x0000 - root

Version 2

Root

[0] **getFeature**(featId) → featureIndex, featureType, featureVersion
[1] **getProtocolVersion**(0, 0, pingData) → protocolNum, targetSw, pingData

Overview

The entry point feature which is always at index 0.

The root feature allows obtaining information from all other features in the device (enumeration) and helps in determining the host software compatibility.

Functions and Events

[0] getFeature(featId) → featureIndex, featureType, featureVersion

Given a desired feature ID, returns its index in the feature table, the feature type bitfield and the feature version.

Parameters

featureId

The ID of the desired feature. The ID 0 is forbidden (root feature ID).

Table 1. getFeature() request packet format

byte \ bit	7	6	5	4	3	2	1	0
0	featureId (MSB)							
1	featureId (LSB)							

Returns

featureIndex

The index in the feature table used to access the feature. The value 0 indicates the feature was not found. Non-root features will start with index value 1.

featureType

Bitfield describing the feature properties:

obsolete

An obsolete feature is a feature that has been replaced by a newer one, but is advertised in order for older SWs to still be able to support the feature (in case the old SW does not know yet the newer one).

hidden

A SW hidden feature is a feature that should not be known/managed/used by end user configuration SW. The host should ignore this type of features.

engineering

A hidden feature that has been disabled for user software. Used for internal testing and manufacturing.

manufacturing_deactivatable

A manufacturing feature that can be permanently deactivated. It is usually also hidden and engineering.

compliance_deactivatable

A compliance feature that can be permanently deactivated. It is usually also hidden and engineering.

featureVersion

The zero based feature version number.

Table 2. getFeature() response packet format

byte \ bit	7	6	5	4	3	2	1	0
0	featureIndex							
1	featureType							
	obsl	hidden	eng	manuf_deact	compl_deact	---	---	---
2	featureVersion							

NOTE

On feature versions: Features have a version number that starts at zero. When a feature version is incremented it means that new functionality has been added to it. Even if new functionality is present, newer versions of a feature will 'always' be backward compatible with all previous versions of a feature. Features that need to break backward compatibility will use a different feature number so de-facto creating a new feature.

[1] getProtocolVersion(0, 0, pingData) → protocolNum, targetSw, pingData

Returns protocol number, target SW, and echoes ping data.

There are two types of host software, 'multidevice host software', that supports many devices in a

single application, and 'single device host software', an application that is dedicated to a single device.

Multidevice host software should test each device's protocol number to validate if it is willing to support it. When a device's protocol number is 4, the targetSw fields further qualifies its protocol number. This allows the hinting to the host SW to be even more specific.

Host software's decision of supporting of a specific hidpp device is a combination of the host SW knowledge of the device features, (a host may decide to drop a device if it does not know some of its features) or it can be a pure strategic decision based on host software user's experience. Example: a 'gaming specific' host SW may not want to support office oriented products, so an office-oriented device released after the gaming host SW has been released can set its protocol number to 4 and its target SW to 'office oriented' so as to avoid the gaming host SW to try to pick up the device.

Once a SW determines it is not supporting a particular device it should perform no further communication with the device. It should not send additional ping requests or enumerate features. This will minimize the use of transaction collision recovery when multiple SW are talking to a device.

If a software does not need to support future devices based on feature set and wants to instead hard-code a table of PIDs of devices it will support, then it can ignore the targetSw field and a targetSw ID does not need to be allocated for that software.

Parameters

zero

Padding so the ping, request and response are in the same location.

zero

Padding so the ping, request and response are in the same location.

pingData

Data to be echoed back. For recovery from timeouts or collisions.

Table 3. getProtocolVersion() request packet format

byte \ bit	7	6	5	4	3	2	1	0
0	zero							
1	zero							
2	pingData							

Returns

protocolNum

The protocol number is a field that hints the host software if it should support the device.
protocolNum = 2 : Intended target SW is Logitech SetPoint
protocolNum = 3 : Intended OEM SW described in targetSw field
protocolNum = 4 : Intended target SW described in targetSw field

targetSw

When (protocolNum >= 3) this field further hints at which software should support the device. Otherwise the value is zero.

pingData

Echoes the ping paramenter as sent in the request. For recovery from timeouts or collisions.

NOTE	On ping data usage: Ping data can be used by the SW to re-synchronize the command/response data flow in case of collision or timeout. Whenever an incorrect response is received or a command times out, the SW should send a ping request and wait for a response with the correct ping data. The ping data sent should be a random nonzero value which does not match the previous ping request that was sent. If the SW is sending a protocol request instead of a ping request the ping value should be 0

Table 4. getProtocolVersion() response packet format

byte \ bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	protocolNum							
1	targetSw							
2	pingData							

targetSw byte	**bit 7**	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
meaning for protocol 3	---	---	---	---	---	---	Dell	Lenovo

Lenovo

Lenovo software is the intended OEM software if this bit is set.

Dell

Dell software is the intended OEM software if this bit is set.

targetSw byte	**bit 7**	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
meaning for protocol 4	TSF	---	---	MPS	WPS	LPM	LGS	LDM

LDM

Logitech Device Manager is the intended target SW if this bit is set.

LGS

Logitech Gaming Software is the intended target SW if this bit is set.

LPM

Logitech Preference Manager is the intended target SW if this bit is set.

WPS

Windows Presenter Software is the intended target SW if this bit is set.

MPS

Mac Presenter Software is the intended target SW if this bit is set.

TSF

Target Software Feature: feature 0x0030 specifies the Target Software if this bit is set.

[ev0] noOperation

This event may be sent by a device as needed to prevent the device from entering low power mode. If sent following a HID response this may allow the device to respond quicker to the next HID command. This event is a no-op and contains no data. Host SW can safely ignore this event.

ChangeLog

- Version 2: Adds deactivatable manufacturing / compliance features bits in feature Type bitmap.
- Version 1: Adds the feature version return value (featVer) to feature function [0]getFeature(featId)
- Version 0: Initial version