Compliance Test Specification
for Personal Healthcare Device Class Specification

Date: January 14, 2010

Revision: 1.1

Scope of this Revision

The 1.1 revision of the specification is intended to describe the testing to be applied to host and device hardware and software based on the Personal Healthcare Device Class Specification, Revision 1.0, plus February 15th, 2008 errata.

Revision History

Revision	Issue Date	Comments
1.0	July 14, 2008	Version implemented in USB PHDC test suite that is currently included in the USB Compliance Test Suite.
1.1	January 14, 2010	Editorial corrections made for publication.

Significant Contributors:

Song Chung (Welch Allyn)	Julie Fleischer (Intel Corporation)	Bob Hoy (Nonin Medical)	John Keys (Intel Corporation)
Matt Miller (Frontline Test Equipment)	Mark Schnell (Cisco Systems)	Pramod Pesara (Intel Corporation)	

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

Table of Contents

1. Introduction

This document provides the compliance criteria and test descriptions for USB hosts and devices that conform to the Personal Healthcare Device Class Specification, Revision 1.0. These criteria are to be met in addition to the

full USB 2.0 compliance criteria. This document is relevant for anyone building a personal healthcare device that implements the Personal Healthcare Device Class Specification. The document is divided into two major sections. The first section lists the compliance criteria and the second section lists the test descriptions used to verify a device's or host's 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 PHDC Specification or other document from where 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 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	UUT

Assertion#: Unique identifier for each spec requirement. The identifier is in the form PHDC_SPEC_SECTION_NUMBER#X where X is a unique integer for a requirement in that section.

Assertion Description: Specific requirement from the specification

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

N/A This item is not explicitly tested in a PHDC test description. Items can be labeled N/A for several reasons – including items that are not testable, not important to test for interoperability, or are indirectly tested by other operations performed by the PHDC compliance test.

TD.1.X This item is covered by the test described in test description 1.X in this specification.

TBD A test description is not yet written for this assertion. The comments field provides more details on such items.

Comments: Provides additional information on requirements that are marked TBD.

UUT: Unit Under Test. Set to Device, Host, or Both, depending on whether the assertion applies to device tests, host tests, or both device and host tests.

2. Test Assertions

Unless otherwise noted, subsection references are to the PHDC Specification.

Assertion #	Assertion Description	Test #	Comment	UUT
Chapter 2 Test Assertions:				
Subsection reference: 2 Management Overview				
Subsection reference: 2.1 Personal Healthcare Use Cases				
Chapter 3 Test Assertions:				
Subsection reference: 3 Functional Characteristics				

Assertion #	Assertion Description	Test #	Comment	UUT
Subsection reference: 3.1 Data Transfer Characteristics				
Subsection reference: 3.2 Personal Healthcare Data QoS Bins				
Chapter 4 Test Assertions:				
Subsection reference: 4 Operational Model				
Subsection reference: 4.1 Device & Host Interactions				
Subsection reference: 4.2 Personal Healthcare Application & Data Layer Assumptions				
Subsection reference: 4.3 Application Meta-Data				
Subsection reference: 4.4 Architecture				
Subsection reference: 4.4.1 IN Endpoints				
4.4.1#1	All PHDC devices **shall** contain a bulk IN endpoint to be used for all reliable data transfers.	3.1.1, 3.2.1		Dev, Host
4.4.1#2	An interrupt IN endpoint **shall** be used if low latency/good reliability data needs to be sent.	3.1.6, 3.2.3		Dev, Host
4.4.1#3	The device **shall** be responsible for inducing data loss by flushing out stale data from the sending buffer.	N/A	I don't know of a way to test. No testing needed since the device determines what "stale" is.	
4.4.1#4	For all IN endpoints, the host PHDC component **shall** either queue a read request on the endpoint or utilize the Get Data Status request, or both, to allow for device-driven communication.	3.2.2		Host
Subsection reference: 4.4.2 OUT Endpoints				
4.4.2#1	All PHDC devices **shall** contain a bulk OUT endpoint.	3.1.1, 3.2.1		Dev, Host
Subsection reference: 4.4.3 Meta-Data Transfer				
Chapter 5 Test Assertions:				
Subsection reference: 5 Descriptors				
Subsection reference: 5.1 Standard Descriptors				
Subsection reference: 5.1.1 Device Descriptor				
5.1.1#1	Either bDeviceClass or bInterfaceClass **shall** be set to TBD – Assigned by USB-IF at the 1.0 level.	3.1.1		Dev, Host

Assertion #	Assertion Description	Test #	Comment	UUT
	As an FYI, the order of preference is: [Standard Interface Descriptor bDeviceClass field] **should** be set to 00h. It **may** be set to 0Fh. [Standard Device Descriptor bInterfaceClass field] **should** be set to 0Fh – Assigned by USB-IF at the 1.0 level.			
5.1.1#2	[Standard Device Descriptor bDeviceSubclass field] is 00h.	3.1.1		Dev, Host
5.1.1#3	[Standard Device Descriptor bDeviceProtocol field] is 00h.	3.1.1		Dev, Host
Subsection reference: 5.1.2 Configuration Descriptor **Subsection reference: 5.1.3 Interface Descriptors**				
5.1.3#1	If multiple interfaces are used, PHDC **shall** be declared at the Interface level. That is, [Standard Interface Descriptor bInterfaceClass field] **shall** be set to 0Fh.	3.1.1		Dev, Host
5.1.3#2	[Standard Interface Descriptor bInterfaceSubclass field] is 00h.	3.1.1		Dev, Host
5.1.3#3	[Standard Interface Descriptor bInterfaceProtocol field] is 00h.	3.1.1		Dev, Host
5.1.3#4	If a device implements multiple personal healthcare functions, one of the following two methods **shall** be used for enabling multi-function devices. 1. Multiple Interfaces – The multi-function device **may** expose one interface per device function. 2. Combination Device – The multi-function device **may** implement only one interface for the multi-function device.	3.1.2		Dev
Subsection reference: 5.2 Class-Defined Descriptors				
Subsection reference: 5.2.1 PHDC Class Function Descriptor				
5.2.1#1	The format of [the PHDC Class Function descriptor] **shall** be as defined in Table 7 PHDC Class Function Descriptor of the PHDC spec.	3.1.3		Dev, Host
5.2.1#2	[The PHDC Class Function descriptor] **shall** follow the interface descriptor with which it is associated and precede the PHDC Function Extension descriptor and the applicable endpoint descriptors.	3.1.3		Dev, Host

Assertion #	Assertion Description	Test #	Comment	UUT
5.2.1#3	The *bmCapability* field is a bitmap that **shall** be set as defined in Table 7 – PHDC Class Function Descriptor.	3.1.3		Dev, Host
Subsection reference: 5.2.2 Function Extension Descriptors				
5.2.2#1	The Function Extension descriptor **shall** follow the PHDC Class Function descriptor and precede the applicable endpoint descriptors.	3.5.2		Dev, Host
Subsection reference: 5.2.3 PHDC QoS Descriptor				
5.2.3#1	Each endpoint descriptor **shall** be followed by a QoS descriptor that defines the latency and reliability characteristics for data being sent over that endpoint.	3.1.3		Dev, Host
5.2.3#2	The format of [the PHDC QoS descriptor] **shall** be as defined in Table 8 – PHDC QoS Descriptor of the PHDC spec.	3.1.3		Dev, Host
5.2.3#3	The bQoSEncodingVersion field **shall** contain the version number of the QoS information encoding.	3.1.3		Dev, Host
5.2.3#4	If a host implementing 01h QoS information encoding receives a *bQoSEncodingVersion* field that is not 01h, it **shall** ignore the descriptor.	3.1.4		Host
5.2.3#5	Assuming *bQoSEncodingVersion* is 01h, if the PHDC QoS Descriptor is for a bulk endpoint, then a combination of bits b1-b5 corresponding to the latency and reliability values the endpoint supports **shall** be set in *bmLatencyReliability* as defined in Table 8 – PHDC QoS Descriptor of the PHDC spec.	3.1.5		Dev, Host
5.2.3#6	Assuming *bQoSEncodingVersion* is 01h, if the PHDC QoS Descriptor is for an interrupt endpoint, then *bmLatencyReliability* bit b0 **shall** be set as listed in Table 9 – QoS Interrupt Endpoint Descriptor Content in the PHDC spec.	3.1.6		Dev, Host
5.2.3#7	If the Meta-Data Message Preamble feature is not implemented *bmLatencyReliability* **shall** point to only one QoS bin, which **shall** be equal to the QoS for all data on that endpoint.	3.1.7		Dev, Host
Subsection reference: 5.2.4 PHDC Meta-Data Descriptor				
5.2.4#1	Each PHDC QoS Descriptor **may** be followed by a PHDC MetaData Descriptor that encapsulates opaque meta-data being sent over that	3.1.3		Dev, Host

Assertion #	Assertion Description	Test #	Comment	UUT
	endpoint.			
5.2.4#2	The format of [the PHDC MetaData] descriptor **shall** be as defined in Table 10 – PHDC Meta-Data Descriptor in the PHDC spec.	3.1.3		Dev, Host
5.2.4#3	The *bLength* field **shall** define the size of this meta-data descriptor, based on the size of the opaque data to be sent. *bLength* **shall** be between 2 and 255. FYI – It **should** be < 32 bytes for best performance on all hosts.	3.1.3		Dev, Host
Chapter 6 Test Assertions:				
Subsection reference: 6 Sending Meta-Data Using the PHDC				
Subsection reference: 6 Sending Meta-Data Using the PHDC				
6#1	[The *Meta-Data Message Preamble* feature] is **optional** for a device manufacturer to implement	3.4.4	If it is implemented, the following assertions need to be tested.	Dev
6#2	A host **shall** always support [the *Meta-Data Message Preamble]* feature.	3.3		Host
6#3	The following [test cases] **shall** apply to Bulk OUT and Bulk IN endpoints. They **shall not** apply to Interrupt IN endpoints which always send low latency and good reliability data.	3.3.1		Dev, Host
6#4	The *Meta-Data Message Preamble* feature **shall** initially be disabled.	3.4.3		Dev, Host
Subsection reference: 6.1 Meta-Data Message Preamble Transfer				
6.1#1	During the *Meta-Data Message Preamble* transfer, a packet containing meta-data information as defined in "Table 11 -- Meta-Data Message Preamble" **shall** be sent.	3.3.2 (Dev)		Dev, Host
6.1#2	The *aSignature* field is a constant and **shall** be set to the 16 byte string "PhdcQoSSignature".	3.3.2 (Dev)		Dev, Host
6.1#3	*bNumTransfers* **shall** never equal zero.	3.3.2 (Dev)		Dev, Host
6.1#4	The *bQoSEncodingVersion* field **shall** contain 01h.	3.3.2 (Dev)		Dev, Host
6.1#5	The *bmLatencyReliability* field **shall** contain a bitmap specifying the QoS information for the transfers that follow. In this case, only one bit **shall** be set in any given bmLatencyReliability field (refer to Section 6.3 for behavior on an invalid	3.3.2 (Dev)		Dev, Host

Assertion #	Assertion Description	Test #	Comment	UUT
	bmLatencyReliability value.).			
6.1#6	The *bOpaqueDataSize* field **shall** contain the number of bytes of opaque QoS or meta-data that are to be transferred in the Meta-Data Message Preamble. Note: The amount of opaque data cannot exceed the endpoint maximum packet size minus 21 bytes to insure short-packet reception and request completion upon receipt of the preamble.	3.3.2 (Dev)		Dev, Host
6.1#7	The *bOpaqueData* field **shall** contain the opaque QoS or meta-data to be transferred.	3.3.2 (Dev)	Can only test for existence of data.	Dev, Host
Subsection reference: 6.2 Data Transfer				
6.2#1	After a *Meta-Data Message Preamble* transfer, a count of *bNumTransfers* data transfers **shall** be sent.	3.3.3 (Dev)		Dev, Host
6.2#2	The first transfer following *bNumTransfers* data transfers **shall** be preceded by new Meta Data Message Preamble.	3.3.3 (Dev)		Dev, Host
Subsection reference: 6.3 Meta-Data Message Preamble Feature Error Conditions				
6.3#1	If a *Meta-Data Message Preamble* is expected but not received by the device, the device **shall** STALL subsequent transactions to the receiving endpoint until the host clears the stall condition.	3.3.4	Invalid message preamble signature condition	Dev
6.3#2	If the device receives a *Meta-Data Message Preamble* containing an invalid *bmLatencyReliability* value, the device **shall** STALL subsequent transactions to the receiving endpoint until the host clears the stall condition.	3.3.5		Dev
6.3#3	If the device receives a *Meta-Data Message Preamble* containing an invalid *bNumTransfers* value, the device **shall** STALL subsequent transactions to the receiving endpoint until the host clears the stall condition.	3.3.6		Dev
6.3#4	To clear a device STALL condition the host **shall** issue a ClearFeature(ENDPOINT_HALT).			Host
6.3#5	If a *Meta-Data Message Preamble* is expected but not received by the host, the host **shall** issue a SET_FEATURE ENDPOINT_HALT request to the device.			Host
6.3#6	If the host receives a *Meta-Data Message Preamble* containing an			Host

Assertion #	Assertion Description	Test #	Comment	UUT
	invalid *bmLatencyReliability* value, the host **shall** issue a SET_FEATURE ENDPOINT_HALT request to the device.			
6.3#7	If the host receives a *Meta-Data Message Preamble* containing an invalid *bNumTransfers* value, the host **shall** issue a SET_FEATURE ENDPOINT_HALT request to the device.			Host
6.3#8	To clear a host issued SET_FEATURE (ENDPOINT_HALT) the host **shall** issue a CLEAR_FEATURE (ENDPOINT_HALT).			Host
6.3#9	After the host has issued the CLEAR_FEATURE (ENDPOINT_HALT), the first data transfer a device **shall** send is a *Meta-Data Message Preamble* transfer, and the host **shall** expect to receive a *Meta-Data Message Preamble* transfer.	3.3.7		Dev, Host
6.3#10	After the host has issued the CLEAR_FEATURE (ENDPOINT_HALT), the first data transfer a host **shall** send is a *Meta-Data Message Preamble* transfer, and the device **shall** expect to receive a *Meta-Data Message Preamble* transfer.	3.3.7		Dev, Host
Subsection reference: 6.4 Determining QoS				
6.4#1	If *bmCapability:b0 = 0* in the *PHDC Class Function Descriptor* then the *bmLatencyRelaibility:b5-b0* field of the *PHDC QoS Descriptor* **shall** have one and only one bit set.	3.1.7		Dev
6.4#2	If *bmCapability:b0 = 1* in the *PHDC Class Function Descriptor* then the *bmLatencyRelaibility:b5-b0* field of the *PHDC QoS Descriptor* **shall** have at least one bit set and **may** have 2 to 6 bits set.	3.1.3		Dev
6.4#3	If *<Meta-Data Message Preamble is disabled>* then **no** *Meta Data Message Preamble* transfers **shall** be sent.	3.3.8		Dev, Host
Chapter 7 Test Assertions:				
Subsection reference: 7 Requests				
Subsection reference: 7.1 Class-Specific Requests				
Subsection reference: 7.1.1 Set/Clear Meta-Data Message Preamble Feature				

Assertion #	Assertion Description	Test #	Comment	UUT
7.1.1#1	The *Meta-Data Message Preamble* feature **shall** be turned on and off via class-specific extensions to the SET_FEATURE and CLEAR_FEATURE requests, respectively.	SET_FEATURE: 3.3.4 through **Error! Reference source not found.** 3.4.5 3.4.9		Dev, Host
7.1.1#2	If the device does not support the *Meta-Data Message Preamble* feature, then it **shall** respond to these requests with a stall, and the host **shall** clear it	3.4.4 3.4.11	This is an optional device feature	Dev, Host
7.1.1#3	The *Meta-Data Message Preamble* feature **shall** be off by default	**Error! Reference source not found.** 3.4.3 3.4.9		Dev, Host
7.1.1#4	The *Meta-Data Message Preamble* feature **shall** require the SET_FEATURE command to turn it on as shown in Table 12 Set/Clear PHDC Meta-Data Message Preamble Feature Requests of the PHDC spec.	3.3.4 3.4.5 3.4.9		Dev, Host
7.1.1#5	The host **should** ensure there are no pending transactions on the device before issuing a SET_FEATURE or CLEAR_FEATURE (FEATURE_PHDC_METADATA) command to avoid synchronization issues with the preamble.	N/A	This is not a testable assertion, and is more of a prerequisite to testing	Host
7.1.1#6	The syntax for these requests **shall** be implemented as shown in Table 12 – Set/Clear PHDC Meta-Data Message Preamble Feature Requests.	3.4.5 3.4.8 3.4.9		Dev, Host
7.1.1#7	The device and host **shall** encode and decode *Meta-Data Message Preamble* messages using this *bQoSEncodingVersion*.	3.4.5 3.4.9		Dev, Host
7.1.1#8	If the host only supports a *bQoSEncodingVersion* of 01h, it **shall** use the SET_FEATURE command to enforce this.	3.4.9		Host
7.1.1#9	If the device only supports a	3.1.3		Host

Assertion #	Assertion Description	Test #	Comment	UUT
	bQoSEncodingVersion of 01h, the host will know this from the *PHDC QoS Descriptor* (refer to Section 5.2.3 of the PHDC spec), and the host **shall** use the SET_FEATURE command to enforce a *bQoSEncodingVersion* of 01h.	3.4.9		
7.1.1#10	The *wIndex* value **shall** specify the target interface for the request.	3.4.5 3.4.9		Dev, Host
7.1.1#11	If either feature request specifies an interface that doesn't exist, the device **shall** respond with a Request Error.	3.4.6 3.4.7		Dev
7.1.1#12	Upon receipt of the SET_FEATURE (FEATURE_PHDC_METADATA), the device **shall** turn on the *Meta-Data Message Preamble* feature.	3.3.4 3.4.5		Dev
7.1.1#13	Upon receipt of the CLEAR_FEATURE (FEATURE_PHDC_METADATA), the device **shall** turn off the *Meta-Data Message Preamble* feature.	3.3.4 3.4.5		Dev
7.1.1#14	Upon the completion of a successful status phase for the set or clear operation, the device **shall** be in the state specified.	3.4.5		Dev
Subsection reference: 7.1.2 Get Data Status				
7.1.2#1	Devices and hosts **shall** support the class-defined Get Data Status request as defined in Table 13 – Get Data Status of the PHDC spec.	3.4.1 3.4.8 3.4.10		Dev, Host
7.1.2#2	When a device receives the request, it **shall** return a bitmap corresponding to the endpoints holding data on the device as described in Table 14 – Information Returned from a PHDC Class-defined Get Data Status Request.	3.4.1		Dev
7.1.2#3	The *wIndex* value **shall** specify the target interface for the request.	3.4.1 3.4.2 3.4.10		Dev, Host
7.1.2#4	If the request specifies an interface that doesn't exist, the device **shall** respond with a Request Error.	3.4.2		Dev
Chapter 8 Test Assertions:				
Subsection reference: 8 Descriptor, Request, & Feature Types Table				
8#1	The Feature Type value **shall** be in the low-order byte of the word (per			Dev, Host

Assertion #	Assertion Description	Test #	Comment	UUT
	USB 2.0 conventions).			
Chapter Appendix A:				
Subsection reference: Appendix A – 1 Introduction				
Subsection reference: Appendix A – 2 PHDC Class Function Descriptor Data/Messaging Codes				
A.2#1	The valid values for the PHDC Class Function descriptor's bPHDCDataCode are listed in Table 18 – Personal Healthcare Data and Messaging Formats.	3.5.1		Dev, Host
Subsection reference: Appendix A – 3 Vendor-Defined Extensions				
A.3#1	If a device manufacturer decides to send a vendor-defined data and messaging format over the PHDC, then bPHDCDataCode **shall** be equal to PHDC_VENDOR.	3.5.1		Dev, Host
A.3#2	[If a device manufacturer decides to send a vendor-defined data and messaging format over the PHDC,] the vendor may define any number of vendor-specific descriptors or other functionality and **shall** enable support for these features in vendor-specific host drivers.	Part of checklist.		Dev, Host
Subsection reference: Appendix A – 4 11073 PHD Specific Extensions				
A.4#1	If a device implements the 11073 PHD extensions, then bPHDCDataCode **shall** be equal to PHDC_11073_20601.	3.5.1		Dev, Host
A.4#2	The descriptor defined in Table 19 – PHDC 11073 PHD Function Extension Descriptor **shall** follow the PHDC Class Function descriptor and precede any endpoint descriptors.	3.5.2		Dev, Host
A.4#3	The format of [the PHDC 11073 PHD Function Extension descriptor] **shall** be as defined in Table 19 – PHDC 11073 PHD Function Extension Descriptor.	3.5.2		Dev, Host
A.4#4	If the bLength value is not an even number, greater than or equal to 4, then it is invalid	3.5.2		Dev, Host
A.4#5	The valid values of bNumDevSpecs are: 0 (indicates no device specializations), 1 (indicates a single device specialization), >1 (indicates a multi-function device).	3.5.3		Dev, Host

Assertion #	Assertion Description	Test #	Comment	UUT
A.4#6	The wDevSpecializations field is a Word or Words that contain the little endian 16-bit value corresponding to the device specialization(s) supported by the device. These values **shall** be as defined in the ISO/IEEE 11073-20601™ PHD specifications, Nomenclature Codes Annex for device Specializations with the caveat that, if there is a discrepancy between ISO/IEEE 11073-20601™ and ISO/IEEE 11073-10101™ then ISO/IEEE 11073-10101™ **shall** be used.	3.5.3		Dev, Host
A.4#7	If bNumDevSpecs is 0, then there **shall** be no wDevSpecializations fields.	3.5.2		Dev, Host
Subsection reference: Appendix A – 5 Descriptor Constants				
A.5#1	Descriptor constants for Appendix A are as defined in Table 20 – Appendix A Descriptor Constants.	3.5.2		Dev, Host

3. Test Descriptions

The following sections list the test descriptions. These cover both device and host tests and list the assumptions about the host that need to be fulfilled in order to make these tests.

3.1 Enumeration

3.1.1 Verify standard USB descriptors

This test verifies that the device implements or host recognizes the standard USB descriptors as defined for the PHDC. It also verifies that the correct number of endpoint descriptors is received.

Covered Assertions

4.4.1#1,

4.4.2#1,

5.1.1#1 through 5.1.1#3,

5.1.3#1 through 5.1.3#4,

Required Testing Resource

None

Preconditions

The device and the host are powered up and running. The device and host have not yet been connected via a USB cable; however, the host and device have been introduced previously (drivers installed, inf files read, etc.).

Overview of Test Steps

The test software performs the following steps.

1. Connect the device to the host via a USB cable.
2. Verify on the host that the device descriptor exists and contains the following values:
   bDeviceClass = 00h (preferred) or 0Fh (also acceptable) [5.1.1#1]
   bDeviceClass = 0Fh if bInterfaceClass = 00h [5.1.1#1]
   bDeviceSubclass = 00h [5.1.1#2]
   bDeviceProtocol = 00h [5.1.1#3]
3. Verify on the host that the configuration descriptor exists.
4. Verify on the host that the interface descriptor for PHDC exists and contains the following values:
   bNumEndpoints = 02h or greater [4.4.1#1, 4.4.2#1]
   bInterfaceClass = 0Fh (preferred) [5.1.1#1]
   bInterfaceClass = 0Fh if bNumInterfaces = 02h or greater [5.1.3#1]
   bInterfaceClass = 0Fh if bDeviceClass = 00h [5.1.1#1]
   bInterfaceSubclass = 00h [5.1.3#2]
   bInterfaceProtocol = 00h [5.1.3#3]
5. Verify on the host that a bulk OUT endpoint descriptor and bulk IN endpoint descriptor exist. [4.4.1#1, 4.4.2#1]

3.1.2 Verify multiple personal healthcare functions

This test is run if a device implements multiple personal healthcare functions. This test verifies that, if a device implements multiple personal healthcare functions, the device uses and/or the host recognizes one of the two specified mechanisms.

Covered Assertions

5.1.3#4

Required Testing Resource

<TBD>

Preconditions

The device and the host are powered up, running, and connected. The device claims to be a multi-function device because it has set PHDC 11073 PHD Function Extension descriptor's bNumDevSpecs to 02h or greater.

Overview of Test Steps

The test software performs the following steps.

1. Verify on the host that the number of interfaces is greater than or equal to the number of device specializations (if using one interface per device specialization) or is greater than or equal to 1 (if using one interface for all device specializations). That is,
   Standard Configuration descriptor's bNumInterfaces >= PHDC 11073 PHD Function Extension descriptor's bNumDevSpecs
   or
   Standard Configuration descriptor's bNumInterfaces >= 01h [5.1.3#4]

3.1.3 Verify class-defined USB descriptors

This test verifies that the device implements or host recognizes the PHDC Class Function descriptor, the PHDC 11073 PHD Function Extension descriptor, and the PHDC QoS descriptor. Verify that if the device implements the meta-data descriptor, then the host recognizes it.

Covered Assertions

5.2.1#1, 5.2.1#2, 5.2.1#3

5.2.3#1, 5.2.3#2, 5.2.3#3

5.2.4#1, 5.2.4#2, 5.2.4#3

6.4#2

Required Testing Resource

<TBD>

Preconditions

The device and the host are powered up and running. The device and host have not yet been connected via a USB cable; however, the host and device have been introduced previously (drivers installed, inf files read, etc.).

Overview of Test Steps

The test software performs the following steps.

1. Connect the device to the host via a USB cable.
2. Verify on the host that the PHDC Class Function descriptor follows a standard interface descriptor. [5.2.1#2]
3. Verify on the host that the PHDC Class Function descriptor exists and contains the following values: [5.2.1#1]
   bLength = 04h
   bDescriptorType = PHDC_CLASSFUNCTION_DESCRIPTOR (20h)
   bPHDCDataCode = PHDC_11073_20601 (02h)
   bmCapability = 00h or 01h [5.2.1#3]
4. Verify on the host that each standard endpoint descriptor is followed by a QoS descriptor. [5.2.3#1]
5. Verify on the host that the PHDC QoS descriptor exists and contains the following values: [5.2.3#2]
   bLength = 04h
   bDescriptorType = PHDC_QOS_DESCRIPTOR
   bQoSEncodingVersion = 01h [5.2.3#3]
   bmLatencyReliability = 02h through 3Fh for bulk endpoints; 01h for interrupt endpoints
6. If the device implements the Meta-Data descriptor, verify on the host that it follows the QoS descriptor. [5.2.4#1]
7. If the device implements the Meta-Data descriptor, verify on the host that the Meta-Data descriptor exists and contains the following values: [5.2.4#2]
   bLength = 2 to 255 [5.2.4#3]
   bDescriptorType = PHDC_METADATA_DESCRIPTOR
   bOpaqueData = <no specific content to test for, but size is bLength-2>

3.1.4 Verify Valid bQosEncoding Version

HOST-ONLY TEST: This test verifies that a host will ignore a QoS descriptor when bQoSEncoding version is greater than 01h. "Ignore" in this sense means it continues to parse through the rest of the endpoints sent.

Covered Assertions

5.2.3#4

Required Testing Resource

Need a device that can simulate a QoS descriptor with a bQoSEncodingVersion > 01h and a bmLatencyReliability of 4. This QoS descriptor should be followed by a QoS descriptor with bQoSEncodingVersion = 01h and a bmLatencyReliability of 8.

Preconditions

The device and the host are powered up and running. The device and host have not yet been connected via a USB cable; however, the host and device have been introduced previously (drivers installed, inf files read, etc.).

Overview of Test Steps

The test software performs the following steps.

1. Connect the device to the host via a USB cable.
2. Verify that the host displays only one QoS descriptor with values:

   bQoSEncodingVersion = 01h
   bmLatencyReliability = 8

3.1.5 Verify Latency Reliability Encoding - BULK

This test verifies that the bmLatencyReliability bitmap is set for bulk endpoints. It cannot verify the accuracy of the bitmap, but it can verify that the bits reserved for interrupt endpoints are not set.

Covered Assertions

5.2.3#5

Required Testing Resource

<TBD>

Preconditions

The device and the host are powered up and running. The device and host have not yet been connected via a USB cable; however, the host and device have been introduced previously (drivers installed, inf files read, etc.).

Overview of Test Steps

The test software performs the following steps.

1. Connect the host and device via a USB cable.
2. For all bulk OUT and IN endpoints, verify on the host that the device displays the following information:

   bQoSEncodingVersion = 01h

   If testing a device,

   bmLatencyReliability > 1 and < 1E

   bmLatencyReliability == an even number (i.e., bit 0 not set)

   If testing a host,

   bmLatencyReliability = values sent from device simulator

3.1.6 Verify Latency Reliability Encoding - INTERRUPT

This test verifies that the bmLatencyReliability bitmap is correctly set for interrupt endpoints.

Covered Assertions

4.4.1#2, 5.2.3#6

Required Testing Resource

Need a test device with interrupt endpoints.

Preconditions

The device and the host are powered up and running. The device and host have not yet been connected via a USB cable; however, the host and device have been introduced previously (drivers installed, inf files read, etc.).

This test is only applicable to hosts and devices with interrupt endpoints.

Overview of Test Steps

The test software performs the following steps.

1. Connect the host and device via a USB cable.
2. For all bulk OUT and IN endpoints, verify on the host that the device displays the following information:

   bQoSEncodingVersion = 01h

   bmLatencyReliability = 01h

3.1.7 Verify Latency/Reliability map with no Meta-Data Message Preamble

This test verifies that the bmLatencyReliability field is equal to a single QoS value when the Meta-Data Message Preamble feature is not available.

Covered Assertions

5.2.3#7, 6.4#1

Required Testing Resource

Need a test device that does not implement the Meta-Data Message Preamble feature.

Preconditions

The device and the host are powered up and running. The device and host have not yet been connected via a USB cable; however, the host and device have been introduced previously (drivers installed, inf files read, etc.).

This test is only applicable to hosts and devices which indicate that it does not support the Meta-Data Preamble feature by clearing bmCapability bit 0 in the PHDC Class Function Descriptor. (If not, this test shall not be run.)

Overview of Test Steps

The test software performs the following steps.

1. Connect the host and device via a USB cable.
2. Verify on the host that the device sends the following values in the PHDC Class Function Descriptor:

   bmCapability = 0

3. Verify on the host that the device sends the following values in the PHDC QoS Descriptor for all endpoints:

   bmLatencyReliability = 1 if endpoint is an interrupt endpoint

   bmLatencyReliability = 2, 4, 8, 16, or 32 if endpoint is a bulk endpoint

3.2 Data Transfers

This should cover data transfers over all endpoints with Meta-Data Message Preamble disabled. All data types in chapter 4 should be sent.

3.2.1 Verify basic communication on bulk endpoints

This test verifies that devices and hosts can send data and messages over bulk endpoints.

Covered Assertions

4.4.1#1, 4.4.2#1

Required Testing Resource

For hosts that implement the 11073 PHD data and messaging format, this test requires a device that can successfully perform 11073 Association. For devices that implement the 11073 PHD data and messaging format, this test requires a host that can successfully perform 11073 Association.

TODO: Add information for vendor-specific data and messaging format testing.

Preconditions

The device and the host are powered up and running and are connected via a USB cable. Enumeration has already occurred. There are no pending transactions on the device or host. Device and host have not performed 11073 Association.

Overview of Test Steps

The test software performs the following steps.

1. Initiate the 11073 Association Request message from the device (on bulk IN endpoint).
2. Ensure that the host receives the Association Request from the device (on bulk IN endpoint).
3. Ensure that the device receives an 11073 Association Response from the host (on bulk OUT endpoint).

3.2.2 Verify device-driven communication

HOST-ONLY TEST: This test verifies that devices can drive communication with the host. Whether the host does this through a queued read request or a Get Data Status request is not necessary to test.

Covered Assertions

4.4.1#4

Required Testing Resource

For hosts that implement the 11073 PHD data and messaging format, this test requires a device that can successfully perform 11073 Association.

TODO: Add information for vendor-specific data and messaging format testing.

Preconditions

The device and the host are powered up and running and are connected via a USB cable. Enumeration has already occurred. There are no pending transactions on the device or host. Device and host have not performed 11073 Association.

Overview of Test Steps

The test software performs the following steps.

1. Initiate the 11073 Association Request message from the device (on bulk IN endpoint).
2. Ensure that the device receives an 11073 Association Response from the host (on bulk OUT endpoint).

3.2.3 Verify basic communication on interrupt endpoints

This test verifies that devices that implement interrupt endpoints can send data over the endpoint and that hosts can read the data.

This test is only run on devices that support interrupt endpoints.

Covered Assertions

4.4.1#2

Required Testing Resource

For hosts that implement the 11073 PHD data and messaging format, this test requires a device that can successfully perform 11073 Association. For devices that implement the 11073 PHD data and messaging format, this test requires a host that can successfully perform 11073 Association.

TODO: Add information for vendor-specific data and messaging format testing.

Preconditions

The device and the host are powered up and running and are connected via a USB cable. Enumeration has already occurred. There are no pending transactions on the device or host. Device and host have not performed 11073 Association.

Overview of Test Steps

The test software performs the following steps.

1. Perform Association and Configuration on the bulk IN and bulk OUT endpoints as in 3.2.1.
2. From the host, post a read on the interrupt IN endpoint.
3. Initiate a transfer from the interrupt IN endpoint. <Note: If the transfer cannot be initiated, exit the test with a message that interrupt communication could not be tested.>
4. Ensure that 11073 measurement data is received.

3.3 Meta-Data Message Preamble Feature

These are tests of the Meta-Data Preamble feature. These tests need to be run on all hosts. These tests only need to be run on devices that indicate they support this feature.

Currently, this document only contains the device tests for this feature. The host tests will most likely not be done by USB-IF at this time since USB-IF does not typically test hosts.

3.3.1 Verify No Meta-Data Preamble on Interrupt EP

DEVICE-ONLY TEST: This test verifies that the interrupt EP does not implement the meta-data preamble feature. It does this by posting a read and ensuring that the response received is not a meta-data preamble.

This test is only applicable to devices that support the optional interrupt endpoint. If a device does not support the optional interrupt endpoint, this test shall not be run.

Covered Assertions

6#3

Required Testing Resource

For hosts that implement the 11073 PHD data and messaging format, this test requires a device that can successfully perform 11073 Association. For devices that implement the 11073 PHD data and messaging format, this test requires a host that can successfully perform 11073 Association.

TODO: Add information for vendor-specific data and messaging format testing.

Preconditions

The device and the host are powered up and running and are connected via a USB cable. Enumeration has already occurred. There are no pending transactions on the device or host. Device and host have not performed 11073 Association.

Device has indicated it supports the Meta-Data Preamble feature by setting bmCapability bit 0 to 1 in the PHDC Class Function Descriptor. (If not, this test shall not be run.)

Overview of Test Steps

The host test software performs the following steps.

1. Initiate the 11073 Association Request message from the device (on bulk IN endpoint).
2. Ensure that the host receives the 11073 Association Request from the device (on bulk IN endpoint).
3. From the host, issue a SET_FEATURE(FEATURE_PHDC_METADATA) request to the device to enable Meta-Data Preamble.
4. Initiate a Meta-Data Preamble plus 11073 Association Response from the host (on bulk OUT endpoint).
5. From the host, post a read on the interrupt IN endpoint.
6. Initiate a transfer from the interrupt IN endpoint. <Note: If the transfer cannot be initiated, exit the test with a message that interrupt communication could not be tested.>
7. Ensure that 11073 measurement data is received, but not a Meta-Data Preamble.

3.3.2 Verify device sends valid Meta-Data Preamble

DEVICE-ONLY TEST: This test verifies that the device sends a valid meta-data preamble to the host.

Covered Assertions

6.1#1, 6.1#2, 6.1#3, 6.1#4, 6.1#5, 6.1#6, 6.1#7

Required Testing Resource

For devices that implement the 11073 PHD data and messaging format, this test requires a host that can successfully perform 11073 Association.

TODO: Add information for vendor-specific data and messaging format testing.

Preconditions

The device and the host are powered up and running and are connected via a USB cable. Enumeration has already occurred. There are no pending transactions on the device or host. Device and host have not performed 11073 Association.

Device has indicated it supports the Meta-Data Preamble feature by setting bmCapability bit 0 to 1 in the PHDC Class Function Descriptor. (If not, this test shall not be run.)

Overview of Test Steps

The host test software performs the following steps.

1. Initiate the 11073 Association Request message from the device (on bulk IN endpoint).
2. Ensure that the host receives the 11073 Association Request from the device (on bulk IN endpoint).
3. From the host, issue a SET_FEATURE(FEATURE_PHDC_METADATA) request to the device to enable Meta-Data Preamble.
4. Initiate a Meta-Data Preamble plus 11073 Association Response from the host (on bulk OUT endpoint).
5. From the host, issue a Meta-Data Preamble plus Get MDS Object request to the device.
6. Ensure that the host receives a Meta-Data Preamble plus MDS Object response from the device. Ensure the Meta-Data Preamble format is as follows:
   aSignature = PhdcQoSSignature (6.1#2)
   bNumTransfers = number of transfers for MDS Object response (must be > 0; most likely will be 01h) (6.1#3)
   bQoSEncodingVersion = 01h (6.1#4)
   bmLatencyReliability = 2, 4, 8, 16, or 32 (6.1#5)
   bOpaqueDataSize = number from 0 to MaxPacket-21 (6.1#6)
   bOpaqueData = data of size bOpaqueDataSize (6.1#7)

3.3.3 Verify valid bNumTransfers

DEVICE-ONLY TEST: Verify that a device sends a valid bNumTransfers value that correctly indicates the number of transfers following. Verify that after bNumTransfers, a new Meta-Data Preamble is sent.

Covered Assertions

6.2#1, 6.2#2

Required Testing Resource

For devices that implement the 11073 PHD data and messaging format, this test requires a host that can successfully perform 11073 Association.

TODO: Add information for vendor-specific data and messaging format testing.

Preconditions

The device and the host are powered up and running and are connected via a USB cable. Enumeration has already occurred. There are no pending transactions on the device or host. Device and host have not performed 11073 Association.

Device has indicated it supports the Meta-Data Preamble feature by setting bmCapability bit 0 to 1 in the PHDC Class Function Descriptor. (If not, this test shall not be run.)

Overview of Test Steps

The host test software performs the following steps.

1. Initiate the 11073 Association Request message from the device (on bulk IN endpoint).
2. Ensure that the host receives the 11073 Association Request from the device (on bulk IN endpoint).
3. From the host, issue a SET_FEATURE(FEATURE_PHDC_METADATA) request to the device to enable Meta-Data Preamble.
4. Initiate a Meta-Data Preamble plus 11073 Association Response from the host (on bulk OUT endpoint).
5. From the host, issue a Meta-Data Preamble plus Get MDS Object request to the device (on bulk OUT endpoint).
6. From the host, issue a read on the device (on bulk IN endpoint).
7. Ensure that a Meta-Data Preamble is received from the device (on bulk IN endpoint).
8. From the host, issue a read bNumTransfers times. (Note: If bNumTransfers is > 1, then multiple GetMDSObject requests may need to be sent to the device in order to receive bNumTransfers worth of data. Alternatively, this test could quit as "untestable" if bNumTransfers is sufficiently large.)
    a. Ensure data is received from the device each time. (6.2#1)
9. From the host, issue a Meta-Data Preamble plus Get MDS Object request to the device.
10. From the host, issue a read on the device.
11. Ensure that a Meta-Data Preamble is received from the device. (6.2#2)

3.3.4 Verify device STALLs when no expected *Meta-Data Message Preamble* transfer is received

DEVICE-ONLY TEST: This test verifies that a device will issue a STALL when it expects, but doesn't receive, a *Meta-Data Message Preamble* transfer.

Covered Assertions

6.3#1, 6.3#4

Required Testing Resource

For devices that implement the 11073 PHD data and messaging format, this test requires a host that can successfully perform 11073 Association.

TODO: Add information for vendor-specific data and messaging format testing.

Preconditions

The device and the host are powered up and running and are connected via a USB cable. Enumeration has already occurred. There are no pending transactions on the device or host. Device and host have not performed 11073 Association.

Device has indicated it supports the Meta-Data Preamble feature by setting bmCapability bit 0 to 1 in the PHDC Class Function Descriptor. (If not, this test shall not be run.)

Overview of Test Steps

The host test software performs the following steps.

1. Initiate the 11073 Association Request message from the device (on bulk IN endpoint).
2. Ensure that the host receives the 11073 Association Request from the device (on bulk IN endpoint).
3. From the host, issue a SET_FEATURE(FEATURE_PHDC_METADATA) request to the device to enable Meta-Data Preamble.
4. Initiate a Meta-Data Preamble plus 11073 Association Response from the host (on bulk OUT endpoint).
5. From the host, do not issue a Meta-Data Preamble. Instead, only issue a Get MDS Object request to the device (on bulk OUT endpoint). [Simulating error condition of Meta-Data Preamble expected, but not received.]
6. Ensure that the device issues a STALL on subsequent bulk OUT endpoint transactions by sending Get MDS Object request message. (6.3#1)
7. Issue a CLEAR_FEATURE ENDPOINT_HALT from the host to clear the STALL. (6.3#4)

3.3.5 Verify device STALLs on invalid *bmLatencyReliability* value

DEVICE-ONLY TEST: This test verifies that a device will issue a STALL when it receives a *Meta-Data Message Preamble* transfer with an invalid *bmLatencyReliability* value.

Covered Assertions

6.3#2

Required Testing Resource

For devices that implement the 11073 PHD data and messaging format, this test requires a host that can successfully perform 11073 Association.

TODO: Add information for vendor-specific data and messaging format testing.

Preconditions

The device and the host are powered up and running and are connected via a USB cable. Enumeration has already occurred. There are no pending transactions on the device or host. Device and host have not performed 11073 Association.

Device has indicated it supports the Meta-Data Preamble feature by setting bmCapability bit 0 to 1 in the PHDC Class Function Descriptor. (If not, this test shall not be run.)

Overview of Test Steps

The host test software performs the following steps.

1. Initiate the 11073 Association Request message from the device (on bulk IN endpoint).
2. Ensure that the host receives the 11073 Association Request from the device (on bulk IN endpoint).

3. From the host, issue a SET_FEATURE(FEATURE_PHDC_METADATA) request to the device to enable Meta-Data Preamble.
4. Initiate a Meta-Data Preamble plus 11073 Association Response from the host (on bulk OUT endpoint).
5. From the host, issue a Meta-Data Preamble where bmLatencyReliability = C0h (invalid value) plus Get MDS Object request to the device (on bulk OUT endpoint).
6. Ensure that the device issues a STALL on the bulk OUT endpoint. (6.3#2)
7. Issue a CLEAR_FEATURE ENDPOINT_HALT from the host to clear the STALL.

3.3.6 Verify device STALLs on invalid *bNumTransfers* value

DEVICE-ONLY TEST: This test verifies that a device will issue a STALL when it receives a *Meta-Data Message Preamble* transfer with an invalid *bNumTransfers* value.

Covered Assertions

6.3#3

Overview of Test Steps

Repeat the steps in test 3.3.5, except in step 5, bmLatency to a valid value and bNumTransfers to 0.

3.3.7 Verify device sends Meta-Data Preamble after a STALL

DEVICE-ONLY TEST: Verify that, after the host has issued the CLEAR_FEATURE (ENDPOINT_HALT), the first data transfer a device sends is a *Meta-Data Message Preamble* transfer, and also the first data transfer a device expects to receive is a *Meta-Data Message Preamble* transfer.

Covered Assertions

6.3#9, 6.3#10

Required Testing Resource

For devices that implement the 11073 PHD data and messaging format, this test requires a host that can successfully perform 11073 Association.

TODO: Add information for vendor-specific data and messaging format testing.

Preconditions

The device and the host are powered up and running and are connected via a USB cable. Enumeration has already occurred. There are no pending transactions on the device or host. Device and host have not performed 11073 Association.

Device has indicated it supports the Meta-Data Preamble feature by setting bmCapability bit 0 to 1 in the PHDC Class Function Descriptor. (If not, this test shall not be run.)

Overview of Test Steps

The host test software performs the following steps.

1. Initiate the 11073 Association Request message from the device (on bulk IN endpoint).
2. Ensure that the host receives the 11073 Association Request from the device (on bulk IN endpoint).
3. From the host, issue a SET_FEATURE(FEATURE_PHDC_METADATA) request to the device to enable Meta-Data Preamble.
4. Initiate a Meta-Data Preamble plus 11073 Association Response from the host (on bulk OUT endpoint).
5. From the host, issue a Meta-Data Preamble where bNumTransfers = 0 (invalid value) plus Get MDS Object request to the device (on bulk OUT endpoint).

6. Ensure that the device issues a STALL on the bulk OUT endpoint. (6.3#3)
7. Issue a CLEAR_FEATURE ENDPOINT_HALT from the host to clear the STALL.
8. From the host, issue a valid Meta-Data Preamble plus Get MDS Object request to the device (on bulk OUT endpoint).
9. Ensure host receives a Meta-Data Preamble plus MDS Object response from the device (on bulk IN endpoint).

3.3.8 Verify Meta-Data Preamble not sent when feature disabled

DEVICE-ONLY TEST: This test verifies that the device does not send a meta-data preamble when that feature is disabled.

Covered Assertions

6.4#3

Required Testing Resource

For devices that implement the 11073 PHD data and messaging format, this test requires a host that can successfully perform 11073 Association.

TODO: Add information for vendor-specific data and messaging format testing.

Preconditions

The device and the host are powered up and running and are connected via a USB cable. Enumeration has already occurred. There are no pending transactions on the device or host. Device and host have not performed 11073 Association.

Device has indicated it supports the Meta-Data Preamble feature by setting bmCapability bit 0 to 1 in the PHDC Class Function Descriptor. (If not, this test shall not be run.)

Overview of Test Steps

The host test software performs the following steps.

1. Before initiating the 11073 Associating Request from the device, issue a CLEAR_FEATURE(FEATURE_PHDC_METADATA) request to the device to disable Meta-Data Preamble.
2. Initiate the 11073 Association Request message from the device (on bulk IN endpoint).
3. Ensure that the host receives the 11073 Association Request from the device (on bulk IN endpoint) with no Meta-Data Preamble preceding it.

3.4 Requests

These are tests of the Device Class-Specific Requests. These tests need to be run on all the devices. If device is issues a Request Error or a STALL on an invalid class request on the control endpoint, the STALL shall be cleared on the next control endpoint set-up request as specified by the USB 2.0 Specification.

3.4.1 Verify Get Data Status request

This test verifies the device implements the Get Data Status request as specified in the PHDC spec.

Covered Assertions

7.1.2#1, 7.1.2#2, 7.1.2#3

Required Testing Resource

11073-20601 Agent or equivalent

Preconditions

This test is run with the PHD in Configured state, and the 11073-20601 agent in the Associating state. The Host does NOT queue any reads to the endpoint under test, as stated in section 4.4.1 of the PHDC spec.

Overview of Test Steps

The test software performs the following steps.

1. Place the PHDC in Configured state.
2. Wait for receipt of an 11073-20601 association request.
3. Before issuing the association response, send a Get Data Status request to the unit under test.
    a. bmRequestType = A1h
    b. bRequest = 00h
    c. wValue = 0000h
    d. wIndex = PHDC interface number
    e. wLength = 2
4. Verify that the device sends a 0000h bitmap during the Data stage of the above request.
5. Continue the 11073-20601 association process, until the agent is in the Operating state.
6. Issue a 11073-20601 Get MDS Object.
7. Send a Get Data Status request to the unit under test.
    a. bmRequestType = A1h
    b. bRequest = 00h
    c. wValue = 0000h
    d. wIndex = PHDC interface number
    e. wLength = 2
8. Verify that the device sends a non-zero bitmap during the Data stage of the above request, indicating that it has a pending transaction.
9. Post a read to the endpoint that is indicating a pending transaction.
10. Verify that the data read from the device is a valid response to the Get MDS Object issued earlier in this test.

Repetitions

Repeat test on every interface exposed by the enumeration process.

3.4.2 Verify Get Data Status error conditions

This test verifies the device responds to the Get Data Status request with invalid data, as specified in the PHDC spec.

Covered Assertions

7.1.2#3, 7.1.2#4

Required Testing Resource

None

Preconditions

This test is run with the PHD in Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Place the PHDC in Configured state.
2. Send a Get Data Status request to the unit under test with an invalid wValue parameter.
   - bmRequestType = A1h
   - bRequest = 00h
   - wValue = 0001h (should be 0000h)
   - wIndex = PHDC interface number
   - wLength = 2
3. Verify that the unit under test responds with a Request Error.
4. Send a Get Data Status request to the unit under test with an invalid wIndex parameter.
   - bmRequestType = A1h
   - bRequest = 00h
   - wValue = 0000h
   - wIndex = Invalid PHDC interface number (bNumInterfaces + 1, from Configuration Descriptor)
   - wLength = 2
5. Verify that the unit under test responds with a Request Error.

Repetitions

Repeat test on every interface exposed by the enumeration process.

3.4.3 Verify Meta-Data Message Preamble if OFF by default

This test verifies the device has its Meta-Data Message Preamble turned OFF by default, as specified in the PHDC spec.

Covered Assertions

7.1.1#3, 6#4

Required Testing Resource

11073-20601 Agent or equivalent

Preconditions

This test is run with the PHD in Configured state, and the 11073-20601 agent in the Operating state. The unit under test supports the Meta-Data preamble, and has it set in the appropriate descriptor.

Overview of Test Steps

The test software performs the following steps.

1. Place the PHDC in Configured state.
2. Place the 11073-20601 agent in the Operating state (11073 association process).
3. Issue a 11073-20601 Get MDS Object.
4. Verify that a QoS Preamble was NOT received before the agent response to the Get MDS Object.

Repetitions

Repeat test on every interface exposed by the enumeration process.

3.4.4 Verify SET_FEATURE / CLEAR_FEATURE request not supported

This test verifies the device responds to the SET_FEATURE request, as specified in the PHDC spec, when the Meta-Data Message Preamble Feature is not supported by the device.

Covered Assertions

7.1.1#2, 6#1

Required Testing Resource

Preconditions

This test is run with the PHD in Configured state. The unit under test does NOT support the MetaData preamble.

Overview of Test Steps

The test software performs the following steps.

1. Place the PHDC in Configured state.
2. Send a Set Meta-Data Message Preamble Feature to the unit under test.
    - bmRequestType = 21h
    - bRequest = 03h
    - wValue = 0101h
    - wIndex = PHDC interface number
    - wLength = 0
3. Verify that the device issued a STALL.
4. Send a Clear Meta-Data Message Preamble Feature to the unit under test.
    - bmRequestType = 21h
    - bRequest = 01h
    - wValue = 0101h
    - wIndex = PHDC interface number
    - wLength = 0
5. Verify that the device issued a STALL.

Repetitions

Repeat test on every interface exposed by the enumeration process.

3.4.5 Verify SET_FEATURE / CLEAR_FEATURE requests

This test verifies the device implements the SET_FEATURE and CLEAR_FEATURE requests as specified in the PHDC spec.

Covered Assertions

7.1.1#1, 7.1.1#4, 7.1.1#6, 7.1.1#7, 7.1.1#10, 7.1.1#12, 7.1.1#13, 7.1.1#14

Required Testing Resource

11073-20601 Agent or equivalent

Preconditions

This test is run with the PHD in the Configured state, and the 11073-20601 agent in the Associating state. The unit under test supports the MetaData Message Preamble, and has it set in the appropriate descriptor.

Overview of Test Steps

The test software performs the following steps.

1. Place the PHDC in Configured state.
2. Wait for receipt of an 11073-20601 association request.
3. Before issuing the association response, send a Set Meta-Data Message Preamble Feature to the unit under test.

- bmRequestType = 21h
- bRequest = 03h
- wValue = 0101h
- wIndex = PHDC interface number
- wLength = 0

4. Continue the 11073-20601 association process, until the agent is in the Operating state.
5. Issue a 11073-20601 Get MDS Object.
6. Verify that a Meta-Data Message Preamble was received before the response to the Get MDS Object.
7. Send a Clear Meta-Data Message Preamble Feature to the unit under test.

- bmRequestType = 21h
- bRequest = 01h
- wValue = 0101h
- wIndex = PHDC interface number
- wLength = 0

8. Issue a 11073-20601 Get MDS Object.
9. Verify that a Meta-Data Message Preamble was NOT received before the response to the Get MDS Object.

Repetitions

Repeat test on every interface exposed by the enumeration process.

3.4.6 Verify SET_FEATURE error conditions

This test verifies the device responds to the SET_FEATURE request with invalid data, as specified in the PHDC spec.

Covered Assertions

7.1.1#11

Required Testing Resource

11073-20601 Agent or equivalent

Preconditions

This test is run with the PHD in Configured state. The unit under test supports the MetaData preamble, and has it set in the appropriate descriptor.

Overview of Test Steps

The test software loops through the following steps using an invalid request field as described in the following table for each iteration. All other fields of the request will have valid values. See test 3.4.5 for an example of the valid request fields that will be used.

Iteration	Field	Value	Description
1	wValue	0201	Invalid bQoSEncodingVersion value
2	wValue	0102	Invalid FEATURE_PHDC_METADATA value
3	wIndex	bNumInterfaces + 1	Derived from Configuration Descriptor

1. Place the PHDC in Configured state.
2. Send a Set Meta-Data Message Feature to the unit under test with an invalid request field, as described above.
3. Verify that a Request Error was returned by the device.

4. Request data on the interface under test.
5. Verify that a Meta-Data Message Preamble was not received before the requested data.

Repetitions

Repeat test on every interface exposed by the enumeration process.

3.4.7 Verify CLEAR_FEATURE error conditions

This test verifies the device responds to the CLEAR_FEATURE request with invalid data, as specified in the PHDC spec.

Covered Assertions

7.1.1#11

Required Testing Resource

11073-20601 Agent or equivalent

Preconditions

This test is run with the PHD in Configured state. The unit under test supports the MetaData preamble, and has it set in the appropriate descriptor.

Overview of Test Steps

The test software repeatedly sends a Clear Meta-Data Message Feature request with an invalid request field as described in the following table for each iteration. All other fields of the request will have valid values. See test 3.4.5 for an example of the valid request fields that will be used.

Iteration	Field	Value	Description
1	wValue	0201	Invalid bQoSEncodingVersion value
2	wValue	0002	Invalid FEATURE_PHDC_METADATA value
3	wIndex	bNumInterfaces + 1	Derived from the Configuration Descriptor

1. Place the PHDC in Configured state.
2. Send a Set Meta-Data Message Preamble Feature to the unit under test.
    - bmRequestType = 21h
    - bRequest = 03h
    - wValue = 0101h
    - wIndex = PHDC interface number
    - wLength = 0
3. Loop through the following steps using an invalid request field as described above.
4. Send a Clear Meta-Data Message Feature to the unit under test with an invalid request field.
5. Verify that a Request Error was returned by the device.
6. Request data from the device.
7. Verify that a Meta-Data Message Preamble was received prior to the requested data.

Repetitions

Repeat test on every interface exposed by the enumeration process.

3.4.8 Verify invalid class request

This test verifies the device responds to an unspecified PHDC request with a Request Error, as specified in the USB 2.0 specification.

Covered Assertions

7.1.1#6, 7.1.2#1

Required Testing Resource

Preconditions

This test is run with the PHD in Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Place the PHDC in Configured state.
2. Send an invalid class request to the unit under test with an invalid:
   - bmRequestType = 21h
   - bRequest = 05h
   - wValue = 0101h
   - wIndex = PHDC interface number
   - wLength = 0
3. Verify the device returns a Request Error.

Repetitions

Repeat test on every interface exposed by the enumeration process.

3.4.9 Verify Host SET_FEATURE/CLEAR_FEATURE request

HOST ONLY TEST: This test verifies the host implements the SET_FEATURE and CLEAR_ FEATURE requests, as specified in the PHDC spec.

Covered Assertions

7.1.1#1, 7.1.1#3, 7.1.1#4, 7.1.1#6, 7.1.1#7, 7.1.1#8, 7.1.1#9, 7.1.1#10

Required Testing Resource

11073-20601 Manager or equivalent that supports the test API specified in section x.x. A golden test device to exercise the test API specified in section x.x. The golden device must send a QoS descriptor with the bQoSEncodingVersion = 01h.

Preconditions

This test is run with the PHD in Configured state. The unit under test supports the MetaData preamble.

Overview of Test Steps

The test software performs the following steps.

1. Place the PHDC in Configured state.
2. Request the Host to send data.
3. Verify the Meta-Data Message Preamble was NOT received prior to the data. (Meta-Data Message Preamble is OFF by default)
4. Request the Host to enable Meta-Data Message Preamble.
5. Verify the SET FEATURE request contains the following values:

- bmRequestType = 21h
- bRequest = 03h
- wValue = 0101h
- wIndex = PHDC interface number (01h)
- wLength = 0

6. Request the Host to send data.
7. Verify the Meta-Data Message Preamble was received prior to the data.
8. Request the Host to disable Meta-Data Message Preamble.
9. Verify the CLEAR_FEATURE request contains the following values:

- bmRequestType = 21h
- bRequest = 01h
- wValue = 0101h
- wIndex = PHDC interface number
- wLength = 0

10. Request the Host to send data.
11. Verify the Meta-Data Message Preamble was NOT received prior to the data.

3.4.10 Verify Host GET_STATUS request

HOST ONLY TEST: This test verifies the host implements the GET_STATUS request, as specified in the PHDC spec.

Covered Assertions

7.1.2#1, 7.1.2#3

Required Testing Resource

11073-20601 Manager or equivalent that supports the test API specified in section x.x. A golden test device to exercise the test API specified in section x.x.

Preconditions

This test is run with the PHD in Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Place the PHDC in Configured state.
2. Request the Host to issue a GET_STATUS request.
3. Verify the GET_STATUS request contains the following values:

- bmRequestType = A1h
- bRequest = 00h
- wValue = 00h
- wIndex = PHDC interface number that is being tested
- wLength = 2

3.4.11 Verify Host clears request STALL

HOST ONLY TEST: This test verifies the host clears the device STALL condition, when it sends a SET_FEATURE / CLEAR_FEATURE request to a device that does NOT support the Meta-Data Message Preamble, as specified in the PHDC spec.

Covered Assertions

7.1.1#2

Required Testing Resource

11073-20601 Manager or equivalent that supports the test API specified in section x.x. A golden test device to exercise the test API specified in section x.x. The golden device must send a PHDC Class Function descriptor with the bmCapability = 01h.

Preconditions

This test is run with the PHD in Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Place the PHDC in Configured state.
2. Request the Host to enable Meta-Data Message Preamble.
3. Issue a STALL to indicate the Meta-Data Message Preamble is not really supported.
4. Verify the Host clears the STALL condition.

3.5 Data/Messages Descriptor Formatting

These tests ensure that enumeration for the data and messaging format descriptors occurs successfully.

3.5.1 Verify valid bPHDCDataCode

This test verifies that the values for bPHDCDataCode are only those listed in Table 18 – Personal Healthcare Data and Messaging Formats in the PHDC spec.

Covered Assertions

A.2#1,

A.3#1,

A.4#1

Required Testing Resource

<TBD>

Preconditions

The device and the host are powered up and running. The device and host have not yet been connected via a USB cable; however, the host and device have been introduced previously (drivers installed, inf files read, etc.).

Overview of Test Steps

The test software performs the following steps.

1. Connect the host and device via a USB cable.
2. Verify on the host that PHDC Class Function descriptor bPHDCDataCode value is one of the following below: [A.2#1]

   PHDC_VENDOR (01h) [A.3#1]

   PHDC_11073_20601 (02h) [A.4#1]

3.5.2 Verify valid PHDC 11073 PHD Function Extension descriptor

This test verifies that the PHDC 11073 PHD Function Extension descriptor is sent in the correct order and with the correct data. This test is only run on devices where bPHDCDataCode = PHDC_11073_20601.

Covered Assertions

5.2.2#1

A.4#2, A.4#3, A.4#4, A.4#7

A.5#1

Required Testing Resource

<TBD>

Preconditions

The device and the host are powered up and running. The device and host have not yet been connected via a USB cable; however, the host and device have been introduced previously (drivers installed, inf files read, etc.).

Also bPHDCDataCode must = PHDC_11073_20601 for this test to be run.

Overview of Test Steps

The test software performs the following steps.

1. Connect the host and device via a USB cable.
2. Verify on the host that the PHDC 11073 PHD Function Extension descriptor immediately follows the PHDC Class Function descriptor. [5.2.2#1, A.4#2]
3. Verify on the host that the PHDC 11073 PHD Function Extension descriptor exists and contains the following values: [A.4#3]
   bLength = 04h or greater and is a multiple of 2 [A.4#4]
   bDescriptorType = PHDC_11073PHD_FUNCTION_DESCRIPTOR (30h) [A.5#1]
   bReserved = 00h
   bNumDevSpecs = (bLength – 4)/2 [A.4#7]

3.5.3 Verify valid bNumDevSpecs and wDevSpecializations

Verify that the bNumDevSpecs and wDevSpecializations values are set to allowed values.

This test is only run on devices where bPHDCDataCode = PHDC_11073_20601.

Covered Assertions

A.4#5, A.4#6

Required Testing Resource

<TBD>

Preconditions

The device and the host are powered up and running. The device and host have not yet been connected via a USB cable; however, the host and device have been introduced previously (drivers installed, inf files read, etc.).

Also bPHDCDataCode must = PHDC_11073_20601 for this test to be run.

Overview of Test Steps

The test software performs the following steps.

1. Connect the host and device via a USB cable.
2. Verify on the host that bNumDevSpecs is >= 0. [A.4#5]

3. Verify on the host that wDevSpecializations is equal to an array of bNumDevSpecs Words. The valid values for these Words are those in ISO/IEEE 11073-10101™; however, we will not check these words for validity in this test suite.

3.6 Multi-Function Devices

This section contains tests for multi-function devices. These tests are required for hosts, but are optional for devices. If a device does not implement multiple functions, then these tests do not need to be run.

3.6.1 Multi-Function Configuration Test

DEVICE-ONLY TEST: Test that devices that implement multiple functions by exposing one interface per function correctly enumerate over all interfaces.

Covered Assertions

See assertions for test cases referenced.

Required Testing Resource

For devices that implement the 11073 PHD data and messaging format, this test requires a host that can successfully perform 11073 Association.

TODO: Add information for vendor-specific data and messaging format testing.

Preconditions

The device and the host are powered up and running. The device and host have not yet been connected via a USB cable; however, the host and device have been introduced previously (drivers installed, inf files read, etc.).

Overview of Test Steps

The host test software performs the following steps.

1. Connect the host and device via USB cable.
2. If the configuration descriptor bNumInterfaces = 1, then exit with success. (If PHDC 11073 PHD Function Extension descriptor.bNumDevSpecs == 1, then this test is not applicable. If it is > 1, then this test will be run via the tests in sections 3.1 through 3.5.)
3. If the configuration descriptor bNumInterfaces > 1, then, for each additional interface repeat the relevant tests in sections 3.1 through 3.5 for each interface where bInterfaceClass = 0Fh.

4. Checklist Items

This section contains the beginning of the checklists to be published for devices and hosts which will run PHDC tests.

4.1 Vendor Data Devices/Hosts

These checklist items apply to devices or hosts that implement vendor-defined data. These are devices or hosts where *bPHDCDataCode* is PHDC_VENDOR.

1. Vendor to provide an application to be used during testing. Application must expose the following functions:
   - PHDCSend(char *msg, int len, int qos)
   - PHDCReceive(char *msg, int len, int qos)
   - This list will be completed if it becomes necessary to test vendor data.
2. Others?

4.2 11073-20601 Devices

These checklist items apply to devices that implement 11073-20601 data. These are devices where *bPHDCDataCode* is PHDC_11073_20601.

Note: The host-driven "Remote Operation Invoke | Get" command with handle 0 (i.e., Get MDS Object) will be used to test data transfers from the device. According to 11073-20601, the device will respond with "Remote Operation Response | Get." This command can be executed as many times as necessary in order to test the device.

1. The device software needs to support sending 11073 PHDC Association, Configuration, and be able to enter into the Operating State.
2. The device software needs to support the device-driven "Remote Operation Invoke|Get" command with handle 0.
3. The device software needs to respond to a "Remote Operation Invoke|Get" with handle 0 with a "Remote Operation Response|Get" response.

4.3 11073-20601 Hosts

These checklist items apply to hosts that implement 11073-20601 data. These are hosts where *bPHDCDataCode* is PHDC_11073_20601.

Note: The device-driven "Remote Operation Invoke | Event Report" command will be used in confirmed mode (i.e. send measurement data and expect a response) to test data transfers from the host. According to 11073-20601, the host will respond with "Remote Operation Response | Confirmed Event Report." This command can be executed as many times as necessary in order to test the host.

1. The host software needs to support 11073 PHDC Association, Configuration, and be able to enter into the Operating State.
2. The host software needs to support the device-driven "Remote Operation Invoke|Event Report."
3. The host software needs to respond to a "Remote Operation Invoke|Event Report" with a "Remote Operation Response|Confirmed Event Report" for the purposes of this testing.
4. The host software needs to expose a mechanism for turning on and off the Meta-Data Message Preamble feature.