that was targeted in the EPResetReq packet.
    b. Verify that the DUT sends a TransferResp packet with the Status Code field set to INVALID_EP_HANDLE_STATE. (7.2.3#3)

# TD 6.19 Reserved

***Reserved for Future Use***

# TD 6.20 Session Tear Down Test

**Assertions Covered**

1. 6.3.15#1, 6.3.15#2, 6.3.15#4, 6.3.15#5, 6.3.15#6, 6.3.15#7, 6.3.15#8, 6.3.15#11, 6.3.15#12, 6.3.15#13, 6.3.17#1, 6.3.17#2, 6.3.17#4, 6.3.17#5, 6.3.17#6, 6.3.27#1, 6.3.27#2, 6.3.27#4, 6.3.38#1, 6.3.38#2, 6.3.38#3, 6.3.43#1, 6.3.43#2, 6.3.43#4, 6.3.43#5, 6.3.43#7, 6.3.43#8, 6.3.43#10, 6.3.43#12, 6.3.43#13, 6.3.43#14, 6.3.43#15

**Required Equipment**

1. MA USB Compliance Host

**Test Steps**

**Case 1 – Host-Initiated Session Tear Down (Alternative 1)**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the Compliance Host.
    b. Enumerate and configure the DUT.
2. Instruct the Compliance Host to send a CancelTransferReq packet to the DUT.
3. Verify that the DUT sends a CancelTransferResp packet in response. (6.3.43#1)
4. Verify that the CancelTransferResp packet has:

a. Type field set to 0. (6.3.43#2)
b. Subtype field set to 41. (6.3.43#2)
c. Status Code field set to a value in Table 6. (6.3.43#4)
d. Dialog Token field set to 0. (6.3.43#5)
e. For endpoints that are not Enhanced SS Bulk endpoints that support the Enhanced SS Stream Protocol, Stream ID field set to 0 (reserved). (6.3.43#7)
f. Request ID field that matches the assigned request ID recorded in step 5. (6.3.43#8)
g. Bits 10 through 31 of DWORD 4 set to 0 (reserved). (6.3.43#10)
h. For an IN transfer, Delivered Sequence Number field set to 0 (reserved). (6.3.43#12)
i. Bits 24 through 31 of DWORD 5 set to 0 (reserved). (6.3.43#13)
j. If Cancellation Status field is not 2, Delivered Byte Offset field is set to 0 (reserved). (6.3.43#15)
k. For an IN transfer Delivered Byte Offset field is 0 (reserved). (6.3.43#15)

5. Instruct the Compliance Host to send an EPInactivateReq to the DUT that targets all endpoints except EP0.
6. Verify that the DUT sends and EPInactivateResp packet. (6.3.11#1)
7. Verify that the EPInactivateResp has:
    a. Type field set to 0. (6.3.11#2)
    b. Subtype field set to 9. (6.3.11#2)
    c. Status Code field set to a value listed in Table 6. (6.3.11#4)
    d. Number of EP Handles with Error field that is equal to the number of EP handles entries in the EP Handle List field. (6.3.11#5)
    e. Number of EP Handles with Error field set to 0 if the Status Code field is 0 (SUCCESS). (6.3.11#6)
    f. Bits 5 through 31 of DWORD 3 set to 0 (reserved). (6.3.11#7)
    g. EP Handle List field that carries a list of the EP handles that failed and is concatenated in 16-bit increments. (6.3.11#8)
8. Instruct the Compliance Host to send an EPHandleDeleteReq packet to the DUT that targets all endpoints except EP0.
9. Verify that the DUT sends an EPHandleDeleteResp packet in response to the EPHandleDeleteReq packet. (6.3.17#1)
10. Verify that the EPHandleDeleteResp packet has:
    a. Type field set to 0. (6.3.17#2)
    b. Subtype field set to 15. (6.3.17#2)
    c. Status Code field set to a value listed in Table 6. (6.3.17#4)
    d. Number of EP Handles with Error field set to the number of EP handles included in the EP Handle List field. (6.3.17#5)
    e. Bits 5 through 31 of DWORD 3 set to 0 (reserved). (6.3.17#6)
    f. EP Handle List field that carries the number of EP handles indicated in the Number of EP Handles with Error field. (6.3.17#7)
11. Instruct the Compliance Host to send an EPInactivateReq packet to the DUT that targets EP0.
12. Verify that the DUT sends and EPInactivateResp packet. (6.3.11#1)
13. Verify that the EPInactivateResp has:
    a. Type field set to 0. (6.3.11#2)
    b. Subtype field set to 9. (6.3.11#2)
    c. Status Code field set to a value listed in Table 6. (6.3.11#4)
    d. Number of EP Handles with Error field that is equal to the number of EP handles entries in the EP Handle List field. (6.3.11#5)
    e. Number of EP Handles with Error field set to 0 if the Status Code field is 0 (SUCCESS). (6.3.11#6)
    f. Bits 5 through 31 of DWORD 3 set to 0 (reserved). (6.3.11#7)
    g. EP Handle List field that carries a list of the EP handles that failed and is concatenated in 16-bit increments. (6.3.11#8)
14. Instruct the Compliance Host to send an EPHandleDeleteReq packet to the DUT that targets EP0.
15. Verify that the DUT sends an EPHandleDeleteResp packet in response to the EPHandleDeleteReq packet. (6.3.17#1)
16. Verify that the EPHandleDeleteResp packet has:
    a. Type field set to 0. (6.3.17#2)
    b. Subtype field set to 15. (6.3.17#2)
    c. Status Code field set to a value listed in Table 6. (6.3.17#4)

   d. Number of EP Handles with Error field set to the number of EP handles included in the EP Handle List field. (6.3.17#5)
   e. Bits 5 through 31 of DWORD 3 set to 0 (reserved). (6.3.17#6)
   f. EP Handle List field that carries the number of EP handles indicated in the Number of EP Handles with Error field. (6.3.17#7)
17. Instruct the Compliance Host to send a DevDisconnectReq packet.
18. Verify that the DUT sends a DevDisconnectResp packet. (6.3.27#1)
19. Verify that the DevDisconnectResp packet has:
   a. Type field set to 0. (6.3.27#2)
   b. Subtype field set to 25. (6.3.27#2)
   c. Status Code field set to a value listed in Table 6. (6.3.27#4)
20. Disconnect the DUT from the Compliance Host.

**Case 2 – Host-Initiated Session Tear Down (Alternative 2)**

1. Connect the DUT to the Compliance Host.
2. Enumerate and configure the DUT.
3. Instruct the Compliance Host to send an EPInactivateReq to the DUT that targets all endpoints except EP0.
4. Verify that the DUT sends and EPInactivateResp packet. (6.3.11#1)
5. Verify that the EPInactivateResp has:
   a. Type field set to 0. (6.3.11#2)
   b. Subtype field set to 9. (6.3.11#2)
   c. Status Code field set to a value listed in Table 6. (6.3.11#4)
   d. Number of EP Handles with Error field that is equal to the number of EP handles entries in the EP Handle List field. (6.3.11#5)
   e. Number of EP Handles with Error field set to 0 if the Status Code field is 0 (SUCCESS). (6.3.11#6)
   f. Bits 5 through 31 of DWORD 3 set to 0 (reserved). (6.3.11#7)
   g. EP Handle List field that carries a list of the EP handles that failed and is concatenated in 16-bit increments. (6.3.11#8)
6. Instruct the Compliance Host to send a ClearTransfersReq packet that targets all endpoints except EP0.
7. Verify that the DUT sends a ClearTransfersResp packet. (6.3.15#1)
8. Verify that the ClearTransfersResp packet has:
   a. Type field set to 0. (6.3.15#2)
   b. Subtype field set to 13. (6.3.15#2)
   c. Status Code field set to a value listed in Table 6. (6.3.15#4)
   d. Number of cancel transfers status blocks field set to the number of cancel transfers status blocks and is equal to the value of Number of cancel transfers information blocks field in the corresponding ClearTransfersReq packet. (6.3.15#5)
   e. Bits 5 through 31 of DWORD 3 set to 0 (reserved). (6.3.15#6)
   f. Verify that each cancel transfers status block in the ClearTransfersResp packet has:
      i. EP Handle field set to the EP Handle Value in the corresponding ClearTransfersReq packet. (6.3.15#7)
      ii. For an endpoint that supports the Enhanced SuperSpeed Stream Protocol, Stream ID field that matches the Stream ID field in the corresponding ClearTransfersReq packet. (6.3.15#8)
      iii. For an endpoint that does not support the Enhanced SuperSpeed Stream Protocol, Stream ID field reserved and set to 0. (6.3.15#8)
      iv. Bits 2 through 31 of DWORD 1 set to 0 (reserved). (6.3.15#11)
      v. Last Request ID equal to the Request ID field in the last TransferResp packet sent by the DUT. (6.3.15#12)
      vi. If target endpoint is an OUT endpoint and the Partial Delivery field is set to 1, verify that the Delivered Sequence Number field is set to the value of the Sequence Number field in the last TransferResp sent by the DUT. (6.3.15#13)
      vii. If target endpoint is not an OUT endpoint or the Partial Delivery field is set to 0, verify that the Delivered Sequence Number field is set to 0 reserved). (6.3.15#13)
9. Instruct the Compliance Host to send an EPHandleDeleteReq packet to the DUT that targets all endpoints except EP0.

10. Verify that the DUT sends an EPHandleDeleteResp packet in response to the EPHandleDeleteReq packet. (6.3.17#1)
11. Verify that the EPHandleDeleteResp packet has:
    a. Type field set to 0. (6.3.17#2)
    b. Subtype field set to 15. (6.3.17#2)
    c. Status Code field set to a value listed in Table 6. (6.3.17#4)
    d. Number of EP Handles with Error field set to the number of EP handles included in the EP Handle List field. (6.3.17#5)
    e. Bits 5 through 31 of DWORD 3 set to 0 (reserved). (6.3.17#6)
    f. EP Handle List field that carries the number of EP handles indicated in the Number of EP Handles with Error field. (6.3.17#7)
12. Instruct the Compliance Host to send an EPInactivateReq packet to the DUT that targets EP0.
13. Verify that the DUT sends and EPInactivateResp packet. (6.3.11#1)
14. Verify that the EPInactivateResp has:
    a. Type field set to 0. (6.3.11#2)
    b. Subtype field set to 9. (6.3.11#2)
    c. Status Code field set to a value listed in Table 6. (6.3.11#4)
    d. Number of EP Handles with Error field that is equal to the number of EP handles entries in the EP Handle List field. (6.3.11#5)
    e. Number of EP Handles with Error field set to 0 if the Status Code field is 0 (SUCCESS). (6.3.11#6)
    f. Bits 5 through 31 of DWORD 3 set to 0 (reserved). (6.3.11#7)
    g. EP Handle List field that carries a list of the EP handles that failed and is concatenated in 16-bit increments. (6.3.11#8)
15. Instruct the Compliance Host to send an EPHandleDeleteReq packet to the DUT that targets EP0.
16. Verify that the DUT sends an EPHandleDeleteResp packet in response to the EPHandleDeleteReq packet. (6.3.17#1)
17. Verify that the EPHandleDeleteResp packet has:
    a. Type field set to 0. (6.3.17#2)
    b. Subtype field set to 15. (6.3.17#2)
    c. Status Code field set to a value listed in Table 6. (6.3.17#4)
    d. Number of EP Handles with Error field set to the number of EP handles included in the EP Handle List field. (6.3.17#5)
    e. Bits 5 through 31 of DWORD 3 set to 0 (reserved). (6.3.17#6)
    f. EP Handle List field that carries the number of EP handles indicated in the Number of EP Handles with Error field. (6.3.17#7)
18. Instruct the Compliance Host to send a DevDisconnectReq packet.
19. Verify that the DUT sends a DevDisconnectResp packet. (6.3.37#1)
20. Verify that the DevDisconnectResp packet has:
    a. Type field set to 0. (6.3.37#2)
    b. Subtype field set to 25. (6.3.37#2)
    c. Device Handle field reserved and set to 0. (6.3.37#3)
    d. Status Code field set to a value listed in Table 6. (6.3.37#4)
21. Disconnect the DUT from the Compliance Host.

**Case 3 – Device-Initiated Session Tear Down**

*Note: Only test Case 3 if the DUT supports device-initiated removal*

1. Connect the DUT to the Compliance Host.
2. Enumerate and configure the DUT.
3. Per vendor instructions, initiate device removal from the DUT.
4. Verify that the DUT sends a DevInitDisconnectReq packet.
5. Verify that the DevInitDisconnectReq has:
    a. Type field set to 0. (6.3.38#1)
    b. Subtype field set to 37. (6.3.38#1)
    c. Status Code field set to value in Table 6. (6.3.38#2)
    d. Device Handle field set to 0 (reserved). (6.3.38#3)

6. Instruct the Compliance Host to send a DevInitDisconnectResp packet in response.
7. Wait for the device disconnection sequence to finish.
8. Instruct the Compliance Host to send a PingReq packet to the DUT.
9. Verify that the DUT does not respond.

## TD 6.21 Remote Wake Test

*Note: Only run this test if the DUT supports remote wake.*

**Assertions Covered**

1. 6.3.32#1, 6.3.32#2, 6.3.32#4, 6.3.32#5, 6.3.50#1, 6.3.50#2, 6.3.50#4, 6.3.50#5
2. 8.2.2.2#4, 8.2.2.2#5

**Required Equipment**

1. MA USB Compliance Host
2. KG USB Mouse (if DUT is a USB2.0 hub)
3. SS/SSP remote wake capable device (if DUT is a USB3.1 hub)

**Test Steps**

1. If the DUT is not connected to the Compliance Host, is connected but MA USB session is not Active, or the Compliance Host is configured to support Link Sleep:
    a. Instruct the Compliance Host to not support Link Sleep in the CapReq packet sent to the DUT during enumeration.
    b. Connect/reconnect the DUT to the Compliance Host.
    c. Enumerate and configure the DUT.
2. If the DUT is a hub:
    a. Connect the Mouse to the DUT.
    b. Enumerate and configure the mouse for remote wake.
3. Instruct the Compliance Host to send an EPInactivateReq packet to the DUT.
4. Instruct the Compliance Host to send a USBSuspendReq packet to the DUT.
5. Instruct the Compliance Host to send a SleepReq packet to the DUT.
6. Wait for the MA session to go inactive.
7. Instruct the Compliance Host to send a TransferReq packet to the DUT.
8. Verify that the DUT sends a TransferResp with the Status Code field set to INVALID_EP_HANDLE_STATE. (8.2.2.2#4)
9. Attempt to use the DUT (jiggle the Mouse if DUT is a hub or otherwise cause DUT to wake and resume).
10. Verify that the DUT sends a RemoteWakeReq packet. (8.2.2.2#5)
11. Verify that the RemoteWakeReq packet has:
    a. Type field set to 0. (6.3.32#1)
    b. Subtype field set to 30. (6.3.32#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.32#2)
    d. USB Device Resumed field set to 1 if USB device is not suspended. (6.3.32#4)
    e. USB Device Resumed field set to 0 if USB device is suspended. (6.3.32#4)
    f. Bits 1 through 31 of DWORD 3 set to 0 (reserved). (6.3.32#5)
12. If the DUT is a SuperSpeed device, verify that the DUT sends a DevNotificationReq.
13. Verify that the DevNotificationReq packet has:
    a. Type field set to 0. (6.3.50#1)
    b. Subtype field set to 48. (6.3.50#1)
    c. Status Code field set to 0 (SUCCESS). (6.3.50#2)
    d. Bits 0 through 3 of DWORD 3 set to 0 (reserved). (6.3.50#4)
    e. Notification Type field set to a value as defined in section 8.5.6 of USB 3.1. (6.3.50#5)
14. Wait for the wake/resume process to complete.
15. Use the DUT to verify that it is active and functional.

# TD 6.22 Invalid USBDevHandleReq Test

**Assertions Covered**

1. 7.3.2.1#4

**Required Equipment**

1. MA USB Compliance Host

**Test Steps**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the Compliance Host.
    b. Enumerate and configure the DUT.
2. Instruct the Compliance Host to send a duplicate USBDevHandleReq packet that is not a retry.
3. Verify that the DUT responds with a USBDevHandleResp packet that has the Status Code field set to 133 (INVALID_REQUEST). (7.3.2.1#4)
4. Repeat for each integrated USB device.

# TD 6.23 Reserved

***Reserved for Future Use***

# TD 6.24 Enhanced SuperSpeed Bulk Stream Test

*Note: Only run this test if the DUT is a hub or UASP-capable.*

**Assertions Covered**

1. 5.2.1.1#3
2. 6.3.45#1, 6.3.45#2, 6.3.45#4, 6.3.45#5, 6.3.45#6, 6.3.45#7, 6.3.45#8, 6.3.45#9, 6.3.45#10, 6.3.47#1, 6.3.47#2, 6.3.47#4

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. KG MA USB Host
4. UASP Capable USB Device (for testing hub)

**Test Steps**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the Compliance Host.
    b. Enumerate and configure the DUT.
2. If the DUT is a hub:
    a. Connect the UASP Capable Device to the DUT.
    b. Enumerate and configure the UASP Capable Device.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Operate the DUT or UASP Capable Device.
5. Verify that the DUT sends an EPOpenStreamResp packet in response to the EPOpenStreamReq packet. (6.3.45#1)
6. Verify that the EPOpenStreamResp packet is received from the DUT within aManagementResponseTime + 2 * aManagementChannelDelay from when the EPOpenStreamReq packet was released to the management channel. (5.2.1.1#3)
7. Verify that the EPOpenStreamResp has:
    a. Type field set to 0. (6.3.45#2)
    b. Subtype field set to 43. (6.3.45#2)
    c. Status Code field set to a value in Table 6. (6.3.45#4)
    d. Number of Streams field set to the number of streams included in the packet. (6.3.45#5)
    e. If the Status Code field is 0 (SUCCESS), the Number of Streams field is less than or equal to the Number of Streams field in the corresponding EPOpenStreamReq packet. (6.3.45#6)
    f. Number of Stream ID Blocks field set to the number of stream ID blocks in the packet. (6.3.45#7)
    g. Stream ID interval blocks are listed in increasing order of stream ID values. (6.3.45#8)

   h. First Stream ID field in a stream ID interval block indicates the value of the first stream ID in the interval block. (6.3.45#9)
   i. Last Stream ID field in a stream ID interval block indicates the value of the last stream ID in the interval block. (6.3.45#10)
8. If the DUT is a hub, remove the UASP-Capable Device.
9. If the DUT is not a hub, eject the DUT.
10. Verify that the DUT sends an EPCloseStreamResp packet in response to the EPCloseStreamReq packet. (6.3.47#1)
11. Verify that the EPCloseStreamResp has:
    a. Type field set to 0. (6.3.47#2)
    b. Subtype field set to 45. (6.3.47#2)
    c. Status Code field set to a value in Table 6. (6.3.47#4)

# TD 6.25 USB Device Reset Test

**Assertions Covered**

1. 5.2.1.1#3
2. 6.3.49#1, 6.3.49#2, 6.3.49#4
3. 7.3.5#5

**Required Equipment**

1. MA USB Compliance Host
2. KG USB Device

**Test Steps**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
   a. Connect/reconnect the DUT to the Compliance Host.
   b. Enumerate and configure the DUT.
2. If the DUT is a hub, connect the KG USB device downstream of the DUT.
3. Instruct the Compliance Host to send a USBDevResetReq packet to the DUT. If the DUT is a hub, the USBDevResetReq should target the KG USB Device. Otherwise, the USBDevResetReq targets the integrated USB device.
4. Verify that the DUT responds with a USBDevResetResp packet. (6.3.49#1, 7.3.5#5)
5. Verify that the USBDevResetResp packet is received from the DUT within aManagementResponseTime + 2 * aManagementChannelDelay from when the USBDevResetReq packet was released to the management channel. (5.2.1.1#3)
6. Verify that the USBDevResetResp has:
   a. Type field set to 0. (6.3.49#2)
   b. Subtype field set to 47. (6.3.49#2)
   c. Status Code field set to value in Table 6. (6.3.49#4)

# TD 6.26 Pending Transfer Multi-Endpoint Test (Hub Only)

**Assertions Covered**

1. 5.4#38

**Required Equipment**

1. KG MA USB Host
2. USB Compliance Device

**Test Steps**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
   a. Connect/reconnect the DUT to the Compliance Host.
   b. Enumerate and configure the DUT.
2. Connect the Compliance Device to the DUT.
3. Enumerate and configure the Compliance Device to have two bulk IN and one bulk OUT endpoint.

**Case 1 – Single Pending Endpoint**

4. Initiate a transfer on one of the IN endpoints.

5. Instruct the Compliance Device to delay responding to the transfer so that the DUT returns a null TransferResp packet with the Status Code set to 135 (TRANSFER_PENDING).
6. Initiate a transfer on the OUT endpoint.
7. Verify that the OUT transfer completes successfully. (5.4#38)

**Case 2 – Multiple Pending Endpoints**

1. Initiate transfers on both IN endpoints.
2. Instruct the Compliance Device to delay responding to both transfers so that the DUT returns a null TransferResp packet for each transfer with the Status Code set to 135 (TRANSFER_PENDING).
3. Initiate a transfer on the OUT endpoint.
4. Verify that the OUT transfer completes successfully. (5.4#38)

# TD 6.27 Get Port Bandwidth Test (Hub Only)

**Assertions Covered**

1. 5.2.1.1#3
2. 6.3.55#1, 6.3.55#2, 6.3.55#3, 6.3.55#4, 6.3.55#5, 6.3.55#6, 6.3.55#7, 6.3.55#8, 6.3.55#9

**Required Equipment**

1. MA USB Compliance Host

**Test Steps**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the Compliance Host.
    b. Enumerate and configure the DUT.
2. Instruct the Compliance Host to send a GetPortBWReq packet to the DUT.
3. Verify that DUT responds with a GetPortBWResp packet. (6.3.55#1)
4. Verify that the GetPortBWResp packet is received from the DUT within aManagementResponseTime + 2 * aManagementChannelDelay from when the GetPortBWResp packet was released to the management channel. (5.2.1.1#3)
5. Verify that the GetPortBWResp packet has:
    a. Type field set to 0. (6.3.55#2)
    b. Subtype field set to 53. (6.3.55#2)
    c. Status Code field set to value in Table 6. (6.3.55#3)
    d. If the DUT does not support GetPortBWReq packets, Status Code field set to 154 (NOT_SUPPORTED). (6.3.55#4)
    e. If the DUT does not support the speed indicated by the value of the Sublink Speed Attribute ID field , Status Code field set to 128 (UNSUCCESSFUL). (6.3.55#5)
    f. If Status Code field is 0 (SUCCESS), the Number of Ports field set to the number of ports in the Port Available Bandwidth List field. (6.3.55#6)
    g. Bits 8 through 31 of DWORD 3 set to 0 (reserved). (6.3.55#7)
    h. If Status Code field is 0 (SUCCESS), the Port Available Bandwidth List field has a list of port available bandwidth reports starting with port number 1 and increasing in order. (6.3.55#8)
    i. Each Port Available Bandwidth List entry is between 0 and 100. (6.3.55#9)
    j. If the Speed field in the GetPortBWReq packet is 0 (LS), 1 (FS), or 2 (HS), verify that:
        i. For USB3 ports, verify that Port Bandwidth Context is 0.
        ii. For USB2 ports with no integrated devices, when Dev Speed is HS, verify that Port Bandwidth Context is either 80 or 0.
        iii. For USB2 ports with no integrated devices, when Dev Speed is FS or LS, verify that Port Bandwidth Context is either 90 or 0.
        iv. For USB2 ports with an integrated USB2 device, when Dev Speed is HS, verify that Port Bandwidth Context is less than or equal to 80.
        v. For USB2 Ports with an integrated USB2 device, when Dev Speed is FS or LS, verify that Port Bandwidth Context is less than or equal to 90.
    k. If the Speed field in the GetPortBWReq packet is 3 (SS) or 4 (SS+) verify that:

i. For USB2 ports, verify that Port Bandwidth Context is 0.
ii. For USB3 ports, verify that Port Bandwidth Context is less than or equal to 90.

**Repetitions**

- Repeat with the Speed field in the GetPortBWReq packet set to 0 (LS), 1 (FS), 2, (HS), 3 (SS), and 4 (SS+)

# TD 6.28 Device-Initiated Session Transition Test

*Note: Only run this test if the DUT* supports *Link Sleep*.

**Assertions Covered**

1. 6.3.56#1, 6.3.56#2, 6.3.56#3, 6.3.56#4, 6.3.56#5, 6.3.58#1, 6.3.58#2, 6.3.58#3
2. 8.2.1.2#1, 8.2.1.2#2, 8.2.1.2#4, 8.2.2.2#2, 8.2.2.2#9

**Required Equipment**

1. MA USB Compliance Host

**Test Steps**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the Compliance Host.
    b. Enumerate and configure the DUT.
2. Attempt to initiate the Link Sleep procedure as indicated by DUT vendor.
3. Verify that the DUT does not send a SleepReq packet unless:
    a. The EP Descriptors received from the DUT indicate that the DUT only has control and/or non-Isochronous IN endpoints. (8.2.1.2#1)
    b. The Compliance Host has received a TransferResp packet from the DUT with the Status Code field set to 135 (TRANSFER_PENDING) for each IN endpoint. (8.2.1.2#1)
    c. The Compliance Host has sent a TransferAck packet to the DUT with the Status Code field set to 135 (TRANSFER_PENDING) for each IN endpoint. (8.2.1.2#1, 8.2.1.2#2)
4. If the DUT sends a SleepReq packet:
    a. verify that the SleepReq packet has:
        i. Type field set to 0. (6.3.56#1)
        ii. Subtype field set to 54. (6.3.56#1)
        iii. Status Code field set to 0 (SUCCESS). (6.3.56#2)
        iv. Device Handle field set to 0 (reserved). (6.3.56#3)
        v. Management Request Timeout field set to non-zero value. (6.3.56#4)
        vi. Data Request Timeout field set to non-zero value. (6.3.56#5)
    b. Instruct the Compliance Host to respond with a SleepResp packet with Status Code field set to 0 (SUCCESS).
    c. Wait for the DUT to move its session state to Session Inactive transition into low power mode.
    d. Instruct the Compliance Host to send a TransferReq packet to the DUT.
    e. Verify that the DUT does not respond. (8.2.1.2#4)
5. Attempt to resume the DUT as indicated by the DUT vendor
6. If the DUT sends a WakeReq packet:
    a. Verify that DUT has exited low power mode. (8.2.2.2#2)
    b. Verify that the WakeReq packet from the DUT has:
        i. Type field set to 0. (6.3.58#1)
        ii. Subtype field set to 56. (6.3.58#1)
        iii. Status Code field set to 0 (SUCCESS). (6.3.58#2)
        iv. Device Handle field set to 0 (reserved). (6.3.58#3)
    c. Instruct the Compliance Host to send a PingReq packet to the DUT.
    d. Verify that the DUT responds with a PingResp packet. (8.2.2.2#2)
7. If the DUT sends a packet requiring a response from the Compliance Host:
    a. verify that:
        i. The EP Descriptors received from the DUT indicate that the DUT only has control and/or non-Isochronous IN endpoints. (8.2.2.2#9)

ii. The Compliance Host has received a TransferResp packet from the DUT with the Status Code field set to 135 (TRANSFER_PENDING) for each IN endpoint. (8.2.2.2#9)
iii. The Compliance Host has sent a TransferAck packet to the DUT with the Status Code field set to 135 (TRANSFER_PENDING) for each IN endpoint. (8.2.2.2#9)

b. Instruct the Compliance Host to respond to the packet from the DUT.

# TD 6.29 Inactive Session Timing Test

**Assertions Covered**

1. 6.3.56#4, 6.3.56#5, 6.3.57#5, 6.3.57#6
2. 8.2.1.1#6, 8.2.1.2#4

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. MA USB Compliance Host

**Test Steps**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the Compliance Host.
    b. Enumerate and configure the DUT.
2. Ensure that the following steps are captured by the MA USB Sniffer.

**Case 1 – Host-Initiated Transition**

1. If the DUT does not support Link Sleep, instruct the Compliance Host to suspend the DUT by sending an EPInactivateReq packet followed by a USBSuspendReq packet for each integrated USB device.
2. Instruct the Compliance Host to send a SleepReq Packet to the DUT with the Management Request Timeout field set to 5000 and the Data Request Timeout field set to 4000.
3. If the DUT responds with a SleepResp packet that has the Status Code field set to 155 (REQUEST_DENIED), instruct the Compliance Host to send a SleepReq packet to the DUT with the same Management Request Timeout and Data Request Timeout fields as the previously received SleepResp packet.
4. If the DUT responds with a SleepResp with Status Code Field set to 0 (SUCCESS), continue with the test. Otherwise end test here or retry from beginning.
5. If the DUT sends any management packets, instruct the Compliance Host to not respond until the number of milliseconds in the Management Request Timeout field has passed.
    a. Verify that the DUT retries the packet. (8.2.1.2#4, 6.3.57#5)
    b. Instruct the Compliance Host to respond to the retried packet within Management Request Timeout value - 2 * aManagementChannelDelay.
    c. Verify that the DUT does not retry the packet. (8.2.1.2#4, 6.3.57#5)
6. If the DUT sends any data or control packets, instruct the Compliance Host to not respond until the number of milliseconds in the Data Request Timeout field has passed.
    d. Verify that the DUT retries the packet. (8.2.1.2#4, 6.3.57#6)
    e. Instruct the Compliance Host to respond to the retried packet within Data Request Timeout value - 2 * aDataChannelDelay.
    f. Verify that the DUT does not retry the packet. (8.2.1.2#4, 6.3.57#5)
7. Instruct the Compliance Host to send a WakeReq packet to the HUT.
8. Verify that the DUT responds within Data Request Timeout value - 2 * aManagementChannelDelay. (8.2.1.1#6, 6.3.56#4)
9. Verify that the DUT is in Session Active by instructing the Compliance Host to send a PingReq packet to the DUT and verifying that the DUT responds with a PingResp packet.

**Case 2 – Device-Initiated Transition**

*Note: only run Case 2 if the DUT supports Link Sleep.*

1. Initiate Link Sleep procedure as indicated by DUT vendor.

2. Instruct the Compliance Host to respond to the SleepReq packet from the DUT with a SleepResp packet that has the Status Code set to 0 (SUCCESS) and Management Timeout and Data Timeout fields equal to the Management Timeout and Data Timeout fields in the SleepReq packet from the DUT.
3. If the DUT sends any management packets, instruct the Compliance Host to not respond until the number of milliseconds in the Management Request Timeout field has passed.
    g. Verify that the DUT retries the packet. (8.2.1.2#4)
    h. Instruct the Compliance Host to respond to the retried packet within Management Request Timeout value - 2 * aManagementChannelDelay.
    i. Verify that the DUT does not retry the packet. (8.2.1.2#4)
4. If the DUT sends any data or control packets, instruct the Compliance Host to not respond until the number of milliseconds in the Data Request Timeout field has passed.
    j. Verify that the DUT retries the packet. (8.2.1.2#4, 6.3.56#5)
    k. Instruct the Compliance Host to respond to the retried packet within Data Request Timeout value - 2 * aDataChannelDelay.
    l. Verify that the DUT does not retry the packet. (8.2.1.2#4, 6.3.56#5)
5. Instruct the Compliance Host to send a WakeReq packet to the HUT.
6. Verify that the DUT responds within Management Request Timeout value - 2 * aManagementChannelDelay. (8.2.1.2#4, 6.3.57#5)

# TD 6.30 Non-Zero Reserved Bit Test

**Assertions Covered**

1. 6.2#4

**Required Equipment**

1. MA USB Compliance Host

**Test Steps**

**Case 1 – Device Enumeration**

1. Disconnect the DUT from the Compliance Host if previously connected.
2. Connect the DUT to the Compliance Host.
3. Start enumeration.
4. Instruct the Compliance Host to send a DevResetReq packet to the DUT with all of the reserved bits set to 1.
5. Verify that the DUT responds with a DevResetResp packet with the Status Code field set to 0 (SUCCESS). (6.2#4)

**Repetitions**

- Repeat test for each of the following packet types with that packet type replacing the DevResetReq/Resp packets.
    - CapReq/Resp
    - USBDevHandleReq/Resp targeting the integrated USB 2.0 device (for hubs and non-SS devices)
    - USBDevHandleReq/Resp targeting the integrated USB 3.0 device (for SS hubs and SS devices)
    - EPHandleReq/Resp
    - ModifyEP0Req/Resp
    - SetUSBDevAddrReq/Resp targeting the integrated USB 2.0 device (for hubs and non-SS devices)
    - SetUSBDevAddrReq/Resp targeting the integrated USB 3.0 device (for SS hubs and SS devices)
    - UpdateDevReq/Resp targeting the integrated USB 2.0 device (for hubs and non-SS devices)
    - UpdateDevReq/Resp targeting the Integrated USB 3.0 device for (SS hubs and non-SS devices)

**Case 2 – Host-Initiated Disconnect**

1. Instruct the Compliance Host to send a CancelTransferReq packet to the DUT with all of the reserved bits set to 1.
2. Verify that the DUT responds with a CancelTransferResp packet with the status Code field set to 0 (SUCCESS). (6.2#4)

- Repeat test for each of the following packet types with that packet type replacing the CancelTransferReq/Resp packets.
    - EPInactivateReq/Resp
    - ClearTransfersReq/Resp

- EPHandleDeleteReq/Resp
- USBDevDisconnectReq/Resp targeting the integrated USB 2.0 device (for hubs and non-SS devices)
- USBDevDisconnectReq/Resp targeting the integrated USB 3.0 device (for SS hubs and SS devices)

# TD 6.31 Incorrect SSID Test

**Assertions Covered**

1. 8.1.2.1#7

**Required Equipment**

1. MA USB Compliance Host

**Test Steps**

**Case 1 – new SSID**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the Compliance Host.
    b. Enumerate and configure the DUT.
2. Instruct the Compliance Host to send a DevDisconnectReq packet to the DUT with an SSID that is different than the SSID in previously sent packets.
3. Verify that the DUT does not respond to the DevDisconnectReq packet. (8.1.2.1#7)
4. Instruct the Compliance Host to send a CapReq packet with an SSID that is different than the SSID in previously sent packets.
5. Verify that the DUT does not respond to the CapReq packet. (8.1.2.1#7)
6. Instruct the Compliance Host to send a PingReq packet with the MA USB Device Address field set to 0xFF and the SSID field set to a value that is different than the SSID in previously sent packets.
7. Verify that the DUT does not respond to the PingReq packet. (8.1.2.1#7)

**Case 2 – Old SSID**

1. Disconnect the DUT from the Compliance Host.
2. Reconnect the DUT to the Compliance Host.
3. Instruct the Compliance Host to enumerate and configure the DUT using a SSID that is different than the SSID in previously sent packets.
4. Instruct the Compliance Host to send a DevDisconnectReq packet with an SSID that is the same as the SSID assigned during enumeration in Case 1.
5. Verify that the DUT does not respond to the DevDisconnectReq packet. (8.1.2.1#7)
6. Instruct the Compliance Host to send a CapReq packet with an SSID that is the same as the SSID assigned during enumeration in Case 1.
7. Verify that the DUT does not respond to the CapReq packet. (8.1.2.1#7)
8. Instruct the Compliance Host to send a PingReq packet with the MA USB Device Address field set to 0xFF and the SSID field set to a value that is the same as the SSID assigned during enumeration in Case 1.
9. Verify that the DUT does not respond to the PingReq packet. (8.1.2.1#7)

# TD 6.32 Device Removable Bit Test (SuperSpeed Hubs Only)

**Assertions Covered**

1. None

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. KG MA USB Host

**Test Steps**

1. Ensure that the following steps are captured by the MA USB Sniffer.
2. Connect the DUT to the Compliance Host.

3. Prompt the user to indicate which hub downstream ports are non-removable.
4. Enumerate and configure the DUT.
5. Verify that the SuperSpeed Hub Descriptor from the Hub has the Device Removable bit set to 1 for any port that has a non-removable USB device attached. (See USB 3.0, Table 10-4)

## TD 6.33 Stall Test (Hubs Only)

**Assertions Covered**

1. 5.4#44, 5.4#48, 5.5#35, 5.5#38

**Required Equipment**

1. MA USB Compliance Host
2. USB Compliance Device that stalls

**Test Steps**

1. If the DUT is not connected to the Compliance Host or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the Compliance Host.
    b. Enumerate and configure the DUT.
2. Connect the USB Compliance Device to the DUT.
3. Enumerate the USB Compliance Device and configure it to have a bulk IN and a bulk OUT endpoint that always stall.
4. Initiate a bulk IN transfer.
5. Verify that the last TransferResp packet from the DUT has the Status Code field set to 136 (TRANSFER_EP_STALL), and the EoT field set to 1. (5.4#44)
6. Verify that the last TransferResp packet from the DUT has a non-zero Remaining Size field. (5.4#48)
7. Initiate a bulk OUT transfer.
8. Verify that the DUT sends a TransferResp packet with the Request ID field set to the Request ID of the stalled transfer, the EoT field set to 1, and the Status Code field set to 136 (TRANSFER_EP_STALL). (5.5#35)
9. Initiate a second bulk OUT transfer that targets the same endpoint as the previous OUT transfer.
10. Verify that the DUT sends a TransferResp packet with the Status Code field set to 132 (INVALID_EP_HANDLE_STATE). (5.5#38)

## TD 6.34 Short Packet Test (Hubs Only)

**Assertions Covered**

1. 5.4#41

**Required Equipment**

1. MA USB Compliance Host
2. Loopback-Capable USB Compliance Device

**Test Steps**

1. If the DUT is not connected to the MA USB Compliance Device or is connected but MA USB session is not Active:
    a. Connect/reconnect the DUT to the MA USB Compliance Host.
    b. Enumerate and configure the DUT.
2. If the USB Compliance Device is not connected to the DUT or is connected but not operational:
    a. Connect/reconnect the USB Compliance Device downstream of the DUT.
    b. Enumerate and configure the USB Compliance Device for loopback transfer.
3. From the Compliance Host system, initiate loopback transfer on the USB Compliance Device.
4. Instruct the USB Compliance Device to return less data for the IN transfer than received for the OUT transfer (i.e. generate a transfer that ends with a short packet)
5. Verify that the DUT returns a TransferResp packet that has the EoT field set to 1 and the Status Code field set to 141 (TRANSFER_SHORT_TRANSFER). (5.4#41)
6. Repeat 10 times.

## TD 6.35 Loopback Test (Hubs Only)

**Assertions Covered**

1. none

**Required Equipment**

1. MA USB Sniffer
2. MA USB Packet Analysis Software
3. KG MA USB Host
4. LS, FS, HS and SS Loopback-Capable USB Compliance Devices[4]

**Test Steps**

**Case 1 – Control Loopback**

1. If the MA USB Host is not connected to the DUT or is connected but MA USB session is not Active:
    a. Connect/reconnect the MA USB Host to the DUT.
    b. Enumerate and configure the DUT.
2. If the USB Compliance Device is not connected to the DUT or is connected but not operational:
    a. Connect/reconnect the USB Compliance Device downstream of the DUT.
    b. Enumerate and configure the USB Compliance Device for loopback transfer.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Initiate an OUT transfer to the control endpoint.
5. Initiate an IN transfer to the control endpoint.
6. Verify that the OUT data matches the IN data (data is the same and is in the same order).

**Repetitions**

- Repeat with LS USB Compliance Device with a max packet size of 8 bytes
- Repeat with FS USB Compliance Device with max packet sizes of 8, 16, 32, and 64 bytes
- Repeat with HS USB Compliance Device with a max packet size of 64 bytes
- Repeat with SS USB Compliance Device with a max packet size of 512 bytes

**Case 2 – Bulk Loopback**

1. If the USB Compliance Device is not operational, reconnect the USB Compliance Device behind the DUT, then enumerate.
2. Configure the USB Compliance Device for bulk endpoint loopback.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Initiate an OUT transfer to the bulk OUT endpoint.
5. Verify that the OUT transfer completes successfully.
6. Initiate an IN transfer to the bulk IN endpoint.
7. Verify that the IN transfer completes successfully.
8. Verify that the OUT data matches the IN data (data is the same and is in the same order).
9. Repeat for 10 iterations.

**Repetitions**

- Repeat with FS USB Compliance Device with max packet sizes of 8, 16, 32, and 64 bytes
- Repeat with HS USB Compliance Device with a max packet size of 512 bytes
- Repeat with SS USB Compliance Device with a max packet size of 1024 bytes
- Repeat with SS USB Compliance Device with max burst size from 1 to 16

**Case 3 – Interrupt Loopback**

1. If the USB Compliance Device is not operational, reconnect the USB Compliance Device behind the DUT, then enumerate.
2. Configure the USB Compliance Device for interrupt endpoint loopback.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Initiate an OUT transfer to the interrupt OUT endpoint.
5. Verify that the OUT transfer completes successfully.

[4] Note: If USB Compliance Devices are not available, KG USB Devices with multiple endpoint types can also be used for testing.

6. Initiate an IN transfer to the interrupt IN endpoint.
7. Verify that the IN transfer completes successfully.
8. Verify that the OUT data matches the IN data (data is the same and is in the same order).
9. Repeat for 10 iterations.

**Repetitions**

- Repeat with LS, FS, HS, and SS USB Compliance Devices
- Repeat for all allowed max packet sizes from 1 to 1040
- Repeat for all allowed max burst size for SS or HS interrupt transfers (1 to 3)

**Case 4 – Isochronous OUT Loopback**

1. If the USB Compliance Device is not operational, reconnect the USB Compliance Device behind the DUT, then enumerate.
2. Configure the USB Compliance Device for isochronous endpoint OUT looped back to bulk IN.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Initiate an OUT transfer with wMaxPacketSize Burst.
5. Verify that the OUT transfer completes successfully.
6. Initiate an IN transfer to the bulk IN endpoint.
7. Verify that the OUT transfer completes successfully.
8. Verify that the OUT data matches the IN data (data is the same and is in the same order).
9. Repeat for 10 iterations.

**Repetitions**

- Repeat with FS, HS, and SS USB Compliance Devices
- Repeat for transfer size of (wMaxPacketSize * Burst / 2) +1
- Repeat for all allowed max packets sizes for isochronous transfers
- Repeat for all allowed max burst size for HS isochronous transfers (1 to 3) and all allowed max burst size for SS isochronous transfers (1 to 16)
- Repeat for all allowed number of bursts (multi) for SS

**Case 5 – Isochronous IN Loopback**

1. If the USB Compliance Device is not operational, reconnect the USB Compliance Device behind the DUT, then enumerate.
2. Configure the USB Compliance Device for isochronous endpoint IN looped back to bulk IN.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Initiate an OUT transfer with wMaxPacketSize Burst.
5. Verify that the OUT transfer completes successfully.
6. Initiate an IN transfer to the bulk IN endpoint.
7. Verify that the IN transfer completes successfully.
8. Verify that the OUT data matches the IN data (data is the same and is in the same order).
9. Repeat for 10 iterations.

**Repetitions**

- Repeat with FS, HS, and SS USB Compliance Devices
- Repeat for transfer size of (wMaxPacketSize * Burst / 2) +1
- Repeat for all allowed max packets sizes for isochronous transfers
- Repeat for all allowed max burst size for HS isochronous transfers (1 to 3) and all allowed max burst size for SS isochronous transfers (1 to 16)
- Repeat for all allowed number of bursts (multi) for SS

**Case 6 – Isochronous streaming OUT**

1. If the USB Compliance Device is not operational, reconnect the USB Compliance Device behind the DUT, then enumerate.

2. Configure the USB Compliance Device for isochronous endpoint OUT looped back to bulk IN.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Stream continuous isochronous traffic of wMaxPacketSize * Burst for 30 minutes.
5. Verify that the Isochronous OUT transfer completed successfully.
6. Verify that the OUT data matches the bulk IN data (data is the same and is in the same order).
7. Repeat for 10 iterations.

**Repetitions**

- Repeat with FS, HS, and SS USB Compliance Devices
- Repeat for transfer size of (wMaxPacketSize * Burst / 2) +1
- Repeat for all allowed max packets sizes for isochronous transfers
- Repeat for all allowed max burst size for HS isochronous transfers (1 to 3) and all allowed max burst size for SS isochronous transfers (1 to 16)
- Repeat for all allowed number of bursts (multi) for SS

**Case 7 – Isochronous streaming IN**

1. If USB Compliance Device is not operational, reconnect the USB Compliance Device behind DUT, then enumerate.
2. Configure the USB Compliance Device for isochronous endpoint IN looped back to bulk OUT.
3. Ensure that the following steps are captured by the MA USB Sniffer.
4. Stream continuous isochronous traffic of wMaxPacketSize * Burst for 30 minutes.
5. Verify that the IN data matches the bulk OUT data (data is the same and is in the same order).
6. Repeat for 10 iterations.

**Repetitions**

- Repeat with FS, HS, and SS USB Compliance Devices
- Repeat for transfer size of (wMaxPacketSize * Burst / 2) +1
- Repeat for all allowed max packets sizes for isochronous transfers
- Repeat for all allowed max burst size for HS isochronous transfers (1 to 3) and all allowed max burst size for SS isochronous transfers (1 to 16)
- Repeat for all allowed number of bursts (multi) for SS