Bluetooth HID Profile	Date / Year-Month-Day 2003/5/22	Approved Version	Revision **1.0**	Document No HID_010_SPC_PFL/1.0
Edited by: Craig Ranta Steve McGowan	e-mail address craigra@microsoft.com steve.mcgowan@intel.com			N.B. Confidential

HUMAN INTERFACE DEVICE (HID) PROFILE

VERSION 1.0 ADOPTED

Abstract

This profile defines how devices with Bluetooth™ wireless communications can use the HID Protocol initially to discover the feature set of a HID, and then communicate with the HID. This profile

further defines how a device with Bluetooth wireless communications can support HID services over the Bluetooth protocol stack using the Logical Link Control and Adaptation Protocol (L2CAP) layer.

Referenced Documents

1. Universal Serial Bus Specification, Version 1.1 (www.usb.org)
2. USB Physical Interface Device Specification (www.usb.org)
3. USB HID Usage Tables, Version 1.1 (www.usb.org)
4. USB Device Class Definition for Human Interface Devices, Version 1.11 (www.usb.org)
5. Bluetooth Core Specification 1.1 (www.bluetooth.com)
6. Bluetooth Profiles Specification 1.1 (www.bluetooth.com)
7. Bluetooth Human Interface Device Marketing Requirements Document, Version 1.0 (www.bluetooth.com)
8. Bluetooth Assigned Numbers document (a web document found at http://www.bluetooth.org/assigned-numbers.htm)
9. Universal Serial Bus Language Identifiers (LANGIDs) (www.usb.org)
10. The Unicode Standard, Worldwide Character Encoding, Version 3.0, The Unicode Consortium, Addison-Wesley Publishing Company, Reading, Massachusetts (www.unicode.com)
11. ISO-10646-1:2000 Annex D, UTF-8 (UCS Translation Format)
12. Bluetooth Human Interface Device (HID) Profile Test Specification, (www.bluetooth.com)
13. Bluetooth Device Identification Specification, Version 0.95b (www.bluetooth.com)

Revision History

Revision	Date	Comments
0.1	19.6.2000	Initial Draft.
0.3	18.8.2000	Added device requirements.
0.4	2.10.2000	Updated pairing/bonding. Added power management section.
0.49	28.11.2000	Added HID over L2CAP subdocument.
0.492	5.1.2001	Added host requirements, reformatting.
0.5	20.1.2001	Fixed inconsistent internal cross-references. Section 7.3.1.1: changed mandatory reporting rate limit to recommendation. Section 5.6 modified. Section 11: made 3 and 5 slot packets, paging optional.
0.6	3.4.2001	Incorporated comments on V0.5
0.7	20.4.2001	Incorporated V0.5 L2CAP comments. Dropped Default Reports.
0.7b	15.5.2001	Incorporated V0.7 draft feedback for Section 7. Other minor formatting changes.
0.7c	16.5.2001	Added signaling flow diagrams in Appendix A. Section 6.7.3: removed option of simultaneous host connections for consistency with virtual cable concept. Section 10: changed inquiry to optional for HIDs.
0.7d	17.5.2001	Updated Boot Mode Operation (Section 7.2.1) to include Report IDs on boot reports. Added HIDSDPDisable attribute (Section 7.11.8). Added Section.
0.7e	1.6.2001	Added Invalid Parameter error to Handshake (Section 7.4.1). Added text to Section 7.2.1 stating that in boot mode, report IDs are used on both the control and interrupt channels. Added note to GET_REPORT Size field in the Request byte (Section 7.4.3). Added note to Section 7.2.1, “Boot Mode Operation” that in boot mode the Report ID shall be included.
0.7f	16.6.2001	Major rewrite of SDP section. Added details to HID_CONTROL relative to suspend operation.
0.8a	18.6.2001	Minor revisions to Section 6, taking into account device-initiated reconnection concepts. Revised Table 26 “connectable mode” behavior for the same reason. Other minor grammatical changes.
0.8b	19.6.2001	Changed HIDSubClass Attribute to refer to the FHS Device Minor Class field.
0.8c	20.6.2001	Dropped HIDReportInterval attribute.
0.8d	21.6.2001	Dropped Sniff Mode text in the HIDRemoteWake discussion. Added HIDSerialNumber attribute. Added Bluetooth Device Identification to document list. Clarified HIDDeviceReleaseNumber attribute.
0.8e	22.6.2001	Deleted HIDSerialNumber attribute. Incorporated numerous clarifications from HID WG comments.
0.8f	5.7.2001	Section 3.2, Figure 2: revised the protocol stack of the HID to show a more cost-effective implementation where the HID software is embedded in the radio processor. Section 7.4.4: clarified text regarding short reports (not allowed except using DATC mechanism). Section 7.12.8: clarified that LanguageBaseAttributeIDList is required by

Revision	Date	Comments
		HIDs. Section 7.4: clarified that payload size includes header, DATC required if payload equals or exceeds MTU. Section 7.45 and 7.46: added references to HIDDeviceSubClass. Section 7.17.1 to 7.17.3: changed the peak bandwidth for Best Effort channels to “unknown”. Updated Table 20 to add new attribute AdditionalProtocolDescriptorLists. Updated format to match example from Johannes Elg of Ericsson. Updated Table 21.
0.9	22.7.2001	Added additional message sequence charts. Added section 14. Clarified wording in various sections. Changed HIDVirtualConnection attribute name to HIDVirtualCable.
0.9a	24.7.2001	Technical editor review. Updated styles and fonts for consistency with other Bluetooth documents.
0.9b	30.7.2001	Removed requirements for pairing when virtually cabled. Clarified that hosts shall initiate security procedures. Clarified HID connection establishment rules (both channels needed, may be configured simultaneously) Added requirement for hosts to try PIN code zero when initiating security procedure to HID. Added references to HID usage tables for keyboard scan codes and pointing device boot packets.
0.9c	26.9.2001	1) changed the terminology from "baseband timeout" to "supervision timeout" as defined by the LMP_supervision_timeout PDU, ending any possible confusion about what we are referring to. 2) Modified section 7.10.1.1. 3) Modified section 7.1.3 to state that the interrupt channel shall be opened anyway. 4) Change the text in section 7.7 to reference the L2CA_ConfigReq service rather than the L2CA_GetInfo service. 5) Corrected the Example SDP.
0.9c2	26.10.2001	7.4.2 Added paragraph on HID_Disconnect packet and added command to Table 4 1,3,2 Added definitions for ACL, PSM, QoS and bonding. 4.4 Clarified that only one interrupt and control channel may be open at one time in a virtually cabled device. 6.3,6.4 Clarified the moment at which a virtual cable is considered established and broken. Devices expected to exit page and page scan when virtually cabled and connected. 6.7.1.6 Added error case when device forgets the host. 7.2.2 Added section on channel initialization and establishement rules. 7.4.3 BufferSize in header byte changed to NOT include the header itself. Table 5, Figure 9 Clarified that biffer size is little-endian 7.10.1.X Clarifications for large packet DATC cases
0.91	13.11.2001	5.4.2 Added application note for keyboard passkey entry Various: Applied IEEE guidelines for use of shall, must, should and changed several instances of must to shall. 5.4.5.10 Added paragraph stating that Limited Inquiry Access Code (LIAC) shall be used to scan for Bluetooth HIDs. 5.2.2 Clarified that the host and device must determine whether SAR is necessary by using the L2CA_Config_Req service. 5.4.5.6 Changed “negotiate” to “set” in reference to the link supervision timeout.

Revision	Date	Comments
0.91a	14.11.2001	7.10.1 added reference to HID_CONTROL packet that specifies VIRTUAL_CABLE_UNPLUG exception. 7.10 Minor text corrections. 7.10.2 Added “Accepted" result.
0.92	3.12.2001	7.4.2 Added paragraph on HID_Disconnect control packet and added command to Table 4. 7.4.3 BufferSize in header byte changed to NOT include the header itself. Table 26: Changed L2CAP information response to Optional and QoS to Mandatory in host. 7.6 Deleted paragraph allowing concatenated reports for consistency with USB HID 8.3 Changed Flush timeout recommendation to infinite. 7.11.13 Added new attribute HIDLinkSupervisionTimeout. 7.11.14 Added new attribute HIDNormallyConnectable. Many other clarifications throughout document.
0.93	5.12.2001	Minor formatting and rewording
0.95draft	8.12.2001	Updated MSCs in Appendix B
0.95a	21.4.2002	Changed support of SNIFF and PARK modes to be optional for HIDs Added HIDBootDevice attribute to tables and examples. Changed text to optionally allow HIDs to initiate authentication after host has paired. Miscellaneous formatting and editing changes.
0.95b	26.4.2002	Added HIDProfileVersion attribute
0.95c	16.5.2002	Incorporated BARB comments: 5.3.1 Clarified requirements for QoS support 5.4.5.5 Clarified responsibility for L2CAP connection re-establishment in automatic reconnection scenario 7.11.13 Added note about multiple profiles sharing one ACL link.
0.95d	26.9.2002	Incorporated errata E2539-E2560, generated from interoperability testing and working group reviews
1.0draftA	11.12.2002	Incorporated errata E2568-E2570. Final scrub for IEEE language usage (should, shall, may, must)
1.0draftB	29.1.2003	Incorporated erratum E2585.
1.0draftC	23.2.2003	Incorporated BARB review comments, which were captured by the following errata: E2594, E2597-E2602 Incorporated erratum E2593
1.0	8.4.2003	Changed HOLD mode support from mandatory to optional for devices, due to Errata E2571 being rejected by the HID working group. Final release for Board of Directors’ approval.
Version 1.0	22.5.2003	Updated title and header

Contributors

Jennifer Bray	Cambridge Silicon Radio
Peter Flittner	Cambridge Silicon Radio
Chris Hubball	Cambridge Silicon Radio
Drew Harrington	Cypress Semiconductor
Fred Jaccard	Cypress Semiconductor
Kelvin Klusendorf	Cypress Semiconductor
Steve McGowan	Intel Corporation
Venkat Yellapeddy	Intel Corporation
Xavier Bize	Logitech
Berni Joss	Logitech
Roland Meyer	Logitech
Rene Sommer	Logitech
Randy Aull	Microsoft Corporation
Fred Bhesania	Microsoft Corporation
Chris Dreher	Microsoft Corporation
Doron Holan	Microsoft Corporation
Terry Lipscomb	Microsoft Corporation
Craig Ranta	Microsoft Corporation
Jay Senior	Microsoft Corporation
Raymond Chiu	Motorola, Inc.
Curtis Stevens	Phoenix Technologies
John Milios	Semtech
Katsuhiro Kinoshita	Toshiba Corporation
Akihiko Sukigawa	Toshiba Corporation
Junko Ami	Toshiba Corporation

Disclaimer and Copyright Notice

ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS OR FOR NONCOMPLIANCE WITH LAWS, RELATING TO USE OF THE SPECIFICATION IS EXPRESSLY DISCLAIMED. BY USE OF THE SPECIFICATION, EACH MEMBER EXPRESSLY WAIVES ANY CLAIM AGAINST BLUETOOTH SIG AND ITS PROMOTER MEMBERS RELATED TO USE OF THE SPECIFICATION.

Bluetooth SIG reserves the right to adopt any changes or alterations to the Specification as it deems necessary or appropriate and to adopt a process for adding new Bluetooth profiles after the release of the Specification.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

1.1 Scope

The Human Interface Device Profile Specification [4] defines the protocols, procedures, and features that shall be used by Bluetooth Human Interface Devices, such as keyboards, pointing devices, gaming devices, and remote monitoring devices. This specification uses the USB (Universal Serial Bus) definition of Human Interface Device (HID) [4] in order to leverage the existing class drivers for USB HID devices.

1.2 Purpose

This document intends to provide a set of general but unambiguous rules for implementing the USB HID protocol over a Bluetooth wireless link, with the goal of creating wireless human interface devices that are interoperable, easy to set up and use, have performance comparable to wired devices, and provide good consumer value. Applications recommendations will be given when appropriate.

1.3 Conventions, Notation, and Definitions

1.3.1 Profile Requirements Notation

In the profile requirements tables (see Sections 8, 9, and 10), the following symbols are used:

"M" for mandatory to support (used for capabilities that shall be used in the profile)

"O" for optional to support (used for capabilities that can be used in the profile)

"C" for conditional support (used for capabilities that shall be used in case a certain other capability is supported)

"X" for excluded (used for capabilities that may be supported by the unit but should never be used in the profile)

"N/A" for not applicable (in the given context it is impossible to use this capability)

Some excluded capabilities are capabilities that, according to the relevant Bluetooth Specification [5], are mandatory. These features may degrade operation of devices following this profile. Therefore, these features shall never be activated while a unit is operating as a unit within this profile. In this specification the word *shall* is used for mandatory requirements, the word *should* is used to express recommendations, and the word *may* is used for options.

1.3.2 Definitions

ACL – Asynchronous Connectionless. A type of data link supported by the Bluetooth baseband,. Quality of Service (QoS) parameters are used to guarantee data delivery on an ACL connection.

Bonded – A device is bonded with a host when it has performed the pairing process and stored the host's Bluetooth address and link key for future use.

BIOS – Basic Input/Output System. Low level firmware which controls a PCs hardware, also self-test and configuration when the PC is powered on.

HID – Human Interface Device. This term is commonly used to refer to both the USB data reporting protocol for Human Interface Devices, and the devices themselves. In this document, when referring to the devices themselves we use "HID" or "HIDs". When referring to the protocol, we will use the "HID protocol".

L2CAP – Logical Link Control and Adaptation Protocol. The highest-level Bluetooth communications layer in the host, which handles device connections as logical data streams.

Pairing - The process of establishing authentication keys between two Bluetooth devices that have not previously encountered each other. In the HID usage models the host normally initiates pairing. A *Trusted Device* is a device that has been paired using a secret key and is explicity marked as trusted.

PID – Physical Interface Device. A special class of HID devices that send physical output as well as input; i.e., force feedback gaming devices.

PIN – Personal Identification Number. Also referred to as the Bluetooth passkey at the user interface.

PSM – Protocol Service Multiplexor. A protocol identifier used by the L2CAP layer to support multiple protocols.

QoS – Quality of Service. A general term that refers to the ability of a data channel to provide guaranteed data bandwidth and/or latency to a device or service.

SDAP – Service Discovery Application Profile.

SDP – Service Discovery Protocol.

USB – Universal Serial Bus.

2 Introduction to the HID Protocol

This document describes how to use the USB Human Interface Device (HID) class devices over a Bluetooth wireless link. Concepts from the USB Specification [1] are used but not explained in this document. **The USB Core [1] and HID Specifications [4, 3, 2] are recommended pre-reading for understanding the content of this document.** See the Referenced Documents section at the beginning of this document. The HID class consists primarily of devices that are used by humans to control the operation of computer systems. Typical examples of HID class devices include:

- Keyboards and pointing devices; for example, standard mouse devices, trackballs, and joysticks
- Front-panel controls; for example, knobs, switches, buttons, and sliders
- Controls that might be found on devices such as telephones, VCR remote controls, games or simulation devices; for example, data gloves, throttles, steering wheels, and rudder pedals
- Devices that may not require human interaction but provide data in a similar format to HID class devices; for example, bar-code readers, thermometers, or voltmeters

The **HID** class was originally targeted at human interface devices; however, HID is very useful for any application that requires low-latency input-output operations to an external interface, and the ability for that device to describe itself. Many typical **HID** class devices include indicators, specialized displays, audio feedback, and force or tactile feedback. Therefore, the **HID** class definition includes support for various types of output directed to the end user. HID enables initialization, and control of self-describing devices.

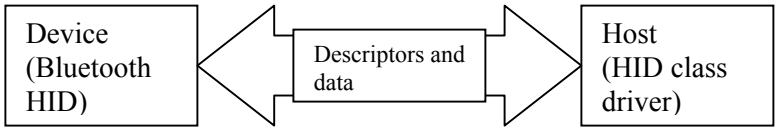
2.1 Overview

The Universal Serial Bus Specification [1] included a brilliant concept…instead of having separate software drivers for each new type of computer peripheral device; it would group devices together that have similar data reporting characteristics into a device “class” and have a single class driver for each group. The devices further have the capability to describe themselves to the class driver; e.g., what controls they have and exactly how they report data, allowing future devices to be developed without the need to modify the host software. Here we will focus on a single USB device class, HID (Human Interface Device), a powerful and extensible data reporting system that was designed specifically for the needs of Human Interface Devices. **HID** is not specific to USB or any other type of physical data transport. It is the intention of this document to describe how to use the **HID** protocol over the Bluetooth wireless communications system, allowing manufacturers of input devices to produce high performance, interoperable, and secure wireless input devices.

Information about a HID device is stored in segments of its non-volatile memory. These segments are called descriptors. An interface descriptor can identify a device as

belonging to one of a finite number of classes. The **HID** class is the primary focus of this document. Other types of device classes described by USB specifications include display, audio, communication, and data storage.

A HID class device uses a corresponding **HID** class driver to retrieve and route all data. The routing and retrieval of data is accomplished by examining the descriptors of the device and the data it provides.



The HID class device descriptor identifies which other **HID** class descriptors are present and indicates their sizes; for example, **Report** and **Physical Descriptors**. A **Report** descriptor describes each piece of data that the device generates and what the data is actually measuring. For example, a **Report** descriptor defines items that describe a position or button state. Item information is used to:

- Determine where to route input; for example, send input to mouse or joystick API
- Allow software to assign functionality to input; for example, use joystick input to position a tank

By examining an item's **Report** descriptor, the **HID** class driver is able to determine the size and composition of data reports from the **HID** class device. **Physical descriptor** sets are optional descriptors that provide information about the part or parts of the human body used to activate the controls on a device.

3 HID Profile Overview

3.1 Profile Dependencies

The Bluetooth HID profile is built upon the Generic Access Profile (GAP), specified in the Bluetooth Profiles Document; see Referenced Documents. In order to provide the simplest possible implementation, the HID protocol runs natively on L2CAP and does not reuse Bluetooth protocols other than the Service Discovery Protocol.

3.2 Profile Stack

Below is an illustration of the software layers that reside in both the host and the human interface device for an example implementation. In this example, the host is a personal computer and has the upper layers of the Bluetooth software running on its native processor and is connected to a Bluetooth radio module via a transport bus such as USB. The HID in this example has its firmware embedded with the radio firmware, running on the same CPU, for the lowest possible cost implementation. Other implementations on the HID side are possible and equally valid.

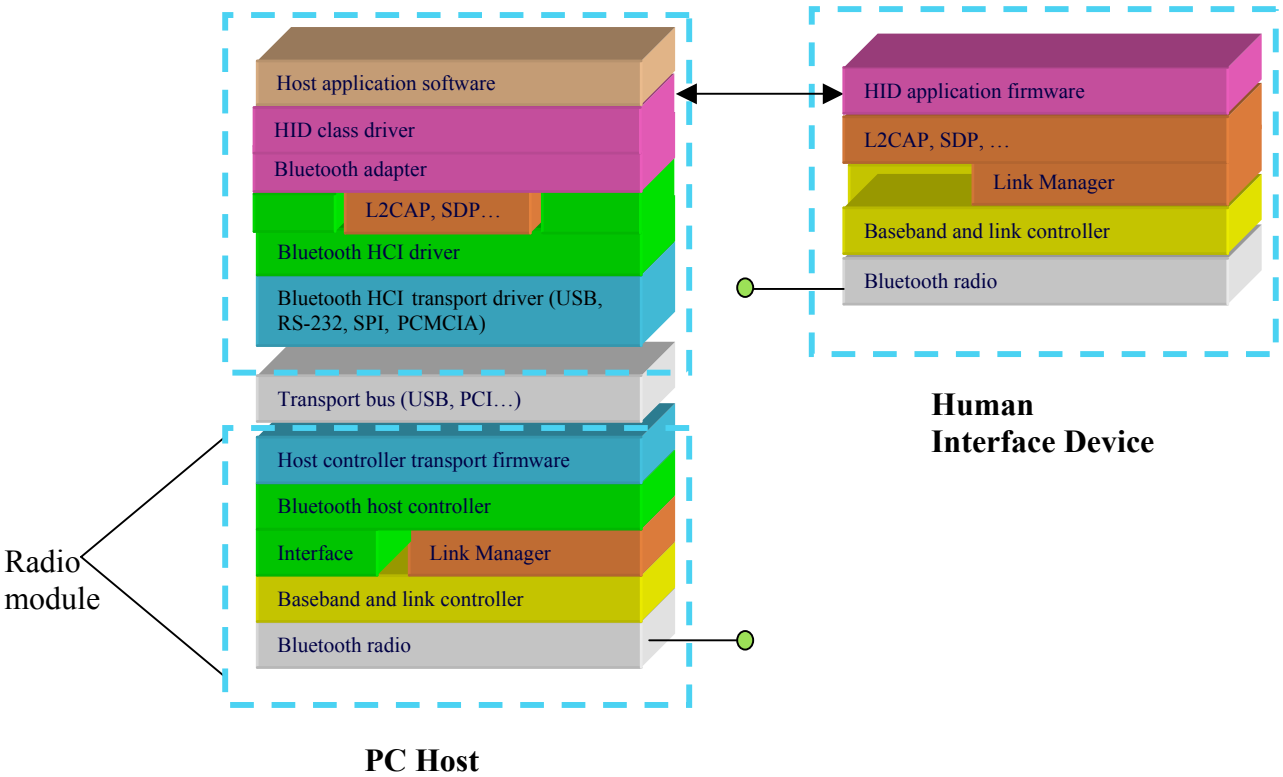


Figure 1: Typical Host and HID Software Stack

3.3 Configurations and Roles

The following roles are defined for this profile:

The *HID* (Human Interface Device) is the device providing the service of human data input and output to and from the host. Because the USB definition of HID includes all devices that report data in a similar fashion to HIDs, other devices such as remote sensors are included which may not interface directly with a human. Examples of HIDs are mice, joysticks, gamepads, keyboards, and also voltmeters and temperature sensors. The HID is normally the slave in the Bluetooth piconet structure.

The *host* is the device using or requesting the services of a Human Interface Device. Examples would be a personal computer, handheld computer, gaming console, industrial machine, or data-recording device. The host is normally the master in the Bluetooth piconet structure.

3.4 User Scenarios

Following are some brief descriptions of how people might use Bluetooth Human Interface Devices. For a more detailed description, refer to the Bluetooth HID Marketing Requirements Document [3].

3.4.1 Desktop Computing Scenario

In the traditional desktop computer scenario, use of Bluetooth wireless computer peripherals will free the desktop from multiple cables and allow input devices to be operated further from the desktop and in positions that are more comfortable. Users will be able to switch between multiple input devices without plugging and unplugging cables. Users will also be able to control different computers or host devices with at different times with a single Bluetooth HID device without concern for connecting cables. The security implemented by Bluetooth keyboards will provide protection from password and other sensitive data eavesdropping through walls. Use of Bluetooth will provide significant improvement in range and reliability over the types of wireless links currently used in these products.

3.4.2 Living Room Scenario

HID devices with Bluetooth wireless technology will enable the ease of multiplayer gaming. The users are no longer tethered to the gaming machine and can be seated casually within a standard sized living room. A game controller with Bluetooth technology can also provide audio input and output which improves the realism of the game and enables wireless chat and voice commands.

Interactive TV, Web TV, and PC-based satellite receiver type devices designed for the living room will also be able to take advantage of Bluetooth HID input devices. Bluetooth keyboards and pointing devices will provide a superior user experience to the existing infrared wireless keyboards. Bluetooth devices will not require line of sight alignment

with the receiver and the two-way capability allows remote displays and user feedback devices.

3.4.3 Conference Room Scenario

A pointing device enabled with Bluetooth wireless technology will allow the presenter in a conference room to control the presentation on a video screen from up to 10 meters away, without the need to be near the host or the projection device. The device need not be designed for operation on a flat surface. Today, such devices are expensive and use proprietary wireless technology. A HID with the Bluetooth wireless technology will standardize connections for these types of devices.

3.4.4 Remote Monitoring Scenario

Battery powered monitoring devices with Bluetooth wireless technology will provide many benefits to the end user. Some examples of these devices include temperature sensors, remote thermostats, security devices, pressure sensors, voltmeters, etc. By using Bluetooth wireless technology and the HID standard, monitoring systems can be installed quickly without running new wires to each of the installed sensors. The low power modes provided by Bluetooth wireless technology will provide long battery life for the remote transmitters. With Bluetooth wireless technology and the HID standard, it will be possible for the host to remotely control and receive data from these types of devices in a standardized way.

3.4.5 Example Applications

Note these applications are listed for reference. They may not all be practical. They may build on, but do not form part of the specification and may be subject to third party intellectual property rights.

Potential HID devices enabled with Bluetooth wireless technology include:

- Computer keyboards and keypads
- Trackballs, mice, and other pointing devices
- Game controllers (gamepads, joysticks, steering wheels, etc.)
- Battery operated sensors (temperature, pressure, security, etc.)
- Simple alphanumeric remote displays
- Universal remote controls
- Bar code scanners

3.5 Profile Fundamentals

This is a brief outline of Bluetooth requirements detailed by this specification:

- Master/Slave roles. Although there are no mandated master/slave roles, it is recommended that Bluetooth Human Interface Devices normally be a slave device, in order to avoid having the host radio multiplex between piconets.

- Discoverability. It is recommended that Human Interface Devices use only limited discoverable mode and non-discoverable mode. Limited discoverable mode allows rapid discovery of HIDs in environments with many Bluetooth devices.
- Authentication/bonding/encryption. This profile requires support for authentication and encryption for keyboards, keypads, and other HIDs that transmit identification or biometric information. This is due to the extraordinary sensitivity of the information that commonly travels from these devices (usernames, passwords, emails, passkeys). It is optional for other types of HIDs.
- Configuration. Bluetooth HIDs should be easy for consumers to configure out of the box. This profile will give examples of systematic configuration of Bluetooth peripherals.
- Performance. Bluetooth HIDs should have responsiveness (latency) similar to wired USB input devices, and provide superior performance to most other types of proprietary wireless input devices. This specification will give some guidelines for achieving the best performance but will not require a certain performance, this being left up to the manufacturer to best determine.
- Battery life. Bluetooth HIDs should fully utilize the power saving mechanisms provided by the Bluetooth Specification [5], such as Park, Hold, and Sniff modes, to achieve battery life comparable with existing wireless human input devices. Some guidelines and examples are given in this document
- Host software. The host will typically implement the HID profile by writing an interface driver (sometimes called a *miniport* driver on a PC host) between a standard HID class driver and the Bluetooth L2CAP and Link Manager layers. Recommended Bluetooth link parameters for connecting to HIDs are specified.
- “Virtual cables.” The concept of virtual cables is introduced and used to describe the 1:1 relationship between a HID and its host, and the rules that govern behavior when virtually cabled.

3.6 Conformance Requirements

If conformance to this profile is claimed, all capabilities indicated mandatory for this profile shall be supported in the specified manner (process mandatory). This also applies to all optional and conditional capabilities for which support is indicated. All mandatory capabilities and optional and conditional capabilities, for which support is indicated, are subject to verification as part of the Bluetooth qualification program. The HID test cases list (see the HID Profile Test Specification [12]) specify the particular protocol behavior that is required of Bluetooth HIDs. Verification of conformance to the HID Specification [4] is required.

4 User Interface Requirements

4.1 First Time Configuration

Wireless human input devices by their nature normally require some configuration to associate them with a particular host. Cabled devices form this association with the host by virtue of a direct electrical connection. For this reason, more configuration steps are required of wireless devices during first time setup. Bluetooth wireless devices optionally have the additional step of passkey (also known as the Personal Identification Number or PIN) entry during device pairing. Out-of-the-box configuration of Bluetooth Human Interface devices should be as simple as practical. A dedicated configuration pushbutton or other simple user action should be used to initiate configuration of the human input device, placing it in limited discoverable mode for 30-180 seconds. The host should prompt the user to press the configuration button, and then discover the device's address automatically. No more actions by the user should be required, unless the device implements security procedures and requires a manual passkey entry from the host or device.

Once the Bluetooth connection is established and the HID descriptors have been passed to the HID class driver on the host, the device will function like a wired USB HID device.

Example first time user interaction for a Bluetooth HID keyboard and a PC with existing Bluetooth capability are:

- User removes keyboard from box and installs batteries.
- User selects mode that causes host to discover and report new devices (using mouse or pointing device, don't assume another keyboard is available). This could be invoked by resetting the host. Alternatively, vendor-supplied software supplied in the box can initiate this mode, or the host simply does periodic device discovery by default. The user may be required to select the desired device from a list of discovered devices.
- Host prompts user to press configuration button on keyboard.
- Host displays message that keyboard was found; to register this device, enter the passkey code 7436 on keyboard
- Bluetooth keyboard is ready for use.

4.2 Latency and Performance

The essential performance requirement is that Bluetooth Human Interface Devices provide responsiveness to changes in input that are equivalent to wired devices. High performance applications, such as gaming, require application response time from a user button press or joystick movement on the order of a single video frame (16.7 ms at 60 Hz refresh). The implementation of the Bluetooth link should add no more than 5 ms

additional latency between the user input and the application response over a wired implementation when the Bluetooth HID is in the active state. For example, if a wired peripheral data report rate is 100 per second and the Bluetooth peripheral report rate is 100 per second, roughly equivalent latency can be achieved, with difference due to delay time through the software stack on both ends and host radio hardware interface.

4.3 Battery Life

The goal for common HID devices (mice, keyboards, gamepads) is to achieve at least three months of operation with three standard AAA or two AA alkaline batteries. The host shall not be required to actively power manage the HID device, with the possible exception of notifying the HID devices when its power state changes. The HID is responsible for its power management.

4.4 Multiple Devices per Host/Multiple Hosts per Device

Bluetooth HIDs shall set no limitation on the number of devices per host (up to the seven simultaneous active devices allowed per piconet). All trusted devices (devices that have either been authenticated or have no security procedures required) shall be allowed to have simultaneous connections to the host, if the host so desires. For example, multiple Bluetooth mice and keyboards are allowed. The behavior in this case shall be the same as for the USB case; i.e., input from all devices is allowed and the data streams are logically ORed together.

Similarly, a single Bluetooth HID may have established a bond or have its address known by multiple hosts. However, if it has declared itself virtually cabled, it is mandatory that the device only support a single host connection, and only one control and one interrupt L2CAP channel to that host, at one time (see SDP attribute HIDVirtualCable). If the device has not declared itself virtually cabled, it is still recommended that the device only support a single host connection at one time, i.e. no more than one SDP, control, and interrupt channel may be open at a time. A Bluetooth HID that implements the Virtual Cable feature shall have sufficient resources to remember a minimum of two hosts, and four hosts is recommended, to make later reconnection easier without passkey entry.

4.5 Security

Bluetooth security measures, such as authentication, bonding, and encryption, are optional in all Bluetooth HIDs except keyboards keypads, and other types of devices which transmit biometric or identification information. Similarly, hosts or host applications that can potentially receive sensitive information from a Bluetooth keyboard or keypad should request a secure connection. This is to ensure that users are not confused by the availability of both secure and non-secure Bluetooth keyboards, and provides a clear value-added security benefit to Bluetooth keyboards over existing wireless keyboards on the market. It is normally the responsibility of the host to initiate security procedures.

4.6 Trusting and Untrusting Devices

Both the host and the HID shall provide a means of unbonding if they provide a means of bonding. It shall also be possible to unbond a device with or without the presence of the other party, in order to resolve an erroneous bonding situation.

4.7 Behavior in Various Error Conditions

4.7.1 Link Loss

When a “virtually cabled” (see Virtual Cables and Connection Re-establishment) Bluetooth HID goes out of range and back in range, up to a maximum timeout period, the link shall be automatically re-established by the host if SDP attribute HIDReconnectInitiate is false and by the HID if HIDReconnectInitiate is true.

4.7.2 Pairing or Bonding Refused by HID

If a HID refuses pairing, the host shall display a comprehensible error message to the user.

4.7.3 Connection Refused by HID

If a HID refuses a connection with a host, when possible the host shall display a comprehensible error message to the user, preferably with a list of possible causes (such as a connection with another host is already active).

5 Bluetooth HID Host Requirements

5.1 Application and Host Requirements

5.1.1 Generic HID Compliance

PC host applications should be developed to work with a HID device independently of the communications bus that connects them. PC HID applications should adhere to the standard HID client rules for the particular platform and operating system in question.

5.1.2 Fixed Format Reporting

Resource-limited non-PC host applications which implement HID may optionally choose to implement only the Boot Protocol, which is the minimum required level of host functionality (see Boot Mode Operation). This fixed-format reporting method eliminates the need for the HID parser. There are boot protocols for pointing devices and keyboards. Support for boot protocol features and commands is optional for hosts.

5.1.3 Rules for Minimum Level of Host Compatibility

To ensure a base level of interoperability between Bluetooth HID hosts and devices, Bluetooth HID hosts that support any type of pointing device or keyboard functionality shall support the corresponding pointing device or keyboard boot protocol, or the normal HID protocol. Support of the normal HID protocol in the host provides boot mode keyboard and mouse support by definition.

5.2 HID Class Driver Support

5.2.1 HID Class Driver Interface to Bluetooth Stack

Hosts with existing support for the HID protocol running over USB shall supply an adapter driver, which generates and decodes the necessary packet header and L2CAP requests for running the HID protocol over a Bluetooth data channel. The host adapter driver shall provide the means for applications to establish and terminate HID protocol connections to a Bluetooth Human Interface Device. Similarly, it shall allow Bluetooth Human Interface Devices to establish and terminate HID protocol connections to the host application after initial pairing is complete. Hosts shall always initiate pairing.

5.2.2 L2CAP HID Protocol Support

The implementation of L2CAP on the host should meet the default minimum MTU (maximum transmission unit) of 48 bytes, although the default value of 672 is recommended. The host BT-HID adapter driver should implement packet segmentation and reassembly up to the largest practical size (64k bytes maximum) in order to maintain compatibility with all possible Bluetooth HIDs, but this is not required for hosts which support only Boot Protocol mode. Both the HID and the Host should look at the other's L2CA_ConfigReq primitive to determine the MTU they use for Segmentation and their own MTU for Reassembly.

5.2.3 SDP Support in Host

The host is required to implement an SDP client. For SDP record details, see Section 7.10.

5.3 General L2CAP Requirements

5.3.1 QoS (Quality of Service) Requirements

The HID protocol was first implemented on the Universal Serial Bus. USB provides hardware that can guarantee maximum data latency to and from HIDs by means of the interrupt pipes. Although Bluetooth 1.1 contains support for synchronous channels (SCO links) with guaranteed latency, these are intended for voice, and all other traffic is placed on the ACL channel. The ACL channel and its higher-level abstractions do not contain hardware mechanisms for prioritizing one L2CAP data channel over another, so any Quality of Service requests at the L2CAP connection layer must be handled by L2CAP or lower layer software. The authors of this document realize that it may not be practical for some hosts to manage the instantaneous bandwidth of several connections, and therefore L2CAP QoS may not be implemented in many versions of the software stack. However, it is recommended for HID hosts to implement the L2CAP QoS interface and be able to perform QoS negotiation with devices. Supporting the interface will eventually allow improved performance for first generation Human Interface Devices when the underlying QoS system is put into place in future specifications.

For example, L2CAP QoS settings for high performance mouse; see Section 7.16.1.

5.3.2 Use of L2CAP Channel Identifiers (CIDs)

In the case of multifunction Human Interface Devices (such as keyboard with integrated pointing device), the HID protocol itself allows distinguishing the origin or destination of the data by means of Report IDs. Thus, different CIDs are not needed for each function in a multifunction device.

HID reports inherently have different Quality of Service requirements. Input and Output reports are defined for data with high priority and low latency requirements, while Feature reports support the background control data. Because of their real-time requirements, Input and Output reports require Guaranteed responses, while Feature reports only have Best Effort expectations. Different L2CAP channels can be assigned different Quality of Service parameters, so we require separating the report types into two L2CAP channels in order to assign different QoS levels. Input and Output reports are carried on the “Interrupt” channel, and Feature reports are carried on the “Control” channel. The host shall be able to handle separated Interrupt and Control channels.

5.4 Link Level Requirements

5.4.1 Authentication, Pairing, Bonding

Support for authentication, pairing, and bonding routines in hosts and applications of Bluetooth Human Interface Devices is optional, although authentication is strongly encouraged and encryption is recommended for host application programs requiring users to enter sensitive information on a Bluetooth keyboard or keypad. It is normally the responsibility of the host to initiate security procedures, however HIDs are optionally allowed to initiate authentication (after initial pairing, initiated by the host, has been completed) to prevent host spoofing. Hosts shall be able to initiate authentication and pairing before and after the baseband connection has been established.

5.4.2 Special Considerations for Keyboards and Keypads

Pairing with Bluetooth keyboards and keypads requires passkey entry on a host that may not have a wired keyboard or other alternate method of entry for the passkey. In this case, it is recommended that the host generate a random passkey code that can be displayed to the user for entry into the Bluetooth keyboard or keypad. A keyboard may be identified without performing Service Discovery by using the specific Class of Device bits that are returned in the FHS packet when the keyboard responds to an Inquiry. The host may sort device responses based on this field.

Application note:

Most keyboards do not have knowledge of what is printed on the keytops, instead sending a scan code to the host which then interprets the code differently depending on what language setting is in effect. When the Bluetooth passkey is entered it is not transmitted to the host, so the keyboard must make some assumptions about what is on the keytops. A safe assumption for manufacturers of multi-language keyboards is that the number keys are consistent between languages. Therefore a host with no alternate means of user input should only ask for a numeric passkey for keyboards.

5.4.3 Hosts with Limited Input Capability

In the case where pairing with a new Bluetooth keyboard is required and there are no other means of responding to prompts for proceeding with the pairing procedure, a non-secure connection may be established for purposes of prompting the host to begin the pairing process. In this case, the host or host application shall ensure that keyboard operation cannot continue without performing the pairing process.

5.4.4 Use of Encryption

Encryption is strongly recommended for host applications that allow entry of sensitive information such as user names and passwords through a Bluetooth keyboard or keypad. All Bluetooth keyboards and keypads shall support any encryption key size, 8 to 128 bits, requested by an application. It is always the responsibility of the host to

initiate the encryption procedure, and hosts shall also always accept encryption requests if encryption is supported.

5.4.5 Connection Handling Rules

5.4.5.1 HID Protocol Connection Establishment

Both HID_Control and HID_Interrupt L2CAP channels shall be established in order for the HID protocol connection to be considered established. The HID_Control connection shall be initiated first. It is permissible for the host (or the device, in the event of device initiated reconnection) to configure both channels simultaneously (i.e. configuration sequence can overlap). The HID channels shall be closed in reverse order, i.e. Interrupt then Control.

5.4.5.2 Reconnection After Host Reset

If SDP attribute HIDReconnectInitiate is False, the host or host application shall be responsible for re-establishing or initiating the connection to the HID after it (the host) has been reset. If HIDReconnectInitiate is True and SDP attribute HIDNormallyConnectable is also True, the host may also attempt to reconnect after it is reset.

Note that a PC host BIOS will not have the capability of reading the HIDReconnectInitiate flag if the Service Discovery Protocol is not implemented. For this reason, it is recommended that keyboards designed for use with a PC always remain connectable (HIDNormallyConnectable = True) so they can be discovered at boot time.

5.4.5.3 Page Mode Support

Host requesting a connection to a HID device shall perform the paging sequence for at least 2.56 seconds, or the minimum time required to reach a device in page scan mode R2 with no SCO connections present.

5.4.5.4 Page Scan Mode Support

Devices that function as slaves of HID remote controls which initiate the connection as masters shall support the appropriate Bluetooth page scan mode which corresponds to the desired remote control response time.

5.4.5.5 Termination and Re-Establishment of Connection

The responsibility for re-establishing a terminated connection shall be determined by the SDP bit HIDReconnectInitiate. See Appendix A: Message Sequence Charts for examples of connection re-establishment.

In the event of a HID-initiated reconnection of a virtually cabled device, the HID pages the host and establishes the ACL connection. The device may perform a master/slave switch after baseband connection establishment. The device may then re-open the

HID_Control and HID_Interrupt L2CAP channels. The host may then decide to read the device descriptors again through the SDP record or simply verify that the device is one that is already known, either with authentication procedures or by reading device ID information from the SDP record. The HID may optionally also perform authentication on the host.

In the event of a terminated connection due to an out-of-range or interference condition, the device SDP boolean value HIDReconnectInitiate, when set to True, indicates that the HID will be primarily responsible for connection re-establishment. If this is false, this indicates the host is primarily responsible for connection re-establishment. If the device has this SDP value set to false, the device shall enter page scan mode in the event of connection loss to allow the host to reconnect. Similarly, if the device has this SDP value set to true, the host is expected to enter page scan mode to allow the device to reconnect. When automatic reconnect is used with HIDs, it is recommended to use the SDP attribute HIDSupervisionTimeout as described in section 7.11.12 to achieve additional responsiveness in connection re-establishment. Devices which declare SDP attribute HIDNormallyConnectable = True are always in page scan mode (when not connected) and may always be paged by the host, regardless of the other SDP settings.

5.4.5.6 Failed Reconnection

If the host or HID attempts reconnection after connection is lost for an unknown reason, either side may time-out and discontinue attempts to reconnect after 30 seconds. Manual user intervention is acceptable to re-initiate the reconnection process after the retry timeout period has expired.

5.4.5.7 Packet Types

HID hosts shall support DM1, POLL, NULL, and FHS packet types to maintain a minimum level of interoperability with devices conforming to the Bluetooth HID Profile.

5.4.5.8 Support of Low Power Link Modes

The Link Manager on HID hosts shall always allow slave-initiated SNIFF modes. SNIFF mode is mandatory for hosts, while PARK and HOLD are optional. Response time to UNSNIFF and UNPARK should be no more than one beacon interval when no SCO collisions occur. Hosts which have large quantities of data to send to the HID while the link is in one of the low-power modes are responsible for UNPARKING or UNSNIFFING the link if this is necessary to transmit the data to the HID with the required throughput.

Note: Response time is typically more than two poll intervals before data can be sent; one for unsniff request, one for host response, then respond with data to next host poll.

5.4.5.9 Inquiry

Hosts performing Inquiry for HIDs that are limited discoverable should be done using the Limited Inquiry Access Code (LIAC). However, HIDs may support either Limited

Discoverable or General Discoverable mode as defined by the Generic Access Profile [6]. See section 6.2.

5.5 Boot Device Support

5.5.1 BIOS Requirements for Boot Device Support

In order to provide complete Bluetooth keyboard and pointing device functionality in a PC-based host, keyboard or mouse control is often needed before the operating system is loaded, for low-level system configuration. In order for Bluetooth keyboards and mice to function like wired keyboards, the PC BIOS must contain firmware that understands how to communicate to keyboards and pointing devices over a Bluetooth radio. If security is an issue, pairing and authentication procedures must also be performed by the BIOS, and the resultant key shared with the operating system. Bluetooth HID pointing devices and keyboards are required to support boot protocol mode.

Since full wireless keyboard and mouse support requires a BIOS update for most personal computers, another method that can be used for backward compatibility is to develop a USB or combination USB/PS2 Bluetooth adapter which emulates the operation of a wired USB keyboard and mouse whenever the HID protocol is set to boot mode. When the protocol is set to normal mode with the SET_PROTOCOL command, the host Bluetooth adapter can enumerate as a normal USB Bluetooth device.

The PC BIOS may use the Class of Device bits in the FHS packet to discover a mouse and keyboard as an alternative to reading the SDP record of the device.

5.5.2 Keyboard Auto-Repeat Functionality

When in the boot protocol mode (entered with the SET_PROTOCOL command), Bluetooth HID keyboards and keypads provide auto key repeat functionality internally, like USB HID keyboards. When in full HID protocol mode, auto-repeat functionality is provided in the host. However, it should be noted that link loss after a key down event might generate unintended keystrokes until the link timeout occurs.

6 Bluetooth HID Device Requirements

Bluetooth Human Interface Devices shall meet the following requirements of the Bluetooth Specification [5]. In addition to the requirements listed below, all devices claiming conformance to this profile shall meet the requirements of the Generic Access Profile specification (see the Bluetooth Profiles Specification [6]).

6.1 Master/Slave Roles

Bluetooth Interface Devices are providers of a service to a host, that service being to provide input from a human to an application program running on the host. Multiple HIDs are typically connected to a single host; for example, a mouse, a keyboard, and a joystick. For this reason, Bluetooth HIDs should normally be implemented as slaves in the Bluetooth link protocol. (There is one possible exception for remote control devices; see Section 6.6.3.2.) If HIDs were masters, each additional HID master using the host radio as its slave would create an additional piconet (by definition), with corresponding reductions in bandwidth efficiency due to inter-piconet packet collisions and timing uncertainties at the host radio.

Although recommending that Bluetooth HIDs be slaves in the Bluetooth link may be viewed as increasing the power consumption (due to the fact that slaves must listen for a poll), sufficient power-saving mechanisms are provided in the Bluetooth protocols to implement a power-efficient HID device as a slave. In any case, implementing a HID as a master is not prohibited and may be more efficient in some cases; see Section 6.5, remote controls.

Automatic reconnection procedures also allow a HID to function as a master during the connection re-establishment process. See Virtual Cables and Connection Re-Establishment for additional details about the reconnection process.

6.2 Discoverability

Human interface devices may be implemented as limited discoverable or general discoverable devices (see [6], Profiles Specification). Limited discoverable mode is strongly recommended for HIDs used as personal devices or devices which always have a 1:1 relationship with a host. For example, HIDs such as mice and keyboards are normally associated with a single host or a fixed number of hosts which are known in advance. However, Bluetooth HIDs can also include industrial devices such as remote sensors and measuring devices, which might need to be available for public use or for multiple hosts which are not known in advance. Similarly, these devices may be located in hard to access areas and requiring a button press or power cycle to initiate limited discoverable mode is not practical. In these cases general discoverable mode might be more appropriate.

For reasons of improving out of the box user experience for consumer devices, it is encouraged to make devices discoverable from when batteries are first installed until they are first paired with a host, with a timeout period.

6.3 Connectibility

Bluetooth Human interface devices that are implemented as slaves indicate their connectibility (whether they are in page scan mode or not) by the state of the SDP attribute HIDNormallyConnectable. If this attribute is True, the device shall stay in Page Scan mode when no active connection is present. If this attribute is false, they may shut down completely when no active connection is present. If the host cannot contact the HID during the initialization sequence, and if the SDP attribute HIDReconnectInitiate is set to TRUE, HIDs are permitted to page the host and perform a master/slave switch to restore the connection. This will help avoid creating the situation where the host is constantly paging devices that have gone out of range (possibly forever) and using up bandwidth in the piconet. However, designers must be aware that it may take several seconds to re-establish the connection, depending on the page and page scan modes used, and the presence of any audio (SCO) connections. The resultant latency may be annoying to the user. Devices wishing to avoid the power consumption of remaining in page scan mode continuously can set the HIDNormallyConnectable SDP attribute to False. In this case the host is the one that must remain in Page Scan mode to be ready to accept a connection request from the device. See Table 1: Host and HID paging behavior as a function of related SDP declarations.

It is recommended that keyboards intended for use with a PC set this attribute to true so the presence of a keyboard can be verified at boot time. See 7.11.13 for a complete description of the HIDNormallyConnectable attribute.

HIDReconnectInitiate	HIDNormallyConnectable	Host action	HID action	Note
False	False	Nothing	Nothing	Host normally unable to connect without user action
False	True	Page	Page Scan	Host restores lost connection
True	False	Page scan	Page	Device restores lost connection
True	True	Page or Page scan	Page and Page scan	Device restores lost connection, but host can also page

Table 1: Host and HID paging behavior as a function of related SDP declarations

Both the HID_Control and HID_Interrupt L2CAP channel L2CA_ConnectRsp primitives shall be received in order for the *virtual cable* connection to be considered “established”. The HID_Control connection shall be initiated first. The configuration of the HID_Control channel may overlap the connection of the HID_Interrupt channel.See section 6.4 for more information on *virtual cables*.

It is recommended that devices always allow role switches in order for the host to manage or prevent creation of a scatternet condition, by setting the proper flag in the LMP_features response packet.

6.4 Virtual Cables and Connection Re-Establishment

The term *virtual cable* is used to indicate that the HID has a 1:1 association with a particular host. It is useful to use this term as an adjunct to the Bluetooth terms *pairing* and *bonding*. Pairing is the process of creating a trusted relationship between two devices, which can be either a transient or semi-permanent relationship. If the resultant authentication keys generated from the pairing process are stored for later re-use, a semi-permanent relationship is established, and the devices are then bonded. However, this says nothing in particular about when devices should try to establish a connection with each other, and what should happen when an established connection is lost for an unknown reason (interference, out of range, dead batteries, etc).

Typically a human interface device is a personal device that is used with one host at a time. When a *cabled* HID is plugged into a host by a user, the user creates both a “bond” between the devices and a 1:1 relationship. To use the device with another host, the cable must be manually unplugged and plugged into another host. User intervention is required.

If a Bluetooth HID is *virtually cabled* to the host, this means five things:

1. The device has set the HIDVirtualCable attribute in its SDP record to True.
2. There is a 1:1 relationship with the host (data cannot flow to 2 or more hosts simultaneously). When a connection is established to a host, a device shall exit page scan or page modes, and inquiry scan or inquiry modes for the duration of the connection.
3. Automatic reconnection is desired if the connection is dropped for any unknown reason.
4. The HID will refuse connections from other hosts to which it is not virtually cabled.
5. The device is explicitly marked as virtually cabled on both sides of the link.

If the connection is dropped unexpectedly (e.g., timeouts):

1. If the SDP attribute HIDReconnectInitiate is true, the device shall attempt to reestablish the connection to the host by entering Page mode.
2. If HIDReconnectInitiate is true, the host shall allow the device to reestablish the connection by entering Page Scan mode. . If the device has HIDNormallyConnectable attribute set to True, then the host may also page the device to re-establish the connection.
3. If the device has set SDP attribute HIDReconnectInitiate to false, it shall allow the host to reconnect by entering Page Scan mode.
4. If HIDReconnectInitiate is false, the host shall attempt to reestablish the connection by entering Page mode.
5. Upon the creation of a new virtual connection or a reconnection, the device shall be in the default *data state*. After the establishment of a new virtual connection, the host shall initialize the *data state* of the device. Upon reconnection, the host shall then restore the *data state* of the device. The restored *data state* will be based on the *data state* prior to the disconnect event, and any pertinent events that have taken place on the host in the interim. The "*data state*" of a device is application dependent. E.g., the *data state* of a keyboard consists of state of the Caps Lock, Num Lock, and Scroll Lock indicators, and its Protocol setting (normal or boot) and idle rate setting if used.

   For example, consider a host has turned on the CAPS LOCK LED on a keyboard and then the connection times out because of an out-of-range condition. When the connection is automatically re-established, the keyboard will have turned off all LEDs, including the CAPS LOCK key (default data state). The host shall restore the data state keyboard LEDs by turning on the CAPS LOCK LED. This example also addresses the case where the user has replaced the batteries and the device lost its last known state.

Similarly, if disconnection of a virtually-cabled Bluetooth HID and reconnection to another host is required, user intervention is required to "unplug" the virtual cable by means of some manual action (such as a button press on the device) before "plugging in" to another host.

If a Bluetooth HID wishes to have the link to the host behave as a “virtual cable”, it shall set the bit HIDVirtualCable to True in its Service Discovery record.

An intentional L2CAP disconnection with reason code “user ended connection” is NOT interpreted as a Virtual Cable Unplug event.

It is mandatory for a device to accept a connection for some period of time during initialization to allow host initiated pairing, and mandatory for the device to accept connection termination from the host. It is optional for the device to initiate or terminate a connection.

6.4.1 Adding Virtually-Cabled Devices

Below is an example of the sequence of events that a host and device will execute to initiate a “plug” operation with a virtual connection. These examples assume a **Connect** button exists on the HID that can be used to manage plugging and unplugging. This is only an example and other methods are also valid.

- User action on Device side: The user presses the **Connect** button on the HID to place the device in **limited discoverable** mode. Any previous connections are disconnected at this time.
- User action on Host side: The user initiates a Discover Devices dialog to perform a limited inquiry. An “Add New Device” button is available on the Discover Devices dialog. The user selects the desired device from a list of devices in range.
- Create and setup connection (including security if applicable).
    - Known devices that previously have had a “virtual cable” with the host will initiate connection setup with previously stored information.
    - Unknown devices will exchange and store information in a file (persistent storage), then connect to each other.
- A virtual cable connection is established when the HIDVirtualCable attribute is True, both the control and interrupt channels have exchanged their L2CAP connection requests and responses, and their L2CAP connections have been configured in both directions.
- In case of a link loss, either side of the “virtual cable” connection can try to restore the connection. The device shall set SDP attribute HIDReconnectInitiate to True if it wishes to initiate the reconnection after link loss. HIDReconnectInitiate identifies which side enters Page Scan versus Page mode. If a device declares the HIDNormallyConnectable attribute and sets it to True, this indicates it will normally be in Page Scan mode regardless of whether it wishes to initiate the reconnection or not.
- Host shall keep the user informed about the status of plugged (in range) devices.
- Plugged devices only communicate with the device on the other end of the virtual cable.

6.4.2 Removing Virtually-Cabled Devices

Below is an example of the sequence of events that a host and device will execute to initiate an “unplug” operation with a virtual connection. Other implementations are possible.

- A virtual cable is unplugged:
    - The user pressing the **Connect** button on the HID will unplug the virtual cable from the current Host. Before a device disconnects, the device shall attempt to inform the host. The action of pressing the “connect” button will then initiate a “plug” operation with another Host.
- Unplugged devices shall be marked as known and put into a “most recently used list” of known devices to facilitate future re-connecting (see Appendix B).
- Unplugged devices need user intervention to establish a connection.

6.5 Power Management

Since in the great majority of cases the Bluetooth HID devices will be battery powered, the goal is to implement Bluetooth HIDs with battery life comparable to competing proprietary wireless technologies. There are features available for power savings that can substantially reduce the power used by the Bluetooth transceiver.

6.5.1 Bluetooth HID Power Management Philosophy

In general, Bluetooth HID devices shall be responsible for their own power management. The host shall not be required to manage the various power states of the device. This would require knowledge of the features and usage model of every specific HID which is connected, which is contrary to the idea of HID; that is, to use a one class driver for a wide variety of Human Interface Devices. However, the host may notify the HID of power state changes in the system (e.g., standby, suspend) which the HID may choose to respond to.

6.5.1.1 Use of Bluetooth Low Power Modes

Battery life is a key product-differentiating feature for wireless human interface devices. The judicious application of Bluetooth’s low-power modes, PARK, SNIFF, and HOLD modes offered by the Bluetooth Link Manager can substantially reduce the average power consumption of Bluetooth HIDs with a corresponding increase in battery life The discussions below are intended to be for application information only; there are no mandatory usages of the low-power modes for Bluetooth HIDs.

6.5.1.2 Use of HOLD Mode

HOLD mode can be requested by the HID when it is known that no communications will occur for a relatively long time. The HOLD request has time as a parameter, which can be negotiated by the master and slave. However, in most cases a HID does not know

this in advance, as a button can be pressed or a pointing device can be moved at any time without warning. If a HID needs to send data during the hold time, there is no protocol mechanism to end the hold time prematurely and send data to the host. Therefore, if HOLD mode is used as a power-saving mechanism, the HOLD time should not exceed the maximum latency desired by the device. Similarly, since events must be stored until the HOLD interval has expired, the interval shall be no longer than the HIDs capacity to store the maximum number of events that could occur during the interval. Support of HOLD is optional for devices.

6.5.1.3 Use of PARK Mode

PARK mode can be used by HIDs to drop to a minimum level of activity to save power, while remaining synchronized to beacons from the master of the piconet. Beacon interval is negotiable and is a latency/power tradeoff. Support of PARK mode is optional for devices and hosts. Park mode has a disadvantage over SNIFF in that data cannot be sent during the scheduled beacon interval; the connection must be first unparked.

6.5.1.4 Use of SNIFF Mode

SNIFF mode provides a way of balancing device latency with power consumption with fine degrees of adjustment. A HID may request various SNIFF intervals to reduce its radio duty cycle and thus its power consumption according to any user model of operation that is appropriate. In some cases, the normal link operating mode can be SNIFF, since with a short SNIFF interval the HID may still be able to transmit data fast enough to meet the latency and data reporting requirements of the application. A key advantage of using SNIFF mode instead of PARK is that the device maintains its active member address and data reports may be sent without exiting this mode. Support of SNIFF mode is optional for devices and mandatory for hosts.

Note: Due to the Bluetooth clock accuracy specifications (+/- 20 ppm), a worst-case clock accuracy situation could potentially require a maximum beacon interval of less than 250 ms in order for devices in Park, Sniff, or Hold modes to stay synchronized. The best link manager implementations will widen the receive window as the beacon interval is increased, but at the expense of additional power consumption. If loss of synchronization occurs, link loss will occur and the link must be re-established by the normal paging and connection establishment procedures.

6.5.1.5 UNSNIFF and UNPARK Response Issues

Use of the SNIFF and PARK modes helps to achieve competitive and reasonable battery life for Bluetooth Human Interface Devices. However, the Bluetooth specification (V1.1) [5] doesn’t provide for time-bounded response to these commands. Although this profile will require the full support of UNPARK and UNSNIFF in the HID host, there is no way to guarantee that the host will respond in a timely fashion given the current specifications. If the host is supporting one or more SCO links, the response time may be significantly longer.

Changing SNIFF or PARK intervals requires a transition to ACTIVE mode and back to SNIFF or PARK. Since the sending of the LMP_unsniff_req by the slave will use a SNIFF slot, and the master LMP_unsniff will use the following slot, there may be up to three SNIFF intervals elapsed before any data can be sent to the host. Designers should take this latency into account, and for latency reasons may want to use the first available SNIFF slot after an event to send the event data, rather than the LMP_unsniff_req.

6.5.1.6 Host Changing of SNIFF and PARK Parameters

Although either master or slave is allowed to change PARK and SNIFF parameters in the Bluetooth Specification [5], host changing of parameters is discouraged for Bluetooth HIDs since it is the responsibility of each HID to power manage its own connection. The host might set parameters that reduce the HID performance intended by the manufacturer, or render it inoperable. The only case where host changing of parameters would be desirable is when the host operating state is changed (e.g., put into suspend or standby by a timer or a user).

6.5.1.7 Example Power State Diagram for Bluetooth HID

Figure 2 shows an example usage of Bluetooth power saving modes in a pointing device. Seven states have been defined which have decreasing levels of power consumption but also decreased responsiveness (latency) to user input. The implementor is free to define more or fewer power management states as the user model dictates; this is only an example.

Busy state. In this mode, the Master is actively polling the HID for data at a rate near 100 polls/second, or about once every 16 slot times. Continued user activity (motion or button presses) keeps the device in Busy state. If there has been no activity for a few seconds (determined by user model), the HID transitions to Idle Active state.

Busy active state. This is a brief return of the connection to active state to renegotiate the SNIFF interval to the Busy state interval time.

Idle active state. This is a brief return of the connection to active state to renegotiate the SNIFF interval to the idle interval time.

Idle state. The slave requests the master to enter SNIFF mode with a SNIFF interval which is chosen based on desired latency and average power consumption. In this example, the SNIFF interval is 50 ms, or about every 80 slots. Although the HID can wake up immediately after an event, it may have to wait up to 100 ms to transmit its data to the host, and therefore must have enough buffer space to store 100 ms of events. If an event occurs, the slave requests the master to leave SNIFF mode. If there is no further activity for a longer period, the HID transitions to Sleep state.

Suspend active state. This is a brief return of the connection to active mode to renegotiate the SNIFF interval to the suspend interval time.

Suspend state. PARK or SNIFF mode with a longer beacon interval can be used for a lower power state called Suspend. In this state, any activity detected will result in the HID requesting to be unparked and transitioning back to the Active state. As long as the master continues transmitting (meaning the host is not turned off) the HID will remain in this mode. If link loss occurs due to the host being turned off without warning, or the host moving out of range, the Lost Link state will be entered.

Lost link state. In lost link state, the HID is in its lowest power consumption state. In order to re-establish a connection with the host when the host is reset, it may periodically listen by way of entering Page Scan mode, or alternatively page the host when an event occurs. It is recommended that Page Scan mode R2 be used as it requires the lowest power and is extremely low duty cycle. This places a requirement on the host for Page mode support (see Section 5.4.5.3). Connection re-establishment will take up to 2.56 seconds or more if the host is supporting SCO connections.

Note: Another lower power, but higher latency option for the lost link state is to shut down completely, wake up, and page the host when activity is detected. This requires support for paging and master/slave switch procedures.

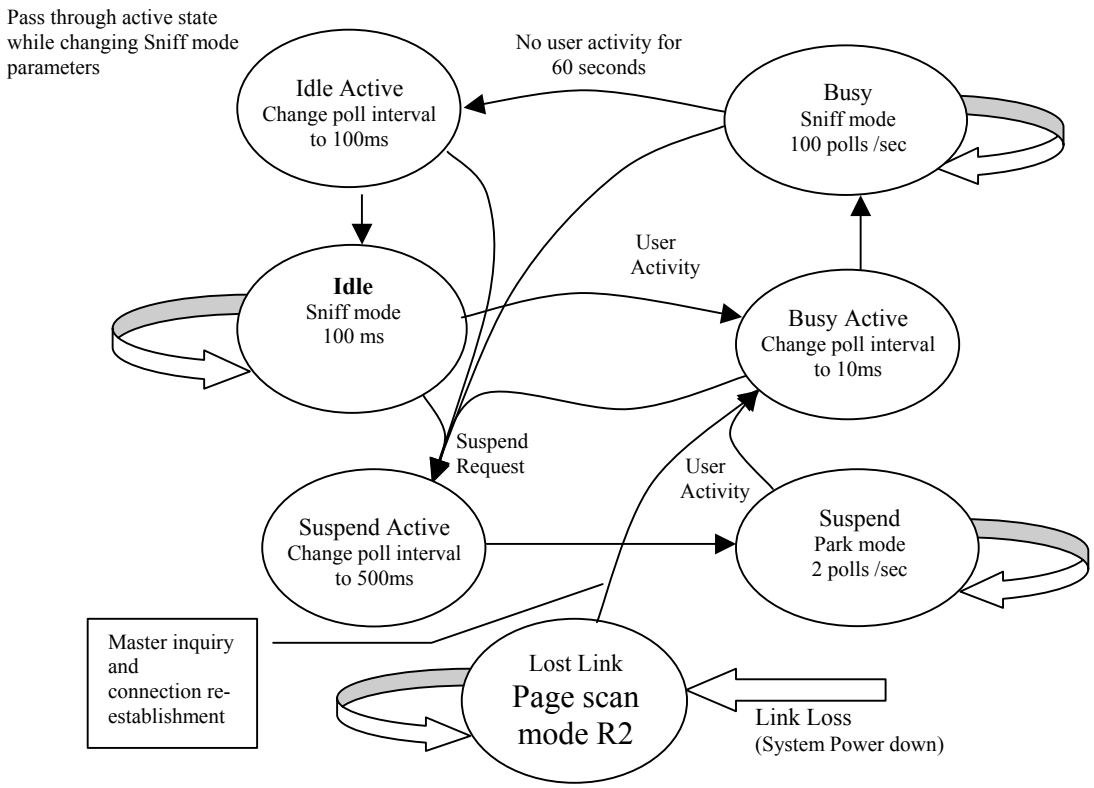


Figure 2: Example Power State Diagram for Bluetooth HID

6.5.2 Power and Latency Tradeoffs

The intervals in the SNIFF and PARK modes can be adjusted to provide a tradeoff between power consumption and latency of the first event that transitions the device into Active mode. This is implementation-specific and depends on the usage model and available power of the particular device.

6.5.2.1 Minimum Duty Cycle in Suspend Mode

Page scan mode R2 specifies that the device need only listen for a page by the master once every 2.56 seconds. Since a page scan takes about 6 ms, the active device duty cycle only needs to be 0.23 percent, low enough for most HIDs to achieve suspend current of less than 100 microamps. Suspend mode can also be implemented with a long interval SNIFF or PARK mode, or with no radio activity at all if the HID is able to page the master and re-establish the connection when it has data to send. In this case the HIDReconnectInitiate SDP bit value will be set to True.

6.5.2.2 Behavior when Host Connection Lost

If a device has explicity marked itself as Virtually Cabled and has set attribute HIDReconnectInitiate, it shall page the device address of the last host that it was connected to and paired with when it needs to transmit data, and no host connection is present. If the host responds, the HID may initiate a master/slave switch and reconnect the L2CAP channels. If the host does not respond, the device shall repeat the process until a timeout value is reached or until the next input occurs. See Figure 3 for a state diagram of desired behavior when the connection is lost.

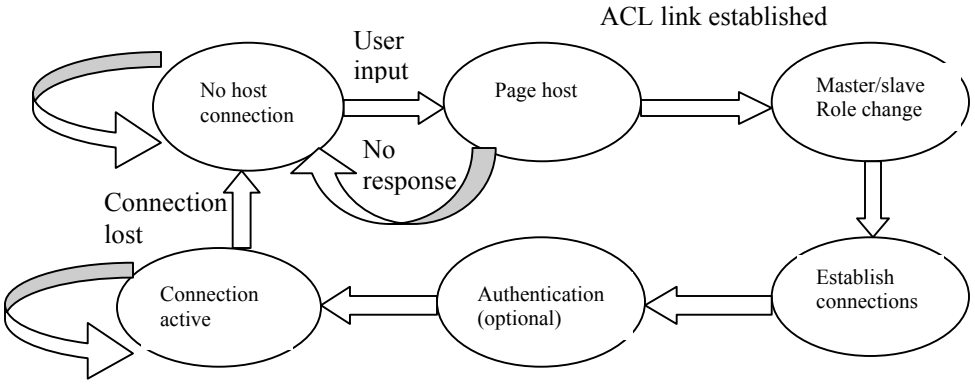


Figure 3: Connection Re-Establishment when Lost

6.5.3 Low Battery Notifications

Battery powered HIDs can optionally utilize the available HID protocol messages to inform the host of a low battery condition. There are standard Usage Tables available for power status and battery reporting in the HID protocol; see USB HID Usage Tables, V1.1.

6.6 Latency and Performance

In general, the requirements and suggestions in this section apply to devices in an active state. Devices in low power modes may be unable to meet these requirements.

6.6.1 General Requirements

The general requirement for Bluetooth HIDs is that they have actual and user-perceived performance that is equal to wired devices. By performance, we mean response time (latency) and throughput (bandwidth). Because of the relatively high bandwidth of the radio and most possible host radio connections (with the exception of RS-232), equal or higher performance than low-speed USB devices should be achievable.

6.6.2 Requirements for Gaming and Pointing Devices

6.6.2.1 Input Latency

The Bluetooth RF link, software stack implementation, and host connection setup parameters should add no more than 10 ms of latency, beyond a wired low-speed USB link, between a user button press or action and the action being available to the application program.

6.6.2.2 Output Latency

The Bluetooth RF link, software stack implementation, and host connection setup parameters should add no more than 10 ms of latency, beyond a wired low-speed USB link, between an application program sending an output command, and the output command being received by the Human Interface Device (e.g., force feedback or display commands).

6.6.2.3 Report Rate

The recommended maximum data report rate for pointing devices and gaming devices is 125 data reports per second (same as low-speed USB). A HID data report will typically use two slots (1.25 ms) of time on a piconet, one for the master poll and one for the HID response. The designer should therefore be aware that a HID reporting at 100 per second will use 1/8 of the piconet bandwidth, and a device reporting at 800 per second will use all the piconet bandwidth. Although this profile does not mandate a maximum reporting rate, it is expected that Bluetooth HIDs be good “citizens” in this regard.

6.6.3 Requirements for Other HIDs

6.6.3.1 Remote Monitoring Devices

The latency and report rate requirements for remote monitoring devices (e.g., voltmeters, pressure sensors, etc.) is highly dependent on the application for which they were designed. A barometer, for example, need not send readings more often than every few minutes. A voltmeter might want to send data reports hundreds of times per second in order to catch a transient condition. It is up to the application and device to

determine the appropriate latency and report rate in these cases on a device-by-device basis. Using the HID protocol, it is possible for such devices to describe to the host the report rate that they require.

6.6.3.2 Remote Control

HID devices developed as remote controls for other devices will also have latencies defined by what type of device is being controlled, but will generally be slower than pointing or gaming devices. Since remote controls may function as masters after configuration, report rate is not an issue and instead it is the time for paging and connection establishment that will determine the response time, as well as the page scan mode of the device being controlled.

6.6.3.3 Other HIDs

Other types of Human Interface Devices not mentioned here may implement any report rate desired which is adequate for their function. It is recommended that the report rate be kept less than 125 reports per second in order to preserve bandwidth in the Bluetooth piconet.

6.6.4 Latency Worksheet

The following worksheet is provided to assist designers in accounting for all the latencies through a system with a Bluetooth Human Interface Device. The typical values are examples only and not intended to reflect an actual implementation.

	Latency Source	Typical Value	Actual Value
1	Button or control interrupt response latency in HID firmware	20 us	
2	Time to read value from control or button	5 us	
3	Time to packetize data in HID report format	30 us	
4	Latency through Bluetooth transport, protocol stack, and host controller interface (x2)	1 ms	
5	Latency through host HID driver	1 ms	
6	Application latency (application poll rate)	10 ms	
7	Video output latency due to video frame rate (1/frame rate)	16.7 ms	
8	Total latency	28.76 ms	

Table 2: Latency Worksheet

6.7 Security

6.7.1 Pairing, Bonding, and Authentication for HIDs

The use of Bluetooth pairing, bonding, and authentication procedures in all HIDs except keyboards and keypads is optional and is up to the device manufacturer to determine.

The lowest cost devices may wish to not support these procedures to reduce the cost associated with non-volatile storage for link keys. The host shall always initiate pairing, however all HIDs may optionally authenticate the host anytime after the host-initiated pairing procedures are completed.

6.7.1.1 Mandatory Requirements for HIDs

The following requirements are mandatory to implement for keyboards and keypads:

- Keyboard and keypad security. Extremely sensitive information such as usernames, passwords, and confidential email regularly originate from these devices. For this reason, Bluetooth keyboards and keypads shall support pairing, bonding, authentication, and encryption since the host application can request any of these procedures. Authentication and pairing shall be supported both before and after HID connection establishment for keyboards and keypads.
- Other types of devices. Certain types of devices, which could be implemented with the HID protocol that transmit biometric or other sensitive data used for purposes of identification (e.g., signature capture pads, fingerprint verification devices, retinal scanners, etc.) shall also support authentication and encryption if requested by the host, both before and after HID connection establishment.

6.7.1.2 Recommended Requirements for HIDs

The following are recommended, but not mandatory, security requirements for Bluetooth Human Interface Devices:

- Pointing devices. Pointing devices such as mice are among the lowest cost devices considered for including Bluetooth technology, and as such are not envisioned to support any security features, at least in the early stages of implementation. Once the device's address is discovered by the host, the host may connect to it at will. Only a single host connection need be supported. In the future, manufacturers may wish to distinguish their devices by adding an authentication sequence and storage for host authentication keys. The position information transmitted by a pointing device is context-sensitive and should not require an encrypted connection.
- Gaming devices. Gaming devices such as gamepads, joystick, wheels, and pedals should not require any of the secure modes of Bluetooth. However, if these devices proliferate, or if there are usage models that have a high user density, manufacturers may choose to add authentication features to avoid connecting to incorrect devices by mistake.
- Remote monitoring devices. For wireless remote monitoring devices such as voltmeters, pressure sensors, and alarm sensors, security procedures are left to the application and the manufacturer to determine. For Bluetooth alarm sensors (for example, a remote door switch) it would be advantageous to use encryption in the data reports to avoid the possibility of another Bluetooth unit "spoofing" a closed sensor when it was actually open, for example. This would provide increased security compared to existing wireless alarm sensors on the market.

HIDs that wish to implement a minimum level of security are recommended to use a default PIN code “0000” (four ASCII zeros) when authentication is requested.

6.7.1.3 Recommended Link Key Types

HIDs that use encryption may use their ***unit key*** as the link key, although this is less secure than using a ***combination key***. The unit key is generated at the time of first initialization and may be shared among various hosts. For optional stronger authentication and encryption, , a combination key may be used, which is unique to each HID-host combination. It will be up to the application to request the key type. A HID that supports use of ***combination keys*** shall also support the use of a unit key. A HID that supports only a unit key will refuse a host that requests to generate a combination key.

Note: It is highly recommended by the Bluetooth Security Experts Group that Bluetooth HID keyboard and keypads support combination keys!

6.7.1.4 Recommended Non-Volatile Storage for Host Keys

Devices that support bonding shall provide some form of non-volatile memory in which to store the 128-bit authentication keys. Devices that use combination keys rather than unit keys will require more memory, because a separate key is needed for every host. Encryption keys are derived from the authentication key and only have a lifetime of the duration of the particular connection, so non-volatile storage is not required for encryption keys. Adequate storage for at least four host keys is recommended for devices supporting combination key authentication, and storage for at least two keys is mandatory.

6.7.1.5 Behavior when Key Storage Capacity Exceeded

If combination keys are used for authentication in a particular HID device implementation, and a new host attempts to pair with a device that has already stored the maximum number of keys, the HID device shall accept the new host and overwrite one of the old keys. This behavior is preferable to refusing the connection with a reason code “pairing not allowed” from the HCI, because not all hosts may propagate this error to the user interface, and even then, it is not clear that the user would know what to do with this message. The preferred method of how the HID device chooses the key to overwrite is based on a Least Recently Used (LRU) algorithm. In this case, the HID may re-order or re-prioritize the list of hosts and host keys in non-volatile memory each time it is authenticated with a host. The most recent host always receives the highest priority. If a new host is added to the list, the lowest priority host will be dropped from the list as the new host is added to the top.

For an illustration of this algorithm, refer to Appendix B: Persistent Storage of Known Devices.

6.7.1.6 Behavior when Key Storage is Lost on a Device

If the host fails to authenticate a device that it has previously established a link key with, then the host should pair with the device via passkey entry and perform the bonding and authentication anew. See 5.4.2 for special passkey considerations for keyboards and keypads.

6.7.1.7 Special Keyboard and Keypad Considerations

- **Pairing and bonding with no means of host passkey input**. Wireless Bluetooth keyboard and keypad devices present a special challenge to the host pairing process, as there may be no means to enter the host passkey before the keyboard is active. There are two recommended implementations, one for when the host has alternate means of passkey entry and another where the host does not have alternate means of passkey entry.
- **Host has alternate means of passkey entry**. If the host has an alternate secure means of passkey entry, such as a touchscreen, built in keypad, or keyboard, or other wired means of numeric input, this means shall be used to enter a variable host-side passkey for pairing with Bluetooth keyboard devices. The Bluetooth keyboard will also have a variable passkey, which the host must prompt for entry as well.
- **Host does not have alternate means of passkey entry**. In this case, the recommended method for pairing is the following sequence:

  1 The host, after performing device discovery and upon discovering that a Bluetooth keyboard is available by examining the Class of Device bits in the FHS packet, makes a temporary unsecure connection to the keyboard that will only accept limited keystrokes to allow the user to either begin or abort the pairing process.

  2 If the pairing process is started, the host generates a random number that is used as a fixed passkey and displayed on the screen. This number is used to generate the authentication challenge. The keyboard shall have a variable passkey.

  3 The user is instructed to enter the displayed number on the keyboard and press the <Enter> key.

  4 The authentication process completes, the temporary connection is broken, and a new encrypted connection may be established by the host using the stored link key.

Application Note

When the host requires an authenticated, encrypted Bluetooth connection to its keyboard, there are two principles to remember when implementing a keyboard or keypad pairing procedure:

1 *It shall not be possible to perform pairing and bonding to any Bluetooth unit without physical access to the unit.*

2 *It shall not be possible to determine the passkey (PIN) of a Bluetooth unit without physical access to the unit.*

If these “physical access” principles are adhered to, security of Bluetooth keyboards can approach that of a wired keyboard. The recommended sequence meets these conditions because physical access is required to read the passkey on the host display and complete the pairing process.

6.7.2 Encryption Support

Support for encrypted connections is optional except in the case of keyboards and keypads, and certain types of user identification devices (see Section 6.7.1.1).

6.7.2.1 Mandatory Requirements

Bluetooth keyboards, keypads, and identification devices claiming conformance to this profile shall support encrypted connections with any key size up to 128 bits if requested by the host. The keyboard or keypad shall not refuse an unencrypted connection if the host requests one. The application program running on the host always determines the security level necessary.

6.7.2.2 Import/Export Restrictions for Encryption Products

As of the time of this writing, there are no major import/export restrictions on Bluetooth devices that support key lengths up to 128 bits for the USA and most European and Asian countries.

In some countries, a Bluetooth device that supports greater than 56-bit encryption is allowed, but they may require a license to import, export, or use the device. It is suggested that the device manufacturer become familiar with the detailed requirements of each country and apply for any required licenses. Please refer to the Bluetooth SIG regulatory group database for country specific information on encryption laws.

6.7.2.3 Optional Requirements for Other HIDs

Manufacturers of Bluetooth Human Interface devices other than keyboards and identification devices may choose to implement encryption as a product-differentiating feature. In this case, encryption shall be implemented in the same manner required of keyboards and keypads.

6.7.3 Multiple Host Connections

Virtually-cabled Bluetooth HIDs are only allowed to support a single ACL connection with a single host at any given time. This is desirable from the HID’s limited resource perspective, for reasons of security, and for consistency with the virtual cable concept.

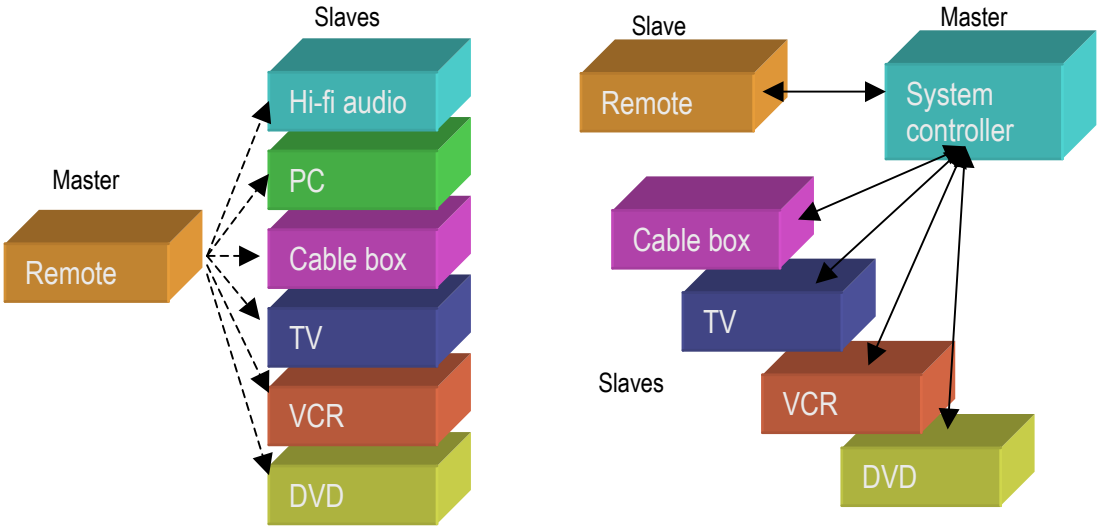
6.8 Bluetooth HID Remote Controls

6.8.1 Remote as Slave with System Controller

In a system of devices with one master controller that is not the remote control, the remote should operate as a slave. One such example would be a PC as piconet master controlling several devices and a remote control operating as one of the slaves, or an A/V home theater receiver as the master controller in a system with the Bluetooth remote as a slave device.

6.8.2 Remote as Piconet Master

For a universal remote control device that is implemented with the HID protocol, it is advantageous from a battery life standpoint to create a connection when there is button activity, transfer the data, and disconnect after a suitable timeout. In this case, the remote control must be the master of the piconet in order to initiate the connection. This method of operation is desirable because of the very small amount of information and the very small duty cycle of operation relative to other types of HIDs makes it very inefficient to operate as a slave (and be constantly polled). The slave devices shall be in a page scan mode that is consistent with the response times desired.



HID remote control as piconet master

HID remote control as piconet slave

Figure 4: Remote Control Configurations

7 Bluetooth HID L2CAP Protocol Specification

7.1 Introduction

This section provides a short overview of Human Interface Devices (HIDs), how they function, and how they communicate over the underlying bus.

7.1.1 Overview

The Bluetooth HID (BT-HID) Profile defines the over-the-air interface for HID devices, which use the Bluetooth RF interface standard to communicate with a host system. In this document there are several assumptions about the layout of the system software stacks on both the device and the host; however, actual implementations may vary as long as they fully support the over-the-air interface defined in this profile.

The Bluetooth HID protocol defines a set of services that can be used between a host capable of supporting HID devices and a BT-HID device. This profile mandates the need for two or more L2CAP channels for conducting transmission of both control and data packets. The BT-HID Header differentiates packet types in a channel.

The communication flow requirements of BT-HID devices depend on the target application. An L2CAP channel represents each communication flow. A local endpoint on the BT-HID device and a buffer on the host terminate L2CAP channels. Information associated with the L2CAP channel is used to identify the respective BT-HID channel. Channel IDs (CIDs) identify the BT-HID channels. See Section 2.1 in the BT “Logical Link Control and Adaptation Protocol Specification” [5] for more information on channels, CIDs, and local endpoints.

A BT-HID host shall open two channels: Control and Interrupt. Separate PSMs are used to distinguish the two channels. The Control channel is always set to the *Best Effort* service type. Low latency data is carried on the Interrupt channel, so the service type will normally be set to *Guaranteed* to ensure Quality of Service. However, if a device does not declare any Input or Output reports, then the Interrupt channel may be set to the *No Traffic* Service type in the respective direction. Refer to Section 5.3.2 in this specification for more details.

Figure 5 illustrates how communication flows are carried over L2CAP channels between the device and host memory buffers. The following sections describe endpoints, channels, and communication flows in more detail.

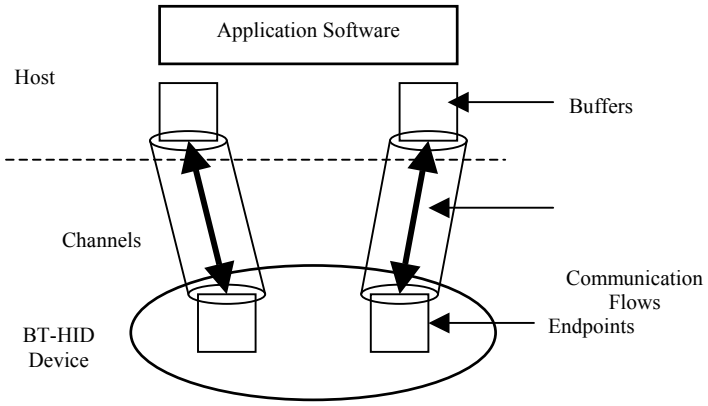


Figure 5: HID Device Communication Flow

HID devices support a variety of data transfer or **Report** types: Input, Output, and Feature. Input and Output reports contain low latency information. Input reports are generated by the device and sent to the host. Output reports are generated by the host and sent to the device. Feature reports are bi-directional, and contain information that is not time-critical. Input, Output, and Feature reports are all optional for Bluetooth HIDs.

The following relationship exists between reports and channels:

- Bi-directional Feature reports are carried on the Control channel.
- Input and Output reports are carried on an Interrupt channel.
- Input and Output reports share the same Interrupt channel; therefore the latency specified with the L2CA_ConfigReq primitive for the Interrupt channel affects both types of reports.
- All BT-HID devices shall communicate via a Control channel and an Interrupt Channel.

The HID Specification [4] defines two special “Boot mode” Report Descriptors, for generic instances of keyboards and mice. A BT-HID keyboard, mouse, or combined keyboard/mouse device shall indicate that they are boot mode-capable by declaring the HIDBootDevice attribute in the SDP record. See Section 7.11.2 for more information.

Two PSMs are defined for BT-HID devices: Interrupt and Control. The values are found in the Bluetooth Assigned Numbers document [8].

7.1.2 Feature Reports

Feature reports assume a **Best Effort** latency requirement. They can carry application-specific data and initialization information that is not time-critical. For instance, a device may use Feature reports to adjust coordinate scaling parameters, enable device options, or determine current device state. Feature reports must be carried on the Control channel. BT-HID devices shall support one Control channel and that channel is always tied to the **HID Control** PSM.

7.1.3 Input Reports

Input reports provide low latency delivery of asynchronous information from the device to the host. For instance, a mouse would use Input reports to send changes in position to the host. All traffic from the device to the host on an Interrupt channel consists of Input reports.

Through an L2CA_ConfigReq primitive, the device declares the latency that it wishes to see on the Interrupt channel. The Bluetooth master is expected to schedule the device at the baseband level with a rate that will meet the requested latency requirement.

Input transfers are optional; if a device does not need to inform the host of asynchronous events, then Input reports do not need to be declared, however the interrupt channel shall be opened (as an unused channel) to simplify the initialization process.

7.1.4 Output Reports

Output reports provide low latency delivery of information from the host to the device. For instance, a force feedback gamepad would use Output reports to trigger force feedback effects in the device. All traffic from the host to the device on an Interrupt channel consists of Output reports.

Through an L2CA_ConfigReq primitive, the device declares the latency that it wishes to see on the Interrupt channel. The device declares the maximum rate at which it can accept consecutive Output transfers, and the host shall not generate transfers which exceed this rate.

Output transfers are optional; if a host does not need to send low latency events to a device, then Output reports do not need to be declared. However, the interrupt channel shall be opened (as an unused channel) to simplify the initialization process.

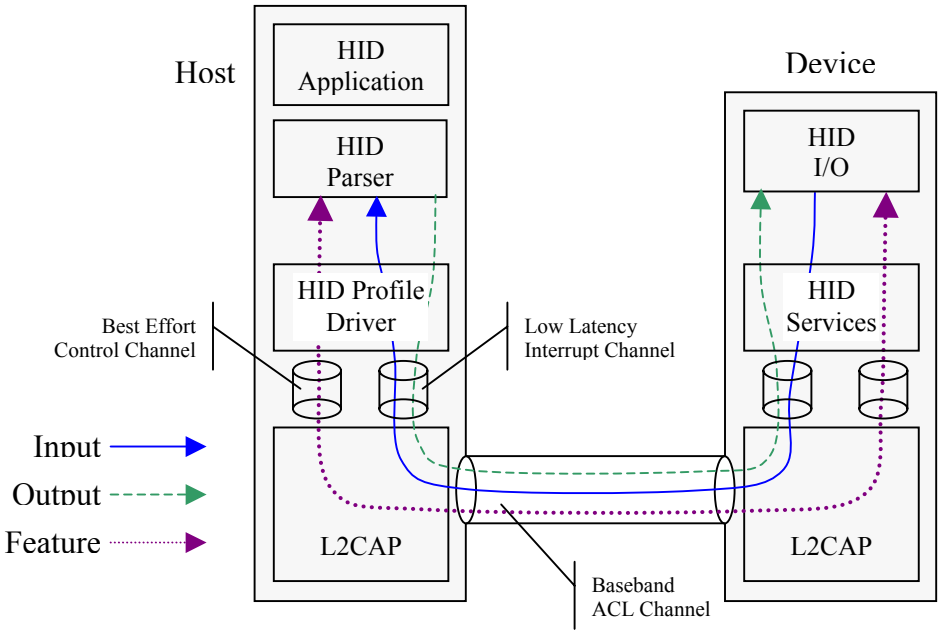
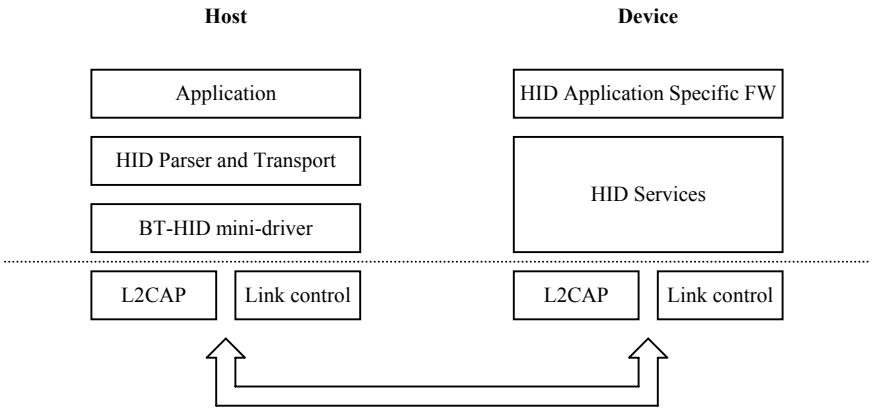


Figure 6: Report Types and L2CAP Channel Mapping

Figure 6 shows how the Input and Output reports are routed over the low latency L2CAP **Interrupt** channel and the bi-directional Feature reports are carried on the best effort L2CAP **Control** channel. L2CAP route all reports of a single baseband ACL channel. Note that the Interrupt channel carries Input reports from the device to the Host, and Output reports from the Host to the Device.

7.2 Architecture

Below is a simplified version of the BT stack as it relates to a HID device.



The host side provides a mini-driver to translate between the HID and Bluetooth stacks. On the Device side, a generic HID Services module provides the services required by any BT-HID implementation. The HID Services module communicates over the L2CAP

interfaces and provides the Bluetooth functionality such as authentication, flow control, packetization of the HID commands.

BT-HID devices use the L2CAP services defined by Bluetooth. In particular BT-HID devices take advantage of the segmentation and reassembly (SAR), and logical connections that L2CAP provides.

7.2.1 Boot Mode Operation

Boot mode was originally defined by USB HID to simplify the design of PC BIOSs; however, it has proved useful for a variety of products with small, embedded operating systems. When a HID device is in Boot mode, a HID parser and SDP client is not required in the host system.

HID devices can support two protocols: Report and Boot. Report protocol is the default mode for all HID devices. The Report protocol requires that a host support a HID parser to interpret the Report Descriptor stored in the SDP information. Boot Protocol is currently only defined for keyboards and mice. When in Boot Protocol, a HID parser is not required because fixed Report Descriptors defined in the USB HID Specification [4] identify the reports generated by the respective devices. Boot Protocol simplifies a BIOS (or embedded application) design by eliminating the need for a HID parser, but it limits the functionality of a keyboard to a basic 103-key PC-AT layout and a mouse to a simple 3 button, 2-axis design.

It is recommended but not required for Human Interface Devices to support all the mouse functions and keyboard scan codes defined by the USB HID Specification [4].

The Minor Device Class field in the FHS packet and the HIDBootDevice SDP attribute are both used to indicate to a host whether a BT-HID device supports Boot Protocol operation and which type; HID keyboard, HID mouse, or a composite of both. HIDBootDevice may seem redundant but is required in case future fixed format reports are defined in addition to the mouse and keyboard formats.

There is a distinction between a “standard” boot report (as defined in the HID Specification [4]) and a Bluetooth boot report. Bluetooth boot devices employ a 1-byte Report ID that precedes the standard HID boot report. Bluetooth boot protocol keyboard reports are 9 bytes (1-byte Report ID + standard 8-byte keyboard boot report), and mouse boot reports are 4 bytes (1-byte Report ID + standard 3-byte mouse boot report). The “standard” boot report portion of each of these Bluetooth boot reports shall conform to the format defined by the respective Boot Report descriptor in the USB HID Specification, Appendix B [4] in order for the data to be correctly interpreted. The keyboard scan codes and pointing device button and XY axis assignments shall conform to the assignments in the USB HID specification [4].

Note: When in boot mode, all reports sent to the device with a SetReport command or over the L2CAP interrupt channel shall include the preceding Report ID. In addition, all reports received from the device with a GetReport command or over the L2CAP interrupt channel shall include the preceding Report ID.

Device	Report ID	Report Size
Reserved	0	N/A
Keyboard	1	9 Bytes
Mouse	2	4 Bytes
Reserved	3-255	N/A

Table 3: Bluetooth HID Boot Reports

Devices that identify themselves as supporting Boot mode shall support the Set_Protocol and Get_Protocol commands. A Bluetooth boot-capable keyboard shall also support Set_Idle and Get_Idle commands. Hosts do not need to support Get/Set_Protocol or Get/Set_Idle if they implement report protocol mode.

Devices may append additional data to boot reports; however, the first bytes of Boot reports shall conform to the format defined by the Boot Report Descriptors in the HID Specification [4], section 4.3, specified by the bInterfaceProtocol byte. Software (BIOS) that receives boot reports shall ignore any appended data in Boot reports. The appended data might be for additional functions on the device that are defined in Report Mode Report Descriptor. Allowing appended data simplifies device design by allowing the same report to be generated by the device, whether it is in Report or Boot protocol mode

The Boot Protocol descriptors in the HID Specification [4] describe reports that a parser-less system (BIOS) would expect to see. However, since the BIOS does not actually read the Report Descriptors when the device is in Boot Protocol, these descriptors are not available from the device. The SDP information only stores the Report Descriptors that are used when the device is in Report Protocol (which may be identical, however)

The default protocol for BT-HID devices after reset is Report. System software that does not support a HID parser shall read the SDP information or Class of Device field from the FHS packet to determine whether the device supports Boot Protocol operation, then it shall issue a SET_PROTOCOL request to place the device into Boot Protocol mode. When the system with a HID parser support is loaded, it shall reset the HID device to restore the default Report Protocol mode without assuming what mode the device is in.

See Section 7.11.2 for more information about the HIDBootDevice attribute.

7.2.2 Channel Initialization

A host or device shall always open both the control and interrupt channels. The Interrupt channel may be open but idle if there are no Input or Output reports declared in the Report Descriptor.

A host or device shall establish the control channel first, then the interrupt channel. Note that “establish” means that an L2CA_ConnectCfm response has been received for the channels L2CAP_ConnectReq. Channel configuration may overlap, however the configuration of both channels shall be complete before sending any data.

A host or device shall always complete the disconnection of the interrupt channel before disconnecting the control channel. L2CAP configuration stage should be short because BT-HID is designed with small CPUs in mind, i.e. keyboards, mice, and joysticks. Most BT-HID devices will only try a couple configuration settings before going to a "no-QoS" setting.

Note that the rules defined by the HIDSDPDisable attribute (section 7.11.8) still apply to the control and interrupt channels.

7.3 BT HID Transaction Header

All messages between a HID device and a host are preceded by a BT-HID Transaction Header (THdr). The Transaction Header is divided into two fields: the Transaction Type and a Parameter. The Transaction Parameter is Transaction Type dependent.

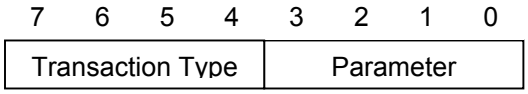


Figure 7: Transaction Header Byte (THdr)

The following table lists the supported Transaction Types:

Hex	Transaction Type	Payload Length (Bytes)
0	HANDSHAKE	1
1	HID_CONTROL	1
2-3	Reserved	
4	GET_REPORT	1 to 4
5	SET_REPORT	1 + Report data payload
6	GET_PROTOCOL	1
7	SET_PROTOCOL	1
8	GET_IDLE	1
9	SET_IDLE	2
A	DATA	1+ Report data payload
B	DATC	1 + Continuation of report data payload
C-F	Reserved	

Table 4: BT-HID Transaction Type Codes

7.4 Transaction Type Descriptions

The Transaction Header (THdr) Types are described below. Transactions consist of a request payload to the device followed by data or a handshake payload from the device. Requests can get report data (GET_REPORT), set report data (SET_REPORT), identify the current protocol (GET_PROTOCOL), change the current protocol (SET_PROTOCOL), identify the current Idle rate (GET_IDLE), or change the current Idle rate (SET_IDLE).

If a GET_ request is parsed without errors by the device, then the device will respond with a DATA payload. If the size of the requested data equals or exceeds the negotiated MTU, then one or more Data Continue (DATC) payloads will follow the DATA payload.

If a SET_ request is parsed without errors by the device, the device will accept the data portion of the payload. If the size of the total payload, including the SET_ header byte(s), equals or exceeds the negotiated MTU, then one or more DATC payloads will follow the SET_ request.

For all SET_ requests and any GET_ requests where an error is detected, then the device will respond with a HANDSHAKE payload.

Note: Reserved fields shall be set to 0 when written and ignored when read.

7.4.1 HANDSHAKE

This code is used to acknowledge requests that do not receive the implied SUCCESSFUL acknowledgement of a DATA payload. The Parameter field identifies the result of the handshake.

Field	Size (Bytes)	Description
Request	1	Bits specifying characteristics of request. 7..4 Transaction Type 0 = HANDSHAKE request 3..0 Result Code 0x0 = SUCCESSFUL. This code is used to acknowledge requests. A device that has correctly received SET_REPORT, SET_IDLE or SET_PROTOCOL payload (including any associated DATC payloads), transmits an acknowledgment to the host. 0x1 = NOT_READY. This code indicates that a device is too busy to accept data. The host shall retransmit the data the next time it schedules the device. 0x2 = ERR_INVALID_REPORT_ID. Invalid report ID transmitted. 0x3 = ERR_UNSUPPORTED_REQUEST. The device does not support the request. 0x4 = ERR_INVALID_PARAMETER. A parameter value is out of range or inappropriate for the request. 0x5-0xD = Reserved 0xE = ERR_UNKNOWN. Device could not identify the error condition. 0xF = ERR_FATAL. Restart is essential to resume functionality.

Table 5: HANDSHAKE Parameter Definition

Note: A device shall return an ERR_UNSUPPORTED_REQUEST result code if an out-of-range or inappropriate Transaction Type is detected.

Note: A device shall parse all fields in a Request byte that are not reserved. If a device detects a value in a “parsed” field that is out of range or inappropriate for the request, then an ERR_INVALID_PARAMETER result code shall be returned.

7.4.2 HID_CONTROL

This code requests a major state change in a BT-HID device. A HID_CONTROL request does not generate a HANDSHAKE response.

The NOP Control Operation is a do-nothing function that can be used to exercise the communications path for diagnostics, debug, and test.

The Hard and Soft Reset Control Operation allows a host to force a device to re-initialize all internal variables. This feature is useful if problems are detected and the device and other recovery procedures have failed. It shall be assumed that a device forgets its Active Member Address after these commands are issued so the device will have to be Paged to bring it back on-line. The Hard Reset Control Operation forces a device to perform a set of Power On Reset Tests (POST). These tests may take a long time to complete. Soft Reset recovery is faster because it only re-initializes the device.

The power management features of Bluetooth consist of changing the modes of operation: Active to Sniff, Active to Park, etc. These modes can be managed to minimize radio power consumption; e.g., a device may be placed in Park mode to temporarily free up an Active Member Address, or to perform a Page or Inquiry. This does not necessarily mean that the services of the HID device are not needed, just that the bandwidth available to it may be limited for a while. The Suspend Control Operation allows the host to explicitly inform the HID device that normal performance is no longer required and that it may power down logic that is not required to wake up the system. The radio subsystem of the device shall remain powered-up with enough functionality to wake up the powered down HID-specific logic. See Section 7.11.10 for more information on the wake up features of HIDs.

When an event occurs that requires the device to wake up the system, the device shall automatically exit suspend mode and power up all circuitry previously shut down by the Suspend Control Operation. If, after a vendor-defined period, the device is unable to reconnect to the system, the device may re-enter suspend mode. If the host decides to exit suspend mode, then it shall send an Exit Suspend Control Operation request to the device.

For example, a mouse may use the Suspend Control Operation request to turn off its shaft encoder LEDs to save power, requiring the user to press a button to wake up the system. A keyboard may lower the frequency that it scans its keys.

Field	Size (Bytes)	Description
Request	1	Bits specifying characteristics of request. 7..4 Transaction Type 1 = HID_CONTROL request 3..0 Control Operation 0 = NOP. No Operation. 1 = HARD_RESET. Device performs Power On System Test (POST) then initializes all internal variables and initiates normal operations. 2 = SOFT_RESET. Device initializes all internal variables and initiates normal operations. 3 = SUSPEND. Go to reduced power mode. 4 = EXIT_SUSPEND. 5 = VIRTUAL_CABLE_UNPLUG. 6-15 = Reserved

Table 6: HID_CONTROL Parameter Definition

A HID_CONTROL packet with a parameter of VIRTUAL_CABLE_UNPLUG can be sent by the host to the device or by the device to the host. This is not a packet that generates a HANDSHAKE packet since the resulting L2CAP disconnection is an implicit acknowledgement. **A HID_CONTROL packet with a parameter of VIRTUAL_CABLE_UNPLUG is the only HID_CONTROL packet a device can send to a host.** A host will ignore all other packets. The recipient of a VIRTUAL_CABLE_UNPLUG packet is responsible for disconnecting the L2CAP channels.

7.4.3 GET_REPORT

This code indicates that the host wants to retrieve a report from the BT-HID device. Upon receipt of this request, the device will return a DATA payload on the Control channel containing the requested report. If the size of the report plus the DATA header equals or exceeds the MTU, then the device will return additional DATC payloads. See Section 7.5.2 for more information.

The difference between Input reports sent on the Control channel and Input reports sent on the Interrupt channel is that Interrupt channel reports are sent whenever one or more fields in the report change. A GET_REPORT(Input) request sent on the Control channel returns the instantaneous state of the fields in an Input report. It does not effect Input reports queued for the Interrupt channel.

Retrieval of an Output report returns a copy of the last report that was received down the Interrupt pipe. If no report has been received, then a device shall return default values or the instantaneous state of the fields, whichever is appropriate.

Retrieval of a Feature report shall return default values or the instantaneous state of the fields, whichever is appropriate.

Polling HID devices using the GET_REPORT transaction is costly in terms of time and overhead, and shall be avoided whenever possible. The GET_REPORT transaction is typically only used by applications to determine the initial state of a device. If state changes occur on a regular basis within the device, then an Input report should be declared so that the changes can be reported over the more efficient interrupt channel.

The type of the report (Input, Output, or Feature) is defined in the Request byte.

The size of the GET_REPORT header may vary from 1 to 4 bytes. The Request byte is always the first byte of the header.

If Report ID main items are declared in the Report Descriptor, then the *Report Type* and the *ReportID* identify the desired report. And a 1-byte *ReportID* field will immediately follow the GET_REPORT Request byte. Otherwise, the *Report Type* field is sufficient to identify the desired report and no *ReportID* field will follow the GET_REPORT Request byte.

Through the SDP or parsing the Report Descriptor, the host knows the size of all reports, where a “report” includes the report data and, if declared, a Report ID. Typically, a host will automatically allocate the appropriate size buffer for the report and the DATA header byte; however, under some circumstances a host may require only a portion of a report. In this case, the Size bit will be set in the Request byte indicating that a 2-byte *BufferSize* field has been added to the GET_REPORT header. Devices shall never return more than a *BufferSize* payload to the host, where the “payload” is comprised of typically a Report ID and the report data. The DATA and DATC header bytes are *not* included in the *BufferSize*. Note that a Report ID shall precede report data

only if Report IDs are declared in the Report Descriptor; otherwise Report IDs shall not be present in GET_REPORT payloads.

If a *ReportID* field exists, the *BufferSize* field will follow it; otherwise, the *BufferSize* field immediately follows the GET_REPORT Request byte

Field	Size (Bytes)	Description
Request	1	Bits specifying characteristics of request. 7..4 Transaction Type 4 = GET_REPORT request 3 Size 0 = The host has allocated a buffer equal to the size of the report. 1 = A 2 byte BufferSize field follows the Report ID. This field indicates the size of the buffer allocated by the host. A device shall limit the returned payload size to BufferSize. Note that the BufferSize must be increased by 1 byte for Boot mode reports to include the Report ID imposed by BT-HID. See Section 7.2.1 for more information on boot mode. 2 Reserved (0) 1..0 Report Type 0 = Reserved 1 = Input 2 = Output 3 = Feature
ReportID	1	(Optional) Report ID of requested report. This field does not exist if no Report IDs are declared in the devices Report Descriptor.
BufferSize	2	(Optional) Maximum number of bytes to transfer during data phase. This field does not exist if the Size field of the Header = 0. BufferSize is little-endian, i.e. the LSB is transmitted first.

Table 7: GET_REPORT Header Definition

The DATA packet returned by a get report is illustrated in Figure 8. Note that the GET_REPORT BufferSize does *not* include the DATA header byte.

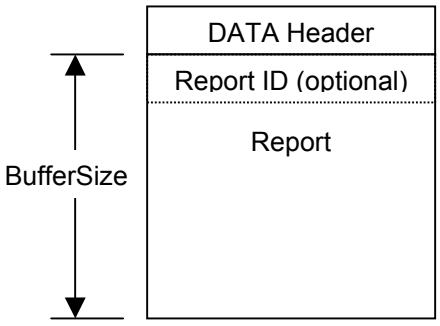


Figure 8: DATA Packet returned by GET_REPORT

If a report including the optional Report ID, possibly truncated to length BufferSize, is equal to or larger than the MTU, then additional packets of type DATC must be sent. Example: the BufferSize = 94 bytes and the MTU = 48 bytes. The device shall send a 48 byte DATA packet followed by a 48 byte DATC, where both packets contain a 1-byte header and 47 bytes of payload. Since the DATC packet containing the last 47 bytes of data completed on a MTU boundary, the device shall send an additional “zero length” (i.e. just the header) DATC packet to mark the end of the transaction. Note that BufferSize is not a segmentation and reassembly mechanism, but only a way for the host to get the first part of a long data report. BufferSize is applied to the total report length before performing SAR, if necessary.

All hosts shall support Get_Report. All devices which declare output reports (host to device transfers) shall support Get_Report.

7.4.4 SET_REPORT

This Transaction Type indicates that the host is sending a report to a BT-HID device. The 1-byte SET_REPORT transaction header is immediately followed by a single report. Only one report can be sent per SET_REPORT transaction.

If the size of the Report Data Payload plus the SET_REPORT transaction header equals or exceeds the MTU, then the device will receive additional DATC payloads. See Section 7.5.2.1 for more information on Large payloads.

The type of the report (Input, Output, or Feature) is defined in the Request field of the SET_REPORT transaction header.

If Report IDs are declared in the Report Descriptor, then a 1-byte Report ID will be the first byte of the Report Data Payload.

The L2CAP Length field includes the size of the report payload and the 1-byte SET_REPORT transaction header (not including subsequent DATC packets, if used). A device shall be capable of receiving the full size of all reports that it declares. Any additional bytes received by the device will be ignored.

The host shall send complete reports. Devices will ignore incomplete reports. Reports may span multiple packets using the DATC header.

Note: In boot mode a 1-byte Report ID shall precede the “standard” HID boot report. See Section 7.2.1 for more information.

All hosts shall support Set_Report. All devices which declare input reports (device to host transfers) shall support Get_Report.

Field	Size (Bytes)	Description
Request	1	Bits specifying characteristics of request. 7..4 Transaction Type 5 = SET_REPORT request 3..2 Reserved (0) 1..0 Report Type 0 = Reserved 1 = Input 2 = Output 3 = Feature
Report Data Payload	n	Report data for the device.

Table 8: SET_REPORT Header Definition

7.4.5 GET_PROTOCOL

This code is used to retrieve the current protocol on the BT-HID device. The device responds with a single byte DATA payload that indicates the current protocol. The format of the GET_PROTOCOL DATA payload is defined in Table 10: GET_PROTOCOL Data Definition.

A device supports this request if the attribute HIDBootDevice is declared True in its SDP record. If a device does not support GET_PROTOCOL it shall return a HANDSHAKE packet indicating an unsupported request.

Get_Protocol is optional for hosts and mandatory for all keyboards and pointing devices.

Field	Size (Bytes)	Description
Request	1	Bits specifying characteristics of request. 7..4 Transaction Type 6 = GET_PROTOCOL request 3..0 Reserved (0)

Table 9: GET_PROTOCOL Parameter Definition

Field	Size (Bytes)	Description
Get Protocol DATA payload	1	Bits specifying characteristics of Get Protocol response payload. 7..1 Reserved (0) 0 Protocol 1 = Report 0 = Boot

Table 10: GET_PROTOCOL Data Definition

7.4.6 SET_PROTOCOL

This code is used to set a specific protocol on the BT-HID device. Special Boot protocols are defined for keyboards and mice. When in Boot protocol, a HID parser is not required because the device only transmits or receives reports in a predefined format. See the HID Specification [4] for a description of the Boot protocols. The default protocol for BT-HID device is Report.

A device supports this request if the HIDBootDevice attribute is declared True in its SDP record.If a device does not support SET_PROTOCOL it shall return a HANDSHAKE packet indicating an unsupported request.

The Parameter field identifies the target protocol. See Table 11: SET_PROTOCOL Parameter Definition.

Set_Protocol is optional for hosts and mandatory for all keyboards and pointing devices.

Field	Size (Bytes)	Description
Request	1	Bits specifying characteristics of request. 7..4 Transaction Type7 = SET_PROTOCOL request 3..1 Reserved (0) 0 Protocol 1 = Report 0 = Boot

Table 11: SET_PROTOCOL Parameter Definition

7.4.7 GET_IDLE

This code is used to retrieve the current Idle setting of the BT-HID device. The device responds with a single byte DATA payload that indicates the current Idle setting. The format of the GET_IDLE DATA payload is defined in Table 13.

The GET_IDLE request is optional for hosts and required for keyboards. For all other HIDs the GET_IDLE request is optional.

Field	Size (Bytes)	Description
Request	1	Bits specifying characteristics of request. 7..4 Transaction Type 8 = GET_IDLE request 3..0 Reserved (0)

Table 12: GET_IDLE Parameter Definition

Field	Size (Bytes)	Description
Get Idle DATA payload	1	Current Idle rate. See Table 14 for a description.

Table 13: GET_IDLE DATA Payload Definition

7.4.8 SET_IDLE

This code is used to set a specific Idle rate of a BT-HID device. Set Idle is optional for hosts and required for keyboards. For all other HIDs the GET_IDLE request is optional.

The Idle interval specifies how often a keyboard sends a copy of the current “key down” information to the host. This command is typically only used when the keyboard is in Boot mode to support BIOS typematic functions, but is active in normal protocol mode. The host can change the Idle rate with the SET_IDLE request to match user preferences for the typematic rate. “Typematic” is the term used for the auto-repeat function when a key is held down.

For example, a HID keyboard always uses Input reports to report changes in key state. If the Idle rate is non-zero, then the keyboard will retransmit the last Input report at the Idle interval until something causes the report contents to change. This could be because a key was pressed or released. When this happens, then the keyboard will restart the idle timer and begin repeating the last Input report until the next change occurs.

The Idle Rate parameter field identifies the target idle rate. The required default idle rate value for keyboards is zero (infinite idle rate).

Note: A device’s idle rate can be set to an interval shorter than the current SNIFF or PARK intervals. Devices should adjust their SNIFF or PARK intervals to maintain the idle transmission rate since some hosts use it for timing information. Hosts are recommended to set the IDLE rate to infinity so that devices can maximize power conservation.

Caution: Use of a non-zero idle rate by a host is NOT recommended due to the adverse effect on power consumption which occurs when the device must transmit data reports continuously. Support for IDLE mode in Bluetooth HID devices is only present for compatibility with USB HID.

Field	Size (Bytes)	Description
Request	1	Bits specifying characteristics of request. 7..4 Transaction Type 9 = SET_IDLE request 3..0 Reserved (0)
Idle Rate	1	When 0 (zero), the Idle Rate is infinite. The device will inhibit Idle reporting forever, only reporting when a change is detected in the report data. When the Idle Rate is non-zero, then a fixed duration is used. The duration will be linearly related to the value, with the LSB being weighted as 4 milliseconds. This provides a range of values from 0.004 to 1.020 seconds, with a 4 millisecond resolution. If the Idle Rate is less than the specified Interrupt channel latency, then reports are generated at the Interrupt channel latency rate. If the given time duration elapses with no change in report data, then a single report will be generated by the endpoint and report inhibition will begin anew using the previous duration.

Table 14: SET_IDLE Parameter Definition

7.4.9 DATA

This Transaction Type identifies a HID payload. All DATA payloads on the Interrupt channel that flow from the device to the host are Input Reports. All DATA payloads on the Interrupt channel that flow from the host to the device are Output Reports. The L2CAP Length field identifies the size of the DATA payload (including the 1-byte DATA header).

The *Report Type* field of DATA transactions on the Control channel will be set to *Other* for responses to Get Idle or Get Protocol requests, and Input, Output, or Feature, for GET_REPORT requests.

A DATA transfer has no associated HANDSHAKE response.

A DATA transfer can be followed by one or more DATC transfers if the size of the payload equals or exceeds the MTU. See Section 7.5.2.1 for more information on the handling of large payloads.

DATA transfers are mandatory for hosts and devices.

The DATA transaction header has the following encoding:

Field	Size (Bytes)	Description
Request	1	Bits specifying characteristics of request. 7..4 Transaction Type 10 = DATA request 3..2 Reserved (0) 1..0 Report Type 0 = Other 1 = Input 2 = Output 3 = Feature
Report data payload	n	Report data for the device.

Table 15: DATA Parameter Definition

7.4.10 DATC

Data Continuation: This code identifies the continuation of a HID payload that equaled or exceeded the negotiated MTU. DATC payloads can be found on the Interrupt or Control channels. The L2CAP Length field identifies the size of the DATC payload (including the DATC header). If the size of the report equals or exceeds the MTU, then the device will expect to receive additional DATC payloads. See Section 7.5.2.1 for more information on the handling of Large payloads.

DATC payloads follow GET_REPORT, SET_REPORT, or DATA payloads until a complete payload is accumulated. A “Short” (L2CAP Length field less than the MTU) DATC payload indicates the last transaction of a payload.

DATC transfers on the interrupt channel are mandatory for hosts which implement the full HID protocol. DATC transfers on the control channel are optional for hosts and devices.

The DATC Transaction Header has the following encoding:

Field	Size (Bytes)	Description
Request	1	Bits specifying characteristics of request. 7..4 Transaction Type 11 = DATC request 3..2 Reserved (0) 1..0 Report Type 0 = Other 1 = Input 2 = Output 3 = Feature
Report data payload	n	Additional report data for the device.

Table 16: DATA Parameter Definition

7.5 Transport

In the following discussion the term **Host** refers to the software stack on the host that supports BT-HID devices.

7.5.1 Payload

Figure 9 illustrates how payloads are translated between the data defined by the HID Specification [4] (top layer) and the baseband Bluetooth packets (bottom layer).

A BT-HID device adds a header (also called the BT-HID Header, Hdr) to the HID payload. Hdr is always 1 byte long; however, one or more bytes of non-data items may immediately follow it.

L2CAP adds another layer of encapsulation, defined by the Bluetooth Specification [5]. If the resulting payload is too large to fit in a single baseband packet, the L2CAP layer will segment the payload into smaller blocks before transmission or assemble segmented received packets into an L2CAP packet.

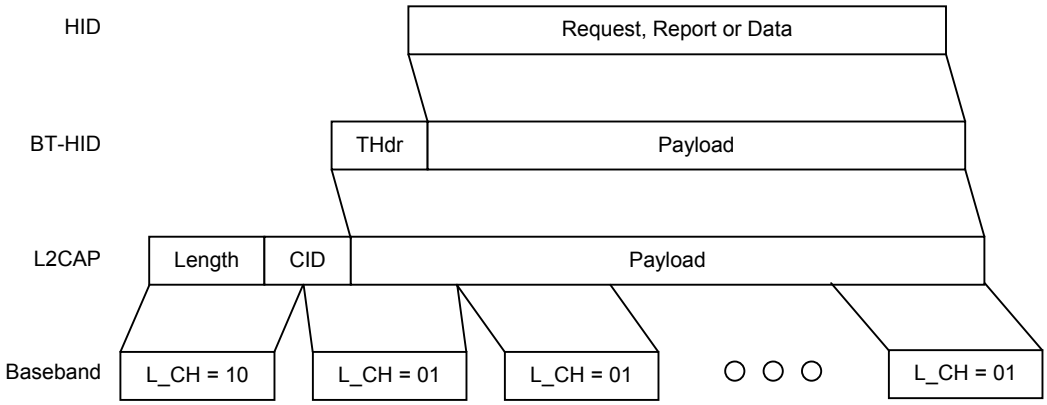


Figure 9: Segmentation in a HID Device

7.5.2 MTU

HID devices and Hosts shall support a Maximum Transmission Unit (MTU) equal or greater than the minimum value of 48 bytes[1], although it is recommended that HID hosts provide support for the default MTU of 672 bytes.

7.5.2.1 Large Payload Handling

(See Section 7.6.) A “Large” payload is any payload that is larger than or equal to maximum size, defined as (MTU – sizeof(Hdr)) bytes, where “sizeof(Hdr)” is the number of bytes in the HID protocol header byte for a given transfer.

To transmit a Large payload the sending side is expected to send an initial MTU size payload that identifies the Transaction Type (GET_REPORT, SET_REPORT, or DATA),

[1] Defined in the L2CAP section of the Bluetooth specification.

followed by DATC transactions. The DATC transactions will continue to be MTU size until the last DATC transaction, which is “Short” (less than MTU size). If the Large Payload size plus overhead equals a multiple of MTU-sized payloads, then an additional (“Short”) DATC transaction shall be sent to mark the completion of the Large payload. The minimum DATC transaction size is 1 byte (only contains the DATC header and no data).

The receiving side will assume that MTU-sized payloads are components of a Large payload and concatenate them. The terminating “short” payload will be concatenated to the previously received MTU-sized payloads and marks the end of the Large payload. The Large payload will then be passed to higher-level software layers for parsing.

For example, assume the MTU is 100 bytes, and the report size is 198 bytes. In this case, the first payload will have a SET_REPORT header (1-byte Request) followed by 99 bytes of report data. The second payload will have a 1-byte DATC header followed by 99 bytes of data. And the third payload will have a 1-byte DATC header followed by 0 bytes of data. The third payload is Short (smaller than the MTU) so it will indicate the end of the payload.

Ideally, an MTU is negotiated that is at least one byte larger than the largest Report generated by the device. This approach will save the overhead of sending Large payloads.

7.5.3 Report Data

It is recommended that data reports be designed to fit within the negotiated L2CAP negotiated MTU size.

Further optimization can be achieved by ensuring that data transmissions are restricted in size to be less than the Payload of a DM1 packet (i.e., 17 Bytes). This payload size minimizes latency and maximizes the number of BT-HID devices that can be supported, by ensuring that no more than one Frame (RX/TX baseband cycle) will be used to move a report.

7.6 Segmentation and Reassembly

If a report defined by a device is larger than the negotiated MTU, then the HID device shall implement segmentation and reassembly (SAR) of up to MTU-sized data payloads. HID hosts are required to support SAR for payloads greater than the negotiated MTU in size. The HID drivers need to implement SAR on top of L2CAP since HID packet report sizes can be as large as 64K bytes, and a small memory constrained host may need to handle HID packets larger than the L2CAP MTU. The key difference is that the HID segmentation and reassembly does not have the concept of an MTU and therefore does not require the host or device to have an amount of RAM equal to the largest possible packet size. The large packets must be assembled or decoded on the fly in this case. To determine whether SAR is necessary on the host side, the HID drivers must evaluate the MTU negotiated for the L2CAP channel, by the L2CA_ConfigReq service.

Hosts which support only Boot mode protocol are not required to support segmentation and reassembly in the HID protocol.

7.6.1 Segmentation

The HID SAR module shall divide data into segments equal to the L2CAP layer's negotiated MTU limit, except for the last segment in a set which shall be less than the L2CAP layer's negotiated MTU limit. If the last segment is equal to the MTU then an extra zero length segment shall be sent after the last payload data segment. This implies that the HID payload field has a limit of MTU – sizeof(Hdr) bytes. The first segment will contain a DATA header and all subsequent segments will contain a DATC header. The length defined by the DATA header will identify the total payload size not including any DATC headers.

7.6.2 Reassembly

If the L2CAP payload received by the HID SAR module equals the channel's MTU for that direction, for a SET_ request or DATA packet, it will append all subsequent DATC packet payloads. The last DATC packet to be concatenated will have length less than the MTU. When all segments are assembled, the payload will be passed to the upper software levels as a single large data object.

7.7 Flow Control

Flow control is not supported by the BT-HID profile. Most HID devices do not require explicit flow control support. If flow control is required, a vendor should use the L2CAP flow control mechanism when it is standardized in a future revision of the core Bluetooth specification

7.8 QoS

This section defines the L2CAP Configure Request parameter values that are recommended for a BT-HID device.

For the Control channel, the default Bluetooth “Best Effort” QoS is assumed. If an Interrupt channel is opened, then the device can define its QoS requirements with the InFlow parameter of the L2CA_ConfigReq primitive. Section 7.16 identifies the recommended values for the Interrupt channel QoS.

Parameter	Size (Octets)	Description
Service Type	1	This field indicates the level of service required.
Token Rate	4	The value of this field represents the maximum sustained rate at which data will be delivered across the baseband layer, in bytes per second. An application may send data at this rate continuously.
Token Bucket Size	4	The value of this field represents the amount of buffering the L2CAP layer will provide, in bytes. This value shall be set to n times the maximum report size that a device will send on a channel, where n is the number of reports to buffer. If the largest report is 20 bytes and the Token Bucket Size is set to 40 bytes, then a minimum of 2 reports will be buffered by the L2CAP layer.
Peak Bandwidth	4	The value of this field, expressed in bytes per second, limits the rate at which back-to-back packets can be loaded into the Token Bucket by applications.
Latency	4	The value of this field represents the maximum acceptable delay between presenting a payload to the L2CAP layer and its initial transmission over the air, expressed in microseconds.
Delay Variation	4	The value of this field is the difference, in microseconds, between the maximum and minimum possible delay that a packet will experience.

Table 17: QoS Parameters

See Section 7.16.3 for example mouse, keyboard, and force-feedback joystick QoS parameter settings.

7.9 Transfers

7.9.1 Control Channel

Most control channel transfers have two phases: a request by the host and a response by the device. **Only one host control channel request shall be outstanding at a time, i.e. a new transfer shall not begin before a transfer in progress is completed** This is also stipulated by the USB HID specification [4].The exception to this rule is that a device may spontaneously send a HID_CONTROL packet that specifies VIRTUAL_CABLE_UNPLUG event. See section 7.4.2 for more information.

7.9.1.1 Timeouts

At the L2CAP level, the only error that can occur is an LMP_supervision_timeout, which can occur on either the request by the host or the response by the device. In either case, the host will receive a response of either a baseband timeout message from its stack or a Handshake payload from the device. The timeout can be due to sustained interference, an out-of-range condition, or the responder has been turned off. If a timeout occurs, the connection to the device is lost. Upon reconnection, the host shall restore the *data state* of the device and reissue the timed out request (unless otherwise aborted by system software).

Default supervisory timeouts are typically 30 seconds. The HID profile recommends that the default supervision timeout be set to 5 seconds if the HIDSupervisionTimeout attribute (section 7.11.12) is not used. See section 7.11.12 of this document and section C:3.24 of the core spec [5] for more information. It is the responsibility of the Host to set the supervision timeout during the HID Service setup sequence.

7.9.1.2 Control Channel GET_

Control GET_ request, retrieves information from the device.

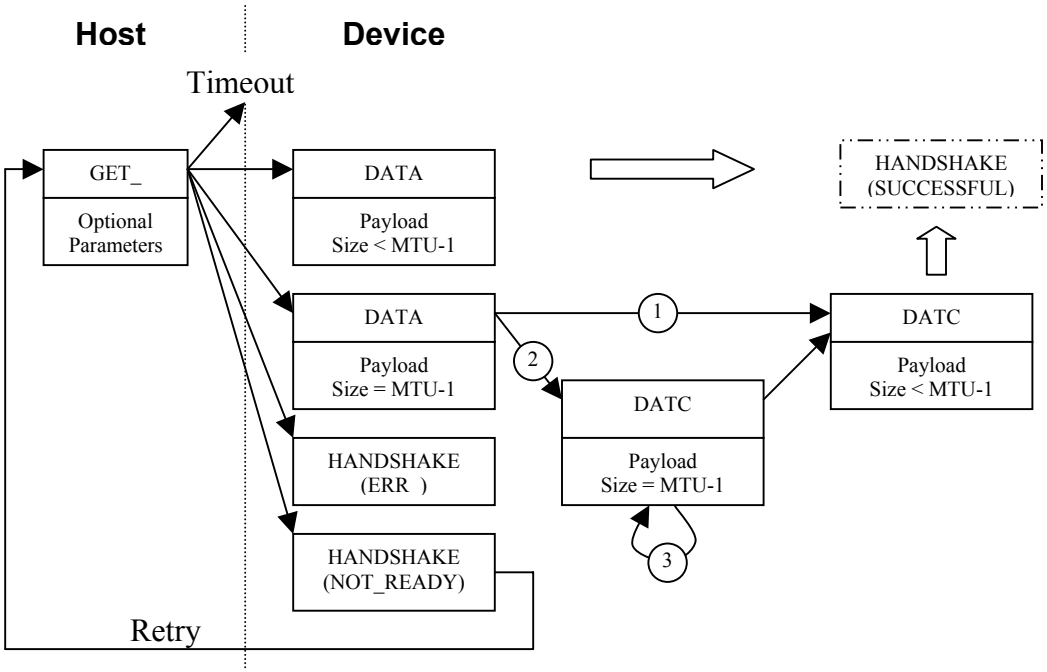


Figure 10: GET Flow chart

When a device receives a GET_ request, the device returns a DATA packet. If the GET_ payload exceeds the MTU then the device shall transmit additional DATC packets (1). Depending on the payload size, one (2) or more (3) DATC packets may be transmitted. The last DATC packet shall always be short (Payload Size +1 < MTU). A short DATA or DATC packet is interpreted as a SUCCESSFUL HANDSHAKE packet for the GET_ request (a physical HANDSHAKE packet is *not* returned by the device).

The device returns an ERR_ HANDSHAKE packet if a problem is detected or a NOT_READY HANDSHAKE packet if there is no data available. The host may retry the GET_ request if a NOT_READY HANDSHAKE packet is received. A timeout shall occur if the device did not receive the GET_ packet or the host does not receive a short DATA or short DATC packet.

Note that GET_ timeouts may occur if the baseband is unable to deliver or receive a packet within the Supervisory Timeout period.

7.9.1.3 Control Channel SET_

A Control SET_ request sends information to the device.

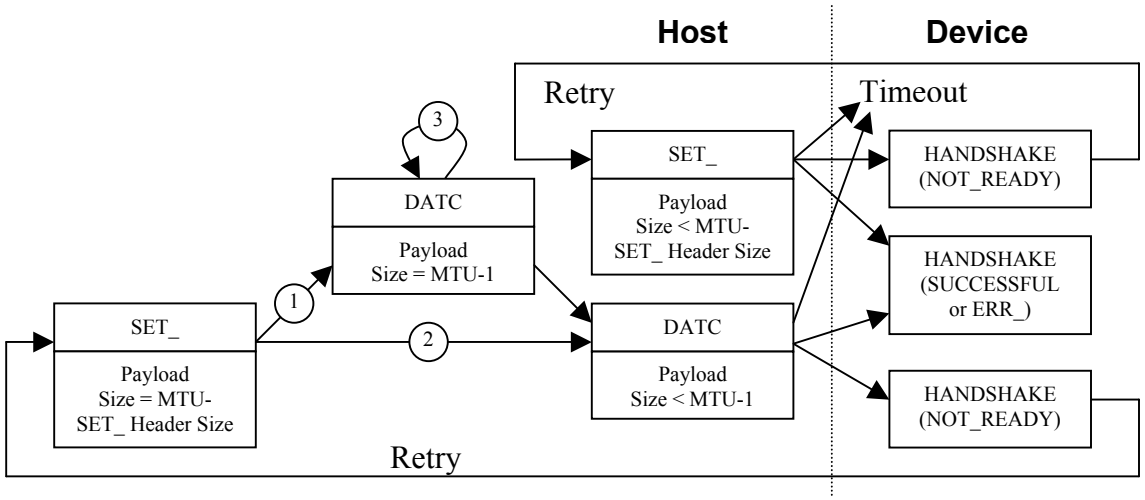


Figure 11: SET_ Flow Chart

If the SET_ payload exceeds the MTU then the host shall transmit additional DATC payloads (1). Depending on the payload size, one (2) or more (3) DATC packets may be transmitted. The last DATC packet shall always be short (Payload Size +1 < MTU).

A short SET_ or DATC packet is interpreted by the device as the completion of a SET_ request. The device will return as a SUCCESSFUL HANDSHAKE packet if no errors were detected.

The device returns an ERR_ HANDSHAKE packet if a problem was detected or a NOT_READY HANDSHAKE packet if the device was not ready to accept data. The host may retry the SET_ request if a NOT_READY HANDSHAKE packet is received. A timeout shall occur if the device did not receive the SET_ packet the host does not receive the HANDSHAKE packet.

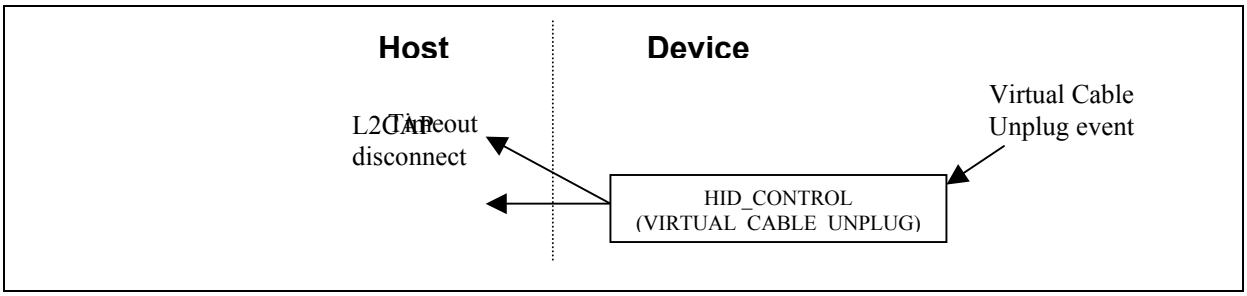
Note that SET_ timeouts may occur if the baseband is unable to deliver or receive a packet within the Supervisory Timeout period.

7.9.1.4 Virtual Cable Unplug

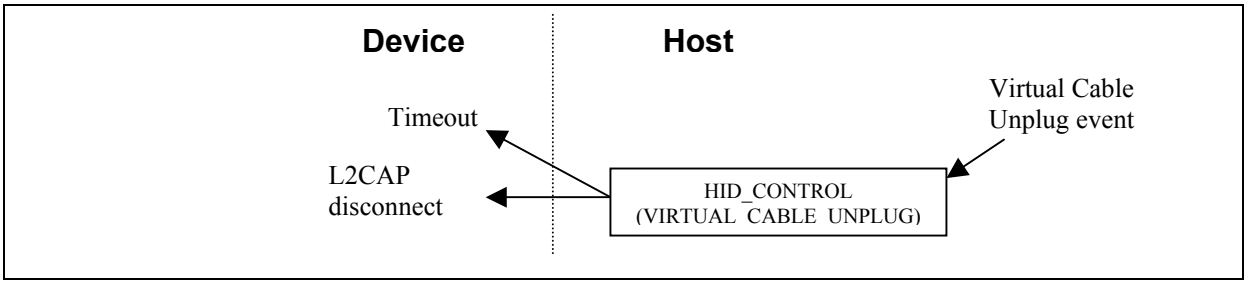
When a host or device determines that it needs to unplug the Virtual Cable (VC), it sends a HID_CONTROL packet with a parameter of VIRTUAL_CABLE_UNPLUG to the other side. The other side acknowledges this packet by sending back L2CAP_Disconnect_Req packets for the Control and Interrupt channels. See section 7.4.2 for more information.

It is recommended that if VC unplug is received by a HID while in PIN code entry mode, the device cancels out of PIN entry mode immediately. In the event PIN entry mode is canceled on the host side, sending VC unplug will then avoid the delay of having to wait for the device to time out (perhaps 30 seconds) before PIN entry is attempted again.

Device initiated Virtual Cable Unplug:



Host initiated Virtual Cable Unplug:



7.9.2 Interrupt Channel

The Interrupt Channel carries asynchronous, low-latency data. DATA and DATC packets are the only packet types transmitted or received on the Interrupt channel. See section 7.5.2.1 for more details on large payload handling.

7.9.2.1 Timeouts

At the L2CAP level, the only channel-related error that can occur is a timeout, which can occur on any transfer. The timeout can be due to sustained interference or an out-of-range condition. It is the responsibility of the Host to set the supervision timeout during the HID Service setup sequence. The recommended default timeout for HIDs is 5 seconds if the SDP attribute HIDSupervisionTimeout is not declared (see section 7.11.12).

If the host detects a timeout, then (assuming an out-of-range condition) the application will be informed. It can determine whether or not to retransmit the DATA messages or to throw them away.

7.9.2.2 Interrupt IN

To minimize latency, the host always accepts interrupt data. Interrupt IN data is a DATA payload from the device on the Interrupt channel.

Interrupt IN data is dumped into a circular queue in the driver. The size of the circular queue depends on system latencies. The selected queue size shall ensure that overflows do not occur during normal operation. If an application does not read the interrupt data in a timely manner, then new interrupt data overwrites old data.

The device can post Interrupt channel DATA payloads to the host at any rate. However, the rate shall never exceed one payload per QoS Latency period.

If the host receives a corrupt input report on the Interrupt channel, the report shall be ignored.

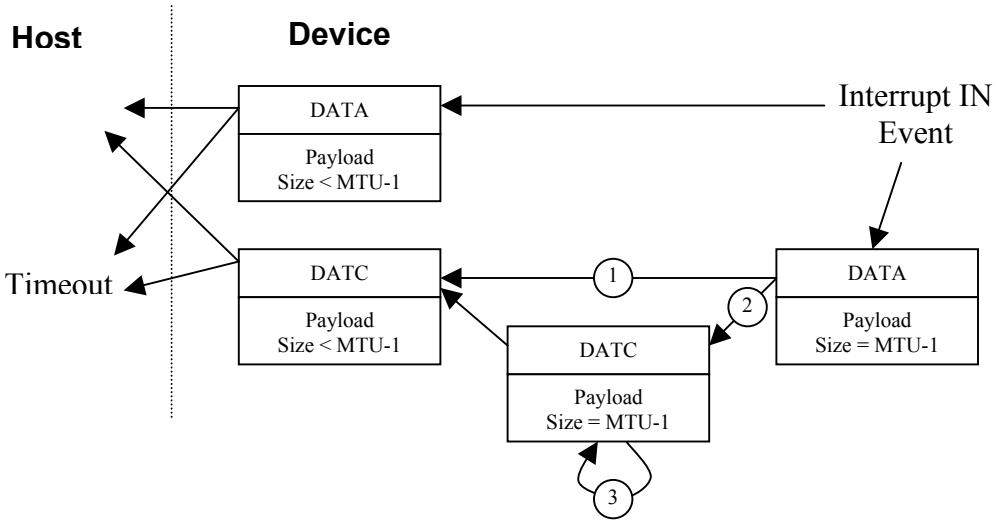


Figure 12: Interrupt IN Flow Chart

If the Interrupt IN payload exceeds the MTU then the device shall transmit additional DATC payloads (1). Depending on the payload size, one (1) or more (2,3) DATC packets may be transmitted. The last DATC packet shall always be short (Payload Size +1 < MTU).

A short DATA or DATC packet is interpreted by the host as the completion of an Interrupt IN payload.

Note that Interrupt IN timeouts may occur if the baseband is unable to receive a packet within the Supervisory Timeout period.

7.9.2.3 Interrupt OUT

To minimize latency, the device always accepts interrupt OUT data. Interrupt OUT data is a DATA payload from the host on the Interrupt channel.

The host can post Interrupt channel DATA payloads to the device at any time. The rate will never exceed one payload per QoS Latency period.

If the device cannot receive the Interrupt OUT data at the maximum data rate then new data may overwrite old data; this is an implementation decision. The QoS Latency can be used to throttle the Interrupt OUT Data to prevent this situation.

If a corrupt output report is received by the device on the Interrupt channel, the device shall ignore the report.

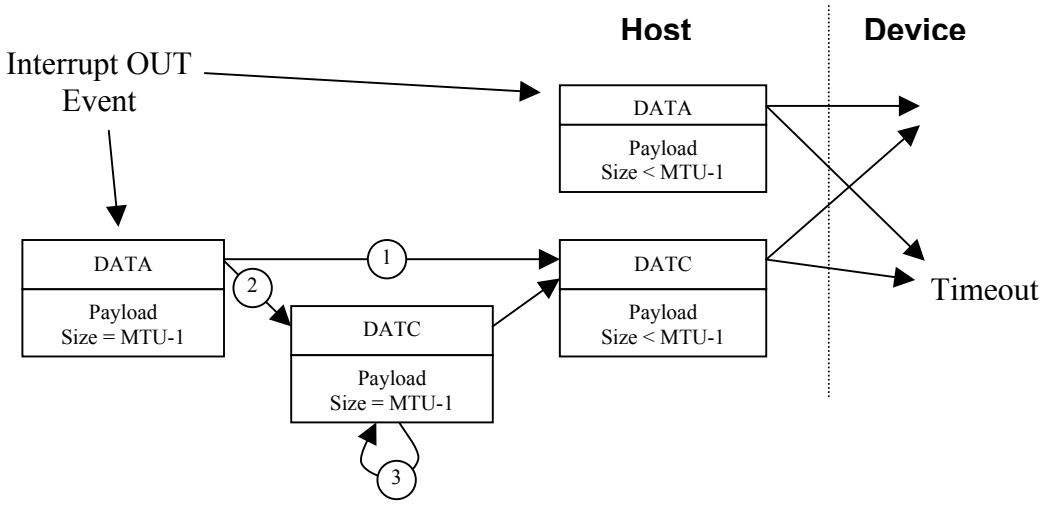


Figure 13: Interrupt OUT Flow Chart

If the Interrupt OUT payload exceeds the MTU then the host shall transmit additional DATC payloads (1). Depending on the payload size, one (1) or more (2,3) DATC packets may be transmitted. The last DATC packet shall always be short (Payload Size +1 < MTU).

A short DATA or DATC packet is interpreted by the device as the completion of an Interrupt OUT payload.

Note that Interrupt OUT timeouts may occur if the baseband is unable to transmit a packet within the Supervisory Timeout period.

7.10 HID Service Class Definitions

A HID device returns the following list of attributes in an SDP_ServiceAttributeResponse:

Note: In the list below: 'Normal' text represents Attribute ID/Value pairs where the Data Element Type of the attribute is not a sequence. **Bold** text indicates a Data Element Sequence. Throughout this section names are assigned to sequences and elements within sequences for ease of reference. *Italic* text represents names assigned to elements of sequences. ***Bold Italic*** text indicates names assigned to Data Element Sequences that are contained in another sequence. ***HIDDescriptor***, *ClassDescriptorType*, and *ClassDescriptorData* are examples of assigned names.

HIDProfileVersion
HIDDeviceReleaseNumber
HIDParserVersion
HIDDeviceSubclass
HIDCountryCode
HIDVirtualCable
HIDReconnectInitiate
HIDSDPDisable
HIDBatteryPower
HIDRemoteWake
HIDSupervisionTimeout
HIDNormallyConnectable
HIDBootDevice

HIDDescriptorList
HIDLANGIDBaseList

The HIDDescriptorList and HIDLANGIDBaseList Data Element Sequences contain one or more Data Element Sequences.

The following attributes display the hierarchy and position of Sequences and Attributes, in the Sequences defined above. Groups of attributes are collected into "named" sequences so that sequences and attributes can be added in later releases without affecting a HID SDP parser's ability to find them. *Any unrecognized sequences or attributes returned by a HID shall be ignored.*

HIDDescriptorList
    HIDDescriptor
    …
    HIDDescriptor

HIDDescriptor
ClassDescriptorType
ClassDescriptorData

HIDLANGIDBaseList
HIDLANGIDBase
…
HIDLANGIDBase

HIDLANGIDBase
LANGID
LanguageBase

To fully support existing HID parsers, a device shall also implement attributes defined in the Bluetooth Device Identification [13] white paper. In particular, the Bluetooth Device Identification profile defines the VendorID, Version, and ProductID attributes. These attributes are required for proper operation with legacy HID parsers.

Attribute	ID	Type & Size[1]	Required	Section
HIDBatteryPower	0x0209	Bool8	O	7.11.9
HIDBootDevice	0x020E	Bool8	M	7.11.11
HIDCountryCode	0x0203	uint8	M	7.11.3
HIDDescriptorList	0x0206	Sequence	M	7.11.6
HIDDeviceReleaseNumber	0x0200	uint16	O	7.11.1
HIDDeviceSubclass	0x0202	uint8	M	7.11.2
HIDLANGIDBaseList	0x0207	Sequence	M	7.11.7
HIDNormallyConnectable	0x020D	Bool 8	O	7.11.13
HIDParserVersion	0x020B	Uint16	M	7.11.14
HIDProfileVersion	0x0201	uint16	M	0
HIDReconnectInitiate	0x0205	Bool8	M	7.11.5
HIDRemoteWake	0x020A	Bool8	O	7.11.10
HIDSDPDisable	0x0208	Bool8	O	7.11.8
HIDSupervisionTimeout	0x020C	Uint16	O	7.11.12
HIDVirtualCable	0x0204	Bool8	M	7.11.4

[1] Notation: uint = Unsigned Integer, 8 = 8-bit, 16 = 16-bit, Bool = Boolean, Array = array of specified data type.

Table 18: SDP Attribute Summary (Alphabetical Order)

Attribute	ID	Type & Size[1]	Required	Section
HIDDeviceReleaseNumber	0x0200	uint16	O	7.11.1
HIDParserVersion	0x0201	uint16	M	0
HIDDeviceSubclass	0x0202	uint8	M	7.11.2
HIDCountryCode	0x0203	uint8	M	7.11.3
HIDVirtualCable	0x0204	Bool 8	M	7.11.4
HIDReconnectInitiate	0x0205	Bool 8	M	7.11.5
HIDDescriptorList	0x0206	Sequence	M	7.11.6
HIDLANGIDBaseList	0x0207	Sequence	M	7.11.7
HIDSDPDisable	0x0208	Bool 8	O	7.11.8
HIDBatteryPower	0x0209	Bool 8	O	7.11.9
HIDRemoteWake	0x020A	Bool 8	O	7.11.10
HIDProfileVersion	0x020B	Uint16	M	7.11.14
HIDSupervisionTimeout	0x020C	Uint16	O	7.11.12
HIDNormallyConnectable	0x020D	Bool 8	O	7.11.13
HIDBootDevice	0x020E	Bool 8	M	7.11.11
	0x020F – 0x03FF	Reserved HID Attributes		
	0x0400 – 0xFFFF	Available for HID Language Strings		

[1] Notation: uint = Unsigned Integer, 8 = 8-bit, 16 = 16-bit, Bool = Boolean.

Table 19: SDP Attribute Summary (Numeric Order)

7.11 Attributes

7.11.1 HIDDeviceReleaseNumber

Attribute Name	Attribute ID	Attribute Value Type
HIDDeviceReleaseNumber	0x0200	16-bit unsigned integer

Description

A numeric expression identifying the device release number in Binary-Coded Decimal. This is a vendor-assigned field, which defines the version of the product identified by the Bluetooth Device Identification [13] VendorID and ProductID attributes. This attribute is intended to differentiate between versions of products with identical VendorIDs and ProductIDs. The value of the field is 0xJJMN for version JJ.M.N (JJ – major version number, M – minor version number, N – sub-minor version number); e.g., version 2.1.3 is represented with value 0x0213 and version 2.0.0 is represented with a value of 0x0200. When upward-compatible changes are made to the device, the minor version number will be incremented. If incompatible changes are made to the device, the major version number will be incremented. NOTE: This attribute should not be used for new designs as it is redundant with the Device Identification Rev. 1.0 “Version Attribute”. HIDParserVersion

Attribute Name	Attribute ID	Attribute Value Type
HIDParserVersion	0x0201	16-bit unsigned integer

Description

Each version of a profile is assigned a 16-bit unsigned integer version number of the base HID Specification [4] that the device was designed to. The value of the field is 0xJJMN for version JJ.M.N (JJ – major version number, M – minor version number, N – sub-minor version number); e.g., version 2.1.3 is represented with value 0x0213 and version 2.0.0 is represented with a value of 0x0200.

7.11.2 HIDDeviceSubclass

Attribute Name	Attribute ID	Attribute Value Type
HIDDeviceSubclass	0x0202	8-bit unsigned integer

Description

The HIDDeviceSubclass attribute is an 8-bit integer, which identifies the type of device (keyboard, mouse, joystick, gamepad, remote control, sensing device, etc.). Keyboards and mice are required to support boot mode operation. In boot mode, a device presents a fixed report, thus negating the requirement for a HID parser.

The Attribute value is identical to the low-order 8 bits of the Class of Device/Service (CoD) field in the FHS packet, where bits 7-2 contain the 6 bit Minor Device Class value (defined in Section 1.2 of the Bluetooth Assigned Numbers document [8]) and bits 1-0 are set to zero. See Section 7.2.1 for more information about boot devices.

HIDDeviceSubclass can differ from the Class of Device field of an FHS packet if the device is a composite device that implements multiple Bluetooth profiles.

7.11.3 HIDCountryCode

Attribute Name	Attribute ID	Attribute Value Type
HIDCountryCode	0x0203	8-bit unsigned integer

Description

The HIDCountryCode attribute is an 8-bit integer, which identifies which country the hardware is localized for. Most hardware is not localized and thus this value would be zero (0). However, keyboards may use the field to indicate the language of the key caps. Devices are not required to place a value other than zero in this field, but some operating environments may require this information. The valid country codes are listed in the HID Specification [4].

7.11.4 HIDVirtualCable

Attribute Name	Attribute ID	Attribute Value Type
HIDVirtualCable	0x0204	8-bit Boolean

Description

The HIDVirtualCable attribute is a boolean value, which indicates whether the device supports virtual connections as described in Section Virtual Cables and Connection Re-Establishment. Devices that have this attribute True indicate that the device supports 1:1 bonding with a host, and the device expects to automatically reconnect if the connection is dropped for any unknown reason.

7.11.5 HIDReconnectInitiate

Attribute Name	Attribute ID	Attribute Value Type
HIDReconnectInitiate	0x0205	8-bit Boolean

Description

The HIDReconnectInitiate attribute is a boolean value, which indicates whether the device initiates the reconnection process or expects the host to. Devices that have this attribute True shall enter Page mode to automatically reconnect to the host if the connection is dropped for any unknown reason. The expectation of the device is that the host is listening for the device reconnection in Page Scan mode. Devices that have this attribute False shall enter Page Scan mode to reconnect to the host if the connection is dropped for any unknown reason. See Section 6.4 Virtual Cables and Connection Re-Establishment for more information.

A device may not necessarily try to reconnect as soon as it detects the loss of a connection. If there is no activity on the HID, then the device may wait until it has data to deliver before initiating a reconnect. This characteristic is referred to as “data driven reconnection”. Only devices that assert the HIDReconnectInitiate attribute (True) can support data-driven reconnections.

Devices which wish to initiate reconnection and also accept pages from the host shall set SDP attribute HIDNormallyConnectable to True.

7.11.6 HIDDescriptorList

Attribute Name	Attribute ID	Attribute Value Type
HIDDescriptorList	0x0206	Data element sequence

Description

The HIDDescriptorList Data Element Sequence performs the function of the HID Descriptor that is defined in Section 6.2 of the HID Specification [4]. The HIDDescriptorList identifies the descriptors associated with the device. Currently there are two types defined: Report and Physical. The other pertinent fields supplied by the HID Descriptor specification, release level and country code, are handled by the HIDParserVersion and HIDCountryCode attributes, respectively.

The HIDDescriptorList is a Data Element Sequence that consists of one or more HIDDescriptors. A HIDDescriptor is a data element sequence containing, minimally, a pair of elements. For compatibility with future versions of the HID profile, addition elements found in a HIDDescriptor shall be ignored.

The first element of the HIDDescriptor, called the ClassDescriptorType, identifies the type of the HID class descriptor found in the second element. The ClassDescriptorType element is identical to the bDescriptorType part of the USB HID descriptor. See Section 7.1 “Standard Requests” of the HID Specification [4] for a table of Class Descriptor Type constants. The USB HID Specification limits the range of values for Class Descriptor Types to values between 0x21 to 0x2F. For BT-HID devices the values outside this range (0x00-0x20 and 0x30-0xFF) are reserved. Note that in Bluetooth implementations, the HID class descriptor type (0x21) shall not be defined. The HID class descriptor information is supplied by the HIDDescriptorList, HIDParserVersion, and HIDCountryCode attributes. The ClassDescriptorType element contains a single byte, unsigned value (Data Element Type = Unsigned Integer (1), Data Element Size = 1 byte (0)).

Value	Descriptor Type
0-0x21	Reserved
0x22	Report
0x23	Physical Descriptor
0x24-0xFF	Reserved

Table 20: Descriptor Type Codes

The second element, called ClassDescriptorData, is a byte array that contains the descriptor identified by the first element. Descriptors are 8-bit unsigned arrays (Data Element Type = Text String (4), Data Element Size = array (5, 6, or 7)).

7.11.7 HIDLANGIDBaseList

Attribute Name	Attribute ID	Attribute Value Type
HIDLANGIDBaseList	0x0207	Data element sequence

Description

The HIDLANGIDBaseList attribute allows Bluetooth strings to be mapped to HID LANGID and string indices. It also allows the definition of HID-related strings that are not defined by Bluetooth.

In order to support human-readable attributes for multiple languages in a single service record, a base attribute ID is assigned for each of the HID languages used in a service record. The human-readable universal attributes are then defined with an attribute ID offset from each of these base values, rather than with an absolute attribute ID.

This HID profile-specific attribute is required for all BT-HID devices and works the same way as the LanguageBaseAttributeIDList attribute. The global Bluetooth attribute

LanguageBaseAttributeIDList is not required if a Bluetooth device only supports a single (primary) language.

The HIDLANGIDBaseList is a Data Element Sequence that consists of one or more HIDLANGIDBases. A HIDLANGIDBase is a data element sequence containing, minimally, two elements for each of the languages used in the service record: a language identifier (LANGID) and a base attribute ID. For compatibility with future versions of the HID profile, additional elements found in a HIDLANGIDBase shall be ignored.

The first element, called the HIDLANGID, contains an identifier representing the natural language ID. The language is encoded according to the “Universal Serial Bus Language Identifiers (LANGIDs)” Specification [9].

The second element, called the HIDLanguageBase, contains an attribute ID that serves as the base attribute ID for the natural language in the service record. Different service records within a server may use different base attribute ID values for the same language.

The value of the HIDLanguageBase element serves as the base attribute ID for the natural language in the service record. Different service records within a server may use different base attribute ID values for the same language.

To facilitate the retrieval of human-readable universal attributes in a principal language, the base attribute ID value for the primary language supported by a service record shall be 0x0100. If a LanguageBaseAttributeIDList attribute is contained in a service record, then the HIDLanguageBase attribute ID value contained in its first element pair shall also be 0x0100.

Note 1: The HID profile requires the definition of the LanguageBaseAttributeIDList attribute in the SDP.

Note 2: The MIB Enum value (second element) in the LanguageBaseAttributeIDList identifies the character set for the respective language. The Bluetooth Specification [5] recommends the use of UTF-8 (UCS Translation Format) [11] encoding, the HID Specification requires the use of UNICODE (UCS-2) [10] encoding. The host is required to translate the UTF-8 to UCS-2).

Note 3: The HID Specification [4] reserves string index 0 to provide a list of the LANGIDs supported by a device. The Bluetooth Specification [5] reserves the string at offset 0 for the Service Name, so the Host must reconstruct the HID LANGID string (0) from the information in the HIDLANGIDBaseList attribute.

7.11.8 HIDSDPDisable

Attribute Name	Attribute ID	Attribute Value Type
HIDSDPDisable	0x0208	8-bit Boolean

Description

The HIDSDPDisable attribute is a boolean value, which indicates whether connection to the SDP channel and Control or Interrupt channels are mutually exclusive. This feature supports devices that have minimal resources, and multiplex those resources between servicing the initialization (SDP) and runtime (Control and Interrupt) channels.

If the HIDSDPDisable attribute is true, the host shall issue an LMP_DisconnectReq primitive to the SDP channel of the device before attempting to open either the Control or the Interrupt channel. If after opening the Control and Interrupt channels the host decides that it wants to access SDP information on the device, then the host shall first close both the Control and Interrupt channels before opening the SDP channel.

If the HIDSDPDisable attribute is true, then a HID shall reject L2CA_ConnectReq commands to open the SDP channel, if the Control or Interrupt channel are open. It shall also reject L2CA_ConnectReq commands to open Control or Interrupt channels, if the SDP channel is open.

If the HIDSDPDisable attribute is false (or not declared), then the SDP, Control, and Interrupt channels can be open at the same time.

Vendors must assess possible conflicts associated with implementing this feature in multi-function devices.

7.11.9 HIDBatteryPower

Attribute Name	Attribute ID	Attribute Value Type
HIDBatteryPower	0x0209	8-bit Boolean

Description

The HIDBatteryPower attribute is a boolean value, which indicates whether the device is battery powered (and requires careful power management) or has some other source of power that requires minimal management. Devices that have this attribute True may modify their Sniff Period or generate LMP_park_req PDUs asking to be placed in park mode, when a sufficiently long period of inactivity has been detected.

If the HIDBatteryPower attribute is false (or not declared), then the device is provided with a continuous power source; i.e., plugged in to the wall.

7.11.10 HIDRemoteWake

Attribute Name	Attribute ID	Attribute Value Type
HIDRemoteWake	0x020A	8-bit Boolean

Description

The HIDRemoteWake attribute is a boolean value, which indicates whether the device considers itself remote wake up-capable. When a system enters a suspend (or standby)

state, this flag shall be used to determine whether the host includes this device in the set of devices that can wake it up. A mouse or keyboard are typical examples of Remote Wake up devices.

If the HIDRemoteWake attribute is false (or not declared), then the device does not consider itself remote wakeup-capable, and the system can exclude this device from the set of devices that can wake it up. If no devices declare the Remote Wake flag True, then the system may completely shut down the radio while suspended.

The host shall notify a wake up-capable device that it is entering a suspended state by issuing a Suspend Control Operation request (see Section 7.4.2). This allows the device to shutdown any circuitry not required to wake up the host.

When an event occurs that requires the device to wake up the system, the device shall automatically exit suspend mode and power up all circuitry. If, after a vendor-defined period, the device is unable to reconnect to the system it may re-enter suspend mode. If the host decides to exit suspend mode, then it shall send an Exit Suspend Control Operation request to the device.

There are several ways for a HID to be “suspended” and then wake up a system. Some require no special features for the device.

Sniff Mode

The device may be placed into Sniff mode with a long interval between sniff slots. While the device is in sniff mode, it shall use the master transmission to remain synchronized with the master. When an event occurs that requires the device to wake up the system, the device shall attempt to unsniff the link in the next available slot; i.e., when the user touches a button on the mouse to wake up the system.

Data Driven Reconnect

If the HIDReconnectInitiate attribute (see Section 7.11.5) is True, then the host may drop the connection to the device while it is suspended and periodically enter Page Scan mode to allow the device to reconnect due to a wake up event. The period between Page Scans depends on the startup latency requirements of the host. If the HIDReconnectInitiate attribute is False, then the Sniff Mode method discussed above should be used to wake up the host.

7.11.11 HIDBootDevice

Attribute Name	Attribute ID	Attribute Value Type
HIDBootDevice	0x020E	8-bit Boolean

HIDBootDevice is an 8-bit Boolean value that when True indicates whether the device supports boot protocol mode and by inference the Set_Protocol and Get_Protocol commands. It is a required attribute and is present in case future fixed format reports

are defined that cannot be represented in the Class of Device field in the Bluetooth FHS packet.

7.11.12 HIDSupervisionTimeout

Attribute Name	Attribute ID	Attribute Value Type
HIDSupervisionTimeout	0x020C	16-bit unsigned integer

Description

The HIDSupervisionTimeout is a 16-bit value which indicates the device vendor's recommended baseband Link Supervision Timeout value in slots. The host can use this value to override the default HID supervision timeout setting. This attribute is optional for the device and host. If it is not declared by the device, the host should apply the following rule for setting the supervision timeout values:

> Use the LMP_Supervision_Timeout PDU to set the timeout to a default value of 5 seconds or longer (>=8000 slots) for HIDs in Active mode and HIDs in SNIFF mode with sniff intervals of less than or equal to 1 second.

Since it is optional for hosts to utilize this SDP value, devices which wish to utilize long interval (>1 second) SNIFF mode must declare HIDSupervisionTimeout and then verify that the host has set the timeout to the declared value, before utilizing long interval SNIFF modes (which might result in a link supervision timeout). The command HCI_Read_Link_Supervision_Timeout can be used by the device to read the timeout value set by the host.

Dynamically changing the SDP HIDSupervisionTimeout value is not supported.

A device in Boot Protocol mode shall not set a sniff interval to a value greater than 1 second. See section 7.2.1 for more information on boot mode.

In the case of a compound device where multiple profiles share the same ACL connection, the manufacturer shall not create SDP records with conflicting supervision timeout values. In the case of a conflict, generally the host should use the shortest timeout.

7.11.13 HIDNormallyConnectable

Attribute Name	Attribute ID	Attribute Value Type
HIDNormallyConnectable	0x020D	8-bit Boolean

Description

HIDNormallyConnectable is an optional Boolean attribute that specifies whether a HID is normally in Page Scan mode (when no connection is active) or not. If the device declares this attribute and sets it to true, it shall remain connectable in order to respond

to a page from the host. If the device declares this attribute false, it may completely shut down its Bluetooth radio when there is no active connection. This attribute may be used with HIDReconnectInitiate for a device to achieve minimum power consumption when no active connection is present.

Keyboards designed for operation with a PC should always set this attribute to True since popular operating systems and BIOSes will be designed to locate a Bluetooth keyboard with no user intervention. This is necessary since no other means of user input may be available before the keyboard is attached.

7.11.14 HIDProfileVersion

Attribute Name	Attribute ID	Attribute Value Type
HIDProfileVersion	0x020B	16-bit unsigned integer

Description

Each device designed to this specification shall include a 16-bit unsigned integer version number of the Bluetooth HID Specification (this document) that the device was designed to. The value of the field is 0xJJMN for version JJ.M.N (JJ – major version number, M – minor version number, N – sub-minor version number); e.g., version 2.1.3 is represented with value 0x0213 and version 2.0.0 is represented with a value of 0x0200.

Item	Definition	Type[6]	Value	Status[7]	Attribute ID
Service Class ID List				**M**	**0x0001**
Service Class 0	HID	UUID	Note 1	M	
Protocol Descriptor List				**M**	**0x0004**
Protocol Descriptor #0		Data Element Sequence		M	
Protocol ID	L2CAP	UUID	L2CAP, Note 1	M	
Parameter #0	PSM	Uint16	HID_Control	M	
Protocol Descriptor #1		Data Element Sequence		M	
ProtocolID	HID	UUID	HID Protocol, Note 1	M	
AdditionalProtocol DescriptorLists				**M**	**0x000d**
ProtocolDescriptorList#0		Data Element Sequence		M	
ProtocolDescriptor#0		Data Element Sequence			
ProtocolID	L2CAP	UUID	L2CAP, Note 1	M	
Parameter#0	PSM	Uint16	HID_Interrupt	M	
ProtocolDescriptor#1		Data Element Sequence		M	
ProtocolID	HID	UUID	HID Protocol,	M	

Item	Definition	Type[6]	Value	Status[7]	Attribute ID
			Note 1		
Service Name	Displayable Text name	Data-Element/ String	'XYZ Wireless Device', Note 3	O	0x0100, Note 2
Service Description	Displayable Text name	Text String	'HID Widget', Note 3	O	0x0101, Note 2
Provider Name	Displayable Text name	Text String	'XYZ Company', Note 3	O	0x0102, Note 2
BluetoothProfile DescriptorList				**M**	**0x0009**
Profile 0		UUID	HID	M	
Parameter for Profile 0	Version	uint16	0x0100, Note 4	M	
LanguageBase AttributeIDList		**Data Element/ Sequence**	**Define Primary (and alternate) language**	**M**	**0x0006**
HIDDeviceRelease Number	**7.11.1**	**uint16**	**0x0100, Note 4**	**M**	**0x0200**
HIDParserVersion	**0**	**uint16**	**0x0111, Note 5**	**M**	**0x0201**
HIDDeviceSubclass	**7.11.2**	**uint8**	**See Bluetooth Defined Numbers Document [8]**	**M**	**0x0202**
HIDCountryCode	**7.11.3**	**uint8**	**See Section 6.2.1 of the HID Specification [4]**	**M**	**0x0203**
HIDVirtualCable	**7.11.4**	**Bool (8)**	**True/False**	**M**	**0x0204**
HIDReconnectInitiate	**7.11.5**	**Bool (8)**	**True/False**	**M**	**0x0205**
HIDDescriptorList	**7.11.6**	**Sequence**		**M**	**0x0206**
HIDLANGIDBaseList	**7.11.7**	**Sequence**		**M**	**0x0207**
HIDSDPDisable	7.11.8	Bool (8)	True/False	O	0x0208
HIDBatteryPower	7.11.9	Bool (8)	True/False	O	0x0209
HIDRemoteWake	7.11.10	Bool (8)	True/False	O	0x020A
HIDBootDevice	**7.11.11**	**Bool (8)**	**True/False**	**M**	**0x020E**
HIDSupervisionTimeout	7.11.12	uint16	See text	O	0x020C
HIDNormallyConnectable	7.11.13	Bool (8)	True/False	O	0x020D
HIDProfileVersion	**7.11.14**	**Uint16**	**0x0100, Note 4**	**M**	**0x020B**

Table 21: SDP Entry for HID Service

Bold = Mandatory attributes

Notes

1 Defined in Assigned Numbers document [8].

2 For national language support for all "displayable" text string attributes, an offset has to be added to the LanguageBaseAttributeIDList value for the selected language (see the SDP Specification, Section E:5.1.14 [5] for details).

3 The ServiceName attribute value suggested here is merely an example; a service-provider may define a more descriptive name for their device.

4 Indicating version 1.0.

5 Indicating version 1.11.

6 Notation: uint = Unsigned Integer, 8 = 8-bit, 16 = 16-bit, Bool = Boolean, Array = array of specified data type.

7 M = Mandatory; O = Optional

7.12 SDP Procedures

To retrieve the service records in support of this profile, the SDP client entity in the host device connects and interacts with the SDP server entity in the HID via the SDP and L2CAP procedures presented in Sections 5 and 6 of SDAP (see reference [6]). In accordance with SDAP (Service Discovery Application Profile), DevA plays the role of the host, while DevB plays the role of the Human Interface Device.

7.13 SDP Requirements

Bluetooth Human Interface Devices shall implement a minimal Bluetooth SDP server, with one exception. Declaration of the LanguageBaseAttributeIDList attribute is required in all HID implementations. Note that this must match the HIDLANGIDBaseList described in Section 7.11.7.

7.14 Example SDP Transactions

This is an example for a mouse with the following database description. The mouse has the following features:

- Provides strings for only one language, English
- Provides Service Name, Service Description, and Provider Name Strings
- Supports the Boot mode protocol
- Reports mouse coordinates every 10 ms.
- Supports Virtual Connections
- Will initiate a reconnection if a connection is lost
- Supports 3 buttons and 2 axes
- Battery powered
- Considers itself a “Remote Wakeup” device
- Generates a single 3-byte Input report

Attribute ID	Attribute Value	Description
ServiceRecordHandle 0x0000	0x0A 0x00010002	uint32 service record handle
ServiceClassIDList 0x0001	0x35 0x03	data element sequence, 3 bytes
	0x19 0x1124	uuid16 HID, Note 1
ProtocolDescriptorList 0x0004	0x35 0x0D	data element sequence, 10 bytes
Sequence Element 0	0x35 0x06 0x19 0x0100	data element seq. for L2CAP descriptor uuid16 L2CAP

Attribute ID	Attribute Value	Description
Element 1	0x09 0x0011	uint16 PSM for BT-HID Control channel
Sequence	0x35 0x03	data element seq. for HID descriptor
Element 0	0x19 0x0011	uuid16 HID, Note 1
LanguageBaseAttributeIDList 0x0006	0x35 0x09	data element sequence, 9 bytes
Element 0	0x09 0x656E	uint16 “en” (English)
Element 1	0x09 0x006A	uint16 UTF-8 encoding
Element 2	0x09 0x0100	uint16 PrimaryLanguageBaseID
BluetoothProfileDescriptorList 0x0009	0x35 0x08	data element sequence, 8 bytes
Sequence	0x35 0x06	data element sequence, 6 bytes
Element 0	0x19 0x0011	uuid16 HID, Note 1
Element 1	0x09 0x0100	uint16 version number
AdditionalProtocolDescriptorLists 0x000D	0x35 0x0C	Data element sequence, 12 bytes
Element 0	0x35 0x0A	data element sequence, 10 bytes
Sequence	0x35 0x03	data element seq. for L2CAP descriptor
Element 0	0x19 0x0100	uuid16 L2CAP
Element 1	0x09 0x0013	uint16 PSM for BT-HID Interrupt channel
Sequence	0x35 0x03	data element seq. for HID descriptor
Element 0	0x19 0x0011	uuid16 HID, Note 1
ServiceName 0x0100 + 0x0000	0x25 0x09	text string, 9 bytes
	“XYZ Mouse”	
ServiceDescription 0x0100 + 0x0001	0x25 0x11	text string, 17 bytes
	“Three button Mouse”	
ProviderName 0x0100 + 0x0002	0x25 0x0B	text string, 11 bytes
	“XYZ Company”	
HIDDeviceReleaseNumber 0x0200	0x09 0x0100	Uint16, Version of HID Device. See Section 7.11.1.
HIDParserVersion 0x0201	0x09 0x0111	Uint16, Version of core USB HID Specification [4] to which the device was designed. See Section 0.
HIDDeviceSubclass 0x0202	0x08 0x80	Uint8, Mouse that supports Boot mode. See Section 7.11.2.
HIDCountryCode 0x0203	0x08 0x21	Uint8, USA. See Section 7.11.3.
HIDVirtualCable 0x0204	0x28 0x01	Boolean8, Virtual Connection supported. See Section 6.4.
HIDReconnectInitiate 0x0205	0x28 0x01	Boolean8, Automatic reconnection supported. See Section 7.11.5.
HIDDescriptorList 0x0206	0x35 0x38	Data element sequence, 56 bytes. See Section 7.11.6.
HID Class Descriptor	0x35 0x36	data element sequence, 54 bytes
Element 0	0x08 0x22	Type = Report Descriptor
Element 1	0x25 0x32	Text String, 50 Byte Report Descriptor
	0x05 0x01 0x09 0x02 0xA1 0x01 0x09 0x01 0xA1 0x00 0x05 0x01 0x09 0x30	USAGE_PAGE (Generic Desktop) USAGE (Mouse) COLLECTION (Application) USAGE (Pointer) COLLECTION (Physical) USAGE_PAGE (Generic Desktop) USAGE (X)

Attribute ID	Attribute Value	Description
	0x09 0x31 0x15 0x81 0x25 0x7F 0x75 0x08 0x95 0x02 0x81 0x06 0xC0 0x05 0x09 0x19 0x01 0x29 0x03 0x15 0x00 0x25 0x01 0x95 0x03 0x75 0x01 0x81 0x02 0x95 0x01 0x75 0x05 0x81 0x03 0xC0	USAGE (Y) LOGICAL_MINIMUM (-127) LOGICAL_MAXIMUM (127) REPORT_SIZE (8) REPORT_COUNT (2) INPUT (Data,Var,Rel) END_COLLECTION USAGE_PAGE (Button) USAGE_MINIMUM (Button 1) USAGE_MAXIMUM (Button 3) LOGICAL_MINIMUM (0) LOGICAL_MAXIMUM (1) REPORT_COUNT (3) REPORT_SIZE (1) INPUT (Data,Var,Abs) REPORT_COUNT (1) REPORT_SIZE (5) INPUT (Cnst,Var,Abs) END_COLLECTION
HIDLANGIDBaseList 0x0207	0x35 0x08	data element sequence, 8 bytes. HID LANGID to Bluetooth language mapping. See Section 7.11.7.
HID LANGID Base	0x35 0x06	data element sequence, 6 bytes
Element 0	0x09 0x0409	Language = English (United States)
Element 1	0x09 0x0100	Bluetooth String Offset
HIDSDPDisable 0x0208	0x28 0x00	Boolean8. The mouse supports control and interrupt, and SDP channels open simultaneously.
HIDBatteryPower 0x0209	0x28 0x01	Boolean8. The mouse is battery powered. See Section 7.11.9.
HIDRemoteWake 0x020A	0x28 0x01	Boolean8. The mouse considers itself remote wake up-capable. See Section 7.11.10.
HIDBootDevice 0x20E	0x28 0x01	Boolean8. The mouse declares itself a bood device. See Section 7.11.11.
HIDProfileVersion 0x20B	0x09 0x0100	Uint16, Version of this Profile Specification to which the device was designed. See Section 7.11.14.

Table 22: Example Service Record for HID Mouse

1. Defined in the Assigned Numbers document [8].

Note: The HID report descriptor defined in figure 23 is an example "Report" mode version of a mouse report descriptor, and it is intentionally not the same as the mouse "Boot" descriptor. The "Boot" mode mouse and keyboard report descriptors are defined in appendix B of the USB HID specification [4].

7.14.1 Example 1: ServiceSearchRequest

The first example is that of an SDP client searching for a device that supports the HID profile. In the example, the SDP server indicates that it has a HID service available. The transaction illustrates:

1 SDP client to SDP server: SDP_ServiceSearchRequest, specifying the only element of the ServiceSearchPattern. The TransactionID is illustrated as tttt.

2 SDP server to SDP client: SDP_ServiceSearchResponse, returning handles the HID service, represented as qqqqqqqq. The Transaction ID is the same value as supplied by the SDP client in the corresponding request ().

```
/* Sent from SDP Client to SDP server */
SDP_ServiceSearchRequest[15]
{
  PDUID[1]
  {
    0x02
  }
  TransactionID[2]
  {
    0xtttt
  }
  ParameterLength[2]
  {
    0x0008
  }
  ServiceSearchPattern[5]
  {
    DataElementSequence[5]
    {
      0b00110 0b101 0x03
      UUID[3]
      {
        /* HIDClassID */
        0b00011 0b001 0x1124
      }
    }
  }
  MaximumServiceRecordCount[2]
  {
    0x0003
  }
  ContinuationState[1]
  {
    /* no continuation state */
    0x00
  }
}

/* Sent from SDP server to SDP client */
SDP_ServiceSearchResponse[18]
{
  PDUID[1]
  {
    0x03
  }
  TransactionID[2]
  {
    0xtttt
  }
  ParameterLength[2]
  {
    0x0009
  }
```

```
  TotalServiceRecordCount[2]
  {
    0x0001
  }
  CurrentServiceRecordCount[2]
  {
    0x0001
  }
  ServiceRecordHandleList[4]
  {
    /* HID service 1 handle */
    0xqqqqqqqq
  }
  ContinuationState[1]
  {
    /* no continuation state */
    0x00
  }
}
}
```

7.14.2 Example 2: Service Attribute Transaction

The second example continues the first example. In Example 1, the SDP client obtained handles a HID service. In Example 2, the client uses one of the returned service handles to obtain the ProtocolDescriptorList attribute for the HID service. The transaction illustrates:

1 SDP client to SDP server: SDP_ServiceAttributeRequest, presenting the previously obtained service handle (the one denoted as qqqqqqqq) and specifying the ProtocolDescriptorList attribute ID (AttributeID 0x0004) as the only attribute requested (other attributes could be retrieved in the same transaction if desired). The TransactionID is illustrated as uuuu to distinguish it from the TransactionID of Example 1.

2 SDP server to SDP client: SDP_ServiceAttributeResponse, returning the ProtocolDescriptorList for the HID service. This protocol stack is assumed to be ((L2CAP,PSM), (HID)). The ProtocolDescriptorList is a data element sequence in which each element is, in turn, a data element sequence whose first element is a UUID representing the protocol, and whose subsequent elements are protocol-specific parameters. In this example, one such parameter is included for the L2CAP protocol. The Transaction ID is the same value as supplied by the SDP client in the corresponding request (uuuu). The Attributes returned are illustrated as a data element sequence where the protocol descriptors are 32-bit UUIDs.

```
/* Sent from SDP Client to SDP server */
SDP_ServiceAttributeRequest[17]
{
  PDUID[1]
  {
    0x04
  }
  TransactionID[2]
  {
    0xuuuu
  }
  ParameterLength[2]
  {
    0x000C
  }
```

```
  ServiceRecordHandle[4]
  {
    0xqqqqqqqq
  }
  MaximumAttributeByteCount[2]
  {
    0x0080
  }
  AttributeIDList[5]
  {
    DataElementSequence[5]
    {
      0b00110 0b101 0x03
      AttributeID[3]
      {
        0b00001 0b001 0x0004
      }
    }
  }
  ContinuationState[1]
  {
    /* no continuation state */
    0x00
  }
}

/* Sent from SDP server to SDP client */
SDP_ServiceAttributeResponse[38]
{
  PDUID[1]
  {
    0x05
  }
  TransactionID[2]
  {
    0xuuuu
  }
  ParameterLength[2]
  {
    0x0017
  }
  AttributeListByteCount[2]
  {
    0x0014
  }
  AttributeList[20]
  {
    DataElementSequence[20]
    {
      0b00110 0b101 0x12

      Attribute[18]
      {
        AttributeID[3]
        {
          0b00001 0b001 0x0004
        }
        AttributeValue[15]
        {
          /* ProtocolDescriptorList */
          DataElementSequence[15]
          {
            0b00110 0b101 0x0D
            /* L2CAP protocol descriptor */
            DataElementSequence[8]
            {
              0b00110 0b101 0x06
              UUID[3]
              {
                /* L2CAP Protocol UUID */
                0b00011 0b001 0x0100
```

```
              }
              PSM[3]
              {
                /* PSM used for BT-HID Control channel */
                0b00001 0b001 0x0011
              }
            }
            /* HID protocol descriptor */
            DataElementSequence[5]
            {
              0b00110 0b101 0x03
              UUID[3]
              {
                /* HID Protocol UUID */
                0b00011 0b001 0x0011
              }
            }
          }
        }
      }
    }
  }
  ContinuationState[1]
  {
    /* no continuation state */
    0x00
  }
}
```

7.14.3 Example 3: ServiceSearchAttributeTransaction

The third example is a form of service browsing, although it is not generic browsing in that it does not make use of SDP browse groups. Instead, an SDP client is searching for specific HID attributes.

The transaction illustrates:

1 SDP client to SDP server: SDP_ServiceSearchAttributeRequest, specifying the generic HIDServiceClassID as the only element of the ServiceSearchPattern. The requested attributes are the ServiceRecordHandle (attribute ID 0x0000), ServiceClassIDList (attribute ID 0x0001), LanguageBaseAttributeIDList (attribute ID 0x0006), ServiceName (attribute ID 0x0100), ServiceDescription (attribute ID 0x0101), and ProviderName (attributeID 0x0102) attributes; as well as several service-specific attributes (AttributeID 0x0200 through 0x020C). Since the first two attribute IDs (0x0000 and 0x0001), the three string attribute IDs (0x0100, 0x0101, and 0x0102) and the service-specific attributes are consecutive, they are specified as attribute ranges. The TransactionID is illustrated as vvvv to distinguish it from the TransactionIDs of the other Examples.

Note that values in the service record's primary language are requested for the text attributes (ServiceName, ServiceDescription and ProviderName) so that absolute attribute IDs may be used, rather than adding offsets to a base obtained from the LanguageBaseAttributeIDList attribute.

Note that values in the service record's primary language are requested for the text attributes (ServiceName, ServiceDescription and ProviderName) so that absolute

attribute IDs may be used, rather than adding offsets to a base obtained from the LanguageBaseAttributeIDList attribute.

2 SDP server to SDP client: SDP_ServiceSearchAttributeResponse, returning the specified attributes for the HID service. Each of the attributes contains illustrative data in the example. The attributes are returned as a data element sequence, where each element is in turn a data element sequence representing a list of attributes. Within each attribute list, the ServiceClassIDList is a data element sequence while the remaining attributes are single data elements. The Transaction ID is the same value as supplied by the SDP client in the corresponding request (0xvvvv). Since the entire result cannot be returned in a single response, a non-null continuation state is returned in this first response.

```
/* Part 1 -- Sent from SDP Client to SDP server */
SdpSDP_ServiceSearchAttributeRequest[33]
{
  PDUID[1]
  {
    0x06
  }
  TransactionID[2]
  {
    0xvvvv
  }
  ParameterLength[2]
  {
    0x001C
  }
  ServiceSearchPattern[5]
  {
    DataElementSequence[5]
    {
      0b00110 0b101 0x03
      UUID[3]
      {
        /* HIDServiceClassID */
        0b00011 0b010 0x1124
      }
    }
  }
  MaximumAttributeByteCount[2]
  {
    0x0190
  }
  AttributeIDList[20]
  {
    DataElementSequence[20]
    {
      0b00110 0b101 0x12
      AttributeIDRange[5]
      {
        0b00001 0b010 0x00000001
      }
      AttributeID[3]
      {
        0b00001 0b001 0x0006
      }
      AttributeIDRange[5]
      {
        0b00001 0b010 0x01000102
      }
      AttributeIDRange[5]
      {
```

```
        0b00001 0b010 0x0200020C
      }
    }
  }
  ContinuationState[1]
  {
    /* no continuation state */
    0x00
  }
}

/* Part 2 -- Sent from SDP server to SDP client */
SdpSDP_ServiceSearchAttributeResponse[384]
{
  PDUID[1]
  {
    0x07
  }
  TransactionID[2]
  {
    0xvvvv
  }
  ParameterLength[2]
  {
    0x00E1
  }
  AttributeListByteCount[2]
  {
    0x00DE
  }
  AttributeLists[222]
  {
    DataElementSequence[222]
    {
      0b00110 0b101 0xDC
      DataElementSequence[220]
      {
        0b00110 0b101 0xDA

        Attribute[8]
        {
          AttributeID[3]
          {
            0b00001 0b001 0x0000
          }
          AttributeValue[5]
          {
            /* service record handle */
            0b00001 0b010 0xhhhhhhhh
          }
        }

        Attribute[8]
        {
          AttributeID[3]
          {
            0b00001 0b001 0x0001
          }
          AttributeValue[5]
          {
            DataElementSequence[5]
            {
              0b00110 0b101 0x03
              UUID[3]
              {
                /* HIDServiceClassID */
                0b00011 0b010 0x1124
              }
            }
          }
        }
```

```
Attribute[14]
{
  AttributeID[3]
  {
    /* LanguageBaseAttributeIDList */
    0b00001 0b001 0x0006
  }
  AttributeValue[11]
  {
    DataElementSequence[11]
    {
      0b00110 0b101 0x09
      Element0[3]
      {
        /* Natural Language Code = English */
        0b00001 0b001 0x656E
      }
      Element1[3]
      {
        /* Character Encoding = UTF-8 */
        0b00001 0b001 0x006A
      }
      Element2[3]
      {
        /* String Base = 0x0100 */
        0b00001 0b001 0x0100
      }
              }
  }
}

Attribute[14]
{
  AttributeID[3]
  {
    0b00001 0b001 0x0100
  }
  AttributeValue[11]
  {
    /* service name */
    0b00100 0b101 0x09
    "XYZ Mouse"
  }
}

Attribute[23]
{
  AttributeID[3]
  {
    0b00001 0b001 0x0101
  }
  AttributeValue[20]
  {
    /* service description */
    0b00100 0b101 0x12
    "Three button mouse"
  }
}

Attribute[16]
{
  AttributeID[3]
  {
    0b00001 0b001 0x0102
  }
  AttributeValue[13]
  {
    /* service provider */
    0b00100 0b101 0x0B
    "XYZ Company"
```

```
  }
}

Attribute[6]
{
  AttributeID[3]
  {
    0b00001 0b001 0x0200
  }
  AttributeValue[3]
  {
    /* HID Device Version = 1.0 */
    0b00001 0b001 0x0100
  }
}

Attribute[6]
{
  AttributeID[3]
  {
    0b00001 0b001 0x0201
  }
  AttributeValue[3]
  {
    /* HID Parser Version = 1.11 */
    0b00001 0b001 0x0111
  }
}

Attribute[5]
{
  AttributeID[3]
  {
    0b00001 0b001 0x0202
  }
  AttributeValue[2]
  {
    /* HID Subclass = Boot Mouse  */
    0b00001 0b000 0x80
  }
}

Attribute[5]
{
  AttributeID[3]
  {
    0b00001 0b001 0x0203
  }
  AttributeValue[2]
  {
    /* HID Country Code = USA */
    0b00001 0b000 33
  }
}

Attribute[5]
{
  AttributeID[3]
  {
    0b00001 0b001 0x0204
  }
  AttributeValue[2]
  {
    /* HID Virtual Connection = True */
    0b00001 0b000 0x01
  }
}

Attribute[5]
{
  AttributeID[3]
```

```
          {
            0b00001 0b001 0x0205
          }
          AttributeValue[2]
          {
            /* HID Reconnect Initiate = True */
            0b00001 0b000 0x01
          }
        }

        Attribute[61]
        {
          AttributeID[3]
          {
            /* HID Descriptor List */
            0b00001 0b001 0x0206
          }
          AttributeValue[58]
          {
            DataElementSequence[58]
            {
              0b00110 0b101 0x38
              /* HID Class Descriptor */
              DataElementSequence[56]
              {
                0b00110 0b101 0x36
                Element0[2]
                {
                  /* Class Descriptor Type = Report */
                  0b00001 0b000 0x21
                }
                Element1[52]
                {
                  /* Class Descriptor Data Array = Mouse Report Descriptor */
                  0b00100 0b101 0x32
0x05, 0x01, 0x09, 0x02, 0xa1, 0x01, 0x09, 0x01,
0xa1, 0x00, 0x05, 0x01, 0x09, 0x30, 0x09, 0x31,
0x15, 0x81, 0x25, 0x7f, 0x75, 0x08, 0x95, 0x02,
0x81, 0x06, 0xc0, 0x05, 0x09, 0x19, 0x01, 0x29,
0x03, 0x15, 0x00, 0x25, 0x01, 0x95, 0x03, 0x75,
0x01, 0x81, 0x02, 0x95, 0x01, 0x75, 0x05, 0x81,
0x03, 0xc0
                }
              }
            }
          }
        }

        Attribute[13]
        {
          AttributeID[3]
          {
            /* HID LANGID Base List */
            0b00001 0b001 0x0207
          }
          AttributeValue[10]
          {
            DataElementSequence[10]
            {
              0b00110 0b101 0x08
              /* HID LANGID Base */
              DataElementSequence[8]
              {
                0b00110 0b101 0x06
                Element0[3]
                {
                  /* Natural Language Code = English (United States) */
                  0b00001 0b001 0x0409
                }
                Element2[3]
                {
```

```
                /* String Base = 0x0100 */
                0b00001 0b001 0x0100
              }
            }
          }
        }
      }

      Attribute[5]
      {
        AttributeID[3]
        {
          0b00001 0b001 0x0208
        }
        AttributeValue[2]
        {
          /* HID SDP Disable = False */
          0b00001 0b000 0x00
        }
      }

      Attribute[5]
      {
        AttributeID[3]
        {
          0b00001 0b001 0x0209
        }
        AttributeValue[2]
        {
          /* HID Battery Power = True */
          0b00001 0b000 0x01
        }
      }

      Attribute[5]
      {
        AttributeID[3]
        {
          0b00001 0b001 0x020A
        }
        AttributeValue[2]
        {
          /* HID Remote Wakeup = True */
          0b00001 0b000 0x01
        }
      }
    }
  }
}
ContinuationState[1]
{
  /* no continuation state */
  0x00
}
```

7.15 Example String Attributes

Assume for this example that the device is an ATM with a HID interface that defines 5 strings in 3 languages: English, German, and Italian. The first two strings are the Service Name and the Provider Name. The next 3 strings are “HID strings”, referenced in the HID Report Descriptor. Each HID string uses a string index defined in the Report Descriptor as the offset from the language attribute ID "base" in the LanguageBaseAttributeIDList. This example device has 3 strings associated with dedicated buttons: “Cancel Transaction”, “Another Transaction”, and “Accept Transaction”.

The Attribute ID bases for the respective languages are defined as follows:

English Base Attribute ID = 0x0100 (Primary)
Italian Base Attribute ID = 0x0400
German Base Attribute ID= 0x0500

The format of the LanguageBaseAttributeIDList is defined in the Bluetooth Specification [5] (see Part E: Section 5.1.7).

Below is an excerpt of the Service Record for the device.

Attribute ID	**Attribute Value**	**Description**
LanguageBaseAttributeIDList 0x0006	0x35 0x1B	data element sequence, 27 bytes
Element 0	0x09 0x656E	uint16 “en” (English)
Element 1	0x09 0x006A	uint16 UTF-8 encoding
Element 2	0x09 0x0100	uint16 PrimaryLanguageBaseID
Element 0	0x09 0x6974	uint16 “it” (Italian)
Element 1	0x09 0x006A	uint16 UTF-8 encoding
Element 2	0x09 0x0400	uint16 PrimaryLanguageBaseID
Element 0	0x09 0x6465	uint16 “de” (German)
Element 1	0x09 0x006A	uint16 UTF-8 encoding
Element 2	0x09 0x0500	uint16 PrimaryLanguageBaseID
HIDLANGIDBaseList 0x0207	0x35 0x18	data element sequence, 24 bytes. HID LANGID to Bluetooth language mapping. See Section 7.11.7.
HID LANGID Base	0x35 0x06	data element sequence, 6 bytes
Element 0	0x09 0x409	Language = English (United States)
Element 1	0x09 0x0100	Bluetooth String Offset
HID LANGID Base	0x35 0x06	data element sequence, 6 bytes
Element 0	0x09 0x0410	Language = Italian (Standard)
Element 1	0x09 0x0400	Bluetooth String Offset
HID LANGID Base	0x35 0x06	data element sequence, 6 bytes
Element 0	0x09 0x0407	Language = German (Standard)
Element 1	0x09 0x0500	Bluetooth String Offset
English Strings, Base = 0x0100		
ServiceName 0x0100 + 0x0000	0x25 0x22	text string, 34 bytes
	“XYZ Automatic Transaction Machine”	
Provider Name 0x0100 + 0x0002	0x25 0x0B	text string, 11 bytes
	“XYZ Company”	
HID String (3) 0x0100 + 0x0003	0x25 0x12	text string, 18 bytes
	“Cancel Transaction”	
HID String (4) 0x0100 + 0x0004	0x25 0x13	text string, 19 bytes
	“Another Transaction”	
HID String (5) 0x0100 + 0x0005	0x25 0x12	text string, 18 bytes
	“Accept Transaction”	
Italian Strings, Base = 0x0400		
ServiceName 0x0400 + 0x0000	0x25 0x26	text string, 38 bytes
	“XYZ Macchina Automatica Di Transazione”	

Attribute ID	Attribute Value	Description
Provider Name 0x0400 + 0x0002	0x25 0x0B "XYZ Company"	text string, 11 bytes
HID String (3) 0x0400 + 0x0003	0x25 0x18 "Annullare La Transazione"	text string, 24 bytes
HID String (4) 0x0400 + 0x0004	0x25 0x15 "Un' Altra Transazione"	text string, 21 bytes
HID String (5) 0x0400 + 0x0005	0x25 0x18 "Accettare La Transazione"	text string, 24 bytes
German Strings, Base = 0x0500		
ServiceName 0x0500 + 0x0000	0x25 0x0C "XYZ Bankomat"	text string, 12 bytes
Provider Name 0x0500 + 0x0002	0x25 0x0B "XYZ Company"	text string, 11 bytes
HID String (3) 0x0500 + 0x0003	0x25 0x13 "Transaktion Beenden"	text string, 19 bytes
HID String (4) 0x0500 + 0x0004	0x25 0x12 "Andere Transaktion"	text string, 18 bytes
HID String (5) 0x0500 + 0x0005	0x25 0x14 "Transaktion Annehmen"	text string, 20 bytes

Table 23: ATM Service Record Excerpt for Multilingual Strings

7.16 Example Flow Spec Settings

7.16.1 Mouse

* Recommended until Flush function fixed in Bluetooth core spec to be ACL link specific

Table 24 is an example of the L2CAP Configuration Request parameters that would be specified for a mouse. This mouse sends 3-byte Input reports over its interrupt channel at a 100 Hz rate, generating 300 Byte/second bursts of real-time coordinates to the host on the Interrupt channel. No traffic is sent on the Interrupt channel from the host to the mouse. The Control channel is used for an occasional command to change modes in the mouse.

The Token Bucket size is set to 4 bytes (includes header). This value allows higher-level software to decide what to do with subsequent mouse data if a packet is delayed for more than the Latency period. For instance, since mice report relative coordinates, the higher-level software may sum the coordinate values of backed-up reports until it is able to post another report to the L2CAP layer.

Parameter	Control Channel Values		Interrupt Channel Values	
	Device to Host	Host to Device	Device to Host	Host to Device
Service Type	**Best Effort**	**Best Effort**	**Guaranteed**	**No Traffic**
Token Rate	800 Bytes/sec.	800 Bytes/sec.	300 Bytes/sec.	None specified
Token Bucket Size	8 Bytes	8 Bytes	4 Bytes	None needed
Peak Bandwidth	Unknown	Unknown	300 Bytes/sec.	Unknown

	Control Channel Values		Interrupt Channel Values	
Latency	Don't care	Don't care	10ms – 100 Hz polling rate	Don't care
Delay Variation	Don't care	Don't care	10ms	Don't care
OutFlushTO	Infinite retries	Infinite retries	Infinite retries*	Infinite retries
InMTU	48 Bytes	672 Bytes	48 Bytes	672 Bytes

** Recommended until Flush function fixed in Bluetooth core spec to be ACL link specific*

Table 24: HID Mouse Configuration Settings

7.16.2 Keyboard

* Recommended until Flush function fixed in Bluetooth core spec to be ACL link specific

Table 25 is an example of the L2CAP Configuration Request parameters that would be specified for a keyboard. A keyboard sends up to 900 Bytes/second of low latency keystroke information to the host on the Interrupt channel. The Interrupt channel is also used by the host to update the LEDs on the keyboard. The Control channel is used for an occasional command to change modes in the keyboard.

Parameter	**Control Channel Values**		**Interrupt Channel Values**	
	Device to Host	**Host to Device**	**Device to Host**	**Host to Device**
Service Type	**Best Effort**	**Best Effort**	**Guaranteed**	**Guaranteed**
Token Rate	800 Bytes/sec.	800 Bytes/sec.	900 Bytes/sec.	200 Bytes/sec.
Token Bucket Size	8 Bytes	8 Bytes	9 Bytes	2 Bytes
Peak Bandwidth	Unknown	Unknown	900 Bytes/sec.	200 Bytes/sec.
Latency	Don't care	Don't care	10 ms – 100 Hz polling rate	Don't care
Delay Variation	Don't care	Don't care	10 ms	Don't care
OutFlushTO	Infinite retries	Infinite retries	Infinite retries*	Infinite retries*
InMTU	48 Bytes	672 Bytes	48 Bytes	672 Bytes

** Recommended until Flush function fixed in Bluetooth core spec to be ACL link specific*

Table 25: HID Keyboard Configuration Settings

7.16.3 Force-Feedback Joystick

* Recommended until Flush function fixed in Bluetooth core spec to be ACL link specific

Table 26 is an example of the L2CAP Configuration Request parameters that would be specified for a force feedback joystick. A force feedback joystick sends up to 600 Bytes/second of low latency coordinates to the host on the Interrupt channel, and the host sends up to 1600 Bytes/second of low latency force feedback effect commands to the joystick. The Control channel is used for an occasional command to change modes and update parameters in the force feedback joystick.

Parameter	**Control Channel Values**		**Interrupt Channel Values**	
	Device to Host	**Host to Device**	**Device to Host**	**Host to Device**
Service Type	**Best Effort**	**Best Effort**	**Guaranteed**	**Guaranteed**
Token Rate	1500 Bytes/sec.	1500 Bytes/sec.	600 Bytes/sec.	1600 Bytes/sec.
Token Bucket Size	15 Bytes	15 Bytes	6 Bytes	16 Bytes
Peak Bandwidth	Unknown	Unknown	600 Bytes/sec.	1600 Bytes/sec.
Latency	Don’t care	Don’t care	10 ms	10 ms
Delay Variation	Don’t care	Don’t care	10 ms	10 ms
OutFlushTO	Infinite retries	Infinite retries	Infinite retries*	Infinite retries*
InMTU	48 Bytes	672 Bytes	48 Bytes	672 Bytes

** Recommended until Flush function fixed in Bluetooth core spec to be ACL link specific*

Table 26: HID Force Feedback Joystick Configuration Settings

8 L2CAP Compatibility Requirements

This section provides a list of supported L2CAP commands for host and device. For abbreviations, see Section 1.3.1.

Function	Command Code	Procedure	Support in Host/HID
Channel types		Connection-oriented channel	M
		Connectionless channel	X
Signalling commands	0x01	Command rejection	M
	0x02	Connection request	M
	0x03	Connection response	M
	0x04	Configure request	M
	0x05	Configure response	M
	0x06	Disconnection request	M
	0x07	Disconnection response	M
	0x08	Echo request	M
	0x09	Echo response	M
	0x0a	Information request	O
	0x0b	Information response	O
Configuration parameter options		Maximum transmission unit	M
		Flush timeout	M
		Quality of service interface	O

Table 27: L2CAP Compatibility

M = Mandatory

O = Optional

X = Connectionless channel not used with this profile, but may be used in conjunction with other applications/profiles.

8.1 Channel Types

Since a connectionless channel type is not used with this profile, broadcast packets are disallowed.

8.2 Notes on Configuration Parameters

The HID's L2CAP implementation shall support a minimum MTU (Maximum Transmission Unit) of 48 bytes, as required by the Bluetooth Specification [5]. Bluetooth has specified the "default" MTU size as 672 bytes; however, due to the extreme memory resource constraints faced by Human Interface Devices it is expected that most HIDs will not support an MTU size this large. Hosts, however, are recommended to support an MTU of at least 672 bytes.

8.3 Flush Timeout

It is recommended that the L2CAP Flush timeout parameter be set to 0xFFFF (infinite) until problems with this function in the L2CAP protocol are addressed in a future revision of the Bluetooth core specification.

8.4 Quality of Service

For high performance gaming and pointing devices intended for use in a multiple slave piconet, it is recommended to negotiate guaranteed service using the L2CA_ConfigReq command, with parameters as recommended in Section 7.16 to ensure the connection latency standards are met. Since full quality of service support in not available in Bluetooth specifications at the time of this writing, unless absolutely required to function, devices should not fail to complete the connection if the host refuses the QoS parameters.

9 Link Manager Compatibility Requirements

In order for hosts to be compatible with all Bluetooth HIDs, they shall implement the mandatory link manager commands specified in the table below. Likewise, in order to support any potential host, all Bluetooth HIDs shall implement the mandatory commands specified in the table. For abbreviations, see Section 1.3.1.

	Link Manager Procedure	**Support in Host**	**Support in Keyboards and Identification devices**	**Support in Other HIDs**
1	Authentication	C1	M	C1
2	Pairing/bonding	O	M	O
3	Change link key	O	O	O
4	Change the current link key	O	O	O
5	Encryption	O	M	O
6	Clock/slot offset request	O	O	O
7	Timing accuracy information request	O	O	O
8	LMP version request	M	M	M
9	Supported features request	M	M	M
10	Master/slave role switch	M	C2	C2
11	Name request	M	O	O
12	Detach request	M	M	M
13	Hold mode	O	O	O
14	Sniff mode	M	O	O
15	Park mode	O	O	O
16	Transmit power control	O	O	O
17	Channel quality driven packet type	O	O	O
18	Quality of service interface	O	O	O
19	SCO links	X	X	X
20	Control of multi-slot packets	O	O	O
21	Optional paging schemes	O	O	O
22	Link supervision timeout	M	M	M
23	Connection request and acknowledge	M	M	M

Table 28: Link Manager Compatibility

"M" for mandatory to support (used for capabilities that shall be used in the profile);
"O" for optional to support (used for capabilities that can be used in the profile);
"C" for conditional support (used for capabilities that shall be used in case a certain other capability is supported);
"X" for excluded (used for capabilities that may be supported by the unit but should never be used in the profile);
"N/A" for not applicable (in the given context it is impossible to use this capability).

C1: Mandatory to support if pairing is supported
C2: Mandatory to support if HIDReconnectInitiate declared to be true

10 Link Control Compatibility Requirements

This section provides a list of the Link Control layer procedures that are required for host and device, summarized in a table using the notation in Section 1.3.1. This includes support for various packet types, inquiry modes, and paging modes. For abbreviations, see Section 1.3.1.

Capability		**Support in Host**	**Support in HID**
Inquiry		M	O
Inquiry scan		X	M
Paging		M	O
Page scan	Type R0	C1	O
	Type R1	C1	O
	Type R2	C1	O
Packet types	ID packet	M	M
	NULL packet	M	M
	POLL packet	M	M
	FHS packet	M	M
	DM1 packet	M	M
	DH1 packet	M	O
	DM3 packet	O	O
	DH3 packet	O	O
	DM5 packet	O	O
	DH5 packet	O	O
	AUX packet	X	X
	HV1 packet	X	X
	HV2 packet	X	X
	HV3 packet	X	X
	DV packet	O	O
	Inter-piconet capabilities	O	O

Table 29: Link Control Compatibility

"M" for mandatory to support (used for capabilities that shall be used in the profile);
"O" for optional to support (used for capabilities that can be used in the profile);
"C" for conditional support (used for capabilities that shall be used in case a certain other capability is supported);
"X" for excluded (used for capabilities that may be supported by the unit but should never be used in the profile);
"N/A" for not applicable (in the given context it is impossible to use this capability).
C1: One of these three modes must be supported by host.

11 Service Discovery Compatibility Requirements

11.1 SDP Protocol Requirements

Bluetooth Human Interface Devices shall implement at least a minimal Bluetooth SDP server, with one exception.

Declaration of the LanguageBaseAttributeIDList attribute is required in all HID implementations. Note that this must match the HIDLANGIDBaseList described in Section 7.11.7.

11.2 SDP Service Record Requirements

See Section 7.10.

12 Summary of Supported Bluetooth Modes

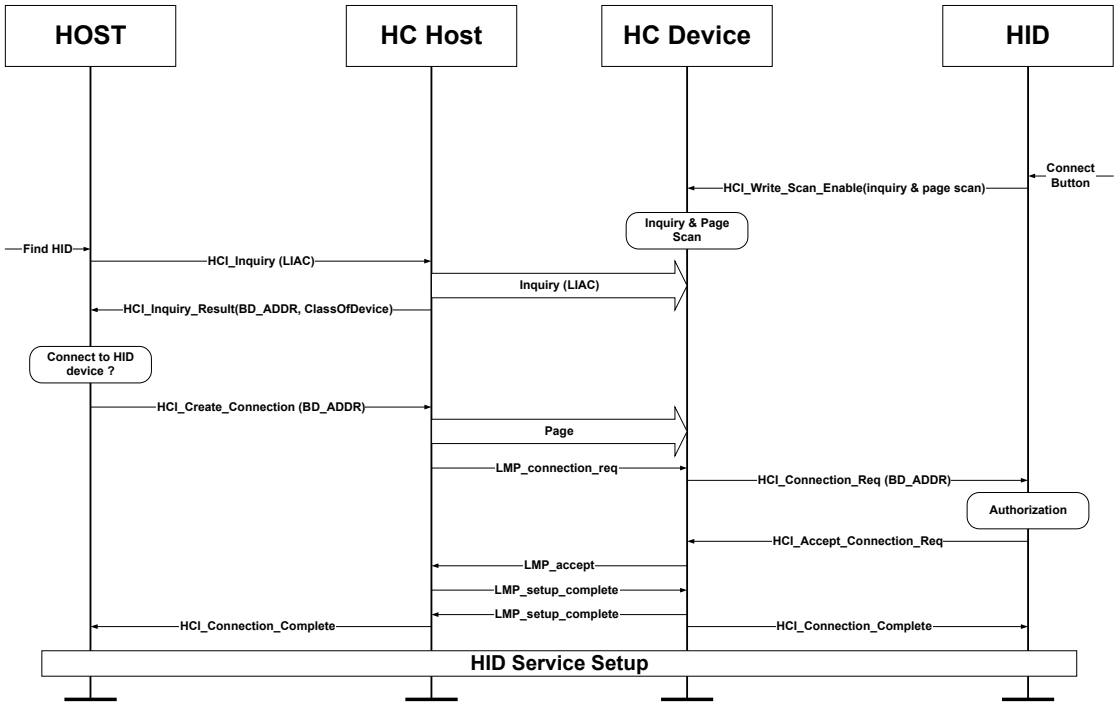
Bluetooth Feature Support	Bluetooth Keyboards and Identification devices	Other Bluetooth HID Devices
Authentication	Mandatory support and usage. One-way authentication mandatory, two-way optional.	Optional
Pairing and Bonding	Mandatory support and usage. One-way authentication mandatory, two-way optional.	Optional
Encryption support	Mandatory support and usage	Optional
Scatternet support (multiple masters)	Not recommended	Optional
Discoverable mode	Limited discoverable (recommended) or general discoverable	Limited discoverable (recommended) or general discoverable
Connectable mode	Connectable in all modes if and only if there is no existing active connection. May initiate connections if HIDReconnectInitiate bit set in the Service Discovery Record. Maximum one host connection recommended.	Maximum one host connection recommended. Recommended to remain connectable if slave only. May initiate connections if HIDReconnectInitiate bit set in the Service Discovery Record.
Bluetooth passkey entry method	Variable passkey.Host has fixed passkey.	Variable or fixed
Low Power Modes SNIFF, HOLD, PARK	SNIFF mandatory for Hosts, others optional for HIDs and Hosts	SNIFF mandatory for Hosts, others optional for HIDs and Hosts

Table 30: Summary of Supported Bluetooth Modes

13 Appendix A: Example Signaling Flows

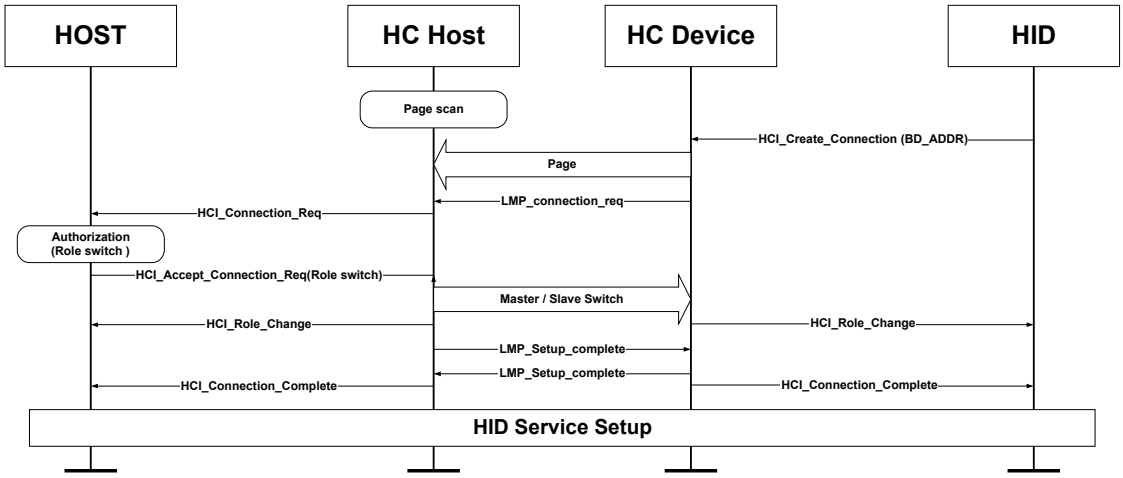
13.1 Setting Up a "Virtual Cable"

The user presses the connect button on the device to put it into discoverable and connectable mode. Any host may now discover the HID and get access to its SDP server and HID services.



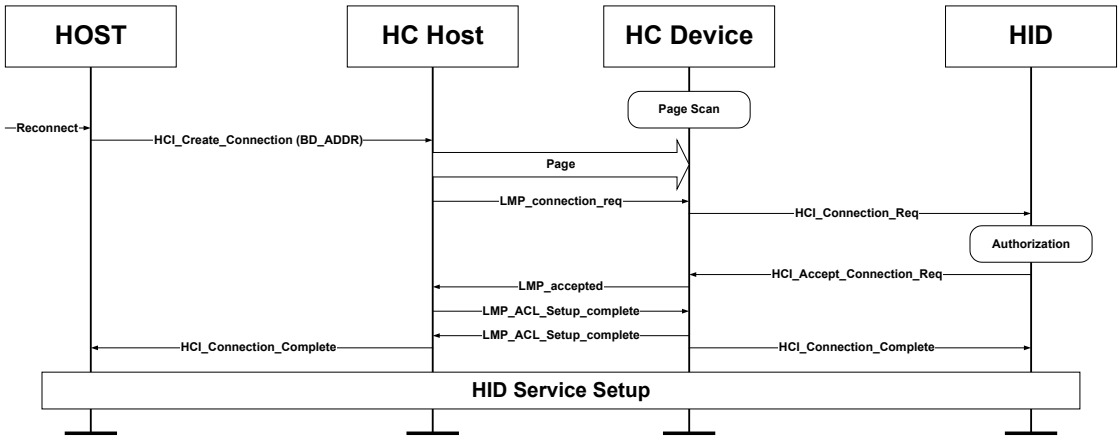
13.2 Automatic Reconnection (HID-Initiated)

After link loss, the device that has its HIDReconnectInitiate bit set to True will try to reconnect to the host it is “virtually cabled” with. The device will page the host and create an ACL connection. The host will accept a connection request from a virtually cabled HID and ask for a role switch.



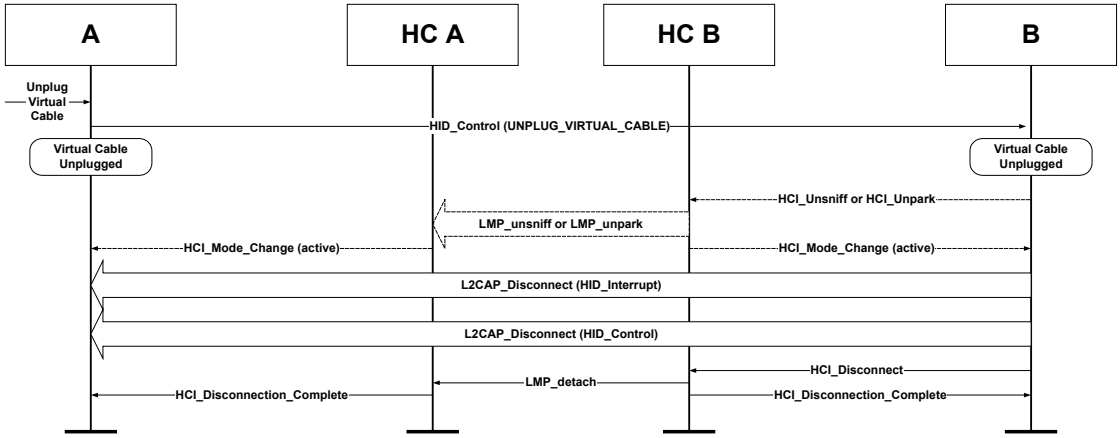
13.3 Automatic Reconnection (Host Initiated)

After link loss, the host can reconnect to a “virtually cabled” device by paging it, if the device has its attribute HIDNormallyConnecable set to True. The HID will accept a connection request only from its unique “virtually cabled” host.



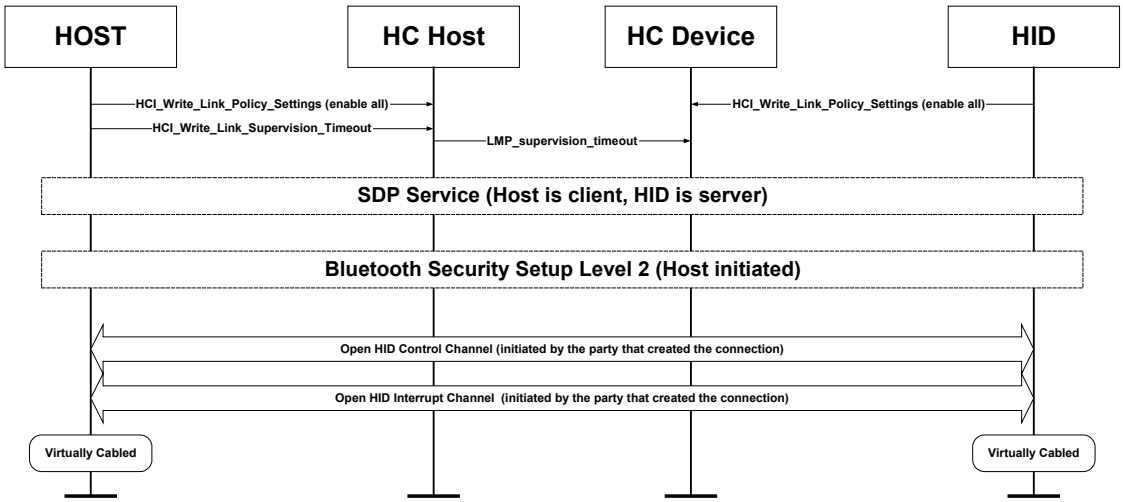
13.4 HID Disconnect (HID or Host Initiated)

The HID channels can be disconnected and the physical connection terminated by pressing (for example) the HID button on the device and disconnecting from the host. The initiator first disconnects the HID interrupt channel, then the HID control channel.



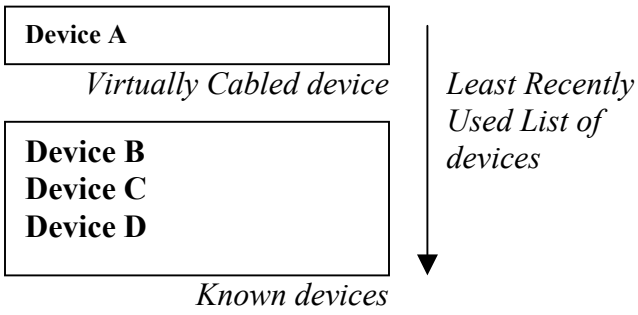
13.5 HID Service Setup

The ACL link between host and device has to be configured after its establishment. The host may open an unsecured L2CAP channel for retrieving information from the SDP server. The host may require a security setup (authentication, encryption) in the case of a keyboard.

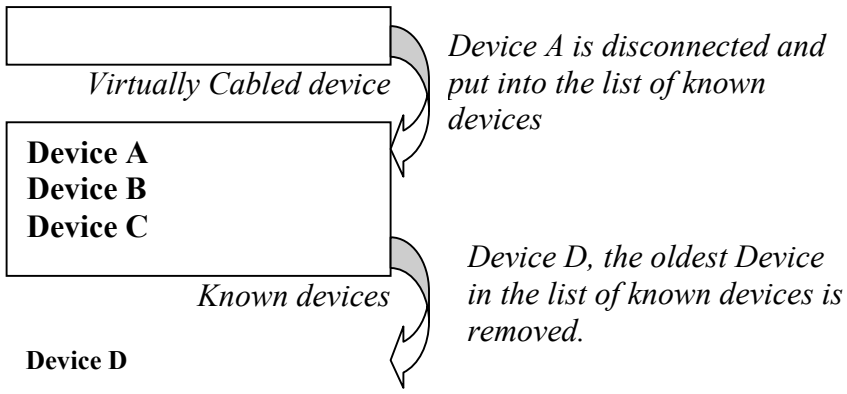


14 Appendix B: Persistent Storage of Known Devices

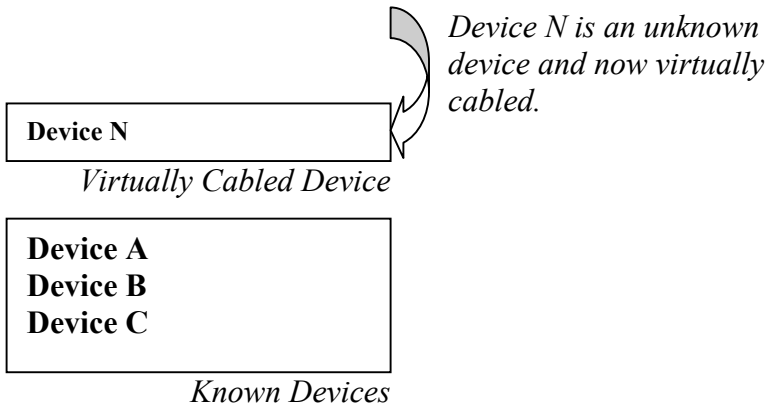
a. Persistent Storage of Virtually Cabled and Known Devices.



b. Disconnecting a virtual cable.



c. Create a virtual cable to an unknown device (may require security setup).



d. Re-establishing a virtual cable to a known device.